ETSI TS 103 666-1 V17.3.0 (2023-06)

Smart Secure Platform (SSP);
Part 1: General characteristics
(Release 17)

Reference
RTS/SET-T103666-1vh30

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 Web server (https://ipr.etsi.org/).

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™** and **LTE™** 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 1 of a multi-part deliverable covering Smart Secure Platform (SSP), as identified below:

Part 1: "General characteristics";

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

Part 3: "Embedded SSP (eSSP) Type 1 characteristics";

Part 4: "Embedded SSP (eSSP) Type 2 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 is part of a series of documents that specify the technical solution for the Smart Secure Platform (SSP), according to the requirements listed in ETSI TS 103 465 [i.2].

The present document contains generic technical solutions for different aspects of SSP functionality. It does not specify any specific type of SSP.

The types of SSP are referred to as classes. The class specifications (for example the integrated SSP technical specification in ETSI TS 103 666-2 [8], the embedded SSP Type 1 technical specification in ETSI TS 103 666-3 [44] or the embedded SSP Type 2 technical specification in ETSI TS 103 666-4 [45]) refer to the present document for common SSP functionality.

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 at https://docbox.etsi.org/Reference/.

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] Void.

[3] ISO/IEC 7816-3: "Identification cards -- Integrated circuit cards -- Part 3: Cards with contacts -- Electrical interface and transmission protocols".

[4] ISO/IEC 7816-4: "Identification cards -- Integrated circuit cards -- Part 4: Organization, security and commands for interchange".

[5] ETSI TS 102 613: "Smart Cards; UICC - Contactless Front-end (CLF) Interface; Physical and data link layer characteristics".

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

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

[8] ETSI TS 103 666-2: "Smart Secure Platform (SSP); Part 2: Integrated SSP (iSSP) characteristics".

[9] ORACLE: "Application Programming Interface, Java Card™ Platform, Classic Edition 3.0.5".

[10] ORACLE: "Runtime Environment Specification, Java Card™ Platform, Classic Edition 3.0.5".

[11] ORACLE: "Virtual Machine Specification Java Card™ Platform, Classic Edition 3.0.5".

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

[13] GlobalPlatform: "Virtual Primary Platform - Network Protocol" Version 2.0.

[14] ETSI TS 102 622: "Smart Cards; UICC - Contactless Front-end (CLF) Interface; Host Controller Interface (HCI)".

[15] Recommendation ITU-T X.680 (08/2015): "Information technology - Abstract Syntax Notation One (ASN.1) & ASN.1 encoding rules".

[16] Recommendation ITU-T X.690 (08/2015): "Information Technology - Abstract Syntax Notation One (ASN.1) & ASN.1 encoding rules".

[17] ETSI TS 102 671: "Smart Cards; Machine to Machine UICC; Physical and logical characteristics".

[18] IETF RFC 7230: "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing".

[19] IETF RFC 2818: "HTTP Over TLS".

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

[21] IETF RFC 793: "Transmission Control Protocol".

[22] GlobalPlatform: "Card Specification", Version 2.3.1.

[23] IETF RFC 768 (August 1980): "User Datagram Protocol".

[24] IETF RFC 7252: "The Constrained Application Protocol (CoAP)".

[25] GlobalPlatform: "UICC Configuration", Version 2.0.

[26] IETF RFC 792: "Internet Control Message Protocol".

[27] IETF RFC 6895: "Domain Name System (DNS) IANA Considerations".

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

[29] IETF RFC 8141: "Uniform Resource Names (URNs)".

[30] IETF RFC 8615: "Well-Known Uniform Resource Identifiers (URIs)".

[31] IETF RFC 3629: "UTF-8, a transformation format of ISO 10646".

[32] ETSI TS 103 713: "Smart Secure Platform (SSP); SPI interface".

[33] ETSI TS 102 705: "Smart Cards; UICC Application Programming Interface for Java Card™ for Contactless Applications".

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

[35] ANSI X9.63:2011: "Public Key Cryptography for the Financial Services Industry - Key Agreement and Key Transport Using Elliptic Curve Cryptography".

[36] BSI Technical Guideline TR-03111: "Elliptic Curve Cryptography", Version 2.0.

[37] FIPS PUB 180-4:2015: "Secure Hash Standard (SHS)".

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

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

[40] NIST SP 800-56A: "Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography" (Revision 3), April 2018.

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

[42] IETF RFC 6960: "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP".

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

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

[45] ETSI TS 103 666-4: "Smart Secure Platform (SSP); Part 4: Embedded SSP (eSSP) Type 2 characteristics (Release 17)".

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] ETSI TR 102 216: "Smart Cards; Vocabulary for Smart Card Platform specifications".

[i.2] ETSI TS 103 465: "Smart Cards; Smart Secure Platform (SSP); Requirements Specification".

3 Definition of terms, symbols, abbreviations and coding conventions

3.1 Terms

For the purposes of the present document, the terms given in ETSI TR 102 216 [i.1] and the following apply:

access control: metadata defining access rights of an accessor or a group of accessors

NOTE: It is an element of the access control list.

access control list: list of access controls attached to the resource

accessor: application which is acting on behalf of an entity, e.g. user or modem

NOTE: The accessor claims an identity when accessing a resource.

accessor authentication: procedure for authentication of an accessor against its credential

accessor credential: means to prove the identity of the accessor, e.g. PIN, fingerprint/minutia, token, signature, etc.

group of accessors: set of accessors

NOTE: A group may be empty.

MBM host domain: SCL host domain residing inside the modem, equivalent to "MBM Host Domain" in GlobalPlatform VPP - Network Protocol [13]

resource: service or information on which access is controlled

SCL gate: equivalent to a "Gate" in GlobalPlatform VPP - Network Protocol [13]

SCL host: equivalent to a "Host" in GlobalPlatform VPP - Network Protocol [13]

SCL host domain: equivalent to a "Host Domain" in GlobalPlatform VPP - Network Protocol [13]

SCL packet: equivalent to a "VNP Packet" in GlobalPlatform VPP - Network Protocol [13]

SCL pipe: equivalent to a "Pipe" in GlobalPlatform VPP - Network Protocol [13]

SCL pipe session: equivalent to a "Pipe Session" in GlobalPlatform VPP - Network Protocol [13]

SCL router: equivalent to the "Router" in GlobalPlatform VPP - Network Protocol [13], clause 3.4

secure element: tamper-resistant dedicated platform, consisting of hardware and software, capable of securely hosting applications and their confidential and cryptographic data and providing a secure application execution environment

SSP application: application running on the top of an SSP OS or an execution environment

SSP class: configuration of the SSP platform in accordance with a business requirement

SSP host: SCL host residing in the SSP host domain

SSP host domain: SCL host domain residing inside the SSP, equivalent to "TRE Host Domain" in GlobalPlatform VPP - Network Protocol [13]

SSP interface session: link between the SSP and an end-point in the terminal, starting when the data link layer is initialized and ending with a subsequent reset of the data link layer or deactivation of the physical interface

SSP OS: operating system compliant with the SSP specifications

System on Chip (SoC): integrated circuit that contains all the required circuitry and components of an electronic system on a single chip

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AAA	Accessor Authentication Application
AAS	Accessor Authentication Service
ACL	Access Control List
AID	Application IDentifier
APDU	Application Protocol Data Unit
ASN.1	Abstract Syntax Notation One
ATR	Answer To Reset
C-APDU	Command - APDU
CA	Certificate Authority
CAT	Card Application Toolkit
CB	Chaining Bit
CI	Certificate Issuer
CLA	CLAss
CLF	ContactLess Frontend
CLK	CLocK
CLT	ContactLess Tunnelling
CPU	Central Processing Unit
CSS	Cascading Style Sheets
DER	Distinguished Encoding Rule
DF	Dedicated File
DNS	Domain Name System
ECC	Elliptic Curve Cryptography
ECDHE	Elliptic Curve Diffie Hellman Ephemeral
EEPROM	Electrically Erasable Programmable Read Only Memory
FCI	File Control Information

FCP	File Control Parameters
FFS	For Further Study
FMD	File Management Data
FQDN	Fully Qualified Domain Name
FS	File System
GCM	Galois/Counter Mode
HCI	Host Controller Interface
HCP	Host Controller Protocol
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP over SSL
I2C	Inter-Integrated Circuit
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
INS	INStruction
IP	Internet Protocol
ISO	International Organization for Standardization
KDF	Key Derivation Function
MBM	Mobile Broadband Modem
MTU	Maximum Transfer Unit
NAA	Network Access Application
NFC	Near Field Communication
NID	Namespace IDentifier
NSS	Namespace Specific String
NVM	Non-Volatile Memory
OCSP	Online Certificate Status Protocol
OID	Object IDentifier
P1	Parameter 1
P2	Parameter 2
PIN	Personal Identification Number
PL	Padding Length
PPS	Protocol and Parameter Selection
RAM	Random Access Memory
RFU	Reserved for Future Use
RNG	Random Number Generator
RO	Read-Only
ROM	Read-Only Memory
SCL	SSP Common Layer
SHDLC	Simplified High Level Data Link Control
SI	SharedInfo
SoC	System on Chip
SPI	Serial Peripheral Interface
SSP	Smart Secure Platform
SWP	Single Wire Protocol
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TLV	Tag Length Value
TRE	Tamper Resistant Element
UDP	User Datagram Protocol
UML	Unified Modelling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
USB	Universal Serial Bus
UTF	Universal character set Transformation Format
UUID	Universally Unique IDentifier
VNP	VPP Network Protocol
VPP	Virtual Primary Platform
XOR	eXclusive OR

3.4 Coding conventions

All bytes specified as RFU shall be set to '00' and all bits specified as RFU shall be set to 0, and their values shall be ignored.

4 Introduction

4.1 Background

The present document defines a secure element which is modular and allows for different combinations of form factors, physical and electrical interfaces, transport layers, file system and security requirements based on the targeted use-case.

This secure element is called Smart Secure Platform (SSP). This platform is decoupled from the form factor and is defined to be flexible to use multiple physical interfaces and transport protocols.

4.2 Document layout

The present document specifies:

- the definition and overview of an SSP;
- the possible physical interface(s) between the terminal and the SSP;
- the possible transport protocol(s) used between the terminal and the SSP;
- the core features and characteristics of an SSP, including security requirements;
- various SSP classes supported, i.e. a standard combination of the options available for the protocol stack and file system.

4.3 References to UICC

The present document contains several references to UICC specifications such as ETSI TS 102 221 [1] and ETSI TS 102 223 [6]. In these cases, the word "UICC" shall be replaced with "SSP" as needed.

Similarly, the word "card" used in ISO/IEC 7816-3 [3] and ISO/IEC 7816-4 [4] shall be replaced with "SSP" as needed.

4.4 ASN.1 syntax

4.4.1 Introduction

The description of some data objects in the present document is based on ASN.1 specified in Recommendation ITU-T X.680 [15] and encoded in TLV structures using Distinguished Encoding Rule (DER) encoding as specified in Recommendation ITU-T X.690 [16]. This provides a flexible description of those data objects. The complete ASN.1 code is divided into a number of ASN.1 sections in the specifications. In order to facilitate the extraction of the complete ASN.1 code from the specification, each ASN.1 section begins with a text paragraph consisting entirely of an ASN.1 start tag, which consists of a double hyphen followed by a single space and the text string "ASN1START" (in all upper case letters). Each ASN.1 section ends with a text paragraph consisting entirely of an ASN.1 stop tag, which consists of a double hyphen followed by a single space and the text "ASN1STOP" (in all upper case letters).

The complete ASN.1 code may be extracted by copying all the text paragraphs between an ASN.1 start tag and the following ASN.1 stop tag in the order they appear, throughout the present document.

4.4.2 Start of ASN.1

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

```
-- ASN1START

SSPDefinitions { itu-t (0) identified-organization (4) etsi (0) smart-secure-platform (3666) part1
(1) }
DEFINITIONS
AUTOMATIC TAGS
EXTENSIBILITY IMPLIED ::=
BEGIN

EXPORTS ALL;

/* Imports */
IMPORTS
    ESSPCapabilities  -- SSPSpecificCapabilities for eSSP
        FROM ESSPDefinitions-- defined in ETSI TS 103 666-3 [44]
    Certificate,  -- RFC5280 Certificate X.509v3
    id-pkix,
    Extensions,  -- RFC5280 X.509v3 extension
    Extension,
    AlgorithmIdentifier,
    Attribute,
    AttributeType,
    AttributeValue,
    AttributeTypeAndValue,
    SubjectPublicKeyInfo,
    UniqueIdentifier,
    Validity,
    Version
        FROM PKIX1Explicit88
    ECDSA-Sig-Value
        FROM PKIX1Algorithms88;

/* Basic types */
maxUInt32 INTEGER ::= 4294967295
UInt32 ::= INTEGER (0..maxUInt32)

/* Common types */
UUID ::= OCTET STRING (SIZE(16))
URI ::= OCTET STRING
Certificates ::= SET OF Certificate
VersionType ::= OCTET STRING(SIZE(2)) -- major/minor version, coded as binary value on byte 1 and 2,
e.g. '0F 00' for v15.0.

METADATUM ::= TYPE-IDENTIFIER

-- ASN1STOP
```

5 SSP architecture

5.1 Overview

The SSP is a secure element platform intended for use in a number of use cases which may have very different requirements. For that reason, the SSP is designed to be a modular platform offering a core set of features as well as a number of options that need to be selected at the time of implementation based on the intended use case. The goal is to enable the best fit for the targeted use case.

SSP classes are defined in order to address these different use cases and in order to limit the possible configurations. An SSP class defines a configuration of the SSP platform, such as:

- the physical interfaces, power states and the form factor;
- communication protocol, e.g. the SSP Common Layer (SCL);
- support for a virtualization layer, called the Primary Platform Interface.

The list of the SSP classes is provided in clause 11. Each class has its own technical specification in this multi-part deliverable.

5.2 SSP software architecture

SSP Applications are programs running in the SSP. Examples for SSP Applications are:

- applications based on a file system and commands, e.g. as defined in ISO/IEC 7816-4 [4];
- applications running in standardized runtime environment, e.g. as defined in ETSI TS 102 241 [12];
- applications running as native applications in a proprietary runtime environment.

SSP implementations typically include an operating system. On top of the operating system, runtime environments and frameworks may provide a standardized interface to SSP Applications.

The operating system may directly access the SSP hardware platform. To improve portability of the operating system, an SSP class may specify a virtualization interface, that splits the operating system functionality into a Primary Platform and a Secondary Platform. The Primary Platform contains the technology-dependent parts of the operating system and the Secondary Platform contains the technology-independent parts. The Secondary Platform and the SSP Applications are combined into a Secondary Platform Bundle.

Figure 5.1 compares the two typical architectures.

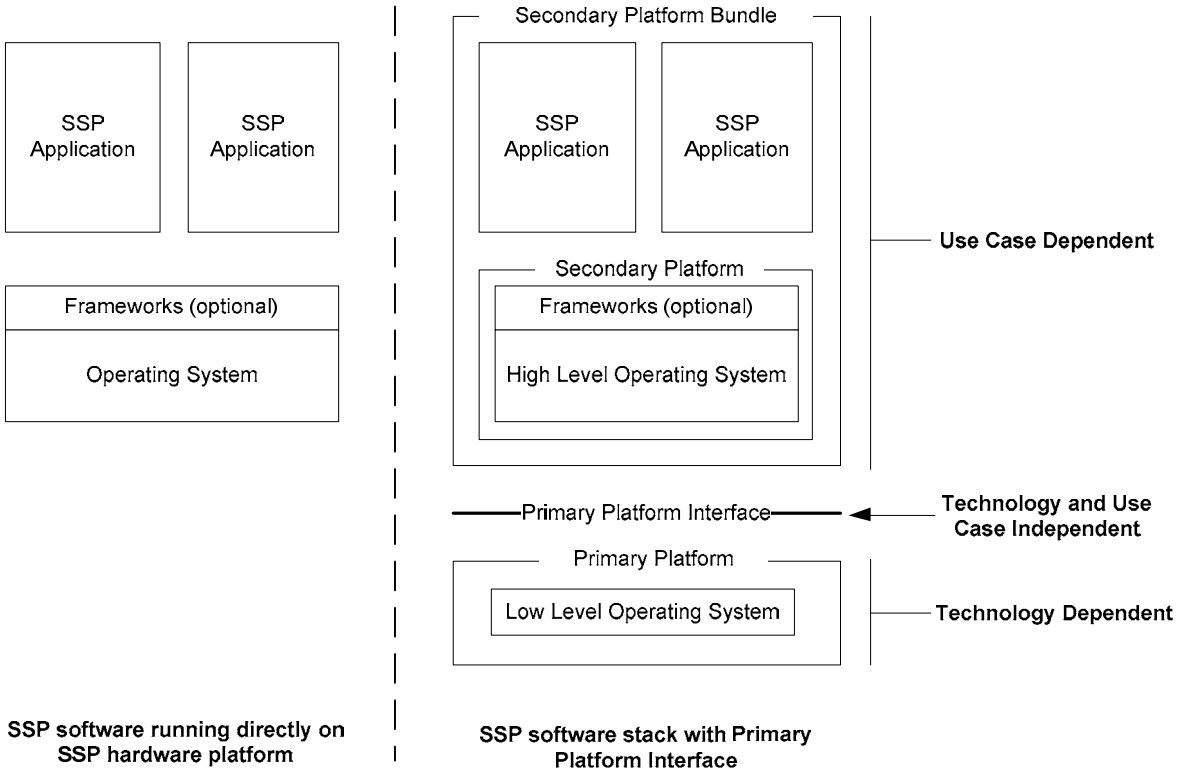

Figure 5.1: SSP software architectures

5.3 SSP hardware architecture

A typical SSP consists of a processing unit, security components, I/O ports and memory (volatile and non-volatile). Figure 5.2 shows a typical SSP.

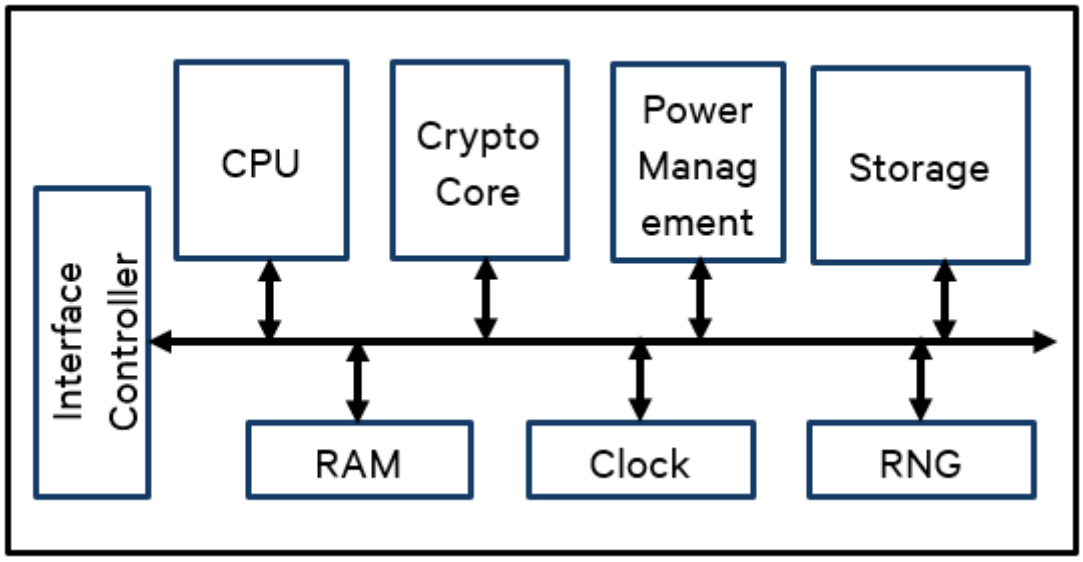

Figure 5.2: Typical SSP hardware architecture

5.4 Protocol stacks

The physical interface(s) between the SSP and the device might be selected from a range of options. The SSP may have multiple physical interfaces.

The data link layer used over the physical interface might also be selected from a range of options.

The SSP should provide means for controlling (e.g. activating, deactivating) the data link and physical layers.

The SSP should provide means for getting notifications from the data link layer on changes to itself or the physical layer (e.g. activation/deactivation of the interface by the terminal).

If indicated by the SSP class, the SSP shall support the SSP Common Layer (SCL) implementation comprised of optional network, transport and session layers, as described in clause 8.

If SSP Common Layer (SCL) is not supported, the SSP may support the UICC architecture as defined in ETSI TS 102 221 [1] and ETSI TS 102 622 [14].

An SSP implemented according to one of the existing form factors in ETSI TS 102 221 [1] and in ETSI TS 102 671 [17] shall support the ISO/IEC 7816-3 [3] interface and the transport of APDUs.

In addition, a mandatory core set of security features is provided, together with a number of optional security features which can be selected depending on the application.

Figure 5.3 shows a generic SSP protocol stack configuration.

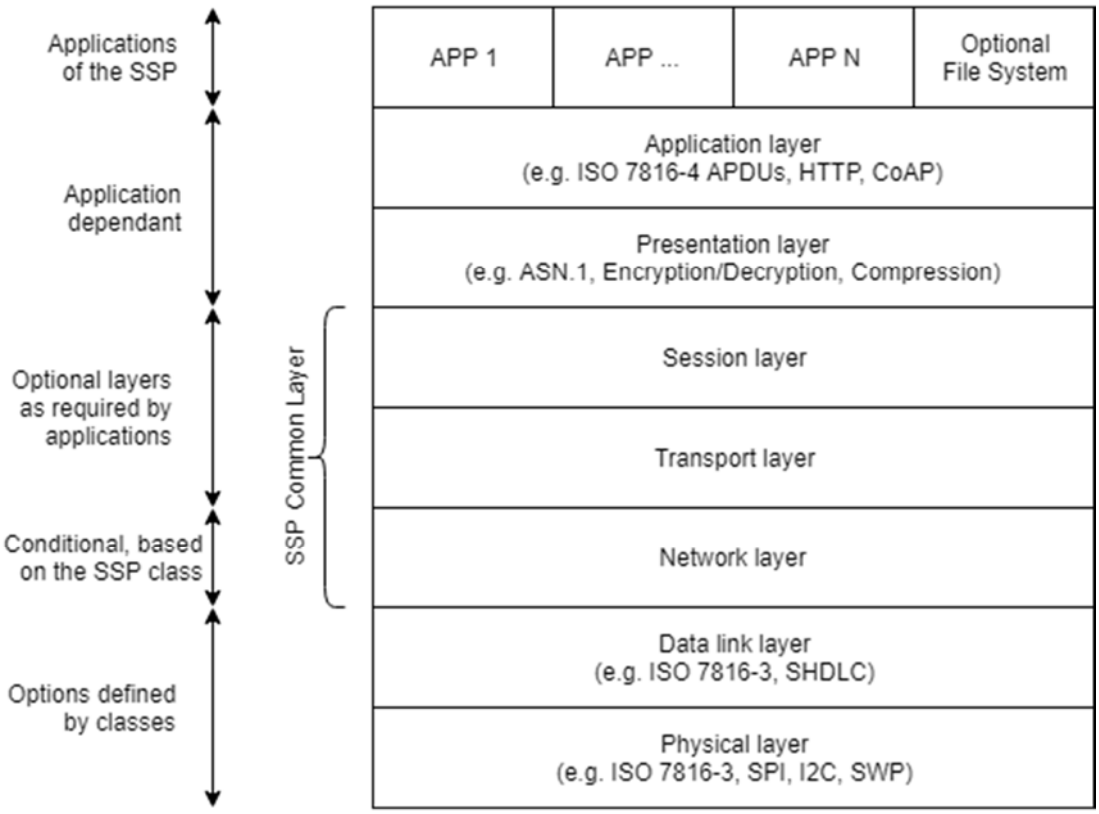

Figure 5.3: SSP protocol stack

5.5 Execution frameworks

The optional or mandatory support of specific execution frameworks is defined for each specific SSP class.

The SSP may support an execution framework as defined for the UICC according to ETSI TS 102 241 [12] based on the Java Card™ Platform [9], [10] and [11].

The definition of additional Execution Frameworks for the SSP is FFS.

5.6 Bundle Interoperability

SSP Secondary Platform Bundles are generated for a specific Primary Platform and are in general not binary interoperable with other Primary Platforms. If an SPB created for one Primary Platform is intended to be loaded onto an SSP with another Primary Platform which is different in hardware or software, even if only in details, a dedicated porting effort is required.

Therefore, the requirement REQ-15-SSP-8.9.1.2.2-34 in ETSI TS 103 465 [i.2] that SPBs shall be interoperable so that any SPB can be downloaded and installed on any SSP is not met.

NOTE: This may be mitigated by creating and keeping Primary Platform-specific versions of a given SPB in combination with appropriate version management capabilities at the SPB Manager.

6 SSP characteristics

6.1 Form factors

The overall definition of the SSP is independent of the form factor, unless specified differently for a particular SSP class.

6.2 Power

6.2.1 Power mode

The following power modes are defined:

- OPERATIONAL: when the SSP performs an internal process or processes incoming data from any of its interfaces. This mode also includes the transmission of data from and to the terminal.
- SUSPENDED: the SSP does not consume any power, with the ability to resume the logical state at a later time (as described in clause 6.9).
- IDLE: the SSP is in idle mode at any other time.

The power mode transition time is the maximum duration it takes the SSP to transition from one specific power mode, once SSP decided to, to another specific power mode.

6.2.2 Power sources

6.2.2.1 Types of power sources

The following power source types are defined for an SSP:

- Interface: power to the SSP is provided by a communication interface according to its definition (e.g. ISO/IEC 7816-3 [3], USB).
- Independent: power source which is not dependent on the power provided by any communication interface (e.g. dedicated power line).

The combined power sources shall provide sufficient power to operate the SSP in accordance to its power mode.

6.2.2.2 Power source of type Interface

Power provided by a communication interface is managed by the interface itself.

6.2.2.3 Power source of type Independent

The following voltage classes for a power source of type Independent are defined as follows, unless specified differently for an SSP class:

- Class A: operational voltage class range is defined in table 5.1 in ETSI TS 102 221 [1].
- Class B: operational voltage class range is defined in table 5.5 in ETSI TS 102 221 [1].
- Class C: operational voltage class range is defined in table 5.9 in ETSI TS 102 221 [1].
- Class D: operational voltage class range is defined in table 5.13 in ETSI TS 102 221 [1].
- Class P: operational voltage class range is proprietary and not defined in the present document.

Supply voltage switching is outside the scope for power sources of type Independent.

Communication interfaces shall operate in relation to the voltage provided by the power source unless specified differently by the communication interface (e.g. ETSI TS 102 613 [5] operates at a fixed voltage level regardless of the supply voltage).

For reliable operation, the power source should meet the following characteristics:

- When the power source is activated, the supply voltage should rise monotonically until reaching the operational voltage range.
- The terminal should activate any communication interfaces only after the supply voltage has reached a stable level within the operational voltage range.

NOTE: This is to avoid harmful cross current.

- When the power source is deactivated, the supply voltage should fall monotonically until reaching 0 V ± 0,4 V referenced to ground.

Before activating the power source again, the supply voltage should remain at 0 V ± 0,4 V referenced to ground for at least 10 ms.

6.2.3 Power consumption

The maximum power consumption is defined as the maximum amount of power used by the SSP when operating in OPERATIONAL power mode.

The overall power provided by the terminal to the SSP shall meet the power consumption of all active interfaces of the SSP and the internal power consumption of the SSP.

The maximum power consumption may be negotiated during the capability exchange procedure, as defined in clause 6.4.2.

6.3 Clock

The SSP shall have its own clock for the processing of all the commands, for the execution of its applications and for the access to its volatile and non-volatile memory, unless specified otherwise by the SSP class. If a physical interface provides a clock (for example, the CLK like in the ISO/IEC 7816-3 [3] interface), this is independent from the internal clock of the SSP and shall not be used for internal processing, but only for the exchange of data over that interface.

The frequency of the clock of the SSP is implementation specific and outside the scope of the present document. The SSP shall make sure that its clock frequency does not cause power consumption in excess to what is negotiated with the terminal.

The SSP shall provide SSP applications with an interface to a time keeping mechanism, which measures elapsed time. The value obtained over this interface shall be based on the clock defined in this clause. Furthermore, this value shall be monotonic and increasing.

6.4 SSP initialization

6.4.1 SSP interface session

The SSP interface session begins when the physical interface and the data link layer are initialized, and the SSP is in a state where it can receive data from an end-point in the terminal or send data to an end-point in the terminal. The SSP may have multiple independent SSP Interface Sessions at the same time, one or more for each physical interface.

The SSP interface session terminates when the data link layer is reset or when the physical interface is powered down, unless the SSP is suspended.

6.4.2 Capability exchange

6.4.2.1 Overall description

The capability exchange procedure is used to inform the SSP of the capabilities of the terminal and to retrieve the capabilities of the SSP.

6.4.2.2 SSP not supporting SCL

For an SSP not supporting the SCL, after the SSP Interface Session is started, the terminal should perform the capability exchange procedure unless specified differently for the physical interface due to timing reasons (e.g. ETSI TS 102 613 [5], clause 12). The capability exchange procedure should also be performed when some of the capabilities have already been exchanged using the specific physical interface or transport interface (e.g. the ATR content).

6.4.2.3 SSP supporting SCL

For an SSP supporting the SCL, the following statements shall apply.

If the UICC APDU gate described in clause 10.2.8.2 is supported, then the capability exchange procedure shall be performed with the EXCHANGE CAPABILITIES command described in clause 10.2.3.2.

In all other cases, the procedure should be performed when a new SCL host is registered on the SCL network controller host.

The procedure is performed by reading the parameter CAPABILITY_EXCHANGE as defined in clause 8.4.5.1.3.

The capability exchange procedure is completed after the SCL host outside the SSP has read the CAPABILITY_EXCHANGE entry in the identity gate registry of the SCL host in the SSP and vice-versa.

6.4.2.4 Capabilities of the terminal

The data field sent by the terminal to the SSP contains the following data structure.

```
-- ASN1START

PhysicalInterfaceType ::= ENUMERATED
{
    ePhysicalInterfaceType-ISO (0),  -- ISO/IEC 7816 interface
    ePhysicalInterfaceType-SWP (1),  -- SWP interface
    ePhysicalInterfaceType-SPI (2),  -- SPI interface
    ePhysicalInterfaceType-I2C (3),  -- I2C interface
    ePhysicalInterfaceType-USB (4)  -- USB interface
}

PhysicalInterface ::= SEQUENCE
{
    aInterfaceType PhysicalInterfaceType,  -- physical interface type
    aMaximumInterfacePowerSupply INTEGER (0..1000) OPTIONAL  -- max power supply of the interface
}

/* Capabilities of the terminal */
TerminalCapability ::= SEQUENCE
{
    aTerminalRelease [0] VersionType,
    aTerminalVendorName [1] UTF8String (SIZE(1..20)) OPTIONAL,
    aPhysicalInterfaces [2] SEQUENCE OF PhysicalInterface OPTIONAL,  -- list of interfaces
    aExternalPowerSupply [3] INTEGER (0..1000) DEFAULT 0,
    aToolkitTerminalProfile [4] OCTET STRING OPTIONAL
}

-- ASN1STOP
```

aTerminalRelease: it indicates the release of the present document that is implemented by the terminal. The major version shall have a value that is greater or equal to '0F', which corresponds to Release 15, as the first release of the SSP.

aTerminalVendorName: it indicates the terminal vendor's name encoded in UTF-8 format, as described in IETF RFC 3629 [31].

aPhysicalInterfaces: it indicates the list of interfaces supported by the terminal:

- **aInterfaceType:** it indicates the type of a physical interface.
- **aMaximumInterfacePowerSupply:** it indicates the maximum current in mA that the terminal can provide over each interface type. If the physical interface does not provide power, value '0' is used.

aExternalPowerSupply: it indicates the maximum current provided by the terminal using the external power supply. The value indicates the current in mA. The terminal shall use the same value on all the interfaces where the Capability Exchange procedure is performed. Value '0' is used when the external power supply is not present.

aToolkitTerminalProfile: it indicates the terminal profile used for the Card Application Toolkit. It is coded as defined in ETSI TS 102 223 [6], clause 5.2. If the TLV is absent, it means that the terminal does not support the Card Application Toolkit.

6.4.2.5 Capabilities of the SSP

The data field sent by the SSP to the terminal contains the following data structure.

```
-- ASN1START

/* Class of the SSP, as defined in clause 11 */
SSPClass ::= ENUMERATED
{
    eSSPClass-Integrated (0),  -- iSSP
    eSSPClass-Embedded-Type1 (1),  -- eSSP Type 1
    eSSPClass-Embedded-Type2 (2),  -- eSSP Type 2
    eSSPClass-Removable (3)  -- rSSP
}

/* SSP class specific capabilities */
SSPSpecificCapabilities ::= CHOICE
{
    aNone [0] NULL,
    aESSPCapabilities [1] ESSPCapabilities   -- eSSP capabilities
}

/* Capabilities of the SSP */
SSPCapability ::= SEQUENCE
{
    aSspRelease [0] VersionType,
    aSspVendorName [1] UTF8String (SIZE(1..20)) OPTIONAL,
    aSspClass [2] SSPClass,
    aClassSpecificCapabilities [3] SSPSpecificCapabilities DEFAULT aNone:NULL,
    aSspUicc [4] SspUiccCapability OPTIONAL,
    aSspUserInterface [5] SSPUserInterface OPTIONAL,
    aPhysicalInterfaces [6] SEQUENCE OF PhysicalInterface OPTIONAL,  -- list of interfaces
    aSspExternalMaxPowerConsumption [7] INTEGER (0..1000) OPTIONAL
}

/* Capabilities of the SSP - UICC*/
SspUiccCapability ::= SEQUENCE
{
    aNumberOfLogicalChannels [0] INTEGER (1..20) DEFAULT 1,
    aProactivePollingRequirement [1] BOOLEAN DEFAULT FALSE,
    aSupportOfUiccFileSystem [2] BOOLEAN DEFAULT FALSE,
    aSupportOfCardApplicationToolkit [3] BOOLEAN DEFAULT FALSE,
    aCardApplicationToolkitCapabilities [4] OCTET STRING OPTIONAL
}

/* Capabilities of the SSP - User interface */
SSPUserInterface ::= SEQUENCE
{
    aUrl [1] OCTET STRING  -- Url for the user interface
}

-- ASN1STOP
```

aSspRelease: it indicates the release of the present document that is implemented by the SSP. The major version shall have a value that is greater or equal to '0F', which corresponds to Release 15, as the first release of the SSP.

aSspVendorName: it indicates the SSP vendor's name encoded in UTF-8 format, as described in IETF RFC 3629 [31].

aSspClass: it indicates the class of the SSP, as defined in clause 11.

aClassSpecificCapabilities: it contains the SSP capabilities specific for the SSP class, if any. The format is defined in the specification for that SSP class.

aSspUicc: it indicates the capabilities of the SSP to support features defined in the UICC platform:

- **aNumberOfLogicalChannels:** it indicates the total number of logical channels, including the default channel, that is supported by the SSP. This value is specific for the interface where the command is exchanged and is applicable only when APDUs are used. It shall have a value between '01' and '14'.
- **aProactivePollingRequirement:** it indicates if the terminal is required to perform the proactive polling, as described in clause 10.2.6.3. This value is specific for the interface where the command is exchanged and is applicable only when APDUs are used. If the value is FALSE, then the proactive polling is not required. In all other cases, this field shall have the value TRUE.
- **aSupportOfUiccFileSystem:** it indicates if the SSP supports the UICC file system, as described in clause 6.6.1. It shall have the value FALSE if the UICC file system is not supported, TRUE otherwise.
- **aSupportOfCardApplicationToolkit:** it indicates if the SSP supports the Card Application Toolkit. It shall have the value FALSE if the Card Application Toolkit is not supported, TRUE otherwise.
- **aCardApplicationToolkitCapabilities:** it indicates the Card Application Toolkit procedures initiated by the terminal that the SSP supports. This field shall be present if the SSP indicates support of the Card Application Toolkit. It is coded as the value in the CAT service list data object defined in ETSI TS 102 223 [6], clause 8.102.

aSspUserInterface: it indicates the support for the SSP user interface. If this field is not present, the SSP does not have support for a user interface:

- **aUrl:** URL used to access the user interface of the SSP, as defined in clause 6.12.1.

aPhysicalInterfaces: indicates the list of interfaces supported by the SSP:

- **aInterfaceType:** it indicates the type of a physical interface.
- **aMaximumInterfacePowerSupply:** it indicates the maximum current in mA that the SSP can consume over each interface type. If the physical interface does not provide power, value '0' is used.

aSspExternalMaxPowerConsumption: it indicates the maximum consumption of the SSP using external power source. The value indicates the maximum current in mA. The presence of this field is SSP class dependant.

6.5 Storage

The Non-Volatile Memory (NVM) can be non-programmable (ROM) or programmable (i.e. EEPROM, flash, etc.).

Whether the NVM is within the SSP or external to the SSP is SSP class dependant. Consequently, the technical specification of each SSP class shall indicate if the NVM is allowed to be internal and/or external.

When the NVM is within the SSP, it shall be isolated and not be accessible outside the SSP.

Mandatory secure storage requirements are described in clause 6.11.2.3.1.

6.6 Data management

6.6.1 UICC file system

The SSP may support the UICC file system as specified in ETSI TS 102 221 [1], clause 8.1, clause 8.2 and clause 8.3, and the associated security features described in ETSI TS 102 221 [1], clause 9. The technical specification of each SSP class shall indicate if it is mandatory, optional or forbidden.

6.6.2 SSP file system

6.6.2.1 Overview

The SSP may support the SSP file system. The technical specification of each SSP class shall indicate if the SSP file system is mandatory, optional or forbidden.

The SSP file system is an organization of data and metadata on an SSP providing storage capabilities.

The SSP file system includes a metadata description of the data organization and a list of commands to access and manage data storage. The SSP file system provides an access control mechanism to restrict access to the stored data.

The internal system implementation of the SSP file system is out of the scope of the present document.

An example of an SSP file system base architecture is provided in figure 6.1.

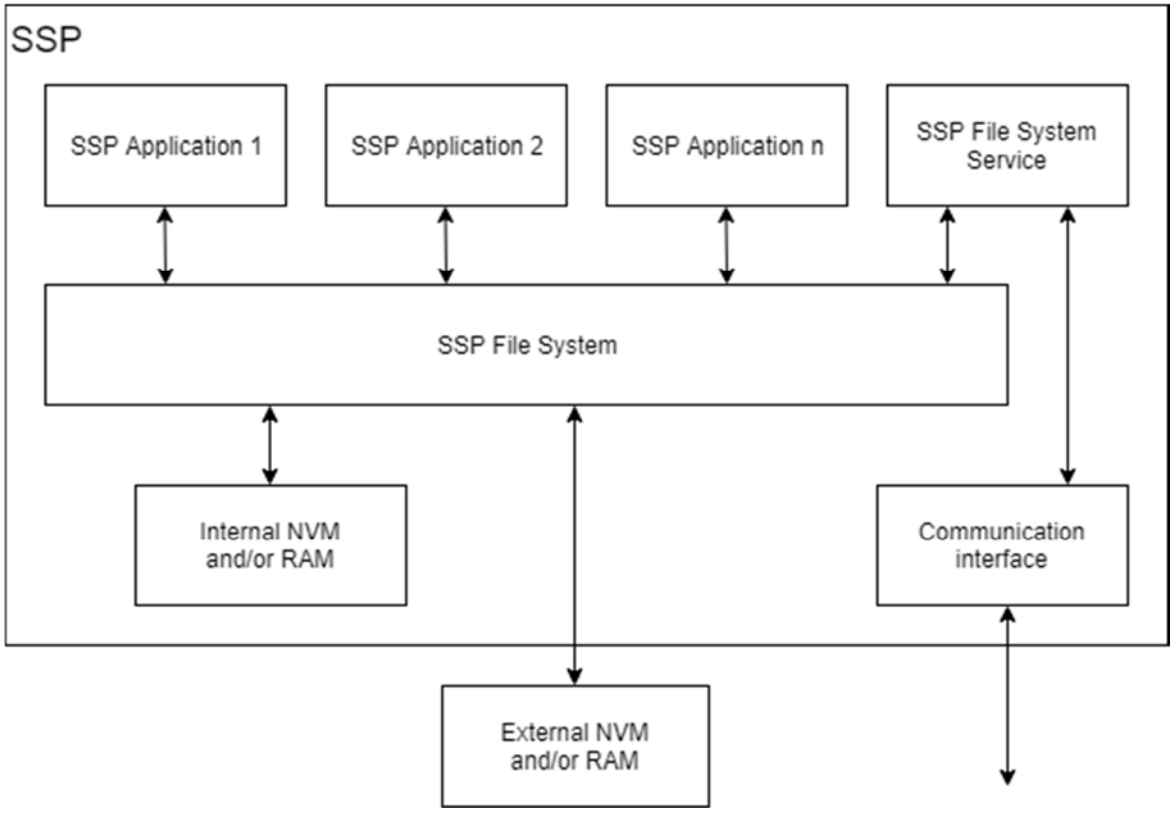

Figure 6.1: Example of SSP file system base architecture

The SSP shall implement an interface between the SSP Applications and the SSP file system in order to make access to the stored data independent of the underlying storage medium. The interface between the SSP Applications and the SSP file system is outside the scope of the present document.

The interface between the SSP file system and the storage on the SSP is outside the scope of the present document.

6.6.2.2 Structure

6.6.2.2.1 Layout

The SSP file system is a collection of objects called nodes, organized in a hierarchical tree. Each node may be an SSP directory, an SSP file or an SSP link. An example of an SSP file system organization is provided in figure 6.2.

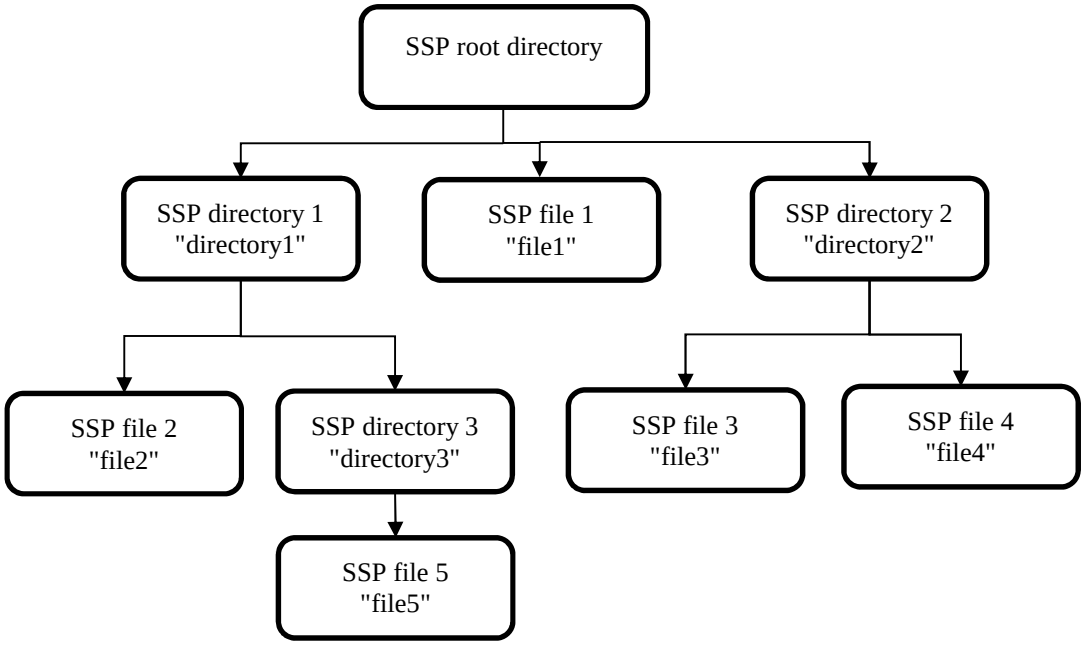

Figure 6.2: Example of SSP file system organization

6.6.2.2.2 Node types

The following node types are defined:

- SSP directory: an SSP directory is a particular node that contains the list of references to other nodes. It contains also a reference to the parent directory. There are two types of directories:
  - SSP directory: it conforms to the above SSP directory definition.
  - SSP root directory: it conforms to the SSP directory definition but does not contain any reference to the parent file object. The SSP file system shall contain only one SSP root directory.
- SSP file: an SSP file is a sequence of data bytes.
- SSP link: an SSP link contains a link to an SSP file. An SSP link shall not link to an SSP directory or to another SSP link.

6.6.2.2.3 Node descriptor

The SSP file system shall allocate a node descriptor per node, containing the following data:

- the node name and the short node name;
- the node type;
- the size of the content (only for SSP files);
- the access control list;
- additional metadata (e.g. proprietary information with limited size).

The storage of the node descriptor is outside the scope of the present document.

The SSP file system shall support a tree of nodes with a minimum height of 5. Each SSP directory shall support a minimum of 256 nodes.

The node descriptor shall be represented with the following ASN.1 syntax.

```
-- ASN1START

/* Node descriptor */
FileSize ::= INTEGER

-- Values of possible OBJECT IDENTIFIERS to identify the metadata types
id-ssp OBJECT IDENTIFIER ::= { itu-t (0) identified-organization (4) etsi (0) smart-secure-platform
(3666) part1 (1) }
id-metadatumtype OBJECT IDENTIFIER ::= { id-ssp metadatumtype (2) }

id-typenull OBJECT IDENTIFIER ::= { id-metadatumtype typenull (1) }
id-typeboolean OBJECT IDENTIFIER ::= { id-metadatumtype boolean (2) }
id-typeinteger OBJECT IDENTIFIER ::= { id-metadatumtype typeinteger (3) }
id-typeoctetstring OBJECT IDENTIFIER ::= { id-metadatumtype typeoctetstring (4) }
id-typeutf8string OBJECT IDENTIFIER ::= { id-metadatumtype typeutf8string (5) }

-- set of possible metadata types defined
MetadatumBasicSet METADATUM ::= {
    { NULL          IDENTIFIED BY { id-typenull } }         |   -- NULL TYPE
    { BOOLEAN       IDENTIFIED BY { id-typeboolean } }      |   -- BOOLEAN TYPE
    { INTEGER       IDENTIFIED BY { id-typeinteger } }      |   -- INTEGER TYPE
    { OCTET STRING  IDENTIFIED BY { id-typeoctetstring } }  |   -- OCTET STRING TYPE-IDENTIFIER
    { UTF8String    IDENTIFIED BY { id-typeutf8string   } }     -- UTF8String
    , ...               -- extensibility, to allow other types in next versions
}

MetaDatum {METADATUM : MetadatumSet} ::= SEQUENCE
{
    aTypeDatum [0] METADATUM.&id({MetadatumSet}),
    aData [1] METADATUM.&Type({MetadatumSet}{@aTypeDatum})
}

NodeDescriptor ::= SEQUENCE
{
    aNodeName NodeName,  -- Node name
    aShortName UUID,  -- Short node name
    aNode CHOICE
    {
        aLink SEQUENCE
        {
            aLinkedFileIdentity NodeIdentity,  -- Identity of the linked SSP file
            aLinkedFileSize FileSize  -- Size of the linked SSP file
        },
        aFile SEQUENCE
        {
            aFileSize FileSize  -- Size of the SSP file
        },
        aDirectory SEQUENCE
        {
        }
    },
    aMetaData SEQUENCE OF MetaDatum {{MetadatumBasicSet}} OPTIONAL,  -- Optional meta data
    aACL SET OF AccessControl OPTIONAL  -- Access Control List attribute
}

-- ASN1STOP
```

Where:

- aDirectory: it indicates that the type of the node is an SSP directory.
- aFile: it indicates that the type of the node is an SSP file and its size in bytes.
- aLink: it indicates that the type of the node is an SSP link, the size and the identity of the linked SSP file.
- aMetaData: if present, it contains a collection of proprietary metadata with limited size. The content of the metadata of SSP links is the metadata of the linked file.
- aACL: if present, it contains a collection of access control. If absent, the node inherits the access control list from its parent node.

6.6.2.2.4 Node identity

All SSP files and SSP directories are referenced by a string, called node name.

The node name of SSP directories and SSP files shall use graphic characters, with a maximum length of 16 bytes after encoding in UTF-8 format, as described in IETF RFC 3629 [31]. The colon character ':' (U+003A) is reserved as the node name separator in the composition of a path. The node names '.' (U+002E) and '..' (two U+002E) are reserved for future usage.

The location of an SSP directory or of an SSP file in the hierarchical tree is described by a path. A path in the hierarchical tree shall be described by a pathname. The SSP file system shall support only the absolute pathname, starting from the root of the hierarchical tree.

A pathname shall be a sequence of one or more node names of SSP directories concatenated by the node name separator, starting from the SSP root directory. The node name of the SSP root directory shall be "SSPFS".

The node reference shall be a string composed by a pathname followed by the node name separator and the node name. Some examples of valid node references related to figure 6.2:

- "SSPFS" identifies the SSP root directory.
- "SSPFS:directory1" identifies the SSP directory 1.
- "SSPFS:directory1:directory3" identifies the SSP directory 3.
- "SSPFS:directory1:file2" identifies the SSP file 2.

All SSP files and SSP directories also have a short node name, which is the UUID version 5 calculated using the domain name system namespace, as defined in IETF RFC 4122 [28] from a URN, as defined in IETF RFC 8141 [29], composed concatenating "urn:etsi.org" (NID), the colon character (U+003A) and the Node reference (NSS).

The short node name may be used to access the node. In this case, the access to the file system appears as completely flat for the SSP file system application and the SSP is responsible to maintain the correct hierarchical structure.

The reference to an SSP directory or SSP file shall be encoded using NodeIdentity as per the following ASN.1 syntax.

```
-- ASN1START

/* Node identity */
NodeName ::= UTF8String (SIZE(1..16))  -- node name encoded in UTF-8
NodeReference ::= SEQUENCE (SIZE(1..6)) OF NodeName  -- pathname and node name

NodeIdentity ::= CHOICE
{
    aShortName UUID,  -- UUID of file reference using absolute pathname
    aNodeReference NodeReference  -- Node reference
}

-- ASN1STOP
```

Where:

- aShortName: it contains the short node name.
- aNodeReference: it contains the node reference.

6.6.2.2.5 File handling

An SSP file can be accessed (i.e. operated) by opening a session. A file session contains all the necessary context to perform operations on one SSP file, including the type of access (e.g. read and/or write). The file session is referenced by a unique identifier called session ID that is provided as a response to the file session open command (FS-OP-FILE-OPEN-Service-Command).

The SSP file system shall support one or more simultaneous file sessions. Several file sessions may apply on the same SSP file. The maximum number of simultaneous file sessions supported by the SSP file system is implementation dependent with a minimum of 2 file sessions and may be retrieved from the capabilities.

The SSP file system supports the following operations related to SSP files:

- opening and closing of an SSP file (i.e. FS-OP-FILE-OPEN-Service-Command, FS-OP-FILE-CLOSE-Service-Command);
- retrieving information of an SSP file (i.e. FS-OP-NODE-GET-INFO-Service-Command);
- reading and writing of data from/to an SSP file (i.e. FS-OP-FILE-READ-Service-Command, FS-OP-FILE-WRITE-Service-Command);
- retrieving the current reading/writing position (i.e. FS-OP-FILE-GET-POSITION-Service-Command).

```
-- ASN1START

AccessMode ::= BIT STRING
{
    eReadAccessMode (0),  -- Access mode: read
    eWriteAccessMode (1)  -- Access mode: write
}
SessionID ::= INTEGER (0..255)

-- ASN1STOP
```

A file session can be opened on an SSP file if all the conditions below are met:

- the SSP file exists in the SSP file system;
- the access conditions of the SSP file are satisfied.

6.6.2.2.6 Administrative operations

The SSP file system supports the following administrative operations:

- retrieving the capabilities of the SSP file system (i.e. FS-ADMIN-GET-CAPABILITIES-Service-Command);
- creating and deleting a node (i.e. FS-ADMIN-CREATE-NODE-Service-Command, FS-ADMIN-DELETE-NODE-Service-Command);
- updating the attributes of a node (i.e. FS-ADMIN-UPDATE-NODE-ATTRIBUTES-Service-Command).

6.6.2.2.7 SSP file system access rights

The SSP file system access rights allow to define if the requested operation is available.

```
-- ASN1START

eFSAccessRight-RequiresSecurePipe AccessorRights ::= { eRight-Bit1 }
eFSAccessRight-ReadContent AccessorRights ::= { eRight-Bit2 }
eFSAccessRight-GetInfo AccessorRights ::= { eRight-Bit3 }
eFSAccessRight-Write AccessorRights ::= { eRight-Bit4 }
eFSAccessRight-UpdateMetadata AccessorRights ::= { eRight-Bit5 }
eFSAccessRight-UpdateACL AccessorRights ::= { eRight-Bit6 }
eFSAccessRight-Delete AccessorRights ::= { eRight-Bit7 }
eFSAccessRight-DeleteChild AccessorRights ::= { eRight-Bit8 }

-- ASN1STOP
```

Where:

- eFSAccessRight-RequiresSecurePipe: this right indicates that, in addition to the permissions required to access the resource, the accessor shall use a secure pipe, as defined in clause 9;
- eFSAccessRight-ReadContent: in case of SSP file and SSP link, this right allows access to read the content. In case of SSP directory, this right allows access to the list of the contained nodes (if the command is allowed);
- eFSAccessRight-GetInfo: this right allows access to retrieve information of a node;
- eFSAccessRight-Write: in case of SSP file and SSP link, this right allows access to write the content. In case of SSP directory, this right allows creation of a node within the SSP directory;

- eFSAccessRight-UpdateMetadata: this right allows the update of the metadata of the node;
- eFSAccessRight-UpdateACL: this right allows the update of the access control list of the node;
- eFSAccessRight-Delete: this right allows the deletion of the node;
- eFSAccessRight-DeleteChild: this right allows the deletion of any node contained in the SSP directory, regardless of the value of eFSAccessRight-Delete of each contained node.

When SSP links are used for operations that access the content of nodes (i.e. FS-OP-FILE-OPEN-Service-Command) or for operations that access the metadata of nodes (i.e. FS-OP-NODE-GET-INFO-Service-Command and FS-ADMIN-UPDATE-NODE-ATTRIBUTES-Service-Command), the SSP shall verify the access control list of both the SSP link and the linked SSP file.

Table 6.1: Applicability of rights

Command	eFSAccessRight-RequiresSecurePipe	eFSAccessRight-ReadContent	eFSAccessRight-GetInfo	eFSAccessRight-Write	eFSAccessRight-UpdateMetadata	eFSAccessRight-UpdateACL	eFSAccessRight-Delete	eFSAccessRight-DeleteChild
FS-OP-NODE-GET-INFO-Service-Command	•	•	•					
FS-ADMIN-UPDATE-NODE-ATTRIBUTES-Service-Command	•				•	•		
FS-ADMIN-CREATE-NODE-Service-Command	•			•				
FS-ADMIN-DELETE-NODE-Service-Command	•						•	•
FS-OP-FILE-OPEN-Service-Command	•	•		•				
FS-OP-FILE-CLOSE-Service-Command	•							
FS-OP-FILE-GET-POSITION-Service-Command	•							
FS-OP-FILE-READ-Service-Command	•							
FS-OP-FILE-WRITE-Service-Command	•							

6.6.2.3 Primitives

6.6.2.3.1 FS-ADMIN-GET-CAPABILITIES-Service-Command

With the command FS-ADMIN-GET-CAPABILITIES-Service-Command, an SSP file system application requests the SSP file system service to retrieve the capabilities of the SSP file system.

```
-- ASN1START

FS-ADMIN-GET-CAPABILITIES-Service-Command ::= [PRIVATE 16] SEQUENCE
{
}

-- ASN1STOP
```

This command has no parameters.

When the request is successful, then the SSP file system service shall include eFS-OK in the response.

```
-- ASN1START

FS-ADMIN-GET-CAPABILITIES-Service-Response-Parameter ::= SEQUENCE
{
    aVersion VersionType,  -- Release of the file system service
```

```
    aSimultaneousFileSessions INTEGER (1..32) DEFAULT 1,  -- Max number of simultaneous file
sessions
    aSimultaneousFileSessionsPerFile INTEGER (1..32) DEFAULT 1,  -- Max number of sessions on a file
    aTotalCapacity INTEGER (0..MAX),  -- Total capacity in bytes
    aFreeCapacity INTEGER (0..MAX),  -- Remaining capacity in bytes
    aMaxMetaDataSizePerNode INTEGER (0..MAX)  -- Maximum metadata size per node
}

FS-ADMIN-GET-CAPABILITIES-Service-Response ::= [PRIVATE 16] SEQUENCE
{
    aFS-Service-Response FS-Service-Response DEFAULT eFS-OK,
    aParameter FS-ADMIN-GET-CAPABILITIES-Service-Response-Parameter OPTIONAL
}

-- ASN1STOP
```

Where:

- aVersion: major and minor release version supported by the file system control service gate;
- aSimultaneousFileSessions: maximum number of simultaneous file sessions supported;
- aSimultaneousFileSessionsPerFile: maximum number of simultaneous file sessions supported on the same file. This value shall be less or equal than aSimultaneousFileSessions;
- aTotalCapacity: total capacity of the SSP file system in bytes;
- aFreeCapacity: remaining free capacity in the SSP file system in bytes;
- aMaxMetaDataSizePerNode: maximum metadata size allowed per node in bytes.

NOTE: The values of the total capacity and the free capacity are independent of the accessor.

6.6.2.3.2 FS-ADMIN-CREATE-NODE-Service-Command

With the command FS-ADMIN-CREATE-NODE-Service-Command, an SSP file system application may create an SSP file, an SSP directory or an SSP link within a hierarchical tree of SSP directories.

The accessor creating a node in an SSP directory shall have the eFSAccessRight-Write access rights on that SSP directory.

```
-- ASN1START

FS-ADMIN-CREATE-NODE-Service-Command ::= [PRIVATE 17] SEQUENCE
{
    aNodeDescriptor NodeDescriptor,  -- Node descriptor
    aNodeDirectoryIdentity NodeIdentity  -- Node identity
}

-- ASN1STOP
```

This command has the following parameters:

- aNodeDescriptor: contains the node descriptor to create a node;
- aNodeDirectoryIdentity: is the SSP Directory into which the new node shall be placed.

The SSP file system service shall ignore the short name included in aNodeDescriptor and compute it.

If the node descriptor indicates an SSP link, the SSP file system service shall ignore the file size and the metadata included in aNodeDescriptor, as the file size and the metadata are provided by the linked SSP file.

When the request is successful, then the SSP file system service shall include eFS-OK in the response.

```
-- ASN1START

FS-ADMIN-CREATE-NODE-Service-Response ::= [PRIVATE 17] SEQUENCE
{
    aFS-Service-Response FS-Service-Response DEFAULT eFS-OK
}
```

```
-- ASN1STOP
```

6.6.2.3.3 FS-ADMIN-DELETE-NODE-Service-Command

With the command FS-ADMIN-DELETE-NODE-Service-Command, an SSP file system application requests the SSP file system service to delete a node.

An accessor is authorized to delete an SSP node if it has the eFSAccessRight-Delete right on the node to be deleted, or if it has the eFSAccessRight-DeleteChild right on the SSP directory containing the node.

The SSP file system shall reject the deletion of a node with the error eFS-NODE-BUSY if a session is ongoing on the node.

The deletion of an SSP directory implies the deletion of all the nodes contained in the SSP directory.

The deletion of an SSP link shall not impact the SSP file that is linked.

After the deletion of a node, all SSP links pointing to that node shall also be deleted by the SSP file system service, irrespective of the delete right to each SSP link.

After an SSP file is erased, it shall not be possible to restore its content.

```
-- ASN1START

FS-ADMIN-DELETE-NODE-Service-Command ::= [PRIVATE 18] SEQUENCE
{
    aNodeIdentity NodeIdentity  -- Node identity
}

-- ASN1STOP
```

This command has the following parameters:

- aNodeIdentity: identity of the node to be deleted.

When the request is successful, then the SSP file system service shall include eFS-OK in the response.

```
-- ASN1START

FS-ADMIN-DELETE-NODE-Service-Response ::= [PRIVATE 18] SEQUENCE
{
    aFS-Service-Response FS-Service-Response DEFAULT eFS-OK
}

-- ASN1STOP
```

6.6.2.3.4 FS-ADMIN-UPDATE-NODE-ATTRIBUTES-Service-Command

With the command FS-ADMIN-UPDATE-NODE-ATTRIBUTES-Service-Command, an SSP file system application requests the SSP file system service to update the access control and the metadata of a node.

The accessor updating the metadata of a node shall have the eFSAccessRight-UpdateMetadata right on that node. If the update is performed on an SSP link, the accessor shall also have the eFSAccessRight-UpdateMetadata right on the linked node. The accessor updating the access control list of a node shall have the eFSAccessRight-UpdateACL right on that node.

```
-- ASN1START

FS-ADMIN-UPDATE-NODE-ATTRIBUTES-Service-Command ::= [PRIVATE 19] SEQUENCE
{
    aNodeIdentity NodeIdentity,  -- Node identity
    aMetaData SEQUENCE OF MetaDatum {{MetadatumBasicSet}} OPTIONAL,  -- New meta data
    aACL SET OF AccessControl OPTIONAL  -- New access control
}

-- ASN1STOP
```

This command has the following parameters:

- aNodeIdentity: identity of the node to update;

- aMetaData: the new meta data of the node;
- aACL: the new access control list of the node.

When the request is successful, then SSP file system service shall include eFS-OK in the response.

```
-- ASN1START

FS-ADMIN-UPDATE-NODE-ATTRIBUTES-Service-Response ::= [PRIVATE 19] SEQUENCE
{
    aFS-Service-Response FS-Service-Response DEFAULT eFS-OK
}

-- ASN1STOP
```

6.6.2.3.5 FS-OP-FILE-OPEN-Service-Command

With the command FS-OP-FILE-OPEN-Service-Command, an SSP file system application requests the SSP file system service to open a file session on a specified SSP file.

The SSP file system application can request the SSP file system service to open the file session on a dedicated data pipe session used for transferring the read or written data.

The accessor opening a session on an SSP file or SSP link shall have the eFSAccessRight-ReadContent and/or the eFSAccessRight-Write right on that node depending on the access mode. If the command is performed on an SSP link, the accessor shall also have the same right(s) on the linked node.

If a dedicated data pipe session is requested, a dynamic identifier is necessary to open the dedicated data pipe session. The command FS-OP-FILE-OPEN-Service-Command has two mode of operation for the generation of the dynamic identifier: the dynamic identifier is either provided by the SSP file system application or by the SSP file system service.

```
-- ASN1START

FS-OP-FILE-OPEN-Service-Command ::= [PRIVATE 20] SEQUENCE
{
    aNodeIdentity NodeIdentity,  -- Node identity
    aAccessMode AccessMode DEFAULT '01'B,  -- Access mode, default: eReadAccessMode
    aGateAppID UUID OPTIONAL,  -- Gate identifier provided by the accessor (only for transfer on a
dedicated data pipe session)
    aDataPipeSession BOOLEAN DEFAULT FALSE  -- Request a dynamic pipe session
}

-- ASN1STOP
```

This command has the following parameters:

- aNodeIdentity: identity of the node;
- aAccessMode: the type of access to the SSP file;
- aGateAppID: the dynamic identifier of the file system data application gate to open the dedicated data pipe session. This gate is linked to the opened SSP file for transferring the read or written data. This parameter shall be used only when the data is exchanged over a dedicated data pipe session. The SSP file system application shall provide a unique aGateAppID per dedicated data pipe session. The aGateAppID may be generated using the version 5 of UUID as specified in IETF RFC 4122 [28], with URN computed by concatenating "urn:etsi.org" (the NID), the colon character (U+003A) and any diversifier;

NOTE 1: As the uniqueness of file system data application gate is provided by the SSP file system application, it is recommended for the SSP file system service to rely on the underlying layer to discriminate between dedicated data pipe sessions.

- aDataPipeSession: requests the SSP file system service to create a dynamic identifier to open a dedicated data pipe session for transferring the read or write data in the SSP file being opened.

NOTE 2: To open a dedicated data pipe session, it is recommended for SSP file system application to request the creation of the dynamic identifier by the SSP file system service.

The fields aGateAppID and aDataPipeSession define if a dedicated data pipe session is requested, and how the dynamic identifier is generated:

- If the dynamic identifier is generated by SSP file system application, the SSP file system application shall issue a FS-OP-FILE-OPEN-Service-Command with the generated identifier provided in aGateAppID and the field aDataPipeSession absent or set to FALSE.
- Otherwise, the dynamic identifier is generated by SSP file system service, the SSP file system application shall issue a FS-OP-FILE-OPEN-Service-Command without aGateAppID and with aDataPipeSession set to TRUE.

If a dedicated data pipe session is not requested, the SSP file system application shall issue a FS-OP-FILE-OPEN-Service-Command without aGateAppID and with aDataPipeSession absent or set to FALSE.

Opening a session on a file sets its current offset pointer to 0.

When the request is successful, then SSP file system service shall include eFS-OK with following parameters in the response.

```
-- ASN1START

FS-OP-FILE-OPEN-Service-Response-Parameter ::= SEQUENCE
{
    aSessionID SessionID,  -- File session
    aGateServID UUID OPTIONAL  -- Gate identifier (only for transfer on dedicated data pipe session)
}

FS-OP-FILE-OPEN-Service-Response ::= [PRIVATE 20] SEQUENCE
{
    aFS-Service-Response FS-Service-Response DEFAULT eFS-OK,
    aParameter FS-OP-FILE-OPEN-Service-Response-Parameter OPTIONAL
}

-- ASN1STOP
```

Where:

- aSessionID: this is the session identifier to reference the SSP file for operation.
- aGateServID: the dynamic identifier of the file system data service gate. The SSP file system service shall provide this parameter only if the dynamic identifier has been requested to be generated by SSP file system service. If the SSP file system service is not able to create a file system data service gate, the file system service shall not provide this parameter. The aGateServID may be generated using the version 5 of UUID as specified in IETF RFC 4122 [28]. The URN is computed by concatenating "urn:etsi.org" (the NID), the colon character (U+003A) and an internal diversifier. The SSP file system service shall prevent conflicting identifiers for dynamically generated gate identifiers.

6.6.2.3.6 FS-OP-FILE-CLOSE-Service-Command

With the command FS-OP-FILE-CLOSE-Service-Command, an SSP file system application requests the SSP file system service to close a specified file session opened by FS-OP-FILE-OPEN-Service-Command command.

```
-- ASN1START

FS-OP-FILE-CLOSE-Service-Command ::= [PRIVATE 21] SEQUENCE
{
    aSessionID SessionID  -- File session
}

-- ASN1STOP
```

This command has the following parameters:

- aSessionID: this is the session identifier to the open SSP file.

If the SSP file system application sends a FS-OP-FILE-CLOSE-Service-Command command while a previous command is ongoing in the same file session, the SSP file system shall perform one of the following operations:

- Terminate the ongoing command and close the ongoing session.

- Reject the FS-OP-FILE-CLOSE-Service-Command command with the error eFS-NODE-BUSY.

When this command is successful then SSP file system service shall include eFS-OK in the response.

```
-- ASN1START

FS-OP-FILE-CLOSE-Service-Response ::= [PRIVATE 21] SEQUENCE
{
    aFS-Service-Response FS-Service-Response DEFAULT eFS-OK
}

-- ASN1STOP
```

If there is a pipe session associated with the aSessionID, the SSP file system application closes this pipe session.

6.6.2.3.7 FS-OP-NODE-GET-INFO-Service-Command

With the command FS-OP-NODE-GET-INFO-Service-Command, an SSP file system application requests the SSP file system service to read the information about an SSP file or an SSP directory.

The accessor retrieving the NodeDescriptor structure shall have the eFSAccessRight-GetInfo right on that node. If the command is performed on an SSP link, the accessor shall also have the eFSAccessRight-GetInfo right on the linked node.

The accessor retrieving a NodeDescriptor structure list of child's node of an SSP directory (i.e. when aContain is set) shall have the eFSAccessRight-ReadContent right on that SSP directory.

```
-- ASN1START

FS-OP-NODE-GET-INFO-Service-Command ::= [PRIVATE 22] SEQUENCE
{
    aNodeIdentity NodeIdentity,  -- Node identity
    aRequestType BIT STRING
    {
        aParent (0),  -- Get info on the parent of the requested node (if not set, get info of
requested node)
        aContain (1),  -- Get info on the child nodes of the requested node (applicable only for SSP
directories)
        aMetaData (2)  -- Include metadata in the response
    } DEFAULT '000'B
}

-- ASN1STOP
```

This command has the following parameters:

- aNodeIdentity: identity of the node;
- aRequestType: indicates the type of the request. The usage of aParent, aContain and aMetaData is as described in table 6.2.

Table 6.2: Coding of aRequestType

aParent	aContain	aMetaData	Reference
0	0	-	The function returns the node descriptor of the node passed in aNodeIdentity.
0	1	-	The function returns the node descriptors of the nodes contained in the node passed in aNodeIdentity.
1	0	-	The function returns the node descriptor of the parent of the node passed in aNodeIdentity.
1	1	-	The function returns the node descriptors of the nodes contained in the parent of the node passed in aNodeIdentity (i.e. the node passed in the aNodeIdentity and its siblings).
-	-	0	The function does not include the metadata in the returned node descriptor(s).
-	-	1	The function includes the metadata in the returned node descriptor(s).

When this command is successful, then the SSP file system service shall include eFS-OK with following parameters in the response.

```
-- ASN1START

FS-OP-NODE-GET-INFO-Service-Response-Parameter ::= SEQUENCE
{
    aNodeDescriptorList SEQUENCE (SIZE (1..255)) OF NodeDescriptor
}

FS-OP-NODE-GET-INFO-Service-Response ::= [PRIVATE 22] SEQUENCE
{
    aFS-Service-Response FS-Service-Response DEFAULT eFS-OK,
    aParameter FS-OP-NODE-GET-INFO-Service-Response-Parameter OPTIONAL
}

-- ASN1STOP
```

Where:

- aNodeDescriptorList: it contains the list of node descriptors requested by the SSP file system application. This list is limited to 255 node descriptors.

6.6.2.3.8 FS-OP-FILE-READ-Service-Command

With the command FS-OP-FILE-READ-Service-Command an SSP file system application requests the SSP file system service to read the content of an SSP file that was previously opened with the command FS-OP-FILE-OPEN-Service-Command.

```
-- ASN1START

FS-OP-FILE-READ-Service-Command ::= [PRIVATE 23] SEQUENCE
{
    aSessionID SessionID,  -- File session
    aOffset UInt32 OPTIONAL,  -- Offset from the beginning of the file
    aNumberOfBytes UInt32 OPTIONAL  -- Number of bytes to read
}

-- ASN1STOP
```

This command has the following parameters:

- aSessionID: this is the session Identifier to reference the SSP file for operation;
- aOffset: start position in the SSP file from offset 0. If omitted, read from the current offset of the SSP file;
- aNumberOfBytes: number of byte to read. If set to 0, the whole SSP file shall be read out.

If the SSP file system application sends a FS-OP-FILE-READ-Service-Command command while a previous command is ongoing in the same file session, the SSP file system shall reject the command with the error eFS-NODE-BUSY.

When this request is successful, then SSP file system service shall include eFS-OK with the following parameters in the response.

```
-- ASN1START

FS-OP-FILE-READ-Service-Response-Parameter ::= SEQUENCE
{
    aSessionID SessionID,  -- File session
    aData OCTET STRING OPTIONAL
}

FS-OP-FILE-READ-Service-Response ::= [PRIVATE 23] SEQUENCE
{
    aFS-Service-Response FS-Service-Response DEFAULT eFS-OK,
    aParameter FS-OP-FILE-READ-Service-Response-Parameter OPTIONAL
}

-- ASN1STOP
```

Where:

- aData: data bytes read from the SSP file. This parameter is used only if the SSP file system application did not pass the gate URI when it opened the file session.

If the read data is received by the SSP file system application on a separate SCL pipe, then the FS-OP-FILE-READ-Service-Response is sent back to the SSP file system application on the same pipe as the FS-OP-FILE-READ-Service-Command after the last data byte has been received on the separate data channel.

6.6.2.3.9 FS-OP-FILE-WRITE-Service-Command

With the command FS-OP-FILE-WRITE-Service-Command, an SSP file system application requests the SSP file system service to write data into an SSP file that was previously opened with the command FS-OP-FILE-OPEN-Service-Command.

```
-- ASN1START

FS-OP-FILE-WRITE-Service-Command ::= [PRIVATE 24] SEQUENCE
{
    aSessionID SessionID,  -- File session
    aOffset UInt32 OPTIONAL,  -- Offset from the beginning of the file
    aDataInfo CHOICE
    {
        aNumberOfBytes UInt32,  -- Number of bytes to write
        aData OCTET STRING  -- Data to write
    }
}

-- ASN1STOP
```

This command has the following parameters:

- aSessionID: this is the session Identifier to reference the SSP file for operation;
- aOffset: start position in the SSP file from offset 0. If omitted, write from the current offset of the SSP file;
- aNumberOfBytes: number of byte to write. The data shall be sent over a pipe session opened to a file system application data gate;
- aData: the data buffer to write into the SSP file from the provided offset. It is recommended to use this option only for short data.

If the SSP file system application sends a FS-OP-FILE-WRITE-Service-Command command while a previous command is ongoing in the same file session, the SSP file system shall reject the command with the error eFS-NODE-BUSY.

When this request is successful then SSP file system service shall include eFS-OK in the response.

```
-- ASN1START

FS-OP-FILE-WRITE-Service-Response-Parameter ::= SEQUENCE
{
    aSessionID SessionID  -- File session
}

FS-OP-FILE-WRITE-Service-Response ::= [PRIVATE 24] SEQUENCE
{
    aFS-Service-Response FS-Service-Response DEFAULT eFS-OK,
    aParameter FS-OP-FILE-WRITE-Service-Response-Parameter OPTIONAL
}

-- ASN1STOP
```

If the write data is sent by the SSP file system application on a separate channel, then the FS-OP-FILE-WRITE-Service-Response is sent back to the SSP file system application on the same pipe as the FS-OP-FILE-WRITE-Service-Command after the last data byte has been received on the separate data channel.

6.6.2.3.10 FS-OP-FILE-GET-POSITION-Service-Command

With the command FS-OP-FILE-GET-POSITION-Service-Command, an SSP file system application requests to SSP file system service to retrieve the current offset position in an SSP file that was previously opened with the command FS-OP-FILE-OPEN-Service-Command.

```
-- ASN1START

FS-OP-FILE-GET-POSITION-Service-Command ::= [PRIVATE 25] SEQUENCE
{
    aSessionID SessionID  -- File session
}

-- ASN1STOP
```

This command has the following parameters:

- aSessionID: this is the session Identifier to reference the SSP file for operation.

If the SSP file system application sends a FS-OP-FILE-GET-POSITION-Service-Command command while a previous command is ongoing in the same file session, the SSP file system shall reject the command with the error eFS-NODE-BUSY.

When the request is successful then the SSP file system service shall include eFS-OK in the response.

```
-- ASN1START

FS-OP-FILE-GET-POSITION-Service-Response-Parameter ::= SEQUENCE
{
    aCurrentOffset UInt32 OPTIONAL  -- Current offset
}

FS-OP-FILE-GET-POSITION-Service-Response ::= [PRIVATE 25] SEQUENCE
{
    aFS-Service-Response FS-Service-Response DEFAULT eFS-OK,
    aParameter FS-OP-FILE-GET-POSITION-Service-Response-Parameter OPTIONAL
}

-- ASN1STOP
```

Where:

- aCurrentOffset: current offset of the SSP file.

6.6.2.4 Response code

6.6.2.4.1 Overview

The SSP file system service provides the following response codes to SSP file system primitives.

```
-- ASN1START

FS-Service-Response ::= ENUMERATED
{
    eFS-OK (0),  -- no error
    eFS-E-CMD-PAR-UNKNOWN (2),  -- unknown or illegal command parameter
    eFS-E-NOK (3),  -- the command has failed
    eFS-NODE-BUSY (9),  -- The file system is already processing an operation on the node
    eFS-NODE-NOT-FOUND (10),  -- Node not found
    eFS-OPERATION-ILLEGAL (11),  -- Illegal operation (e.g. opening a file with a directory identity
instead a file identity)
    eFS-NOT-ENOUGH-SPACE (12),  -- The operation exceeds the size limit of a file
    eFS-BAD-SESSSION-ID (13),  -- the session identifier related to a file does not exist
    eFS-ACL-RULES-VIOLATIONS (14), -- the operation of the administration violates the ACL rules
associated to a node
    eFS-MAX-FILE-SESSION-REACHED (15)  -- the maximum number of file sessions has been reached
}

-- ASN1STOP
```

Where:

- eFS-OK: Command completed successfully;
- eFS-E-CMD-PAR-UNKNOWN: Format of the command parameters is wrong;
- eFS-E-NOK: Command was rejected and/or not completed;

- eFS-NODE-BUSY: The file system is already processing an operation on the node;
- eFS-NODE-NOT-FOUND: Node not found;
- eFS-OPERATION-ILLEGAL: Illegal operation (e.g. opening a file with a directory identity instead a file identity);
- eFS-NOT-ENOUGH-SPACE: The operation exceeds the size limit of a file or the size limit of the metadata;
- eFS-BAD-SESSSION-ID: The session identifier related to a file does not exist;
- eFS-ACL-RULES-VIOLATIONS: The operation of the administration violates the ACL rules associated to a node;
- eFS-MAX-FILE-SESSION-REACHED: The maximum number of file sessions has been reached.

6.6.2.4.2 Response code to SSP file system primitives

Table 6.3 shows for each primitive the possible response code returned.

Table 6.3: SSP FS server commands/responses

Command	eFS-OK	eFS-E-CMD-PAR-UNKNOWN	eFS-E-NOK	eFS-NODE-BUSY	eFS-NODE-NOT-FOUND	eFS-OPERATION-ILLEGAL	eFS-NOT-ENOUGH-SPACE	eFS-BAD-SESSSION-ID	eFS-ACL-RULES-VIOLATIONS	eFS-MAX-FILE-SESSION-REACHED
FS-ADMIN-GET-CAPABILITIES-Service-Command	•		•						•	
FS-ADMIN-CREATE-NODE-Service-Command	•	•	•			•	•		•	
FS-ADMIN-DELETE-NODE-Service-Command	•	•	•	•	•	•			•	
FS-ADMIN-UPDATE-NODE-ATTRIBUTES-Service-Command	•	•	•		•	•	•		•	
FS-OP-FILE-OPEN-Service-Command	•	•	•	•	•				•	•
FS-OP-FILE-CLOSE-Service-Command	•	•	•					•		
FS-OP-NODE-GET-INFO-Service-Command	•	•	•		•				•	
FS-OP-FILE-READ-Service-Command	•	•	•	•	•	•		•		
FS-OP-FILE-WRITE-Service-Command	•	•	•	•	•	•	•	•		
FS-OP-FILE-GET-POSITION-Service-Command	•	•	•	•	•	•		•		

6.7 SSP identification

The SSP identification mechanism for the SSP is dependent on the SSP class and is specified for each class.

6.8 Runtime environment

6.8.1 CAT Runtime Environment

The SSP has the option of supporting the CAT Runtime Environment as specified in ETSI TS 102 241 [12] based on the Java Card™ Platform [9], [10] and [11]. This clause shall apply when the CAT Runtime Environment is supported.

Card application toolkit specific fields in the capability exchange procedure indicate the support and the capabilities for the card application toolkit in both the terminal and the SSP.

On SSPs that implement SCL, the SSP may support the mechanism of the UICC APDU service gate described in clause 10.2.8.2. In this case, the CAT Runtime Environment shall:

- Send and receive APDUs as defined in ETSI TS 102 221 [1], via the UICC APDU service gate defined in clause 10.2.8.2.2.
- Issue an EVT_TOOLKIT_REQUEST defined in clause 10.2.8.2.3.3 if a proactive command has to be sent to the terminal.
- Map the SSP command EXCHANGE CAPABILITIES defined in clause 10.2.3.2 to the events EVENT_PROFILE_DOWNLOAD and EVENT_FIRST_COMMAND_AFTER_ATR defined in ETSI TS 102 241 [12].

On SSPs that implement SCL, the SSP may support the mechanism of the CAT gate described in clause 10.8. In this case, the CAT Runtime Environment shall:

- Send and receive CAT commands and responses, via the CAT application gate defined in clause 10.8.3.
- Trigger the applets based on events received by the CAT application gate, replacing the APDU based triggering mechanism defined in ETSI TS 102 241 [12], clause 6.1.
- Map the capability exchange procedure (defined in clause 6.4.2.3) to the events EVENT_PROFILE_DOWNLOAD and EVENT_FIRST_COMMAND_AFTER_ATR defined in ETSI TS 102 241 [12].

On SSPs that implement the UICC file system, the events EVENT_EXTERNAL_FILE_UPDATE and EVENT_REMOTE_FILE_UPDATE shall be raised according to ETSI TS 102 241 [12] on update operations on the UICC file system.

The SSP has the option of supporting the Contactless Framework as defined in ETSI TS 102 705 [33] based on the Java Card™ Platform [9], [10] and [11]. On SSPs that implement the SCL, the Contactless Framework shall register an HCI gate defined in clause 10.7.2.

6.9 SSP suspension

SSP suspension can be used by the terminal to suspend the SSP when access is not required for long periods of time, in order to reduce the overall power consumption. The usage of the suspension mechanism is allowed only if the SSP has a single active physical interface. When the SSP is suspended, the terminal deactivates the physical interface to the SSP, following the sequence specified for that physical interface. If the SSP has an independent power source, the power may be removed by the terminal after the physical interface is deactivated.

NOTE: Suspension of SSPs with multiple physical interfaces is FFS.

The procedure can be used only when it is indicated as supported by the SSP in the capability exchange procedure.

The terminal shall maintain the logical status as before the suspension and it shall resume the SSP for any event for which it had previously registered.

To resume the SSP, the terminal shall first perform the initialization of the SSP as described in clause 6.4, including the capability exchange procedure. The electrical parameters shall remain unchanged during and after the resume operation.

The support of SSP suspension is currently supported for APDU only and the corresponding APDU command is defined in clause 10.2.7. In case SCL is used, the suspension of the SSP may only be used when there is a single dynamic pipe to the SSP and it is the pipe for transporting APDUs, as defined in clause 10.2.8. The SSP suspension shall be rejected in all other cases.

6.10 SSP Applications

6.10.1 Overview

SSP Applications may have different characteristics, as described in clause 5.2.

SSP Applications may be implemented on top of an Execution Framework, as described in clause 5.5. One such Execution environment is the CAT Runtime Environment specified in ETSI TS 102 241 [12] based on the Java Card™ Platform [9], [10] and [11].

The SSP shall support one or more SSP Applications capable of processing data.

The SSP shall allow one or more SSP Applications to exchange data with other entities outside the SSP. One SSP Application shall not block another SSP Application from exchanging data with the terminal:

- on a different SSP interface session; or
- on the same SSP interface session, if supported by the protocol stack of the interface.

NOTE: Some SSPs may not be able to process the data of two or more SSP Applications at the same time, due to potential restrictions of the execution environment and/or of the application protocol (e.g. APDUs).

6.10.2 Ownership and security considerations

If the SSP implements the CAT Runtime Environment according to ETSI TS 102 241 [12], the rules and mechanisms for the management of Applications on the UICC shall apply. These are based on the GlobalPlatform Card Specification [22], its Amendments and the GlobalPlatform UICC Configuration [25] as described in ETSI TS 102 226 [7].

If the SSP implements applications based on the file system (ISO/IEC 7816-4 [4]), the rules are for further study.

If the SSP implements native applications, the rules are for further study.

6.10.3 Lifecycle management

The lifecycle management depends on the type of the SSP Application:

- SSP Applications running in the CAT Runtime Environment according to ETSI TS 102 241 [12]. In this case the rules and mechanisms for the management of the lifecycle of Security Domains and Applications according to GlobalPlatform Card Specification [22] and ETSI TS 102 226 [7] shall apply.
- SSP Applications running as native applications.
- SSP Applications running on a new type of execution framework.

The rules of the lifecycle of SSP Applications are defined for each SSP class.

6.10.4 Identification and discovery

The identification and the discovery of the SSP Applications depend on the protocol stack used on the SSP interface session.

6.11 SSP security

6.11.1 SSP security architecture

The SSP is intended to provide a programmable, secure execution environment for applications.

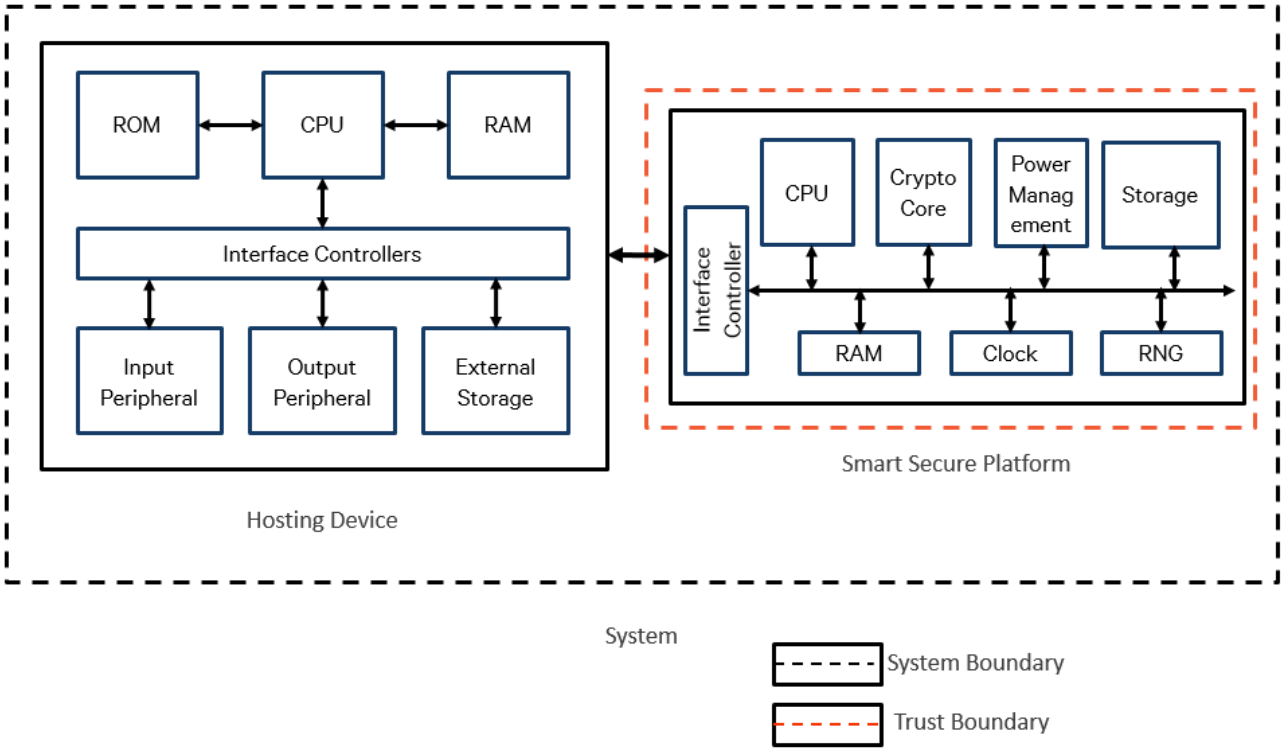

Figure 6.3: SSP Security

Typical hardware components within the SSP include:

- Central Processing Unit, described as CPU in figure 6.3, which is isolated from the rest of the system.
- An internal clock which is not shared with any component outside the SSP.
- A Random Number Generator, described as "RNG" in figure 6.3.
- A Crypto core for execution of cryptographic operations. The SSP shall not use any external processor or co-processor for the execution of cryptographic operations.
- Random Access Memory, described as "RAM" in figure 6.3.
- Storage: the SSP may have its own non-volatile memory contained in the SSP itself or rely on the external storage.
- Power Management Unit governing the power functions of the SSP.

Any entity external to the SSP shall not be able to directly access any hardware or software component within the SSP.

6.11.2 Mandatory requirements

6.11.2.1 Overview

This clause contains the list of all mandatory requirements that shall be implemented by all SSP classes. The implementation details are left to the individual SSP class if not defined in this clause. Additional mandatory requirements can be defined by an SSP class.

6.11.2.2 Security of SSP executable code

SSP shall provide confidentiality, integrity, and replay protection (i.e. ability to prevent outdated executable code from running on the same SSP and ability to prevent executable code of an SSP from running on another SSP) for any executable code inside the SSP. Any SSP executable code shall be authenticated by the SSP entity that loads it.

6.11.2.3 Privacy of data

6.11.2.3.1 Secure storage

The SSP code and data shall be exclusively processed within the SSP.

The SSP code shall not be exposed outside the SSP in clear text. The SSP data shall only be exposed outside the SSP under the control of the SSP. If SSP code and data need to be stored outside the SSP, they shall be encrypted and integrity protected.

All the credentials used to encrypt the code and data shall only be stored and used within the SSP. The SSP shall depend only on its own cryptographic means.

The SSP shall implement mechanisms to prevent that an older version of the non-volatile storage can be re-used after it was superseded by a new SSP transaction.

6.11.2.4 SSP transactions

An SSP transaction starts when the SSP receives a command to process and terminates when the SSP sends the response for that command. The transaction may be started by a command from the terminal, from the network or from an application running in the SSP itself.

If the status of the non-volatile memory needs to be modified after the execution of an SSP transaction, the SSP shall perform the update of the non-volatile memory before providing the response of the transaction to the client that initiated it. This includes the fact that it shall not be possible to restore the previous state of the non-volatile memory. If the NVM modification has not been successful for any reason, the previous content of NVM shall be restored.

6.11.2.5 Attack resistance

The SSP shall be resistant to various attacks including but not limited to:

- Side channel attacks such as simple power-analysis, differential power-analysis and timing analysis. Fault injection via voltage and clock frequency alterations, exposure to extreme light or temperatures.
- Physical probing or tampering.
- Injection via well-crafted input messages into the SSP.
- Analysis through usage of test circuitry.

The levels of resistance and attack prevention schemes are left to the specific SSP class.

6.11.3 Optional requirements

6.11.3.1 Overview

This clause contains the list of optional requirements that may be implemented by each SSP class. The implementation details are left to the SSP class if not defined in this clause. Additional optional requirements can be defined by an SSP class.

6.11.3.2 Random number generator

An SSP may have its own Random Number Generator (RNG). The characteristics of the RNG depend on the SSP class and are defined in the corresponding specification.

6.11.3.3 Remote provisioning

The SSP may include an optional secure mechanism in order to allow remote provisioning of its software components, including applications, part of or all the operating system. The mechanisms for the remote provisioning depend on the SSP class and are defined in the corresponding specification.

6.11.3.4 Remote auditing

Remote auditing is defined as the assessment of the integrity of the SSP hardware platform and optionally of some of the software components of the SSP by an entity outside the terminal. The assessment shall ensure with a coverage higher than 80 % that the SSP hardware platform and the optional software components have not changed since the reference SSP used for the certification.

NOTE: 80 % is a compromise between cost, time and coverage.

Remote auditing process is optional. If supported:

- The SSP class shall define an interface to the remote audit function of the SSP accessible from SSP Applications.
- The SSP class may define an interface to the remote audit function of the SSP accessible from terminal.
- The results of the remote audit function operations from the terminal interface shall be different than the ones collected from the SSP applications interface when using the same input parameters of the remote audit function.

6.11.4 Security certification

6.11.4.1 Overview

A certification process may be defined for each SSP class.

These certification processes shall help the secure application provider to assess the level of trust it can give to the SSP and thus assess if its secure applications can be hosted by this particular SSP.

6.12 User interface

6.12.1 Web-based user interface

6.12.1.1 Overview

The SSP may implement a web server, based on the HTTP(s) protocol as defined in IETF RFC 7230 [18] and in IETF RFC 2818 [19], in order to provide a user interface for the user to access the contents and the services available in the SSP. The user interface is accessed by the terminal using the web browser.

NOTE: The web browser used to access the SSP user interface may be implemented as a standalone application or integrated within another application (e.g. WebView).

The technologies used by the SSP to provide the user interface are outside the scope of the present document (e.g. HTML version, CSS support, JavaScript support and so on).

If the SSP supports the web-based user interface, it shall:

- indicate the URL to be used for the entry page in the capability exchange, as defined in clause 6.4.2.5;
- support the SCL protocol, as defined in clause 8;
- open a TCP server socket with local access only using the TCP control gate, as defined in clause 10.4, using the same local port as indicated in the URL.

The web server in the SSP is accessed by the terminal using the URL retrieved during the exchange capability procedure, and using the TCP gates of the SCL protocol.

6.12.1.2 Port values

The SSP user interface should use the TCP port number 3516 for HTTP and the port 4116 for HTTP over TLS. Both ports are already reserved by IANA. Port 3516 is reserved as "smartcard Port" and port 4116 as "smartcard-TLS".

NOTE: If the terminal has two or more active SSPs, possible conflict in the port may result in inability of the terminal to present the user interface of some of the SSPs.

6.12.1.3 Presentation of SSP user interface

The icon and corresponding text to indicate the availability of the SSP user interface to the user may be retrieved using the following URLs defined in the well-known URI format as defined in IETF RFC 8615 [30]:

- Icon: SSP user interface URL as defined in clause 6.4.2.5 followed by "/.well-known/icon.png".
- Text: SSP user interface URL as defined in clause 6.4.2.5 followed by "/.well-known/text".

The text shall be encoded in UTF-8 as defined in IETF RFC 3629 [31].

The mechanism to display the text and the icon (e.g. font, image resizing, etc.) is outside the scope of the present document.

The mechanism used by the terminal to display the SSP user interface to the user is outside the scope of the present document.

6.13 Accessor authentication

6.13.1 Overview

The accessor authentication service is the service in the SSP responsible of authenticating external entities accessing resources on the SSP. The entities involved in this process are:

- accessors;
- access control(s);
- access control list(s);
- resources;
- accessor authentication service.

Access control list(s) are associated with resources. The access control list contains the access control(s) which store the rights and permissions to be verified when an accessor tries to access the resource.

Prior to access to a resource, an accessor has to authenticate itself based on some credentials which are stored in the accessor authentication service. After successful authentication, an accessor has access to the resource within the limits defined in the access control list of the resource.

The accessor authentication service serves as a repository of known accessors and their credentials and provides services for the verification of these credentials.

Figure 6.4 illustrates the relationships between these entities.

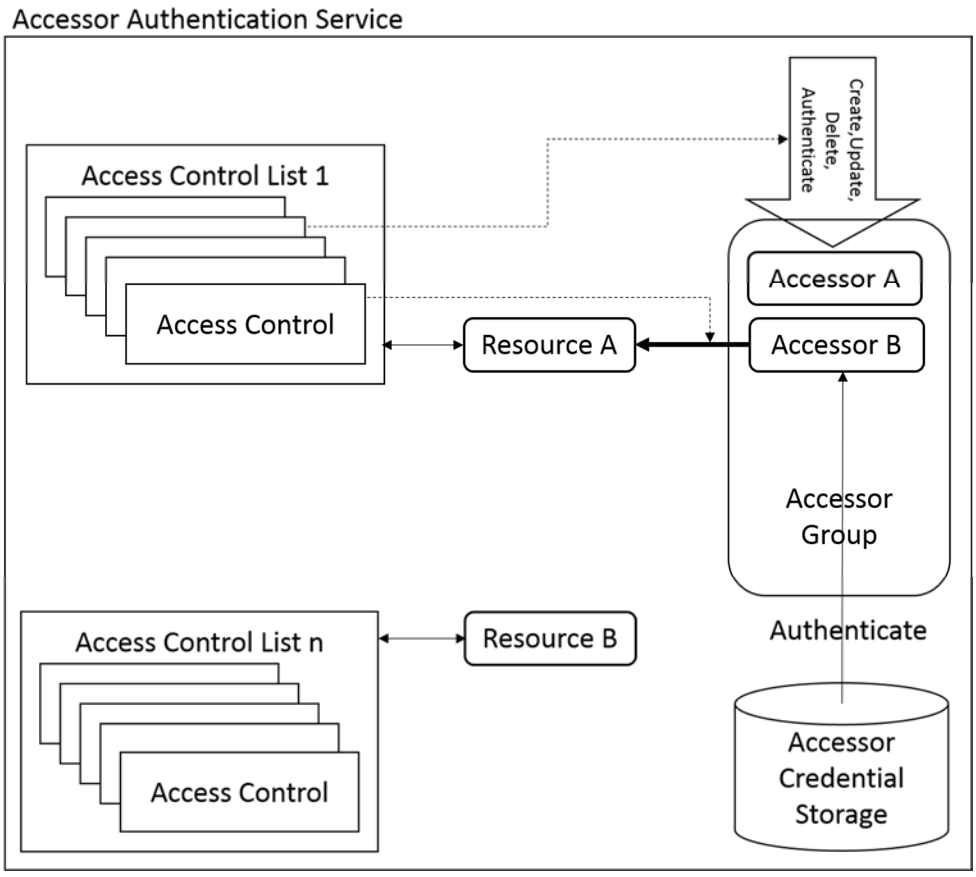

Figure 6.4: Relationships between accessors and resources

6.13.2 Access control

6.13.2.1 Overview

An access control indicates the rights on the resource for an authenticated accessor. An access control may include an additional accessor called grantor. The list of operations allowed by the rights in the access control is application dependent.

When the grantor is present in the access control, the access to the resource is permitted to the accessor if both the accessor and the grantor are authenticated. The rights provided in the access control are only for the accessor and are independent of any rights of the grantor.

6.13.2.2 Description

The access control shall be represented with the following ASN.1 description.

```
-- ASN1START

AccessControl ::= SEQUENCE
{
    aAccessorIdentity AccessorIdentity,  -- Identity of the accessor accessing the resource
    aAccessorRights AccessorRights,  -- Accessor rights (e.g. delete, update).
    aGrantorIdentity AccessorIdentity OPTIONAL  -- Identity of the grantor
}

-- ASN1STOP
```

where:

- aAccessorIdentity: identity of the accessor accessing the resource.
- aAccessorRights: access rights of the accessor accessing the resource, as defined in clause 6.13.2.3.

- aGrantorIdentity: grantor giving access to the accessor to access the resource.

Figure 6.5 illustrates the structure of an access control.

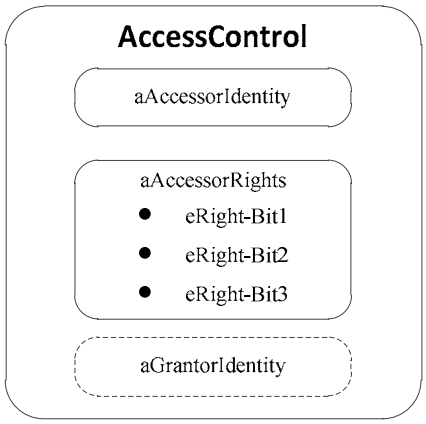

Figure 6.5: Structure of the access control

Examples of access control are present in Annex H.

6.13.2.3 Accessor rights to a resource

The accessor rights indicate the rights that an accessor has on a resource after it has been successfully authenticated using the accessor authentication service described in clause 10.9.

An accessor may have zero or more rights, which are represented as a bitmask. A detailed description of the meaning of each right is defined by the service accessing the resource.

```
-- ASN1START

AccessorRights ::= BIT STRING
{
    eRight-Bit1 (0),
    eRight-Bit2 (1),
    eRight-Bit3 (2),
    eRight-Bit4 (3),
    eRight-Bit5 (4),
    eRight-Bit6 (5),
    eRight-Bit7 (6),
    eRight-Bit8 (7),
    eRight-Bit9 (8),
    eRight-Bit10 (9),
    eRight-Bit11 (10),
    eRight-Bit12 (11),
    eRight-Bit13 (12),
    eRight-Bit14 (13),
    eRight-Bit15 (14),
    eRight-Bit16 (15),
    eRight-Bit17 (16),
    eRight-Bit18 (17),
    eRight-Bit19 (18),
    eRight-Bit20 (19),
    eRight-Bit21 (20),
    eRight-Bit22 (21),
    eRight-Bit23 (22),
    eRight-Bit24 (23),
    eRight-Bit25 (24),
    eRight-Bit26 (25),
    eRight-Bit27 (26),
    eRight-Bit28 (27),
    eRight-Bit29 (28),
    eRight-Bit30 (29),
    eRight-Bit31 (30),
    eRight-Bit32 (31)
} (SIZE(32))
```

```
-- ASN1STOP
```

6.13.3 Access control list

An Access Control List (ACL) is a list of access controls, as defined in clause 6.13.2.

The access control list shall be represented with the following ASN.1 description.

```
-- ASN1START

AccessControlList ::= SET OF AccessControl  -- Access control list

-- ASN1STOP
```

6.13.4 Accessor

6.13.4.1 Overview

An accessor identifies an application which is acting on behalf of an entity, e.g. user or modem. The accessor claims an identity when accessing a resource.

There are two separate types of accessors: users and groups.

Accessors of type users have associated accessor conditions and credentials, that are used to authenticate the accessor.

Accessors of type group are only used to include multiple accessors of type user, so that access control lists of resources can be manipulated more conveniently, applying certain access rules to multiple accessors of type user at the same time. Accessors of type group have no associated conditions and credentials.

The authentication of one member of the group does not imply the authentication of the other members of the group.

The accessor shall be represented with the following ASN.1 description.

```
-- ASN1START

Accessor ::= [PRIVATE 8] CHOICE
{
    aAccessorGroup AccessorGroup,
    aAccessorUser AccessorUser
}

AccessorGroup ::= SEQUENCE
{
    aAccessorIdentity AccessorIdentity,  -- Identity of the accessor
    aMembersOfGroup SET OF AccessorIdentity,  -- Members of the group
    aACL AccessControlList  -- Access control list
}

AccessorUser ::= SEQUENCE
{
    aAccessorIdentity AccessorIdentity,  -- Identity of the accessor
    aAccessorConditions AccessorConditions OPTIONAL,  -- Accessor conditions
    aACL AccessControlList  -- Access control list
}

-- ASN1STOP
```

Where:

- aAccessorIdentity: identity of the accessor, assigned by the creating accessor via the accessor authentication service defined in clause 10.9.
- aMembersOfGroup: list of all the accessors in the group. All accessors in the group shall be of type AccessorUser.
- aAccessConditions: list of accessor conditions, as defined in clause 6.13.4.4, which need to be verified against credentials in order to authenticate the accessor (see AAS-OP-AUTHENTICATE-ACCESSOR-Service-Command in clause 6.13.5.5).

- aACL: access control list to access the accessor (seen as a resource) using the accessor authentication service defined in clause 10.9 (e.g. delete or update the accessor).

Figure 6.6 illustrates the structure of the information related to the accessor.

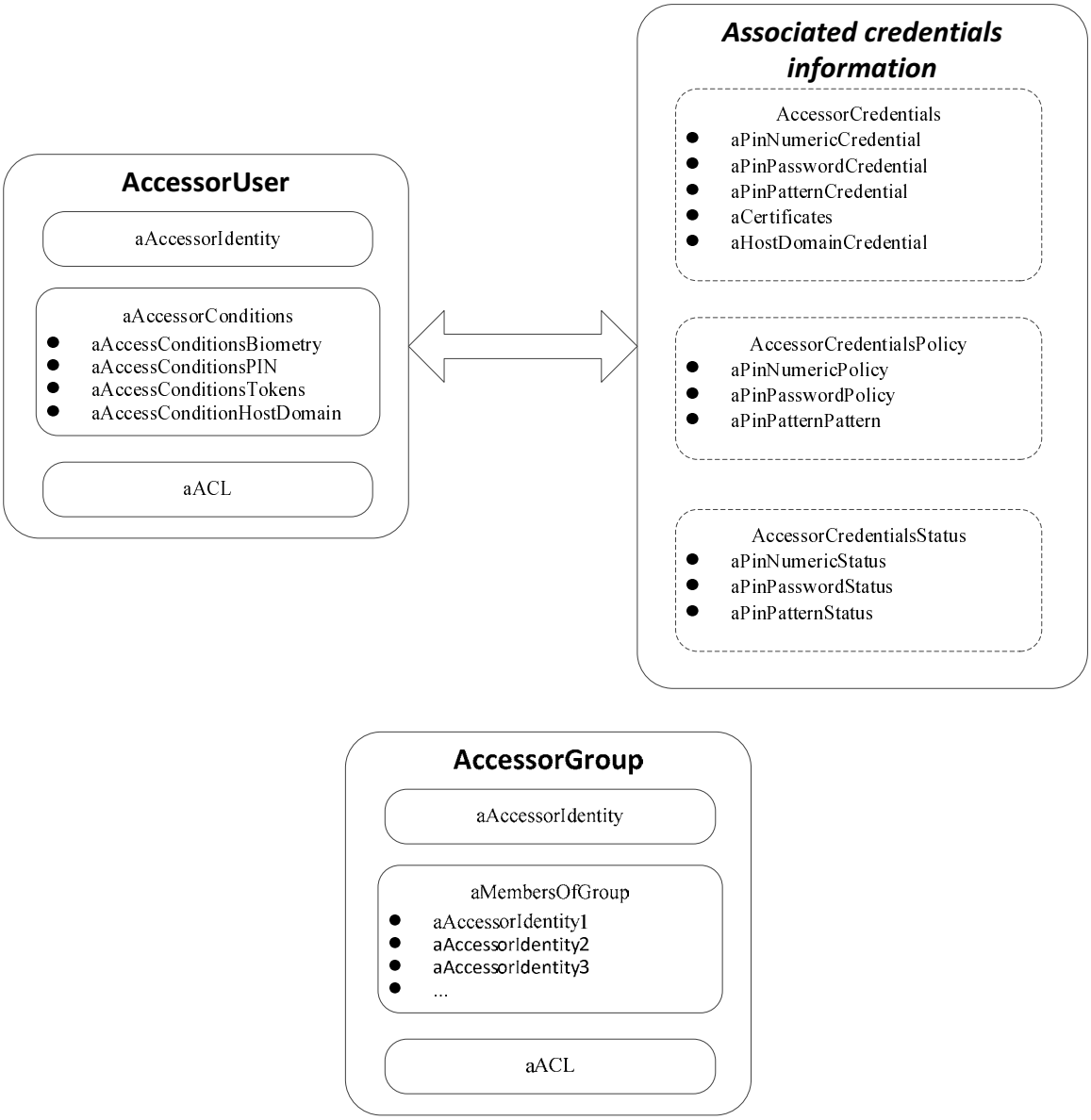

Figure 6.6: Structure of the accessors

6.13.4.2 Anonymous accessor

The anonymous accessor is an accessor that does not require any credential to be authenticated. An anonymous accessor shall always be present in the authentication service.

6.13.4.3 Accessor identity

Each accessor is identified by a unique accessor identity. The accessor identity is assigned by the creator of the accessor at the time of creation using UUID version 5 calculated using the domain name system namespace, as defined in IETF RFC 4122 [28] from a URN, as defined in IETF RFC 8141 [29]. The URN shall be the concatenation of "urn:" and domain name of the organization defining the accessor (for NID section), the colon character (U+003A), "SSP:ASN.1", the colon character (U+003A) and the accessor name. The accessor identity for a given accessor shall not change over time.

The accessor authentication service shall prevent that two accessors have the same accessor identity.

```
-- ASN1START

AccessorIdentity ::= UUID  -- accessor identity

-- ASN1STOP
```

Table 6.4 defines some of the predefined accessor identity values.

Table 6.4: Pre-defined accessor identities

Accessor	NID	NSS	Pre-computed identifier
Anonymous accessor	urn:etsi.org	SSP:ASN.1:Anonymous	4E46645F-E600-5A70-AD7A-60D6E5345E0B

6.13.4.4 Accessor conditions

An accessor may have zero or more conditions described in AccessorConditions, which indicate the types of credentials that shall be used to authenticate the accessor. At least one condition shall be satisfied to properly authenticate the accessor.

```
-- ASN1START

AccessorConditions ::= SEQUENCE
{
    aAccessConditionsBiometry AccessorConditionsBiometry OPTIONAL,
    aAccessConditionsPIN AccessorConditionsPIN OPTIONAL,
    aAccessConditionsTokens AccessorConditionsToken OPTIONAL,
    aAccessConditionHostDomain AccessConditionHostDomain OPTIONAL
}

-- ASN1STOP
```

where:

- aAccessConditionsBiometry: related to biometric recognition;
- aAccessConditionsPIN: related to the PIN verification;
- aAccessConditionsTokens: related to cryptographic signature/challenge verification;
- aAccessConditionHostDomain: related to the host domain hosting the accessor.

The biometric based conditions (AccessorConditionsBiometry) are reserved for future usage.

```
-- ASN1START

AccessorConditionsBiometry ::= [PRIVATE 9] BIT STRING
{
    eReservedForFuture (0)  -- Reserved for future usage
} (SIZE(32))

-- ASN1STOP
```

The PIN based conditions (AccessorConditionsPIN) shall be one of the following conditions.

```
-- ASN1START

AccessorConditionsPIN ::= [PRIVATE 10] BIT STRING
{
    ePinNumeric (0),  -- The user shall present a numeric PIN
    ePinPassword (1),  -- The user shall present a password
    ePinPattern (2)  -- The user shall present a graphical pattern
} (SIZE(32))

-- ASN1STOP
```

Where:

- ePinNumber: the user shall present its PIN, which is composed of a number of numeric digits (from 0 to 9);

- ePinPassword: the user shall present its password;
- ePinPattern: the user shall present a graphical pattern.

The token based conditions (AccessorConditionsToken) shall be one of the following conditions.

```
-- ASN1START

AccessorConditionsToken ::= [PRIVATE 11] BIT STRING
{
    eTokenCertificate (0)  -- A token verification by using the certificate shall be performed
} (SIZE(32))

-- ASN1STOP
```

Where:

- eTokenCertificate: the authentication procedure defined in clause 9.4.1 shall be successful.

The host domain condition (AccessConditionHostDomain) shall be:

```
-- ASN1START

AccessConditionHostDomain ::= [PRIVATE 12] BOOLEAN

-- ASN1STOP
```

The Boolean value indicates if an accessor is authenticated if the host domain identifier of the accessor is in the list of host domain identifiers indicated by HostDomainCredential.

6.13.4.5 Access rights

The access rights attached to the access control list of an accessor allow to define if the requested operation is available.

```
-- ASN1START

eAASAccessRight-RequiresSecurePipe AccessorRights ::= {eRight-Bit1 }
eAASAccessRight-Create AccessorRights ::= { eRight-Bit2 }
eAASAccessRight-Delete AccessorRights ::= { eRight-Bit3 }
eAASAccessRight-Update AccessorRights ::= { eRight-Bit4 }
eAASAccessRight-UpdateACL AccessorRights ::= { eRight-Bit5 }
eAASAccessRight-UpdateGroup AccessorRights ::= { eRight-Bit6 }
eAASAccessRight-UpdateCredentialPolicy AccessorRights ::= { eRight-Bit7 }
eAASAccessRight-UpdateCredentialStatus AccessorRights ::= { eRight-Bit8 }

-- ASN1STOP
```

Where:

- eAASAccessRight-RequiresSecurePipe: this right indicates that, in addition to the permissions required to access the resource, the accessor shall use a secure pipe, as defined in clause 9. This right is only relevant for administrative commands;
- eAASAccessRight-Create: this right allows the creation of a new accessor;
- eAASAccessRight-Delete: this right allows the deletion of the accessor;
- eAASAccessRight-Update: this right allows the update of the conditions and credentials of the accessor (valid only for an accessor of type user, this bit shall be ignored if set for accessor of type group);
- eAASAccessRight-UpdateACL: this right allows the update of the access control list of the accessor;
- eAASAccessRight-UpdateGroup: this right allows the update of the members of the group (valid only for an accessor of type group, this bit shall be ignored if set for accessor of type user);
- eAASAccessRight-UpdateCredentialPolicy: this right allows the update of the credential policy (valid only for an accessor of type user, this bit shall be ignored if set for accessor of type group);
- eAASAccessRight-UpdateCredentialStatus: this right allows the update of the status of credentials (valid only for an accessor of type user, this bit shall be ignored if set for accessor of type group).

Table 6.5: Applicability of rights

Command	eAASAccessRight-RequiresSecurePipe	eAASAccessRight-Create	eAASAccessRight-Delete	eAASAccessRight-Update	eAASAccessRight-UpdateACL	eAASAccessRight-UpdateGroup	eAASAccessRight-UpdateCredentialPolicy	eAASAccessRight-UpdateCredentialStatus
AAS-OP-GET-CAPABILITIES-Service-Command								
AAS-ADMIN-CREATE-ACCESSOR-Service-Command	•	•						
AAS-ADMIN-UPDATE-ACCESSOR-Service-Command	•			•	•	•	•	•
AAS-ADMIN-DELETE-ACCESSOR-Service-Command	•		•					
AAS-OP-ACCESS-SERVICE-Service-Command								
AAS-OP-AUTHENTICATE-ACCESSOR-Service-Command								
AAS-OP-GET-CHALLENGE-Service-Command								

6.13.4.6 Operations on an accessor

6.13.4.6.1 Creation

An accessor may be created by any other accessor that has the eAASAccessRight-Create right in the access control list of the accessor authentication service, as retrieved using AAS-OP-GET-CAPABILITIES-Service-Command described in clause 6.13.5.1.

The creation of an accessor is performed using the AAS-ADMIN-CREATE-ACCESSOR-Service-Command command of the accessor authentication service, as described in clause 6.13.5.2.

6.13.4.6.2 Deletion

An accessor may be deleted by another accessor if this has the eAASAccessRight-Delete right and is authenticated. Upon deletion of an accessor, the SSP shall remove all entries in the access control lists of its resources pointing to the accessor.

The deletion of an accessor is performed using the AAS-ADMIN-DELETE-ACCESSOR-Service-Command command of the accessor authentication service, as described in clause 6.13.5.4.

6.13.4.6.3 Update of the access control list

The access control list of an accessor may be updated by any accessor that has the eAASAccessRight-UpdateACL right and is successfully authenticated.

The update of the access control list of an accessor is performed using the AAS-ADMIN-UPDATE-ACCESSOR-Service-Command command of the accessor authentication service, as described in clause 6.13.5.3.

6.13.4.6.4 Update of the conditions and credentials

The accessor conditions and the corresponding credentials may be updated by any accessor that has the eAASAccessRight-Update and is authenticated. An accessor shall not be able to modify its own accessor conditions or credentials, if it is not explicitly listed in its own access control list.

The update of the access control list of an accessor is performed using the AAS-ADMIN-UPDATE-ACCESSOR-Service-Command command of the accessor authentication service, as described in clause 6.13.5.3.

6.13.4.6.5 Update of the group list

The list of members of an accessor group may be modified by any accessor that has the eAASAccessRight-UpdateGroup and is authenticated.

The update of the members of a group of accessors is performed using the AAS-ADMIN-UPDATE-ACCESSOR-Service-Command command of the accessor authentication service, as described in clause 6.13.5.3.

6.13.4.6.6 Update of the credential status and policy

The credential status of an accessor may be updated by any accessor that has the eAASAccessRight-UpdateCredentialStatus and is authenticated. The credentials policy of an accessor may be updated by any accessor that has the eAASAccessRight-UpdateCredentialPolicy and is authenticated.

The update of the credential status and the credential policy of an accessor is performed using the AAS-ADMIN-UPDATE-ACCESSOR-Service-Command command of the accessor authentication service, as described in clause 6.13.5.3.

6.13.4.7 Accessor credentials

An accessor may define its credentials and store them in a private storage assigned to the accessor authentication service.

The object AccessorCredentials is a collection of credentials and shall be represented with the following ASN.1 description.

```
-- ASN1START

/* Maximum size of each side of the entry panel for pattern */
maxEntryPanelDimension INTEGER ::= 10

/* Coordinate of point in pattern from the top-left corner, starting with index 1 */
PatternPoint ::= SEQUENCE
{
    x INTEGER (1.. maxEntryPanelDimension),  -- X coordinate
    y INTEGER (1.. maxEntryPanelDimension)  -- Y coordinate
}

/* Certification path */
TokenCredential ::= SEQUENCE
{
    aCertificatesAAS [0] Certificates,  -- Set of X.509 certificates of the accessor authentication
service
    aCertificateCIAAA [1] Certificates OPTIONAL --Certificates of the CI of the accessor
authentication application
}

AccessorCredentials ::= [PRIVATE 13] SEQUENCE
{
    -- Credentials of type PIN
    aPinNumericCredential [0] NumericString (SIZE(4..255)) OPTIONAL,  -- Numeric PIN
    aPinPasswordCredential [1] PrintableString (SIZE(4..255)) OPTIONAL,  -- Password
    aPinPatternCredential [2] SEQUENCE (SIZE(4..255)) OF PatternPoint OPTIONAL,  -- Graphical
pattern

    -- Credentials for the token based verification
    aTokenCredential [10] TokenCredential OPTIONAL,  -- Token credential

    -- Credentials of type host domain
    aHostDomainCredential [20] SET OF UUID  -- Set of SCL host domains

    -- Credentials of type biometric: for future usage
}

-- ASN1STOP
```

Where:

- aPinNumericCredential: numeric PIN. The string contains the sequence of digits of the PIN. The string shall not contain spaces;

- aPinPasswordCredential: password. The string contains a case-sensitive password;
- aPinPatternCredential: graphical pattern. A sequence of points creating a pattern. The size of the entry panel where the pattern is drawn is implementation dependent, but both the width and the height shall have at least 3 points and at most 10 points. The length of the pattern shall be between 4 and 255 points. The same point may appear more than once in the pattern, if allowed by the credential policy;
- aTokenCredential: a sequence embedding a set of X.509 certificates for the certification path of the accessor authentication service, a set of CI certificates for validating the certification path of the accessor authentication application. The AAS shall support at least a certification path of 4 certificates. The AAS certification path should not exceed 6 certificates;
- aHostDomainCredential: list of SCL host domains.

6.13.4.8 Accessor credential policy

An accessor may have policy for its credential. The policy will ensure management rules for the credentials. The policy rules should be stored in a private storage assigned to the accessor authentication service.

The object AccessorCredentialsPolicy is a set of policy for credential types. The policy (AccessorCredentialsPolicy) used for the AccessorCredentials are defined in the following parameters.

```
-- ASN1START

PinNumericPolicy ::= SEQUENCE
{
    aIsDisableForbidden BOOLEAN DEFAULT FALSE,  -- Disabling forbidden
    aMinSize INTEGER (4..255) DEFAULT 4,  -- Minimum size of PIN
    aMaxSize INTEGER (4..255) DEFAULT 255,  -- Maximum size of PIN
    aMaxAttempts INTEGER (0..255) DEFAULT 0  -- Maximum number of attempts
}

PinPasswordPolicy ::= SEQUENCE
{
    aMinSize INTEGER (4..255) DEFAULT 4,  -- Minimum length of password
    aMaxSize INTEGER (4..255) DEFAULT 255,  -- Maximum length of password
    aRequiresLowerCaseLetter BOOLEAN DEFAULT FALSE,  -- At least one lower case letter is required
    aRequiresUpperCaseLetter BOOLEAN DEFAULT FALSE,  -- At least one upper case letter is required
    aRequiresNumber BOOLEAN DEFAULT FALSE,  -- At least on numeric digit is required
    aRequiresSymbol BOOLEAN DEFAULT FALSE,  -- At least one special character is required
    aMaxAttempts INTEGER (0..255) DEFAULT 0  -- Maximum number of attempts
}

PinPatternPolicy ::= SEQUENCE
{
    aMinSize INTEGER (4..255) DEFAULT 4,  -- Minimum number of points in pattern
    aMaxSize INTEGER (4..255) DEFAULT 255,  -- Maximum number of points in pattern
    aEntryPanelMinSize INTEGER (3.. maxEntryPanelDimension) DEFAULT 3,
    aSamePointMultipleTimes BOOLEAN DEFAULT FALSE,  -- If a point can occur multiple times
    aMaxAttempts INTEGER (0..255) DEFAULT 0  -- Maximum number of attempts
}

AccessorCredentialsPolicy ::= SEQUENCE
{
    aPinNumericPolicy PinNumericPolicy OPTIONAL,  -- Numeric PIN policy
    aPinPasswordPolicy PinPasswordPolicy OPTIONAL,  -- Password policy
    aPinPatternPolicy PinPatternPolicy OPTIONAL  -- Graphical pattern policy
}

-- ASN1STOP
```

Where:

- aPinNumericPolicy: indicates the policy for the credentials of type numeric PIN. It includes:
  - aIsDisableForbidden: indicates if PIN can be disabled, if not present Pin can be disable.
  - aMinSize: minimum size for PIN.
  - aMaxSize: maximum size for PIN, if not present, maximum size is limited to 255.

  - aMaxAttempts: maximum number of attempts allowed for the PIN. The value 0 indicates that an infinite number of attempts is allowed.

- aPinPasswordPolicy: indicates the policy for the credentials of type password. It includes:
  - aMinSize: minimum size for password.
  - aMaxSize: maximum size for password, if not present, maximum size is limited to 255.
  - aRequiresLowerCaseLetter: indicates if the password shall contain at least one lower case letter.
  - aRequiresUpperCaseLetter: indicates if the password shall contain at least one upper case letter.
  - aRequiresNumber: indicates if the password shall contain at least one numeric digit (i.e. between '0' and '9').
  - aRequiresSymbol: indicates if the password shall contain at least one symbol that is not a letter or a number.
  - aMaxAttempts: maximum number of attempts allowed for the password. The value 0 indicates that an infinite number of attempts is allowed.

- aPinPatternPolicy:
  - aMinSize: minimum number of points in the pattern.
  - aMaxSize: maximum number of points in the pattern, if not present, maximum size is limited to 255.
  - aEntryPanelMinSize: minimum size of the width and the height of the pattern. The entry panel of the pattern may be a rectangular, as far as both sides have a size that is at least equal to aEntryPanelMinSize.
  - aSamePointMultipleTimes: indicates if the same point can appear multiple times in the pattern.
  - aMaxAttempts: maximum number of attempts allowed for the pattern. The value 0 indicates that an infinite number of attempts is allowed.

NOTE 1: The credential of type host domain is not intended to be changed by the accessor and therefore has no defined policy.

NOTE 2: The token based credential has no policy.

6.13.4.9 Accessor credential status

Credentials may have their own status. The credential status should be stored in a private storage assigned to the accessor authentication service.

The status (AccessorCredentialsStatus) used for the credential is defined in the following parameters.

```
-- ASN1START

AccessorCommonCredentialStatus ::= SEQUENCE
{
    aIsDisabled BOOLEAN DEFAULT TRUE,  -- indicates if credential is disabled
    aRemainingAttempts INTEGER (0..255) OPTIONAL  -- remaining number of attempts
}

PinNumericCredentialStatus ::= SEQUENCE
{
    aCommonStatus AccessorCommonCredentialStatus
}

PinPasswordCredentialStatus ::= SEQUENCE
{
    aCommonStatus AccessorCommonCredentialStatus
}

PinPatternCredentialStatus ::= SEQUENCE
{
    aCommonStatus AccessorCommonCredentialStatus
}
```

```
AccessorCredentialsStatus ::= SEQUENCE
{
    aPinNumericStatus PinNumericCredentialStatus OPTIONAL,
    aPinPasswordStatus PinPasswordCredentialStatus OPTIONAL,
    aPinPatternStatus PinPatternCredentialStatus OPTIONAL
}

-- ASN1STOP
```

Where:

- aIsDisabled: indicates if the related credential is disabled (authentication not needed);
- aRemainingAttempts: indicates the number of attempts remaining. 0 indicates that the credential is no more useable, no presence indicates that no maximum number of retry is defined.

NOTE 1: The credential of type host domain has no status.

NOTE 2: The token based credential has no status.

6.13.5 Primitives

6.13.5.1 AAS-OP-GET-CAPABILITIES-Service-Command

With the command AAS-OP-GET-CAPABILITIES-Service-Command, the authentication application may retrieve the capabilities of the accessor authentication service.

```
-- ASN1START

AAS-GET-CAPABILITIES-Type ::= ENUMERATED
{
    eGlobalAuthenticationService (0),  -- retrieve user accessors available in the SSP host
    eAccessorStatus (1)  -- retrieve status related to the accessor authentication service gate
}

AAS-OP-GET-CAPABILITIES-Service-Command ::= [PRIVATE 16] SEQUENCE
{
    aRequestType AAS-GET-CAPABILITIES-Type
}

-- ASN1STOP
```

This command has the following parameters:

- eGlobalAuthenticationService: request all user accessors available in the SSP host;
- eAccessorStatus: request information related to the accessor using the authentication service gate.

NOTE: aRequestType set to aGlobalAuthenticationService is useful for the anonymous authentication service.

When the request is successful then accessor authentication service gate shall include eAAS-OK in the response.

```
-- ASN1START

AAS-OP-GET-CAPABILITIES-Service-Response-Parameter ::= CHOICE
{
    aGlobalAuthenticationService SEQUENCE  -- for aRequestType set to eGlobalAuthenticationService
    {
        aAASVersion VersionType,  -- release of the AAS service
        aAccessorList SET OF Accessor,  -- List of accessors
        aACL AccessControlList  -- Access control list
    },
    aAccessorStatus SEQUENCE  -- for aRequestType set to eAccessorStatus
    {
        aIsAuthenticated BOOLEAN,  -- indicates if the accessor is authenticated
        aAccessorConditions AccessorConditions,  -- accessor conditions
        aAccessorCredentialsStatus AccessorCredentialsStatus OPTIONAL,  -- status of credentials of
the accessor
        aAccessorCredentialsPolicy AccessorCredentialsPolicy OPTIONAL  -- policies for the
credentials of the accessor
    }
```

```
}

AAS-OP-GET-CAPABILITIES-Service-Response ::= [PRIVATE 16] SEQUENCE
{
    aAAS-Service-Response AAS-Service-Response DEFAULT eAAS-OK,
    aParameter AAS-OP-GET-CAPABILITIES-Service-Response-Parameter OPTIONAL
}

-- ASN1STOP
```

Where:

- aGlobalAuthenticationService: contains the response in case of aRequestType set to eGlobalAuthenticationService. It contains:
  - aAASVersion: major and minor release version supported by the accessor authentication service;
  - aAccessorList: list of all the accessors available in the SSP host;
  - aACL: access control list of the accessor authentication service.
- aAccessorStatus: contains the response in case of aRequestType set to eAccessorStatus. It contains:
  - aIsAuthenticated: indicates if the accessor is authenticated in this accessor authentication service;
  - aAccessorConditions: accessor conditions to be authenticated;
  - aAccessorCredentialsStatus: status of the credentials in this accessor authentication service;
  - aAccessorCredentialsPolicy: policies for the credentials in this accessor authentication service.

The access control list of the accessor authentication service indicates which accessors can create new accessors using the eAASAccessRight-Create right. It also indicates who can update the access control list of the accessor authentication service, using the eAASAccessRight-UpdateACL right.

6.13.5.2 AAS-ADMIN-CREATE-ACCESSOR-Service-Command

With the command AAS-ADMIN-CREATE-ACCESSOR-Service-Command, an accessor may create another accessor and store its initial credentials. The accessor may create another accessor if the accessor authentication service grants the eAASAccessRight-Create right to this accessor.

```
-- ASN1START

AAS-ADMIN-CREATE-ACCESSOR-Service-Command ::= [PRIVATE 17] SEQUENCE
{
    aAccessor Accessor,  -- Accessor to be created
    aCredential AccessorCredentials OPTIONAL,  -- Credentials for the accessor
    aCredentialsPolicy AccessorCredentialsPolicy OPTIONAL,  -- Policy for the provided accessors
    aCredentialsStatus AccessorCredentialsStatus OPTIONAL  -- Status of credentials
}

-- ASN1STOP
```

This command has the following parameters:

- aAccessor: the definition of the accessor to be created;
- aCredential: initial credentials of the accessor to be created (present only during creation of an accessor of type user). aCredentialsPolicy is present and the credentials are not conformant with the policies described in that, then the error eAAS- POLICY-RULES-VIOLATIONS shall be returned;
- aCredentialsPolicy: policy for the credentials of the accessor to be created (shall not be present if aAccessorConditions is not present);
- aCredentialsStatus: initial status of the credentials of the accessor to be created (shall not be present if aAccessorConditions is not present).

When the request is successful, then accessor authentication service gate shall include eAAS-OK in the response.

```
-- ASN1START

AAS-ADMIN-CREATE-ACCESSOR-Service-Response ::= [PRIVATE 17] SEQUENCE
{
    aAAS-Service-Response AAS-Service-Response DEFAULT eAAS-OK
}

-- ASN1STOP
```

6.13.5.3 AAS-ADMIN-UPDATE-ACCESSOR-Service-Command

With the command AAS-ADMIN-UPDATE-ACCESSOR-Service-Command, an accessor may update the credentials stored within a private storage of another accessor or of itself. The accessor may update:

- The conditions and credentials if it has the eAASAccessRight-Update right. If credential policies are present in the command or previously in the accessor and the credentials are not conformant with the policies, then the error eAAS- POLICY-RULES-VIOLATIONS shall be returned.
- The access control list if it has the eAASAccessRight-UpdateACL right.
- The members of the group if it has the eAASAccessRight-UpdateGroup right.
- The credential policies if it has the eAASAccessRight-UpdateCredentialPolicy right.
- The credential status if it has eAASAccessRight-UpdateCredentialStatus right.

The command shall be rejected with eAAS-ACL-RULES-VIOLATIONS if it contains any element for which the accessor does not have the rights to update.

```
-- ASN1START

AAS-ADMIN-UPDATE-ACCESSOR-Service-Command ::= [PRIVATE 18] SEQUENCE
{
    aAccessorIdentity AccessorIdentity,  -- Identity of the accessor
    aMembersOfGroup SET OF AccessorIdentity OPTIONAL,  -- Members of the group
    aACL AccessControlList OPTIONAL,  -- Access control list
    aSetAccessorConditions AccessorConditions OPTIONAL,  -- Conditions to be set
    aRemoveAccessorConditions AccessorConditions OPTIONAL,  -- Conditions to be removed
    aSetCredential AccessorCredentials OPTIONAL,  -- Credentials to be set
    aRemoveCredential AccessorConditions OPTIONAL,  -- List of credentials to be removed
    aCredentialsPolicy AccessorCredentialsPolicy OPTIONAL,  -- Credential policy
    aCredentialsStatus AccessorCredentialsStatus OPTIONAL  -- Status of credentials
}

-- ASN1STOP
```

This command has the following parameters:

- aAccessor-Identity: the accessor identity of the accessor to be updated;
- aMembersOfGroup: the updated list of the accessors in a group;
- aACL: the updated access control list for the accessor;
- aSetAccessorConditions: the access conditions that need to be added;
- aRemoveAccessorConditions: the access conditions that need to be removed. The removal of an access condition does not imply the deletion of the corresponding credentials or the change of the status;
- aSetCredential: the new values of credentials to be updated;
- aRemoveCredential: the list of credentials that need to be removed from the SSP. The status of all credentials included in this list shall be disabled;
- aCredentialsPolicy: the updated credential policy. The values of credential policies that are not included in the command shall not be modified;
- aCredentialsStatus: the updated credential status. The status values of credentials that are not included in the command shall not be modified.

When the request is successful then accessor authentication service gate shall include eAAS-OK in the response.

```
-- ASN1START

AAS-ADMIN-UPDATE-ACCESSOR-Service-Response ::= [PRIVATE 18] SEQUENCE
{
    aAAS-Service-Response AAS-Service-Response DEFAULT eAAS-OK
}

-- ASN1STOP
```

6.13.5.4 AAS-ADMIN-DELETE-ACCESSOR-Service-Command

With the command AAS-ADMIN-DELETE-ACCESSOR-Service-Command, the accessor authentication application may delete another accessor. The accessor may delete another accessor if it grants the eAASAccessRight-Delete to this latter accessor.

```
-- ASN1START

AAS-ADMIN-DELETE-ACCESSOR-Service-Command ::= [PRIVATE 19] SEQUENCE
{
    aAccessorIdentity AccessorIdentity  -- Identity of the accessor to delete
}

-- ASN1STOP
```

This command has the following parameters:

- aAccessorIdentity: the identity of the deleted accessor.

When the request is successful then accessor authentication service gate shall include eAAS-OK in the response.

```
-- ASN1START

AAS-ADMIN-DELETE-ACCESSOR-Service-Response ::= [PRIVATE 19] SEQUENCE
{
    aAAS-Service-Response AAS-Service-Response DEFAULT eAAS-OK
}

-- ASN1STOP
```

6.13.5.5 AAS-OP-AUTHENTICATE-ACCESSOR-Service-Command

With the command AAS-OP-AUTHENTICATE-ACCESSOR-Service-Command, the accessor authentication application provides the credentials for performing the authentication.

```
-- ASN1START

AccessorTokenCredential ::= SEQUENCE
{
    aAuthenticationToken AuthenticationToken,  -- The authentication token generated by the AAA
    aTokenCertificationPath [20] Certificates  -- the certification path for verifying the
authentication token
}

AAS-OP-AUTHENTICATE-ACCESSOR-Service-Command ::= [PRIVATE 22] SEQUENCE
{
    aCredential CHOICE
    {
        aPinNumericCredential [10] NumericString,  -- Numeric PIN
        aPinPasswordCredential [11] PrintableString,  -- Password
        aPinPatternCredential [12] SEQUENCE (SIZE(4..255)) OF PatternPoint,  -- Graphical pattern
        aAccessorTokenCredential [20] AccessorTokenCredential,  -- Authentication token credential
        aHostDomainCredential [30] NULL
    }
}

-- ASN1STOP
```

This command has the following parameters:

- aCredential: the additional credentials as:
  - aPinNumericCredential: numeric PIN credential;
  - aPinPasswordCredential: password credential;
  - aPinPatternCredential: pattern credential;
  - aAccessorTokenCredential: authentication token credential containing:
    - aAuthenticationToken: the authentication token as defined in the clause C.2.2;
    - aAccessorTokenCertificationPath: the certification path which end entity certificate signs the authentication token generated by the accessor authentication application as defined in clause C.3;
  - aHostDomainCredential: the accessor is authenticated if the command is issued by an host inside an host domain which has its UUID listed in credentials of type host domain.

NOTE: The credential of type host domain can be verified using the response of the Link service gate command LINK_GET_HOST_LIST defined in GlobalPlatform VPP - Network Protocol [13], clause 5.3.2.2.

When the request is successful then accessor authentication service gate shall include eAAS-OK in the response.

```
-- ASN1START

AAS-OP-AUTHENTICATE-ACCESSOR-Service-Response-Parameter ::= SEQUENCE
{
    aCredentialsStatus AccessorCredentialsStatus OPTIONAL,  -- Status of credentials after the
command
    aAuthenticationToken AuthenticationToken OPTIONAL  -- AuthenticationToken generated by the
accessor authentication service
}

AAS-OP-AUTHENTICATE-ACCESSOR-Service-Response ::= [PRIVATE 22] SEQUENCE
{
    aAAS-Service-Response AAS-Service-Response DEFAULT eAAS-OK,
    aParameter AAS-OP-AUTHENTICATE-ACCESSOR-Service-Response-Parameter OPTIONAL
}

-- ASN1STOP
```

Where:

- aCredentialsStatus: the status of the credentials after the execution of the request;
- aAuthenticationToken: authentication token as defined in clause C.2.2.

6.13.5.6 AAS-OP-ACCESS-SERVICE-Service-Command

With the command AAS-OP-ACCESS-SERVICE-Service-Command, an accessor provides the parameter for initiating a pipe session between a service and application gate identified by aServiceIdentifier identifier. This allows access to resources that are protected by an access control list (e.g. SSP files in the SSP file system).

NOTE 1: It is implementation dependant to allow or reject access to services for each accessor, regardless of its authentication status.

The pipe session created after the execution of this command may use the secure pipe, depending on the parameter of that command. The error code eAAS-E-NOK is returned if the usage of secure pipe is not requested by the accessor in the command, but it is required by the service.

NOTE 2: It is implementation dependant if the service requires the usage of secure pipes.

This command shall be executed only after successful authentication of the accessor, or the SSP shall reject it with the value eAAS-NOT-AUTHENTICATED.

```
-- ASN1START

AAS-OP-ACCESS-SERVICE-Service-Command ::= [PRIVATE 20] SEQUENCE
```

```
{
    aServiceIdentifier UUID,  -- Identifier of the service
    aUseSecurePipe BOOLEAN DEFAULT FALSE  -- Indication of secure pipe is requested
}

-- ASN1STOP
```

This command has the following parameters:

- aServiceIdentifier: identifier of a service in the SSP host;
- aUseSecurePipe: indicates if a secure pipe is required to access the service.

To initiate a pipe session between an accessor authentication application gate and its accessor authentication service gate, the accessor authentication application shall issue an AAS-OP-ACCESS-SERVICE-Service-Command with aServiceIdentifier set to the accessor identity (identical to the accessor authentication service gate identifier as defined in clause 10.9.1).

In addition, the accessor authentication application can request a secure pipe with the accessor authentication service (e.g. for administrative purposes) setting aUseSecurePipe set to TRUE.

When the request is successful then accessor authentication service gate shall include eAAS-OK in the response.

```
-- ASN1START

AAS-OP-ACCESS-SERVICE-Service-Response-Parameter ::= SEQUENCE
{
    aGateIdentifier UUID  -- Identifier of the service gate
}

AAS-OP-ACCESS-SERVICE-Service-Response ::= [PRIVATE 20] SEQUENCE
{
    aAAS-Service-Response AAS-Service-Response DEFAULT eAAS-OK,
    aParameter AAS-OP-ACCESS-SERVICE-Service-Response-Parameter OPTIONAL
}

-- ASN1STOP
```

Where:

- aGateIdentifier: identifier of the service gate dynamically allocated by the accessor authentication service.

6.13.5.7 AAS-OP-GET-CHALLENGE-Service-Command

With the command AAS-OP-GET-CHALLENGE-Service-Command, an accessor provides the parameter for authenticating an accessor using token mechanism.

```
-- ASN1START

AAS-OP-GET-CHALLENGE-Service-Command ::= [PRIVATE 21] SEQUENCE
{
}

-- ASN1STOP
```

This command has no parameters.

When the accessor authentication request is successful then the accessor authentication service gate shall include eANY-OK in the response.

```
-- ASN1START

AAS-OP-GET-CHALLENGE-Service-Response-Parameter ::= SEQUENCE
{
    aChallenge OCTET STRING (SIZE (16)),
    aCertificates Certificates  -- Certificates of the accessor authentication service
}

AAS-OP-GET-CHALLENGE-Service-Response ::= [PRIVATE 21] SEQUENCE
{
    aAAS-Service-Response AAS-Service-Response DEFAULT eAAS-OK,
    aParameter AAS-OP-GET-CHALLENGE-Service-Response-Parameter OPTIONAL
```

```
}
-- ASN1STOP
```

Where:

- aChallenge: challenge used for performing the mutual authentication between the accessor authentication service and the accessor authentication application. The challenge should be a random number of at least 128 bits. The way the challenge is generated is implementation dependant.

6.13.6 Response code

6.13.6.1 Overview

The accessor authentication service provides the following response codes to accessor authentication service primitives.

```
-- ASN1START

AAS-Service-Response ::= ENUMERATED
{
    eAAS-OK (0),  -- Operation successful
    eAAS-E-CMD-PAR-UNKNOWN (2),  -- Unknown parameters used for an operation
    eAAS-E-NOK (3),  -- Operation failed
    eAAS-ACL-RULES-VIOLATIONS (14),  -- The operation violates the ACL conditions
    eAAS-NOT-AUTHENTICATED (15),  -- The accessor is not authenticated
    eAAS-POLICY-RULES-VIOLATIONS (16)  -- The operation violates the credentials policy
}

-- ASN1STOP
```

Where:

- eAAS-OK: The command is completed successfully;
- eAAS-E-CMD-PAR-UNKNOWN: An unknown parameter is used for an operation;
- eAAS-E-NOK: The operation failed;
- eAAS-ACL-RULES-VIOLATION: The operation of the administration violates the ACL conditions associated to an accessor;
- eAAS-NOT-AUTHENTICATED: The accessor is not authenticated;
- eAAS-POLICY-RULES-VIOLATION: The operation violates the credential policy.

6.13.6.2 Response codes to accessor authentication service commands

Table 6.6 shows the possible response codes returned for each accessor authentication service command.

Table 6.6: Accessor authentication service commands/responses

Command	eAAS-OK	eAAS-E-CMD-PAR-UNKNOWN	eAAS-E-NOK	eAAS-ACL-RULES-VIOLATIONS	eAAS-NOT-AUTHENTICATED	eAAS-POLICY-RULES-VIOLATIONS
aAAS-OP-GET-CAPABILITIES-Service-Command	•		•			
aAAS-ADMIN-CREATE-ACCESSOR-Service-Command	•	•	•	•		•
aAAS-ADMIN-UPDATE-ACCESSOR-Service-Command	•	•	•	•		•
aAAS-ADMIN-DELETE-ACCESSOR-Service-Command	•	•	•	•		
aAAS-OP-ACCESS-SERVICE-Service-Command	•	•	•		•	
aAAS-OP-AUTHENTICATE-ACCESSOR-Service-Command	•	•	•			

7 Physical interfaces

7.1 Overview

The SSP may have multiple physical interfaces of the same type or of different types. These interfaces may connect to the same or to different end-points in the terminal (e.g. baseband, NFC controller).

When the SSP contains two or more interfaces, each of them is completely independent, both electrically and logically. This implies that signalling on a contact assigned to one interface shall not affect the state of other contacts assigned to another interface. Similarly, an operation performed on one interface shall not alter the logical state of any other interface.

If two or more interfaces are activated, the order of activation and deactivation is decided by the terminal.

7.2 Reset

There are three types of reset of the physical interface:

- **Reset with dedicated line:** this reset requires the presence of a dedicated line in the physical interface that indicates the reset (e.g. the RST line on the ISO/IEC 7816-3 [3] physical interface).
- **Logical reset:** this reset is performed sending a command over the physical interface to indicate the reset to the SSP (e.g. RSET frame in SHDLC or USB Reset). This command may be sent at the data link layer or any layer above.
- **Hard reset:** this reset is performed removing the power, if present, provided by the physical interface to the SSP (e.g. cold reset for the ISO/IEC 7816-3 [3] physical interface).

Each physical interface shall support at least one reset type.

If the power provided by one physical interface is the only source of power of the SSP, a hard reset of that physical interface causes the reset of the entire SSP. In all other cases, a reset performed on any interface shall not interfere with the operations on the other interfaces, or with the operational state of the SSP itself.

7.3 ISO/IEC 7816 interface

7.3.1 Electrical specifications

7.3.1.1 Electrical specifications of the interface

The provisions of ETSI TS 102 221 [1], clause 5 shall apply with the following exceptions:

- The SSP may support a clock up to 20 MHz for the ISO/IEC 7816-3 [3] physical interface.
- The SSP shall use an internal clock for the processing, when this is mandated by the SSP class. The SSP may use an internal clock in all other cases.

7.3.1.2 Contacts

The provisions of ETSI TS 102 221 [1], clause 4.5 shall apply with the following exception:

- References to the usage of contacts C4 and C8 for the Inter-Chip USB interface.

7.3.2 Initial communication establishment procedures

7.3.2.1 SSP interface activation and deactivation

The provisions of ETSI TS 102 221 [1], clause 6.1 shall apply with the same exceptions described in clause 7.3.1.2 of the present document.

7.3.2.2 Supply voltage switching

The provisions of ETSI TS 102 221 [1], clauses 6.2.0, 6.2.1 and 6.2.2 shall apply.

The maximum power consumption of the SSP after ATR shall be restricted to the minimum power supply values indicated in ETSI TS 102 221 [1], table 6.4, until a different value is negotiated using the SSP capability exchange procedure, described in clause 6.4.2 in the present document.

7.3.2.3 Answer To Reset content

The ATR shall be the first string of bytes sent from the SSP to the terminal after a reset has been performed. The ATR is defined in ISO/IEC 7816-3 [3].

The historical bytes indicate to the external world how to use the SSP. The information carried by the historical bytes shall follow ISO/IEC 7816-4 [4].

Additionally, the provisions of ETSI TS 102 221 [1], clauses 6.3.2 and 6.3.3 shall apply.

During ATR handling properties sent with the ATR that are not related to the ISO/IEC 7816-3 [3] physical interface shall be ignored by the terminal. Additional properties available in the ATR are negotiated between the SSP and the terminal during the capability exchange procedure described in clause 6.4.2 of the present document.

7.3.2.4 PPS procedure

The provisions of ETSI TS 102 221 [1], clause 6.4 shall apply.

7.3.2.5 Reset procedure

The provisions of ETSI TS 102 221 [1], clause 6.5 shall apply.

The warm reset is a reset with dedicated line, as described in clause 7.2. The cold reset is a hard reset, as described in clause 7.2.

7.3.2.6 Clock stop mode

The provisions of ETSI TS 102 221 [1], clause 6.6 shall apply.

7.3.2.7 Bit/character duration and sampling time

The provisions of ETSI TS 102 221 [1], clause 6.7 shall apply.

7.3.2.8 Error handling

The provisions of ETSI TS 102 221 [1], clause 6.8 shall apply.

7.3.3 Data link protocols

7.3.3.1 Overview

The provisions of ETSI TS 102 221 [1], clause 7.0 shall apply, with the exceptions listed below.

Only the protocol T=1 is mandatory for the terminal.

The SSP shall support the protocol T=1.

7.3.3.2 Character frame

The provisions of ETSI TS 102 221 [1], clause 7.2.1 shall apply.

7.3.3.3 Protocol T=1

The provisions of ETSI TS 102 221 [1], clause 7.2.3 shall apply.

7.4 SPI interface

The provisions of ETSI TS 103 713 [32] shall apply.

7.5 I2C interface

The usage of the I2C interface for SSP is for future study.

7.6 SWP interface

The provisions of ETSI TS 102 613 [5] shall apply with the following exceptions:

- Contacts names C1 (Vcc), C5 (Gnd), C6 (SWIO), C2 (RSET), C3 (CLK), C7 (IO), C4 (IC_DP), C8 (IC_DM) are irrelevant.
- The provisions of ETSI TS 102 613 [5], clauses 5.2, 5.3, 5.4, 6.1, 6.2.1 and 7.1.1 are optional, as the SSP needs not to support an ISO/IEC 7816-3 [3] interface nor form factor defined in ETSI TS 102 221 [1].
- The SSP power modes and interface activation sequences described in ETSI TS 102 613 [5] clauses 6.2.2.0, 6.2.2.3, 6.2.3.4, 6.2.6, 7.1.2, 8.3.2 and 8.4 are SSP class dependant.

7.7 USB interface

The usage of the USB interface for SSP is for future study.

7.8 Proprietary interface

In addition to standardized physical interfaces, some SSP classes may have a proprietary interface.

The SSP shall be able to trigger communication over this proprietary physical interface.

Further definition of this physical interface is outside the scope of the present document.

8 SSP Common Layer (SCL)

8.1 Introduction

The SSP may support the SSP Common Layer (SCL) implementation comprised of optional network, transport and session layers.

The SCL is the common protocol layer in the protocol stack of the SSP. The SCL is independent of any of its optional underlying and upper communication layers. The SCL is supported by several underlying communication layers defined in optional SSP classes.

SCL shall be implemented using VNP, as specified in the GlobalPlatform VPP - Network Protocol [13]. The relevant sections and exceptions are described in the following clauses.

The word "VNP" used in GlobalPlatform VPP - Network Protocol [13] shall be replaced with "SCL" as needed.

The word "TRE" used in GlobalPlatform VPP - Network Protocol [13] shall be replaced with "SSP" as needed.

8.2 SCL network

The provisions of GlobalPlatform VPP - Network Protocol [13], clause 3 shall apply, with the exception listed below:

- One of the end points of any dynamic pipe shall be in the SSP host domain or in the network controller host.

NOTE: The usage of pipes between two hosts outside the SSP host domain is for further study.

The SSP host domain identifier is the TRE Host domain identifier defined in GlobalPlatform VPP - Network Protocol [13], clause 3.3.3, i.e. "urn:globalplatform.org:TRE".

Table 8.1 defines the URN for the additional gates defined in the present document, other than the core gates referenced from GlobalPlatform VPP - Network Protocol [13], clause 3.9. All UUIDs are calculated using the version 5 of the UUID as specified in IETF RFC 4122 [28], using the domain name system namespace.

Table 8.1: Gates URN

Gate	NID	NSS	Pre-computed identifier
UICC APDU service gate	urn:etsi.org	SSP:HCI:UICC-APDU	B9A3405D-1017-59AD-B959-2689DBEFF652
SSP FS control service gate	urn:etsi.org	SSP:ASN.1:FS_Control	366BD642-D7DE-584A-BD3B-A3DCE29FC075
TCP control service gate	urn:etsi.org	REE:ASN.1:TCP_Control	F3DBA7CC-3551-5170-BC79-8BED75AA37AA
		MBM:ASN.1:TCP_Control	8EC8017B-B734-533D-AAA0-FF6D693EA85C
		TEE:ASN.1:TCP_Control	727A3D1D-B52D-50CB-B20B-BCA7E9EE25CF
UDP service gate	urn:etsi.org	REE:ASN.1:UDP_Service	34E27B41-3B9A-59A9-9BA4-2B91292DAFEA
		TEE:ASN.1:UDP_Service	0091E79A-9A10-53D9-88AF-187DF566713B
		MBM:ASN.1:UDP_Service	ADCE4843-A058-50F2-A98D-5D3C334504B0
CRON service gate	urn:etsi.org	REE:ASN.1:CRON_Service	D67ABDB2-91AC-5B2E-8DF9-A53591E987C0
		TEE:ASN.1:CRON_Service	E5C6D5E1-6376-5B2D-A158-F11B5E7BA7AE
		MBM:ASN.1:CRON_Service	51FE5F0F-3BAA-506B-8CB5-AFD7562268E8
HCI gate	urn:etsi.org	REE:HCP:HCI_Service	213CA645-9A22-5C5D-B340-60212840015B
CLT gate	urn:etsi.org	REE:CLT:CLT_Service	0164A522-9555-57F6-BE65-AFD61C01D93A
CAT service gate	urn:etsi.org	REE:HCI:CAT_Service	FF00453F-B0D5-59CE-B0D4-3AE178432F73
		MBM:HCI:CAT_Service	3D16542C-691F-53DB-A62A-B5AEF296159B

The data acknowledgement mechanism (EVT_ADM_RECEIVED) and the credit-based data flow control (EVT_ADM_CREDIT) described in clause 8.5.3 shall not apply unless otherwise specified in the gate description.

8.3 Protocol layers

8.3.1 Overview

The provisions of GlobalPlatform VPP - Network Protocol [13], clauses 4.1 and 4.2 shall apply, with the exception listed below:

- the VNP MTU value of the link service gate registry shall be 20 bytes or greater (as defined in clause 8.4.3.1.2).

For proper operation, the protocol stack underlying the SCL shall provide a means for managing the underlying flow control.

There shall be an optional means for controlling (e.g. activating, deactivating) the underlying protocols and for getting the notifications from an underlying protocol (e.g. activation/deactivation of the interface by the terminal).

8.3.2 Network layer

The provisions of GlobalPlatform VPP - Network Protocol [13], clause 4.3 shall apply.

8.3.3 Transport layer

The provisions of GlobalPlatform VPP - Network Protocol [13], clause 4.4 shall apply.

8.3.4 Session layer

The session layer is mapped on the pipe session as defined in GlobalPlatform VPP - Network Protocol [13], clause 3.10.2.

8.4 SCL core services

8.4.1 Overview

The provisions of GlobalPlatform VPP - Network Protocol [13], clause 5.1 shall apply.

8.4.2 Common core features

The provisions of GlobalPlatform VPP - Network Protocol [13], clause 5.2 shall apply.

8.4.3 Link gate

8.4.3.1 Link service gate

8.4.3.1.1 General description

The provisions of GlobalPlatform VPP - Network Protocol [13], clause 5.3 shall apply, with the following additions.

NOTE: The credit-based data flow control mechanism and the data acknowledgement mechanism are not used in the link service gate.

8.4.3.1.2 Additional registry entries

The following entry defined in GlobalPlatform VPP - Network Protocol [13], clause 5.3.5 of the link service gate registry is modified according to the table 8.2.

Table 8.2: Additional registry entries in the link service gate

Type	Identifier	Parameter	Access Right	Comment	Length	Default
Mandatory	'05'	VNP_MTU	RO	MTU value of the SSP	2	20

8.4.3.1.3 SSP_MTU

It contains the value in bytes of the MTU of the link layer between the SCL router and the SSP. The entry shall have a value equal to or greater than 20.

An SCL host shall be able to send an SCL packet to the SSP without fragmentation, if the size of the SCL packet is less or equal to the value provided in this registry. The SCL router shall be able to forward to the SSP any SCL packet with a size equal or smaller than the value provided in this registry, without any further fragmentation.

8.4.3.2 Link application gate

The provisions of GlobalPlatform VPP - Network Protocol [13], clause 5.4 shall apply.

NOTE: The credit-based data flow control mechanism and the data acknowledgement mechanism are not used in the link application gate.

8.4.4 Administration gate

8.4.4.1 Administration service gate

The provisions of GlobalPlatform VPP - Network Protocol [13], clause 5.5 shall apply, with the exception listed in clause 8.5.3.

NOTE: The credit-based data flow control mechanism and the data acknowledgement mechanism are not used in the administration service gate for its own usage (e.g. the reception of an event EVT_ADM_BIND does not trigger the emission of EVT_ADM_RECEIVED nor EVT_ADM_CREDIT).

8.4.4.2 Administration application gate

The provisions of GlobalPlatform VPP - Network Protocol [13], clause 5.6 shall apply.

NOTE: The credit-based data flow control mechanism and the data acknowledgement mechanism are not used in the administration service gate for its own usage (e.g. the reception of an event EVT_ADM_BIND does not trigger the emission of EVT_ADM_RECEIVED nor EVT_ADM_CREDIT).

8.4.5 Identity gate

8.4.5.1 Identity service gate

8.4.5.1.1 General description

The provisions of GlobalPlatform VPP - Network Protocol [13], clause 5.7 shall apply, with the following additions.

The identity service gate should not list gates that are created dynamically as a result of an operation on a service of the SSP (e.g. the file system data service gates described in clause 10.3.5, the TCP data service gates described in clause 10.4.6 or services initiated using the accessor authentication service described in clause 10.9).

NOTE: The identity service gate may return a different list of gates to each host.

The credit-based data flow control mechanism and the data acknowledgement mechanism shall not be used in the identity service gate.

8.4.5.1.2 Additional registry entries

The following additional entries defined in GlobalPlatform VPP - Network Protocol [13], clause 5.7.5 of the identity service gate are modified according to the table 8.3.

Table 8.3: Additional registry entries in the identity service gate

Type	Identifier	Parameter	Access Right	Comment	Length	Default
Optional	'06'	NB_PIPE_SESSION_GATE_LIST	RO	Array of Number, between 1 and 255 of Pipe Sessions per Gate Identifier ordered as in the GATE_LIST parameter	Variable	1
Mandatory	'80'	CAPABILITY_EXCHANGE	RO	Contains the capabilities of the Host	Variable	-
Optional	'81'	GATE_URN_LIST	RO	URN-Description-List object	Variable	-

8.4.5.1.3 CAPABILITY_EXCHANGE

The capabilities of the host are coded with ASN.1 syntax as defined in:

- clause 6.4.2.4, for SCL hosts outside the SSP host domain;
- clause 6.4.2.5, for SCL hosts inside the SSP host domain.

8.4.5.1.4 GATE_URN_LIST

The GATE_URN_LIST provides an ASN.1 object containing an array of URNs according to IETF RFC 8141 [29] used to compute gate identifiers and the UUID resulting from the computation. The Identity Application Gate may use this entry for service discovery. The GATE_URN_LIST may have less, but shall not have more URNs than UUIDs listed in the GATE_LIST entry. All URNs provided in the GATE_URN_LIST shall be present in the GATE_LIST.

```
-- ASN1START

/* Identity Gate */
URN-Description ::= SEQUENCE
{
    aURN-Readable PrintableString,  -- URN string used to compute UUID
    aURN-UUID UUID  -- UUID of the gate computed from aURN-Readable present in GATE_LIST
}
URN-Description-List ::= SEQUENCE OF URN-Description
```

```
-- ASN1STOP
```

8.4.5.2 Identity application gate

The provisions of GlobalPlatform VPP - Network Protocol [13], clause 5.8 shall apply.

In addition, the additional entries in the gate registry defined in clause 8.4.5.1.2 shall apply.

The credit-based data flow control mechanism and the data acknowledgement mechanism shall not be used in the identity application gate.

8.4.6 Loopback gate

8.4.6.1 Loopback service gate

The provisions of GlobalPlatform VPP - Network Protocol [13], clause 5.9 shall apply with the following exception:

- The credit-based data flow control mechanism and the data acknowledgement mechanism are optional for the loopback service gate in the SSP host and their support is indicated in the LOOPBACK_CAPABILITIES registry, as indicated in clause 8.4.6.3.

8.4.6.2 Loopback application gate

The provisions of GlobalPlatform VPP - Network Protocol [13], clause 5.10 shall apply with the following exception:

- The credit-based data flow control mechanism and the data acknowledgement mechanism shall be enabled in the loopback application gate outside the SSP host according to the support indicated by the loopback service gate in the LOOPBACK_CAPABILITIES registry, as indicated in clause 8.4.6.3.

8.4.6.3 Registry

The following registry entry is defined for the loopback service gate in the SSP host.

Table 8.4: Registry entry in the loopback gate

Type	Identifier	Parameter	Access Right	Comment	Length	Default
Optional	'80'	LOOPBACK_CAPABILITIES	R	Capabilities of the loopback service gate	1	'00'

The capabilities of the loopback service gate are defined in table 8.5.

Table 8.5: Coding of the loopback service gate capabilities

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
-	-	-	-	-	-	0	0	No support of neither the data acknowledgment mechanism nor the pipe-level credit-based flow control
-	-	-	-	-	-	0	1	Support of only the data acknowledgment mechanism
-	-	-	-	-	-	1	0	RFU
-	-	-	-	-	-	1	1	Support of both the credit-based data flow control and the data acknowledgment mechanisms
X	X	X	X	X	X	-	-	RFU
Bits b3 to b8 shall be ignored by the loopback application gate.								

The value of the LOOPBACK_CAPABILITIES parameter of the loopback service gate in the SSP host shall indicate the support for pipe-level data acknowledgement mechanism and credit-based flow control. The support for these is mandatory for the SSP host if the SSP host supports at least one other gate that requires them. In all other cases, the support for pipe-level data acknowledgement mechanism and credit-based flow control in the SSP host is optional.

8.5 SCL procedures

8.5.1 Host registration

The provisions of GlobalPlatform VPP - Network Protocol [13], clause 6.1.2 shall apply.

8.5.2 Host deregistration

The provisions of GlobalPlatform VPP - Network Protocol [13], clause 6.1.4 shall apply.

8.5.3 Pipe management

The provisions of GlobalPlatform VPP - Network Protocol [13], clause 6.2 shall apply, with the following additions in the binding procedure described in GlobalPlatform VPP - Network Protocol [13], clause 6.2.1:

- a host shall not request a pipe binding for a service gate if this service gate has already a pipe session for this host.

8.5.4 Registry access

The provisions of GlobalPlatform VPP - Network Protocol [13], clause 6.3 shall apply.

8.5.5 Hosts and gates discovery

The provisions of GlobalPlatform VPP - Network Protocol [13], clause 6.4 shall apply.

8.5.6 Loopback testing

The provisions of GlobalPlatform VPP - Network Protocol [13], clause 6.5 shall apply.

9 Secure SCL

9.1 Protocol stack

Figure 9.1 illustrates the protocol supporting a secure communication layer between a generic service X and application X.

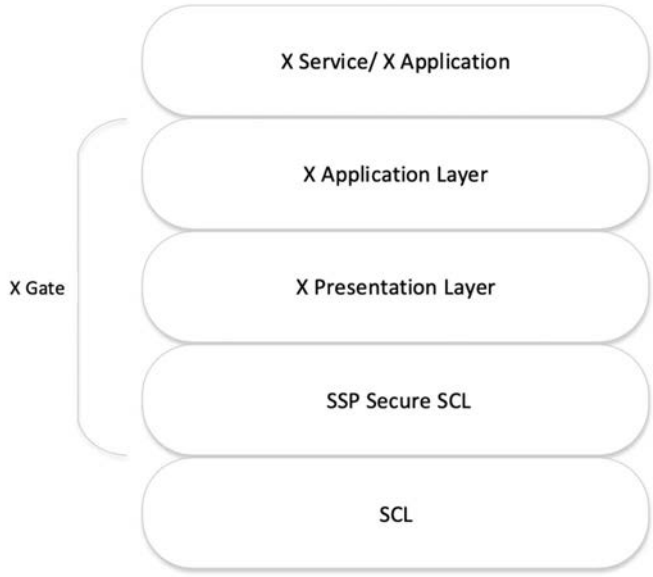

Figure 9.1: Protocol stack for secure SCL

9.2 Structure of secure SCL message

The messages, data streams, commands and responses between a service gate and an application gate may be authenticated and their data encrypted in secure SCL messages conveyed in SCL packets. The value of the DIVERSIFIER defined in clause C.4.2 shall be the logical XOR of the aChallenge value as defined in clause 6.13.5.7 and the gate identifier (aGateIdentifier) as defined in clause 6.13.5.6.

Figure 9.2 defines the structure of a secure SCL message in the case where a stream is sent. In the scenario shown an application sends a message to the Secure SCL layer.

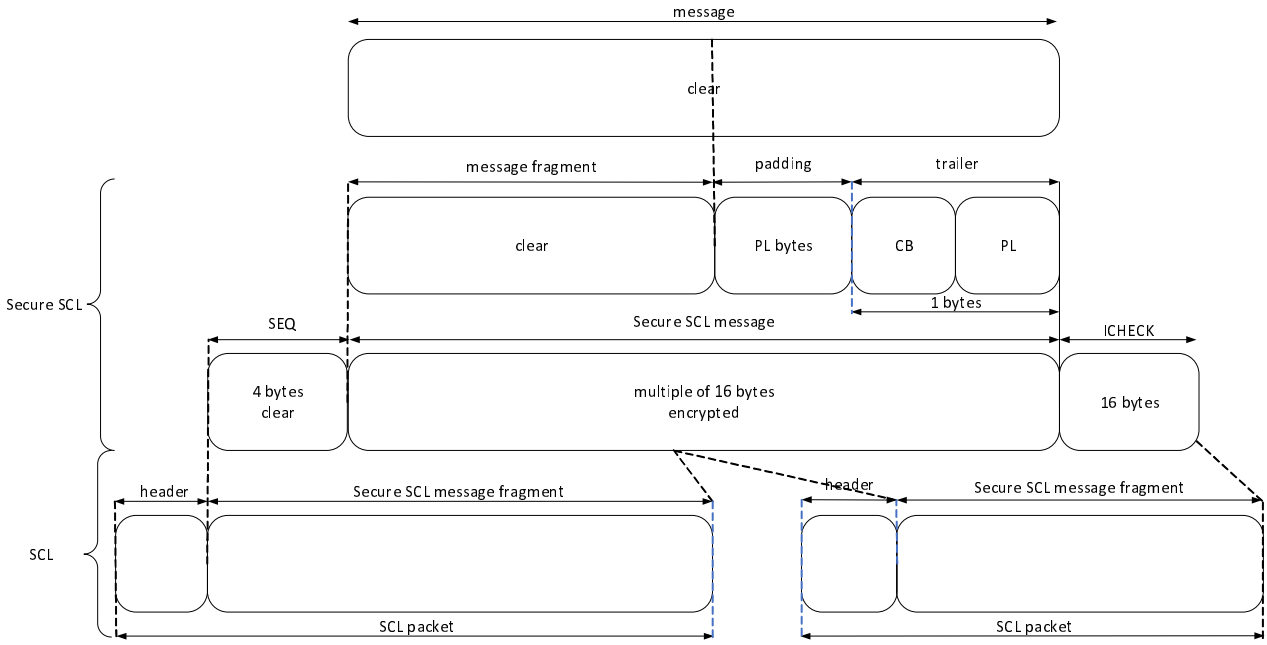

Figure 9.2: Structure of secure SCL message

Where:

- SEQ: 32 bit unsigned integer coded in big endian. SEQ^{32BIT} contains a monotonic counter managed by the gate sending the secure SCL message. SEQ shall be initialized to 1 after the successful AAS-OP-ACCESS-SERVICE-Service-Response. The SEQ counter shall be incremented after the sending of the secure SCL message.
- PL: Number of padding bytes appended to the message fragment (see padding below). PL is coded on bits 1 to 7.
- CB: Chaining bit of the clear text message fragment is coded on bit 8. CB is used by the secure SCL to fragment and reassemble the application layer message. CB shall be set to 1 for the last or only fragment of a message.
- Padding: PL zero bytes which are padding the message fragment in order to obtain a length of the secure SCL message which is a multiple of 16 bytes.
- Secure SCL message: Contains the cryptogram of the structure consisting of message fragment, padding, CB and PL. The cryptogram is generated by using a stream cipher algorithm identified by StreamCipherIdentifier value (see clause C.2.2).
- ICHECK: Integrity check of the secure SCL message using the stream cipher algorithm identified by StreamCipherIdentifier value.

If the stream cipher algorithm is the GCM then each gate supporting the secure SCL shall manage two GCM monotonic counters which shall be incremented after encrypting and decrypting each of the 128 bit blocks constituting the secure SCL message. The GCM counter shall be set to 1 for each secure SCL message by their transmitter. If a secure SCL message is corrupted then the pipe session shall be closed and new AAS-OP-ACCESS-SERVICE-Service-Command command shall be performed by the accessor. If the AAS-OP-ACCESS-SERVICE-Service-Command command is successful then the accessor authentication service shall reply with an AAS-OP-ACCESS-SERVICE-Service-Response containing a new and randomized gate identifier (aGateIdentifier).

The secure SCL message results from the encryption by using the security function as defined in clause C.3.5. The SharedInfo SI (see clause C.4.2), used by this security function, shall be deduced from the Accessor Authentication Service Protocol as defined in clause 10.9. The way SI is transferred from the accessor authentication gate to the gate using the secure SCL is implementation dependent.

The application layer message fragmentation depends on the buffering capability at the gate level of the host which sends or receives the message and is implementation dependent.

The secure SCL message, SEQ and ICHECK are passed to/from the SCL layer (see clause 8.3) as the service data unit of the SCL transport layer.

9.3 Security protocol

9.3.1 Overview

The security protocol between an accessor authentication service and application has the following steps:

- The initialization of the shared secret by using the security protocol described in clause 9.3.2.
- The generation of shared keys between a service and an application using the secure SCL described in clause 9.3.3.

9.3.2 Shared secret initialization

The shared secret initialization is using the security protocol defined in figure 9.3.

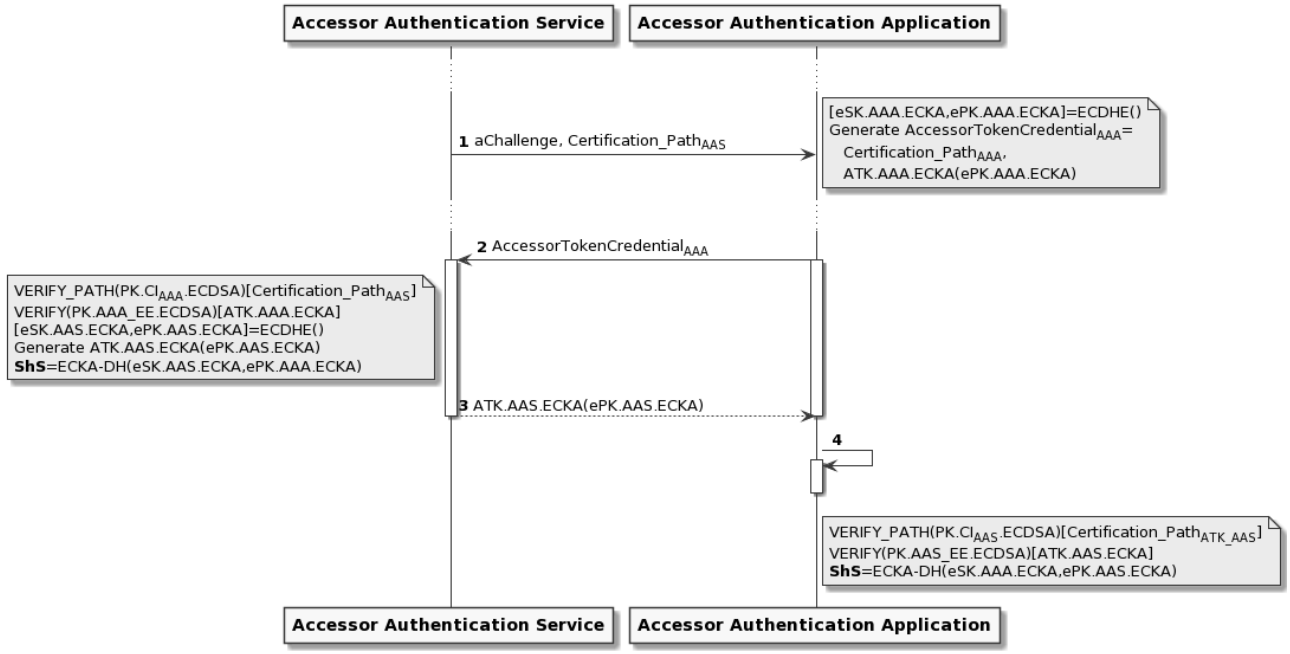

Figure 9.3: Secure SCL security protocol shared secret initialization

The procedure supporting the security protocol is the following:

1) The Accessor Authentication Service (AAS) generates a challenge (aChallenge) and sends it with the certification path of the accessor authentication service to the Accessor Authentication Application (AAA).

2) The accessor authentication application performs the following operations:

   - Generation of an ephemeral key pair (ePK.AAA.ECKA, eSK.AAA.ECKA).
   - Generation of the ATK.AAA.ECKA authentication token which is validated by the end entity certificate of the Certification_Path$_{ATK_AAA}$. The ATK.AAA.ECKA authentication token as defined in clause C.2.2, signed by private key coupled with the public key in the end entity certificate of the Certification_Path$_{ATK_AAA}$. ATK.AAA.ECKA embeds ePK.AAA.ECKA in the subjectPublicKeyInfo field of the TBSToken.
   - Generation of the AccessorTokenCredential combining the Certification_Path$_{ATK_AAA}$ and the ATK.AAA.ECKA authentication token.
   - Send the AccessorTokenCredential to the accessor authentication service.

3) The Accessor Authentication Service performs the following operations:

   - Validation of Certification_Path$_{ATK_AAA}$ by using PK.CI$_{AAA}$.ECDSA public key.
   - Validation of the ATK.AAA.ECKA by using the public key of the end entity certificate of Certification_Path$_{ATK_AAA}$.
   - Generation of an ephemeral key pair (ePK.AAS.ECKA, eSK.AAS.ECKA).
   - Generation of the ATK.AAS.ECKA authentication token as defined in clause C.2.2, signed by private key coupled with the public key of the end entity certificate of Certification_Path$_{ATK_AAS}$.
   - Generation of the ATK.AAS.ECKA authentication token. ATK.AAS.ECKA embeds ePK.AAS.ECKA in the subjectPublicKeyInfo field of the TBSToken.
   - Computation of the shared secret ShS by using ECKA_DH (anonymous Diffie-Hellman ECC key agreement) with the ephemeral key pair eSK.AAS.ECKA and ePK.AAA.ECKA as defined in clause C.4.
   - Send the ATK.AAS.ECKA authentication token to the accessor authentication application.

4) The accessor authentication application performs the following operations:

   - Validation of Certification_Path$_{ATK_AAS}$ by using PK.CI$_{AAS}$.ECDSA public key.
   - Validation of the ATK.AAS.ECKA by using the public key of the end entity certificate of Certification_Path$_{ATK_AAS}$.
   - Computation of the shared secret ShS by using ECKA_DH (anonymous Diffie-Hellman ECC key agreement) with the ephemeral key pair eSK.AAA.ECKA and ePK.AAS.ECKA.

If the accessor authentication service validates the Certification_Path$_{ATK_AAA,}$ and the ATK.AAA.ECKA authentication token containing its aChallenge challenge then the accessor authentication application is authenticated.

If the accessor authentication application validates the Certification_Path$_{ATK_AAS,}$ and the ATK.AAS.ECKA authentication token containing the aChallenge challenge then the accessor authentication service is authenticated.

NOTE: The case where the accessor authentication application fails to authenticate the service is currently not specified. Care should be used in the implementation to make sure that resources are freed.

Subsequent to the above procedure, both the accessor authentication service and accessor authentication application share the secret ShS. This shared secret is the seed for deriving the keys for the secure SCL communication.

9.3.3 Secure SCL shared keys generation

From the shared secret ShS obtained from the procedure described in clause 9.3.2, any service or application may initiate the generation of the key data by using the KDF function defined in clause C.4 and a DIVERSIFIER^{128BIT} that is equal to aGateIdentifier defined in clause 6.13.5.6.

This generation of key data is performed for each AAS-OP-ACCESS-SERVICE-Service-Command as defined in clause 6.13.5.7 and when the secure SCL is required.

9.4 Accessor authentication service procedure

9.4.1 Initialization

Figure 9.4 illustrates the exchanges between the accessor authentication service and accessor authentication application gates for supporting the security protocol defined in clause 9.3 and details the parameters of the procedure.

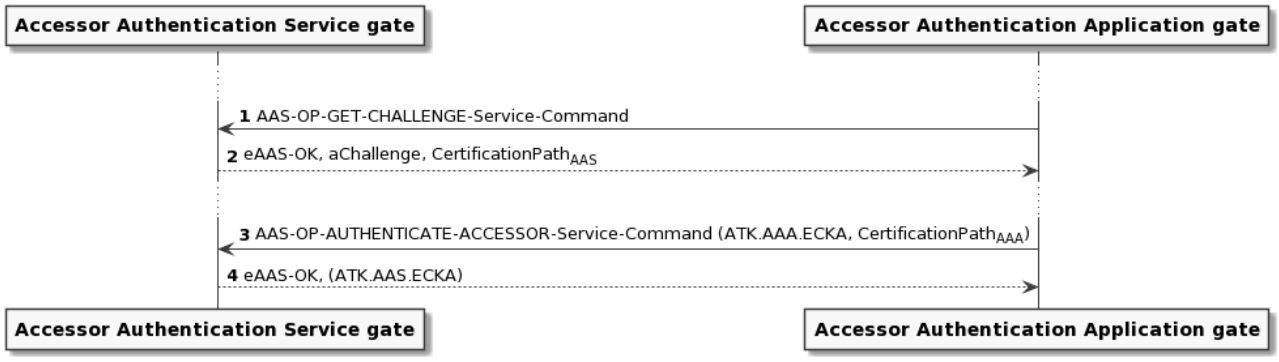

Figure 9.4: Accessor authentication service initialization

The procedure supporting the accessor authentication security scheme is the following:

1) The accessor authentication application gate requests the initialization of the security protocol as defined in clause 9.3.

2) The accessor authentication service gate returns the aChallenge and the certification path of the accessor authentication service.

3) The accessor authentication application gate requests an authentication based on the authentication token with its AccessorTokenCredential containing its authentication token and its certification path as parameter.

4) The accessor authentication service gate returns its authentication token (ATK.AAS.ECKA) as parameter.

10 Communication layers above SCL

10.1 Overview

The SSP may support multiple application protocols.

An SCL gate shall not send a command when this gate is waiting for a response to a previous command.

10.2 APDU protocol

10.2.1 Introduction

This clause describes the interaction between the SSP and the terminal using APDUs.

10.2.2 Command-response pairs

10.2.2.1 General definition

The provisions of ISO/IEC 7816-4 [4], clause 5.1 shall apply, with the exceptions listed in the clauses below.

10.2.2.2 CLA byte

The provisions of ETSI TS 102 221 [1], clause 10.1.1 shall apply.

10.2.2.3 INS byte

The INS byte indicates the command to process and its meaning depends on the application selected on the logical channel indicated in the CLA byte of the command APDU.

The following instructions are applicable on the basic logical channel.

Table 10.1: Coding of Instruction Byte

Command APDUs	INS	Clause	Support (M/O/C)
SELECT	'A4'	10.2.3.3	O
MANAGE CHANNEL	'70'	10.2.4.2	O
EXCHANGE CAPABILITIES	'7A'	10.2.3.2	M

Additional instructions are applicable on the basic logical channel, depending on the SSP capabilities and the selected application. Some examples are provided in the subsequent clauses.

The values '6X' and '9X' are invalid.

10.2.2.4 Coding of SW1 and SW2

The status bytes SW1 SW2 indicate the status of the UICC at the end of a command. The meaning depends on the application selected on the logical channel indicated in the CLA byte of the command APDU. If no application is selected and the SSP has support for the UICC file system, the provisions of ETSI TS 102 221 [1], clause 10.2.2 shall apply.

The value '61XX' is reserved as a special value when APDUs are transported over ISO/IEC 7816-4 [4] interface and shall not be used for other purposes.

NOTE: The handling of warning status words is defined as per ISO/IEC 7816-4 [4].

10.2.3 SSP commands

10.2.3.1 Overview

The following clauses describe the list of commands that may be supported by the SSP.

NOTE: This list is not exhaustive. Additional commands may be supported depending on the selected application (e.g. GET CHALLENGE, AUTHENTICATE and GET IDENTITY as defined in ETSI TS 102 221 [1]).

10.2.3.2 EXCHANGE CAPABILITIES

10.2.3.2.1 Description

This command is used to inform the SSP of the capabilities of the terminal and to retrieve the capabilities of the SSP. This command shall be executed immediately after the SSP Interface Session is started. The command might be executed again if some of the capabilities change.

The SSP and the terminal shall use the values exchanged during the last execution of this command.

The values of the EXCHANGE CAPABILITIES command take precedence over any equivalent value exchanged over the physical interface (for example, using the ATR) or over the transport interface.

10.2.3.2.2 Command parameters

Table 10.2

Code	Value
CLA	As specified in clause 10.2.2.2
INS	As specified in table 10.1
P1	'00'
P2	'00'
Lc	Length of subsequent data field or empty
Data	Capabilities of the terminal
Le	'00'

10.2.3.2.3 Command data

The command data contains a sequence of TLVs, coded as per clause 6.4.2.4.

10.2.3.2.4 Command response

The command data contains a sequence of TLVs, coded as per clause 6.4.2.5.

10.2.3.3 SELECT

The provisions of ISO/IEC 7816-4 [4] and of ETSI TS 102 221 [1] for the SELECT command with P1 = '04' ("Select by DF name") shall apply. The coding of P2 is described in table 10.3.

Table 10.3: Coding of P2

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	-	-	0	-	-	-	-	Other values are outside the scope of the present document
-	X	X	-	-	-	-	-	SSP Application session control:
-	0	0	-	-	-	-	-	- Activation/Reset
-	1	0	-	-	-	-	-	- Termination (see note 2)
-	-	-	-	0	0	-	-	Return FCI template (see note 1)
-	-	-	-	0	1	-	-	Return FCP template
-	-	-	-	1	0	-	-	Return FMD template (see note 1)
-	-	-	-	1	1	-	-	No data returned
-	-	-	-	-	-	X	X	Occurrence:
-	-	-	-	-	-	0	0	- First or only occurrence
-	-	-	-	-	-	0	1	- Last occurrence
-	-	-	-	-	-	1	0	- Next occurrence
-	-	-	-	-	-	1	1	- Previous occurrence
NOTE 1: Only applicable for SSP Applications compliant with ISO/IEC 7816-4 [4]. NOTE 2: Only applicable for SSP Applications compliant with ETSI TS 102 221 [1].								

10.2.4 Logical channels

10.2.4.1 Overview

Logical channels are defined in ISO/IEC 7816-4 [4].

The SSP can support up to 20 logical channels, which are 19 logical channels in addition to the basic logical channel 0. The SSP indicates the maximum number of supported logical channels during the capability exchange with the terminal. Channel 0 is always available and open.

A logical channel is opened by using a MANAGE CHANNEL command, in which the card assigns a channel number and returns it in the response. The logical channel remains open until it is explicitly closed by a MANAGE CHANNEL command, or if the connection between the terminal and the SSP entity handling the APDUs is deactivated.

The terminal shall not open more logical channels than the SSP supports.

10.2.4.2 MANAGE CHANNEL

The provisions of ETSI TS 102 221 [1], clause 11.1.17 shall apply.

Support for this command is mandatory if the SSP indicates support for logical channels during the capability exchange procedure.

10.2.5 UICC file system commands

10.2.5.1 Overview

This clause describes the behaviour of the SSP and the terminal when the SSP indicates support for the UICC file system.

10.2.5.2 Methods for selecting a file

The provisions of ETSI TS 102 221 [1], clause 8.4 shall apply.

10.2.5.3 Reservation of file IDs

The provisions of ETSI TS 102 221 [1], clause 8.6 shall apply.

10.2.5.4 Additional commands

In addition to the commands described in clause 10.2.3, these additional commands shall be supported by the SSP on the default logical channel.

Table 10.4

Command	Reference	Notes
SELECT	ETSI TS 102 221 [1], clause 11.1.1	P1 values: '00', '01', '03', '08' and '09'
READ BINARY	ETSI TS 102 221 [1], clause 11.1.3	
UPDATE BINARY	ETSI TS 102 221 [1], clause 11.1.4	
READ RECORD	ETSI TS 102 221 [1], clause 11.1.5	
UPDATE RECORD	ETSI TS 102 221 [1], clause 11.1.6	
SEARCH RECORD	ETSI TS 102 221 [1], clause 11.1.7	
INCREASE	ETSI TS 102 221 [1], clause 11.1.8	
VERIFY PIN	ETSI TS 102 221 [1], clause 11.1.9	
CHANGE PIN	ETSI TS 102 221 [1], clause 11.1.10	
DISABLE PIN	ETSI TS 102 221 [1], clause 11.1.11	
ENABLE PIN	ETSI TS 102 221 [1], clause 11.1.12	
UNBLOCK PIN	ETSI TS 102 221 [1], clause 11.1.13	
DEACTIVATE FILE	ETSI TS 102 221 [1], clause 11.1.14	
ACTIVATE FILE	ETSI TS 102 221 [1], clause 11.1.15	
RETRIEVE DATA	ETSI TS 102 221 [1], clause 11.3.1	
SET DATA	ETSI TS 102 221 [1], clause 11.3.2	

10.2.5.5 Security features

The provisions of ETSI TS 102 221 [1], clause 9 shall apply.

10.2.6 Card Application Toolkit

10.2.6.1 Overview

When the SSP indicates support for Card Application Toolkit according to ETSI TS 102 223 [6], the provisions of ETSI TS 102 221 [1], clause 7.4.2 shall apply.

When the physical interface used to transport APDUs allows the SSP to remotely wake up the terminal in case of proactive command, the SSP shall use that mechanism to inform the terminal of a pending proactive command. In this case, the terminal shall use the FETCH command APDU (see ETSI TS 102 221 [1]) to get the pending proactive command.

In the other cases, the SSP can reply '91XX' in place of '9000' to indicate that a proactive command is pending.

The terminal uses the FETCH command APDU to get the pending proactive command. The terminal sends to the SSP the response of the proactive command execution with the TERMINAL RESPONSE command APDU.

In all cases, the terminal shall send the FETCH and TERMINAL RESPONSE commands on the basic logical channel 0, even if the command to which the card replied with '91XX' was sent on a logical channel different from the basic logical channel.

10.2.6.2 Terminal profile

The Card Application Toolkit terminal profile allows the SSP to determine what the terminal is capable of, and the SSP shall then limit its instruction range accordingly.

If the terminal supports the Card Application Toolkit, the terminal profile shall be included in the capability exchange procedure.

The content of the terminal profile is defined in ETSI TS 102 223 [6], clause 5.2.

10.2.6.3 Proactive polling

When the proactive polling is indicated as required in the capability exchange procedure, the terminal shall perform proactive polling as defined in ETSI TS 102 221 [1], with the following exceptions:

- it is optional when the physical interface used to transport APDUs allows the SSP to remotely wake up the terminal;
- it is optional when the Card Application Toolkit is not supported by the SSP.

10.2.6.4 Additional commands

In addition to the commands described in clause 10.2.3, these additional commands shall be supported by the SSP on the default logical channel.

Table 10.5

Command	Reference	Notes
ENVELOPE	ETSI TS 102 221 [1], clause 11.2.2	
FETCH	ETSI TS 102 221 [1], clause 11.2.3	
TERMINAL RESPONSE	ETSI TS 102 221 [1], clause 11.2.4	
STATUS	ETSI TS 102 221 [1], clause 11.1.2	Only if proactive polling is indicated as required

10.2.7 SSP suspension

If the SSP suspension is supported, these additional commands shall be supported by the SSP on the default logical channel.

Table 10.6

Command	Reference	Notes
SUSPEND UICC	ETSI TS 102 221 [1], clause 11.1.22	

SUSPEND UICC defines the mechanism of going into suspend and how to resume from suspend. It further specifies timing behaviour related to suspend and resume.

10.2.8 APDU transfer over SCL

10.2.8.1 Overview

APDUs may be carried over SCL, on SSPs that implement SCL. To support UICC applications and UICC platforms on an SSP that implements the SCL, the SSP may provide one or more APDU service gates.

Annex B describes possible implementation options for APDU service gate(s) in the SSP.

10.2.8.2 UICC APDU gate

10.2.8.2.1 UICC APDU overview

This clause defines the gates for supporting the UICC APDU over SCL protocol. Figure 10.1 illustrates a platform supporting this gate.

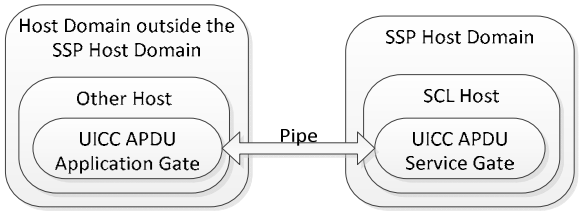

Figure 10.1: Platform supporting the UICC APDU service

An SSP host shall contain no more than one UICC APDU service gate.

Each SCL host outside the SSP host domain shall not create more than one pipe session to the UICC APDU service gate for each SSP host.

The communication between the UICC APDU service gate and the UICC APDU application gate uses the presentation layer defined in ETSI TS 102 622 [14], clause 5.2 and the specific part of the transport layer with the Go-and-Wait data flow control defined in clause 10.2.8.2.4.

The UICC APDU gates reuses the mechanisms described in ETSI TS 102 622 [14], clause 12.

10.2.8.2.2 UICC APDU service gate

The UICC APDU service gate URN supports the syntax as defined in clause 8.2, with the values specified in table 8.1.

The UICC APDU service gate shall support the commands, events and registry as described in ETSI TS 102 622 [14], clause 12.2.

10.2.8.2.3 UICC APDU application gate

10.2.8.2.3.1 Commands

The UICC APDU application gate shall support the commands described in ETSI TS 102 622 [14], clause 12.3.1.

10.2.8.2.3.2 Events

The UICC APDU application gate shall support the commands described in ETSI TS 102 622 [14], clause 12.3.2, with the addition of the following events.

Table 10.7

Value	Event
'13'	EVT_TOOLKIT_REQUEST

10.2.8.2.3.3 EVT_TOOLKIT_REQUEST

This event shall be sent by the UICC APDU service gate in idle state to indicate to the UICC APDU application gate that a proactive command is pending. After receiving this event, the UICC APDU application gate shall send an APDU containing the STATUS command to allow the SSP to start a proactive session.

This event has no parameters.

10.2.8.2.4 State diagram for the UICC APDU gate

Figure 10.2 specifies the states and the transitions that an APDU gate shall take during its operation.

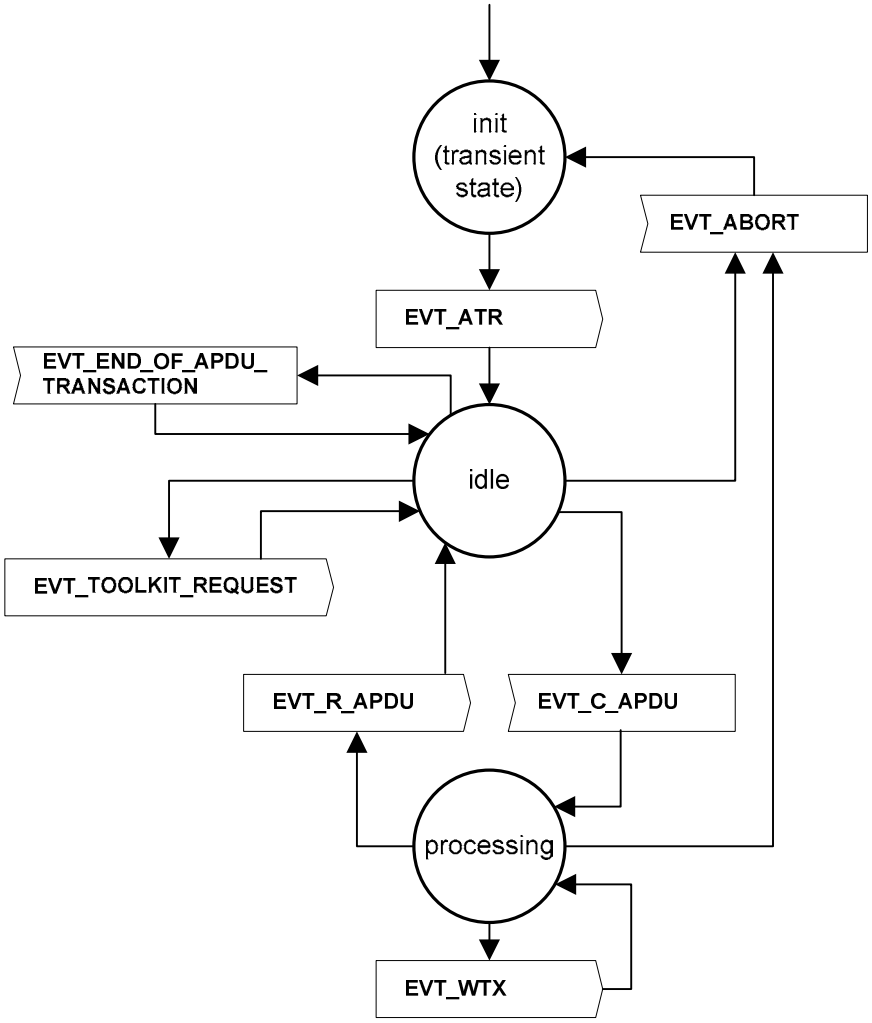

Figure 10.2: State transitions

Other events or commands received on the UICC APDU gate shall not change its state, with the exception of EVT_ADM_UNBIND which can be received in any state.

10.3 File system protocol

10.3.1 Overview

The SSP file system, as defined in clause 6.6.2, may be accessed by entities outside the SSP using the SSP file system service over the SCL protocol.

This clause defines the interaction between the SSP file system service with another SCL host.

The SSP file system service resides in an SSP host and shall contain a single file system control service gate.

NOTE: The SSP file system is accessed using the anonymous accessor, unless the SSP file system application performs the authentication using the AAS-OP-ACCESS-SERVICE-Service-Command through the accessor authentication service gate.

The SSP file system application resides in an SCL host outside the SSP host domain and shall contain a single file system control application gate.

Figure 10.3 illustrates a file system application accessing via a file system control application gate and the file system control service gate, the SSP file system service to perform operations on the SSP file system.

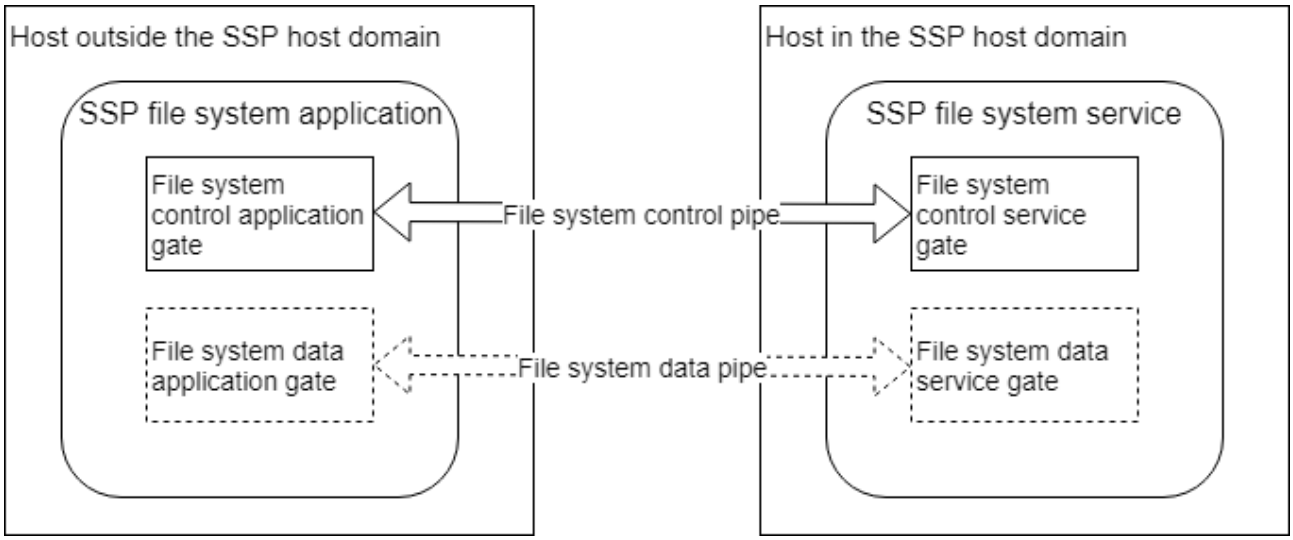

Figure 10.3: SSP file system platform

Both the SSP file system application and the SSP file system service may contain multiple file system data application/service gates.

The usage of the file system data service/application gates is optional if the amount of data to exchange per read and write command is sufficiently small. The data may then be exchanged via the file system control pipe.

The file system control service gate URN supports the syntax defined in clause 8.2, with the values specified in table 8.1.

10.3.2 Presentation layer

The file system control service gate and the file system control application gate implement an ASN.1 presentation layer using the definitions in clause 6.6.2.

10.3.3 File system control service gate

10.3.3.1 Overview

An SSP file system application may access to the SSP file system service interfacing the SSP File System via a pipe session between a file system control application gate and a file system control service gate.

The file system control service gate is in charge of:

- conveying the administrative (ADMIN) and operational (OP) commands;
- creating a dynamic gate identifier or triggering the opening of a dedicated data pipe session as defined in clause 8 connecting a data stream between the file system data service gate and a file system data application gate within the SSP file system service and the SSP file system application.

10.3.3.2 Commands

The file system control service gate supports the following commands.

```
-- ASN1START

FS-CONTROL-SERVICE-GATE-Commands ::= [APPLICATION 2] CHOICE
{
    aFS-ADMIN-GET-CAPABILITIES-Service-Command FS-ADMIN-GET-CAPABILITIES-Service-Command,
    aFS-ADMIN-CREATE-NODE-Service-Command FS-ADMIN-CREATE-NODE-Service-Command,
    aFS-ADMIN-DELETE-NODE-Service-Command FS-ADMIN-DELETE-NODE-Service-Command,
    aFS-ADMIN-UPDATE-NODE-ATTRIBUTES-Service-Command FS-ADMIN-UPDATE-NODE-ATTRIBUTES-Service-
Command,
    aFS-OP-FILE-OPEN-Service-Command FS-OP-FILE-OPEN-Service-Command,
```

```
    aFS-OP-FILE-CLOSE-Service-Command FS-OP-FILE-CLOSE-Service-Command,
    aFS-OP-NODE-GET-INFO-Service-Command FS-OP-NODE-GET-INFO-Service-Command,
    aFS-OP-FILE-READ-Service-Command FS-OP-FILE-READ-Service-Command,
    aFS-OP-FILE-WRITE-Service-Command FS-OP-FILE-WRITE-Service-Command,
    aFS-OP-FILE-GET-POSITION-Service-Command FS-OP-FILE-GET-POSITION-Service-Command
}

-- ASN1STOP
```

All the commands are described in clause 6.6.2.3.

10.3.3.3 Responses

The gate shall support the responses defined as follows.

```
-- ASN1START

FS-CONTROL-SERVICE-GATE-Responses ::= [APPLICATION 1] CHOICE
{
    aFS-ADMIN-GET-CAPABILITIES-Service-Response FS-ADMIN-GET-CAPABILITIES-Service-Response,
    aFS-ADMIN-CREATE-NODE-Service-Response FS-ADMIN-CREATE-NODE-Service-Response,
    aFS-ADMIN-DELETE-NODE-Service-Response FS-ADMIN-DELETE-NODE-Service-Response,
    aFS-ADMIN-UPDATE-NODE-ATTRIBUTES-Service-Response FS-ADMIN-UPDATE-NODE-ATTRIBUTES-Service-
Response,
    aFS-OP-FILE-OPEN-Service-Response FS-OP-FILE-OPEN-Service-Response,
    aFS-OP-FILE-CLOSE-Service-Response FS-OP-FILE-CLOSE-Service-Response,
    aFS-OP-NODE-GET-INFO-Service-Response FS-OP-NODE-GET-INFO-Service-Response,
    aFS-OP-FILE-READ-Service-Response FS-OP-FILE-READ-Service-Response,
    aFS-OP-FILE-WRITE-Service-Response FS-OP-FILE-WRITE-Service-Response,
    aFS-OP-FILE-GET-POSITION-Service-Response FS-OP-FILE-GET-POSITION-Service-Response
}

-- ASN1STOP
```

All the responses are described in clause 6.6.2.4.

10.3.3.4 Events

The gate has no dedicated events.

10.3.4 File system control application gate

10.3.4.1 Overview

The file system control application gate provides access to services for administration and operation on the SSP file system using the SSP file system control service gate between an SSP file system application and an SSP file system service.

All file sessions and file system data pipe sessions shall be closed upon closure of the pipe session between the file system control application gate and the file system control service gate.

10.3.4.2 Commands

The gate has no dedicated commands.

10.3.4.3 Responses

The gate has no dedicated responses.

10.3.4.4 Events

The gate has no dedicated events.

10.3.5 File system data service gate

10.3.5.1 Overview

The file system data service gate provides access to a file stream between an SSP file system application and an SSP file system service.

A pipe session between a file system data service gate and a file system data application gate shall be open when a request to open an SSP file with FS-OP-FILE-OPEN-Service-Command command is successful with either the aGateAppID value or the aDataPipeSession value set to TRUE. If the aGateAppID is provided, the SSP file system service shall open a pipe session. If the aDataPipeSession value is set to TRUE, the SSP file system service shall create a dedicated gate identifier (i.e. aGateServID).

The file system data service gate shall implement the credit-based data flow control and data acknowledgement as defined in clause 8.5.3.

The pipes between the file system data service gate and the file system data application gate allow conveying a data stream. The SCL packets shall have the CB bit (chaining bit) always set to 0.

10.3.5.2 Commands

The gate has no dedicated commands.

10.3.5.3 Responses

The gate has no dedicated responses.

10.3.5.4 Events

The gate has no dedicated events.

10.3.6 File system data application gate

10.3.6.1 Overview

The provisions of the file system data service gate defined in clause 10.3.5.1 shall apply also to this gate.

10.3.6.2 Commands

The gate has no dedicated commands.

10.3.6.3 Responses

The gate has no dedicated responses.

10.3.6.4 Events

The gate has no dedicated events.

10.4 Transmission Control Protocol support

10.4.1 Overview

This clause describes the interaction of SSP Applications and services with external entities using the TCP/IP protocol suite. The protocol allows the SSP to serve TCP connections in server and/or client mode to communicate with entities over an IP network. The SSP does not have a TCP/IP stack, but relies on the TCP/IP stack in the terminal.

This clause defines a mechanism that allows the SCL host to send and receive a TCP data stream over SCL.

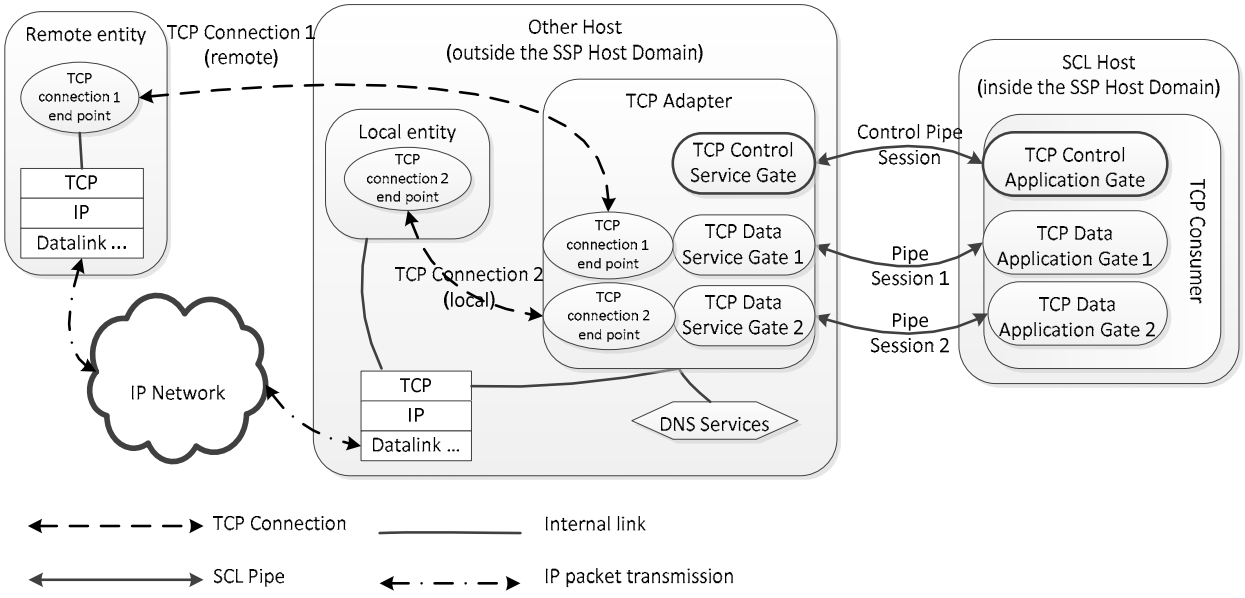

Figure 10.4: TCP support overview

The TCP adapter is a component of an SCL host residing outside the SSP host domain and it interfaces to the TCP consumer via the TCP data service gate(s). The TCP adapter interfaces to the TCP/IP stack of the terminal to communicate over an IP network. The TCP data service gate implements the protocol stack to communicate over the SCL network to a TCP data application gate for bridging the TCP adapter and TCP consumer. The TCP adapter may support the DNS resolution of Fully Qualified Domain Names using services of the terminal. The TCP adapter contains the logic to interconnect TCP streams with TCP data service gates. The TCP adapter is in charge of:

- the creation of the TCP connection;
- the resolution of DNS address, when this is required;
- the triggering of a pipe session as defined in clause 8 for connecting the TCP adapter and the TCP consumer in the SSP.

The TCP consumer resides in an SSP host and it uses the TCP control application gate and the TCP data application gate(s).

The TCP adapter shall contain only one TCP control service gate.

If an SSP host supports the TCP protocol, it shall contain only one TCP consumer, which includes a single TCP control application gate. The SCL host may implement logic for sharing the access to the TCP adapter between multiple SSP Applications.

An SCL host may contain one or more TCP data service gates.

Each TCP data application gate is exclusive for a single SSP Application.

The TCP control service gate URN supports the syntax as defined in clause 8.2 with the values specified in table 8.1.

10.4.2 Management of TCP connections

10.4.2.1 TCP connection request

10.4.2.1.1 TCP active connection request (client mode)

Upon request of the TCP consumer to establish an active connection, the TCP adapter shall process a connection establishment to the remote endpoint indicated in the request. If an FQDN is provided in the request, the TCP adapter shall perform a DNS resolution if supported. If the DNS resolution is not supported by the TCP adapter, or the establishment of the connection failed, the TCP adapter shall indicate to the TCP consumer that the connection request failed with the appropriate error indicator.

After the TCP connection is successfully established, the TCP adapter shall open a TCP data pipe to the TCP data application gate identifier indicated by the TCP consumer. This shall be interpreted by the TCP consumer that the TCP connection was successfully established.

10.4.2.1.2 TCP passive connection request (server mode)

Upon request of the TCP consumer to establish a connection as passive, the TCP adapter shall bind and listen to the port provided in the request. In case of failure, the TCP adapter shall indicate to the TCP consumer that the connection failed with the appropriate error indicator. The TCP adapter shall support multiple passive requests by the TCP consumer with different TCP data application gate identifiers to the same TCP port in order to allow multiple incoming TCP connections on the same port.

When a connection is successfully established to the listening TCP port, the TCP adapter shall request the TCP consumer to accept the incoming connection before completing the TCP handshake. If the TCP consumer accepts the incoming connection, the TCP adapter shall open a TCP data pipe to the TCP data application gate identifier indicated by the TCP consumer in the request. If the TCP consumer rejects the incoming connection, the TCP adapter shall close the incoming TCP connection. The TCP adapter shall accept additional incoming TCP connections to the same port until all TCP data application gate identifiers corresponding to this port are used.

10.4.2.2 TCP connection established

All data received by the TCP adapter in its TCP endpoint shall be transferred via the related TCP data pipe to the TCP consumer. All data sent by the TCP consumer on the TCP data pipe shall be sent to its corresponding TCP endpoint.

10.4.2.3 TCP end of connection

The TCP session between the remote TCP endpoint and the TCP adapter is bound to the corresponding TCP Data pipe session. If a TCP session is closed by the remote TCP endpoint, the TCP adapter shall close the related TCP data pipe session: in this case, the connection identifier is immediately released and the TCP consumer does not need to send the request to close the TCP connection. If the TCP data application gate ends the TCP data pipe session, the TCP adapter shall terminate the connection to the remote TCP endpoint.

If the TCP control pipe session is closed, all TCP connections and TCP data pipe sessions shall be closed. All passive connection requests are terminated.

10.4.3 Presentation layer

The TCP control and application gates implement an ASN.1 presentation layer using the following definitions.

```
-- ASN1START

/* TCP Service: definitions */
IPV6Addr ::= OCTET STRING (SIZE(16))
IPV4Addr ::= OCTET STRING (SIZE(4))
IPAddress ::= CHOICE {aIPV4 IPV4Addr, aIPV6 IPV6Addr}
FQDN ::= UTF8String
ConnectionID ::= INTEGER (0..255)

-- ASN1STOP
```

10.4.4 TCP control service gate

10.4.4.1 Overview

The TCP control service gate provides access to services that manage the TCP connections between the TCP adapter and remote entities on behalf of the TCP consumer.

10.4.4.2 Commands

10.4.4.2.1 List of commands

The gate shall support the following commands.

```
-- ASN1START

TCP-CONTROL-SERVICE-GATE-Commands ::= [APPLICATION 2] CHOICE
{
    aTCP-REQUEST-CONNECTION-Service-Command TCP-REQUEST-CONNECTION-Service-Command,
    aTCP-GET-STATUS-CONNECTION-Service-Command TCP-GET-STATUS-CONNECTION-Service-Command,
    aTCP-CLOSE-CONNECTION-Service-Command TCP-CLOSE-CONNECTION-Service-Command
}

-- ASN1STOP
```

10.4.4.2.2 TCP-REQUEST-CONNECTION-Service-Command

With the command TCP-REQUEST-CONNECTION-Service-Command, a TCP consumer requests the TCP adapter to open a TCP connection with a set of parameters.

The TCP consumer may request to establish the connection by passing the IP address or the FQDN value of the server. If the FQDN value is used, the TCP adapter is responsible to perform the DNS resolution, if the feature is supported.

Depending on the connection mode, the TCP adapter shall perform the following:

- In case of eActive, it shall initiate a TCP connection, as described in IETF RFC 793 [21], section 3.4.
- In case of ePassiveLocal or ePassiveAny, it shall open a listening port to accept incoming TCP connections.

This command has the following parameters.

```
-- ASN1START

NetworkParameters ::= SEQUENCE
{
    aBearerType INTEGER  -- Bearer type
    {
        eDefaultBearer (0),  -- Default Bearer, as defined in ETSI TS 102 223 [6], clause 8.52
        eWWAN (1),  -- WWAN Bearer
        eWLAN (2)  -- WLAN local breakout
    } OPTIONAL,
    aNetworkAccessName OCTET STRING OPTIONAL,  -- Network Access Name
    aUserLogin OCTET STRING OPTIONAL,  -- User login for the network
    aUserPassword OCTET STRING OPTIONAL  -- User password for the network
}

TCP-REQUEST-CONNECTION-Service-Command ::= [PRIVATE 16] SEQUENCE
{
    aConnectionMode [0] INTEGER
    {
        ePassiveLocal (0),  -- TCP mode for LISTEN/ACCEPT for client local to the device only
        ePassiveAny (1),  -- TCP mode for LISTEN/ACCEPT for either local or remote client
        eActive (2)  -- TCP Mode as client for CONNECT
    },
    aDestinationAddress [1] CHOICE  -- Destination address
    {
        aIP IPAddress,  -- IP address
        aFQDN FQDN  -- Fully Qualified Domain Name of the server
    } OPTIONAL,
    aPortNumber [2] INTEGER(1..65535),  -- Port number
    aGateID [3] UUID,  -- Gate identifier as defined in clause 8.2
```

```
    aTimeout [4] INTEGER OPTIONAL,  -- Time unit is second
    aNetworkParameters [5] NetworkParameters OPTIONAL  -- Network parameters
}

-- ASN1STOP
```

Where:

- aConnectionMode: It defines the mode of the connection that can be passive or active of TCP adapter as defined previously in this clause;
- aDestinationAddress: It defines, for connections in active mode the end point destination address of the connection establishment. It can be an IPv4 or IPv6 Address or Full Qualified Domain Name. This parameter shall be ignored by TCP adapter for connections in passive mode i.e. aConnectionMode with value ePassiveLocal or ePassiveAny. This parameter is mandatory for connection in active mode i.e. aConnectionMode with value eActive;
- aPortNumber: It defines the end point destination port number for connections in active mode, i.e. aConnectionMode with value eActive. For connections in passive mode, i.e. aConnectionMode with value ePassiveLocal or ePassiveAny, it defines the port number that the TCP adapter shall listen on;
- aGateID: It is the UUID of the TCP data gate that will be associated to the opened TCP connection;
- aTimeout: The timeout value is duration of time before the terminal stops the attempt to connect to a remote sever;
- aNetworkParameters: Indicates the network parameters using which the TCP connection shall be established. The NetworkParameters is composed of the following parameters:
  - aBearerType: It indicates the bearer type on which the TCP connection shall be established;
  - aNetworkAccessName: The network access name provides information to the terminal necessary to identify the gateway entity which provides interworking with an external packet data network. If the parameter is not present, the terminal may use the default network access name in the terminal configuration or the default subscription value. It is defined in clause 8.70 of ETSI TS 102 223 [6];
  - aUserLogin and aUserPassword: If the terminal equipment supports a remote access login feature, it gives necessary information for authentication as described in ETSI TS 102 223 [6], clauses 6.6.27.2 and 6.6.27.4. The format and content of the data (data coding scheme and text string) is described in clause 8.15 of ETSI TS 102 223 [6].

For all the connection modes, the TCP adapter shall send the response immediately after starting the procedure to establish the TCP connection.

```
-- ASN1START

TCP-REQUEST-CONNECTION-Service-Response-Parameter ::= SEQUENCE
{
    aConnectionID [0] ConnectionID  -- Connection ID
}

TCP-REQUEST-CONNECTION-Service-Response ::= [PRIVATE 16] SEQUENCE
{
    aTCP-Control-Service-Response TCP-Control-Service-Response DEFAULT eTCP-OK,
    aParameter TCP-REQUEST-CONNECTION-Service-Response-Parameter OPTIONAL
}

-- ASN1STOP
```

When the connection request is successful then TCP adapter shall include eTCP-OK in the response:

- aConnectionID: This parameter is a unique identifier provided by the TCP adapter upon request of a TCP connection by the TCP consumer. This identifier shall be unique across all open TCP sessions.

10.4.4.2.3 TCP-CLOSE-CONNECTION-Service-Command

With the command eTCP-CLOSE-CONNECTION-Service-Command a TCP consumer may request the TCP adapter to close an open TCP connection or a listening TCP socket.

```
-- ASN1START

TCP-CLOSE-CONNECTION-Service-Command ::= [PRIVATE 17] SEQUENCE
{
    aConnectionID [0] ConnectionID  -- Connection ID
}

-- ASN1STOP
```

This command has the following parameters:

- aConnectionID: This parameter is the connection identifier of TCP connection to close.

When successful the TCP adapter shall include eTCP-OK in the response.

```
-- ASN1START

TCP-CLOSE-CONNECTION-Service-Response-Parameter ::= SEQUENCE
{
    aConnectionID [0] ConnectionID  -- Connection ID
}

TCP-CLOSE-CONNECTION-Service-Response ::= [PRIVATE 17] SEQUENCE
{
    aTCP-Control-Service-Response TCP-Control-Service-Response DEFAULT eTCP-OK,
    aParameter TCP-CLOSE-CONNECTION-Service-Response-Parameter OPTIONAL
}

-- ASN1STOP
```

Where:

- aConnectionID: This parameter is the connection identifier of the closed TCP connection.

10.4.4.2.4 TCP-GET-STATUS-CONNECTION-Service-Command

With the command TCP-GET-STATUS-CONNECTION-Service-Command a TCP consumer may request the TCP adapter to return status information about a TCP connection.

```
-- ASN1START

TCP-GET-STATUS-CONNECTION-Service-Command ::= [PRIVATE 18] SEQUENCE
{
    aConnectionID [0] ConnectionID  -- Connection ID
}

-- ASN1STOP
```

This command has the following parameters:

- aConnectionID: This parameter is the connection identifier of TCP connection to scan for retrieving information.

When successful the TCP adapter shall include eTCP-OK in the response.

```
-- ASN1START

TCP-GET-STATUS-CONNECTION-Service-Response-Parameter ::= SEQUENCE
{
    aConnectionID [0] ConnectionID,  -- Connection ID
    aStateOfConnection [1] INTEGER  -- State of the connection
    {
        eLISTEN (0),  -- TCP mode for LISTEN state (awaiting a connection request from a TCP client)
        eESTABLSHMENT-IN-PROGRESS (1),  -- TCP handshake is in progress
        eESTABLISHED (2),  -- TCP handshake has been completed
        eCLOSED (3)  -- TCP connection is CLOSED or not OPENED
    }
}

TCP-GET-STATUS-CONNECTION-Service-Response ::= [PRIVATE 18] SEQUENCE
{
    aTCP-Control-Service-Response TCP-Control-Service-Response DEFAULT eTCP-OK,
    aParameter TCP-GET-STATUS-CONNECTION-Service-Response-Parameter OPTIONAL
}
```

```
-- ASN1STOP
```

Where:

- aConnectionID: This parameter is the identifier of the TCP connection;
- aStateOfConnection: This parameter describes the state of the TCP connection and can have the following values:
  - The eLISTEN state means that the TCP connection is awaiting a connection request from a TCP client.
  - The eESTABLSHMENT-IN-PROGRESS state means that the TCP handshake is in progress and the connection is being established.
  - The eESTABLISHED state means that the TCP handshake has been successfully completed and the TCP connection is established with the TCP client.
  - The eCLOSED state means that the TCP connection has been closed.

10.4.4.3 Responses

The gate shall support the responses defined as follows.

```
-- ASN1START

TCP-CONTROL-SERVICE-GATE-Responses ::= [APPLICATION 1] CHOICE
{
    aTCP-REQUEST-CONNECTION-Service-Response TCP-REQUEST-CONNECTION-Service-Response,
    aTCP-GET-STATUS-CONNECTION-Service-Response TCP-GET-STATUS-CONNECTION-Service-Response,
    aTCP-CLOSE-CONNECTION-Service-Response TCP-CLOSE-CONNECTION-Service-Response
}

-- ASN1STOP
```

The gate shall support the error codes defined in table 10.8 as follows.

```
-- ASN1START

TCP-Control-Service-Response ::= ENUMERATED
{
    eTCP-OK (0),  -- no error
    eTCP-E-CMD-PAR-UNKNOWN (2),  -- unknown or illegal command parameter
    eTCP-E-NOK (3)  -- the command has failed
}

-- ASN1STOP
```

Table 10.8: Description of error codes

Error code	Description
eTCP-OK	Command completed successfully
eTCP-E-CMD-PAR-UNKNOWN	The format of the command parameters is wrong
eTCP-E-NOK	Command was rejected and/or not completed

Table 10.9: TCP control service gate commands/responses

Command	eTCP-OK	eTCP-E-CMD-PAR-UNKNOWN	eTCP-E-NOK
TCP-REQUEST-CONNECTION-Service-Command	•	•	•
TCP-CLOSE-CONNECTION-REQUEST-Service-Command	•	•	•
TCP-GET-STATUS-CONNECTION-Service-Command	•	•	•

10.4.4.4 Events

The gate has no dedicated events.

10.4.5 TCP control application gate

10.4.5.1 Overview

The TCP adapter handles the TCP connection state change events and the connection acceptance for the TCP passive mode.

10.4.5.2 Commands

10.4.5.2.1 List of commands

The gate supports the following commands.

```
-- ASN1START

TCP-CONTROL-APPLICATION-GATE-Commands ::= [APPLICATION 2] CHOICE
{
    aTCP-ACCEPT-CONNECTION-Application-Command TCP-ACCEPT-CONNECTION-Application-Command
}

-- ASN1STOP
```

10.4.5.2.2 TCP-ACCEPT-CONNECTION-Application-Command

With the command TCP-ACCEPT-CONNECTION-Application-Command a TCP adapter indicates an incoming TCP client connection to a TCP server connection requested by the TCP consumer.

```
-- ASN1START

TCP-ACCEPT-CONNECTION-Application-Command ::= [PRIVATE 16] SEQUENCE
{
    aConnectionID [0] ConnectionID,  -- Connection ID
    eSourceIP [1] IPAddress OPTIONAL,  -- IP address of the incoming connection
    aSourcePortNumber [2] INTEGER(1..65535)  -- Source port of the incoming connection
}

-- ASN1STOP
```

This command has the following parameters:

- aConnectionID: This parameter is the identifier of the accepted TCP connection;
- eSourceIP: This parameter is the IP address of the remote or local TCP destination;
- aSourcePortNumber: This parameter is the TCP port number corresponding to the incoming TCP client.

The TCP consumer shall include eTCP-OK in the response if the TCP client connection is accepted. Otherwise, the TCP consumer shall include eTCP-E-NOK in the response if the connection is rejected due to internal reasons. The TCP adapter shall terminate the incoming TCP connection and move back to eLISTEN state.

```
-- ASN1START

TCP-ACCEPT-CONNECTION-Application-Response ::= [PRIVATE 16] SEQUENCE
{
    aTCP-Control-Application-Response TCP-Control-Application-Response DEFAULT eTCP-OK
}

-- ASN1STOP
```

10.4.5.3 Responses

The gate shall support the responses defined as follows.

```
-- ASN1START

TCP-CONTROL-APPLICATION-GATE-Responses ::= [APPLICATION 1] CHOICE
{
    aTCP-REQUEST-CONNECTION-Application-Response TCP-ACCEPT-CONNECTION-Application-Response
}

-- ASN1STOP
```

The gate shall support the error codes defined in table 10.10 as follows.

```
-- ASN1START

TCP-Control-Application-Response ::= ENUMERATED
{
    eTCP-OK (0),  -- no error
    eTCP-E-CMD-PAR-UNKNOWN (2),  -- unknown or illegal command parameter
    eTCP-E-NOK (3)  -- the command has failed
}

-- ASN1STOP
```

Table 10.10: TCP control application gate commands/responses

Command	eTCP-OK	eTCP-E-CMD-PAR-UNKNOWN	eTCP-E-NOK
TCP-ACCEPT-CONNECTION-Application-Command	•	•	•

10.4.5.4 Events

10.4.5.4.1 List of events

The TCP consumer supports the following events.

```
-- ASN1START

TCP-CONTROL-APPLICATION-GATE-Events ::= [APPLICATION 0] CHOICE
{
    aEVT-TCP-ERROR-Application-Event EVT-TCP-ERROR-Application-Event
}

-- ASN1STOP
```

10.4.5.4.2 EVT-TCP-ERROR-Application-Event

With the event EVT-TCP-ERROR-Application-Event, the TCP adapter notifies the TCP consumer that an error occurred.

```
-- ASN1START

EVT-TCP-ERROR-Application-Event ::= [PRIVATE 16] SEQUENCE
{
    aConnectionID [0] ConnectionID,  -- Connection ID
    aErrorCode [1] INTEGER  -- Error code
    {
        eUNREACHABLE (1),
        eREDIRECTION (2),
        eTIMEOUT (3),
        eIP-HEADER-WRONG (4),
        eACCESS-TECHNOLOGY-ERROR (5),
        eTERMINAL-BUSY (6),
        eNETWORK-BUSY (7),
        eCALL-CONTROL-INTERACTION-ERROR (8),
        eDNS-RESOLUTION-ERROR (9),
        eLINK-DROPPED (10)
    },
    aErrorInfo [2] OCTET STRING OPTIONAL  -- Additional error information
}

-- ASN1STOP
```

Where:

- aConnectionID: This parameter is the identifier of the TCP connection to which the error is related;
- aErrorCode: This parameter is the identifier of the error code and it has one of the following values:
  - eUNREACHABLE: This parameter means that the destination IP address is unreachable as described in ICMP messages defined in IETF RFC 792 [26]. In this case, the aErrorInfo parameter shall be completed with the code value defined in clause "Destination Unreachable Message" of IETF RFC 792 [26];
  - eREDIRECTION: This parameter means that a redirection occurs in the route to convey the message as described in clause "Redirect Message" of IETF RFC 792 [26]. In this case, the aErrorInfo parameter shall be completed with the code value defined in clause "Redirect Message" of IETF RFC 792 [26];
  - eTIMEOUT: This parameter means that the timeout defined in the opening of the connection has expired;
  - eIP-HEADER-WRONG: This parameter means that the message format is wrong as described in clause "Parameter Problem Message" of IETF RFC 792 [26]. In this case, the aErrorInfo parameter shall be completed with the code value defined in clause "Parameter Problem Message" of IETF RFC 792 [26];
  - eACCESS-TECHNOLOGY-ERROR: This parameter means that the Access Technology was unable to process the command as described in clause 4.12 of ETSI TS 102 223 [6];
  - eTERMINAL-BUSY: This parameter means that terminal is currently unable to process the command as described in clause 8.12.2 of ETSI TS 102 223 [6]. In this case, the aErrorInfo parameter shall be completed with the value defined in clause 8.12.2 of ETSI TS 102 223 [6];
  - eNETWORK-BUSY: This parameter means that the network is currently unable to process the command as described in clause 8.12.3 of ETSI TS 102 223 [6]. In this case, the aErrorInfo parameter shall be completed with the value defined in clause 8.12.3 of ETSI TS 102 223 [6];
  - eCALL-CONTROL-INTERACTION-ERROR: This parameter means that the connection required to establish the TCP communication was blocked by the terminal due to the call control by NAA, as described in clause 7.3 of ETSI TS 102 223 [6]. In this case, the aErrorInfo parameter shall be completed with the value defined in clause 8.12.8 of ETSI TS 102 223 [6];
  - eDNS-RESOLUTION-ERROR: This parameter means that the destination FQDN could not be resolved by the DNS server. In this case, the aErrorInfo parameter shall be completed with the code value defined in IETF RFC 6895 [27], clause 2.3;

- eLINK-DROPPED: This parameter means that the Bearer Link of the TCP connection has dropped (due to network failure or user cancellation) as described in clause 7.5.11 of ETSI TS 102 223 [6];

- aErrorInfo: This parameter is the additional information associated to the aErrorCode.

10.4.6 TCP data service gate

10.4.6.1 Overview

The TCP data service gate provides access to a data stream between the TCP adapter and the TCP consumer.

The pipe session is opened when a requested TCP connection is successfully established. The TCP session shall be closed as soon as the pipe session is closed.

The TCP data service gate shall implement the credit-based data flow control mechanism and data acknowledgement defined in clause 8.5.3.

The pipes between the TCP adapter and the TCP consumer allow conveying a data stream. The SCL packets shall have the CB bit (Chaining bit) always set to 0.

10.4.6.2 Commands

The gate has no dedicated commands.

10.4.6.3 Responses

The gate has no dedicated responses.

10.4.6.4 Events

The gate has no dedicated events.

10.4.7 TCP data application gate

10.4.7.1 Overview

The provisions of the TCP data service gate defined in clause 10.4.6 shall apply also to this gate.

10.4.7.2 Commands

The gate has no dedicated commands.

10.4.7.3 Responses

The gate has no dedicated responses.

10.4.7.4 Events

The gate has no dedicated events.

10.4.8 Application protocols

10.4.8.1 HTTP(S) protocol

The SSP may support HTTP as defined in IETF RFC 7230 [18] or HTTPS as defined in IETF RFC 2818 [19] using the mechanism described in the clauses above.

10.4.8.2 TLS protocol

The SSP may support the TLS protocol using the mechanism described in the clauses above. If supported, TLS shall be compliant with IETF RFC 8446 [20].

10.5 User Datagram Protocol support

10.5.1 Overview

This clause defines the mechanism used by an SSP Application to send and receive UDP datagrams, as defined in IETF RFC 768 [23], to and from external entities. The SSP does not have an UDP/IP stack, but relies on the UDP/IP stack in the terminal.

Figure 10.5 illustrates a valid UDP platform.

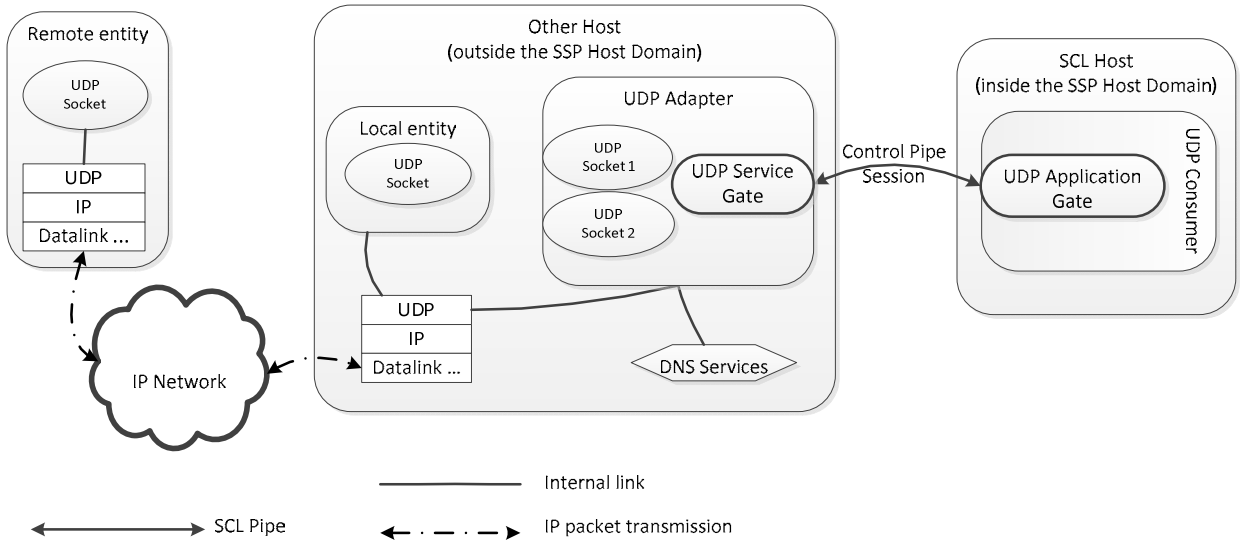

Figure 10.5: Valid UDP platform

The UDP adapter is a component of an SCL host residing outside the SSP host domain and interfaces to the UDP service gate. The UDP adapter interfaces to the UDP/IP stack of the terminal to communicated over an IP network. The UDP adapter may support the DNS resolution of Fully Qualified Domain Names using services of the terminal. The UDP adapter is in charge of:

- creating the UDP sockets;
- the resolution of DNS address, when this is required;
- transferring incoming UDP packets to the appropriate UDP application gate;
- sending outgoing UDP packets.

The UDP consumer resides in an SSP host and it uses the UDP application gate.

The UDP adapter shall contain only one UDP service gate.

If an SSP host supports the UDP protocol, it shall contain only one UDP consumer, which includes a single UDP application gate. The SCL host may implement logic for sharing the access to the UDP adapter between multiple SSP Applications.

The UDP service gate URN supports the syntax as defined in clause 8.2 with the values specified in table 8.1.

If the pipe session between the UDP service gate and the UDP application gate is closed, all UDP sockets shall be terminated.

10.5.2 Presentation layer

The UDP service and application gates implements an ASN.1 presentation layer using the definitions of the TCP service and application gates, as described in clause 10.4.3, with the additional following definitions.

```
-- ASN1START

SocketID ::= INTEGER (0..255)

-- ASN1STOP
```

10.5.3 UDP service gate

10.5.3.1 Overview

The UDP service gate provides access to services to manage UDP sockets and send UDP datagrams to remote entities on behalf of the UDP consumer. The UDP service gate may send an event to the UDP application gate while it is waiting for a response from it.

10.5.3.2 Commands

10.5.3.2.1 List of commands

The gate shall support the following commands.

```
-- ASN1START

UDP-SERVICE-GATE-Commands ::= [APPLICATION 2] CHOICE
{
    aUDP-REQUEST-SOCKET-Command UDP-REQUEST-SOCKET-Command,
    aUDP-CLOSE-SOCKET-Command UDP-CLOSE-SOCKET-Command
}

-- ASN1STOP
```

10.5.3.2.2 UDP-REQUEST-SOCKET-Command

With the command UDP-REQUEST-SOCKET-Command, a UDP consumer via the UDP application gate requests the UDP adapter via the UDP service gate to open a UDP port to send and receive UDP datagrams.

If the port number is not defined, the UDP adapter assigns an available port.

This command has the following parameters.

```
-- ASN1START

UDP-REQUEST-SOCKET-Command ::= [PRIVATE 16] SEQUENCE
{
    aPortNumber [1] INTEGER(1..65535) OPTIONAL,  -- UDP port on the terminal
    aNetworkParameters [2] NetworkParameters OPTIONAL,  -- Network parameters
    aLocalOnly [3] BOOLEAN OPTIONAL  -- if UDP datagrams from remote entities are allowed
}

-- ASN1STOP
```

Where:

- aPortNumber: it defines the UDP port number on the terminal. If the parameter is missing, the port will be automatically allocated by the UDP adapter;
- aNetworkParameters: it contains the parameters for the network connection required for the UDP socket to be created. The coding is the same as the NetworkParameters defined in clause 10.4.4.2.2;
- aLocalOnly: if it has value TRUE, then the UDP socket can only accept UDP datagrams from entities in the terminal. If it has value FALSE or it is not present, then the UDP socket can accept UDP datagrams from any remote entity.

When the requested socket is created successfully, then UDP service gate shall respond with eUDP-OK with the following parameters.

```
-- ASN1START

UDP-REQUEST-SOCKET-Response-Parameter ::= SEQUENCE
{
    aSocketID SocketID  -- Socket identifier
}

UDP-REQUEST-SOCKET-Response ::= [PRIVATE 16] SEQUENCE
{
    aUDP-Service-Response UDP-Service-Response DEFAULT eUDP-OK,
    aParameter UDP-REQUEST-SOCKET-Response-Parameter OPTIONAL
}

-- ASN1STOP
```

Where:

- aSocketID: this parameter is a unique identifier provided by the UDP adapter upon request to create a UDP socket by the UDP consumer. This identifier shall be unique across all UDP sockets.

10.5.3.2.3 UDP-CLOSE-SOCKET-Command

With the command UDP-CLOSE-SOCKET-Command a UDP consumer via the UDP application gate may request the UDP adapter via the UDP service gate to close the open UDP socket.

This command has the following parameters.

```
-- ASN1START

UDP-CLOSE-SOCKET-Command ::= [PRIVATE 17] SEQUENCE
{
    aSocketID SocketID  -- Socket identifier
}

-- ASN1STOP
```

Where:

- aSocketID: This parameter is the UDP socket identifier to close.

When successful the UDP application gate shall respond with eUDP-OK with following parameters.

```
-- ASN1START

UDP-CLOSE-SOCKET-Response-Parameter ::= SEQUENCE
{
    aSocketID SocketID  -- Socket identifier
}

UDP-CLOSE-SOCKET-Response ::= [PRIVATE 17] SEQUENCE
{
    aUDP-Service-Response UDP-Service-Response DEFAULT eUDP-OK,
    aParameter UDP-CLOSE-SOCKET-Response-Parameter OPTIONAL
}

-- ASN1STOP
```

Where:

- aSocketID: This parameter is the identifier of the closed UDP socket.

10.5.3.3 Responses

The gate shall support the responses defined as follows.

```
-- ASN1START

UDP-SERVICE-GATE-Responses ::= [APPLICATION 1] CHOICE
{
    aUDP-REQUEST-SOCKET-Response UDP-REQUEST-SOCKET-Response,
    aUDP-CLOSE-SOCKET-Response UDP-CLOSE-SOCKET-Response
}

-- ASN1STOP
```

The gate shall support the error codes defined in table 10.11 as follows.

```
-- ASN1START

UDP-Service-Response ::= ENUMERATED
{
    eUDP-OK (0),  -- No Error
    eUDP-E-CMD-PAR-UNKNOWN (2),  -- Unknown or illegal command parameter
    eUDP-E-NOK (3),  -- the command has failed
    eUDP-E-PORT-NOT-AVAILABLE (10)  -- The UDP port is not available
}

-- ASN1STOP
```

Table 10.11: Description of error codes

Error code	Description
eUDP-OK	Command completed successfully
eUDP-E-CMD-PAR-UNKNOWN	The format of the command parameters is wrong
eUDP-E-NOK	Command was rejected and/or not completed
eUDP-E-PORT-NOT-AVAILABLE	The UDP port is not available

Table 10.12: UDP service gate commands/responses

Command	eUDP-OK	eUDP-E-CMD-PAR-UNKNOWN	eUDP-E-NOK	eUDP-E-PORT-NOT-AVAILABLE
UDP-REQUEST-SOCKET-Command	•	•	•	•
UDP-CLOSE-SOCKET-Command	•	•	•	

10.5.3.4 Events

10.5.3.4.1 List of events

The UDP service gate supports the following events.

```
-- ASN1START

UDP-SERVICE-GATE-Events ::= [APPLICATION 0] CHOICE
{
    aEVT-UDP-DATAGRAM-OUT-Service-Event EVT-UDP-DATAGRAM-OUT-Service-Event
}

-- ASN1STOP
```

10.5.3.4.2 EVT-UDP-DATAGRAM-OUT-Service-Event

With the event EVT-UDP-DATAGRAM-OUT-Service-Event, the UDP consumer via the application gate requests the UDP adapter via the UDP service gate to send a UDP datagram.

The UDP consumer may request to send the UDP datagram by passing the IP address or the FQDN value of the server. If the FQDN value is used, the UDP adapter is responsible to perform the DNS resolution, if the feature is supported.

```
-- ASN1START

EVT-UDP-DATAGRAM-OUT-Service-Event ::= [PRIVATE 16] SEQUENCE
{
    aSocketID SocketID,  -- Socket identifier
    aDestinationAddress CHOICE  -- Destination address
    {
        aIP IPAddress,  -- IP address
        aFQDN FQDN  -- Fully Qualified Domain Name of the server
    },
    aDestinationPortNumber INTEGER(1..65535),  -- UDP port to send the UDP datagram
    aData OCTET STRING (SIZE (1..65507))  -- Data to send
}

-- ASN1STOP
```

Where:

- aSocketID: this parameter is the UDP socket identifier to send the UDP datagram;
- aDestinationAddress: it defines the destination address of the outgoing UDP datagram. It can be an IPv4 or IPv6 Address or Full Qualified Domain Name;
- aDestinationPortNumber: it defines the UDP destination port number;
- aData: this parameter contains the payload to be sent in the UDP datagram.

10.5.4 UDP application gate

10.5.4.1 Overview

The UDP application gate handles connection state change events, the events related to incoming UDP datagrams and the event associated to errors occurred while sending outgoing UDP datagrams.

10.5.4.2 Commands

The gate has no dedicated commands.

10.5.4.3 Responses

The gate has no dedicated responses.

10.5.4.4 Events

10.5.4.4.1 List of events

The UDP application gate supports the following events.

```
-- ASN1START

UDP-APPLICATION-GATE-Events ::= [APPLICATION 0] CHOICE
{
    aEVT-UDP-DATAGRAM-IN-Application-Event EVT-UDP-DATAGRAM-IN-Application-Event,
    aEVT-UDP-ERROR-Application-Event EVT-UDP-ERROR-Application-Event
}

-- ASN1STOP
```

10.5.4.4.2 EVT-UDP-DATAGRAM-IN-Application-Event

With the event EVT-UDP-DATAGRAM-IN-Application-Event, the UDP adapter via the UDP service gate conveys to the UDP consumer via the UDP application gate a datagram received on an open UDP socket.

```
-- ASN1START

EVT-UDP-DATAGRAM-IN-Application-Event ::= [PRIVATE 17] SEQUENCE
{
    asocketID SocketID,  -- Socket identifier
    aSourceIP IPAddress,  -- IP address of the incoming UDP datagram
    aSourcePortNumber INTEGER(1..65535),  -- UDP port of the incoming UDP datagram
    aData OCTET STRING (SIZE (1..65507))  -- Received data
}

-- ASN1STOP
```

Where:

- aSocketID: this parameter is the identifier of the UDP socket that received the incoming UDP datagram;
- aSourceIP: it contains the source IP address of the incoming UDP datagram. It can be an IPv4 or IPv6 Address;
- aSourcePortNumber: it indicates the source port number of the incoming UDP datagram;
- aData: this parameter contains the payload of the incoming UDP datagram.

10.5.4.4.3 EVT-UDP-ERROR-Application-Event

With the event EVT-UDP-ERROR-Application-Event, the UDP adapter via the UDP service gate notifies the UDP consumer via the UDP application gate that an error occurred.

```
-- ASN1START

EVT-UDP-ERROR-Application-Event ::= [PRIVATE 16] SEQUENCE
{
    aSocketID SocketID,  -- Socket identifier
    aErrorCode INTEGER  -- Error code
    {
        eUNREACHABLE (1),
        eREDIRECTION (2),
        eIP-HEADER-WRONG (4),
        eACCESS-TECHNOLOGY-ERROR (5),
        eTERMINAL-BUSY (6),
        eNETWORK-BUSY (7),
        eCALL-CONTROL-INTERACTION-ERROR (8),
        eDNS-RESOLUTION-ERROR (9),
        eLINK-DROPPED (10),
        eSOCKET-ID-INVALID (11)
    },
    aErrorInfo OCTET STRING OPTIONAL  -- Additional error information
}

-- ASN1STOP
```

Where:

- aSocketID: this parameter is the identifier of UDP socket to which the error is related;
- aErrorCode: this parameter is the identifier of the error code and it has one of the following values:
  - eUNREACHABLE: this value means that the destination IP address is unreachable as described in ICMP messages defined in IETF RFC 792 [26]. In this case, the aErrorInfo parameter shall be completed with the code value defined in clause "Destination Unreachable Message" of IETF RFC 792 [26];
  - eREDIRECTION: this value means that a redirection occurs in the route to convey the message as described in clause "Redirect Message" of IETF RFC 792 [26]. In this case, the aErrorInfo parameter shall be completed with the code value defined in clause "Redirect Message" of IETF RFC 792 [26];

- eIP-HEADER-WRONG: this value means that the message format is wrong as described in clause "Parameter Problem Message" of IETF RFC 792 [26]. In this case, the aErrorInfo parameter shall be completed with the code value defined in clause "Parameter Problem Message" of IETF RFC 792 [26];
- eACCESS-TECHNOLOGY-ERROR: this value means that the Access Technology was unable to process the command;
- eTERMINAL-BUSY: this value means that terminal is currently unable to process the command . The aErrorInfo parameter shall be completed with the value defined in clause 8.12.2 of ETSI TS 102 223 [6]. The term UICC shall be understood as SSP;
- eNETWORK-BUSY: this value means that the network is currently unable to process the command as described in clause 8.12.3 of ETSI TS 102 223 [6]. In this case, the aErrorInfo parameter shall be completed with the value defined in clause 8.12.3 of ETSI TS 102 223 [6]. The term UICC shall be understood as SSP;
- eCALL-CONTROL-INTERACTION-ERROR: this value means that the connection required to establish the UDP communication was blocked by the terminal due to the call control by NAA. In this case, the aErrorInfo parameter shall be completed with the value defined in clause 8.12.8 of ETSI TS 102 223 [6]. The term UICC shall be understood as SSP;
- eDNS-RESOLUTION-ERROR: this value means that the destination FQDN could not be resolved by the DNS server. In this case, the aErrorInfo parameter shall be completed with the code value defined in IETF RFC 6895 [27], clause 2.3;
- eLINK-DROPPED: this value means that the Bearer Link of the UDP connection has dropped (due to network failure or user cancellation) as described in clause 7.5.11 of ETSI TS 102 223 [6];
- eSOCKET-ID-INVALID: this value means that the UDP service notifies that the socket identifier is invalid;

- aErrorInfo: this parameter is the additional information associated to the aErrorCode.

10.5.5 Application protocols

10.5.5.1 CoAP over UDP Protocol

The SSP may support CoAP over UDP as defined in IETF RFC 7252 [24] using the mechanism described in the clauses above.

10.6 CRON service support

10.6.1 Overview

This clause defines a mechanism that allows an SSP host to register a timer in a CRON service in order to receive a notification (i.e. event) at a given time and date in the future. Figure 10.6 illustrates a valid platform supporting the CRON service.

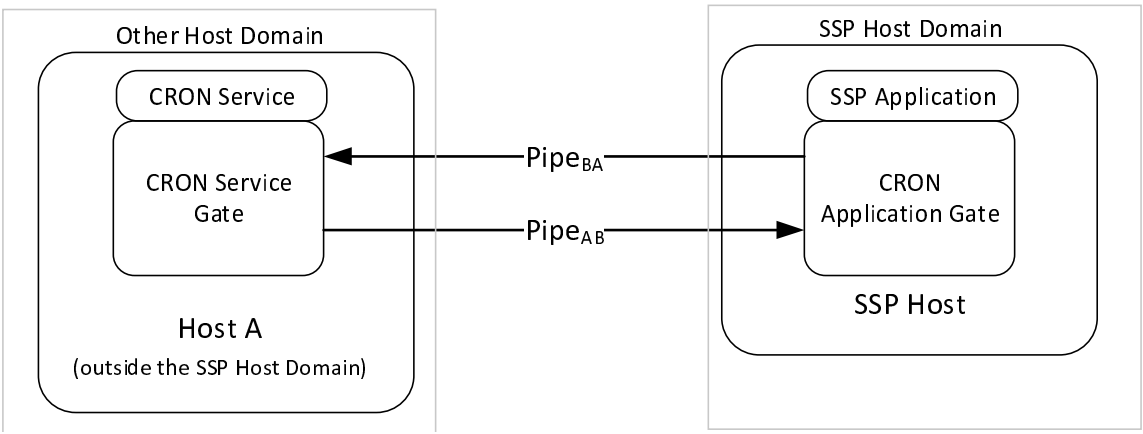

Figure 10.6: Valid platform supporting a CRON service

If the CRON service is supported by an SCL host outside the SSP host domain, then it shall contain only one CRON service gate.

An SCL host residing in the SSP host domain may contain one CRON application gate and shall not have any CRON service gate.

The CRON service gate URN supports the syntax as defined in clause 8.2 with the values specified in table 8.1.

10.6.2 Presentation layer

The CRON service gate and the CRON application gates implements an ASN.1 presentation layer using the following definitions.

```
-- ASN1START

/* CRON service: definitions*/
CRON-ID ::= INTEGER (0..255)

-- ASN1STOP
```

10.6.3 CRON service gate

10.6.3.1 Overview

The CRON service gate provides access to a CRON service that manages the notification at a given date and time.

The CRON service gate is in charge of:

- the registration of a timer containing the time and date information for a notification requested by the SSP Application;
- sending a notification (i.e. event) to the SCL host that requested the timer upon its expiration.

The time information used by the CRON service may not be reliable or accurate. SSP applications shall not rely on the time and date provided by the CRON service if they need an accurate source of time.

10.6.3.2 Commands

10.6.3.2.1 List of commands

The CRON service gate supports the following commands.

```
-- ASN1START

/* CRON Service: commands */
CRON-SERVICE-GATE-Commands ::= [APPLICATION 2] CHOICE
```

```
{
    aCRON-REQUEST-TIMER-Command CRON-REQUEST-TIMER-Command,
    aCRON-READ-DATE-Command CRON-READ-DATE-TIME-Command,
    aCRON-KILL-TIMER-Command CRON-KILL-TIMER-Command,
    aCRON-KILL-ALL-TIMERS-Command CRON-KILL-ALL-TIMERS-Command
}

-- ASN1STOP
```

10.6.3.2.2 CRON-REQUEST-TIMER-Command

With the command CRON-REQUEST-TIMER-Command, an SSP Application within the SSP host requests the CRON service to create a timer, in order to be notified when it expires.

The initial notification is sent either at an absolute time and date, or after a certain amount of time from the CRON-REQUEST-TIMER-Command.

```
-- ASN1START

CRON-REQUEST-TIMER-Command ::= [PRIVATE 16] SEQUENCE
{
    aInitialNotificationDateTime CHOICE  -- Time of the initial notification
    {
        aDateTimeAbsolute GeneralizedTime,  -- Absolute date and time
        aTimeRelative INTEGER (50..65535)  -- Time in units of 100 milliseconds from the time of the
request
    },
    aPeriod INTEGER (10..65535) OPTIONAL  -- Interval of periodic notifications after the initial
notification. The period is in units of 100 milliseconds
}

-- ASN1STOP
```

This command has the following parameters:

- aInitialNotificationDateTime: it indicates either a date and time in the future or a timer from the current time for the CRON service to notify the requesting SSP Application;
- aPeriod: if present, it defines the periodicity time of the notifications that will be issued periodically after aInitialNotificationDateTime, until the timer is killed.

If the SSP Application requests the timer at an absolute time and the CRON Service does not support it, then the CRON service shall reject the CRON-REQUEST-TIMER-Command, responding back eCRON-E-NO-ABSOLUTE-TIME.

If the SSP Application requests the timer at an absolute time in the past, then the CRON service shall reject the CRON-REQUEST-TIMER-Command, responding back eCRON-E-NOK.

When the CRON request is successful then CRON service gate shall respond with eCRON-OK with the following parameters.

```
-- ASN1START

CRON-REQUEST-TIMER-Response-Parameter ::= SEQUENCE
{
    aCRON-ID CRON-ID,  -- CRON id
    aPersistantOverPowerCycle BOOLEAN  -- Indication of persistence across power cycles
 }

CRON-REQUEST-TIMER-Response ::= [PRIVATE 16] SEQUENCE
{
    aCRON-Service-Response CRON-Service-Response DEFAULT eCRON-OK,
    aParameter CRON-REQUEST-TIMER-Response-Parameter OPTIONAL
}

-- ASN1STOP
```

Where:

- aCRON-ID: is the identifier of the CRON request;
- aPersistantOverPowerCycle: the value FALSE indicates that the timer is not persistent in case of power cycle of the terminal. The value TRUE indicates that the timer is persistent in case of power cycle of the terminal.

If a persistent timer expires while the terminal is powered off, then the terminal shall send the notification as soon as possible.

10.6.3.2.3 CRON-READ-DATE-TIME-Command

With the command CRON-READ-DATE-TIME-Command a CRON application gate may request to retrieve the UTCTime of the request.

```
-- ASN1START

CRON-READ-DATE-TIME-Command ::= [PRIVATE 17] SEQUENCE
{
}

-- ASN1STOP
```

This command has no parameters.

When successful the CRON Service shall respond with eCRON-OK with the following parameters.

```
-- ASN1START

CRON-READ-DATE-TIME-Response-Parameter ::= SEQUENCE
{
    aDateTime GeneralizedTime
}

CRON-READ-DATE-TIME-Response ::= [PRIVATE 17] SEQUENCE
{
    aCRON-Service-Response CRON-Service-Response DEFAULT eCRON-OK,
    aParameter CRON-READ-DATE-TIME-Response-Parameter OPTIONAL
}

-- ASN1STOP
```

Where:

- aDateTime: it is the UTCTime when the CRON service received the request.

10.6.3.2.4 CRON-KILL-TIMER-Command

With the command CRON-KILL-TIMER-Command a CRON application gate may request to kill a timer previously registered in the CRON service.

```
-- ASN1START

CRON-KILL-TIMER-Command ::= [PRIVATE 18] SEQUENCE
{
    aCRON-ID CRON-ID  -- CRON id
}

-- ASN1STOP
```

This command has the following parameters:

- aCRON-ID: identifier of the timer to kill.

When successful the CRON application gate shall respond with eCRON-OK with following parameters.

```
-- ASN1START

CRON-KILL-TIMER-Response ::= [PRIVATE 18] SEQUENCE
{
    aCRON-Service-Response CRON-Service-Response DEFAULT eCRON-OK
}

-- ASN1STOP
```

Non-periodic timers are implicitly killed from the CRON service when they expire, and the SSP Application within the SSP host does not need to kill them after receiving the associated notification.

10.6.3.2.5 CRON-KILL-ALL-TIMERS-Command

With the command CRON-KILL-ALL-TIMERS-Command a CRON application gate may request to kill all timers registered by an SSP host.

```
-- ASN1START

CRON-KILL-ALL-TIMERS-Command ::= [PRIVATE 19] SEQUENCE
{
}

-- ASN1STOP
```

This command has no parameters.

When successful the CRON application gate shall respond with eCRON-OK with no parameters.

```
-- ASN1START

CRON-KILL-ALL-TIMERS-Response ::= SEQUENCE
{
    aCRON-Service-Response CRON-Service-Response DEFAULT eCRON-OK
}

-- ASN1STOP
```

10.6.3.3 Responses

The gate supports the responses defined as follows.

```
-- ASN1START

/* CRON Service: responses */
CRON-SERVICE-GATE-Responses ::= [APPLICATION 1] CHOICE
{
    aCRON-REQUEST-TIMER-Response CRON-REQUEST-TIMER-Response,
    aCRON-READ-DATE-TIME-Response CRON-READ-DATE-TIME-Response,
    aCRON-KILL-TIMER-Response CRON-KILL-TIMER-Response,
    aCRON-KILL-ALL-TIMERS-Response CRON-KILL-ALL-TIMERS-Response
}

-- ASN1STOP
```

The gate supports the error codes defined in table 10.13 as follows.

```
-- ASN1START

CRON-Service-Response ::= ENUMERATED
{
    eCRON-OK (0),  -- No Error
    eCRON-E-CMD-PAR-UNKNOWN (2),  -- Unknown or illegal command parameter
    eCRON-E-NOK (3),  -- the command has failed
    eCRON-E-NO-ABSOLUTE-TIME (20)  -- the terminal does not support the absolute time
}

-- ASN1STOP
```

Table 10.13: Description of error codes

Error code	Description
eCRON-OK	Command completed successfully
eCRON-E-CMD-PAR-UNKNOWN	The format of the command parameters is wrong
eCRON-E-NOK	Command was rejected and/or not completed
eCRON-E-NO-ABSOLUTE-TIME	The terminal does not support the absolute time

Table 10.14: CRON control service gate commands/responses

Command	eCRON-OK	eCRON-E-CMD-PAR-UNKNOWN	eCRON-E-NOK	eCRON-NO-ABSOLUTE-TIME
CRON-REQUEST-TIMER-Command	•	•	•	•
CRON-KILL-TIMER-Command	•	•	•	
CRON-KILL-ALL-TIMERS-Command	•	•		
CRON-READ-DATE-TIME-Command	•			•

10.6.3.4 Events

The gate has no dedicated events.

10.6.4 CRON application gate

10.6.4.1 Commands

The gate has no dedicated commands.

10.6.4.2 Responses

The gate has no dedicated responses.

10.6.4.3 Events

10.6.4.3.1 List of events

The gate shall support the following events.

```
-- ASN1START

/* CRON Service: events */
CRON-APPLICATION-GATE-Events ::= [APPLICATION 0] CHOICE
{
    aCRON-ELAPSED-TIMER-Event CRON-ELAPSED-TIMER-Event
}

-- ASN1STOP
```

10.6.4.3.2 CRON-ELAPSED-TIMER-Event

With the CRON-ELAPSED-TIMER-Event, the CRON service notifies an SSP Application that a timer has elapsed.

```
-- ASN1START

CRON-ELAPSED-TIMER-Event ::= [PRIVATE 16] SEQUENCE
{
    aCRON-ID CRON-ID  -- CRON id
}

-- ASN1STOP
```

This event has the following parameters:

- aCRON-ID: it is the identifier of the timer that has elapsed.

10.7 Contactless related applications support

10.7.1 Introduction

This clause describes the emulation of an HCI host from an SSP host by using the SCL.

10.7.2 HCP tunnelling over SCL

10.7.2.1 Overview

This clause defines the gates for supporting the ETSI TS 102 622 [14] HCP tunnelling over SCL.

A SCL pipe session allows the tunnelling of HCP packets as defined in ETSI TS 102 622 [14] to/from HCI host controller as defined in ETSI TS 102 622 [14].

Figure 10.7 illustrates a platform supporting this gate.

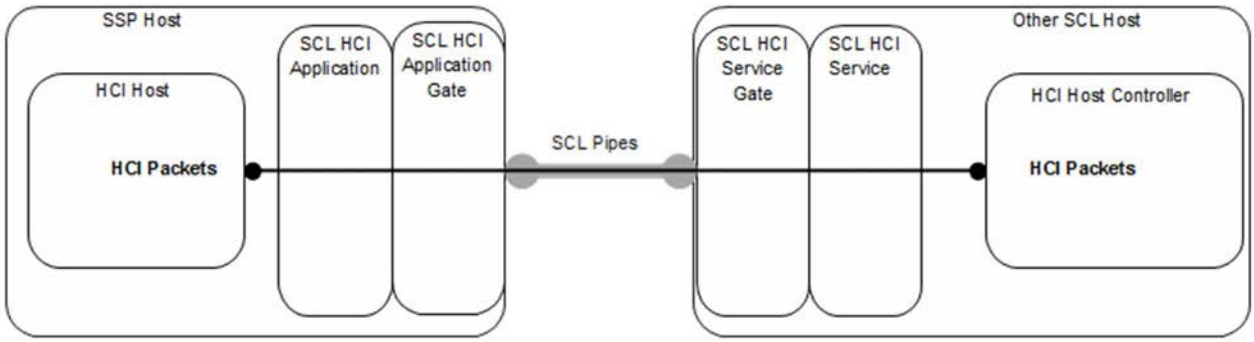

Figure 10.7: Platform supporting the HCI service

The SSP host shall at most support a single pipe session to HCI service gate.

The HCI protocol accommodation is based on the protocol tunnelling described in GlobalPlatform VPP - Network Protocol [13], clause B.3.

The presentation layer of the message conveyed over SCL is the HCP as defined in ETSI TS 102 622 [14], clause 5.1.

The session of the HCI protocol uses the session initialization defined in ETSI TS 102 622 [14], clause 8.4 with the assumption that the outcome of the identity check mechanism of the HCI lower layers is always successful.

NOTE: The possibility to skip the identity check is only applicable for non-removable SSPs. It is for further study if this rule needs to be modified for removable SSPs.

10.7.2.2 SCL HCI service gate

The SCL HCI service gate provides access to a SCL HCI fragmentation and reassembly service that manages the transfer of HCP packets as defined in ETSI TS 102 622 [14] from/to a CLF compliant with the ETSI TS 102 622 [14].

The SCL HCI Service shall embed the HCP packet from the HCI Host Controller in SCL message fragments of SCL packet to the SCL HCI application gate towards the SCL HCI application in the SSP host.

The gate has no commands, responses or events.

10.7.2.3 SCL HCI application gate

The SCL HCI application gate provides access to a SCL HCI application that emulates an HCI host as defined in ETSI TS 102 622 [14].

The SCL HCI application shall reassembly the HCP packet as defined in ETSI TS 102 622 [14] from the message fragments of SCL packets for the HCI host in the SSP host.

The gate has no commands, responses or events.

10.8 Card Application Toolkit (CAT) over SCL

10.8.1 Overview

This clause defines the gates for supporting the commands and responses corresponding to Card Application Toolkit defined in ETSI TS 102 223 [6] over SCL protocol. Figure 10.8 illustrates a platform supporting this gate.

The concept of envelope commands, envelope responses, proactive commands and terminal responses is as defined in ETSI TS 102 223 [6].

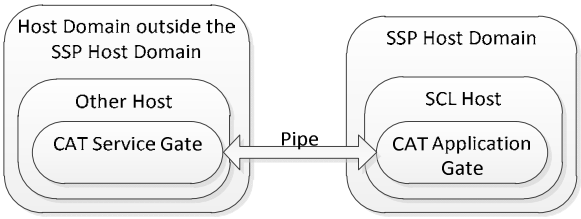

Figure 10.8: Platform supporting the CAT service

If the CAT application gate is supported, the CAPABILITY_EXCHANGE entry in the identity gate registry of the SCL host in the SSP shall indicate support for Card Application Toolkit.

The SSP host shall contain no more than one CAT application gate.

Each SSP host shall create no more than one pipe session to CAT service gates. If there are multiple hosts supporting the CAT service gate, the SSP host shall use the host in the MBM host domain, if present. If that is not present, the selection of the CAT service gate is implementation dependent.

The communication between the CAT service gate and the CAT application gate uses the presentation layer defined in ETSI TS 102 622 [14], clause 5.2.

When the SSP host has a pipe session to a CAT service gate, the SSP shall behave as described in clause 6.8.1.

10.8.2 Structure of Card Application Toolkit (CAT) communications

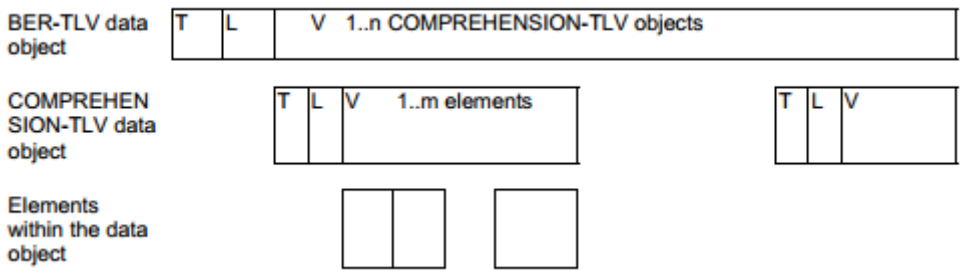

Figure 10.9: Data objects

CAT commands and responses are sent over the SCL pipe as BER-TLV data objects.

The tag of a BER-TLV is a constant value, length one byte, indicating it is a CAT command.

The length is coded onto 1, 2 or 3 bytes according to ETSI TS 101 220 [34], clause 7.1.2. Table 10.15 details this coding.

Table 10.15: Length encoding

Length	Byte 1	Byte 2	Byte 3
0 to 127	length ('00' to '7F')	not present	not present
128 to 255	'81'	length ('80' to 'FF')	not present
256 to 65 535	'82'	Length ('01 00' to 'FF FF')	

Any values for byte 1, byte 2 or byte 3 that are not shown above shall be treated as an error and the whole message shall be rejected.

NOTE: The byte 4 in ETSI TS 101 220 [34] is not used.

The value part of the BER-TLV data object consists of COMPREHENSION-TLV data objects, as shown in the description of the COMPREHENSION-TLV data objects on individual commands. It is mandatory for COMPREHENSION-TLV data objects to be provided in the order given in the description of each command. New COMPREHENSION-TLV data objects can be added to the end of a command.

The structure of COMPREHENSION-TLV tags is defined in ETSI TS 101 220 [34].

10.8.3 CAT application gate

10.8.3.1 Overview

This clause defines the events that the CAT application gate supports.

The events defined in the following clauses shall be sent to this gate.

10.8.3.2 Commands

The Gate has no dedicated commands.

10.8.3.3 Responses

The Gate has no dedicated responses.

10.8.3.4 Events

10.8.3.4.1 Supported events

The CAT application gate supports the events listed in table 10.16. The events are described in the following clauses. The states in which each event can be sent are defined in clause 10.8.5.

Table 10.16: Supported events

Value	Event
'10'	EVT_ENVELOPE_CMD
'11'	EVT_TERMINAL_RESPONSE

10.8.3.4.2 EVT_ENVELOPE_CMD

This event shall be used by the host outside the SSP in order to send an envelope command (as defined in ETSI TS 102 223 [6]) to the CAT application gate.

This event has the following parameter.

Table 10.17

Description	Length
Envelope command	N

The contents of the parameter are as defined in ETSI TS 102 223 [6], clause 7, with the exception of length parameters which is described in clause 10.8.2.

10.8.3.4.3 EVT_TERMINAL_RESPONSE

This event shall be used by the host outside the SSP in order to send a terminal response (as defined in ETSI TS 102 223 [6]) to the CAT application gate. This is a response to EVT_PROACTIVE_CMD sent by the CAT application gate.

This event has the following parameter.

Table 10.18

Description	Length
Terminal response	N

The contents of the parameter are as defined in ETSI TS 102 223 [6], clause 6.8, with the exception of length parameters which is described in clause 10.8.2.

10.8.3.5 Registry

The gate has no registry entry.

10.8.4 CAT service gate

10.8.4.1 Overview

This clause defines the events that a CAT service gate supports.

The CAT service gate may accept only an implementation specific maximum concurrent number of pipes from other SCL hosts.

The events defined in the following clauses shall be sent to this gate.

The CAT service gate URN supports the syntax as defined in clause 8.2, with the values specified in table 8.1.

10.8.4.2 Commands

The gate has no dedicated commands.

10.8.4.3 Responses

The gate has no dedicated responses.

10.8.4.4 Events

10.8.4.4.1 Supported events

The CAT service gate supports the events listed in table 10.19. The events are described in the following clauses. The states in which each event can be sent are defined in clause 10.8.5.

Table 10.19: Supported events

Value	Event
'10'	EVT_PROACTIVE_CMD
'11'	EVT_ENVELOPE_RSP

10.8.4.4.2 EVT_PROACTIVE_CMD

This event shall be used by the SSP host in order to send a proactive command (as defined in ETSI TS 102 223 [6]) to the CAT service gate.

This event has the following parameter.

Table 10.20

Description	Length
Proactive command	N

The contents of the parameter are as defined in ETSI TS 102 223 [6], clause 6.6, with the exception of length parameters which is described in clause 10.8.2.

10.8.4.4.3 EVT_ENVELOPE_RSP

This event shall be used by the SSP host in order to send an envelope response to the CAT service gate. This is a response to EVT_ENVELOPE_CMD sent by CAT service gate.

This event has the following parameter.

Table 10.21

Description	Length
Envelope response	N

The contents of the above parameter shall contain an optional response payload followed by SW1/SW2 status words as defined in ETSI TS 102 223 [6], clause 7. The length parameters for the optional response payload shall be as described in clause 10.8.2.

10.8.4.5 Registry

The gate has no registry entry.

10.8.5 State diagram for the CAT application gate

The states of the CAT application gate are:

- TK_ST_INIT: state of the gate when an open pipe exists to the gate but Capability Exchange indicates no terminal support for Card Application Toolkit.
- TK_ST_IDLE: state of the gate when no Toolkit commands are being processed.
- TK_ST_PCMD: state of the gate when one or more proactive commands are sent out and the terminal response is not yet received for all.
- TK_ST_ENV (transient): state of the gate when handling an envelope command.

The states and the transitions of the CAT application gate are shown in figure 10.10.

The CAT application gate shall only send proactive commands when it is in TK_ST_IDLE state.

NOTE: The dotted lines in figure 10.10 are shown to indicate the possibility to expand the protocol in future to support multiple proactive commands at the same time. This is similar to the CAT protocol over APDUs as defined in ETSI TS 102 223 [6] which allows for the possibility of two or more proactive commands in parallel, even if this is prohibited by ETSI TS 102 223 [6], clause 6.3.

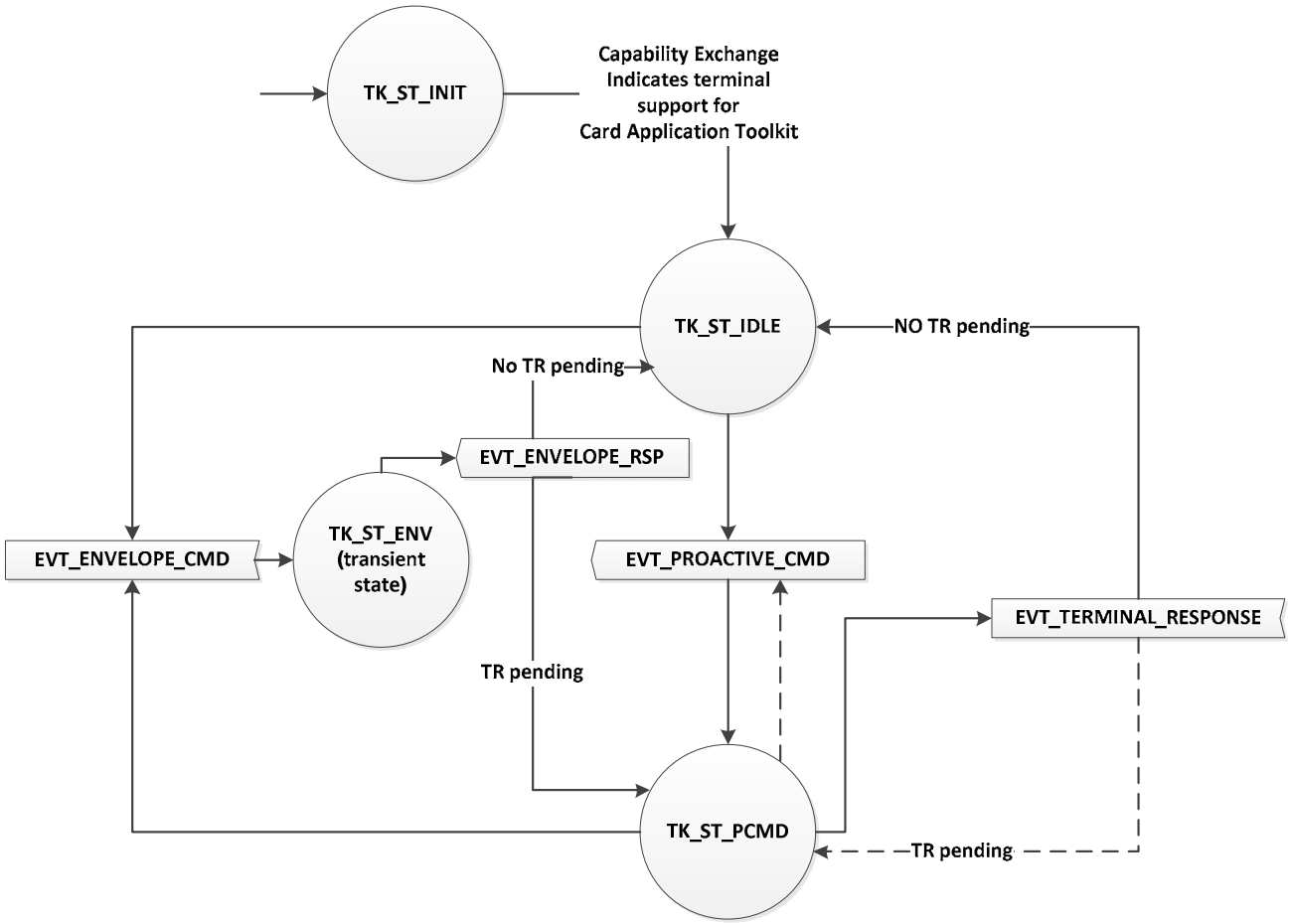

Figure 10.10: State transitions

10.9 Access control protocol

10.9.1 Introduction

The accessor authentication service allows authenticating an accessor to access SSP resources.

The SSP shall have a dedicated accessor authentication service gate for each accessor. The accessor authentication service gate is automatically created at the creation of the accessor. The identifier of the gate has the same value of the accessor identity, as defined in clause 6.13.4.3.

Hosts outside the SSP host domain may contain one or more accessor authentication application gates.

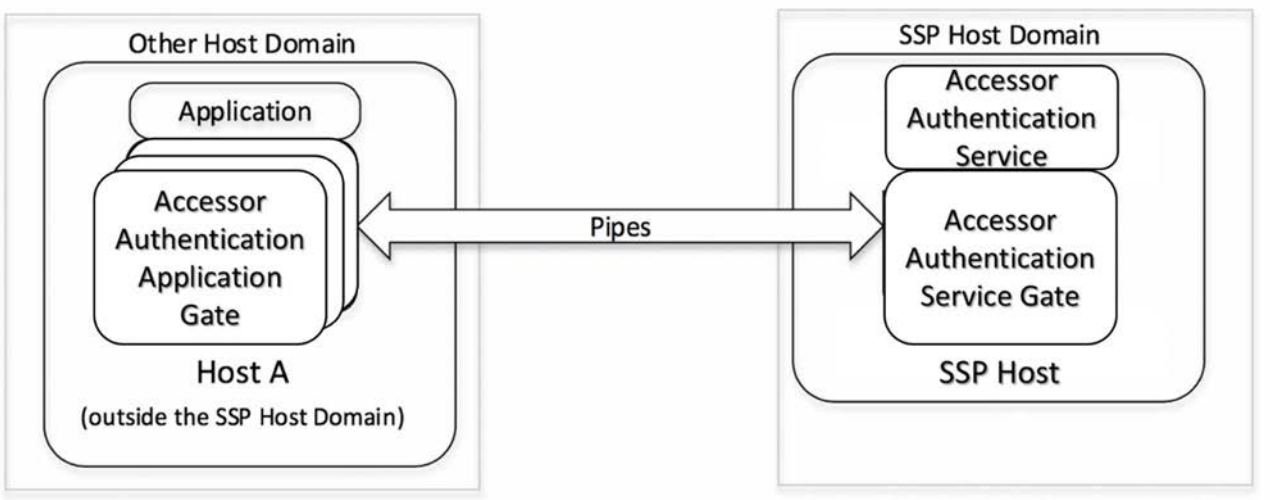

Figure 10.11: Accessor authentication service platform

The authentication of an accessor using a given pipe session shall not imply the authentication of the same accessor for a different host.

The closure of the pipe session to an accessor authentication service gate where the accessor was successfully authenticated shall result in the fact that the corresponding accessor is no longer authenticated for the host, without any impact on the authentication status of the same accessor for other hosts. The pipe sessions created using the accessor authentication service shall remain open.

NOTE 1: The closure of the pipe session has impact only on subsequent checks of the access control list. This implies that access to some resources will remain valid, if the check of the access control list was performed before the closure of the pipe session to the accessor authentication service.

After an accessor is successfully authenticated using a pipe session, it may be used to grant permissions to other accessors authenticated from any host, using the mechanism described in clause 6.13.2.

All pipe sessions to an accessor authentication service gate shall be closed by the accessor authentication service when the corresponding accessor is deleted. The accessor authentication service shall remove the accessor authentication service gate after closing the pipe sessions.

NOTE 2: If an accessor sends the command AAS-ADMIN-DELETE-ACCESSOR-Service-Command to delete itself, the accessor authentication service may not send the response for the command when successful, as the pipe session is closed.

If the credentials and/or conditions of an accessor are modified, the authentication status of the accessor is not affected.

10.9.2 Presentation layer

The accessor authentication control service gate and the accessor authentication control application gate implement an ASN.1 presentation layer using the definitions in clause 6.13.5.

10.9.3 Accessor authentication service gate

10.9.3.1 Overview

An accessor authentication application may access to the accessor authentication service via a pipe session between a accessor authentication application gate and a accessor authentication service gate.

10.9.3.2 Commands

The accessor authentication control service gate supports the following commands.

```
-- ASN1START

AAS-SERVICE-GATE-Commands ::= [APPLICATION 2] CHOICE
{
    aAAS-OP-GET-CAPABILITIES-Service-Command AAS-OP-GET-CAPABILITIES-Service-Command,
    aAAS-ADMIN-CREATE-ACCESSOR-Service-Command AAS-ADMIN-CREATE-ACCESSOR-Service-Command,
    aAAS-ADMIN-UPDATE-ACCESSOR-Service-Command AAS-ADMIN-UPDATE-ACCESSOR-Service-Command,
    aAAS-ADMIN-DELETE-ACCESSOR-Service-Command AAS-ADMIN-DELETE-ACCESSOR-Service-Command,
    aAAS-OP-ACCESS-SERVICE-Service-Command AAS-OP-ACCESS-SERVICE-Service-Command,
    aAAS-OP-AUTHENTICATE-ACCESSOR-Service-Command AAS-OP-AUTHENTICATE-ACCESSOR-Service-Command,
    aAAS-OP-GET-CHALLENGE-Service-Command AAS-OP-GET-CHALLENGE-Service-Command
}

-- ASN1STOP
```

All the commands are described in clause 6.13.4.

10.9.3.3 Responses

The gate shall support the responses defined as follows.

```
-- ASN1START

AAS-SERVICE-GATE-Responses ::= [APPLICATION 1] CHOICE
{
    aAAS-OP-GET-CAPABILITIES-Service-Response AAS-OP-GET-CAPABILITIES-Service-Response,
    aAAS-ADMIN-CREATE-ACCESSOR-Service-Response AAS-ADMIN-CREATE-ACCESSOR-Service-Response,
    aAAS-ADMIN-UPDATE-ACCESSOR-Service-Response AAS-ADMIN-UPDATE-ACCESSOR-Service-Response,
    aAAS-ADMIN-DELETE-ACCESSOR-Service-Response AAS-ADMIN-DELETE-ACCESSOR-Service-Response,
    aAAS-OP-ACCESS-SERVICE-Service-Response AAS-OP-ACCESS-SERVICE-Service-Response,
    aAAS-OP-AUTHENTICATE-ACCESSOR-Service-Response AAS-OP-AUTHENTICATE-ACCESSOR-Service-Response,
    aAAS-OP-GET-CHALLENGE-Service-Response AAS-OP-GET-CHALLENGE-Service-Response
}

-- ASN1STOP
```

All the responses are described in clause 6.13.6.

10.9.3.4 Events

The gate has no dedicated events.

10.9.4 Accessor authentication application gate

10.9.4.1 Overview

The accessor authentication application gate provides access to authenticate an accessor using the accessor authentication service gate.

10.9.4.2 Commands

The gate has no dedicated commands.

10.9.4.3 Responses

The gate has no dedicated responses.

10.9.4.4 Events

The gate has no dedicated events.

11 SSP classes

11.1 Overview

Multiple classes of SSP are defined, based on the form factor and hardware architecture, the communication protocol and other specific characteristics. Specific requirements for each SSP class are described in the corresponding specification, as defined in table 11.1.

Table 11.1: List of SSP classes

Class	Description	Specification
iSSP	SSP integrated in the SoC	ETSI TS 103 666-2 [8]
eSSP Type 1	SSP implemented as an embedded discrete component	ETSI TS 103 666-3 [44]
eSSP Type 2	SSP implemented as an embedded discrete component (based on Primary Platform / Secondary Platform architecture)	ETSI TS 103 666-4 [45]
rSSP	SSP implemented as a removable discrete component	For future release

Annex A (informative): Example of SCL flow

The provisions of GlobalPlatform VPP - Network Protocol [13], Annex A apply.

Annex B (informative): Support for UICC applications over SCL

B.1 UICC APDU service gate

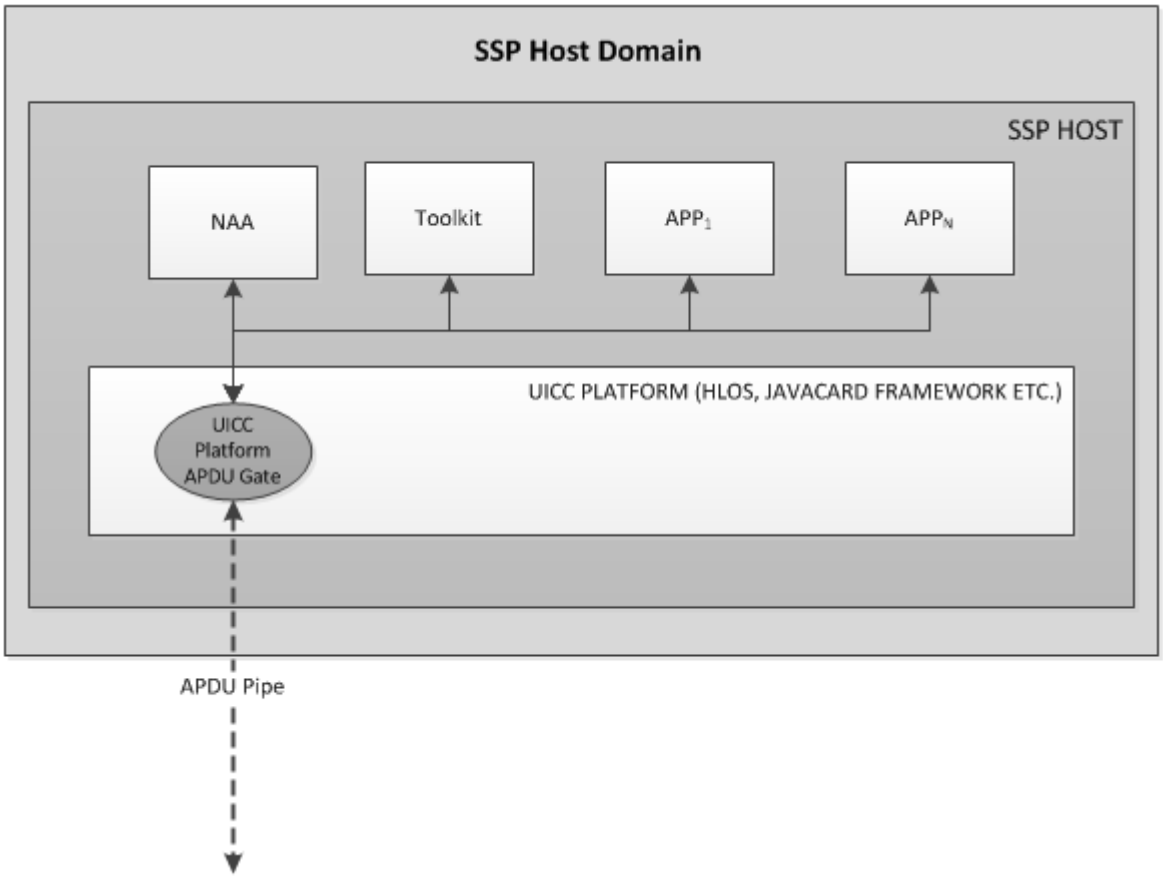

Figure B.1: UICC APDU service gate

The UICC platform within the SSP registers with the SCL router as a SCL host. It also defines the UICC APDU service gate to which SCL pipes are connected. Pipes transfer APDUs between applications in the UICC platform and an external host domain (e.g. the baseband processor).

All APDUs directed to any application are received by the UICC APDU service gate. Dispatch of APDUs to the correct application is based on AID and channel ID as defined in ETSI TS 102 221 [1].

The identifier of the UICC APDU service gate is described in table 8.1.

Annex C (normative): SCL secure pipe

C.1 Overview

This annex describes the parameters and functions that are required to implement the Secure SCL as described in clause 9.

The security operations are based on Elliptic-Curve Diffie-Hellman protocol (ECKA-DH) as defined in BSI Technical Guideline TR-03111 [36]. This allows two parties to share the same secret, necessary to establish a secure channel between them. Both parties shall share the same ECC domain parameters as defined in clause C.3.3.1 in order to be able to build the shared secret to derive a set of keys.

The generated authentication tokens are used to:

- Ensure the mutual authentication done by verifying their full certification path as well as the challenge they contain.
- Carry the parameters needed for encryption and decryption (algorithm and key size).

C.2 Authentication tokens

C.2.1 Description

Authentication tokens are certificates of ephemeral key pairs generated on demand by the Accessor Authentication Application and the Accessor Authentication Service.

The authentication tokens used by the Accessor Authentication Service and by the Accessor Authentication Application are based on a subset of a X.509 certificate version 3 and shall be signed by the end entity certificate of a certification path. There are two authentication tokens ATK.AAS.ECKA and ATK.AAA.ECKA have the AutenticationToken ASN.1 type:

- ATK.AAS.ECKA is the authentication token validated by the end entity certificate of $Certification\_Path_{ATK\_AAS}$. ATK.AAS.ECKA is signed by the private key associated to the end entity certificate of the $Certification\_Path_{ATK\_AAS}$.
- ATK.AAA.ECKA is the authentication token validated by the end entity certificate of $Certification\_Path_{ATK\_AAA}$. ATK.AAA.ECKA is signed by the private key associated to the end entity certificate of the $Certification\_Path_{ATK\_AAA}$.

C.2.2 Format

```
The ASN.1 description of the authentication token is the following:
-- ASN1START

AuthenticationToken ::= SEQUENCE
{
    tbsToken TBSToken,
    signatureAlgorithm AlgorithmIdentifier,
    signature ECDSA-Sig-Value
}

TBSToken ::= SEQUENCE
{
    version [0] Version DEFAULT v1,
    subjectPublicKeyInfo SubjectPublicKeyInfo,
    signatureAlgorithm AlgorithmIdentifier,
    aATK-Content ATK-Content,
```

```
    extensions [8] Extensions OPTIONAL
}

Key-Size ::= INTEGER
{
    e128 (0),  -- 128 Bit Key size
    e256 (1)  -- 256 Bit Key size
}

StreamCipherIdentifier ::= INTEGER
{
    aAES-CGM-StreamCipherIdentifier (0)  -- AES GCM algorithm
}

ATK-Content ::= SEQUENCE
{
    aChallenge OCTET STRING (SIZE (16)),  -- Challenge
    aKey-Size Key-Size,
    aStreamCipherIdentifier StreamCipherIdentifier
}

-- ASN1STOP
```

C.3 Certification paths

C.3.1 Description

The certification path is defined in IETF RFC 5280 [38], clause 3.2.

All certificates of a certification path shall use the same ECC domains (see clause C.3.3.1) and the same algorithms.

A different CI may be used for the verification of each certification path.

There are two certification paths. Certification paths are provisioned as part of the SSP. Certification paths are the following:

- The certification path of the application is Certification_Path$_{AAA}$.
- The certification path of the service is Certification_Path$_{AAS}$.

The authentication tokens are validated by the end entity certificate of their respective certification paths:

- ATK.AAA.ECKA authentication token is validated by the end entity of the Certification_Path$_{AAA}$.
- ATK.AAS.ECKA authentication token is validated by the end entity of the Certification_Path$_{AAS}$.

C.3.2 Format

The certificates shall be compliant with X.509 version 3 as described in IETF RFC 5280 [38] and shall support the DER format as defined in Recommendation ITU-T X.690 [16].

A certification path is as described in IETF RFC 5280 [38].

C.3.3 Identifiers and value used by certificate management

C.3.3.1 ECC domain parameters

At least one of the following curves shall be supported for the security functions:

- NIST P-256, defined in Digital Signature Standard [40].
- brainpoolP256r1, defined in IETF RFC 5639 [41].

C.3.3.2 Algorithm identifiers and parameters

This clause provides the values to be set in 'AlgorithmIdentifier.algorithm' and 'AlgorithmIdentifier.parameters' fields of the certificate as defined in IETF RFC 5280 [38] for each of the algorithms used in this annex.

For section 'subjectPublicKeyInfo' the following settings shall apply:

- 'AlgorithmIdentifier.algorithm' field shall be set to:
  - If the value of key usage is set to "digitalSignature(0)": "iso(1) member-body(2) us(840) ansi-X9-62(10045) keyType(2) id-ecPublicKey(1)" as defined in IETF RFC 5480 [39].
  - If the value of key usage is set to "keyAgreement(4)": "id-ecc key-establishment(5) 2" as defined in BSI Technical Guideline TR-03111 [36].
- '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 [41].
  - 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 [41].
  - For NIST P-256: "iso(1) member-body(2) us(840) ansi-X9-62(10045) curves(3) prime(1) secp256r1(7)" as defined in IETF RFC 5480 [39].
  - For NIST P-384: "iso(1) identified-organization(3) certicom(132) curve(0) secp384r1(34)" as defined in IETF RFC 5480 [39].

For sections 'signature' and 'signatureAlgorithm' the following settings shall apply:

- 'AlgorithmIdentifier.algorithm' field shall be set to: "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 [43].
- 'AlgorithmIdentifier.parameters' field shall be omitted as defined in IETF RFC 5758 [43], section 3.2.

C.3.3.3 Certificate policy OID

A Certification Policy in a certificate is represented by an OID as defined in IETF RFC 5280 [38]. The definition of OIDs for role identification using the ETSI root node 0.4.0.3666.1 is as follows.

```
-- ASN1START

id-ssp OBJECT IDENTIFIER ::= { itu-t (0) identified-organization (4) etsi (0) smart-secure-platform
(3666) part1 (1) }
id-role OBJECT IDENTIFIER ::= { id-ssp role (1) }
id-role-aaa OBJECT IDENTIFIER ::= { id-role aaa (1) }
id-role-aas OBJECT IDENTIFIER ::= { id-role aas (2) }
id-role-aas-ee OBJECT IDENTIFIER ::= { id-role-aas ee(1) }
id-role-aaa-ee OBJECT IDENTIFIER ::= { id-role-aaa ee(1) }

-- ASN1STOP
```

C.3.3.4 Certificate revocation

All online CAs may provide an OCSP interface as defined in IETF RFC 6960 [42]. All CA certificates may include the Authority Information Access according to IETF RFC 5280 [38], section 4.2.2.1.

When included in a certificate, the Authority Information Access Extension shall contain the AccessDescription identified by the object identifier id-ad-ocsp (OBJECT IDENTIFIER ::= { id-ad 1 }) and containing the URL of the OCSP interface.

All CA certificates shall be revocable individually.

All End Entity certificates shall not be revocable individually.

C.3.4 Long term keys

The Accessor Authentication Security Protocol requires the following long term keys that shall be provisioned as part of the SSP.

Table C.1: Long term keys

Grantor/Accessor	Name of key	Description
CI_{AAS}	$PK.CI_{AAS}$.ECDSA	Public key of the CI_{AAS}.
	$SK.CI_{AAS}$.ECDSA	Private key of the CI_{AAS}.
CI_{AAA}	$PK.CI_{AAA}$.ECDSA	Public key of the CI_{AAA}.
	$SK.CI_{AAA}$.ECDSA	Private key of the CI_{AAA}.
Accessor authentication service	PK.AAS.ECDSA	Public key of the accessor authentication service
	SK.AAS.ECDSA	Private key of the accessor authentication service
Accessor authentication application	PK.AAA.ECDSA	Public key of the accessor authentication application
	SK.AAA.ECDSA	Private key of the accessor authentication application

C.3.5 Functions

The management of the certificates and certificate paths needs the following functions.

Table C.2: Functions

Function	Description
ECDHE	Generates an ephemeral ECC key pair as defined in BSI Technical Guideline TR-03111 [36]
GENTOKEN	Builds an authentication token as defined in ANSI X9.63:2011 [35] and returns the new resulting certification path. Needs a challenge, a certification path and an ephemeral key pair as parameters
VERIFY_PATH	Verifies the validity of each certificate of a certification path. Returns true if all certificates are valid

C.4 Datagram encryption

C.4.1 Principle

Datagrams are encrypted and decrypted using keys derived from the shared secret generated with an anonymous Diffie-Helmann Key agreement (ECKA-DH) as defined in BSI Technical Guideline TR-03111 [36]. Key data are derived from the shared secret using $KDF_{X9.63}()$ as defined in BSI Technical Guideline TR-03111 [36]. The size of the key data generated is dependent on the algorithm used for the datagram encryption.

C.4.2 SharedInfo

$KDF_{X9.63}()$ as defined in BSI Technical Guideline TR-03111 [36] has a parameter, named SharedInfo that is built as follows:

- $SI = KEY\_TYPE^{8BIT} || ALGORITHM^{8BIT} || KEY\_LENGTH^{8BIT} || DIVERSIFIER^{128BIT}$

where:

- KEY_TYPE is the type of the key as defined in table C.3.
- ALGORITHM = '90' for the AES-GCM-128/256 algorithm.
- KEY_LENGTH is the length of the key in bytes as defined in table C.3.
- DIVERSIFIER is a parameter for diversifying the key data according to a context.

Table C.3 defines the parameters for KDF according to the keys and their length.

Table C.3: Key parameters for AES-GCM-128/256

Size of the key	KEY_LENGTH	KEY_TYPE
128 bits	'10'	'10'
256 bits	'20'	'20'

C.4.3 Encryption/decryption

The encryption is done using AES-GCM-128 or AES-GCM-256 as defined in ANSI X9.63 [35] depending on the key length that is used.

The keys are extracted from the key data generated thanks to KDF algorithm as defined in FIPS PUB 180-4 [37].

The length of the key data to generate depends on the key size to be used:

$$\text{Key data} = K \mid IV \mid \text{padding}$$

For each secure SCL message, IVC is derived from the IV extracted from the Key Data.

$$IVC = \text{AES-128}[K](SEQ^{32BIT} \mid IV^{96BIT})$$

or

$$IVC = \text{AES-256}[K](SEQ^{32BIT} \mid IV^{96BIT})$$

IVC is truncated to the first 96 bits and shall be used as IV parameter for the AES-GCM-128 or AES-GCM-256.

Annex D (informative): TCP service procedures

D.1 Connection request in active mode

Figure D.1 illustrates the procedure used by a TCP consumer to request a TCP adapter to connect to a TCP server using the TCP control service gate and the TCP control application gate.

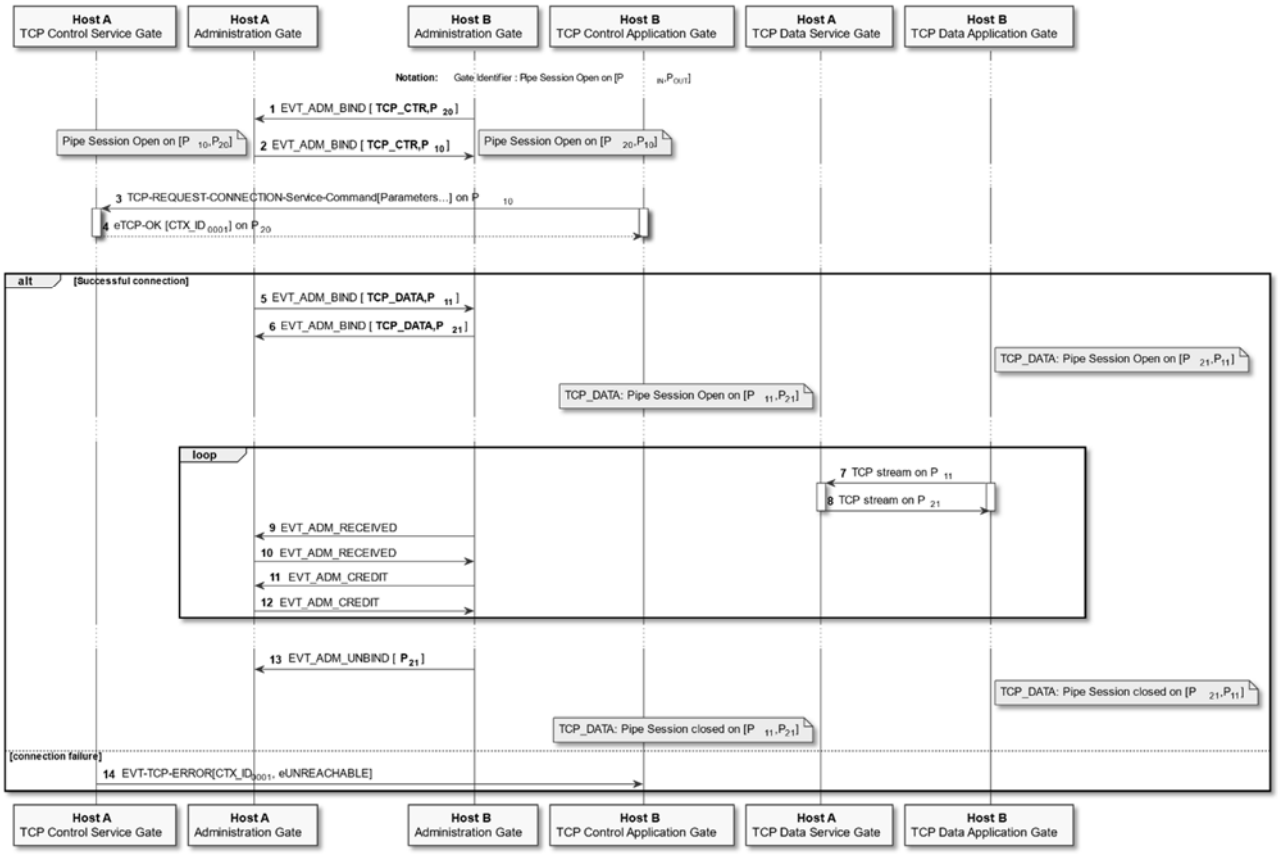

Figure D.1: TCP connection active mode

The procedure has 14 steps:

1) The host B requests the opening of a pipe session on the TCP control service gate of the host A.

2) The host A confirms this pipe session opening.

3) The host B requests a TCP connection by sending the command TCP-REQUEST-CONNECTION-Service-Command.

4) The host A confirms the request and returns the identifier of the TCP connection.

5) The connection is successful then the host A requests the opening of a pipe session to a TCP data application gate of the host B.

6) The host B confirms the pipe session opening.

7) The TCP data application gate of the host B sends a data stream to TCP data service gate of the host A by using the pipe P_{11}.

8) The TCP data service gate of the host A sends a data stream to TCP data application gate of the host A by using the pipe P_{21}.

9) The administration gate of the host B reports to the administration gate of the host A the length of the data received from the pipe P_{11}.

10) The administration gate of the host B reports to the administration gate of the host A the length of the data received from the pipe P_{21}.

11) The administration gate of the host B notifies the credit for the P_{21} to the administration gate of the host A.

12) The administration gate of the host A notifies the credit for the P_{11} to the administration gate of the host B. The procedure may reiterate to the step 7.

13) The host B notifies the host A about the pipe session closing for Pipe P_{11} and P_{21}.

14) If the TCP connection has failed then the host A notifies the host B about the connection error.

D.2 Connection request in passive mode

Figure D.2 illustrates the procedure used by a TCP consumer to request a TCP adapter to open a TCP socket for incoming TCP connections using the TCP control service gate and the TCP control application gate.

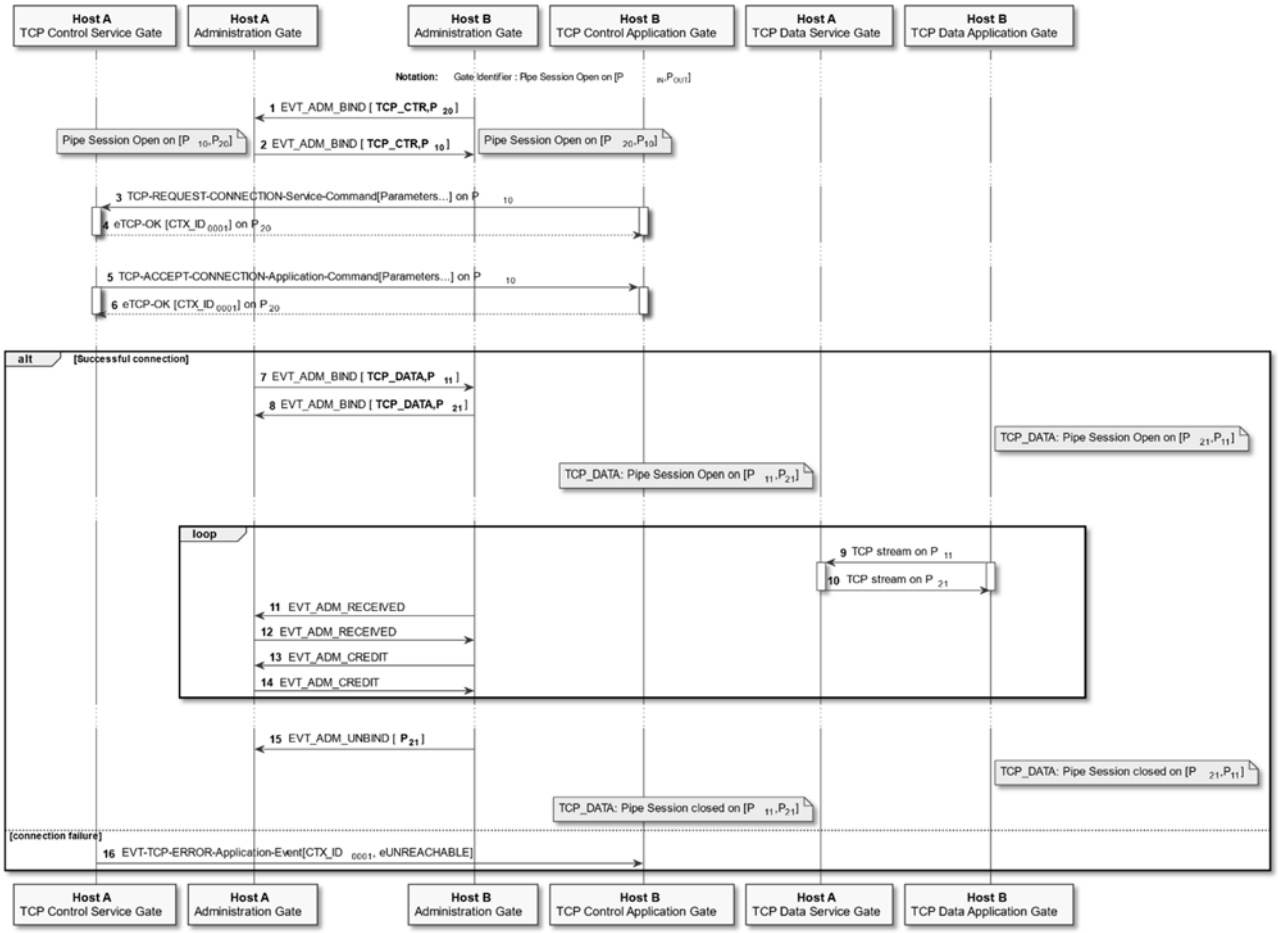

Figure D.2: TCP connection passive mode

The procedure has 16 steps:

1) The host B requests the opening of a pipe session on the TCP control service gate of the host A.

2) The host A confirms this pipe session opening.

3) The host B requests a TCP connection by sending the command TCP-REQUEST-CONNECTION-Service-Command.

4) The host A confirms the request and returns the identifier of the TCP connection.

5) When the host A receives an input TCP connection, the host A requests the TCP connection acceptance from the host B.

6) The host B confirms the acceptance to the host A.

7) The connection is successful then the host A requests the opening of a pipe session to a TCP data application gate of the host B.

8) The host B confirms the pipe session opening.

9) The TCP data application gate of the host B sends a data stream to TCP data service gate of the host A by using the pipe P_{11}.

10) The TCP data service gate of the host A sends a data stream to TCP data application gate of the host A by using the pipe P_{21}.

11) The administration gate of the host B reports to the administration gate of the host A the length of the data received from the pipe P_{21}.

12) The administration gate of the host A reports to the administration gate of the host B the length of the data received from the pipe P_{11}.

13) The administration gate of the host B notifies the credit for the P_{21} to the administration gate of the host A.

14) The administration gate of the host A notifies the credit for the P_{11} to the administration gate of the host B. The procedure may reiterate to the step 9.

15) The host B notifies the host A about the pipe session closing for pipe P_{11} and P_{21}.

16) If the TCP connection has failed, e.g. IP node unreachable, then the Host A notifies the Host B about the connection error.

D.3 Connection request failure

Figure D.3 illustrates the procedure used by a TCP consumer to request a TCP adapter to connect to a TCP server using the TCP control service gate and the TCP control application gate, in the case where this fails.

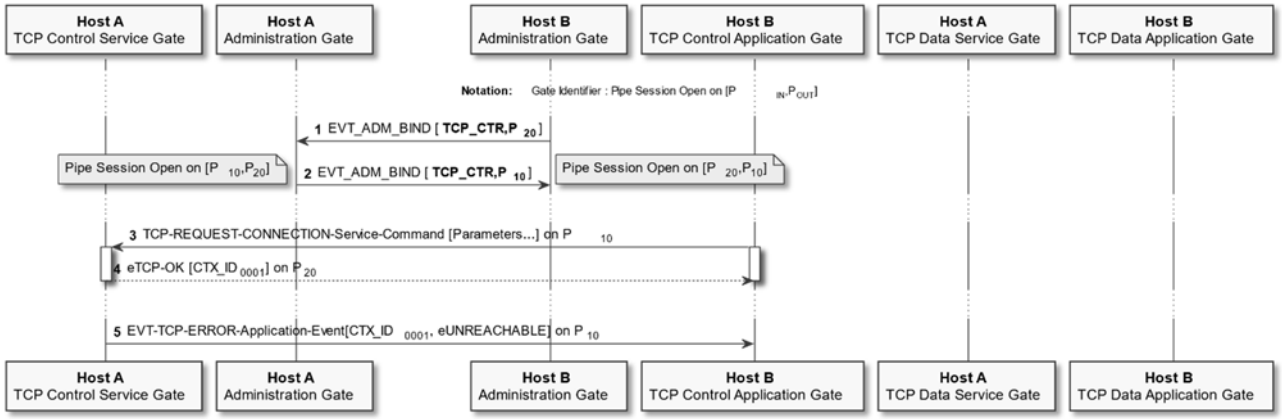

Figure D.3: TCP connection failure

The procedure has 5 steps:

1) The host B requests the opening of a pipe session on the TCP control service gate of the host A.
2) The host A confirms this pipe session opening.
3) The host B requests a TCP connection by sending the command TCP-REQUEST-CONNECTION-Service-Command.
4) The host A confirms the request and returns the identifier of the TCP connection.
5) If the TCP connection has failed then the host A notifies the host B about the connection error.

D.4 Connection closing

Figure D.4 illustrates the procedure in which a host requires the closing of a TCP connection in passive mode.

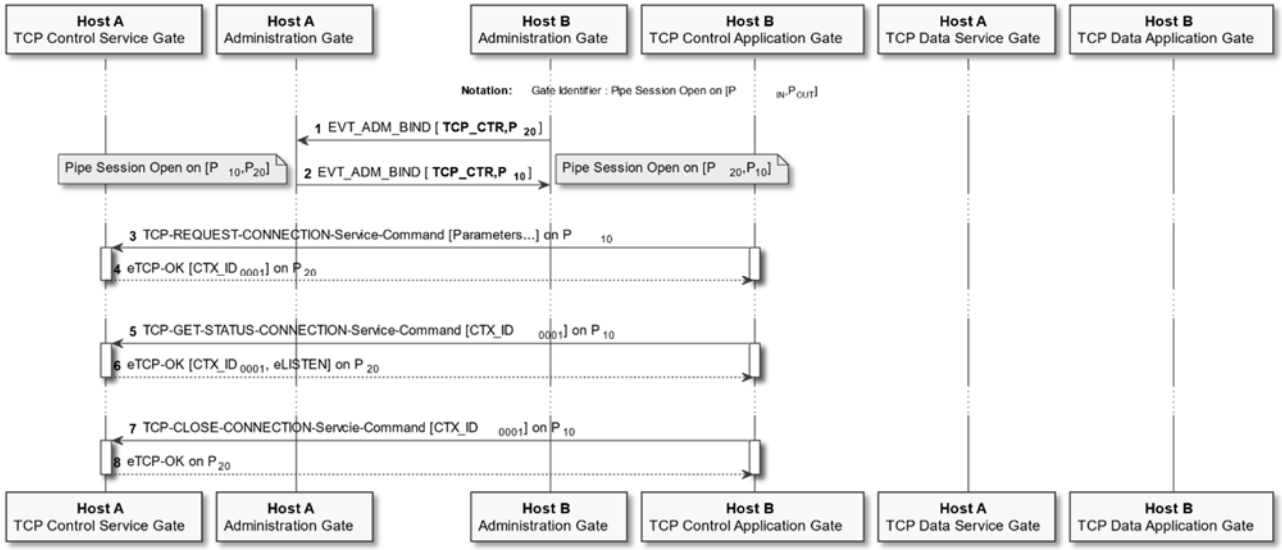

Figure D.4: TCP connection closing

The procedure has 8 steps:

1) The host B requests the opening of a pipe session on the TCP control service gate of the host A.
2) The host A confirms this pipe session opening.
3) The host B requests a TCP connection by sending the command TCP-REQUEST-CONNECTION-Service-Command.
4) The host A confirms the request and returns the identifier of the TCP connection.
5) The host B requests the status of TCP connection to the host A.
6) The host A reports the status of the TCP connection.
7) The host B requests the closing of TCP connection to the host A.
8) The host A confirms the closing of the TCP connection.

Annex E (informative): UDP service procedures

E.1 UDP socket request

Figure E.1 illustrates the procedure in which the UDP consumer requests a UDP socket from the UDP adapter.

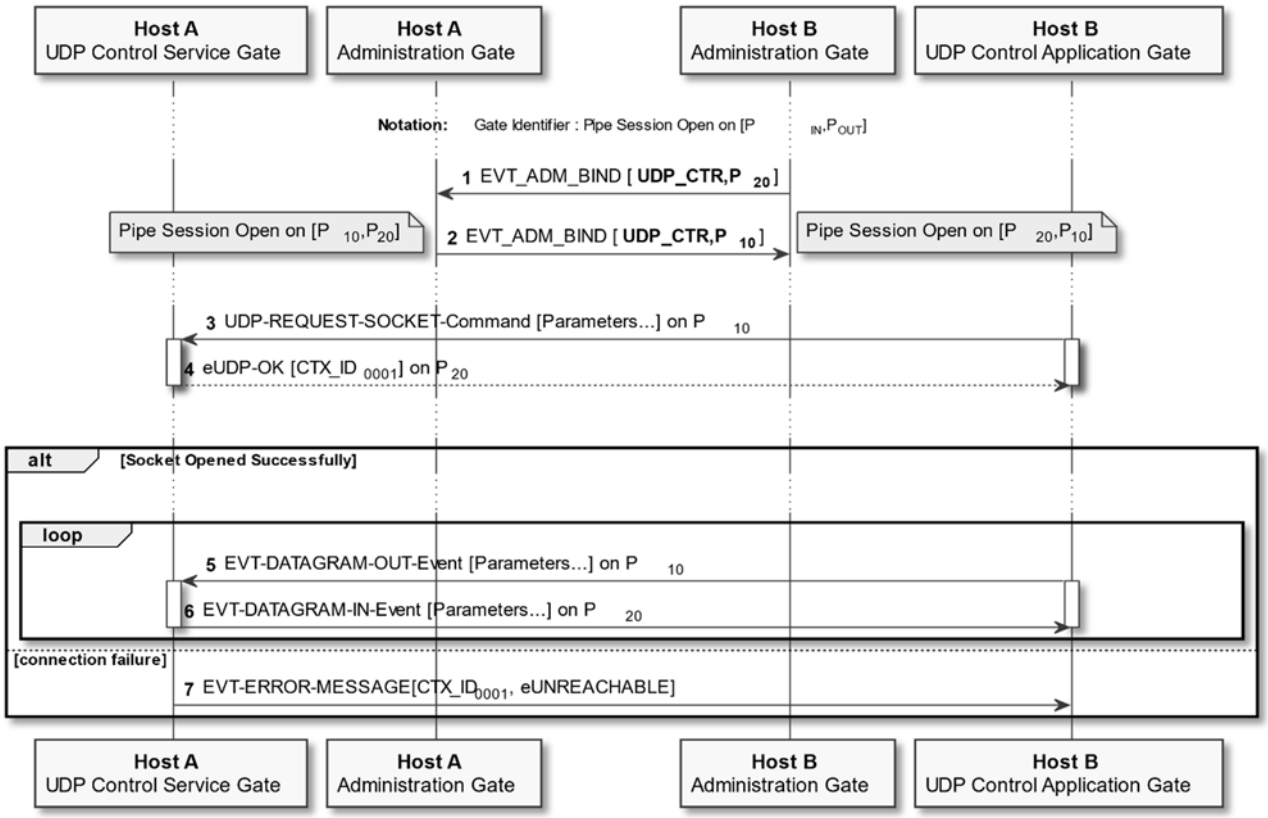

Figure E.1: UDP socket request

The procedure has 7 steps:

1) The host B requests the opening of a Pipe session on the UDP service gate of the host A.

2) The host A confirms this pipe session opening.

3) The host B requests a UDP socket by sending the command UDP-REQUEST-SOCKET-Command.

4) The host A confirms the request and returns the identifier of the UDP socket.

If the UDP socket was created successfully, steps 5 and 6 are executed and may be reiterated multiple times:

5) The host B sends a datagram via host A by sending the event EVT-UDP-DATAGRAM-OUT-Service-Event using as parameters the identifier of the UDP socket and the datagram data.

6) The host B receives a datagram via host A by receiving the event EVT-UDP-DATAGRAM-IN-Application-Event using as parameters the identifier of the UDP socket and the datagram data.

Otherwise, if the creation of the UDP socket has failed, step 7 is executed:

7) If the UDP connection has failed then the host A notifies the host B about the transmission error.

E.2 Socket closing

Figure E.2 illustrates the procedure in which the UDP consumer requests to close a UDP socket.

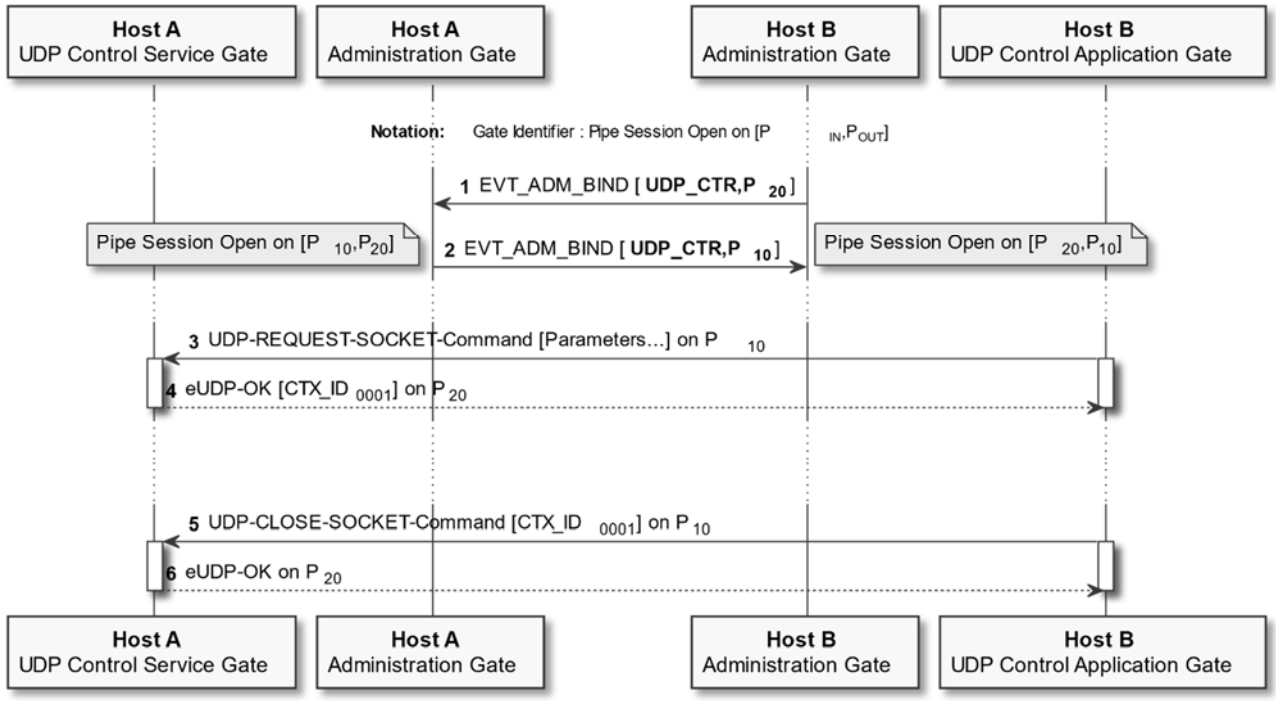

Figure E.2: UDP socket closing

The procedure has 6 steps:

1) The host B requests the opening of a pipe session on the UDP service gate of the host A.
2) The host A confirms this pipe session opening.
3) The host B requests a UDP socket by sending the command UDP-REQUEST-SOCKET-Command.
4) The host A confirms the request and returns the identifier of the UDP socket.
5) The host B requests the closing of UDP socket to the host A.
6) The host A confirms the closing of the UDP socket.

Annex F (informative): CRON service procedures

F.1 CRON timer request

Figure F.1 illustrates the procedure in which an SSP Application within the SSP host can request a timer.

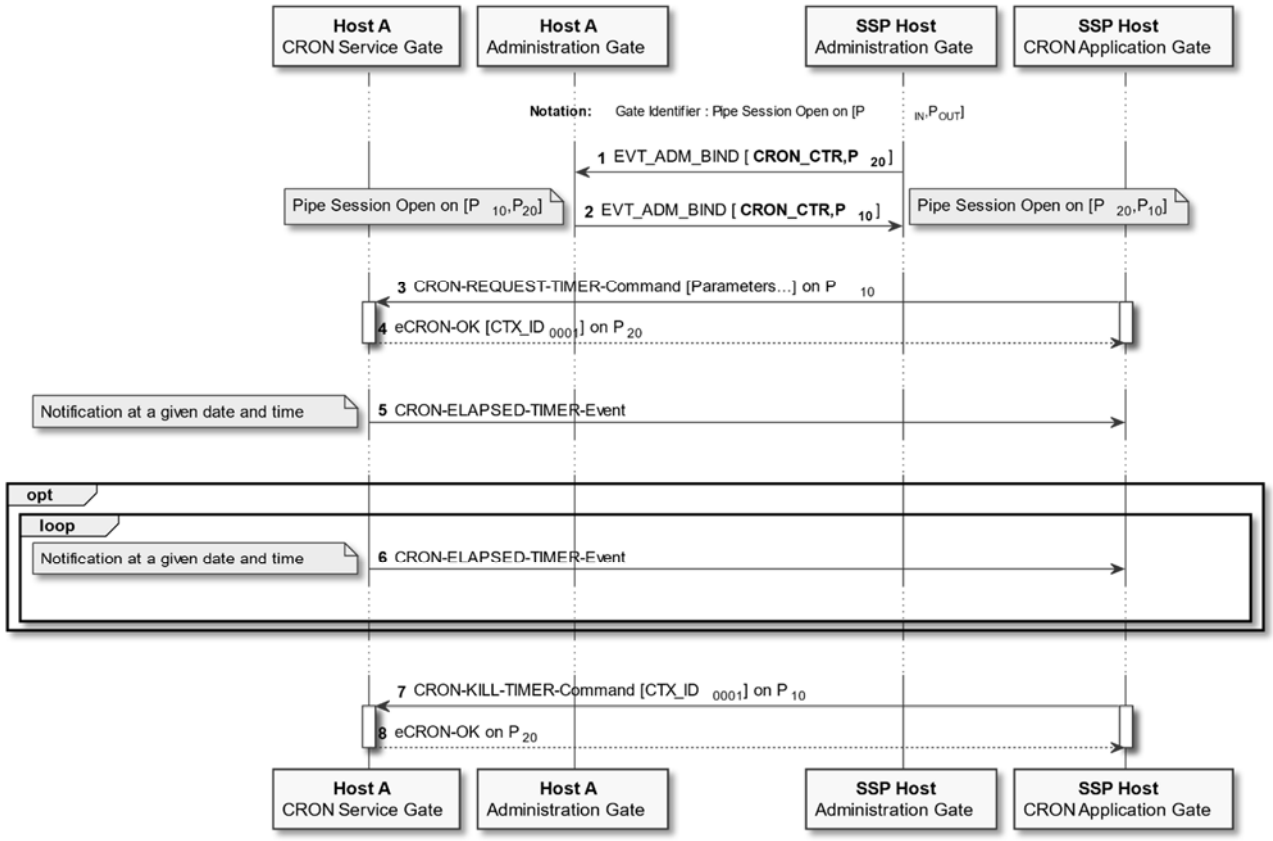

Figure F.1: CRON timer request

The procedure has 8 steps:

1) The SSP Application within the SSP host requests the opening of a pipe session on the CRON service gate of the host A.

2) The host A confirms this pipe session opening.

3) The SSP Application within the SSP host requests a (periodic) timer sending the command CRON-REQUEST-TIMER-Command.

4) The CRON service confirms the request and returns the identifier of the timer.

5) The CRON service notifies the SSP Application within the SSP host.

6) If a periodic event is required, the CRON service notifies periodically the SSP Application within the SSP host domain.

7) The SSP Application within the SSP host domain requests the killing of the timer.

8) The CRON service confirms the killing of the timer.

F.2 CRON timer killing

Figure F.2 illustrates the procedure in which a host requires to kill a timer within the CRON service.

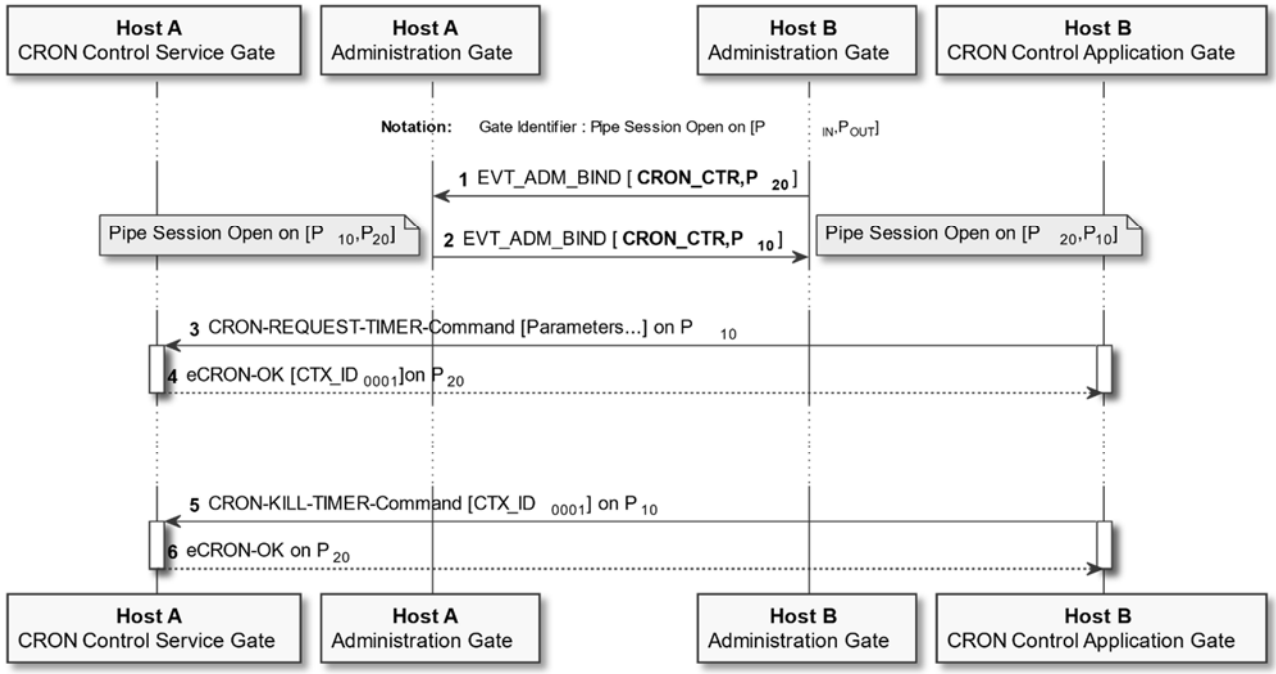

Figure F.2: CRON timer killing

The procedure has 6 steps:

1) The host B requests the opening of a pipe session on the CRON service gate of the host A.
2) The host A confirms this pipe session opening.
3) The host B requests a timer by sending the command CRON-REQUEST-TIMER-Command.
4) The CRON service confirms the request and returns the identifier of the timer.
5) The host B requests the killing of a timer to the host A.
6) The CRON Service confirms the killing of the timer.

F.3 CRON read date command

Figure F.3 illustrates the procedure in which a host requires reading the date and time from the CRON service gate.

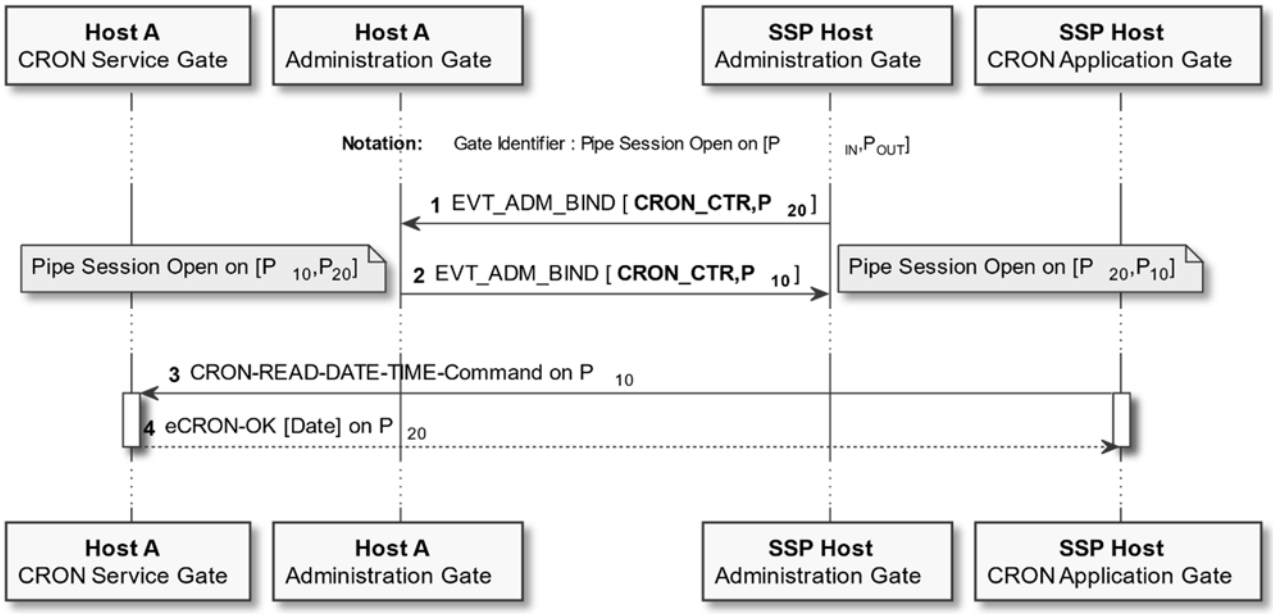

Figure F.3: CRON Read Date

The procedure has 4 steps:

1) The SSP Application within the SSP host domain requests the opening of a pipe session on the CRON service gate of the host A.

2) The host A confirms this pipe session opening.

3) The SSP Application within the SSP host domain requests reading the date from the CRON service gate with the command CRON-READ-DATE-TIME-Command.

4) The CRON service returns the date and the time.

Annex G (informative): File system protocol service procedures

G.1 File reading

Figure G.1 illustrates the procedure in which a host requiring the file system to read data from an SSP file.

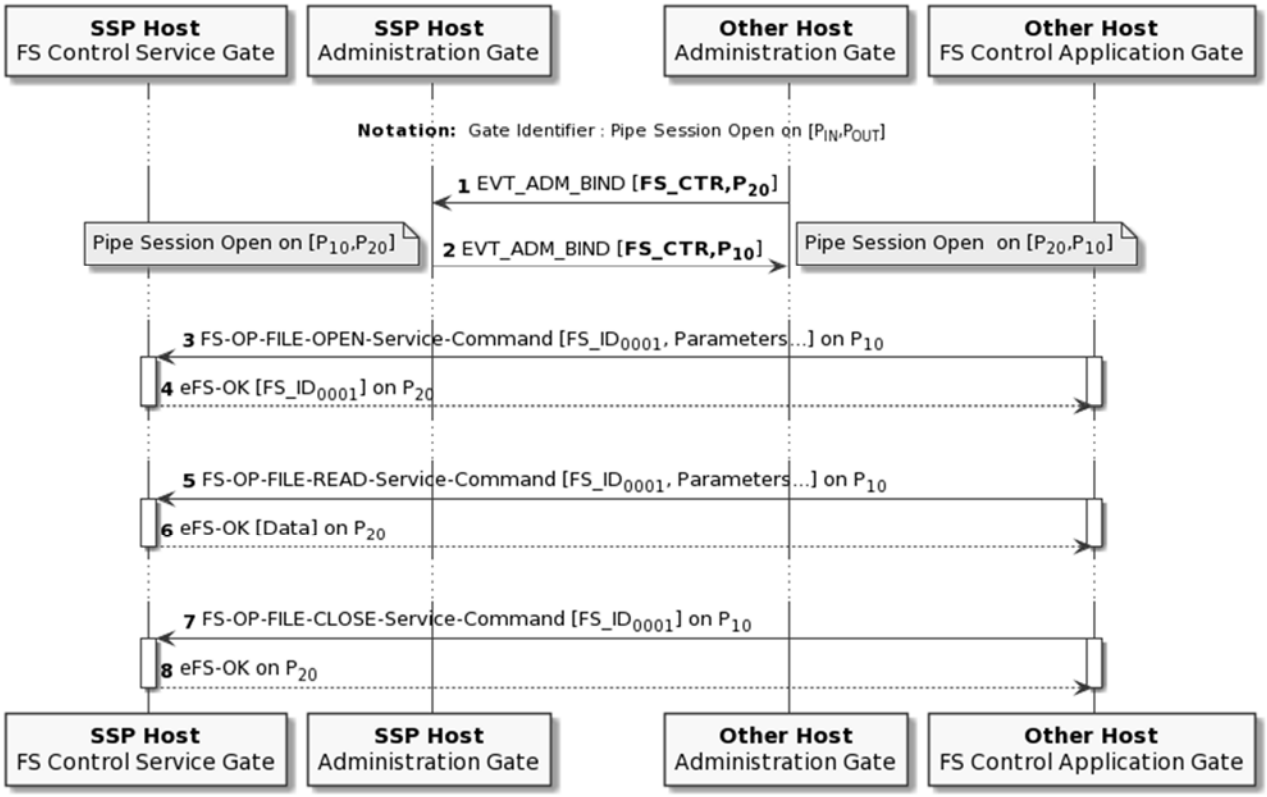

Figure G.1: File reading

The procedure has 8 steps:

1) The other host requests the opening of a pipe session on the file system control service gate of the SSP host.

2) The SSP host confirms this pipe session opening.

3) The other host requests the opening of a node by sending the command FS-OP-FILE-OPEN-Service-Command.

4) The SSP host confirms the command and returns the file session identifier.

5) The other host requests a reading of data from a node by sending the command FS-OP-FILE-READ-Service-Command.

6) The SSP host confirms the command and returns the data.

7) The other host requests the closing of the file session by sending the command FS-OP-FILE-CLOSE-Service-Command.

8) The SSP host confirms the command.

G.2 File writing

Figure G.2 illustrates the procedure in which a host requiring the file system to write data to an SSP file.

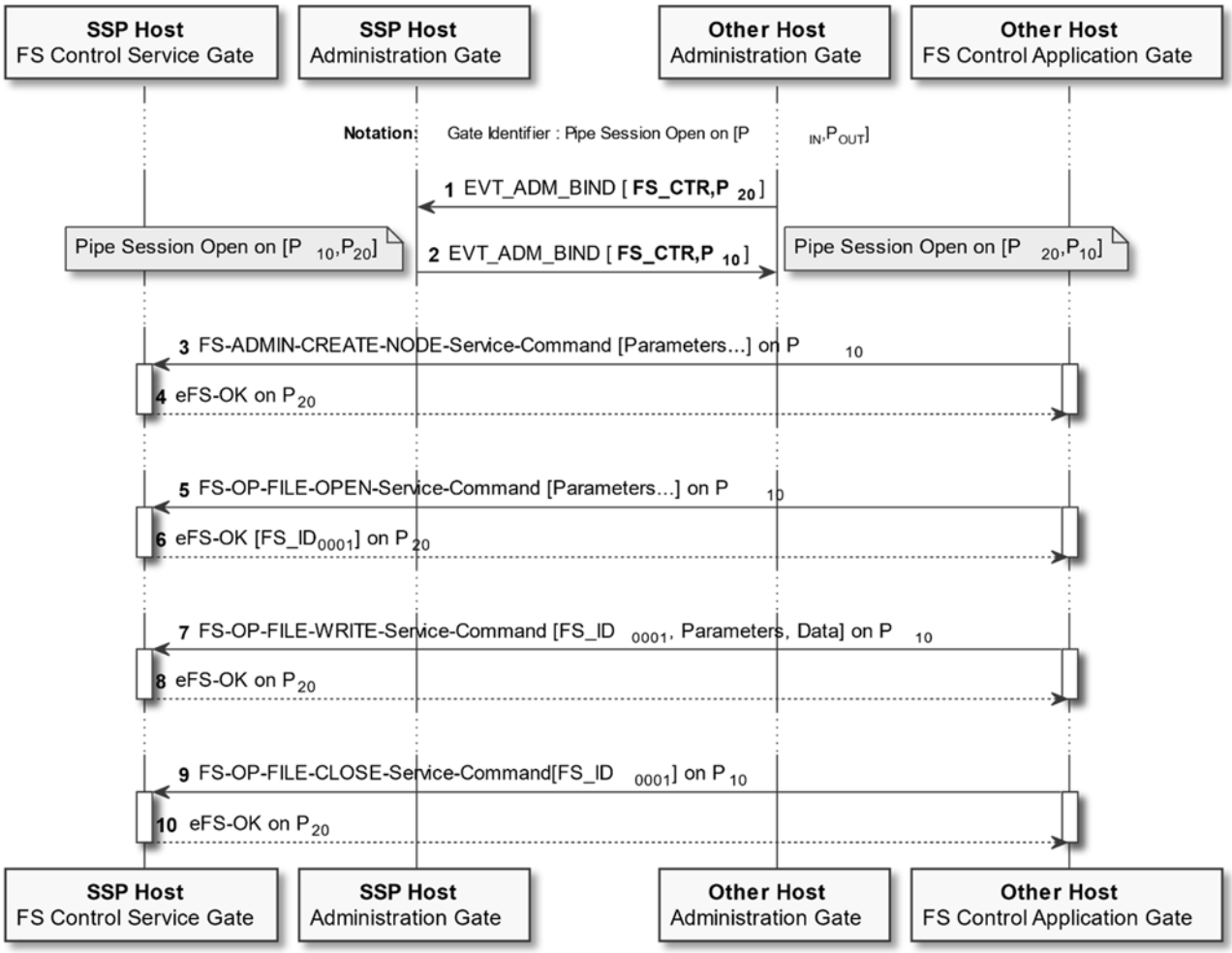

Figure G.2: File writing

The procedure has 10 steps:

1) The other host requests the opening of a pipe session on the file system control service gate of the SSP host.
2) The SSP host confirms this pipe session opening.
3) The other host requests a creation of a node by sending the command FS-ADMIN-CREATE-NODE-Service-Command.
4) The SSP host confirms the command.
5) The other host requests the opening of a node by sending the command FS-OP-FILE-OPEN-Service-Command.
6) The SSP host confirms the command and returns the file session identifier.
7) The other Host requests writing data to a node by sending the command FS-OP-FILE-WRITING-Service-Command.
8) The SSP host confirms the command.
9) The other host requests the closing of the file session by sending the command FS-OP-FILE-CLOSE-Service-Command.

10) The SSP host confirms the command.

G.3 File writing from data stream

G.3.1 Data stream gate identifier provided by SSP file application

Figure G.3 illustrates the procedure in which a host requires the file system to write data to an SSP file by using a data stream, the stream gate identifier being provided by the SSP file system application.

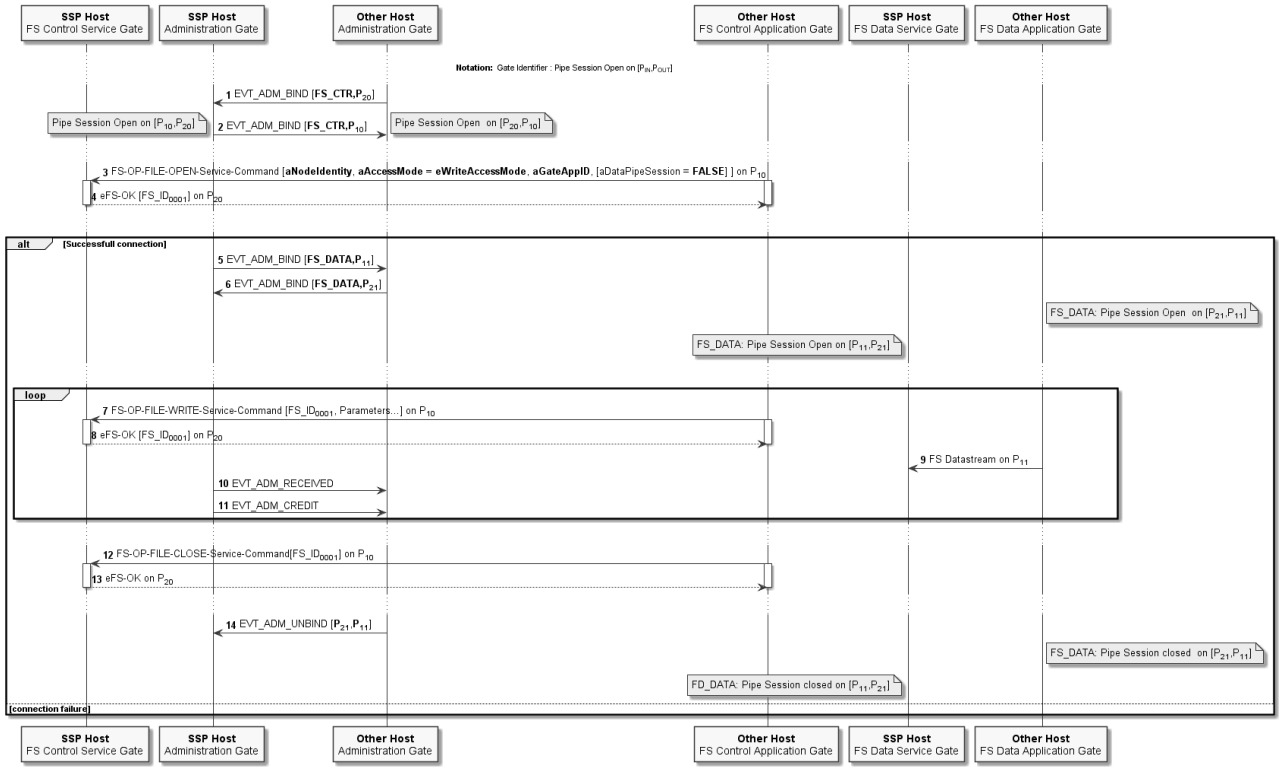

Figure G.3: File writing from data stream

The procedure has 14 steps:

1) The other host requests the opening of a pipe session on the file system control service gate of the SSP host.

2) The SSP host confirms this pipe session opening.

3) The other host requests the opening of a node by sending the command FS-OP-FILE-OPEN-Service-Command with aGateAppID provided and no aDataPipeSession.

4) The SSP host confirms the command and returns the file session identifier.

5) The SSP host requests the opening of a pipe session to the file system data application gate.

6) The other host confirms the pipe session.

7) The other host requests writing data to a node by sending the command FS-OP-FILE-READ-Service-Command.

8) The SSP host confirms the command.

9) The other host sends the data to the SSP host.

10) The SSP host informs the other host about the received data.

11) The SSP host sends to the other host a data credit.

12) The other host requests the closing of the file session by sending the command FS-OP-FILE-CLOSE-Service-Command.

13) The SSP host confirms the command.

14) The other host closes the pipe sessions.

G.3.2 Data stream gate identifier provided by SSP file service

Figure G.3b illustrates the procedure in which a host requires the file system to write data to an SSP file by using a data stream, the stream gate identifier being provided by the SSP file system service.

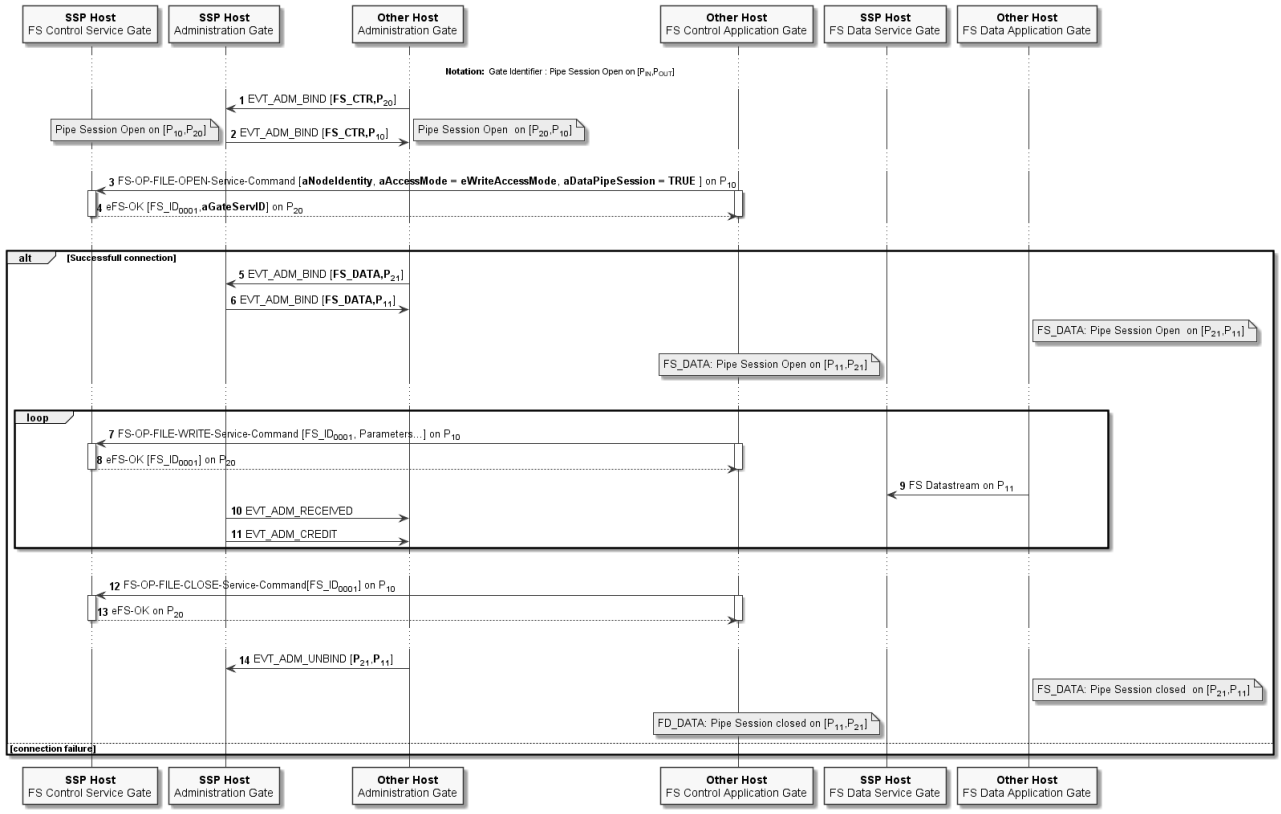

Figure G.3b: File writing from data stream

The procedure has 14 steps:

1) The other host requests the opening of a pipe session on the file system control service gate of the SSP host.

2) The SSP host confirms this pipe session opening.

3) The other host requests the opening of a node by sending the command FS-OP-FILE-OPEN-Service-Command with aDataPipeSession set to TRUE.

4) The SSP host confirms the command and returns the file session identifier with the dynamic gate identifier.

5) The other host requests the opening of a pipe session to the file system data service gate.

6) The SSP host confirms the pipe session.

7) The other host requests writing data to a node by sending the command FS-OP-FILE-READ-Service-Command.

8) The SSP host confirms the command.

9) The other host sends the data to the SSP host.

10) The SSP host informs the other host about the received data.

11) The SSP host sends to the other host a data credit.

12) The other host requests the closing of the file session by sending the command FS-OP-FILE-CLOSE-Service-Command.

13) The SSP host confirms the command.

14) The other host closes the pipe sessions.

G.4 File reading from data stream

G.4.1 Data stream gate identifier provided by SSP file application

Figure G.4 illustrates the procedure in which a host requires the file system to read data from an SSP file by using a data stream, the stream gate identifier being provided by the SSP file system application.

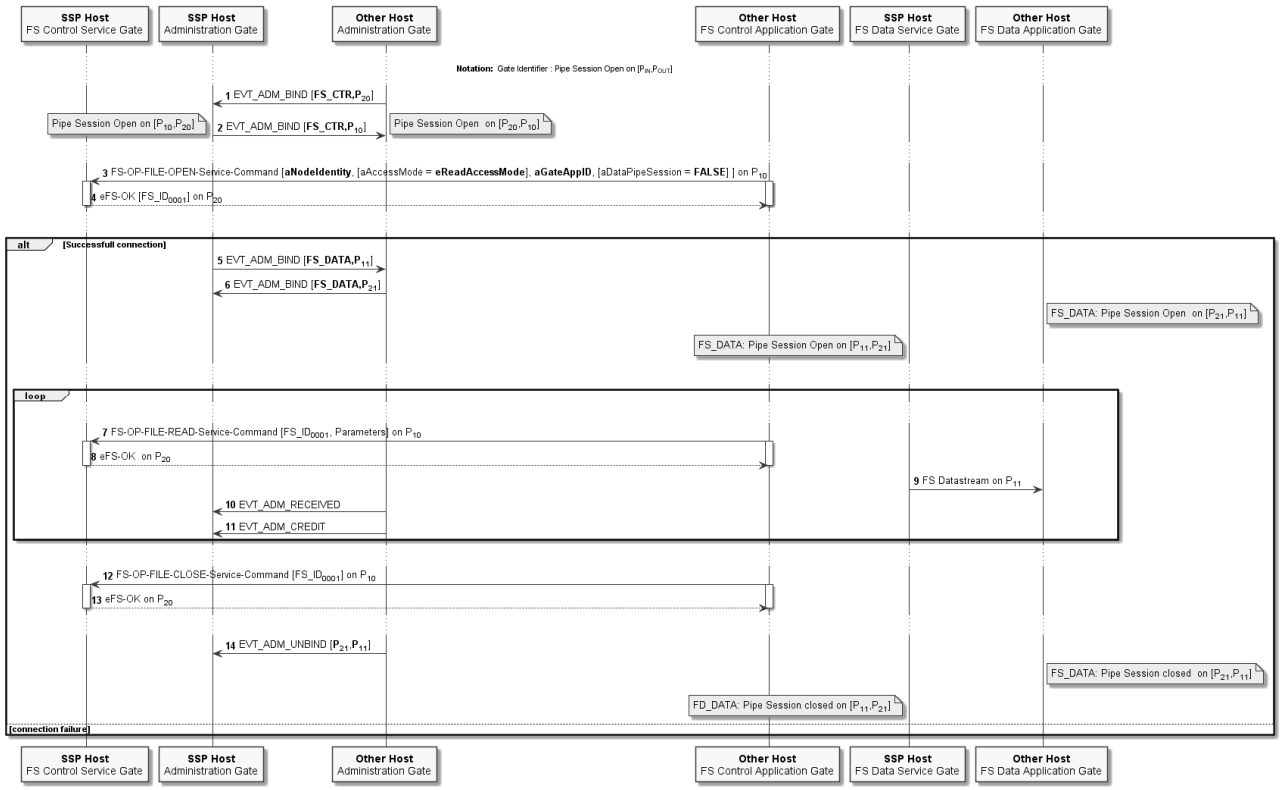

Figure G.4: File reading from data stream

The procedure has 14 steps:

1) The other host requests the opening of a pipe session on the file system control service gate of the SSP host.

2) The SSP host confirms this pipe session opening.

3) The other host requests the opening of a node by sending the command FS-OP-FILE-OPEN-Service-Command command aGateAppID provided and no aDataPipeSession.

4) The SSP host confirms the command and returns the file session identifier.

5) The SSP host requests the opening of a pipe session to the file system data application gate.

6) The other host confirms the pipe session.

7) The other host requests a reading of data from a node by sending the command FS-OP-FILE-READ-Service-Command.

8) The SSP host confirms the command.

9) The SSP host sends the data to the other host.

10) The other host informs the SSP host about the received data.

11) The other host sends to the SSP host a data credit.

12) The other host requests the closing of the file session by sending the command FS-OP-FILE-CLOSE-Service-Command.

13) The SSP host confirms the command.

14) The other host closes the pipe sessions.

G.4.2 Data stream gate identifier provided by SSP file service

Figure G.4b illustrates the procedure in which a host requires the file system to read data from an SSP file by using a data stream, the stream gate identifier being provided by the SSP file system service.

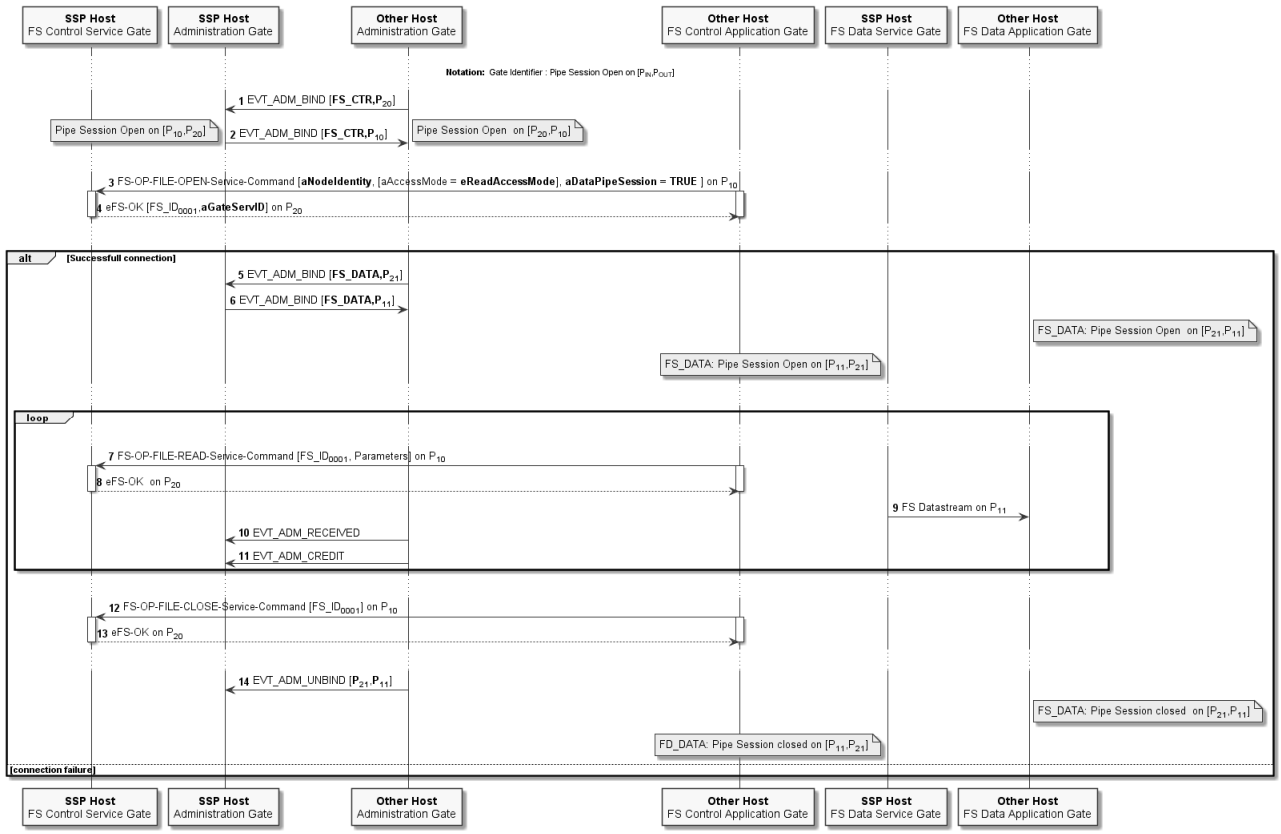

Figure G.4b: File reading from data stream

The procedure has 14 steps:

1) The other host requests the opening of a pipe session on the file system control service gate of the SSP host.

2) The SSP host confirms this pipe session opening.

3) The other host requests the opening of a node by sending the command FS-OP-FILE-OPEN-Service-Command command with aDataPipeSession set to TRUE.

4) The SSP host confirms the command and returns the file session identifier with the dynamic gate identifier.

5) The other host requests the opening of a pipe session to the file system data service gate.

6) The SSP host confirms the pipe session.

7) The other host requests a reading of data from a node by sending the command FS-OP-FILE-READ-Service-Command.

8) The SSP host confirms the command.

9) The SSP host sends the data to the other host.

10) The other host informs the SSP host about the received data.

11) The other host sends to the SSP host a data credit.

12) The other host requests the closing of the file session by sending the command FS-OP-FILE-CLOSE-Service-Command.

13) The SSP host confirms the command.

14) The other host closes the pipe sessions.

G.5 File reading and get the position

Figure G.5 illustrates the procedure in which a host requiring the file system to get the position in an SSP file.

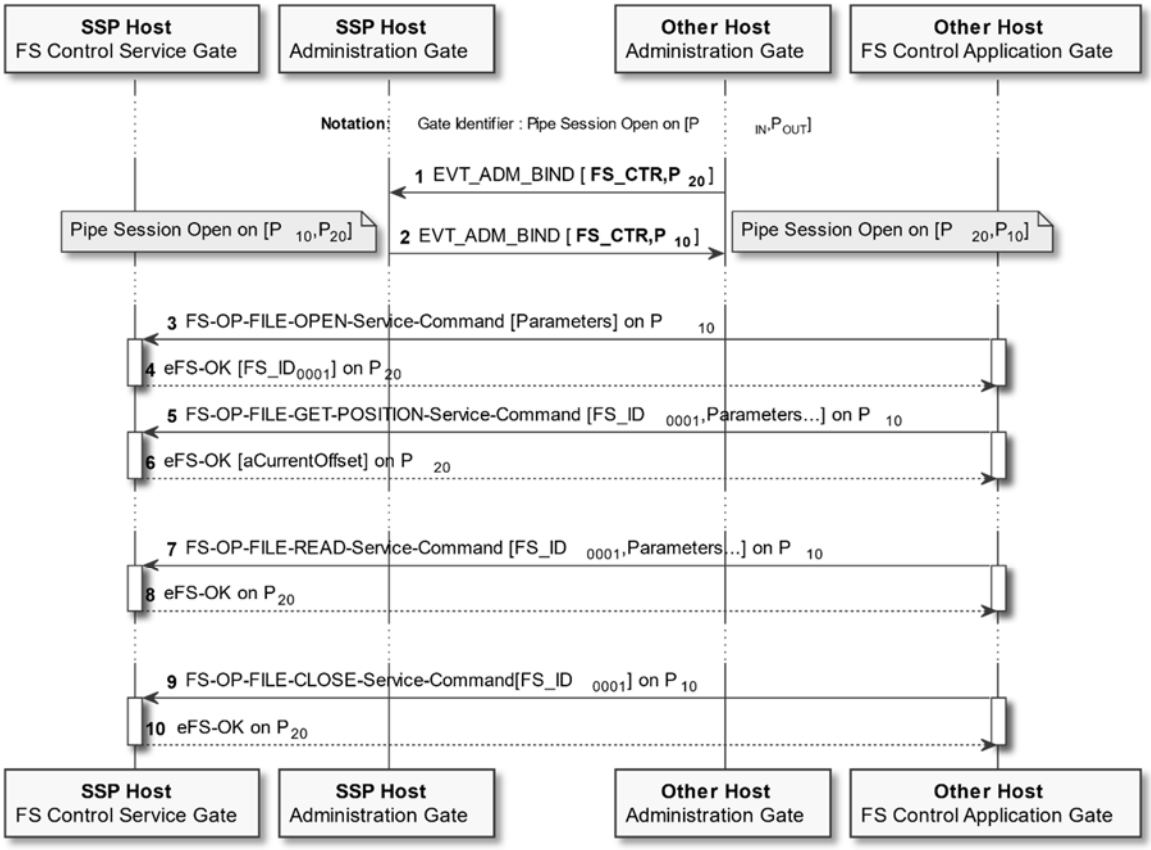

Figure G.5: File get position

The procedure has 10 steps:

1) The other host requests the opening of a pipe session on the file system control service gate of the SSP host.

2) The SSP host confirms this pipe session opening.

3) The other host requests the opening of a node by sending the command FS-OP-FILE-OPEN-Service-Command.

4) The SSP host confirms the command and returns the file session identifier.

5) The other host requests to get a position in the SSPfile by sending the command FS-OP-FILE-GET-POSITION-Service-Command.

6) The SSP host confirms the command.

7) The other host requests a reading of data from a node by sending the command FS-OP-FILE-READ-Service-Command.

8) The SSP host confirms the command and returns the data.

9) The other host requests the closing of the file session by sending the command FS-OP-FILE-CLOSE-Service-Command.

10) The SSP host confirms the command.

Annex H (informative): Example of access control

NOTE: This annex needs to be completed with examples of access control list.

Annex I:
Void

Annex J (informative): Accessor authentication service procedures

J.1 Accessor creation

Figure J.1 illustrates the procedure in which a host requiring an accessor creation.

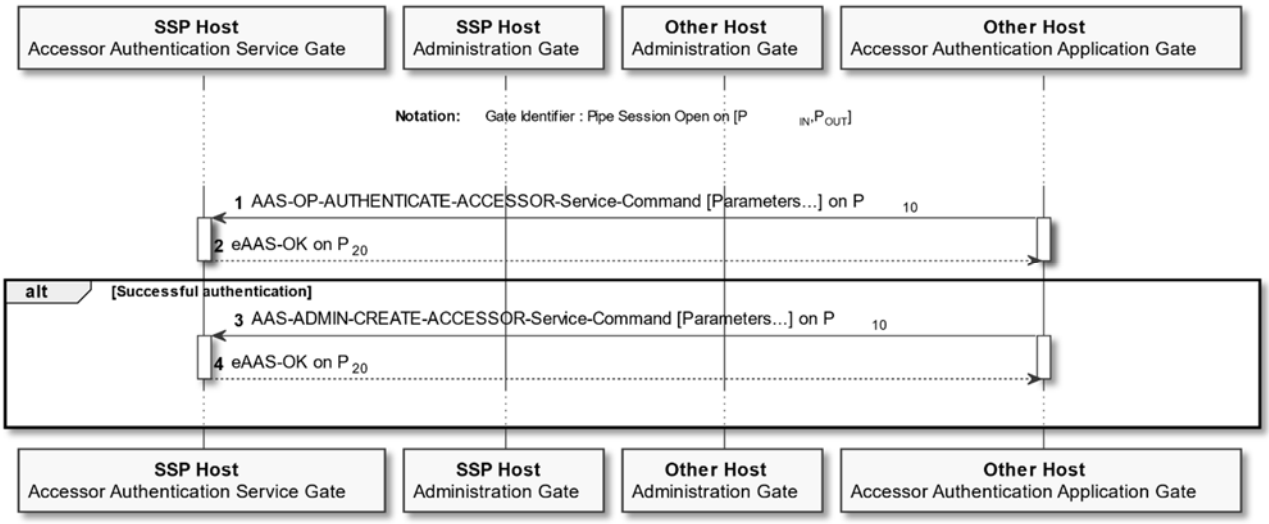

Figure J.1: Accessor creation

The procedure has 4 steps:

1) The other host requests the authentication of an accessor by sending the AAS-OP-AUTHENTICATE-ACCESSOR-Service-Command command.

2) The SSP host confirms the successful authentication of the accessor.

3) The other host requests the creation of an accessor by sending the command AAS-ADMIN-CREATE-ACCESSOR-Service-Command command.

4) The SSP host confirms the successful authentication of the accessor.

J.2 Accessor deletion

Figure J.2 illustrates the procedure in which a host requiring an accessor deletion.

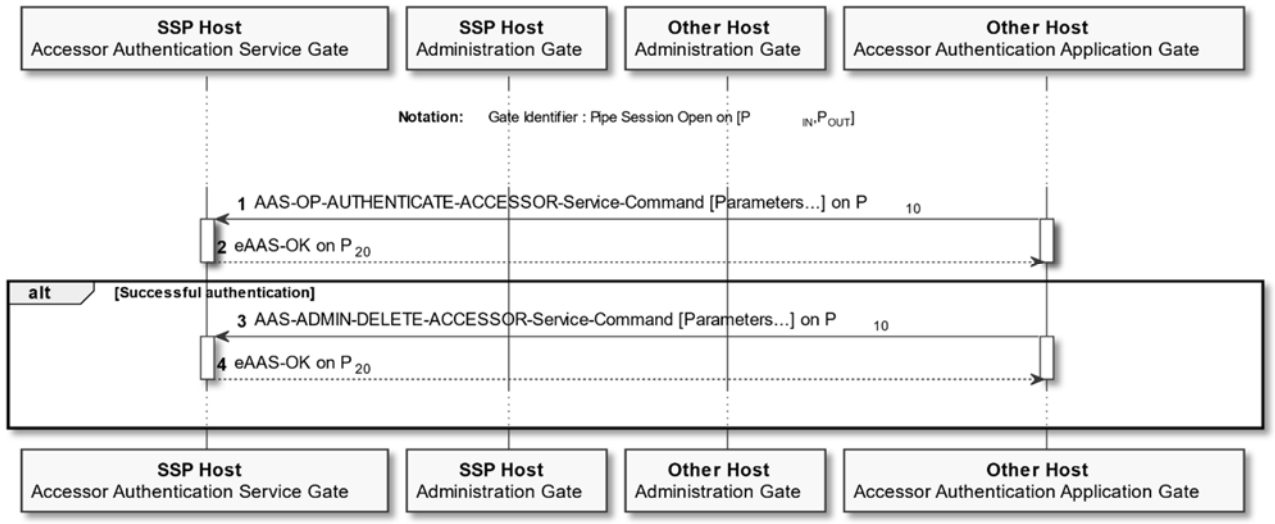

Figure J.2: Accessor deletion

The procedure has 4 steps:

1) The other host requests the authentication of an accessor by sending the AAS-OP-AUTHENTICATE-ACCESSOR-Service-Command command.

2) The SSP host confirms the successful authentication of the accessor.

3) The other host requests the deletion of a deleted accessor by sending the command AAS-ADMIN-DELETE-ACCESSOR-Service-Command command.

4) The SSP host confirms the successful deletion of the accessor.

J.3 Accessor update

Figure J.3 illustrates the procedure in which a Host requiring the update of the credentials of an accessor.

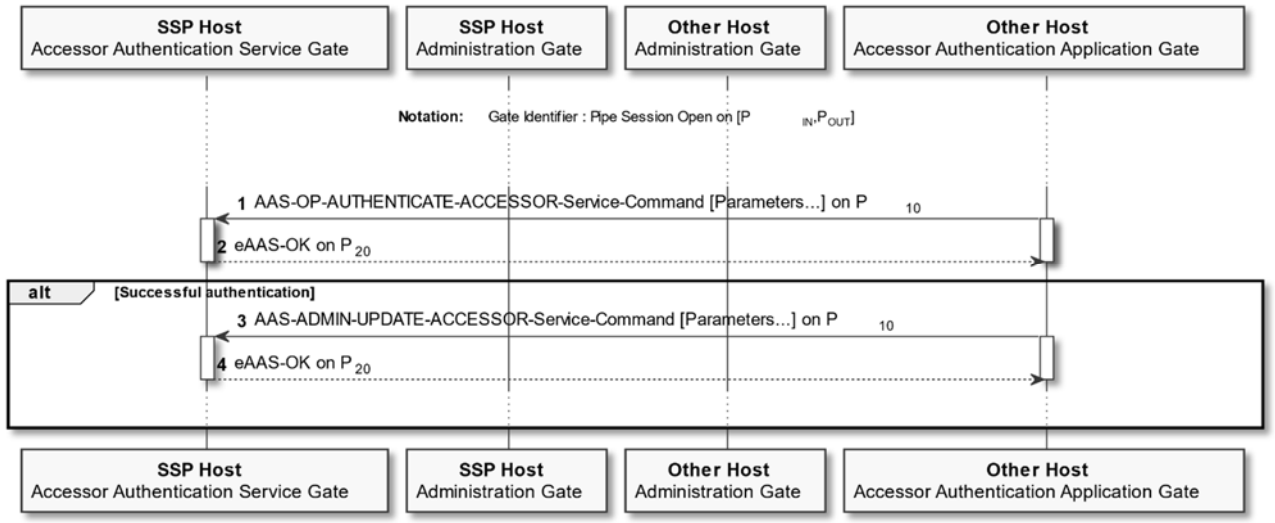

Figure J.3: Accessor credentials update

The procedure has 4 steps:

1) The other host requests the authentication of an accessor by sending the AAS-OP-AUTHENTICATE-ACCESSOR-Service-Command command.

2) The SSP host confirms the successful authentication of the accessor.

3) The other host requests the update of the credential of the accessor by sending the command AAS-ADMIN-UPDATE-ACCESSOR-Service-Command command.

4) The SSP host confirms the successful authentication of the accessor.

J.4 Accessor authentication session opening

Figure J.4 illustrates the procedure in which a host requiring a accessor authentication service to open a pipe session to a generic service gate (e.g. SSP file system control service gate), identified as XXX service gate below.

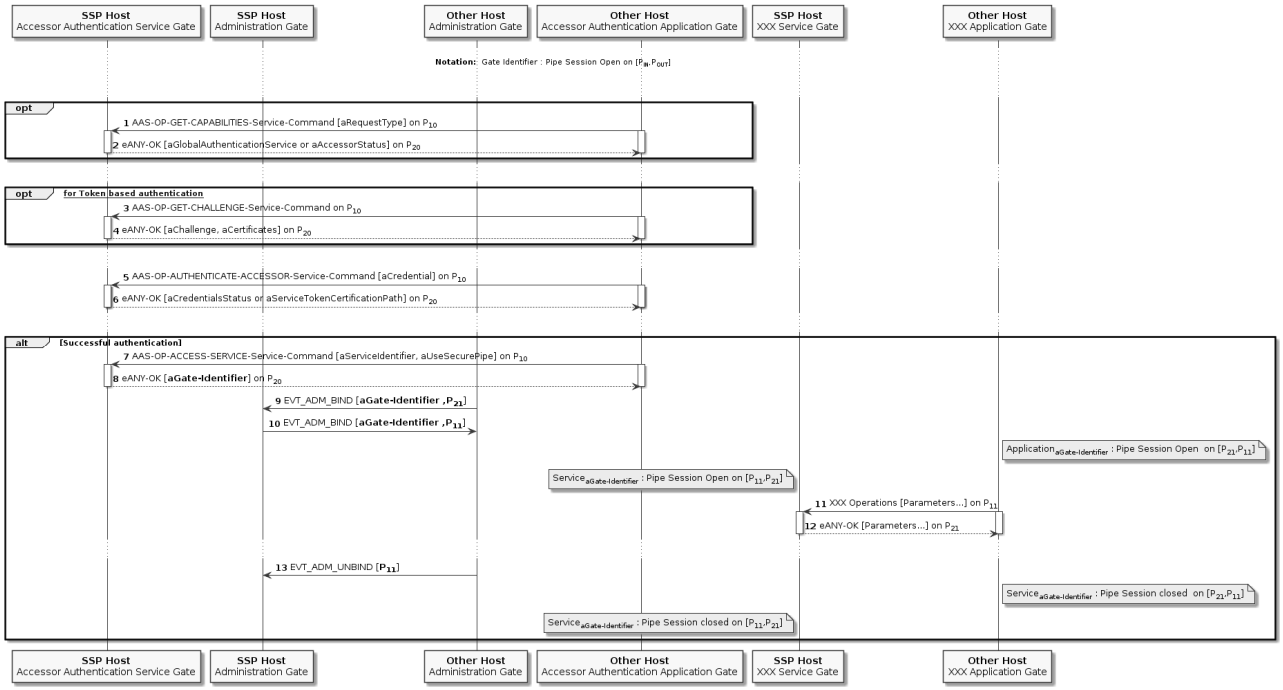

Figure J.4: Accessor authentication session opening

The procedure has 13 steps:

1) The other host requests the authentication service capability from the SSP host by sending the AAS-OP-GET-CAPABILITIES-Service-Command command.

2) The SSP host returns the accessor authentication service capability.

3) The other host requests the authentication of an accessor by sending the command AAS-OP-GET-CHALLENGE-Service-Command command.

4) The SSP host confirms the successful operation.

5) The other host requests the authentication of an accessor by sending the command AAS-OP-AUTHENTICATE-ACCESSOR-Service-Command command with its credentials.

6) The SSP host confirms the successful authentication of the accessor.

7) The other host requests a session to a service by sending the command AAS-OP-ACCESS-SERVICE-Service-Command with the service identified by aServiceIdentifier.

8) The SSP host dynamically creates a gate to the requested service and sends an answer to the other host with the identifier of the dynamically created gate.

9) The other host triggers the opening of a pipe session on the XXX service gate.

10) The SSP host confirms the pipe session.

11) The other host requests a XXX service operation.

12) The SSP host confirms the operation.

13) The other host closes the pipe session to the XXX service gate.

Annex K (informative): UML code of figures

Table K.1 contains the link to the UML code used to generate some of the figures in the present document.

Table K.1: Link to UML code

Figure	Link to UML code
Figure 9.3	https://forge.etsi.org/rep/set/ts_103666-1_general_characteristics/raw/v16.5.0/figures/figure_9.3.plantuml
Figure 9.4	https://forge.etsi.org/rep/set/ts_103666-1_general_characteristics/raw/v16.5.0/figures/figure_9.4.plantuml
Figure D.1	https://forge.etsi.org/rep/set/ts_103666-1_general_characteristics/raw/v16.5.0/figures/figure_D.1.plantuml
Figure D.2	https://forge.etsi.org/rep/set/ts_103666-1_general_characteristics/raw/v16.5.0/figures/figure_D.2.plantuml
Figure D.3	https://forge.etsi.org/rep/set/ts_103666-1_general_characteristics/raw/v16.5.0/figures/figure_D.3.plantuml
Figure D.4	https://forge.etsi.org/rep/set/ts_103666-1_general_characteristics/raw/v16.5.0/figures/figure_D.4.plantuml
Figure E.1	https://forge.etsi.org/rep/set/ts_103666-1_general_characteristics/raw/v16.5.0/figures/figure_E.1.plantuml
Figure E.2	https://forge.etsi.org/rep/set/ts_103666-1_general_characteristics/raw/v16.5.0/figures/figure_E.2.plantuml
Figure F.1	https://forge.etsi.org/rep/set/ts_103666-1_general_characteristics/raw/v16.5.0/figures/figure_F.1.plantuml
Figure F.2	https://forge.etsi.org/rep/set/ts_103666-1_general_characteristics/raw/v16.5.0/figures/figure_F.2.plantuml
Figure F.3	https://forge.etsi.org/rep/set/ts_103666-1_general_characteristics/raw/v16.5.0/figures/figure_F.3.plantuml
Figure G.1	https://forge.etsi.org/rep/set/ts_103666-1_general_characteristics/raw/v16.5.0/figures/figure_G.1.plantuml
Figure G.2	https://forge.etsi.org/rep/set/ts_103666-1_general_characteristics/raw/v16.5.0/figures/figure_G.2.plantuml
Figure G.3	https://forge.etsi.org/rep/set/ts_103666-1_general_characteristics/raw/v16.5.0/figures/figure_G.3.plantuml
Figure G.3b	https://forge.etsi.org/rep/set/ts_103666-1_general_characteristics/raw/v16.5.0/figures/figure_G.3b.plantuml
Figure G.4	https://forge.etsi.org/rep/set/ts_103666-1_general_characteristics/raw/v16.5.0/figures/figure_G.4.plantuml
Figure G.4b	https://forge.etsi.org/rep/set/ts_103666-1_general_characteristics/raw/v16.5.0/figures/figure_G.4b.plantuml
Figure G.5	https://forge.etsi.org/rep/set/ts_103666-1_general_characteristics/raw/v16.5.0/figures/figure_G.5.plantuml
Figure J.1	https://forge.etsi.org/rep/set/ts_103666-1_general_characteristics/raw/v16.5.0/figures/figure_J.1.plantuml
Figure J.2	https://forge.etsi.org/rep/set/ts_103666-1_general_characteristics/raw/v16.5.0/figures/figure_J.2.plantuml
Figure J.3	https://forge.etsi.org/rep/set/ts_103666-1_general_characteristics/raw/v16.5.0/figures/figure_J.3.plantuml
Figure J.4	https://forge.etsi.org/rep/set/ts_103666-1_general_characteristics/raw/v16.5.0/figures/figure_J.4.plantuml

Annex L (normative): ASN.1 definitions

L.1 End of ASN.1

```
-- ASN1START

END

-- ASN1STOP
```

L.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_103666-1_general_characteristics/raw/v17.3.0/asn1/asn1_syntax.asn1.

Annex M (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
03/10/2019	SCP#89	SCP(19)000212	-	-	-	Version 15.0.0 first publication	-	15.0.0
20/12/2019	SCP#90	SCP(19)000240r1	001	1	F	Correction of the identity and loopback flow control and data acknowledgement mechanism	15.0.0	15.1.0
02/04/2020	SCP#91	SCP(20)091015	002	-	D	Supporting multiple APDU gates on SSP host domain	15.1.0	15.2.0
02/04/2020	SCP#91	SCP(20)091016	003	-	F	Annex J Accessor Authentication corrections	15.1.0	15.2.0
02/04/2020	SCP#91	SCP(20)091017	004	-	F	Corrections of the description of the accessor's properties	15.1.0	15.2.0
29/07/2020	SCP#94	SCP(20)000038	005	-	F	Loopback gate flow control data ack	15.2.0	15.3.0
29/07/2020	SCP#94	SCP(20)000039	006	-	F	Terminal Capabilities enhancement	15.2.0	15.3.0
29/07/2020	SCP#94	SCP(20)000040	007	-	D	Minor editorial corrections	15.2.0	15.3.0
29/07/2020	SCP#94	SCP(20)000041	008	-	F	SWP completion	15.2.0	15.3.0
29/07/2020	SCP#94	SCP(20)000042	009	-	B	SSP Capabilities enhancement	15.2.0	15.3.0
29/07/2020	SCP#94	SCP(20)000052	010	-	F	APDU commands additions	15.2.0	15.3.0
29/07/2020	SCP#94	SCP(20)000054	012	-	F	Clarification of the secure SCL message description	15.2.0	15.3.0
29/07/2020	SCP#94	SCP(20)000053	011	-	B	Introduction of eSSP Type 1	15.3.0	16.0.0
24/09/2020	SCP#95	SCP(20)000105	014	-	A	Clarification on clause 7.3.2.3 Answer To Reset content	15.3.0	16.0.0
19/10/2020	-	-	-	-	-	Minor editorial fixes	16.0.0	16.0.1
17/12/2020	SCP#97	SCP(20)000163	016	-	A	Accessor Authentication Service Command correction on Accessor Conditions	16.0.1	16.1.0
17/12/2020	SCP#97	SCP(20)000165	018	-	A	Fix ASN.1 syntax	16.0.1	16.1.0
11/03/2021	SCP#98	SCP(21)000013	020	-	A	Correction of clause 7.3.2.2	16.1.0	16.2.0
11/03/2021	SCP#98	SCP(21)000015	022	-	A	Improvements of the description of the Accessor authentication service	16.1.0	16.2.0
10/06/2021	SCP#100	SCP(21)000077	024	-	A	Addition of a algorithm identifier in the Authentication Token	16.2.0	16.3.0
10/06/2021	SCP#100	SCP(21)000080	026	-	A	File system data GateId type modification	16.2.0	16.3.0
10/06/2021	SCP#100	SCP(21)000082	028	-	A	Missing parameter in the Accessor Authentication Service	16.2.0	16.3.0
10/06/2021	SCP#100	SCP(21)000084	030	-	A	Addition of an error code for the EVT-UDP-ERROR-Application-Event event	16.2.0	16.3.0
10/06/2021	SCP#100	SCP(21)000086	032	-	A	Clarification for accessors of type group.	16.2.0	16.3.0
10/06/2021	SCP#100	SCP(21)000088	034	-	A	Align TCP error information with UDP error information	16.2.0	16.3.0
10/06/2021	SCP#100	SCP(21)000090	036	-	A	CRON-KILL-ALL-TIMERS-Command cannot fail	16.2.0	16.3.0
10/06/2021	SCP#100	SCP(21)000092	038	-	A	Addition of a credential at the accessor creation	16.2.0	16.3.0
30/09/2021	SCP#101	SCP(21)000140	040	-	A	Clarification on the service identifier for the AAS	16.3.0	16.4.0
25/03/2022	SET#104	SET(22)000044	42	-	A	Clarification on SSP Bundle Interoperability	16.4.0	16.5.0
25/03/2022	SET#104	SET(22)000045	43	-	B	FS data service UUID uniqueness	16.4.0	16.5.0
25/03/2022	SET#104	SET(22)000047	45	-	A	Clarification about ACL requesting a secure pipe session	16.4.0	16.5.0
25/03/2022	SET#104	SET(22)000049r1	47	1	A	Addition of a sequence field in secure SCL	16.4.0	16.5.0
25/03/2022	SET#104	SET(22)000050	48	-	B	Alignment with VNP 2.0	16.4.0	16.5.0
08/07/2022	SET#106	SET(22)000118r1	49	1	B	Introduction of Class D	16.5.0	17.0.0

Change history								
Date	**Meeting**	**Plenary Doc**	**CR**	**Rev**	**Cat**	**Subject/Comment**	**Old**	**New**
08/07/2022	SET#106	SET(22)000119	50	-	B	Addition of reference to class eSSP Type 2	16.5.0	17.0.0
22/09/2022	SET#107	SET(22)000202	53	-	A	Clarification of the endpoints of a dynamic pipe	17.0.0	17.1.0
08/12/2022	SET#108	SET(22)000223r1	56	1	A	Remove usage of ANY DEFINED BY from ASN.1 in SSPCapability for SspClass	17.1.0	17.2.0
09/03/2022	SET#109	SET(23)000033	59	-	A	Remove usage of ANY DEFINED BY from ASN.1 in MetaDatum	17.2.0	17.3.0

History

Document history		
V17.0.0	August 2022	Publication
V17.1.0	November 2022	Publication
V17.2.0	February 2023	Publication
V17.3.0	June 2023	Publication