GLOBALPLATFORM®
THE STANDARD FOR SECURE DIGITAL SERVICES AND DEVICES

GlobalPlatform Technology

Secure Element Management Service

Card Specification v2.3 – Amendment I

Version 1.0.0.6 (target 1.1)

Public Review

February 2021

Document Reference: GPC_SPE_121

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

Contents

Figures

Tables

1 Introduction

This document defines an extension of the GlobalPlatform Card Specification [GPCS] aiming at changing the traditional Card Content Management model by:

- Extending the point-to-point application deployment to a broadcasted deployment and management of Java Card-based services hosted on a secure IC device through standardized GlobalPlatform card administration commands.
- Extending and refining the existing GlobalPlatform delegation model through the use of certificates.

This specification uses SE-based devices as an example, but this management service is not limited to this type of smart card and can be applicable to any capable Java Card secure IC device.

1.1 Audience

This amendment is intended primarily for card manufacturers and application developers implementing GlobalPlatform card solutions.

It is assumed that the reader is familiar with smart cards and smart card production and, in particular, with the GlobalPlatform Card Specification [GPCS] and the Java Card specifications [JC API], [JC VM], and [JC RE].

1.2 IPR Disclaimer

Attention is drawn to the possibility that some of the elements of this GlobalPlatform specification or other work product may be the subject of Intellectual Property Rights (IPR) held by GlobalPlatform members or others. For additional information regarding any such IPR that have been brought to the attention of GlobalPlatform, please visit https://www.globalplatform.org/specificationsipdisclaimers.asp. GlobalPlatform shall not be held responsible for identifying any or all such IPR, and takes no position concerning the possible existence or the evidence, validity, or scope of any such IPR.

1.3 References

Table 1-1: Normative References

Standard / Specification	Description	Ref
GlobalPlatform Card Specification	GlobalPlatform Card Specification v2.3.1	[GPCS]
GPCS Amendment A	GlobalPlatform Card Technology – Confidential Card Content Management – Card Specification v2.3 – Amendment A v1.2	[Amd A]
GPCS Amendment D	GlobalPlatform Card Technology – Secure Channel Protocol '03', Card Specification v2.2 – Amendment D v1.1.1	[Amd D]
GPCS Amendment H	GlobalPlatform Card Technology – Executable Load File Upgrade, Card Specification v2.3 – Amendment H v1.1	[Amd H]
GlobalPlatform API	GlobalPlatform Card API (org.globalplatform) v1.7 or higher	[GPC API]
GlobalPlatform Common Implementation Configuration	GlobalPlatform Card – Common Implementation Configuration v2.0	[CIC]
ANSI X9.62-2005	Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA)	[ANSI X9.62]
BSI TR-03111, Version 2.0	BSI Technical Guideline TR-031111: Elliptic Curve Cryptography, 2012-06-28	[TR-03111]
ITU	Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)	[ITU-T Rec X690]
NIST, SP800-38A	Recommendation for Block Cipher Modes of Operation: Methods and Techniques – NIST Special Publication 800-38A	[SP-800-38A]
NIST, SP800-56A	Special Publication 800-56A, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography	[SP-800-56A]
RFC 5639	Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation	[RFC 5639]

Table 1-2: Informative References

Standard / Specification	Description	Ref
Java Card API	Application Programming Interface, Java Card™ Platform, v3.0.1 Classic Edition, Sun Microsystems, Inc., May 2009	[JC API]
Java Card VM	Virtual Machine Specification, Java Card™ Platform, v3.0.1 Classic Edition, Sun Microsystems, Inc., May 2009	[JC VM]
Java Card JCRE	Runtime Environment Specification, Java Card™ Platform, v3.0.1 Classic Edition, Sun Microsystems, Inc., May 2009	[JC RE]

1.4 Terminology and Definitions

Terms used in this document are generally defined in [GPCS].

Additional items are listed in Table 1-3.

Table 1-3: Terminology and Definition

Term	Description
CA-KLCC	Certification Authority for Confidential Key Loading Card Certificates.
CA-SEMS	The SEMS Certification Authority owns the CA-SEMS.ENC and CA-SEMS.AUT key pairs. It is responsible for the management of these key pairs, which are used to generate and sign the CERT.SP.AUT.
CA-SEMS Privilege	A Privilege coded in the CERT.SP.AUT to define the ability for a CERT.SP.AUT certificate holder to rotate the SEMS Application keys.
CERT.CA-KLCC.{AUT, ECKA, ECDSA}	Certificate set holding a public key PK.CA-KLCC.{AUT, ECKA, ECDSA} (per scheme supported) used to verify the signature of one certificate of CERT.CASD.{AUT, ECKA, ECDSA}.
CERT.CASD.{AUT, ECKA, ECDSA}	Certificate holding a public key PK.CASD.{AUT, ECKA, ECDSA} (per confidential key setting scheme supported), defined in section 4.5, used to verify the signature of the data returned by the SEMS Application. It is diversified per SE and stored within the SEMS Application and/or CASD. It can be retrieved by the SEMS_GET_DATA command.
CERT.SP.AUT	Certificate, defined in section 4.5.1, provided to an SP by the CA-SEMS. It embeds the PK.SP.AUT used to verify the signature of the SEMS script. It contains the CCM rights assigned to the SP.
Group of SEs	Group of Secure Elements sharing the same keys PK.CA-SEMS.AUT and SK.CA-SEMS.ENC.
PK.CA-KLCC.{AUT, ECKA, ECDSA}	Public key (embedded in CERT.CA-KLCC.{AUT, ECKA, ECDSA}) used to verify the signature of a CERT.CASD.{AUT, ECKA, ECDSA} certificate returned by the SEMS Application. It is diversified per SE.

Term	Description
PK.CASD.{AUT, ECKA, ECDSA}	CASD public key set (related to SK.CASD.{AUT, ECKA, ECDSA}) embedded in CERT.CASD.{AUT, ECKA, ECDSA}. Depending on the scheme applied, this (respective) public key is used to verify the authenticity and the integrity of the on-board generated key or for an EC Key Agreement by the OCE. It is diversified per SE.
PK.CA-SEMS.AUT	CA-SEMS public key (related to SK.CA-SEMS.AUT) used to verify the certificates signed by the CA-SEMS; i.e. CERT.SP.AUT. PK.CA-SEMS.AUT is stored in the SEMS Application. All SEs of a given group may share the same PK.CA-SEMS.AUT.
PK.CA-SEMS.ENC	CA-SEMS public key used by the Service Provider to initiate the encryption process of a SEMS script. It is used in computing the main encryption key K, encrypting the SEMS script as described in section 4.6. All SEs of a given group may share the same PK.CA-SEMS.ENC.
PK.SP.AUT	Service Provider public key (embedded in CERT.SP.AUT) used by the SEMS Application to verify the authenticity of a SEMS script. CERT.SP.AUT is contained in SEMS scripts.
PK.SP.ECKA	Service Provider public key (related to SK.SP.ECKA.S3) used in the generation of a shared secret based on an ECKA algorithm. It is embedded in the command (defined in section 7.12) that is sent to the SEMS Application to trigger the on-board key generation.
PK.SP.ENC.{S1,S4}	Service Provider public key (related to SK.SP.ENC.{S1,S4}) used to encrypt on-board generated keys returned by the SEMS Application. It is embedded in commands, defined in section 7.11 and 7.12, sent to the SEMS Application to trigger the on-board key generation.
Rich OS	Mobile Equipment (ME) Operating System on which the SP Application and the SEMS Device Agent run.
Secure Element Issuer (SEI)	Entity that usually holds a particular Security Domain in the SE: the Issuer Security Domain (ISD). Card Content Management (CCM) operations that can be performed on this ISD are associated with the Security Domain Manager role. The SEI allows the installation of the SEMS Application and chooses CA-KLCC.
SEMS Application	SEMS Security Domain or Java Card applet instance processing the SEMS script and making use of the virtual I/O interface to transfer GlobalPlatform CCM APDUs to the targeted Application.
SEMS Application APDU	Command and Response APDUs, defined in section 6, exchanged between the SEMS Device Agent and the SEMS Application.
SEMS Application Provider	Entity responsible for the SEMS Application Java Card Applet.
SEMS Device Agent	Rich OS SEMS Application running on the ME.
SEMS Key Rotation	Operation to change the keys of the SEMS Application within a (group of) SEs.

Term	Description
SK.CA-KLCC.{AUT, ECKA, ECDSA}	Private key set (related to PK.CASD.{AUT, ECKA, ECDSA} and securely stored in the CA-KLCC) used to sign the respective certificate of CERT.CASD.{AUT, ECKA, ECDSA}. It is related to public key set PK.CA-KLCC.{AUT, ECKA, ECDSA} respectively.
SK.CASD.{AUT, ECKA, ECDSA}	Private key set (related to PK.CASD.{AUT, ECKA, ECDSA} and securely stored in the SE) used by the SEMS Application to guarantee the authenticity and integrity of the on-board generated key returned by the SEMS Application. It is diversified per SE.
SK.CA-SEMS.AUT	CA-SEMS private key (related to PK.CA-SEMS.AUT) used to sign a CERT.SP.AUT certificate(s). It is valid for a given group of SEs and kept in the CA-SEMS secure environment.
SK.CA-SEMS.ENC	CA-SEMS private key (related to PK.CA-SEMS.ENC and stored in the SE) used by the SEMS Application to decrypt an incoming SEMS script. The same SK.CA-SEMS.ENC is used for a given group of SEs.
SK.SP.ECKA.S3	Service Provider private key used by the Service Provider to generate a Shared Secret based on the ECKA Algorithm.
SK.SP.ENC.{S1, S4}	Service Provider private key used by the Service Provider to decrypt the data returned by the SEMS Application.
SP Application	Rich OS SP Application running on the ME. It can be seen as part of the Service Provider wallet. It is referenced by the Caller Package Name.

1.5 Abbreviations and Notations

Table 1-4: Abbreviations and Notations

Abbreviation / Notation	Meaning
. || .	A || B means A concatenated with B
AES	Advanced Encryption Standard
AID	Application Identifier
AMSD	Authorized Management Security Domain
APDU	Application Protocol Data Unit
API	Application Programming Interface
ASP	Application Specific Parameters
AUT	Authentication
C_MAC	Command MAC
CA	Certification Authority
CAPDU	Command APDU
CASD	Controlling Authority Security Domain
CCM	Card Content Management

Abbreviation / Notation	Meaning
C-DATA	Command Data Field
CERT	Certificate
CSR	Certificate Signing Request
DEK	Data Encryption Key
DES	Data Encryption Standard
DOid	Data Object ID
DR	Derivation Random
E[K,D]	Encryption of D with key K
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
ECKA	Elliptic Curve Key Agreement
ELF	Executable Load File
EM	Executable Module
ENC	Encryption
FCI	File Control Information
GP	GlobalPlatform
H_n	SHA-256 of the encrypted SEMS command_n concatenated with H_{n+1}
ISD	Issuer Security Domain
ISO/IEC	International Organization for Standardization / International Electrotechnical Commission
KA	Key Agreement
KDF	Key Derivation Function
KVN	Key Version Number
LV	Length Value
MAC	Message Authentication Code
ME	Mobile Equipment (e.g. Mobile Phone, Wearable Device)
OBGK	On-Board Generated Keys
OCE	Off-Card Entity
PK	Public part of a Private-Public key pair
R_MAC	Response MAC
RAPDU	Response APDU
RGK	Randomly Generated Key
RID	Registered application Identifier
SCP	Secure Channel Protocol

Abbreviation / Notation	Meaning
SD	Security Domain
SE	Secure Element
SEI	Secure Element Issuer
SEMS	Secure Element Management Service
SHA-256	Secure Hash Algorithm with 256-bit digest
ShS	Shared Secret
SK	Private part of a Private-Public key pair
SN	Serial Number
SP	Service Provider
SP TSM	Service Provider TSM
SPSD	Service Provider Security Domain
SSD	Supplementary Security Domain
SW	Status Word
TLV	Tag Length Value
TSM	Trusted Service Manager

1.6 Revision History

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

Table 1-5: Revision History

Date	Version	Description
March 2018	1.0	Public Release
January 2020	1.0.0.1	• Updated the Amendment A version. • Defined new tags 'C5', 'C6', '4D' in CERT.SP.AUT. • Replaced the key derivation defined in section 4.8.2.1 with a reference to the key derivation defined in [Amd A] section 3.5.2. • Updated SEMS_APDU, SEMS_INSTALL_FOR_LOAD, and SEMS_DELETE to support new tags 'C5', 'C6', '4D'. • For SEMS_INSTALL_FOR_INSTALL command: o Corrected typo in Install Options field. o It is now possible to assign the Global Delete privilege (if allowed by CERT.SP.AUT). • Corrected P2 parameter in SEMS_DELETE. • Corrected description in SEMS_PUT_KEY command for PUT_KEY modes and C-DATA field. • Added new commands to manage an ELF update session for a single ELF with different old and new ELF AID and to manage a multi-ELF upgrade session: SEMS_BEGIN_MANAGE_ELF_UPGRADE and SEMS_END_MANAGE_ELF_UPGRADE.
September 2020	1.0.0.2	Tech writer update
October 2020	1.0.0.4	Member Review
February 2021	1.0.0.6	Clarified SEMS_BEGIN_MANAGE_ELF_UPGRADE Status Words Public Review
TBD	v1.1	Public Release

2 Use Case and Requirements

Most of today's connected wearable and digital devices allow easy third-party software installation. Service and Content Providers as well as end users benefit from easy-to-use deployment and upgrade processes. Device Application deployment and upgrade processes are user friendly, well defined, and established through software application "stores" under the control of the end user.

However, managing content and services within an SE (i.e. in the form of Java Card applets) is more complex: Today's conventional secure service deployment model consists of having TSMs perform Card Content Management on behalf of the SE provider and the cardholder. On-card post-issuance application management is jointly managed by the SEI TSM and the different SP TSMs; it requires knowledge of key data which is diversified for each SE and therefore necessitates bespoke command and response data for each of them.

The main objective of the SE Management Service (SEMS) defined in this specification is to reduce Card Content Management costs through simplification of the administration mechanism and migration from an SEI TSM-based model to a Service Provider TSM-centric model.

Switching from a one-to-one and 24h-a-day synchronous relationship between a Service Provider and an SE towards a one-to-many and asynchronous relationship enables a reduction in the involvement of the SEI TSM and therefore minimizes costs. Most Card Content Management steps and rights associated with a given Service Provider are encapsulated into a single generic secured script applying to a given group of SEs and transferred via any modern application store available for Mobile Equipment (MEs).

New Card Content Management rights are defined as part of the delegation mechanism whereby it is possible for a certification authority to limit the actions that may be performed on a group of SEs for each given Service Provider.

The main simplification of the role of the Service Provider comes from the fact that SE diversified specific key data is no longer required to launch an administration session. This is a key feature for new entrants on the market.

The goal of the SEMS is to reduce the need of the SEI TSM to a minimum so that it is needed only at manufacturing time during card initialization. The SEMS CA plays a similar role.

This specification uses SE-based devices as an example, but this management service is not limited to this type of smart card and can be applicable to any capable Java Card secure IC device.

Requirement Summary

The SEMS shall enable broadcasted management and personalization of SDs, ELFs, and Application instances of a group of SEs via secured but generic (i.e. not specific to a particular SE) scripts. Traditional GlobalPlatform Card Content Management operations are extracted and executed on a given group of SEs without requiring the intervention of an SEI TSM. Being secured, those scripts may be transmitted via non-secure channels (typically via applications running on any Rich OS Device Host embedding an SE). The security protection is checked and lifted and the associated Card Content Management Rights are enforced on the target SEs. The SEMS shall define an extension of the current GlobalPlatform Card Content Management delegation mode and allow certificates to be used to assign rights to the Service Provider for pre-defined groups of SEs. A management right shall consist of allowing a given GlobalPlatform Card Content Management operation for a certain number of iterations. A special AID applicable for Application or Security Domain supporting the SEMS mechanism, as defined in this specification, shall be defined. It shall be possible to generate a single SEMS script that can be executed on SEs personalized with different CA-SEMS. Requiring a separate SEMS script for each CA-SEMS would greatly increase the complexity of SEMS script deployment, which might lead to a non-optimal solution in practice.

3 Secure Element Management Service

For any given group of SEs which are managed as described in section 3.1, the Service Provider generates key pairs and certificates allowing only pre-defined GlobalPlatform Card Content Management (CCM) operations to be performed up to a certain number of times on those particular SEs through SEMS.

The following GlobalPlatform CCM operations may be delegated:

- Creation of Security Domains (with or without the Authorized Management privilege)
- Secure Channel key injection (on-board or off-board Key Generation) in SDs
- Loading and deletion of ELFs
- Instantiation and deletion of applets
- Applet personalization with non-diversified data
- Key rotation of the SEMS on-card entity in case of change of ownership or for security reasons

On receipt of a SEMS CCM script, each built-in certificate is checked by the SEMS Application residing on each SE.

3.1 Management per Group of SEs

Two different types of grouping can be formed by two different roles: the CA-SEMS (see section 3.2.1) and the SP (see section 3.2.3). Depending on the use case, one can decide to group per CA-SEMS credentials.

For CA credentials, groups of SEs may be created at the discretion of the CA-SEMS. Such a group shares the same PK.CA-SEMS.AUT and SK.CA-SEMS.ENC keys (described in section 4.5.1).

It is strongly recommended that each CERT.SP.AUT certificate has a different PK.SP.AUT key so that the revocation of a certificate leads to the revocation of the PK.SP.AUT key. If the same PK.SP.AUT key were shared among multiple CERT.SP.AUT, revocation of the certificate would not lead to the revocation of the key but to the revocation of multiple certificates, which may not be the desired result. If a Service Provider wants to manage multiple services independently, it is preferable that the Service Provider requests multiple CERT.SP.AUT with a different Service Provider identifier (i.e. certificate holder identifier) per service.

3.2 Main Roles in SEMS Ecosystem

The main roles participating in the SEMS mechanism are:

- The SEMS Certification Authority (CA-SEMS)
- The Key Loading Card Certificates Certification Authority (CA-KLCC)
- The Service Provider (SP)
- The SEMS Application Provider

3.2.1 SEMS Certification Authority

The SEMS Certificate Authority (CA-SEMS) has the following responsibilities:

- Manage the CA-SEMS.AUT and CA-SEMS.ENC key pairs described in section 4.5.1.
- Release CERT.SP.AUT certificate(s), defined in section 4.5.1, to a Service Provider on receipt of a Certificate Signing Request issued by a Service Provider.

3.2.2 Key Loading Card Certificates Certification Authority

The Key Loading Card Certificates Certificate Authority (CA-KLCC) has the following responsibilities:

- Manage the CASD.AUT, CASD.ECKA, and CASD.ECDSA key pairs described in section 4.5.1.
- Release CA-KLCC certificate(s), defined in section 4.5.1, to a Service Provider on its request, so that the Service Provider can verify the (data origin) authenticity of the SEMS Application responses.

3.2.3 Service Provider

The Service Provider deploys and operates services on groups of SEs. The Card Content Management scope is defined through CERT.SP.AUT certificates that are generated and provided by the CA-SEMS. The Service Provider (also referred to as the SEMS SP certificate holder) generates, secures, and broadcasts generic SEMS scripts to MEs, thus allowing the execution of standard GlobalPlatform CCM operations on groups of SEs.

A SEMS script is a collection of CERT.SP.AUT certificate(s), frame(s) containing the script signature and data used to decrypt the SEMS commands, and encrypted and integrity protected SEMS commands.

3.2.3.1 SP Off-Card Backend Entity

The Service Provider’s off-card backend entity generates SP.AUT key pairs and issues Certificate Signing Requests to the CA-SEMS in order to be able to generate SEMS scripts.

3.2.3.2 SP Off-Card Device Entity (SP Application)

The Service Provider’s off-card device entity (“App”) runs on the ME Rich OS layer. It uses the functionalities exposed by the SEMS Device Agent to forward the SEMS script contents to the SEMS on-card entity and to collect the returned data.

3.2.3.3 SP On-Card Entity

The Service Provider’s on-card entity(ies) are GlobalPlatform elements created, installed, deleted, and personalized via the SEMS mechanism.

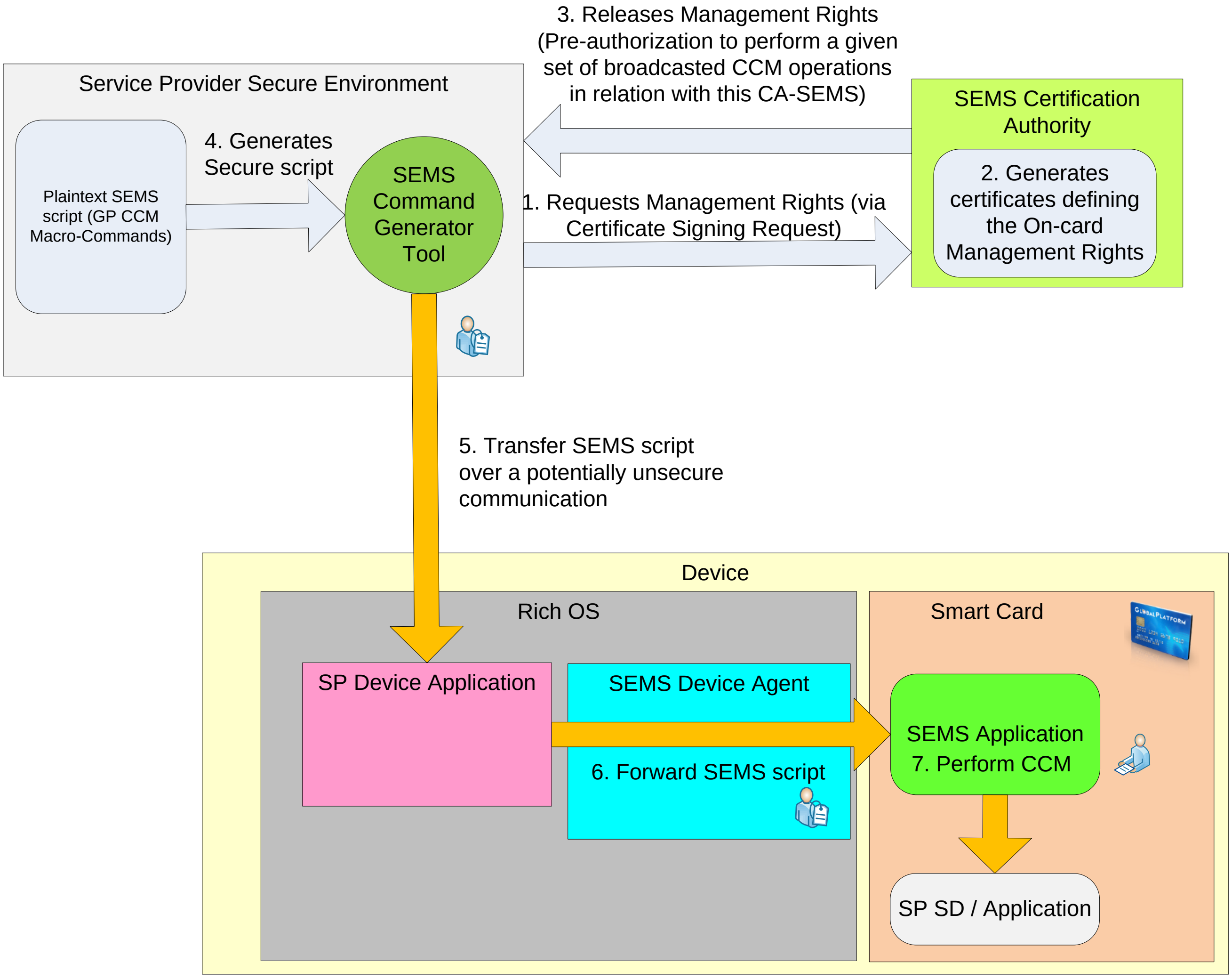
3.2.4 SEMS Application Provider

The SEMS Application Provider, functioning as a special Service Provider (separately described in section 3.2.3), is responsible for installing and provisioning the SEMS Application. It has a strong trust relationship to the CA-SEMS (described in section 3.2.1) and may be authorized to rotate the CA-SEMS keys or, if the SEMS Application is implemented as a Java Card applet, to update the SEMS Application using the SEMS Updater described in section 4.15.

3.3 Runtime Processing

Figure 3-1 illustrates components and steps needed to perform broadcasted CCM operations via the SEMS.

Figure 3-1: Broadcasting of CCM Operations via SEMS


3. Releases Management Rights (Pre-authorization to perform a given set of broadcasted CCM operations in relation with this CA-SEMS)
Service Provider Secure Environment
Plaintext SEMS script (GP CCM Macro-Commands)
4. Generates Secure script
SEMS Command Generator Tool
1. Requests Management Rights (via Certificate Signing Request)
SEMS Certification Authority
2. Generates certificates defining the On-card Management Rights
5. Transfer SEMS script over a potentially unsecure communication
Device
Rich OS
SP Device Application
SEMS Device Agent
6. Forward SEMS script
Smart Card
SEMS Application
7. Perform CCM
SP SD / Application

Step 0:

On the different MEs of a pre-defined group, the SEMS Device Agent is installed in the Rich OS layer. The SEMS Application is installed and personalized. The installation and personalization of the SEMS Application is out of scope of this specification.

Step 1:

Via a Certificate Signing Request (CSR), the Service Provider requests the assignment of SEMS CCM rights in order to be allowed to broadcast and execute CCM commands through the SEMS Device Agent and Application. The CCM rights (type of GlobalPlatform CCM operation + maximum number of CCM operations) are defined within CERT.SP.AUTs that are contained in each SEMS script processed by the SEMS Application. The CSR shall contain the PK.SP.AUT.

Steps 2 and 3:

The CA-SEMS generates the following certificates and delivers them to the Service Provider:

- CERT.SP.AUT certificate that contains the key (PK.SP.AUT) used to verify the SEMS script authenticity and the CCM rights delegated to the Service Provider. The CERT.SP.AUT is embedded in the secured SEMS script generated by the Service Provider.
- CERT.CA-SEMS.ENC certificate providing the key (PK.CA-SEMS.ENC) used to encrypt SEMS scripts, as described in section 4.6.1.

Step 4:

The Service Provider generates a secured SEMS script (hashed, signed, and encrypted). The SEMS script contains both the CERT.SP.AUT and the SEMS command(s). The SEMS commands may create an SD, load an ELF, create an applet or SD instance, delete an instance of an ELF, etc.

Step 5:

The Service Provider provides the secured SEMS script to its Device Application by any means (i.e. via a secure or insecure OTA channel or simply by embedding it into its own Device Application freely available for download from any given software application store).

Step 6:

The SP Application triggers the forwarding of the (now locally available) SEMS script by the SEMS Device Agent towards the ME's SE for execution.

Step 7:

The SEMS Application processes SEMS script security, checks CCM rights, and processes SEMS commands.

SEMS scripts are generic to a certain extent in the sense that any given SEMS script may be executed only on the SE members of a pre-defined group of SEs where the corresponding SEMS software components have been installed and configured with the same SK.CA-SEMS.ENC and PK.CA-SEMS.AUT keys (see section 3.1).

Each SEMS script may be encrypted by the SP on behalf of the CA-SEMS (which itself has a trust relationship with the SEMS Application which takes care of the SEMS command on-card decryption before script execution). The corresponding ciphertext is signed by the SP using SK.SP.AUT, and the CERT.SP.AUT is appended to the SEMS script.

4 System Architecture

The SEMS software components installed on mobile or wearable devices are:

- The SP Device Application
- The SEMS Device Agent
- The SEMS Application

4.1 SP Device Application Definition

The SP Device Application resides on the Rich OS layer of the ME. It uses the functionalities exposed by the SEMS Device Agent. The SP Device Application receives SEMS scripts and calls the API of the SEMS Device Agent to forward them to the SEMS Application.

The specification of the SP Device Application is out of scope of this specification, as is the communication channel between the SP Device Application and the Service Provider Secure Environment.

There is no security requirement regarding the storage of a SEMS secured script received from the Service Provider Secure Environment or generated by the SEMS.

4.2 SEMS Device Agent Definition

The SEMS Device Agent resides on the Rich OS layer of the ME. It exposes an interface to the SP Device Application(s) to retrieve the filename and location of the SEMS script to be executed.

The SEMS Device Agent is responsible for reading SEMS commands from a locally-available SEMS script and forwarding them to the SEMS Application for processing and execution by the targeted Application. The SEMS Device Agent stores the response messages from the SEMS Application in a local file that can then be fetched by the SP Device Application.

4.3 SEMS Application Definition

The SEMS Application is an on-card Application that can process SEMS commands. It may be implemented as either a Java Card applet or a Security Domain. The SEMS Application, regardless of its implementation as an SD or a Java Card applet, has the unique ability to forward APDUs via the virtual I/O interface defined in section 4.7.1. The OPEN shall grant access to the virtual I/O interface only to the SEMS Application and SEMS Updater, which are identified as described in section A.1.

The SEMS Application implements integrity verification, authentication checking, and the decryption mechanism of the SEMS script as described in section 4.6.2; it enforces the CCM rights defined within the Service Provider certificate as described in section 4.5.1; and it processes the SEMS commands. The SEMS commands and the responses to the execution of the SEMS commands are transferred to the SEMS Application by, respectively, the message and the response message of the PROCESS SCRIPT COMMAND APDU, defined in section 6.2.

The SEMS Application is responsible for:

- Checking the authentication and integrity of the SEMS script by verifying the CERT.SP.AUT and the signature of the SEMS script as described in section 4.6.2.
- Integrity verification and decryption of the SEMS commands embedded in the SEMS script as described in section 4.6.2.
- Enforcing the SEMS CCM rights defined in the CERT.SP.AUT contained in the SEMS script as described in section 4.5.2.
- Processing the SEMS commands and forwarding the generated APDU to the Application selected with the SEMS_SELECT (defined in section 7.3), through the Virtual I/O defined in section 4.7.1.
- Retrieving the APDU response returned by the selected Application through the Virtual I/O, defined in section 4.7.1, and building the SEMS command response returned to the SEMS Agent in the PROCESS SCRIPT COMMAND APDU response, defined in section 6.2.

4.3.1 SEMS Command Overview

Table 4-1 lists the GlobalPlatform CCM-related SEMS commands:

Table 4-1: SEMS Commands

Name	Description
SEMS_SELECT	Selects an Application referenced by its AID.
SEMS_APDU	Sends any APDU to a selected Application.
SEMS_BEGIN_PERSO	Marks the start of a sequence of personalization commands.
SEMS_END_PERSO	Marks the end of a sequence of personalization commands.
SEMS_INSTALL_FOR_LOAD	Initiates the loading or update of an ELF.
SEMS_LOAD	Loads an ELF.
SEMS_INSTALL_FOR_INSTALL	Installs a new Application instance.
SEMS_DELETE	Deletes an ELF or an Application.
SEMS_PULL_KEY	Triggers generation of the on-board SCP key set (as described in section 4.8.2.1) and loading the key set into the SD selected by the SEMS Application.
SEMS_PULL_KEY_GP	Triggers generation of the on-board SCP key set, either based on a variant of GlobalPlatform scenario #1 using PK scheme (described in section 4.8.2.2) or based on GlobalPlatform scenario #3 (according to section 4.8.2.3).
SEMS_PUT_KEY	Loads an SCP key set into an SD created with the SEMS script and whose state is SELECTABLE.
SEMS_GET_DATA	Retrieves a data object from the SEMS Application (i.e. CERT.CASD.AUT).
SEMS_BINDING_SE	Binds the execution of a SEMS command to a specific SE referenced by its Serial Number.
SEMS_KEY_ROTATION	Triggers the rotation of the SEMS ECDSA keys (i.e. the keys used to verify a SEMS signature and to decrypt SEMS commands).
SEMS_BEGIN_MANAGE_ELF_UPGRADE	Marks the start of an ELF upgrade session.
SEMS_END_MANAGE_ELF_UPGRADE	Marks the end of an ELF upgrade session.

4.4 Card Content Management Rights

The CA-SEMS assigns CCM rights to Service Providers who wish to use the SEMS mechanism. The CCM rights are stored within certificate(s); i.e. CERT.SP.AUT, as specified in section 4.5, contained in SEMS scripts as defined in section 5. The goal is to define the CCM scope (i.e. the GlobalPlatform CCM operation(s)) that the SEMS SP certificate holder (i.e. the Service Provider) may execute on a given group of SEs via the SEMS mechanism. SEMS CCM rights typically define rights and quotas for a given group of SEs; i.e. individual rights to execute GlobalPlatform CCM commands together with the maximum number of iterations for each type of CCM command. The SEMS CCM rights may be assigned to the following GlobalPlatform CCM operations performed through SEMS:

- The creation of SDs with Authorized Management (AMSD). The right specifies the maximum number of AMSDs that this SEMS SP certificate holder is allowed to create on an SE.
- The creation of SDs without Authorized Management. The right specifies the maximum number of SDs that this SEMS SP certificate holder is allowed to create. This right is not applicable if the same certificate allows the instantiation of an AMSD.
- The loading of ELFs. The right specifies the maximum number of ELFs that this SEMS SP certificate holder is allowed to load. This right is not applicable if the same certificate allows the instantiation of an AMSD.
- The installation of Applications. The CCM right specifies the maximum number of Applications that this SEMS SP certificate holder is allowed to install on an SE. This right is not applicable if the same certificate allows the instantiation of an AMSD.

As part of the SEMS security principles, it shall be possible to install an Application either:

- From an ELF that was previously loaded with a SEMS script by the same Service Provider as the one performing the install operation, irrespective of the License byte value set during the ELF loading; or
- From an ELF that was previously loaded with a SEMS script by another Service Provider if the License byte set during the ELF loading was '00'; or
- From any ELF (previously loaded through a SEMS script or not, e.g. preloaded) if the CERT.SP.AUT of the current SEMS script contains tag '4B' with that ELF AID.

The usage of an ELF loaded with a SEMS script may be subject to an ELF License. An ELF License is the concatenation of the maximum number of Application instances that may be created from the ELF referenced by the given AID and with the AID itself. An ELF License gives the right to create a maximum number of Applications from the ELF per SE. A CERT.SP.AUT may contain multiple ELF Licenses. The owner of an ELF does not need an ELF License to create an Application from the loaded ELF. The owner of an ELF is identified by tag '5F20' of CERT.SP.AUT that was used during loading. ELFs preloaded by the card vendor (i.e. not loaded with a SEMS script) may also be associated with a specific owner (i.e. value of tag '5F20') in a proprietary way, which is out of scope.

The SEMS shall only consider the CCM rights and licenses within the CERT.SP.AUT that is used to authenticate the currently executed SEMS script. The SEMS shall not track the rights and licenses stored in CERT.SP.AUT embedded in previous SEMS scripts that have already been executed.

4.5 SEMS Key and Certificate Definitions

4.5.1 Keys and Certificates Versus Roles

A SEMS script is authenticated and its integrity and confidentiality are protected with the cryptographic mechanism described in section 4.6. In this version of the specification, the keys mentioned below (asymmetric as well as symmetric) are described in section 4.6.3 and are used to ensure the authentication, integrity, and confidentiality of SEMS scripts.

SEMS cryptography relies on five asymmetric key pairs:

(SK.CA-SEMS.ENC, PK.CA-SEMS.ENC)

- These keys are used to decrypt / encrypt SEMS scripts. The key pair is owned by the SEMS Certification Authority (CA-SEMS). The immediate purpose of this key pair is to protect the key used for encryption and decryption of the SEMS script as described in section 4.6.
  - The private key SK.CA-SEMS.ENC is stored within the SEMS Application. It is used to retrieve the key in order to decrypt SEMS scripts (as described in section 4.6.2). After the personalization of the SEMS Application, the SK.CA-SEMS.ENC key may be rotated by the SEMS Application Provider on behalf of the CA-SEMS with the SEMS_KEY_ROTATION command, described in section 7.18. The same SK.CA-SEMS.ENC shall be loaded in all the SEMS Applications of a given group of SEs.
  - The public key PK.CA-SEMS.ENC is provided to the Service Provider by the CA-SEMS. The public encryption key is used to protect the randomly chosen key that is used to encrypt a SEMS script.
  - The detailed usage of this key pair is described in Figure 4-2 in section 4.6.

(SK.CA-SEMS.AUT, PK.CA-SEMS.AUT)

- These keys are used to generate / verify a certificate signature. The key pair is owned by CA-SEMS.
  - The private key SK.CA-SEMS.AUT is kept in the CA-SEMS environment. It is used to sign the CERT.SP.AUT certificates that are provided to SPs that are given the right to perform certain CCM operations with the SEMS scripts.
  - The public key PK.CA-SEMS.AUT is stored within the SEMS Application. It is used to verify the signature of the CERT.SP.AUT certificate contained in the SEMS script. After the personalization of the SEMS Application, the PK.CA-SEMS.AUT key may be rotated with the SEMS_KEY_ROTATION command, described in section 7.18. The same PK.CA-SEMS.AUT shall be set for all the SEs of a given group.

(SK.SP.AUT, PK.SP.AUT)

- These keys are used to generate / verify the SEMS script signature. The key pair is owned by the Service Provider. A Service Provider may own several CERT.SP.AUT certificates.
  - The private key SK.SP.AUT is stored in the secure environment of the Service Provider. It is used to sign a SEMS script.
  - The public key PK.SP.AUT is stored within the CERT.SP.AUT certificate which is signed with the CA-SEMS private key SK.CA-SEMS.AUT. This key is used to verify the signature of a SEMS script.

(SK.CASD.{AUT, ECKA, ECDSA}, PK.CASD.{AUT, ECKA, ECDSA})

- These keys are used to sign / verify the authenticity and the integrity of the on-board generated key or for an EC Key Agreement (ECKA). The key pair for CASD.ECDSA might (in this version of this specification) be stored in the SEMS Application whereas the other two key pairs (regarding CASD.AUT and CASD.ECKA) are supposed to be stored in the CASD with KVN '74' (according to [Amd A]). The location of the corresponding private keys and their associated certificates (containing the respective public key) are implementation specific. These key pairs are diversified per SE. The provisioning of SEMS Applications is out of scope of this specification.
    - The private keys SK.CASD.{AUT, ECKA, ECDSA} are stored in the SE and used to sign the on-board generated keys (OBGK) returned by the SEMS Application or used in ECKA. The private keys SK.CASD.{AUT, ECKA, ECDSA} cannot be rotated.
    - The public keys PK.CASD.{AUT, ECKA, ECDSA} are stored in the CERT.CASD.{AUT, ECKA, ECDSA} certificates which are signed by the respective SK.CA-KLCC.{AUT, ECKA, ECDSA} (root) private key of the CA-KLCC. The CERT.CASD.{AUT, ECKA, ECDSA} certificates are stored in the SE and may be retrieved with the SEMS_GET_DATA command, described in section 7.14.
    - The (SK.CASD.ECDSA, PK.CASD.ECDSA) key pair is mandatory while (SK.CASD.AUT, PK.CASD.AUT) and (SK.CASD.ECKA, PK.CASD.ECKA) are optional.
    - CERT.CASD.ECDSA, SK.CASD.ECDSA are used in the key setting described in section 4.8.2.1. CERT.CASD.AUT and SK.CASD.AUT are used in the key setting described in section 4.5.4. CERT.CASD.ECKA and SK.CASD.ECKA are used in the [Amd A] scenario #3 implemented with the SEMS_PULL_KEY_GP command, which is described in section 7.12.

(SK.SP.ENC.{S1,S4}, PK.SP.ENC.{S1,S4}) or (SK.SP.ECKA.S3, PK.SP.ECKA.S3)

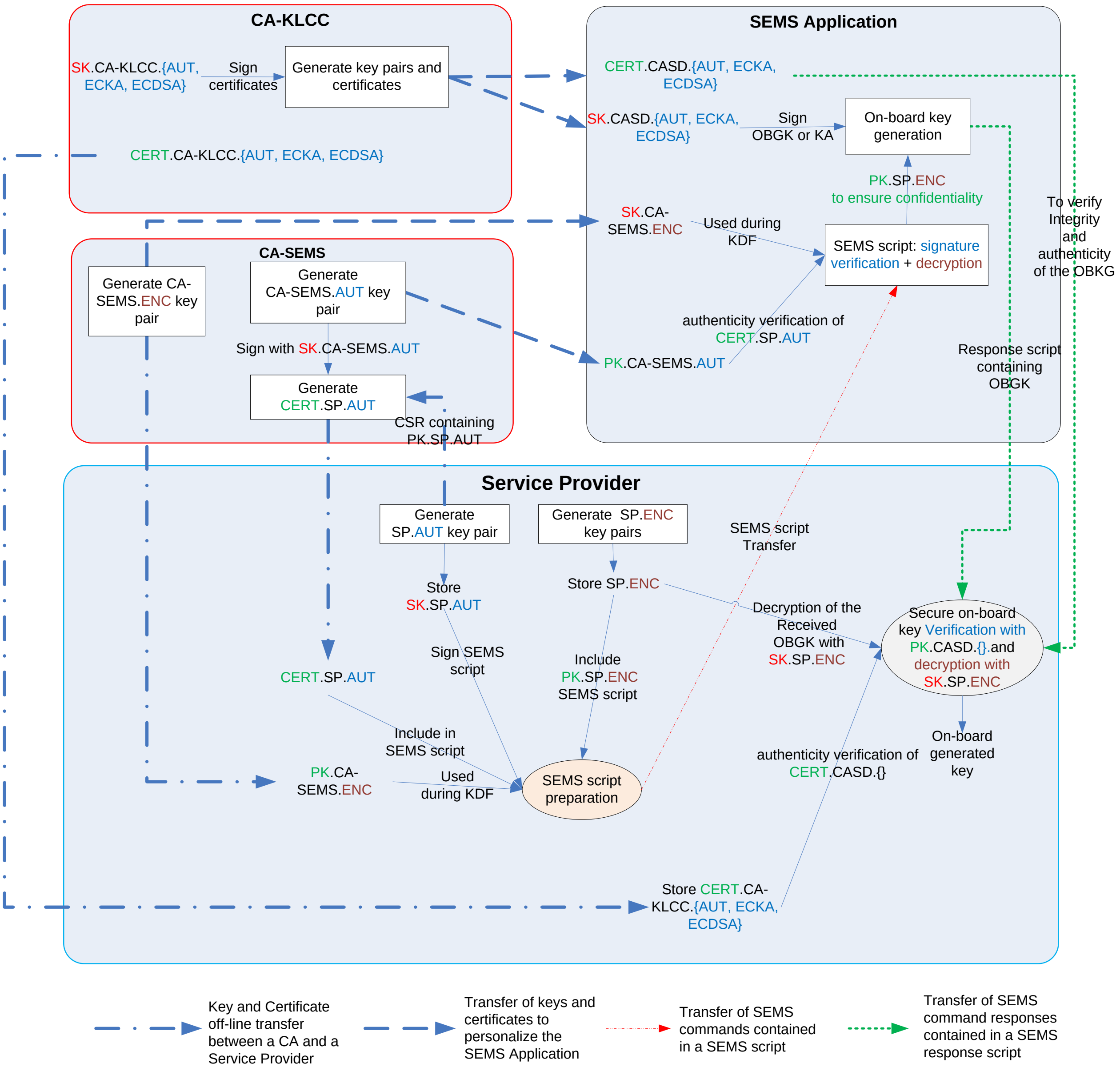
- Depending on the scenario, these keys are used to encrypt / decrypt the SD key generated by the SEMS Application or for a key agreement (see section 4.8.2). The SP generates this key pair once.
    - The private key SK.SP.ENC.{S1,S4} or SP.PK.ECKA.S3 is stored in the Service Provider secure environment.
    - The public key PK.SP.ENC.{S1,S4} or PK.SP.ECKA.S3 is inserted within SEMS scripts in the SEMS command triggering the on-board key generation. The SEMS Application applies the encryption of the on-board generated key with PK.SP.ENC on behalf of the SP or uses an ECKA algorithm.

The process of provisioning the SEMS Application and/or the SE with CERT.CASD.{AUT, ECKA, ECDSA}, SK.CASD.{AUT, ECKA, ECDSA}, SK.CA-SEMS.ENC and PK.CA-SEMS.AUT is out of scope of this specification.

The transfer of credentials from the CA-SEMS to the SE and the storage of CERT.CASD.{AUT, ECKA, ECDSA}, SK.CASD.{AUT, ECKA, ECDSA}, SK.CA-SEMS.ENC and PK.CA-SEMS.AUT shall be ensured to be integrity protected and received from a genuine OCE.

The transfer from the CA to the SE and the SEMS Application shall ensure the confidentiality of SK.CA-SEMS.ENC.

Figure 4-1: Overview of Keys and Certificates


CA-KLCC
SK.CA-KLCC.{AUT, ECKA, ECDSA}
Sign certificates
Generate key pairs and certificates
CERT.CA-KLCC.{AUT, ECKA, ECDSA}
SEMS Application
CERT.CASD.{AUT, ECKA, ECDSA}
SK.CASD.{AUT, ECKA, ECDSA}
Sign OBGK or KA
On-board key generation
PK.SP.ENC
to ensure confidentiality
To verify Integrity and authenticity of the OBKG
SK.CA-SEMS.ENC
Used during KDF
SEMS script: signature verification + decryption
authenticity verification of CERT.SP.AUT
PK.CA-SEMS.AUT
Response script containing OBGK
CA-SEMS
Generate CA-SEMS.ENC key pair
Generate CA-SEMS.AUT key pair
Sign with SK.CA-SEMS.AUT
Generate CERT.SP.AUT
CSR containing PK.SP.AUT
Service Provider
Generate SP.AUT key pair
Generate SP.ENC key pairs
SEMS script Transfer
Store SK.SP.AUT
Store SP.ENC
Decryption of the Received OBGK with SK.SP.ENC
Secure on-board key Verification with PK.CASD.{}.and decryption with SK.SP.ENC
Sign SEMS script
Include PK.SP.ENC SEMS script
CERT.SP.AUT
Include in SEMS script
On-board generated key
authenticity verification of CERT.CASD.{}
PK.CA-SEMS.ENC
Used during KDF
SEMS script preparation
Store CERT.CA-KLCC.{AUT, ECKA, ECDSA}
Key and Certificate off-line transfer between a CA and a Service Provider
Transfer of keys and certificates to personalize the SEMS Application
Transfer of SEMS commands contained in a SEMS script
Transfer of SEMS command responses contained in a SEMS response script

4.5.2 CERT.SP.AUT – Service Provider Certificate

In the current version of this specification, a maximum length of 512 bytes for the SP certificate shall be supported. The SEMS Application may reject the transmission of longer certificates.

Table 4-2: CERT.SP.AUT

Tag	Length (bytes)	Description					Presence
'7F21'	Var.	Service Provider certificate					
		Tag	**Length**	**Description**			
		'93'	16	Certificate Serial Number			Mandatory
		'42'	16	CA-SEMS Identifier			Mandatory
		'5F20'	16	Subject identifier (a.k.a. certificate holder)			Mandatory
		'95'	1	Key usage. Shall be '82' meaning digital signature verification.			Mandatory
		'5F25'	4	Effective Date (YYYYMMDD, BCD format)			Optional
		'5F24'	4	Expiration Date (YYYYMMDD, BCD format)			Optional
		'45'	8	CA-SEMS signature verification key identifier			Mandatory
		'53'	Var.	Card Content Management permission			Conditional
				Tag	**Length**	**Description**	
				'48'	8	CCM rights (see Table 4-4)	Mandatory
				'C9'	Var.	Defines the Application Specific Parameters (see Table 4-6) that may be present in a SEMS_INSTALL_FOR_INSTALL (see [GPCS] section 3.3.1.3)	Optional
				'C5'	1,3	Allowed privileges for Security Domain installation. Tag 'C5' is coded as in section 11.1.2 of [GPCS]. If tag 'C5' is not present, then 'C5' is equal to 'FF EF F0'.	Optional
				'C6'	1,3	Allowed privileges for Applet installation. Tag 'C6' is coded as in section 11.1.2 of [GPCS]. If tag 'C6' is not present, then 'C6' is equal to '1E 87 30'.	Optional
				'4B'	6..17	ELF license: concatenation of two values: • The maximum number of Applications that may be created from the ELF (coded on one byte) • The ELF AID	Optional
				'4C'	1	SEMS Privileges (See Table 4-5)	Optional

Tag	Length (bytes)	Description					Presence
				'4D'	6…17	Tag '4D' is the concatenation of a policy byte, coded on one byte, followed by an AID. The AID shall be the AID of a preloaded ELF and Applications available on the SE before the Card life cycle state transitions to SECURED. The policy byte describes the operations allowed on the AID. It shall be coded as defined in Table 4-8. Multiple tags '4D' may be added to CERT.AP.AUT	0-N
		'5F37'	64	Signature: Concatenation of r and s component (r || s)			Mandatory
		'7F49'	Var.	Public key PK.SP.AUT			
				Tag	**Length**	**Description**	
				'86'	65	Public key PK.SP.AUT coded as '04' || X || Y	Mandatory
				'8F'	1-2	Key type according to [GPCS] Table B-2	Optional

The signature is computed using ECDSA, specified in [GPCS] section B.4.3, on all inner TLVs ('93'... '7F49') excluding tag '5F37' as defined in Table 4-3. The CERT.SP.AUT certificate shall be signed with the private key SK.CA-SEMS.AUT of a CA-SEMS referenced by inner tag '42'.

The public ECC key stored in tag '86' uses uncompressed encoding as described in the BSI Technical Guideline TR-03111 [TR-03111].

The signature stored in tag '5F37' is coded in plain format as specified in [TR-03111]; i.e. it is the concatenation of the byte string representation of r and s.

Table 4-3: Data Signed to Generate the Signature of the CERT.SP.AUT Certificate

Tag	Length (bytes)	Description					Presence
'7F21'	Var.	Service Provider certificate					
		Tag	**Length**	**Description**			
		'93'	16	Certificate Serial Number			Mandatory
		'42'	16	CA-SEMS Identifier			Mandatory
		'5F20'	16	Subject identifier (a.k.a. certificate holder)			Mandatory
		'95'	1	Key usage. It must be '82', meaning digital signature verification.			Mandatory
		'5F25'	4	Effective Date (YYYYMMDD, BCD format)			Optional
		'5F24'	4	Expiration Date (YYYYMMDD, BCD format)			Optional
		'45'	8	CA-SEMS signature verification key identifier			Mandatory
		'53'	Var.	Card Content Management permission			Conditional
				Tag	**Length**	**Description**	
				'48'	8	CCM rights (see Table 4-4)	Mandatory
				'C9'	Var.	Defines the Application Specific Parameters (see Table 4-6) that may be present in a SEMS_INSTALL_FOR_INSTALL (see [GPCS] section 3.3.1.3)	Optional
				'C5'	1,3	Allowed privileges for Security Domain installation. Tag 'C5' is coded as in [GPCS] section 11.1.2. If tag 'C5' is not present, then 'C5' is equal to 'FF EF F0'.	Optional
				'C6'	1,3	Allowed privileges for Applet installation. Tag 'C6' is coded as in [GPCS] section 11.1.2. If tag 'C6' is not present, then 'C6' is equal to '1E 87 30'.	Optional
				'4B'	6..17	ELF license	Optional
				'4C'	1	SEMS Privileges (see Table 4-5)	Optional

Tag	Length (bytes)	Description					Presence
				'4D'	6…17	Tag '4D' is the concatenation of a policy byte, coded on one byte, followed by an AID. The AID shall be the AID of a preloaded ELFs and Applications available on the SE before the Card life cycle state transitions to SECURED. The policy byte describes the operations allowed on the AID. It shall be coded as defined in Table 4-8. Multiple tags '4D' may be added to CERT.AP.AUT.	0-N
		'7F49'	Var.	Public key PK.SP.AUT			Mandatory
				Tag	**Length**	**Description**	
				'86'	65	Public key PK.SP.AUT encoded as '04' || X || Y	Mandatory
				'8F'	1-2	Key type according to [GPCS] Table B-2	Optional

The tag order shall be as described in Table 4-2 except for the sub-tags of tag '53'.

Tag '53' shall appear at least once in the certificate chain. The Card Content Management permissions assigned in tag '53' are evaluated at the execution of SEMS_INSTALL_FOR_LOAD (section 7.7), or during SEMS_INSTALL_FOR_INSTALL (section 7.9), or during SEMS_KEY_ROTATION (section 7.18), or during the SEMS Application update process (section 4.15). A CERT.SP.AUT certificate shall not be rejected because Card Content Management permissions are not consistent with the Card Content Management permissions of any CERT.SP.AUT certificate of the chain. The most constrained permission over the CERT.SP.AUT certificate of the chain is considered by SEMS commands (i.e. SEMS_INSTALL_FOR_LOAD, SEMS_INSTALL_FOR_INSTALL, or SEMS_KEY_ROTATION) or by the SEMS package update.

Table 4-4: Card Content Management Rights

Bytes 1-2	Bytes 3-4	Bytes 5-6	Bytes 7-8
Number of SDs with AM that may be created	Number of SDs without AM that may be created	Number of ELFs that may be loaded	Number of applet instances (i.e. excluding SD) that may be created

Tag '4B' may be repeated several times. The other tags shall be present once at most.

Table 4-5: Privileges – Tag '4C'

b8	b7	b6	b5	b4	b3	b2	b1	Description
-	-	-	-	-	-	1	-	CA-SEMS Privilege
-	-	-	-	-	-	0	-	No CA-SEMS Privilege
					1		-	SEMS Application Provider Privilege
					0		-	No SEMS Application Provider Privilege
x	x	x	x	x	-	-	x	RFU

The CA-SEMS privilege allows delegation of the SEMS Application key rotation. A script to rotate the SEMS Application keys shall contain a CERT.SP.AUT certificate with bit b2 of tag '4C' set.

The SEMS Application Provider Privilege allows delegation of the SEMS Application ELF update rights. A script containing an update of the SEMS Application Java Card applet shall contain a CERT.SP.AUT certificate with bit b3 of tag '4C' set. The SEMS Application Provider should be the only entity that may be assigned the CA-SEMS Privilege (sub-tag '4C' bit b2 set) to alter the credentials of a specific SEMS Application.

If tag 'C9' is not present, no restrictions apply to the Application Specific Parameters set at the creation of an AMSD or SD.

Table 4-6: Application Specific Parameters – Tag 'C9'

Tag	Length (bytes)	Description	Presence
'82'	4 or 8	Allowed parameter values for the inbound extradition of Applications / ELFs (See Table 4-7.)	Optional
'83'	4	Allowed parameter values for the deletion of associated Applications (See Table 4-7.)	Optional
'87'	4 or 8	Allowed parameter values for the outbound extradition of associated Applications / ELFs (See Table 4-7.)	Optional

The Subject Identifier (tag '5F20') uniquely identifies the owner of the public key that is certified.

Sub-tag '82', '83', and '87' of tag 'C9' define the allowed values that may be assigned to the sub-tags '82', '83', and '87' of the Application Specific Parameters (ASP) that may be present in a SEMS_INSTALL_FOR_INSTALL command. A similar logic is applied for tags '82', '83', and '87' of CERT.SP.AUT; the main difference is that bits as defined in GlobalPlatform Common Implementation Configuration ([CIC]) Table 3-5 no longer refer to each pre-defined acceptance policy but rather to pre-set combinations of acceptance policies (see Table 4-7).

If tag '82' of the ASP is coded on 1 byte and if the bit of the lowest byte of tag '82' of CERT.SP.AUT at the position of tag '82' of the ASP shifted by 3 to the right is set, then tag '82' is allowed; otherwise it is not.

If tag '82' of the ASP is coded on 2 bytes:

- The 32 least significant bits of tag '82' of CERT.SP.AUT are used to verify whether the lowest byte of tag '82' of the ASP is allowed.
- The 32 most significant bits of tag '82' of CERT.SP.AUT are used to verify whether the highest byte of tag '82' of the ASP is allowed.

Table 4-7 provides the explicit values of the masks for the different values of bits b8-b4 of tag '82', '83', and '87'.

Table 4-7: Mapping Table for ASPs

	Bits b8-b4			
Value	**Dec**	**Hex**	**Bin**	**Mask**
00 .. 07	0	00	00000	00000001
08 .. 0F	1	01	00001	00000002
10 .. 17	2	02	00010	00000004
18 .. 1F	3	03	00011	00000008
20 .. 27	4	04	00100	00000010
28 .. 2F	5	05	00101	00000020
30 .. 37	6	06	00110	00000040
38 .. 3F	7	07	00111	00000080
40 .. 47	8	08	01000	00000100
48 .. 4F	9	09	01001	00000200
50 .. 57	10	0A	01010	00000400
58 .. 5F	11	0B	01011	00000800
60 .. 67	12	0C	01100	00001000
68 .. 6F	13	0D	01101	00002000
70 .. 77	14	0E	01110	00004000
78 .. 7F	15	0F	01111	00008000
80 .. 87	16	10	10000	00010000
88 .. 8F	17	11	10001	00020000
90 .. 97	18	12	10010	00040000
98 .. 9F	19	13	10011	00080000
A0 .. A7	20	14	10100	00100000
A8 .. AF	21	15	10101	00200000
B0 .. B7	22	16	10110	00400000
B8 .. BF	23	17	10111	00800000
C0 .. C7	24	18	11000	01000000
C8 .. CF	25	19	11001	02000000
D0 .. D7	26	1A	11010	04000000
D8 .. DF	27	1B	11011	08000000
E0 .. E7	28	1C	11100	10000000
E8 .. EF	29	1D	11101	20000000
F0 .. F7	30	1E	11110	40000000
F8 .. FF	31	1F	11111	80000000

For example, if tag '82' of ASP is '00000' and if the bit of tag '82' of CERT.SP.AUT at position 0 is set, then tag '82' is allowed.

If tag '82' of ASP is 'C0' (tag '82' shifted by 3 equals '11000b' = '18') and if the bit of tag '82' of CERT.SP.AUT at position 24 is not set, tag '82' is not valid.

Table 4-8: Policy Byte for Pre-created Instance/ELF – Tag '4D'

b8	b7	b6	b5	b4	b3	b2	b1	Description
-	-	-	-	-	-	-	1	Allow INSTALL command (for registry update, extradition, personalization) on the AID.
-	-	-	-	-	-	1	-	Allow DELETE command on the AID.
-	-	-	-	-	1	-	-	Allow the ELF update for the ELF AID.
-	-	-	-	1	-		-	Allow SET/GET STATUS for the AID.
x	x	x	x	x	-	-	-	RFU

4.5.2.1 Supplementary CA

This specification allows the usage of CA hierarchies mandating a CA as root and conditional supplementary CAs. The certificates of these supplementary CAs shall conform to the description of CERT.SP.AUT as shown in Table 4-2. The order of the SP certificates retrieved from a SEMS script has to match the order of verification of the certificates in the chain. The process of the SP’s certificate verification is shown in Figure 4-5.

Tag '42' of a child certificate shall match tag '5F20' of its parent certificate.

Tag '5F37' of a child certificate shall be verified with the public key contained in tag '7F49' of its parent certificate.

4.5.3 CERT.CASD.ECDSA

The structure of the CERT.CASD.ECDSA certificate is shown in Table 4-9.

Table 4-9: CERT.CASD.ECDSA

<table>
<tr><th>Tag</th><th>Length (bytes)</th><th colspan="5">Description</th><th>Presence</th></tr>
<tr><td rowspan="13">'7F21'</td><td rowspan="13">Var.</td><td colspan="5">CERT.CASD.ECDSA</td><td></td></tr>
<tr><td><b>Tag</b></td><td><b>Length</b></td><td colspan="3"><b>Description</b></td><td><b>Mandatory</b></td></tr>
<tr><td>'93'</td><td>16</td><td colspan="3">SE Serial Number</td><td>Mandatory</td></tr>
<tr><td>'42'</td><td>1..16</td><td colspan="3">CA-KLCC Identifier</td><td>Mandatory</td></tr>
<tr><td>'5F20'</td><td>1..16</td><td colspan="3">Subject Identifier (a.k.a. certificate holder)</td><td>Mandatory</td></tr>
<tr><td>'95'</td><td>1</td><td colspan="3">Key Usage, shall be '82' meaning digital signature verification</td><td>Mandatory</td></tr>
<tr><td>'5F24'</td><td>4</td><td colspan="3">Expiration Date of CA(YYYYMMDD, BCD format)</td><td>Mandatory</td></tr>
<tr><td>'45'</td><td>Var.</td><td colspan="3">Security Domain Image Number; '00' if SEMS Application is not implemented as SD</td><td>Mandatory</td></tr>
<tr><td>'5F37'</td><td>64</td><td colspan="3">Signature (Concatenation of r and s coded as ‘r || s’)</td><td>Mandatory</td></tr>
<tr><td rowspan="4">'7F49'</td><td rowspan="4">Var.</td><td colspan="3">Public key PK.CASD.ECDSA</td><td></td></tr>
<tr><td><b>Tag</b></td><td><b>Length</b></td><td><b>Description</b></td><td></td></tr>
<tr><td>'86'</td><td>65</td><td>Public key PK.CASD.ECDSA coded as '04' || X || Y</td><td>Mandatory</td></tr>
<tr><td>'8F'</td><td>1-2</td><td>Key type according to [GPCS] Table B-2</td><td>Optional</td></tr>
</table>

The Subject Identifier (tag '5F20') uniquely identifies the owner of the public key that is certified. In the optionally supported case of CA hierarchy, the content of tag '42' of a child certificate shall match the content of tag '5F20' of its parent certificate.

The signature is computed using ECDSA, specified in [GPCS] section B.4.3, on all inner TLVs ('93'... '7F49') excluding tag '5F37'. The tags shall be processed in the order described in Table 4-10. The signature is computed using the CA-KLCC (root) private key; i.e. SK.CA-KLCC.ECDSA.

The public ECC key stored in tag '86' uses uncompressed encoding as described in [TR-03111].

The signature stored in tag '5F37' is coded in plain format as specified in [TR-03111]; i.e. it is the concatenation of the byte string representation of r and s.

Table 4-10: Data Signed to Generate the Signature of the CERT.CASD.ECDSA Certificate

<table>
<tr><th>Tag</th><th>Length (bytes)</th><th colspan="5">Description</th><th>Presence</th></tr>
<tr><td rowspan="12">'7F21'</td><td rowspan="12">Var.</td><td colspan="5">CERT.CASD.ECDSA</td><td>Mandatory</td></tr>
<tr><th>Tag</th><th>Length</th><th colspan="3">Description</th><th>Mandatory</th></tr>
<tr><td>'93'</td><td>16</td><td colspan="3">SE Serial Number</td><td>Mandatory</td></tr>
<tr><td>'42'</td><td>1..16</td><td colspan="3">CA-KLCC identifier</td><td>Mandatory</td></tr>
<tr><td>'5F20'</td><td>1..16</td><td colspan="3">Subject Identifier (a.k.a. certificate holder)</td><td>Mandatory</td></tr>
<tr><td>'95'</td><td>1</td><td colspan="3">Key Usage shall be '82', meaning digital signature verification.</td><td>Mandatory</td></tr>
<tr><td>'5F24'</td><td>4</td><td colspan="3">Expiration Date of CA(YYYYMMDD, BCD format)</td><td>Mandatory</td></tr>
<tr><td>'45'</td><td>1</td><td colspan="3">00</td><td>Mandatory</td></tr>
<tr><td rowspan="4">'7F49'</td><td rowspan="4">67</td><td colspan="3">Public key PK.CASD.ECDSA</td><td>Mandatory</td></tr>
<tr><th>Tag</th><th>Length</th><th>Description</th><th></th></tr>
<tr><td>'86'</td><td>65</td><td>Public key PK.CASD.ECDSA coded as '04' || X || Y</td><td>Mandatory</td></tr>
<tr><td>'8F'</td><td>1-2</td><td>Key type according to [GPCS] Table B-2</td><td>Optional</td></tr>
</table>

4.5.4 CERT.CASD.AUT

The CERT.CASD.AUT is specified in [Amd A] Table 3-4.

For an RSA key with a length of 1024 bits, the signature algorithm is defined in [GPCS] section B.3.1.1.

For an RSA key length strictly longer than 1024 bits, the signature algorithm is defined in [GPCS] section B.3.2.1.

4.5.5 CERT.CASD.ECKA

The CERT.CASD.ECKA is specified in [Amd A] Table 3-6. The signature algorithm is defined in [GPCS] section B.4.4.

4.6 Secured SEMS Script

4.6.1 Encryption and Signature Creation

The entity processing a SEMS script in plaintext knows two ECC keys:

- A private ECDSA key, denoted SK.SP.AUT. The SP.AUT key pair has been generated by the entity that processes the SEMS script. The private key SK.SP.AUT is used to sign the SEMS scripts while the public key PK.SP.AUT is included in the CERT.SP.AUT certificate embedded in the SEMS scripts and is used to verify the SEMS script signature by SEMS Application.
- An ECDH key, denoted PK.CA-SEMS.ENC, to generate a Shared Secret (ShS) used to derive an AES key, denoted K. In this version of the specification, the key size of this AES key K is 128 bits.

Figure 4-2 shows the encryption and signature processes of a plaintext script made of multiple SEMS commands. Each plaintext message contains the decryption key (Ki) to be used for the decryption of the following message. Each message contains a hash of the following message, and only the first message is signed with the signature key.

In Figure 4-2, the upper rounded box shows the confidentiality chaining and the lower box shows the integrity chaining (hashing and signature generation). Figure 4-3 shows the processing flow of plaintext SEMS commands.

All encryption AES keys are randomly generated. For each given ciphertext, the chained encryption mechanism requires the receiving party to decrypt the previously received message and extract the embedded deciphering key before being able to decrypt the next received message.

The first plaintext message consists of a randomly generated key K_1 only (no payload). Each following plaintext message consists of the concatenation of the payload (i.e. each SEMS command (PT_1-PT_n)) and a different randomly generated key (K_2-K_n). The key in each message is used to encrypt the following SEMS command.

The key K, used to encrypt the first random key K_1, is derived with the KDF from a shared secret established with the private key eSK.SP.ENC of an ECDH key generated by the SP combined with PK.CA-SEMS.ENC (see section 4.6.3). The private key eSK.SP.ENC is randomly generated each time that a script is generated. The public key ePK.SP.ENC is stored in the Authentication Frame defined in Table 5-1.

Each plaintext block (PT_i + K_{i+1}) is encrypted with the previously encrypted key K_i to create the corresponding ciphertext block CT_i.

The signature of the whole encrypted SEMS script (Concatenation of CT_0…CT_n) takes place in two steps:

1. The iteratively chained hashing process starts from the last ciphertext (CT_n). Each of the following hash values H_i is computed from the concatenation of each ciphertext CT_{i+1} and the previously computed hash H_{i+1}. The last encrypted command CT_n is concatenated with a random 32-byte array.
2. The signature of the concatenation of the first ciphertext, containing ePK.SP.ENC and the final hash value H_1 (as computed in the previous step). The signature is calculated with the SK.SP.AUT private key using ECDSA as described in [GPCS] section B.4.3.

Finally, the CERT.SP.AUT certificate, defined in section 4.5.1, is added to the secured script.

Figure 4-2: SEMS Encryption and Signing Scheme

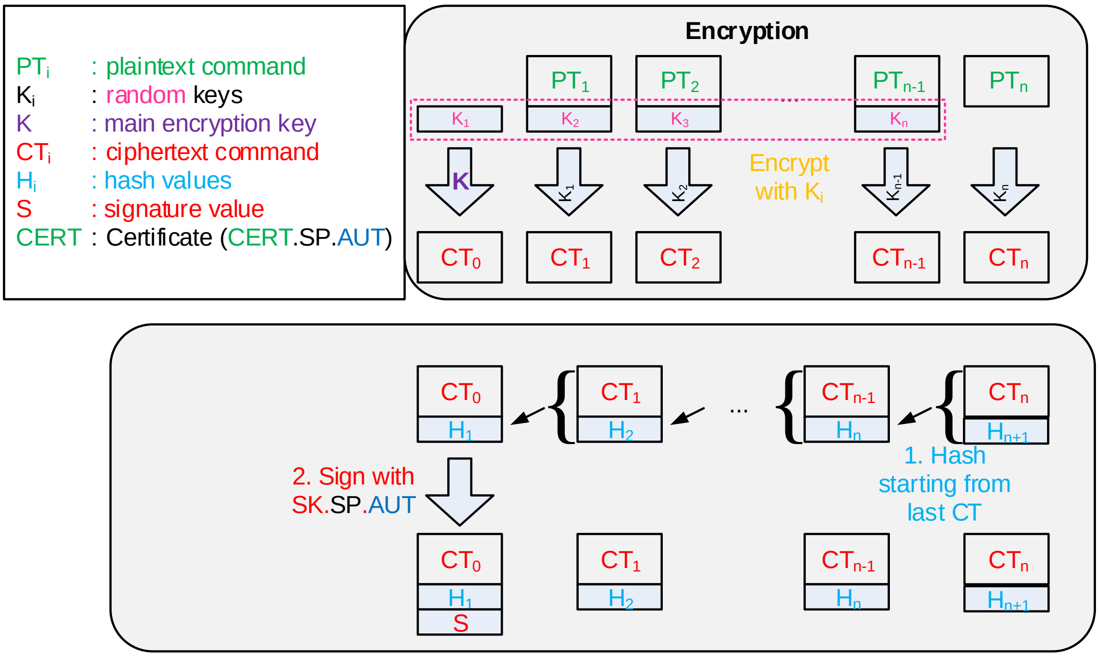
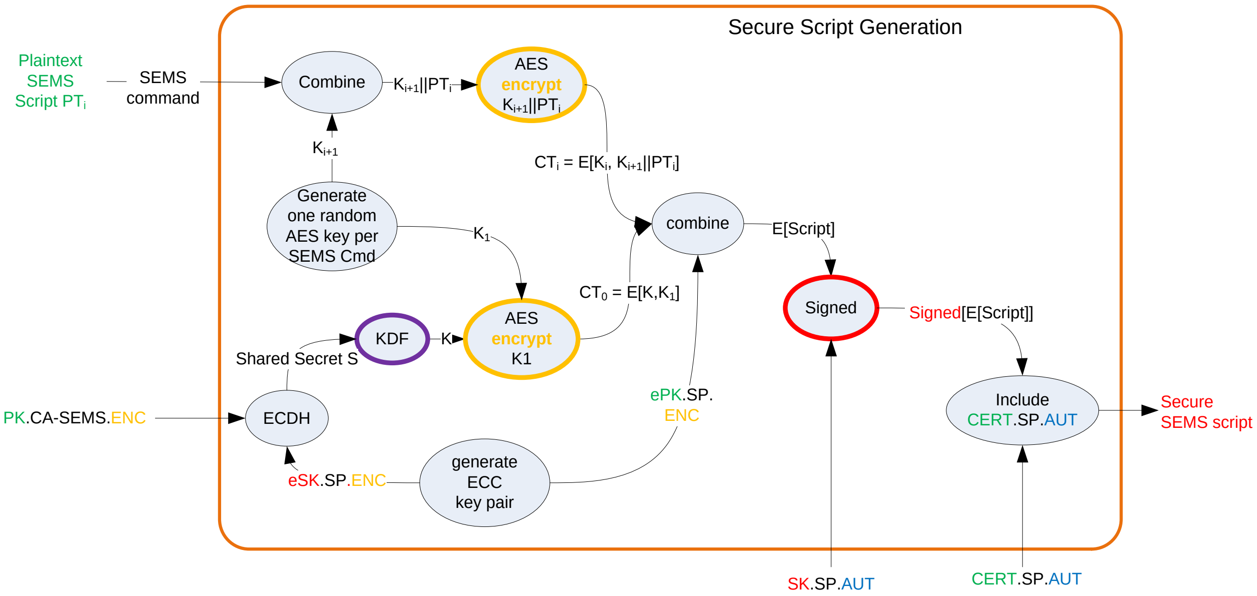


Figure 4-3: Generation of a Secured SEMS Script



4.6.2 Decryption and Signature Verification

For the decryption and signature verification process, the SEMS Application is personalized with two ECC keys:

- An ECDSA public key, denoted PK.CA-SEMS.AUT, used to verify the first received CERT.SP.AUT certificate
- An ECDH private key, denoted SK.CA-SEMS.ENC, used to generate a Shared Secret which itself is later used to derive an AES key, denoted K

Figure 4-4 shows the processing flow of a secured SEMS script:

1. Processing of CERT.SP.AUT (defined in Table 4-2):
   a. Verify the authenticity of a CERT.SP.AUT with the PK.CA-SEMS.AUT key stored by the SEMS Application. Once the CERT.SP.AUT has been authenticated, the SEMS Application retrieves the PK.SP.AUT key and the content of tag '53' if present. This process is illustrated in Figure 4-5.
   b. Optionally, in a CERT.SP.AUT certificate chain, verify the authenticity of the supplementary received CERT.SP.AUT with the PK.SP.AUT retrieved from the previous CERT.SP.AUT. Once the CERT.SP.AUT signature has been verified, the SEMS Application retrieves the PK.SP.AUT key and the content of tag '53' if present. Step 1.b is repeated until all CERT.SP.AUT certificates of the chain are verified, as shown in Figure 4-5.
2. Processing of the Authentication Frame described in Figure 4-4:
   a. Verify the script signature using the latest retrieved PK.SP.AUT.
      i. The ECDSA signature is computed over the Authentication Frame without the signature; i.e. ePK.SP.ENC || AES[K, K_1] || H_1.
   b. Compute the AES key K.
      i. Retrieve the ePK.SP.ENC key from the Authentication Frame.
      ii. Compute the shared secret from ePK.SP.ENC and SK.CA-SEMS.ENC using the ECDH cryptographic algorithm.
      iii. Derive the AES key K from the computed shared secret using the KDF function described in section 4.6.3.5.
      iv. Decrypt the AES key K_1 with AES in (with no padding) using the AES key K.
   c. Retrieve H_1, the SHA-256 computed over the first encrypted SEMS command concatenated with H_2, used to verify the integrity of the first SEMS command, from the Authentication Frame.
3. Processing of secured SEMS commands (as described in Figure 4-2):
   a. Verify the integrity of the SEMS command by comparing the latest retrieved SHA-256 with the computed SHA-256 over AES-CBC[K_{n-1}, K_n || SEMS command] || SHA-256. If the retrieved SHA-256 value matches the computed one, then the integrity of the SEMS command is verified.
   b. Extract the SHA-256 value from the secured SEMS command.
   c. Decrypt the AES key K_n concatenated with the SEMS command using the AES cryptographic algorithm in the CBC mode with the former stored AES key K_{n-1}. Retrieve the AES keys from the decrypted secured SEMS command for the next consecutive operation round of this step (step 3).
   d. Execute the decrypted SEMS command.

Figure 4-4: Authentication and Decryption Process of a Secured Script

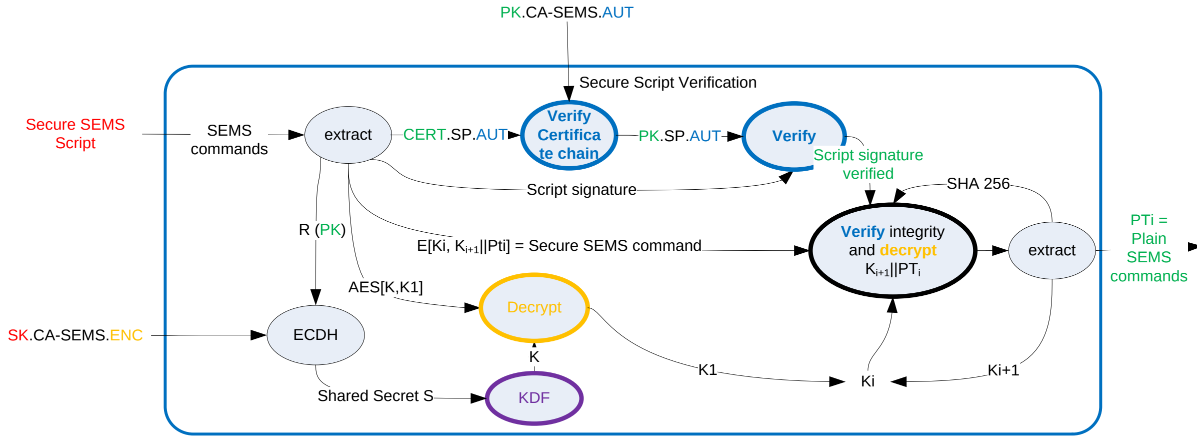


Figure 4-5: Certificate Chain Verification

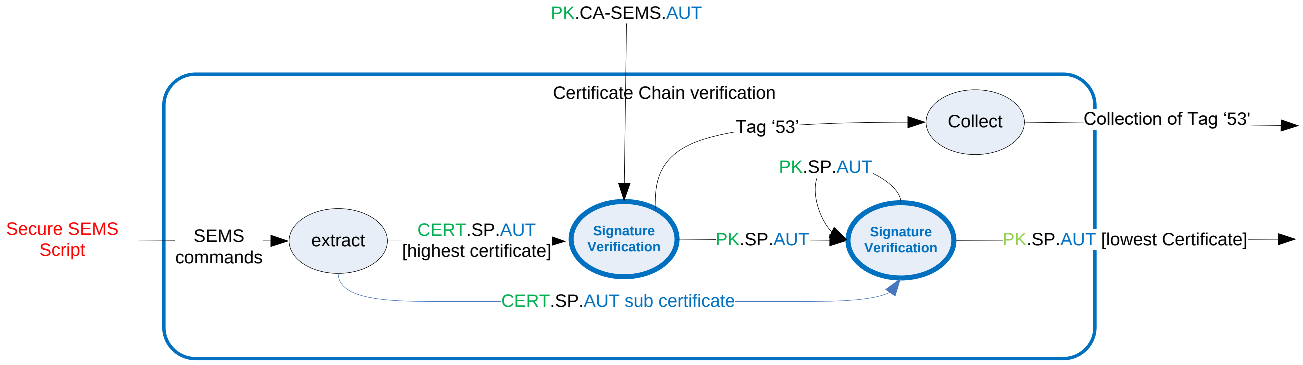
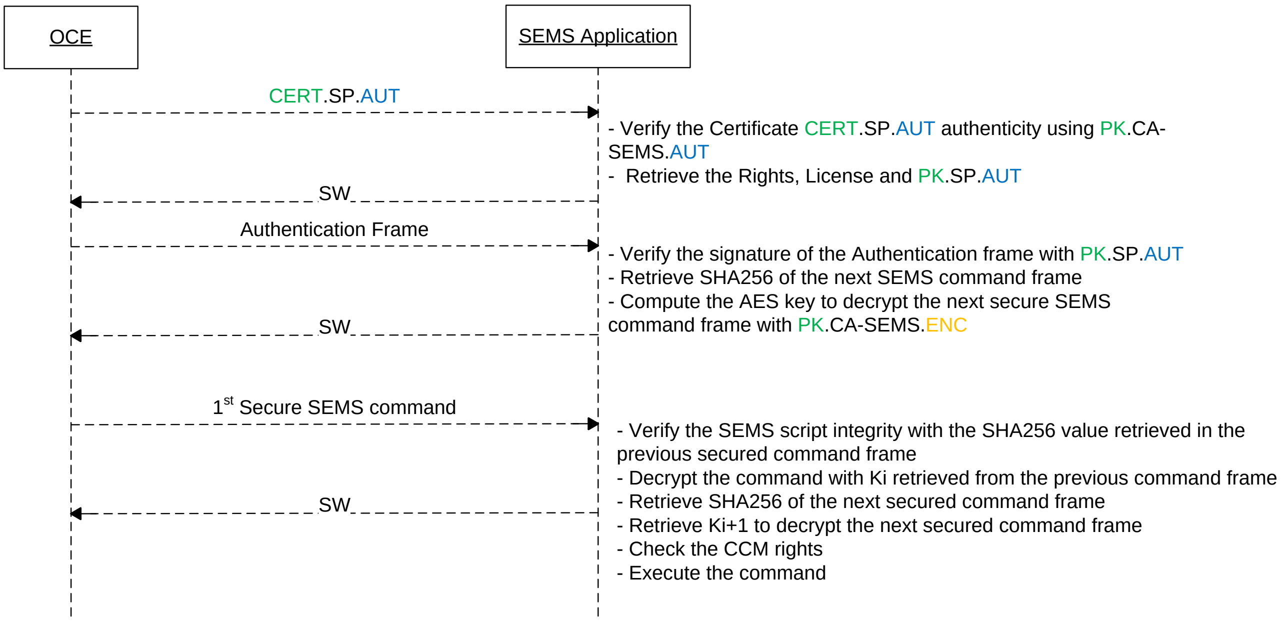


Figure 4-6: Sequence Diagram – Authentication and Decryption of a Secured Script


OCE
SEMS Application
CERT.SP.AUT
- Verify the Certificate CERT.SP.AUT authenticity using PK.CA-SEMS.AUT
- Retrieve the Rights, License and PK.SP.AUT
SW
Authentication Frame
- Verify the signature of the Authentication frame with PK.SP.AUT
- Retrieve SHA256 of the next SEMS command frame
- Compute the AES key to decrypt the next secure SEMS command frame with PK.CA-SEMS.ENC
SW
1st Secure SEMS command
- Verify the SEMS script integrity with the SHA256 value retrieved in the previous secured command frame
- Decrypt the command with Ki retrieved from the previous command frame
- Retrieve SHA256 of the next secured command frame
- Retrieve Ki+1 to decrypt the next secured command frame
- Check the CCM rights
- Execute the command
SW

An authentication starts after a successful verification of the Authentication Frame signature.

4.6.2.1 Terminate Session and Reset Authentication State

The SEMS Application shall terminate an authentication session and reset the authentication state (e.g. clear session keys and chaining data), on any of the following:

- The processing of SELECT command of the SEMS Application or of the SEMS Updater, defined in section 6.3
- Successful processing of the STORE DATA command, defined in section 6.4
- Receipt of a CERT.SP.AUT, defined in section 4.5.2, signed with the SK.CA-SEMS.AUT, defined in section 4.5.1
- A failed verification of a CERT.SP.AUT certificate, defined in section 4.5.2
- A failed verification of the integrity of a secured SEMS command, defined in section 5.1.2
- PROCESS SCRIPT COMMAND APDU, defined in section 6.2, with P1 = '80'

Note: The session is not closed if the processing of STORE DATA fails.

4.6.3 Cryptographic Algorithm Details

4.6.3.1 ECC Curve

The current version of this specification supports only curves with ECC key lengths of 256 bits according to [GPCS] Table B-1. CERT.SP.AUT tag '7F49' sub-tag '8F' (see Table 4-2) may explicitly identify the ECC curve according to [GPCS] Table B-2. If CERT.SP.AUT does not include this optional parameter information, the ECC public key PK.SP.AUT (contained in sub-tag '86' of tag '7F49' as stated in Table 4-2) shall be based on the ECC curve brainpoolP256r as specified in the Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation [RFC 5639].

4.6.3.2 ECDH

The Elliptic Curve Diffie-Hellman shall be performed according to the NIST Special Publication 800-56A [SP-800-56A].

4.6.3.3 ECDSA

The ECDSA shall be performed as specified in [GPCS] section B.4.3.

4.6.3.4 AES-CBC

The AES in CBC mode for encryption and decryption shall be performed as specified in the NIST Special Publication 800-38A [SP-800-38A] with Initial Vector equal to 0 and no padding.

4.6.3.5 Key Derivation Function (KDF)

In this version of this specification, the KDF returns the most significant 128 bits of the applied Hashing algorithm SHA-256 as defined in [SP-800-56A]. These returned most significant 128 bits become the AES key K.

4.6.3.6 Recommendation

Since the SK.CA.SEMS.ENC key is a static key, the SEMS script encryption scheme, described in section 4.6.1, does not have the Perfect Forward Secrecy property. A SEMS script should not contain secrets. It is therefore strongly recommended to use the confidential key setting, based on a pull model, described in section 4.8.2, to personalize keys in a secure domain rather than the SEMS_PUT_KEY described in section 7.14.

Note that since the private keys to sign the CERT.SP.AUT or to generate the signature in the Authentication Frame are securely stored in the servers, it is not possible to forge the signature of a CERT.SP.AUT certificate or of an Authentication Frame.

By design, a SEMS script can be replayed. To prevent a malicious phone application from executing a SEMS_DELETE command, the phone application that executes a script creating an application and a script deleting the created Applet shall be the same. The phone application is identified by a unique identifier retrieved from the Operating System.

4.7 SEMS Application Software Architecture

There shall exist a trust relationship between the Security Domain receiving CAPDUs via the virtual I/O interface, as described in section 4.7.1.1, and the SEMS Application that generated and forwarded these CAPDUs. This trust relationship is based on the CERT.SP.AUT certificate holder identifier (content of tag '5F20') saved in the registry entry during the install operation and checked by the OPEN during APDU forwarding on the virtual I/O interface described in section 4.7.1.

4.7.1 Trusted APDU Transfer through the Virtual I/O

The SEMS Application uses a virtual I/O interface owned by the OPEN to transfer APDUs virtually to and from the Target Application (selected by the SEMS Application via initial processing of a SEMS_SELECT command, as defined in section 7.3). This virtual I/O interface provided by the OPEN enables the on-card entity (SD / Application instance) to process SEMS Application-originated APDU commands exactly as if they were sent directly via a physical I/O interface, and enables the SEMS Application to forward GlobalPlatform CCM commands to specific Security Domains. The access to this virtual I/O is reserved for the SEMS Application. This rule shall be enforced by the OPEN.

Implementing and interacting with this type of virtual I/O interface requires the use of a proprietary API, which is out of scope of this specification.

The support of supplementary logical channels on this virtual interface type is not defined in this version of the specification.

Here, the goal is to approach the existing standardized entry point of every on-card Application, namely the `process` method, to enable Command APDUs and Response APDUs to be managed like regular "external" APDUs; i.e. CAPDUs going through the communication layer for the active interfaces (contact, contactless, etc.). The `APDU` object used for the virtual I/O interface should be separated from the `APDU` object processed by the SEMS Application and managed by the OPEN on the virtual I/O interface.

4.7.1.1 Trust Relationship on the Virtual I/O Accessing the Target Application

Initially, the SEMS Application has a trust relationship only with the Security Domain associated with the SEMS Application. This SD might be the ISD, but could be a specific SEMS Application Provider SSD (with AM privilege).

To establish an additional trust relationship, the OPEN shall compare the certificate holder of the currently processed SEMS script and the certificate holder which had been used to create the Target Application (the application that the CAPDU on the virtual I/O shall be forwarded to). If the OPEN determines that the entities are identical, a trust relationship is established between the SEMS Application and the Target Application. For this purpose, the OPEN queries the SEMS Application for the content of tag '5F20' of the currently active CERT.SP.AUT, and a match is performed on the saved content for the Target Application. The certificate holder identifier (tag '5F20') is saved by the OPEN during the creation of an Application during successful processing of a SEMS_INSTALL_FOR_INSTALL command (as defined in section 7.9), or during the successful processing of a SEMS_INSTALL_FOR_LOAD command for loaded ELFs (see section 7.7). Note that when the Receiving Entity is a SEMS Updater (as described in section 4.15), this query would fail but a separate callback might be added to evaluate the content of tag '4C' retrieved from the SEMS Application.

Except for the Security Domain associated with the SEMS Application (as explained above), an Application that is not associated with a certificate holder never trusts a SEMS script.

4.7.1.2 Virtual I/O Runtime Behavior

The SEMS Application generates the CAPDU on the buffer provided by a separate APDU object before approaching the OPEN dispatch mechanism in order to forward this command to the Target Application. In addition to the traditional APDU dispatch mechanism (see [GPCS] section 6.3), the Runtime Behavior is altered as the OPEN shall perform the following checks when a CAPDU is forwarded on the virtual I/O interface:

- The Receiving Entity (on the physical I/O interface) is either SEMS Application or SEMS Updater (identified by its Application instance AID).
- The Target Application exists in the GlobalPlatform Registry and its Application Life Cycle state equals SELECTABLE.
- If the Target Application has the Security Domain privilege, it shall have been created by a SEMS script (see section 7.9).

If all the above checks are passed, the OPEN shall pass the command towards the Target Application (if already selected via SEMS_SELECT processing defined in section 7.3) through this virtual I/O interface. Similar to the traditional CAPDU dispatch described in [GPCS] section 6.3, the OPEN shall initialize the required components of the runtime environment to accept an incoming CAPDU on the virtual I/O interface for this Target Application.

If the Target Application trusts the SEMS script (as described in section 4.7.1.1) then, if the Target Application queries the current Security Level (according to [GPCS] Table 10-1), the response shall indicate the Security state of the current secured SEMS script (AUTHENTICATED || C_MAC || C_DECRYPTION and conditionally R_MAC || R_ENCRYPTION). In this case, the GlobalPlatform API for secure communication (unwrap, for example) returns without any operation performed, since the actual interface is the virtual I/O. The SEMS Application generates the CAPDU, conditionally including the Secure Messaging bit in the CLA byte.

If the Target Application does not trust the SEMS script (as described in section 4.7.1.1) then, if the Target Application queries the current Security Level (according to [GPCS] Table 10-1), the response shall be NO_SECURITY_LEVEL.

The Response APDU is conveyed back to the SEMS Application.

The OPEN has to retrieve the AID of the Application that has been selected by the SEMS Application. Indeed, only this Application may receive CAPDUs from the SEMS Application and send back RAPDUs through the virtual I/O interface. The OPEN shall return '6999' if no Application selection took place beforehand through the SEMS_SELECT command execution on the virtual I/O interface.

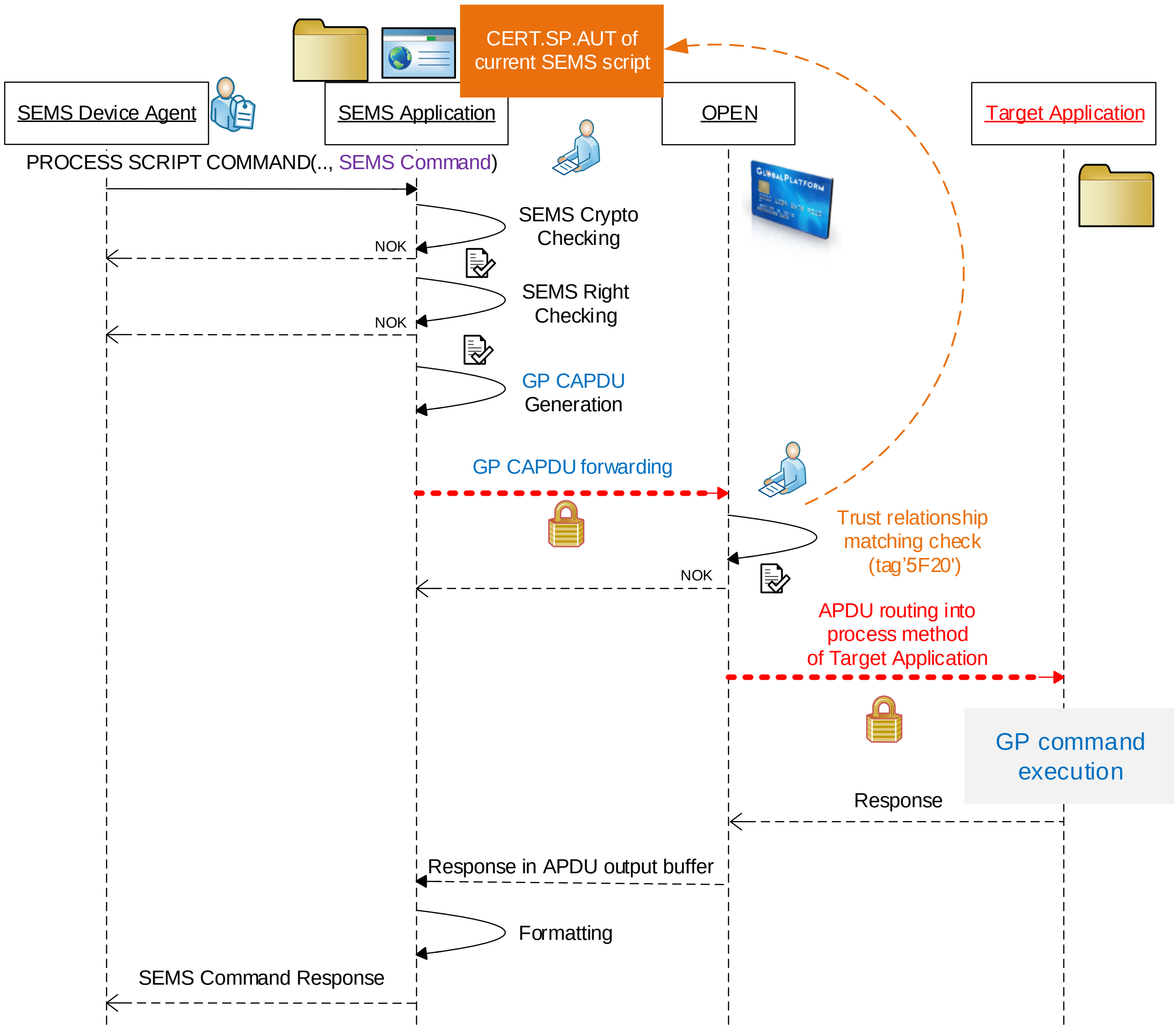
After the selected Application has processed the CAPDU and placed the RAPDU, the OPEN shall clear the non-filled part of the APDU buffer in order to avoid any information leakage.

The processing of a SEMS command is described in Figure 4-1:

- The SEMS Application receives the SEMS command via the PROCESS SCRIPT COMMAND APDU command.
- Once the integrity and authentication have been checked, the SEMS command is decrypted and the SEMS Application generates the corresponding APDU. In this version of the specification, the state of this generated APDU reference is automatically set to `APDU.STATE_FULL_INCOMING`.
- If the SEMS command is a SEMS_SELECT command, the OPEN processes it to select a Target Application on the virtual I/O interface.
- The generated APDU is forwarded through the virtual I/O interface to the `process` method of the Target Application.

- The Target Application may process the APDU as though the APDU had been received via a physical I/O interface.
- The OPEN shall ensure the following modified behavior of the API provided with the `APDU` reference:
  - `getIncomingLength` method returns the incoming data length of the generated APDU.
  - `getProtocol` method returns protocol and media used for the original APDU received by the Receiving Entity (SEMS Application).
  - Methods `isSecureMessagingCLA`, `isCommandChainingCLA`, `isISOInterindustryCLA`, and `isValidCLA` refer to the CLA byte in the APDU generated by the SEMS Application.
  - `receiveBytes` method is not supported with special behavior in the current version of this specification and simply returns '0' unless one of the conditions checked by Java Card implementation already throws an `APDUException` as documented in `APDU` [JC API].
  - `sendBytes` method saves `len` more bytes from APDU buffer at specified offset `bOff`. `sendBytesLong` saves `len` more bytes from `outData` byte array starting at specified offset `bOff` and may re-use the APDU buffer for this purpose according to `APDU`.
  - `setOutgoing` method sets the data transfer direction to outbound and obtains the expected length of response with regards to the APDU generated by the SEMS Application. `setOutgoingLength` method sets the actual length of response data on the APDU reference generated by the SEMS Application.
  - `setOutgoingAndSend` is a combination of `setOutgoing`, `setOutgoingLength` followed by `sendBytes`. Once this method is invoked, the methods `sendBytes` and `sendBytesLong` cannot be invoked.
- The OPEN shall ensure the following modified behavior of the API provided with the `SecureChannel` interface:
  - `getSecurityLevel` returns a Security Level value based on both the trust relationship existing (between the certificate holder and the Target Application) and the Security state of the secured SEMS script. If there is no trust relationship, then the returned value shall be NO_SECURITY_LEVEL.
  - Methods `wrap` and `unwrap` do not have to perform any operation on the payload. Nevertheless, the returned length information requires a corresponding Security state of the secured SEMS script (regarding conditional C_DECRYPTION and R_ENCRYPTION, see above).
  - The `decryptKey` method may throw an exception if the key strength of the contained key material is higher than the script encrypting key provides, depending on the platform issuer’s security policy.
- The conveyed response from the Target Application is forwarded back to the SEMS Application through the same virtual I/O interface. The PROCESS SCRIPT COMMAND APDU returns the response to the SEMS Device Agent as described in section 6.2.6.
- At the end of script processing, or if a new Target Application is selected via SEMS_SELECT command processing (see section 7.3), or when a SEMS Application becomes deselected, the Application currently selected on the virtual I/O interface becomes deselected. Therefore, the `deselect` method is invoked and its transient memory assigned with `JCSystem.CLEAR_ON_DESELECT` is released if the same Application is not selected on another (physical) I/O interface.

Figure 4-7: SEMS Command Forwarding to an Application through “Virtual I/O Interface”


CERT.SP.AUT of current SEMS script
SEMS Device Agent
SEMS Application
OPEN
Target Application
PROCESS SCRIPT COMMAND(.., SEMS Command)
SEMS Crypto Checking
NOK
SEMS Right Checking
NOK
GP CAPDU Generation
GP CAPDU forwarding
Trust relationship matching check (tag’5F20’)
NOK
APDU routing into process method of Target Application
GP command execution
Response
Response in APDU output buffer
Formatting
SEMS Command Response

4.8 Security Domain Creation and Confidential Set-up of Secure Channel Key Set

4.8.1 Security Domain Creation

A Security Domain may be created with the SEMS_INSTALL_FOR_INSTALL command.

To create a Security Domain with Authorized Management, the CCM rights defined in CERT.SP.AUT shall allow the creation of a Security Domain with Authorized Management.

To create a Security Domain without Authorized Management, the CCM rights defined in the CERT.SP.AUT shall allow the creation of a Security Domain with or without Authorized Management.

4.8.2 Confidential Key Setting

The SEMS Application supports the following confidential key setting scenarios:

- The scheme, defined in section 4.8.2.1, is a mandatory scheme implemented with two SEMS commands SEMS_GET_DATA and SEMS_PULL_KEY.
- A variant of GlobalPlatform confidential key setting scenario #1, defined in section 4.8.2.2, is an optional scheme implemented with the two SEMS commands SEMS_GET_DATA and SEMS_PULL_KEY_GP.
- The GlobalPlatform confidential key setting scenario #3, defined in [Amd A], is an optional key setting implemented with the two SEMS commands SEMS_GET_DATA and SEMS_PULL_KEY_GP.

SEMS_GET_DATA, defined in section 7.12, is used to retrieve a certificate stored in the SE.

SEMS_PULL_KEY, defined in section 7.11, and SEMS_PULL_KEY_GP, defined in section 7.13, are used to trigger the on-board generated keys (OBGK).

4.8.2.1 SEMS Confidential Set-up of Secure Channel Key Set

The SEMS_PULL_KEY command is used to pull the Security Domain SCP key set generated and loaded into the newly created Security Domain. The SEMS_PULL_KEY command sends an ECC public key PK.SP.ENC.S4 so that it can be used by the SEMS Application on behalf of the Service Provider to apply encryption on the RGK that is used consecutively to generate the different SCP keys.

The integrity and the authenticity of PK.SP.ENC.S4 is ensured by the SEMS script, which is signed by the Service Provider using the SK.SP.AUT private key and the SEMS integrity chaining described in Figure 4-2 on page 42.

Figure 4-8 depicts a sequence of SEMS commands to create a Security Domain and personalize the Security Domain keys with SCP keys generated on-board.

The SE Application is personalized with a CERT.CASD.ECDSA certificate, containing the public key PK.CASD.ECDSA, and a private key SK.CASD.ECDSA.

The SEMS_PULL_KEY triggers the on-board key generation process and conveys the public key PK.SP.ENC.S4. The SEMS Application performs this random key generation on the SE to derive SCP keys as described in [Amd A] section 3.5.2. The randomly generated key becomes AES encrypted using a key obtained by the following process:

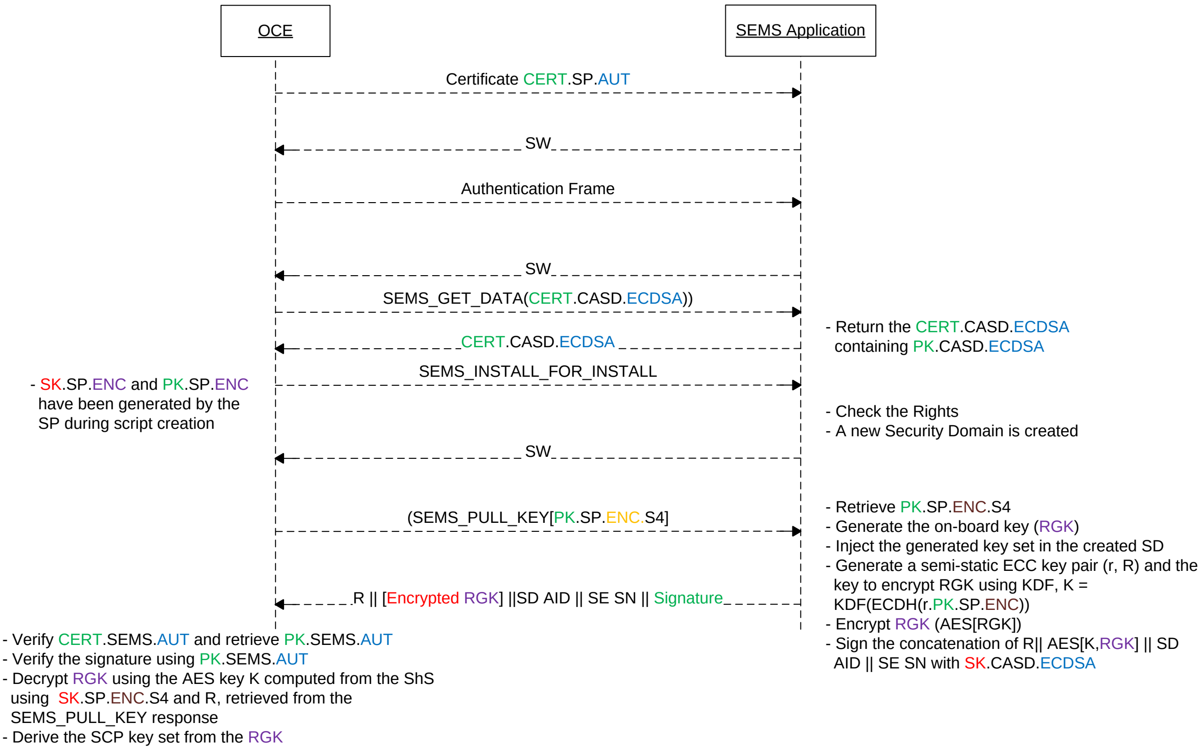
- A semi-static ECC key pair (r, R) is generated on-board by the SEMS Application.
- A Shared Secret (ShS) being the result of the ECDH of the (static) private key r of the semi-static ECC key pair and the public key PK.SP.ENC.S4 is computed.
- The AES key K is derived from the KDF defined in section 4.6.3.5, applied to the ShS.

- The randomly generated key is AES encrypted using the key K.

A signature is computed over the Public key R of the semi-static ECC key concatenated with the SD AID, SE Serial Number, and the AES encrypted on-board generated key.

The ECDSA private key SK.CASD.ECDSA is used to generate the signature with the ECDSA algorithm as defined in [GPCS] section B.4.3. The signature is computed over the concatenation of the semi-static ECC key R, the AES Encrypted RGK with the AES key K, the SD AID, and the SE SN.

Figure 4-8: Security Domain Creation and Pull Key Model


OCE
SEMS Application
Certificate CERT.SP.AUT
SW
Authentication Frame
SW
SEMS_GET_DATA(CERT.CASD.ECDSA))
- Return the CERT.CASD.ECDSA containing PK.CASD.ECDSA
CERT.CASD.ECDSA
SEMS_INSTALL_FOR_INSTALL
- SK.SP.ENC and PK.SP.ENC have been generated by the SP during script creation
- Check the Rights
- A new Security Domain is created
SW
(SEMS_PULL_KEY[PK.SP.ENC.S4]
- Retrieve PK.SP.ENC.S4
- Generate the on-board key (RGK)
- Inject the generated key set in the created SD
- Generate a semi-static ECC key pair (r, R) and the key to encrypt RGK using KDF, K = KDF(ECDH(r.PK.SP.ENC))
- Encrypt RGK (AES[RGK])
- Sign the concatenation of R|| AES[K,RGK] || SD AID || SE SN with SK.CASD.ECDSA
R || [Encrypted RGK] ||SD AID || SE SN || Signature
- Verify CERT.SEMS.AUT and retrieve PK.SEMS.AUT
- Verify the signature using PK.SEMS.AUT
- Decrypt RGK using the AES key K computed from the ShS using SK.SP.ENC.S4 and R, retrieved from the SEMS_PULL_KEY response
- Derive the SCP key set from the RGK

All keys mentioned in Figure 4-8 are defined in section 4.5.1.

The SEMS_PULL_KEY command, defined in section 7.11, triggers the generation of SCP keys and their assignment to the newly created Security Domain. The key generated on-board shall be encrypted with the PK.SP.ENC.S4 key and signed with the SK.CASD.AUT key before being returned as part of the response to the SEMS_PULL_KEY command.

On receipt of the SEMS response script, the Off-Card Entity shall perform the following operations:

- Verify the signature of the CERT.CASD.ECDSA certificate (defined in Table 4-9) with the PK.CA-KLCC.ECDSA pubic key, extracted from CERT.CA-KLCC.ECDSA, and extract the public key PK.CASD.ECDSA from the CERT.CASD.ECDSA certificate. The CERT.CA-KLCC.ECDSA certificate shall have been received off-line by the OCE.
- Verify the signature of the SEMS_PULL_KEY response with PK.CASD.ECDSA.
- Retrieve the SD AID and SE Serial Number (SN).
    - Verify that the SE Serial Number matches the Serial Number stored in the CERT.CASD.ECDSA certificate defined in section 4.5.3.
    - Verify that the SD AID matches the SD AID set in the SEMS script.

- Decrypt RGK using the private key SK.SP.ENC and the public key R of the semi-static ECC key pair retrieved from the SEMS_PULL_KEY response as defined in section 7.11.4.
- Derive the SCP keys from RGK as described in [Amd A] section 3.5.2.

  Note that the SEMS Application shall not keep RGK in memory once the SEMS_PULL_KEY has been processed.

4.8.2.2 Variant based on GlobalPlatform Amendment A Scenario #1

SEMS is personalized with a CERT.CASD.AUT certificate, defined in section 4.5.4, containing the PK.CASD.AUT public key, and a private key SK.CASD.AUT. The OCE, e.g. Service Provider, has generated a key pair SP.ENC.S1. The SEMS_PULL_KEY_GP command, defined in section 7.12, triggering the on-board generated key, contains the public key PK.SP.ENC.S1. In response to the on-board generated key command (i.e. SEMS_PULL_KEY_GP, defined in section 7.12), SEMS generates a Randomly Generated Key (RGK). The three Security Domain SCP keys shall be derived from the RGK key following the derivation scheme described in [Amd A] section 3.5.2.

The RGK in the response of SEMS_PULL_KEY_GP shall be encrypted using the PK.SP.ENC.S1 key and signed with the SK.CASD.AUT private key as described in [Amd A] section 3.5.2.

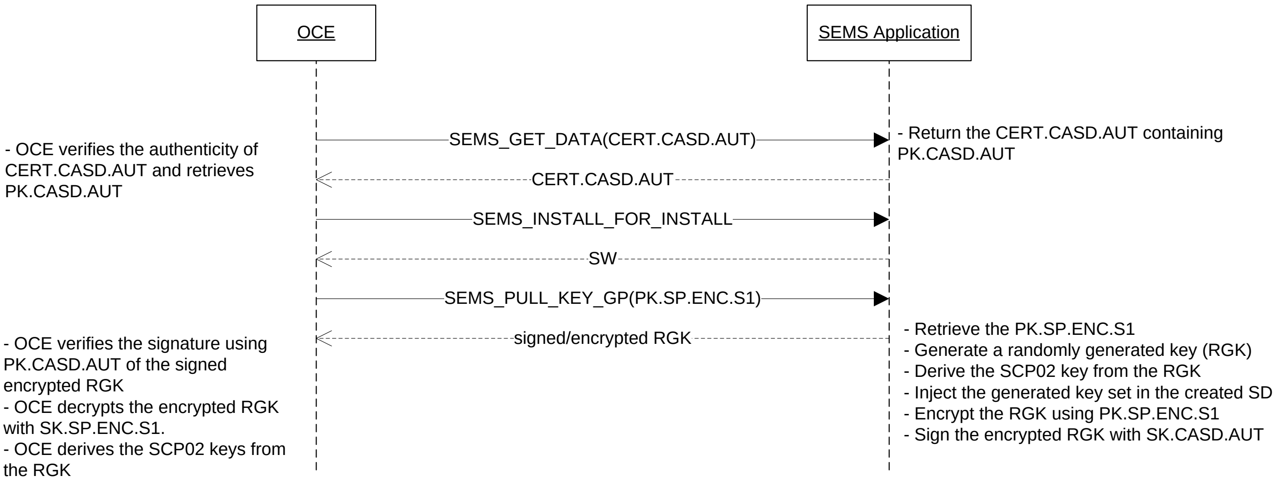
For an RSA key with a length of 1024 bits:

- The signature algorithm shall be performed as defined in [GPCS] section B.3.1.1.
- The encryption algorithm shall be performed as defined in [GPCS] section B.3.1.2.

For an RSA key length longer than 1024 bits:

- The signature algorithm shall be performed as defined in [GPCS] section B.3.2.1.
- The encryption algorithm shall be performed as defined in [GPCS] section B.3.2.2.

Figure 4-9: Security Domain Creation and Pull Key Model based on Amendment A Scenario #1

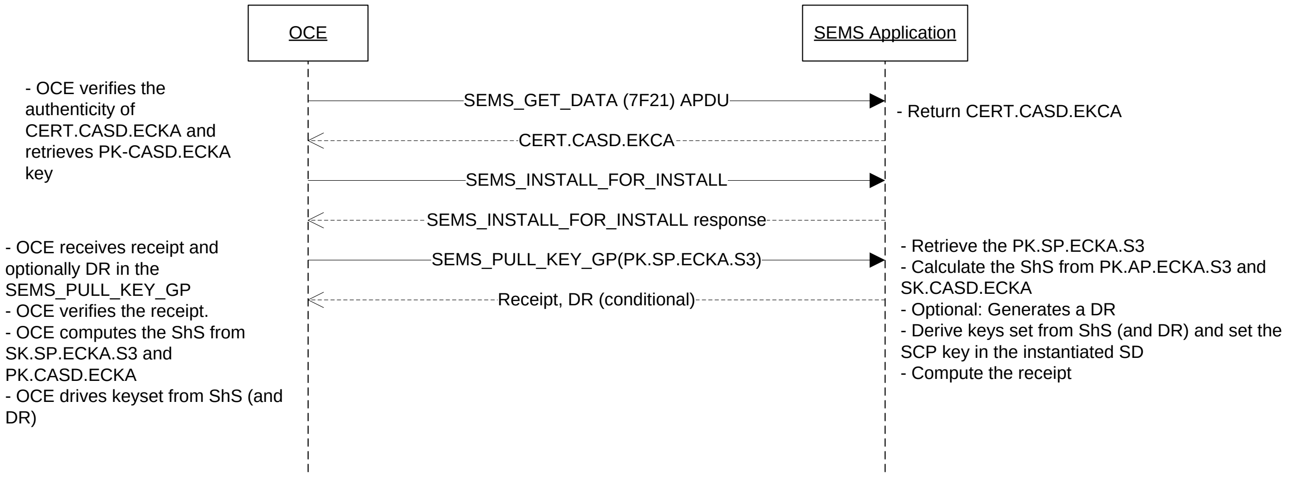

OCE
SEMS Application
SEMS_GET_DATA(CERT.CASD.AUT)
- Return the CERT.CASD.AUT containing PK.CASD.AUT
- OCE verifies the authenticity of CERT.CASD.AUT and retrieves PK.CASD.AUT
CERT.CASD.AUT
SEMS_INSTALL_FOR_INSTALL
SW
SEMS_PULL_KEY_GP(PK.SP.ENC.S1)
signed/encrypted RGK
- OCE verifies the signature using PK.CASD.AUT of the signed encrypted RGK
- OCE decrypts the encrypted RGK with SK.SP.ENC.S1.
- OCE derives the SCP02 keys from the RGK
- Retrieve the PK.SP.ENC.S1
- Generate a randomly generated key (RGK)
- Derive the SCP02 key from the RGK
- Inject the generated key set in the created SD
- Encrypt the RGK using PK.SP.ENC.S1
- Sign the encrypted RGK with SK.CASD.AUT

4.8.2.3 SEMS Implementation of GlobalPlatform Amendment A Scenario #3

The SEMS Application may implement the GlobalPlatform confidential key setting based on the key agreement model as described in [Amd A] with SEMS_GET_DATA, defined in section 7.14, and SEMS_PULL_KEY, defined in section 7.12.

SEMS is personalized with a CERT.CASD.ECKA certificate, defined in section 4.5.5, containing the PK.CASD.ECKA public key, and a private key SK.CASD.ECKA. The OCE (e.g. Service Provider) has generated a key pair SP.ECKA.S3. The SEMS_PULL_KEY_GP command, defined in section 7.12, triggering the on-board generated key, contains the public key PK.SP.ECKA.S3. The receipt is computed as described in [Amd A]. The SD targeted by SEMS_PULL_KEY_GP is personalized with an SCP key set as defined in [Amd A].

Figure 4-10: Security Domain Creation and Pull Key Model based on a Key Agreement


OCE
SEMS Application
- OCE verifies the authenticity of CERT.CASD.ECKA and retrieves PK-CASD.ECKA key
SEMS_GET_DATA (7F21) APDU
- Return CERT.CASD.EKCA
CERT.CASD.EKCA
SEMS_INSTALL_FOR_INSTALL
SEMS_INSTALL_FOR_INSTALL response
- OCE receives receipt and optionally DR in the SEMS_PULL_KEY_GP
- OCE verifies the receipt.
- OCE computes the ShS from SK.SP.ECKA.S3 and PK.CASD.ECKA
- OCE drives keyset from ShS (and DR)
SEMS_PULL_KEY_GP(PK.SP.ECKA.S3)
Receipt, DR (conditional)
- Retrieve the PK.SP.ECKA.S3
- Calculate the ShS from PK.AP.ECKA.S3 and SK.CASD.ECKA
- Optional: Generates a DR
- Derive keys set from ShS (and DR) and set the SCP key in the instantiated SD
- Compute the receipt

4.9 Loading and Update of an ELF

The SEMS Application loads an ELF via the SEMS_INSTALL_FOR_LOAD command followed by one or more SEMS_LOAD commands depending on the size of the ELF. The SEMS_INSTALL_FOR_LOAD command is used to initiate the ELF loading operation while the SEMS_LOAD command(s) contain the actual ELF.

Two cases shall be considered for an ELF being loaded through a SEMS script:

- If the ELF is not present on the SE, then it shall be loaded and associated with the SD given as a parameter in the SEMS_INSTALL_FOR_LOAD command. If the SD AID field is empty, the loaded ELF is associated with the selected SD.
- If the ELF is already present on the SE, then [Amd H] shall be applied to update the ELF.

If the execution of a script to load/update an ELF is interrupted because of an OS reboot or a power down of the ME, the same script should be replayed by the SEMS Device Agent.

If loading is successful, the ELF shall be internally associated with the certificate holder identifier (tag '5F20' of CERT.SP.AUT) for further control of its usage.

4.10 Application Installation and Personalization

The SEMS Application shall only personalize an Application that has been created through a SEMS script by the same SP identified through the certificate holder identifier as defined in sections 7.7 and 7.9. Of course, personalization data will be the same for all cards that are addressed by the same script.

The SEMS mechanism supports the following personalization schemes described in [GPCS]:

- The personalization of an Application through an SD where the Application is installed by issuing an INSTALL [for personalization] APDU and subsequent STORE DATA commands as described in [GPCS] section 7.3.3
- The personalization of an Application by selecting the Application and issuing Application-specific APDU as described in [GPCS] section 7.3.4

No SEMS command corresponding to the INSTALL [for personalization] or INSTALL [for registry update] has been defined. However, those commands may be conveyed through the SEMS_APDU command described in section 7.4. If data is to be returned by the SEMS_APDU, it shall be stored in clear text within the response script.

A SEMS script may contain commands to install and personalize a new application instance. Personalization may be done either through the SPSD or directly with the Application. Installation and personalization may require the atomicity property; i.e. after the instantiation of an Application, all personalization commands shall be executed to consider that the personalization performed via SEMS script processing is complete. If the personalization sequence is interrupted because of a power loss or reset of the SE or a failed SEMS command, the SEMS Application shall delete the Application instance referenced in SEMS_BEGIN_PERSO on receipt of the first verification of the Authentication Frame (see section 5.1.4).

The SEMS_BEGIN_PERSO and SEMS_END_PERSO commands are used as markers to identify a sequence of SEMS commands that are to be executed atomically. Such a sequence of commands will be called a PERSO block.

4.10.1 SEMS_BEGIN_PERSO

When the SEMS Application encounters a SEMS_BEGIN_PERSO command, it does the following:

- If a SEMS_BEGIN_PERSO command has already been received (but no matching SEMS_END_PERSO command), the SEMS Application simply aborts processing (the SEMS script is malformed).
- If the Current Personalized Application AID is empty, the SEMS Application sets its value to the Application AID passed as an argument to the SEMS_BEGIN_PERSO command.

The functional behavior of the SEMS_BEGIN_PERSO command is described in section 7.5.

4.10.2 SEMS_END_PERSO

The SEMS Application clears the value of the Current Personalized Application AID.

4.11 Deletion of Applications

The SEMS Application shall enforce the following rules.

- An Application including an SSD shall only be deleted if the following requirements are met:
  - The SEMS_DELETE is contained in a SEMS script provided by the same SP Device Application that has provided the script which has created the instance of the Application to be deleted. The SP Device Application is identified by an individual reference value, e.g. the digest value of the SP Device Application package name, forwarded by the SEMS Device Agent as described in Step 3 of section A.1 via a STORE DATA command (see section 6.4).
  - The SEMS_DELETE is contained in a SEMS script signed by the same SEMS SP certificate holder that signed the script which created the instance of the Application to be deleted (see sections 7.7 and 7.9).
- A Security Domain shall only be deleted:
  - If the SD has no associated entities: neither SDs nor Applications nor ELFs.
- An ELF shall only be deleted:
  - If there are no Applications instantiated from the ELF to be deleted.
  - If the SEMS_DELETE is contained in a SEMS script signed by the same SEMS SP certificate holder that signed the script used to load the ELF to be deleted.

4.12 Script Bound to an SE

The purpose of the SEMS_BINDING_SE command as defined in section 7.17 is to restrict the execution of a SEMS script to the specific SE referenced in the command. In this case, the SEMS Application executes all SEMS commands following SEMS_BINDING_SE only if the Serial Number given as input to SEMS_BINDING_SE matches the SE Serial Number on the target ME. (The SE serial number is defined in section 7.17.2.)

4.13 Rotation of SEMS Application Keys

When the SE is produced, the SEMS Application Manager is responsible for the personalization of the SEMS Application with ECC keys; i.e. SK.CA-SEMS.ENC and PK.CA-SEMS.AUT (see section 4.5.1). However, when the SE is produced, the final SEMS Application Manager (and implicitly the OCE that is the final owner of the SEMS Application) may not be known or may change after the device has been released to the consumer. Such a change of SEMS Application Provider may require the so-called rotation of the ECC keys stored in the SEMS Application, meaning a replacement of these secret credentials. Those SEMS Application keys may be rotated either during device manufacturing or after the device is already released to the consumer. The SEMS_KEY_ROTATION command, as described in section 7.18, is the SEMS command to replace these SEMS Application keys.

Once a device has been released to the end user, the SEMS Application Provider may still be changed. It is possible to assign ownership of the SEMS Application (identified in CERT.SP.AUT tag '5F20' as described in section 4.5.1) to the new SEMS Application Provider. In this case, the previous SEMS Application Provider shall generate a script containing its certificate to authorize the SEMS_KEY_ROTATION command. When a SEMS Application is installed that has keys that will need to be replaced, it is recommended that the script be bound to the SE as discussed in section 4.12.

4.14 Teardown Management

If the execution of a SEMS script is interrupted because of a reboot of the ME or an unexpected communication error or battery off, the SEMS Device Agent might re-execute the script from scratch upon the start of the SE.

The SEMS Application shall keep track of the script execution status. The SEMS Device Agent may retrieve the script execution status via the GET DATA command (see section 6.1). The script execution status is set on receipt of the first PROCESS SCRIPT COMMAND APDU and shall be reset in either of the following conditions:

- The PROCESS SCRIPT COMMAND APDU returns an SW other than '9000' or '63XX'.
- The PROCESS SCRIPT COMMAND [P1 = Last Block] completes.

The SEMS script status shall be stored in Non-Volatile Memory on the SE for consistent behavior in the event of a teardown of the SE.

4.15 SEMS Updater

The only purpose of the SEMS Updater is to allow the update of a SEMS Application with a SEMS script. This section applies only if the SEMS Application is implemented as a Java Card applet. In order to restore the credentials required for operating the secured SEMS scripts, the SEMS Application has to implement the `OnUpgradeListener` interface defined in [Amd H] allowing its data to persist across the ELF Upgrade. The Update process itself is described in [Amd H]. The SEMS Updater is able to apply the APDU forwarding via virtual I/O interface as described in section 4.7.1.1.

The SEMS Application identifies during instantiation of the SEMS Updater (via the SEMS CA privilege defined in the CERT.SP.AUT as described in section 4.5.1) that this special Application has been created. For the purpose of supporting the SEMS Application update process, a special SW is returned to the SEMS Device Agent Application by the PROCESS SCRIPT COMMAND (see Table 6-8), indicating that the SEMS Updater shall be selected.

The only SD the SEMS Updater is allowed to select via the virtual I/O interface is the Security Domain associated with the SEMS Application. The OPEN shall reject any other attempt by the SEMS Updater to select a Target Application via the virtual I/O interface.

The upload and install of the SEMS Updater shall be done via a SEMS script with a CERT.SP.AUT in which tag '4C' bit b2 is set. It is the responsibility of the SEMS Application Provider to provision a sufficiently capable SEMS Updater. Initially, the only SD that the SEMS Application has a trust relationship with (i.e. the only SD that the SEMS Application can send commands to) is the SD associated with the SEMS Application. This is probably the ISD, but it could be a specific SEMS Application Provider AMSD. The same initial trust relationship shall exist for the SEMS Updater applet that shall be associated with the same SD as the SEMS Application.

Note: The SEMS_INSTALL_FOR_LOAD of the SEMS Updater is restricted by the SEMS Application, which checks the content of tag '42' and tag '45' in CERT.SP.AUT according to section 4.5.1.

The AID of the new ELF is implicitly known to the SEMS Updater loaded. On successful selection the SEMS Updater:

- The SEMS Updater selects the associated SD on the virtual I/O interface.
- The SEMS Updater queries the associated SD for the existence of a new SEMS Application ELF with a GET STATUS command with P1 = '20' (see [CIC] section 5.5).
  Note: SEMS Application AIDs follow the scheme shown in Table A-1.
- If the search for the new SEMS Application ELF is successful, the SEMS Updater prepares the MANAGE ELF UPGRADE [start/resume] command CAPDU as defined in [Amd H] section 4.1 to initiate the ELF Upgrade Process. It is recommended that Upgrade Session Option coding be applied (sub-tag '80' in tag 'A1' should set all Upgrade Options according to [Amd H] Table 4-4).
- The SEMS Updater performs the Upgrade Session and, on successful completion of the ELF Upgrade process, returns the MANAGE ELF UPGRADE Response Data Field and a status word of '9000' notifying the SEMS Device Agent about the successful completion of the SEMS Application update and that the new SEMS Application may now be selected.

5 SEMS Script Format

5.1 SEMS Command Script Format

5.1.1 Plaintext SEMS Command Script Format

A plaintext SEMS script is the concatenation of plaintext SEMS commands coded as TLVs, as described in section 7. A plaintext SEMS script shall be secured as described in section 5.1.2 before it can be deployed.

The maximum length of any (single) plaintext SEMS command (described in section 7) is 192 bytes.

Of the 255 bytes (maximum length of the APDU command data field), 32 bytes are occupied by the SHA256. Of the remaining 223 bytes, a maximum of 208 bytes can be utilized to hold the ciphertext due to block cipher restrictions. Of the 208 bytes, 16 bytes are occupied by the encryption key of the next ciphertext. This leaves a maximum of 192 bytes to hold the plaintext SEMS command. See also sections 4.6.1 and 5.1.5.

5.1.2 Secured SEMS Command Script Format

The secured script format is made of:

- One or more couples of Certificate and Authentication Frames
    - A Certificate Frame shall be followed by an Authentication Frame (section 5.1.4) signed with the private key associated with the CERT.SP.AUT (section 4.5.1).
- The set of secured SEMS commands; i.e. the encrypted and signed SEMS commands as defined in section 4.6.1

The TLVs are BER encoded.

5.1.3 Certificate Frame

Each Certificate Frame contains the certificate (chain of) CERT.SP.AUT as defined in section 4.5.2.

5.1.4 Authentication Frame

The structure of the Authentication Frame is described in Table 5-1.

Table 5-1: Authentication Frame

<table>
<tr><th>Tag</th><th>Length</th><th colspan="3">Description</th><th>Presence</th></tr>
<tr><td rowspan="3">'60'</td><td rowspan="3">186..188
'81BA'… '81BC'</td><td colspan="3">Authentication Frame</td><td></td></tr>
<tr><th>Tag</th><th>Length</th><th>Description</th><th>Mandatory</th></tr>
<tr><td>'41'</td><td>183..185</td><td>ePK.SP.ENC (65 bytes)
|| E[K, K_1] (16 bytes)
|| H_1 (32 bytes)
|| Signature (70 .. 72 bytes)</td><td>Mandatory</td></tr>
</table>

Length fields for the tags are coded according to ASN.1 BER-TLV (see [ITU-T Rec X690]).

The public key ePK.SP.ENC uses uncompressed encoding as defined in [TR-03111] section 3.2.1; i.e. '04' || X || Y.

E[K,K_1] is the encryption of K_1 with the key K using the AES cypher algorithm.

H_1 is SHA-256 applied to the first encrypted command, as described in section 5.1.5, i.e. E[K_1, K_2 || SEMS command_2] || H_2.

The signature is computed on ePK.SP.ENC || E[K, K_1] || H_1 (32 bytes) using the ECDSA algorithm and must be verified with the PK.SP.AUT extracted from the CERT.SP.AUT certificate.

The concatenation of ePK.SP.ENC with E[K, K_1], the H_1 hash value and the Signature (coded as described in Table 5-2 within sub-tag '30') equals the required format to apply a secured SEMS script as described in section 4.6.1 and shown in Figure 4-2.

The Signature (sub-tag '30'), with the content in plain format specified in [TR-03111], is structured as follows:

Table 5-2: Signature Format

<table>
<tr><th>Tag</th><th>Length</th><th colspan="3">Description</th><th>Presence</th></tr>
<tr><td rowspan="4">'30'</td><td rowspan="4">68..70</td><td colspan="3">Signature</td><td></td></tr>
<tr><th>Tag</th><th>Length</th><th>Description</th><th>Mandatory</th></tr>
<tr><td>'02'</td><td>'32' or '33'</td><td>r component</td><td>Mandatory</td></tr>
<tr><td>'02'</td><td>'32' or '33'</td><td>s component</td><td>Mandatory</td></tr>
</table>

5.1.5 Secured SEMS Command Format

The secured SEMS command format is described in Table 5-3.

Table 5-3: Secured SEMS Command Format

Tag	Length	Value Description
'40'	Var., not to exceed 261	One of the following: • 80 A0 P1 00 LC (E[K_n, K_{n+1} || SEMS command_n] || H_{n+1}) 00 (non-last SEMS command, P1='0x') • 80 A0 P1 00 LC (E[K_n, K_{n+1} || SEMS command_n] || 32-byte random array) 00 (last SEMS command, P1='8x') See section 6.2.4 for details on the coding of P1.

With H_n = SHA-256(E[K_n, K_{n+1} || SEMS command_n] || H_{n+1}).

The encryption is described in section 4.6.1.

5.2 SEMS Response Script Format

The SEMS Response Script is a sequence of SEMS command responses formatted as defined in Table 5-4.

Table 5-4: SEMS Command Response Format

Tag	Length	Value Description			Presence
'61'	xx	SEMS Command Response			Mandatory
		Tag	**Length**	**Value Description**	
		'43'	1-2	Tag that has been executed by the SEMS Application. It shall be one of the following tag values: • '7F21' for a CERT.SP.AUT • '60' for an Authentication Frame • '40' for a secured SEMS command	Mandatory
		'44'	Var.	One of the following: • SW • SEMS command response || SW	Mandatory

See section 6.2.6 for more details.

6 SEMS Application APDU

6.1 GET DATA Command APDU

6.1.1 Definition and Scope

The GET DATA command is used to retrieve data from the SEMS Application.

6.1.2 Command Message

The GET DATA command is coded according to the following table:

Table 6-1: GET DATA Command Message

Code	Value	Meaning
CLA	'80' – '83' or 'C0' – 'CF'	See [GPCS] section 11.1.4.
INS	'CA'	GET DATA
P1	'00'	Reference control parameter P1
P2	'xx'	Reference control parameter P2; see Table 6-2.
Lc	-	Not present
Data	-	The Data field shall be empty.
Le	00	Length expected

6.1.3 Reference Control Parameter P2

P2 defines the tag of the data object to be read.

Table 6-2: GET DATA Command Reference Control Parameter P2

P2	Meaning
'43'	SK.CA-SEMS.ENC Identifier
'46'	SEMS execution script status defined in Table 6-3
'42'	CA-SEMS Identifier
'45'	CA-SEMS signature verification key identifier

6.1.4 Data Field Sent in the Command Message

The data field of the GET DATA command is empty.

6.1.5 Response Message

6.1.5.1 Data Field Returned in the Response Message

The GET DATA response data field shall contain a TLV coded data object referred to in P2 of the command message.

Table 6-3: Data Field Returned in GET DATA Response Message

Tag	Length	Data Field
'43'	24	SK.CA-SEMS.ENC Identifier
'46'	1	'00': The script has been completely executed. '01': Script execution was interrupted because of teardown. (See section 4.14.)
'42'	16	CA-SEMS Identifier
'45'	8	CA-SEMS signature verification key identifier

6.1.5.2 Status Word Returned in the Response Message

Successful execution of the command shall be indicated by SW '9000'.

This command may return either a status word identifying a general error condition as listed in [GPCS] section 11.1.3 or the following status word.

Table 6-4: Status Word in GET DATA Response Message

SW1	SW2	Meaning
'6A'	'88'	Reference data not found

6.2 PROCESS SCRIPT COMMAND APDU Command

6.2.1 Definition and Scope

The PROCESS SCRIPT COMMAND APDU command is used to process a CERT.SP.AUT, defined in Table 4-2, an Authentication Frame, defined in Table 5-1, or a secured SEMS command, defined in Table 5-2.

6.2.2 Preconditions

The SEMS Application state shall be PERSONALIZED, as defined in Table A-2.

6.2.3 Command Message

The PROCESS SCRIPT COMMAND APDU command is coded according to the following table:

Table 6-5: PROCESS SCRIPT COMMAND APDU Command Message

Code	Value	Meaning
CLA	'80' – '83' or 'C0' – 'CF'	See [GPCS] section 11.1.4.
INS	'A0'	PROCESS SCRIPT COMMAND
P1	'xx'	Reference control parameter P1
P2	'00'	Reference control parameter P2
Lc	'xx'	Length of data field
Data	'xx..xx'	Data field
Le	-	Not present

6.2.4 Reference Control Parameter P1

Reference Control Parameter P1 shall be:

Table 6-6: PROCESS SCRIPT COMMAND APDU Reference Control Parameter P1

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
-	-	-	-	-	-	-	0	No remaining bytes (command fully received)
-	-	-	-	-	-	-	1	More bytes available (same command / chaining)
0	-	-	-	-	-	-	-	More commands
1	-	-	-	-	-	-	-	Last command

6.2.5 Data Field Sent in the Command Message

The data field contains either a CERT.SP.AUT, an Authentication Frame, or a secured SEMS command.

6.2.5.1 Certificate

If the data field contains an SP certificate:

- The SEMS Application shall reset the authentication state (e.g. clear session keys and chaining data).
- SW '6985' shall be returned if the previously processed APDU is neither a STORE DATA command containing tag '4F' nor a PROCESS SCRIPT COMMAND APDU with the parent CERT.SP.AUT of a certificate chain.
- SW '6A80' shall be returned if:
    - A mandatory tag of CERT.SP.AUT is missing.
    - The tag order of CERT.SP.AUT is incorrect. The order of tags is shown in Table 4-2.
    - The length of a TLV including tag '7F21' is incorrect.
    - The format of PK.SP.AUT is incorrect; e.g. leading '04' is missing.
    - The data field of tag '95' is other than '82'.
    - Tag '42' data field of CERT.SP.AUT does not match the CA-SEMS identifier stored in the SEMS Application.
    - Tag '45' data field of CERT.SP.AUT does not match the CA-SEMS signature verification key identifier stored in the SEMS Application.

The SEMS Application shall verify the signature of the input certificate with the public key PK.CA-SEMS.AUT or PK.SP.AUT (in case of certificate chains) as described in section 4.6.2. If the signature is incorrect, SW '6982' shall be returned. If the certificate signature is successfully verified, the SEMS Application shall retrieve operational required data; i.e. the SP certificate public key PK.SP.AUT (tag '86') and tag '53' content to retrieve CCM privileges.

6.2.5.2 Authentication Frame

If the data field contains an Authentication Frame:

- The previously processed command shall be a PROCESS SCRIPT COMMAND APDU command whose data field contains a CERT.SP.AUT. Otherwise, SW '6985' shall be returned.
- If the SEMS state is PERSONALIZED and if the previously processed APDU is not a successfully processed PROCESS SCRIPT COMMAND whose data field contains a CERT.SP.AUT certificate, then SW '6985' shall be returned.
- If the length of the Authentication Frame is incorrect, then SW '6700' shall be returned.
- If the format of the Authentication Frame is incorrect (e.g. wrong format of signature, format of the public key due to missing leading '04'), then SW '6985' shall be returned.
- The SEMS Application shall verify the signature of the Authentication Frame, with the PK.SP.AUT public key (tag '86') of the CERT.SP.AUT.
  - If the signature is incorrect, the Authentication Frame is rejected and SW '6982' shall be returned.
  - If the signature is successfully verified:
    - The SEMS Application shall set the authentication state. (See section 4.7.1.2)
    - The key K used to decrypt the first SEMS command is generated as described in section 4.6.2, and the SHA-256 (E[K, K_1 || first SEMS command] || H_2)), as defined in section 5.1.5, is retrieved.

6.2.5.3 Secured SEMS Command

The secured SEMS command is decrypted after a successful integrity verification. Then the key to decrypt the next command and the SHA-256 to verify the integrity of the next command are retrieved. Finally, the SEMS command is executed.

The most recent successfully executed APDU command shall be either a PROCESS SCRIPT COMMAND APDU command with Authentication Frame or a secured SEMS command.

6.2.6 Response Message

6.2.6.1 Data Field Returned in the Response Message

Table 6-7: Data Field Returned in PROCESS SCRIPT COMMAND APDU Response Message

<table>
<tr><th>Tag</th><th>Length</th><th colspan="3">Value Description</th></tr>
<tr><td rowspan="4">'61'</td><td rowspan="4">8</td><td colspan="3">Response in event of CERT.SP.AUT processing</td></tr>
<tr><th>Tag</th><th>Length</th><th>Value Description</th></tr>
<tr><td>'43'</td><td>2</td><td>'7F21'</td></tr>
<tr><td>'44'</td><td>2</td><td>SW as defined in Table 6-8</td></tr>
<tr><td rowspan="4">'61'</td><td rowspan="4">7</td><td colspan="3">Response in event of Authentication Frame processing</td></tr>
<tr><th>Tag</th><th>Length</th><th>Value Description</th></tr>
<tr><td>'43'</td><td>1</td><td>'60'</td></tr>
<tr><td>'44'</td><td>2</td><td>SW as defined in Table 6-8</td></tr>
<tr><td rowspan="4">'61'</td><td rowspan="4">Var.</td><td colspan="3">Response in event of SEMS command processing</td></tr>
<tr><th>Tag</th><th>Length</th><th>Value Description</th></tr>
<tr><td>'43'</td><td>1</td><td>'40'</td></tr>
<tr><td>'44'</td><td>Var.</td><td>SEMS command response || SW as defined in section 7</td></tr>
</table>

The PROCESS SCRIPT COMMAND APDU response data field returned in the event of SEMS command processing shall contain the (unchained) RAPDU which the Target Application transmitted to the SEMS Application via the virtual I/O interface (as defined in section 4.7.1.2). If the Target Application responds to an unchained CAPDU by using response chaining (as described in [GPCS] section 11.1.5.2), the SEMS Application shall include in the response to the PROCESS SCRIPT COMMAND APDU only the first RAPDU, followed by SW '6310'.

6.2.6.2 Status Word Returned in the Response Message

Successful execution of the command shall be indicated by SW '9000'.

When this command processes a SEMS command, it may return a status word as defined in section 7.

When this command processes CERT.SP.AUT or an Authentication Frame, it may return either a status word identifying a general error condition as listed in [GPCS] section 11.1.3 or one of the following status words.

Table 6-8: Status Word in PROCESS SCRIPT COMMAND APDU Response Message

SW1	SW2	Meaning
'63'	'20'	Successful execution of the command. SEMS Updater shall be selected.
'69'	'82'	Security Status not satisfied: Authentication/integrity verification failed.
'69'	'85'	Condition of use not satisfied.
'69'	'87'	Expected secure messaging data object missing.
'6F'	'12'	SEMS command not supported.

6.3 SELECT Command

6.3.1 Definition and Scope

The SELECT command is identical to the SELECT command described in [GPCS] section 11.9 except for the response message. If a SEMS Application does not have the Security Domain privilege, the SELECT command shall respond with a Data Field as defined in section 6.3.5.1. If the SEMS Application has the Security Domain privilege, the SELECT command shall respond as defined in [GPCS] section 11.9.3.1.

6.3.2 Command Message

The SELECT command is coded according to the following table:

Table 6-9: SELECT Command Message

Code	Value	Meaning
CLA	'00' - '03' or '40' - '4F'	See [GPCS] section 11.1.4.
INS	'A4'	SELECT
P1	'04'	Select by Name
P2	'xx'	Reference control parameter P2
Lc	'xx'	Length of AID
Data	'xx..xx'	AID of the SEMS Application or of the SEMS Updater
Le	-	Not present

6.3.3 Reference Control Parameter P2

Reference Control Parameter P2 shall be coded according to the following table:

Table 6-10: SELECT Command Reference Control Parameter P2

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
				0	0			Return FCI
						0	0	First or only occurrence
						1	0	Next occurrence

6.3.4 Data Field Sent in Command Message

The data field of the SELECT command shall contain the AID of the SEMS Application as defined in section A.1.

6.3.5 Response Message

6.3.5.1 Data Field Returned in the Response Message

The SELECT response data field shall contain an FCI coded according to the following table:

Table 6-11: Data Field Returned in SELECT Response Message

<table>
<tr><th>Tag</th><th>Length</th><th colspan="5">Description</th><th>Presence</th></tr>
<tr><td rowspan="8">'6F'</td><td rowspan="8">50</td><td colspan="5">File Control Information (FCI Template)</td><td>Mandatory</td></tr>
<tr><td><b>Tag</b></td><td><b>Length</b></td><td colspan="3"><b>Description</b></td><td><b>Mandatory</b></td></tr>
<tr><td>'84'</td><td>14</td><td colspan="3">SEMS Application AID or SEMS Updater AID</td><td>Mandatory</td></tr>
<tr><td>'9F08'</td><td>2</td><td colspan="3">Version of the SEMS Application or SEMS Updater</td><td>Mandatory</td></tr>
<tr><td rowspan="4">'65'</td><td rowspan="4">28</td><td colspan="3">Identifier of the SEMS Application Key Set Identifier</td><td>Mandatory</td></tr>
<tr><td><b>Tag</b></td><td><b>Length</b></td><td><b>Description</b></td><td></td></tr>
<tr><td>'42'</td><td>16</td><td>CA-SEMS Identifier</td><td>Mandatory</td></tr>
<tr><td>'45'</td><td>8</td><td>CA-SEMS signature verification key Identifier</td><td>Mandatory</td></tr>
</table>

6.3.6 Status Word Returned in the Response Message

Successful execution of the command shall be indicated by SW '9000'.

This command may return either a status word identifying a general error condition as listed in [GPCS] section 11.1.3 or one of the following status words.

Table 6-12: Status Word in SELECT Response Message

SW1	SW2	Meaning
'67'	'00'	Wrong LC length
'69' '6A'	'99' '82'	SEMS Application or SEMS Updater is not installed or rejects its selection because its internal Life Cycle state is rejecting the Application instance selection. The actual SW is implementation-specific and is out of scope of this specification.

6.4 STORE DATA Command

6.4.1 Definition and Scope

The STORE DATA command is used by the SEMS Device Agent to provide the SEMS Application with an identifier of the caller (SP Device Application), e.g. the digest value of the SP Device Application package name.

6.4.2 Preconditions

The SEMS Application Life Cycle state shall be PERSONALIZED.

6.4.3 Command Message

The STORE DATA command is coded according to the following table:

Table 6-13: STORE DATA Command Message

Code	Value	Meaning
CLA	'80' – '83' or 'C0' – 'CF'	See [GPCS] section 11.1.4.
INS	'E2'	STORE DATA
P1	'00'	Reference control parameter P1
P2	'00'	Reference control parameter P2
Lc	'xx'	Length of data field
Data	'xx..xx'	Data field

6.4.4 Data Field Sent in the Command Message

The data field of the STORE DATA command shall contain the TLV described in the following table:

Table 6-14: Data Field Sent in STORE DATA Command Message

Tag	Length	Description	Presence
'4F'	Var.	SP Device Application identifier, e.g. the digest value of the SP Device Application package name of the Device Application that calls the SEMS Device Agent	Mandatory

The authentication session is not closed if the processing of STORE DATA fails. (Section 4.6.2.1 describes conditions under which the authentication session is closed.)

6.4.5 Response Message

6.4.5.1 Data Field Returned in the Response Message

The STORE DATA response data field shall be empty.

6.4.5.2 Status Word Returned in the Response Message

Successful execution of the command shall be indicated by SW '9000'.

This command may return either a status word identifying a general error condition as listed in [GPCS] section 11.1.3 or one of the following status words.

Table 6-15: Status Word in STORE DATA Response Message

SW1	SW2	Meaning
'6A'	'80'	Incorrect parameters in the command data field The provided tag is unknown; i.e. other than '4F'. The tag length is wrong.
'6A'	'86'	Wrong P1-P2

7 SEMS Commands

7.1 General Status Words

Table 7-1: General Status Words for SEMS Commands

SW1	SW2	Meaning
'63'	'10'	Successful execution of the SEMS command, and the data response field contains an APDU to be forwarded to the SEMS Device Agent
'69'	'85'	Condition of use not satisfied: • The SEMS command is executed in a wrong state.
'6A'	'80'	Wrong SEMS command length
'6A'	'88'	Wrong SEMS command data

The SEMS Device Agent may store the SW and the response data field in a response file. Depending on the returned SW:

- SW = '9000' or '6310' means that the SEMS command has been successfully executed; the SEMS Device Agent shall extract the next SEMS command from the SEMS script and forward it to the SEMS Application.
- SW = error means that the SEMS command failed and the SEMS Device Agent shall stop the execution of the SEMS script.

7.2 SEMS Command Format

SEMS commands are coded as TLV structures. TLV lengths are BER-encoded. Tags are listed in Table 7-2.

Table 7-2: Tags of SEMS Commands

Tag	SEMS Command	Description
'49'	SEMS_SELECT	Selects an Application referenced by its AID.
'4A'	SEMS_APDU	Sends any APDU to a selected Application.
'4E'	SEMS_BEGIN_PERSO	Marks the start of a sequence of personalization commands.
'4F'	SEMS_END_PERSO	Marks the end of a sequence of personalization commands.
'51'	SEMS_INSTALL_FOR_LOAD	Initiates the loading or update of an ELF.
'52'	SEMS_LOAD	Loads an ELF.
'53'	SEMS_INSTALL_FOR_INSTALL	Installs a new Application instance.
'54'	SEMS_DELETE	Deletes an ELF or an Application.
'56'	SEMS_PULL_KEY	Triggers generation of the on-board SCP key set (as described in section 4.8.2.1) and loading the key set into the SD selected by the SEMS Application.
'57'	SEMS_GET_DATA	Retrieves a data object from the SEMS Application (i.e. CERT.CASD.AUT).
'58'	SEMS_BINDING_SE	Binds the execution of a SEMS command to a specific SE referenced by its Serial Number.
'59'	SEMS_KEY_ROTATION	Triggers the rotation of the SEMS ECDSA keys (i.e. the keys used to verify a SEMS signature and to decrypt SEMS commands).
'5B'	SEMS_PUT_KEY	Loads an SCP key set into an SD created with the SEMS script and whose state is SELECTABLE.
'5D'	SEMS_PULL_KEY_GP	Triggers generation of the on-board SCP key set, either based on a variant of GlobalPlatform scenario #1 using PK scheme (described in section 4.8.2.2) or based on GlobalPlatform scenario #3 (according to section 4.8.2.3).
'5E'	SEMS_BEGIN_MANAGE_ELF_UPGRADE	Marks the start of an ELF Upgrade session
'5F'	SEMS_END_MANAGE_ELF_UPGRADE	Marks the end of an ELF Upgrade session.

SEMS commands are issued from the SEMS Device Agent to the SEMS Application with the PROCESS SCRIPT COMMAND APDU defined in section 6.2.

7.3 SEMS_SELECT Command (Tag '49')

7.3.1 Definition and Scope

The SEMS_SELECT command is used to select an Application instance (SD or Java Card applet) by its AID on the virtual I/O interface (as described in section 4.7.1).

7.3.2 Precondition

The Application whose AID is provided as a parameter of the SEMS_SELECT command shall be present on the SE.

The registry entry for this targeted Application instance shall have been created by the SEMS mechanism using a secured SEMS script signed by the same certificate holder as the current executed script. (The certificate holder's identifier is contained in certificate tag '5F20', as discussed in section 4.4.)

The Application whose AID is provided as a parameter shall be selectable.

7.3.3 Data Field Format

The SEMS_SELECT command is formatted as follows:

Table 7-3: SEMS_SELECT Data Field Format

Data Element	Length	Description	M/O/C
Logical Channel ID	1	RFU	M
Application Selection Options	1	Contains the P2 parameter of the SELECT command.	M
Length of Application AID	1	Contains the length of the Application AID.	M
Application AID	5..16	Contains the Application AID.	M

7.3.4 Response Message

This command may return either a general status word as listed in section 7.1 or one of the following status words.

Table 7-4: SEMS_SELECT Status Words

SW1	SW2	Meaning
'6A'	'82'	Application not found
'69'	'99'	Application not selected

7.4 SEMS_APDU Command (Tag '4A')

7.4.1 Definition and Scope

The SEMS_APDU command is used to send an APDU to the selected Application.

7.4.2 Precondition

An Application shall have been selected successfully with SEMS_SELECT.

The APDU command described in the data field (see section 7.4.3) shall not be a SELECT command.

If the selected application is a Security Domain, then

- The APDU command described in the data field shall not be one of the following GlobalPlatform APDUs [GPCS]:
  - INSTALL [for load]
  - INSTALL [for install]
  - INSTALL [for install and make selectable]
  - DELETE
- Other APDU commands shall be transferred to the SD but may further be rejected because of the current security level.
- If the APDU command described in the data field corresponds to one of the following GlobalPlatform APDUs (defined in [GPCS]) and the target AID listed in that APDU is not created by the certificate holder of the currently processed SEMS script or the Install/Get/Set Status privilege is not allowed in the tag '4D' that contains the target AID, then the SEMS APDU shall be rejected:
  - INSTALL [for personalization]
  - INSTALL [for extradition]
  - INSTALL [for registry update]
  - GET STATUS
  - SET STATUS
- If the APDU command described in the data field corresponds to the GlobalPlatform APDU MANAGE ELF UPGRADE with P1 indicating start of ELF upgrade session, then the previous command of SEMS_APDU (MANAGE ELF UPGRADE) shall be a SEMS_BEGIN_MANAGE_ELF_UPGRADE; otherwise the error SW '6985' shall be returned.

The SEMS Application shall compare the certificate holder of the currently processed SEMS script and the certificate holder which had been used for creation of the SD targeted by the APDU command described in the data field. If they are not identical, the security level shall be considered NO_SECURITY_LEVEL (see sections 4.7.1.1 and 4.7.1.2).

7.4.3 Data Field Format

The SEMS_APDU command is formatted as follows:

Table 7-5: SEMS_APDU Data Field Format

Data Element	Length	Description	M/O/C
Logical Channel ID	1	RFU	M
APDU	4..n	Contains the APDU formatted as one of: CLA INS P1 P2 CLA INS P1 P2 Le CLA INS P1 P2 Lc C-DATA CLA INS P1 P2 Lc C-DATA Le	M

7.4.4 Response Message

The response message contains the response data and the SW of the processed APDU.

This command may return either a general status word as listed in section 7.1 or the following status word.

Table 7-6: SEMS_APDU Status Words

SW1	SW2	Meaning
'69'	'85'	Condition of use not satisfied: Unauthorized APDU is given as parameter.

7.5 SEMS_BEGIN_PERSO Command (Tag '4E')

7.5.1 Definition and Scope

The SEMS_BEGIN_PERSO command is used to mark the beginning of a set of personalization APDU commands. Any SEMS commands may appear between the SEMS_BEGIN_PERSO and the SEMS_END_PERSO commands.

If the sequence of SEMS commands starting with the SEMS_BEGIN_PERSO command and ending with the SEMS_END_PERSO command is interrupted, i.e. if any SEMS command in the sequence is rejected with an error or a card tearing occurs, the Application with the same AID as the one given as a parameter in the SEMS_BEGIN_PERSO command shall be deleted upon the next PROCESS SCRIPT COMMAND APDU command containing an Authentication Frame.

If the Application with the same AID as the one given as a parameter within the SEMS_BEGIN_PERSO command is already present on the SE, the sequence of SEMS commands starting with the SEMS_BEGIN_PERSO command and ending with the SEMS_END_PERSO command shall be parsed but not executed by the SEMS Application. A status word of '9000' shall be returned for all commands.

The SEMS_BEGIN_PERSO and the SEMS_END_PERSO commands shall not be nested.

7.5.2 Data Field Format

The SEMS_BEGIN_PERSO command is formatted as follows:

Table 7-7: SEMS_BEGIN_PERSO Data Field Format

Data Element	Length	Description	M/O/C
Logical Channel ID	1	RFU	M
Length of Application AID	1		M
Application AID	5..16	Contains the AID of the application being personalized	M

7.5.3 Response Message

This command may return either a general status word as listed in section 7.1 or the following status word.

Table 7-8: SEMS_BEGIN_PERSO Status Words

SW1	SW2	Meaning
'69'	'85'	Condition of use not satisfied: Nested SEMS_BEGIN_PERSO command.

7.6 SEMS_END_PERSO Command (Tag '4F')

The SEMS_END_PERSO command is used to mark the end of a set of personalization APDU commands.

The SEMS_END_PERSO command does not contain data.

7.6.1 Data Field Format

This command has no data field.

7.6.2 Response Message

This command may return either a general status word as listed in section 7.1 or the following status word.

Table 7-9: SEMS_END_PERSO Status Words

SW1	SW2	Meaning
'69'	'85'	Condition of use not satisfied: Nested SEMS_END_PERSO command.

7.7 SEMS_INSTALL_FOR_LOAD Command (Tag '51')

7.7.1 Definition and Scope

The SEMS_INSTALL_FOR_LOAD command is used to start the loading of an ELF. It shall be followed by a sequence of SEMS_LOAD commands. Additionally, this command implicitly includes support for the ELF Upgrade mechanism specified in [Amd H].

If the SE supports [Amd H], a SEMS_INSTALL_FOR_LOAD command generates the following commands:

- MANAGE ELF UPGRADE with the ELF AID as specified in [Amd H] (including all options set), if the ELF is already present on the SE
- INSTALL [for load] with the ELF AID, Security Domain AID, Load File Data Block Hash, and Load Parameters

The current Security Level is applied to the above APDU commands.

If a Security Domain is defined in the SEMS_INSTALL_FOR_LOAD command, the ELF is loaded in the given SD. This rule applies even if the ELF is present on the SE in an SD other than the SD given in the SEMS_INSTALL_FOR_LOAD command. The OPEN shall save the SEMS SP certificate holder identifier (content of tag '5F20' of CERT.SP.AUT) in the registry for the ELF loaded.

If no Security Domain is defined in the SEMS_INSTALL_FOR_LOAD command, the ELF is loaded in the selected SD if the ELF is not present on the SE; otherwise, the new ELF is loaded in the SD where the old ELF was loaded.

7.7.2 Precondition

The loading of an ELF shall be rejected if it is not allowed by the CERT.SP.AUT certificate or if the holder of this certificate has already loaded more ELFs than allowed by CCM rights (tag '48') while not having the right to create an AMSD.

If the ELF to be loaded is already loaded on the SE and has been loaded via the SEMS mechanism, it may only be updated by the SEMS SP certificate holder that initially loaded the ELF or if update is allowed by the tag '4D' for the current certificate holder.

If the ELF is already loaded on the SE and if the ELF version of the ELF present on the SE is higher than or equal to the version specified in the SEMS_INSTALL_FOR_LOAD command, then the SEMS_INSTALL_FOR_LOAD command and the following SEMS_LOAD command shall be silently ignored.

ELFs that have not been loaded via the SEMS mechanism cannot be updated with a SEMS script.

The update of the ELF shall be rejected if any instance created from an EM of the ELF does not expose the `OnUpgradeListener` interface defined in [Amd H].

7.7.3 Data Field Format

The SEMS_INSTALL_FOR_LOAD command is formatted as follows:

Table 7-10: SEMS_INSTALL_FOR_LOAD Data Field Format

Data Element	Length	Description	M/O/C
Logical Channel ID	1	RFU	M
License byte	1	• 0 means that no tag '4B' containing the ELF AID is required in the CERT.SP.AUT to create an instance from the ELF. • 1 means that a tag '4B' containing the ELF AID shall be present in the CERT.SP.AUT to create an instance from the ELF. The Service Provider who loads the ELF can create an instance from the ELF even if its CERT.SP.AUT does not contain tag '4B' containing the ELF AID.	M
Length of ELF Version	1		M
ELF Version	2	Contains the version of the ELF. The first byte is the major version and the second byte is the minor version present in the ELF header.	M
Length of ELF AID	1		M
ELF AID	5..16	Contains the AID of the ELF.	M
Length of Security Domain AID	1		M
Security Domain AID	0 or 5..16	If not provided, the ELF is loaded in the currently selected Security Domain.	C
Length of Load File Data Block Hash	1		M
Load File Data Block Hash	0..n	See [GPCS] section C.2.	C
Length of Load Parameters field	1..2		M
Load Parameters field	0..n	See [GPCS] section 11.5.2.3.7.	C
Length of Load Token	1-3		M
Load Token	0..n	See [GPCS] section 11.5.	C

If the ELF is already present on the SE, the License byte shall be ignored but shall be present.

7.7.4 Response Message

This command may return either a general status word as listed in section 7.1 or the following status word.

Table 7-11: SEMS_INSTALL_FOR_LOAD Status Words

SW1	SW2	Meaning
'69'	'85'	Condition of use not satisfied: • The ELF is present on the SE but has not been loaded via the SEMS mechanism. • No right to load an ELF. • The ELF AID referenced in the SEMS_BEGIN_PERSO command is different from the one referenced in the SEMS_INSTALL_FOR_LOAD command.

7.8 SEMS_LOAD Command (Tag '52')

7.8.1 Definition and Scope

The SEMS_LOAD command is used to load an ELF.

The format of the C-DATA field is given in [GPCS] Table 11-58.

The SEMS_LOAD command shall generate a LOAD APDU command. The current Security Level is applied to that APDU command.

7.8.2 Precondition

The previous SEMS command shall be a successful SEMS_INSTALL_FOR_INSTALL.

7.8.3 Data Field Format

The SEMS_LOAD command is formatted as follows:

Table 7-12: SEMS_LOAD Data Field Format

Data Element	Length	Description	M/O/C
Logical Channel ID	1	RFU	M
P1	1	Contains the LOAD P1 (More blocks/last block)	M
P2	1	Contains the LOAD P2 (Block number)	M
Lc	1	Contains the length of the C-DATA field	M
C-DATA	1..n	Contains the LOAD Data field	M

7.8.4 Response Message

This command may return either a general status word as listed in section 7.1 or the following status word.

Table 7-13: SEMS_LOAD Status Words

SW1	SW2	Meaning
'6A'	'80'	Wrong SE response

7.9 SEMS_INSTALL_FOR_INSTALL Command (Tag '53')

7.9.1 Definition and Scope

The SEMS_INSTALL_FOR_INSTALL command is used to explicitly install an Application (SD or applet instance).

7.9.2 Precondition

The instantiation of an Application or SD shall be allowed by the CERT.SP.AUT.

If the CERT.SP.AUT certificate holder is not allowed to create at least one SD with Authorized Management privilege, the instantiation of the Application shall be rejected if the CERT.SP.AUT does not allow the instantiation or if the SEMS SP certificate holder has already created the maximum number of instances allowed by the CERT.SP.AUT.

If the instantiation of an Application from the ELF requires a License, the SEMS_INSTALL_FOR_INSTALL command is rejected in either of the following cases:

- If the CERT.SP.AUT does not include the required ELF License, or
- If the SEMS SP certificate holder has created the maximum number of instances allowed and does not have the right to create an AMSD.

The SEMS_INSTALL_FOR_INSTALL command shall fail if the privileges defined in the payload are not defined in tag 'C5' or 'C6' of CERT.SP.AUT.

The SEMS_INSTALL_FOR_INSTALL command shall fail if all of the following are true:

- The ELF has not been loaded by a SEMS script performed by the same SP certificate holder.
- The Application to create is not an SD.
- The ELF AID of the Application to be instantiated does not match the content of tag '4B' of the currently active CERT.SP.AUT.

If the Application to create is an SD, the OPEN saves the SEMS SP certificate holder identifier (content of tag '5F20' of CERT.SP.AUT).

7.9.3 Data Field Format

The SEMS_INSTALL_FOR_INSTALL command is formatted as follows:

Table 7-14: SEMS_INSTALL_FOR_INSTALL Data Field Format

Data Element	Length	Description	M/O/C
Logical Channel ID	1	RFU	M
Install Options	1	Indicates whether the INSTALL [for install] or the INSTALL [for install and make selectable] command shall be executed. See Table 7-15.	M
Length of ELF AID	1		M
ELF AID	5..16	Contains the AID of the ELF.	M
Length of Executable Module AID	1		M
Executable Module AID	5..16	Contains the AID of the Executable Module.	M
Length of Application AID	1		M
Application AID	5..16	Contains the AID of the Application.	M
Length of Privileges	1	Shall be '1' or '3'.	M
Privileges	1, 3	Privileges as specified in [GPCS] section 11.1.2. The Global Delete privilege is not allowed.	C
Length of Install Parameters field	1..2	As specified in [GPCS] Table 11-43	M
Install Parameters field	2..n	As specified in [GPCS] section 11.5.2.3.7	C
Length of Install Token	1	0 (No Token)	M

The SEMS_INSTALL_FOR_INSTALL command shall generate an INSTALL [for install] or an INSTALL [for install and make selectable] APDU command.

The Install Options data field is formatted as follows:

Table 7-15: Install Options Data Field Format

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0	0	0	0	1	0	0	'04' : INSTALL [for install]
0	0	0	0	1	1	0	0	'0C' : INSTALL [for install and make selectable]

The current Security Level is applied to the above APDU command.

7.9.4 Response Message

This command may return either a general status word as listed in section 7.1 or the following status word.

Table 7-16: SEMS_INSTALL_FOR_INSTALL Status Words

SW1	SW2	Meaning
'69'	'85'	Condition of use not satisfied: • No right to create an instance. • No ELF license. • Privilege not authorized.

7.10 SEMS_DELETE Command (Tag '54')

7.10.1 Definition and Scope

The SEMS_DELETE command is used to explicitly delete an ELF or an Application.

If the Application or ELF to be deleted is not present on the SE, SEMS_DELETE returns SW '9000'.

7.10.2 Precondition

An Application including an SSD shall only be deleted if all of the following conditions are satisfied:

- If the SEMS script, including the SEMS_DELETE command, is provided by the SP Device Application which provided the SEMS script used to create the instance of the Application to be deleted (through STORE DATA of tag '4F' as described in section 6.4.4).

  If the Application has been deleted with a GlobalPlatform DELETE APDU command, the check of the ME Application which created the instance and requested the deletion is not performed.
- If the SEMS script, including the SEMS_DELETE command, is signed by the same SEMS SP certificate holder as the SEMS script which created the instance of the Application to be deleted or the deletion is allowed by tag '4D'.

An ELF loaded through GlobalPlatform commands (i.e. without SEMS command) may not be deleted with a SEMS script.

Further, a Security Domain shall only be deleted:

- If the SD has no associated entities: neither SDs nor Applications nor ELFs.

An ELF shall only be deleted if both of the following are true:

- If there is no Application instantiated from the ELF to be deleted.
- If the SEMS script including the SEMS_DELETE command is signed by the same SEMS SP certificate holder as the SEMS script used to load or update the ELF to be deleted.

7.10.3 Data Field Format

The SEMS_DELETE command is formatted as follows:

Table 7-17: SEMS_DELETE Data Field Format

Data Element	Length	Description	M/O/C
Logical Channel ID	1	RFU	M
P1	1	'00'	M
P2	1	'00' or '80'	M
Lc	1	Contains DELETE Lc	M
C-DATA	Var.	Contains DELETE C-DATA	M

7.10.4 Response Message

This command may return either a general status word as listed in section 7.1 or one of the following status words.

Table 7-18: SEMS_DELETE Status Words

SW1	SW2	Meaning
'69'	'85'	Condition of use not satisfied: • Precondition not satisfied. • SEMS SP certificate holder cannot delete the Application/ELF.
'6A'	'88'	Reference data not found.

7.11 SEMS_PULL_KEY Command (Tag '56')

7.11.1 Definition and Scope

The SEMS_PULL_KEY command is used to load an initial key set into an SD. This key set is generated by the SEMS Application.

The SEMS_PULL_KEY command takes as its input parameter the ECDSA public key to be used for the encryption of the generated key. The corresponding private key is stored securely by the external entity (typically the SP) which generated the SEMS script containing the SEMS_PULL_KEY command.

After the successful integrity verification and decryption of the SEMS_PULL_KEY command (using PK.SP.AUT and SK.CA-SEMS.ENC), the following operations shall be performed:

- The SEMS Application randomly generates a 16-byte key (RGK) and derives the three SCP keys.
- The SCP keys are loaded into the created SD with the PUT KEY command as defined in [GPCS].
- The SEMS Application is requested to encrypt the RGK as described in section 4.8.2.1.
    - RGK is encrypted with an AES-CBC with IV = 0, no padding.
    - A key pair (r, R) is generated.
        - r is the private key and R is the public key.
    - The AES 128-bit key is the result of the 128 most significant bits of SHA-256 applied on the ECDH result using the PK.SP.ENC.S4.
    - The ephemeral public key R is concatenated to the encrypted RGK.
- The SEMS Application is requested to sign with the SK.CASD.AUT private key following the concatenation of:
    - The ephemeral public key R
    - The encrypted RGK
    - The AID of the created SD
    - The SE SN

The signature algorithm used is ALG_ECDSA_SHA_256.

If the verification of the SEMS_PULL_KEY fails, the created SD shall be deleted.

7.11.2 Precondition

The SD AID of the SEMS_PULL_KEY command shall be the same as the Application AID passed as an argument of the most recently executed SEMS_INSTALL_FOR_INSTALL command.

7.11.3 Data Field Format

The SEMS_PULL_KEY command is formatted as follows:

Table 7-19: SEMS_PULL_KEY Data Field Format

Data Element	Length	Description	M/O/C
Logical Channel ID	1	RFU	M
Key version number	1	KVN of the parent SD	M
Length of the parent SD AID	1	Length in the range [5..16]	M
AID of the parent SD	5..16	AID of the parent SD	M
Length of the SSD AID	1	Length in the range [5..16]	M
SSD AID	5..16	AID of the SSD into which the key set is populated	M
KVN	1	Key Version Number; identifies a key or group of keys that are already present on the card. A value of '00' indicates that a new key or group of keys is being added. (The new Key Version Number is indicated in the data field of the command message). The Key Version Number is coded from '01' to '7F' (bits b1 to b7). Bit b8 is set to 0.	M
Key Type	1	Key type as coded in [GPCS] section 11.1.8 Supported key types are: • '82' (i.e. SCP02) • '88' (i.e. SCP03)	M

Key Length	1	Key Length:
		for SCP03: 16 bytes
		for SCP02: 16 bytes
			M

Length of PK.SP.ENC	1	Length of the PK.SP.ENC	M
PK.SP.ENC.S4	65	Public key for the encryption of the SEMS-generated key. The public key shall use uncompressed encoding as defined in [TR-03111].	M

7.11.4 Response Message

Table 7-20: SEMS_PULL_KEY Response Message

Data Element	Length	Description	M/O/C
Length of R	1	Length of ephemeral public key R	M
R	65	Ephemeral public key in uncompressed encoding as defined in [TR-03111] section 3.2.1, meaning ('04' || X || Y)	M
Length of the RGK	1	Length of the encrypted generated key concatenated to the SD AID and SE SN	M
Encrypted RGK	Var.	Encrypted generated key	M
Length SD AID	1	Length of SD AID	M
SD AID	5..16	AID of the created SD	M
Length SE Serial Number	1	Length of the Serial Number	M
SE SN	Var.	Serial Number	M
Signature length	length	Length of the signature	M
Signature	70	The signature is calculated on all previous fields including length value using SK.CASD.AUT with the ALG_ECDSA_SHA_256 algorithm. The format of the signature is given in Table 7-21.	M

Table 7-21: Signature Format

Tag	Length	Value Description			Presence
'30'	68	Signature			Mandatory
		Tag	**Length**	**Value Description**	
		'02'	32	r component of the signature	Mandatory
		'02'	32	s component of the signature	Mandatory

This command may return either a general status word as listed in section 7.1 or the following status word.

Table 7-22: SEMS_PULL_KEY Status Words

SW1	SW2	Meaning
'69'	'85'	Condition of use not satisfied.

7.12 SEMS_PULL_KEY_GP Command (Tag '5D')

7.12.1 Definition and Scope

This command is used to personalize an SD with a confidential personalization of a secure channel key set compliant with the variant of GlobalPlatform scenario #1 described in section 4.8.2.2 or with GlobalPlatform scenario #3 defined in [Amd A].

(A SEMS_PULL_KEY_GP command with length equal to 0 may be sent to retrieve the remaining byte(s) that could not be retrieved in the previous SEMS_PULL_KEY_GP. For details, see section 7.13.)

7.12.2 Precondition

- The CASD shall have been personalized with the SK.CASD.AUT and CERT.CASD.AUT to build the response of scenario #1.
- The CASD shall have been personalized with the SK.CASD.ECKA and CERT.CASD.ECKA to build the response of scenario #3.

7.12.3 Data Field Format

The SEMS_PULL_KEY_GP command is formatted as follows:

Table 7-23: SEMS_PULL_KEY_GP Data Field Format

Data Element	Length	Description	M/O/C
Logical Channel ID	1	RFU	M
Command Control	1	00: MORE meaning that remaining data need to be transferred with SEMS_PULL_KEY_GP 01: LAST meaning that all data have been transferred	M
KVN	1	KVN of the ascendant SD	M
Length of the ascendant SD AID	1	Length in the range [5..16]	M
AID of the ascendant SD	5..16	AID of the ascendant SD	M
Length of the SSD AID	1	Length in the range [5..16]	M
SSD AID	5..16	AID of the SSD into which the key set is populated	M
'A6'	Var.	Control Reference Template for the key generation as described in Table 7-24	M
'7F49'	Var.	• RSA Public key as described in Table 7-26 if value field of tag '90' of tag 'A6' = '01' • ECC ephemeral public key as described in Table 7-27 if the first byte of the value field of tag '90' of tag 'A6' = '03'	M

Table 7-24: Control Reference Template for Key Generation

Tag	Length (bytes)	Description			Presence
'A6'	Var.	Control Reference template for confidentiality			
		Tag	**Length**	**Description**	**Mandatory**
		'90'	1 or 2	Byte 1 '01': Pull Model; i.e. scenario #1 '03': Key agreement; i.e. scenario #3 Byte 2 Scenario parameters as described in Table 7-25; present only if byte 1 = '03'	Mandatory
		'95'	1	Key Usage Qualifier '5C': 1 secure based channel (not supported by mode 3) '10': 3 secure channel keys (not supported by mode 1)	Mandatory
		'96'	1	Key Access according to [GPCS] Table 11-18 For mode 1, key access shall be '00'.	Optional
		'80'	1	Key type is restricted to '80' or '82' (i.e. SCP02) if tag '90' is '01'. Key type is restricted to '80' or '82' (i.e. SCP02) or '88' (i.e. SCP03) if the first byte of tag '90' is '03'.	Mandatory
		'81'	1	Key length (in bytes)	Mandatory
		'82'	1	Key Identifier = '00' – '7F'. If not present, the default key identifier is '01'	Optional
		'83'	1	Key Version Number = '01' – '6F'. If not present, the default KVN is '01' for scenario #1.	Optional
		'91'	Var.	00 = Initial value of sequence counter for scenario #1 Any value is supported for scenario #3.	Optional
		'45'	1-n	Security Domain Image Number	Optional
		'84'	1-n	Host ID (shall be present for scenario #3 only if b3 of scenario parameters is set)	Conditional

Tag 'A6' implemented by SEMS is restricted to the generation of secure channel keys of the same type and length.

Tag '82' indicates the Key Identifier assigned to the first key. The assigned Key Identifier shall be incremented by 1 for each subsequent key. If tag '82' (Key Identifier) is not present, a Key Identifier of '01' shall be assigned to the first key.

The sequence counter is initialized to 0 for mode 1.

If the indicated key set already exists, then it shall be replaced; otherwise a new key set shall be created.

Table 7-25: Parameters for Scenario #3

b8	b7	b6	b5	b4	b3	b2	b1	Description
-	-	-	-	-	-	-	1	Do not delete existing key
-	-	-	-	-	-	1	-	Include Derivation Random in the key derivation function (for details, see [Amd A] section 3.5.4)
-	-	-	-	-	1	-	-	Include Host and Card ID in key derivation process
x	x	x	x	x	-	-	-	RFU

Table 7-26: RSA Public Key

<table>
<tr><th>Tag</th><th>Length (bytes)</th><th colspan="3">Description</th><th>Presence</th></tr>
<tr><td rowspan="4">'7F49'</td><td rowspan="4">Var.</td><td colspan="3">Public key</td><td></td></tr>
<tr><th>Tag</th><th>Length</th><th>Description</th><th>Mandatory</th></tr>
<tr><td>'82'</td><td>1 or 3</td><td>Public key exponent</td><td>Mandatory</td></tr>
<tr><td>'81'</td><td>Var.</td><td>Public key modulus</td><td>Mandatory</td></tr>
</table>

Table 7-27: ECC Public Key

DGI	Length (bytes)	Description	Presence
'7F49'	Var.	Public key formatted using uncompressed encoding 04 || X || Y	Mandatory

7.12.4 Response Message

7.12.4.1 Scenario #1

Table 7-28: SEMS_PULL_KEY_GP Response Message for Scenario #1

Length	Description
1-3	Length of the signature '00' – '80', or '81 80' – '81 FF', or '82 01 00' – '82 FF FF' e.g. '80' for a 1024-bit RSA key, '82 01 00' for a 2048 RSA key
Var.	Signature
1-3	Length of the Remainder Data '00' – '80', or '81 80' – '81 FF', or '82 01 00' – '82 FF FF' e.g. '80' for a 1024-bit RSA key, '82 01 00' for a 2048 RSA key
Var.	Remainder data

7.12.4.2 Scenario #3

Table 7-29: SEMS_PULL_KEY_GP Response Message for Scenario #3

Tag	Length	Description	Presence
'85'	16	DR value generated by SEMS Application	Conditional
'86'	Var.	Receipt	Mandatory

7.12.4.3 Status Word Returned by the Command

This command may return either a general status word as listed in section 7.1 or one of the following status words.

Table 7-30: SEMS_PULL_KEY_GP Status Words

SW1	SW2	Meaning
'6A'	'80'	• The key type, key length is not consistent with the intended secure channel protocol. • If mode 3, bit b1 of byte 2 is set to 0. • If mode 1, tag '84' is given. • Value field of tag '95' is other than '5C'. • Value field of tag '91' is other than 0. • Tag length of tag '90', '91', '95', '96', '80', '81', '82', '83' is not correct. • RSA Public key is incorrectly formatted. • Tag '96' is not '00' if mode 1.
'69'	'85'	Condition of use not satisfied: • Private key has not been personalized. • Previous command is neither SEMS_PULL_KEY_GP nor SEMS_PULL_KEY nor SEMS_INSTALL_FOR_INSTALL.

7.13 SEMS_PULL_KEY_GP Command (Tag '5D') with Length = 0

7.13.1 Definition and Scope

The SEMS_PULL_KEY_GP command with length equal to 0 may be sent to retrieve the remaining byte(s) that could not be retrieved in the previous SEMS_PULL_KEY_GP.

If there is no more data to return, then no data is returned and the SW is '9000'.

7.13.2 Precondition

The previous SEMS command shall be a successfully executed SEMS_PULL_KEY_GP.

7.13.3 Data Field Format

This command has no data field.

7.13.4 Response Message

The command returns at maximum the next 245 bytes. It may return 0 bytes if there is no remaining byte.

This command may return either a general status word as listed in section 7.1 or the following status word.

Table 7-31: Status Words for SEMS_PULL_KEY_GP with Data Field Length 0

SW1	SW2	Meaning
'69'	'85'	Condition of use not satisfied: • The previous SEMS command is not SEMS_PULL_KEY_GP.

7.14 SEMS_PUT_KEY (Tag '5B')

7.14.1 Definition and Scope

The SEMS_PUT_KEY command is used to load an SCP02 / SCP03 key set in an SD created through the SEMS mechanism. The key set may be diversified by the SEMS Application.

7.14.2 Preconditions

The most recent successfully executed command shall be the SEMS_INSTALL_FOR_INSTALL command (having instantiated the SD referenced in the current SEMS_PUT_KEY command) or a SEMS_PULL_KEY command targeting the SD referenced in the current SEMS_PUT_KEY command.

The SD to be personalized shall be defined without the Authorized Management privilege.

7.14.3 Data Field Format

Table 7-32: SEMS_PUT_KEY Data Field Format

Data Element	Length	Description	M/O/C
Logical Channel ID	1	RFU	M
Key version number of the parent SD SCP	1	KVN of the parent SD	M
Length of the AID of the parent SD	1	The length value shall be in the range [5..16]	M
AID of the parent SD	5..16	AID of the parent SD	M
Mode	1	• '01': Key set is defined in C-DATA • '02': Key set is defined in C-DATA and is diversified by SEMS Application	M
Length of the SSD AID	1	The length value shall be in the range [5..16]	M
SSD AID	5..16	AID of the SSD where the key set is populated to.	M
P1 of GP PUT KEY	1	Key Version Number; identifies a key or group of keys already present on the card. A value of '00' indicates that a new key or group of keys is being added. (The new Key Version Number is indicated in the data field of the command message). The Key Version Number is coded from '01' to '7F' (bits b1 to b7). Bit b8 is set to 0.	M

Data Element	Length	Description	M/O/C
P2 of GP PUT KEY	1	Reference Control Parameter; defines a Key Identifier and specifies whether one or multiple keys are contained in the data field. When one key is contained in the command message data field, the reference control parameter indicates the Key Identifier of this key. When multiple keys are contained in the command message data field, the reference control parameter indicates the Key Identifier of the first key in the command data field. Each subsequent key in the command message data field has an implicit Key Identifier that is sequentially incremented by one, starting from this first Key Identifier. The Key Identifier is coded from '00' to '7F'; i.e. bits b1 to b7. Bit b8 shall be 1, meaning that multiple keys are provided in C-DATA.	M
LC	1	Length of C-DATA	M
C-DATA	Var.	• For all Mode values, C-DATA contains [KVN + Key type length + Key Type + length of Key Data Value + Key Data Value] with Key Data Value = [encryption key + MAC key + DEK]. • If Mode is '02', the Key Data Value contains the Base keys. The Base key set is used to derive SCP keys diversified with the SE SN, as described in section 7.14.4.	M

7.14.4 SCP Key Generation

The SCP03 Security Domain keys are derived from the key given in the C-DATA field in the following way:

- KENC^{Div} = AES(KENC)[16-byte SE Serial Number]
- KMAC^{Div} = AES(KMAC)[16-byte SE Serial Number]
- KDEK^{Div} = AES(KDEK)[16-byte SE Serial Number]

The SCP02 Security Domain keys are derived from the key given in the C-DATA field in the following way:

- KENC^{Div} = DES3(KENC)[8 MSB byte SE SN] || DES3(KENC)[8 LSB byte SE SN]
- KMAC^{Div} = DES3(KMAC)[8 MSB byte SE SN] || DES3(KMAC)[8 LSB byte SE SN]
- KDEK^{Div} = DES3(KDEK)[8 MSB byte SE SN] || DES3(KDEK)[8 LSB byte SE SN]

7.14.5 Response Message

This command may return either a general status word as listed in section 7.1 or one of the following status words.

Table 7-33: SEMS_PUT_KEY Status Words

SW1	SW2	Meaning
'69'	'85'	Condition of use not satisfied: • The previous command is neither SEMS_INSTALL_FOR_INSTALL nor SEMS_PUT_KEY nor SEMS_PULL_KEY. • The previous SEMS_INSTALL_FOR_INSTALL or SEMS_PUT_KEY or SEMS_PUT_KEY command does not reference the SD targeted by the current command.
'6A'	'88'	Reference data not found: • The referenced key set has not been personalized in the SEMS Application.

7.15 SEMS_GET_DATA Command (Tag '57')

7.15.1 Definition and Scope

The SEMS_GET_DATA command is used to retrieve data from the SEMS.

(A SEMS_GET_DATA command with length equal to 0 may be sent to retrieve the remaining byte(s) that could not be retrieved in the previous SEMS_GET_DATA. For details, see section 7.16.)

7.15.2 Data Field Format

The SEMS_GET_DATA command is formatted as follows:

Table 7-34: SEMS_GET_DATA Data Field Format

Data Element	Length	Description	M/O/C
DOid length	1	Length of the DOid parameter	M
DOid	1 or 2	The parameter DOid defines the data object to be returned. See Table 7-35.	M
Search criteria length	1	Search criteria length	O
Search criteria	Var.	TLV defined in Table 7-36	C

Table 7-35: DOid List Supported by SEMS

DOid	Description	M/O/C
'7F21'	CERT.CASD.{AUT, ECKA, ECDSA}	M

Table 7-36: Search Criteria Tag Supported by SEMS

Tag	Length	Value
'90'	1	'01': CERT.CASD.AUT '03': CERT.CASD.ECKA 'A0': CERT.CASD.ECDSA

If DOid equals '7F21' without any search criteria, then SEMS_GET_DATA returns CERT.CASD.ECDSA.

If DOid equals '7F21' and the SC tag is present, then SEMS_GET_DATA shall return:

- CERT.CASD.AUT if the SC data field value is '01'
- CERT.CASD.ECKA if the SC data field value is '03'
- CERT.CASD.ECDSA if the SC data field value is 'A0'

7.15.3 Response Message

The structure of the CERT.CASD.ECDSA certificate is defined in section 4.5.3. The structure of CERT.CASD.AUT is specified in [Amd A] Table 3-4. The structure of CERT.CASD.ECKA is specified in [Amd A] Table 3-6.

This command may return either a general status word as listed in section 7.1 or one of the following status words.

Table 7-37: SEMS_GET_DATA Status Words

SW1	SW2	Meaning
'69'	'85'	Condition of use not satisfied: • Requested data has not been personalized.
'6A'	'80'	• Wrong length of DOid • Wrong search criteria length • Search criteria wrongly formatted
'6A'	'88'	Wrong SEMS command data • Unknown Data Object ID • Unknown search criteria tag • Unknown value in the search criteria data field

7.16 SEMS_GET_DATA Command (Tag '57') with Length = 0

7.16.1 Definition and Scope

The SEMS_GET_DATA tag with length equal to 0 may be sent to retrieve the remaining byte that could not be retrieved in the previous SEMS_GET_DATA.

7.16.2 Precondition

The previous SEMS command shall be a successfully executed SEMS_GET_DATA.

7.16.3 Data Field Format

This command has no data field.

7.16.4 Response Message

The command returns at maximum the next 245 bytes. It may return 0 bytes if there is no remaining byte.

This command may return either a general status word as listed in section 7.1 or the following status word.

Table 7-38: Status Words for SEMS_GET_DATA with Data Field Length 0

SW1	SW2	Meaning
'69'	'85'	Condition of use not satisfied: • The previous SEMS command is not SEMS_GET_DATA.

7.17 SEMS_BINDING_SE (Tag '58')

7.17.1 Definition and Scope

All SEMS commands executed after the SEMS_BINDING_SE command within a SEMS script are rejected with SW '6985' unless the Serial Number of the SE matches the value given in the SEMS_BINDING_SE command.

7.17.2 Data Field Format

The SEMS_BINDING_SE command is formatted as follows:

Table 7-39: SEMS_BINDING_SE Data Field Format

Data Element	Length	Description	M/O/C
Length	1	Length of the Serial Number	M
Serial Number	Var.	SE serial number	O

The SE serial number is defined as the concatenation of the CERT.CASD.ECDSA CA identifier, set in tag '42', with the CERT.CASD.ECDSA serial number, set in tag '93'.

7.17.3 Response Message

This command may return either a general status word as listed in section 7.1 or the following status word.

Table 7-40: SEMS_BINDING_SE Status Words

SW1	SW2	Meaning
'69'	'85'	Condition of use not satisfied.

7.18 SEMS_KEY_ROTATION Command (Tag '59')

7.18.1 Definition and Scope

The SEMS_KEY_ROTATION command is used to rotate the PK.CA-SEMS.AUT public key and/or the SK.CA-SEMS.ENC private key, defined in section 4.5.1.

7.18.2 Precondition

The CERT.SP.AUT, successfully verified by SEMS, shall contain tag '4C' with bit b2 set to 1 (see Table 4-5).

7.18.3 Data Field Format

The SEMS_KEY_ROTATION command is formatted as follows:

Table 7-41: SEMS_KEY_ROTATION Data Field Format

Data Element	Length	Description	M/O/C
RFU	1	RFU	M
Length of SK.CA-SEMS.ENC	1	Length of the new SK.CA-SEMS.ENC	M
SK.CA-SEMS.ENC	32	New SK.CA-SEMS.ENC	O
Length of SK.CA-SEMS.ENC ID	1	Length of the SK.CA-SEMS.ENC identifier	M
SK.CA-SEMS.ENC ID	24	SK.CA-SEMS.ENC identifier. It shall be present if SK.CA-SEMS.ENC is present.	C
Length of PK.CA-SEMS.AUT	1	Length of the new PK.CA-SEMS.AUT	M
PK.CA-SEMS.AUT	65	New PK.CA-SEMS.AUT	O
Length of CA-SEMS ID	1	Length of the new CA-SEMS identifier	M
CA-SEMS ID	16	New Entity identifier. It shall be present if PK.CA-SEMS.AUT is present.	C
Length of Key ID	1	Length of the new CA-SEMS signature verification key identifier	M
Key ID	8	New CA-SEMS signature verification key identifier. It shall be present if PK.CA-SEMS.AUT is present.	C

7.18.4 Response Message

This command may return either a general status word as listed in section 7.1 or the following status word.

Table 7-42: SEMS_KEY_ROTATION Status Words

SW1	SW2	Meaning
'69'	'85'	Condition of use not satisfied.

7.19 SEMS_BEGIN_MANAGE_ELF_UPGRADE Command (Tag '5E')

7.19.1 Definition and Scope

The SEMS_BEGIN_MANAGE_ELF_UPGRADE command is used to mark the start of an ELF upgrade process as defined in [Amd H], while the end of the upgrade is marked with SEMS_END_MANAGE_ELF_UPGRADE.

SEMS_BEGIN_MANAGE_ELF_UPGRADE shall compare the version of the package present on the SE and the version given as parameter. If the version given as parameter is strictly greater than the version of the ELF whose AID is given in parameters, then the SEMS_APDU (MANAGE ELF UPGRADE) shall be executed. If the version given as parameter is equal or lower than the version of the ELF whose AID is given in parameters, then the behavior of the SEMS_APDU (MANAGE ELF UPGRADE) depends on the Action parameter and is defined in the following table.

Table 7-43: Action Parameter Description

b8	b7	b6	b5	b4	b3	b2	b1	Action Parameter	Description
-	-	-	-	-	-	0	1	IGNORE	The SEMS_APDU (MANAGE ELF UPGRADE) are silently ignored.
-	-	-	-	-	-	1	0	EXECUTE	The SEMS_APDU (MANAGE ELF UPGRADE) are executed.
-	-	-	-	-	-	1	1	INTERRUPT	SEMS_BEGIN_MANAGE_ELF_UPGRADE returns the error SW '6985'.
x	x	x	x	x	x	-	-		RFU

The SEMS_BEGIN_MANAGE_ELF_UPGRADE command shall determine whether the following conditions are true for each ELF AID in the List of ELFs data element:

- If the ELF AID was loaded with SEMS, then the entity that signed the script to update the ELF AID is the entity that signed the script to load it, as identified in tag '5F20' of the CERT.SP.AUT Certificate, i.e. Subject identifier (a.k.a. certificate holder).
- If the ELF AID was pre-loaded, then in the CERT.SP.AUT Certificate, a tag '4D' indicates that the ELF AID has the update privilege.

If for any ELF AID in the List of ELFs data element either condition above is not met, then the error SW '6985' shall be returned.

7.19.2 Precondition

SEMS_BEGIN_MANAGE_ELF_UPGRADE and SEMS_END_MANAGE_ELF_UPGRADE shall not be nested. SEMS_BEGIN_PERSO and SEMS_BEGIN_MANAGE_ELF_UPGRADE shall not be nested.

7.19.3 Data Field Format

The SEMS_BEGIN_MANAGE_ELF_UPGRADE command is formatted as follows:

Table 7-44: SEMS_BEGIN_MANAGE_ELF_UPGRADE Data Field Format

Data Element	Length	Description	M/O/C
RFU	1	RFU	M
Length of ELF AID	1	Length of ELF AID	M
ELF AID	5..16	ELF AID present on the SE. Used to verify the version.	M
Length of ELF version	1	Length of ELF version	M
Version of the new ELF	2	The first byte is the major version and the second byte is the minor version present in the CAP file header.	M
Action	1	As defined in Table 7-43: Action Parameter Description	M
List of ELFs	1	Number of ELF AIDs	M
Length of ELF AID	1	Length of ELF AID	C
ELF AID	5..16	ELF AID present on the SE	C
…	…	…	…
Length of ELF AID	1	Length of ELF AID	C
ELF AID	5..16	ELF AID present on the SE	C

Note 1: The List of ELFs only contains the ELF AIDs for which the update privilege shall be verified and does not necessarily contain all the ELF AIDs that may be included in subsequent SEMS_APDU (MANAGE ELF UPGRADE) commands.

Note 2: The ELF AID used to verify the version shall be duplicated in the List of ELFs data element if the update privilege on this ELF AID needs to be verified.

7.19.4 Response Message

This command may return either a general status word as listed in section 7.1 or the following status word.

Table 7-45: SEMS_BEGIN_MANAGE_ELF_UPGRADE Status Words

SW1	SW2	Meaning
'69'	'85'	If any of the following conditions occurs: • The SEMS_BEGIN_MANAGE_ELF_UPGRADE and SEMS_BEGIN_PERSO are nested within each other • The List of ELFs contains the AID of an ELF: o that was not loaded with SEMS, or o that was loaded with a secured SEMS script with different holder identifier (tag '5F20'), or o that was preloaded but the corresponding permission for ELF upgrade (tag '4D' in CERT.SP.AUT) is missing • The version given as parameter is equal or lower than the version of the ELF AID given as parameter and Action parameter is equal to INTERRUPT • Preconditions for successful update as specified in [Amd H] are not satisfied
'6A'	'88'	ELF AID is not present on the SE.

7.20 SEMS_END_MANAGE_ELF_UPGRADE (Tag '5F')

7.20.1 Precondition

The SEMS_BEGIN_MANAGE_ELF_UPGRADE shall have been successfully executed.

7.20.2 Data Field Format

The SEMS_END_ MANAGE_ELF_UPGRADE command has no parameter.

7.20.3 Response Message

This command may either return a general error condition as listed in section 7.1 or the following error condition.

Table 7-46: SEMS_END_MANAGE_ELF_UPGRADE Status Words

SW1	SW2	Meaning
'69'	'85'	Condition of use not satisfied. • SEMS_BEGIN_MANAGE_ELF_UPGRADE is missing.

Annex A SEMS Implementation Details

A.1 Application Identifiers and SEMS State Coding

Table A-1: SEMS Application Identifier Definition

Component		AID

SEMS Application	Executable Load File	`GP RID || SEMS || 00 || version (2 bytes)`
		`(A0 00 00 01 51 53 45 4D 53 00 01 00)`

	Executable Module	ELF AID with masked `version` set to `'00 00'`
	Application Instance	`EM AID || 01`

SEMS Updater	Executable Load File	`GP RID || SEMS || FF || version (2 bytes)`
		`(A0 00 00 01 51 53 45 4D 53 FF 01 00)`

	Executable Module	ELF AID with masked `version` set to `'FF FF'`
	Application Instance	`EM AID || 01`

Table A-2 defines the coding of the states that are involved in a SEMS command.

Table A-2: SEMS State Coding Definition

State Name	Hex	Definition
SELECTABLE	'5A'	The SEMS Application has been instantiated but not personalized with all needed credentials defined in section 4.5.
PERSONALIZED	'A5'	The SEMS Application has been instantiated and personalized with all needed credentials defined in section 4.5.

Details on Loading and Update of an ELF

An ELF is upgraded if all of the following conditions are met:

- The version given in the SEMS_INSTALL_FOR_LOAD is newer than the version of the already present ELF (otherwise the subsequent SEMS_LOAD commands are ignored).
- The ELF in the SE has been loaded by the SEMS SP certificate holder that signed the SEMS script containing the ELF to be updated. Note that an ELF that has not been loaded via the SEMS service shall not be updated with a SEMS script.
- Each EM implements the `OnUpgradeListener` interface defined in [Amd H], allowing its data to persist across different ELF upgrades.

If the above conditions are satisfied, the SEMS Application shall perform the following operations for all Application instances created from the ELF being upgraded:

1. Retrieve the privileges and Application Life Cycle state. The SEMS Application only manages the (contact) Application Life Cycle states defined by GlobalPlatform; i.e. INSTALLED, LOCKED, and SELECTABLE. If bits b4 to b7 of the Application Life Cycle state are set or if the contactless Life Cycle State is assigned, it is the responsibility of the Application to back up and restore those additional Life Cycle states.

2. Trigger the backup of the Application(s) instantiated from this ELF (see [Amd H]) by having the OPEN call the `onSave()` and `onCleanup()` methods defined in the `OnUpgradeListener` interface exposed by each Application. For a contactless application, the content of the Contactless registry shall be retrieved by the Applications and saved like any other Application data.
3. Delete all the Applications that have been created from the ELF to be updated and then delete the ELF.
4. Execute the SEMS_INSTALL_FOR_LOAD command and all subsequent SEMS_LOAD commands. The new ELF is loaded on the SE and associated with the SD which is passed as an argument within the SEMS_INSTALL_FOR_LOAD command, if present. Otherwise, the loaded ELF is associated with the SD where the ELF was installed before the execution of the script. If the ELF cannot be extradited to the SD given as parameter, the script is stopped: The ELF present on the SE has been deleted as well as all instances created from the ELF.

   If the SEMS_INSTALL_FOR_INSTALL or a SEMS_LOAD command fails, the SEMS stops the loading of the ELF but keeps all saved data until:

   - A further attempt to load the ELF succeeds.
   - A SEMS_DELETE successfully deletes the ELF.
5. Re-create all Application instances available before the ELF deletion and restore their privileges and Application Life Cycle states. The SEMS Application does not automatically restore the Application Specific Life Cycle state. Thus, bits b7 to b4 of the Application Life Cycle state and the Contactless Life Cycle state byte are not restored by the SEMS Application. It is the responsibility of the application itself to restore this Life Cycle state information.

   The SEMS recreates the Application instances in the SD(s) where they were installed before the ELF deletion. If the extradition of an instance to its original SD fails, the instance will remain in the SD where the ELF has been loaded.
6. Restore the saved Application data and the Contactless registry (see [Amd H]) by having the OPEN call the `onRestore()` method defined in the `OnUpgradeListener` interface exposed by each Application.

The SEMS cannot update a library when it is referenced by one or more Application(s).

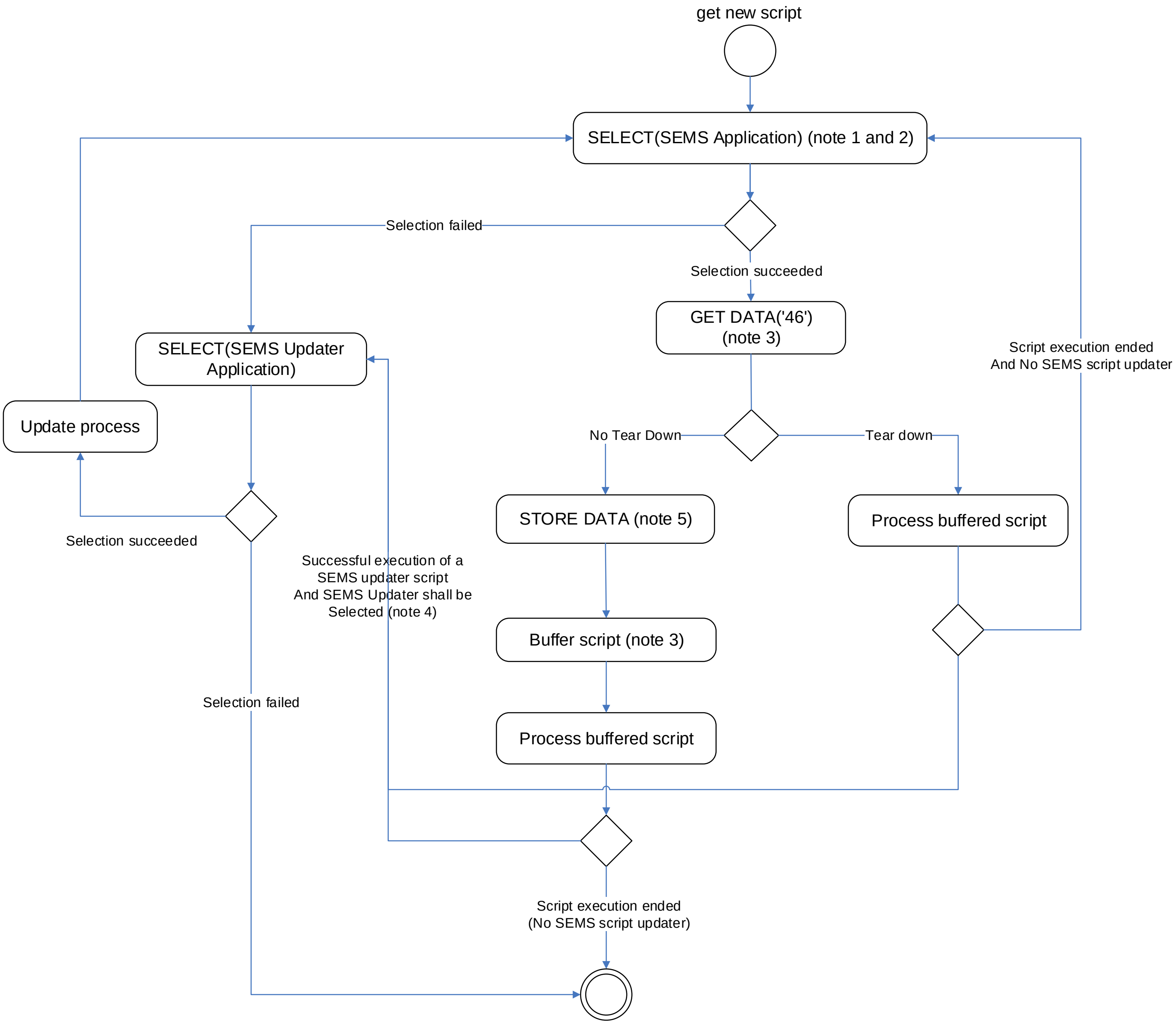
The update of the SEMS Registry following the successful loading of an ELF is described in section 7.7. Note that if the selected Security Domain is the SEMS AMSD (the AMSD the SEMS Application is associated with), it is recommended to associate the ELF with the SEMS AMSD and not to extradite the ELF to another SD to ensure the update of the ELF since the SEMS AMSD may not have the right to delete the ELF loaded in another SD. To load an ELF in an SD other than the SEMS AMSD, it is recommended to authenticate with the ISD.

If the execution of a script to update an ELF is interrupted because of an OS reboot or a teardown of the ME, the SEMS Device Agent should replay the same script.

Annex B SEMS Device Agent

Figure B-1 describes the control flow of the SEMS Device Agent. The SEMS Device Agent takes a SEMS script as input.

Figure B-1: SEMS Device Agent Process


get new script
SELECT(SEMS Application) (note 1 and 2)
Selection failed
Selection succeeded
GET DATA('46') (note 3)
Script execution ended And No SEMS script updater
SELECT(SEMS Updater Application)
Update process
No Tear Down
Tear down
STORE DATA (note 5)
Process buffered script
Selection succeeded
Successful execution of a SEMS updater script And SEMS Updater shall be Selected (note 4)
Buffer script (note 3)
Selection failed
Process buffered script
Script execution ended (No SEMS script updater)

Note 1: The SEMS Device Agent selects either the SEMS Application or the SEMS Updater. The SEMS Updater is selected if the SEMS Application is not present (meaning that the SEMS Application update has been initiated as described in section 4.15).

Note 2: The SEMS Device Agent buffers tag '42' and tag '45' extracted from the SELECT response. Tag '45' and tag '42' uniquely identify the SEMS-CA.

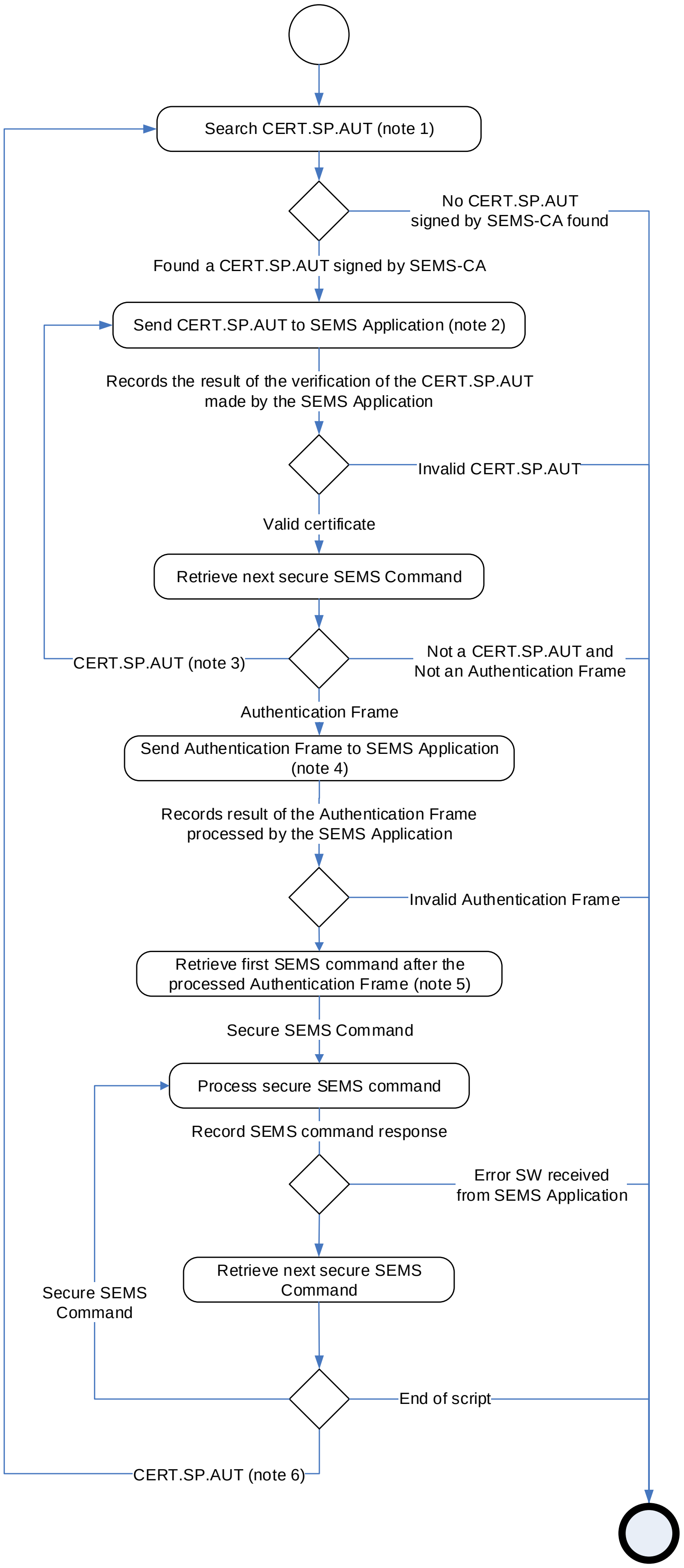
Note 3: If the previous SEMS script processing has been interrupted because of teardown, the SEMS Device Agent shall replay the buffered script then process the new script, otherwise the SEMS Device Agent buffers and processes the new script as described in Figure B-2.

Note 4: Once the SEMS Updater is instantiated (after the new version of the SEMS Application ELF has been loaded), PROCESS SCRIPT COMMAND returns an SW (see Table 6-8) indicating that the SEMS Updater shall be selected to finalize the SEMS update process. If there is a teardown, then the SEMS Updater will be selected on the next SEMS script processing.

If the update process for SEMS Application has started, the SEMS Application will reject selection until the update process is completed.

Note 5: The SEMS Device Agent sends the SP Device Application identifier to the SEMS Application. See section 6.4.3 for details on the SP Device Application identifier.

Figure B-2: Process Buffered SEMS Script Flow Diagram


Search CERT.SP.AUT (note 1)
No CERT.SP.AUT signed by SEMS-CA found
Found a CERT.SP.AUT signed by SEMS-CA
Send CERT.SP.AUT to SEMS Application (note 2)
Records the result of the verification of the CERT.SP.AUT made by the SEMS Application
Invalid CERT.SP.AUT
Valid certificate
Retrieve next secure SEMS Command
CERT.SP.AUT (note 3)
Not a CERT.SP.AUT and Not an Authentication Frame
Authentication Frame
Send Authentication Frame to SEMS Application (note 4)
Records result of the Authentication Frame processed by the SEMS Application
Invalid Authentication Frame
Retrieve first SEMS command after the processed Authentication Frame (note 5)
Secure SEMS Command
Process secure SEMS command
Record SEMS command response
Error SW received from SEMS Application
Retrieve next secure SEMS Command
Secure SEMS Command
End of script
CERT.SP.AUT (note 6)

Note 1: The SEMS Device Agent looks for the first CERT.SP.AUT certificate of which tags '42' and '45' match the buffered tags '42' and '45' (see Figure B-2) returned by the SELECT command.

Note 2: The SEMS Device Agent sends the matching CERT.SP.AUT in a PROCESS SCRIPT COMMAND APDU command, defined in section 6.2, to the SEMS Application.

Note 3: In the case of a certificate chain, the SEMS Device Agent sends all chained CERT.SP.AUT before sending the Authentication Frame.

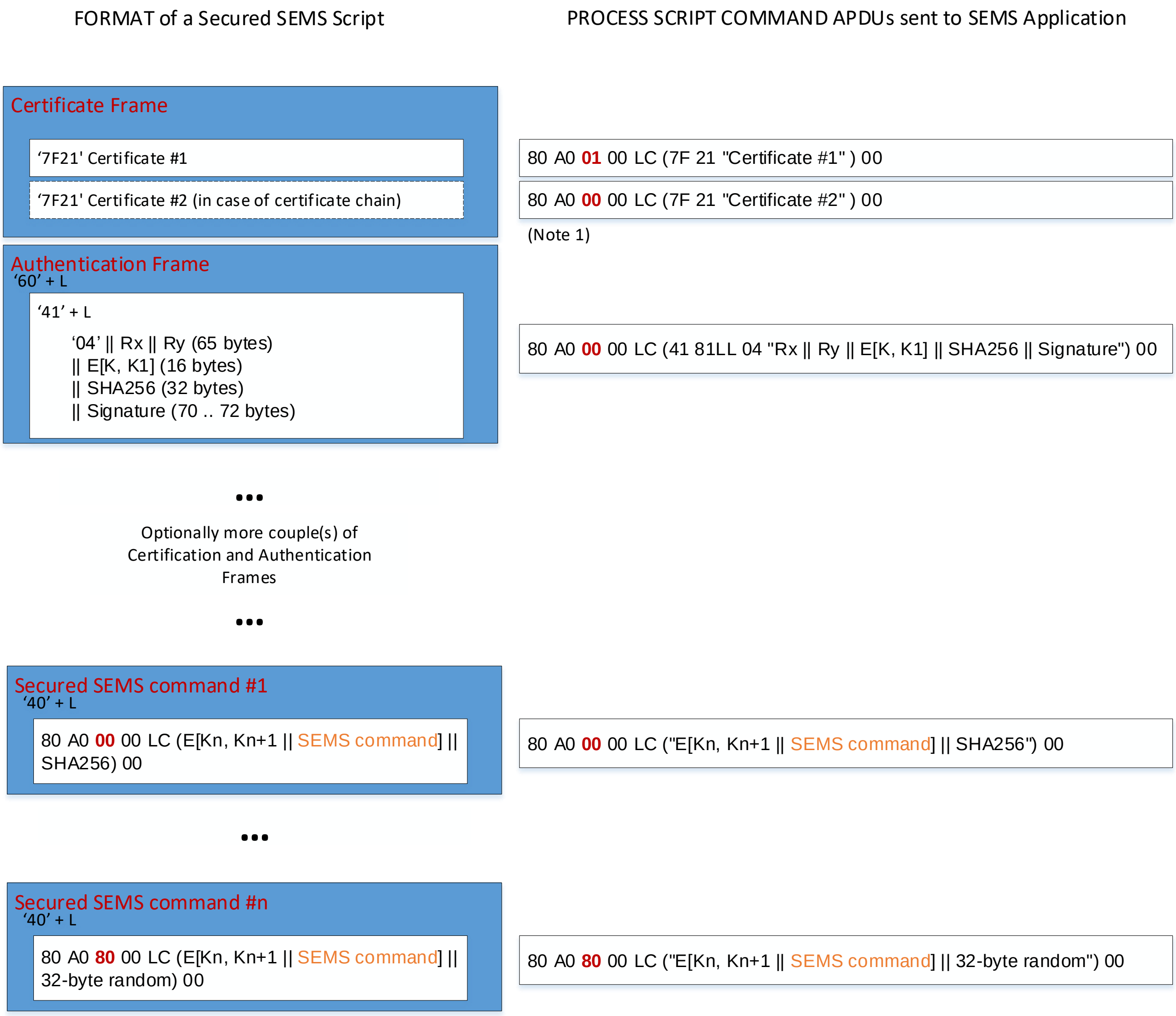
Note 4: SEMS Device Agent sends an Authentication Frame in a PROCESS SCRIPT COMMAND APDU command, defined in section 6.2, to the SEMS Application.

Note 5: After processing the Authentication Frame, the SEMS Device Agent shall skip all CERT.SP.AUT and Authentication Frames until it finds a secured SEMS command as defined in Table 5-3.

Note 6: After a SEMS command, it is possible that the SEMS Device Agent needs to process a CERT.SP.AUT if the SEMS script contains several sequences of CERT.SP.AUT followed by an Authentication Frame and a sequence of SEMS commands.

Figure B-3 summarizes the content of a Secured SEMS Script and the corresponding CAPDUs sent to the SEMS Application by the SEMS Device Agent.

Figure B-3: PROCESS SCRIPT COMMANDs Extracted from the Secured SEMS Script


FORMAT of a Secured SEMS Script
PROCESS SCRIPT COMMAND APDUs sent to SEMS Application
Certificate Frame
'7F21' Certificate #1
'7F21' Certificate #2 (in case of certificate chain)
80 A0 01 00 LC (7F 21 "Certificate #1") 00
80 A0 00 00 LC (7F 21 "Certificate #2") 00
(Note 1)
Authentication Frame
'60' + L
'41' + L
'04' || Rx || Ry (65 bytes)
|| E[K, K1] (16 bytes)
|| SHA256 (32 bytes)
|| Signature (70 .. 72 bytes)
80 A0 00 00 LC (41 81LL 04 "Rx || Ry || E[K, K1] || SHA256 || Signature") 00
•••
Optionally more couple(s) of Certification and Authentication Frames
•••
Secured SEMS command #1
'40' + L
80 A0 00 00 LC (E[Kn, Kn+1 || SEMS command] || SHA256) 00
80 A0 00 00 LC ("E[Kn, Kn+1 || SEMS command] || SHA256") 00
•••
Secured SEMS command #n
'40' + L
80 A0 80 00 LC (E[Kn, Kn+1 || SEMS command] || 32-byte random) 00
80 A0 80 00 LC ("E[Kn, Kn+1 || SEMS command] || 32-byte random") 00

Note 1: Certificates may not fit into a single PROCESS SCRIPT COMMAND. In this case the SEMS Device Agent splits the certificates into multiple PROCESS SCRIPT COMMANDs. Only the very first PROCESS SCRIPT COMMAND will set P1 to '01'.