GLOBALPLATFORM®

THE STANDARD FOR MANAGING APPLICATIONS ON SECURE CHIP TECHNOLOGY

GlobalPlatform

Card Specification

Version 2.3

Public Release

October 2015

Document Reference: GPC_SPE_034

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

Part I
Introduction

1 Introduction

GlobalPlatform is an organization that has been established by leading companies from the payments and communications industries, the government sector and the vendor community, and is the first to promote a global infrastructure for smart card implementation across multiple industries. Its goal is to reduce barriers hindering the growth of cross-industry, multiple Application smart cards. The smart card issuers will continue to have the freedom to choose from a variety of cards, terminals, and back-end systems.

For smart cards to reach their true potential, consumers need to be able to use them for a wide variety of functions. For example, the cards can be used with mobile phones to make purchases over the Internet as well as to securely access a PC. Smart cards should also be cost effective and easily multifunctional.

Beginning in the mid-1990s, a number of very significant breakthroughs occurred in the chip card industry with the introduction of open systems specifications for Application development. The leading technologies in this area are Java Card™ and MULTOS™. These technology specifications provide an important contribution to the solution towards the multi-Application chip card vision, such as common programming standards allowing Application portability between different card specific implementations.

Then in 2001 a PKI-based card content management framework was defined by NICSS (the Next generation IC Card System Study group) especially for the governmental sector market. The concept of NICSS is to separate the application provider from card issuer in a trusted manner whereby applications providers can 'rent' card memory space from the card issuer. This creates a new business model for each stakeholder. Additionally, in the mobile telecommunication sector ETSI have been addressing card content management 'over the air' for the secure management of the SIM and third generation UICC.

Through the Open Platform initiative, first Visa International and now GlobalPlatform have been working with the chip card industry to deliver a missing and critically important chip card standard — a hardware-neutral, vendor-neutral, Application-independent card management specification. This new specification provides a common security and card management architecture that protects the most important aspect of a chip card system investment — the infrastructure.

GlobalPlatform defines a flexible and powerful specification for Card Issuers to create single- and multi-Application chip card systems to meet the evolution of their business needs. The specification allows them to choose the card technology that is right for them today while also ensuring that they can migrate, if necessary, to a different card technology in the future without significant impact to their infrastructure.

This specification describes the GlobalPlatform Specifications that shall be implemented on GlobalPlatform smart cards.

In this specification, some elements are identified as 'deprecated'. Such elements are no longer maintained and no evolution of these elements will be considered in the future. There is no guarantee that these elements will be present in a future release of this specification. Therefore, it is not recommended to include deprecated elements in new products or to reference them in new specification documents.

1.1 Audience

This specification is intended primarily for card manufacturers and application developers developing GlobalPlatform card implementations. Although this specification defines card components, command interfaces, transaction sequences, and interfaces that can be common across many different industries, it does not detail the implementation of the lower layers security, which may vary from one industry to the other.

This specification is also intended for a more general audience as it describes the generic security concepts and the various actors involved in a multi-Application Card Management System.

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 v2.2 Amendment A	GlobalPlatform Card Specification 2.3 Amendment A v1.1 – Confidential Card Content Management	[Amd A]
GlobalPlatform Card Specification v2.2 Amendment B	GlobalPlatform Card Specification 2.2 Amendment B v1.1.3 – RAM over HTTP	[Amd B]
GlobalPlatform Card Specification v2.2 Amendment C	GlobalPlatform Card Specification 2.3 Amendment C v1.2 – Contactless Services	[Amd C]
GlobalPlatform Card Specification v2.2 Amendment D	GlobalPlatform Card Specification 2.2 Amendment D v1.1.1 – Secure Channel Protocol '03'	[Amd D]
GlobalPlatform Card Specification v2.2 Amendment E	GlobalPlatform Card Specification 2.3 Amendment E v1.0.1 – Security Upgrade for Card Content Management	[Amd E]
GlobalPlatform Card Specification v2.2 Amendment F	GlobalPlatform Card Specification 2.2 Amendment F v1.0 – Secure Channel Protocol '11'	[Amd F]
GlobalPlatform System Protocol Discovery Mechanism Specification	GlobalPlatform System, System Protocol Discovery Mechanism Specification v1.0	[GP SPDM]
GlobalPlatform KMS	GlobalPlatform Key Management System Functional Requirements, version 1.0	[KMS Req]
GlobalPlatform MS	GlobalPlatform Messaging Specification, version 1.0	[Messaging]
GlobalPlatform SCMS	GlobalPlatform Smart Card Management System Functional Requirements, version 4.0	[SCMS Req]
GlobalPlatform Systems Scripting Language Specification	GlobalPlatform Systems Scripting Language Specification, version 1.1.0	[Scripting Lang]
ANSI X9.52	Triple Data Encryption Algorithm Modes of Operation, draft, 1996	[ANSI X9.52]

Standard / Specification	Description	Ref
ANSI X9.62-2005	Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA)	[ANSI X9.62]
BSI TR-02102, Version 1.0	BSI Technische Richtlinie TR-02102: Kryptographische Verfahren: Empfehlungen und Schlüssellängen	[TR 02102]
BSI TR-03111, Version 1.11	BSI Technical Guideline TR-03111: Elliptic Curve Cryptography	[TR 03111]
CWA 14890-1	CEN Workshop Agreement – Application Interface for smart cards used as Secure Signature Creation Devices – Part 1: Basic requirements, March 2004	[CWA 14890-1]
ETSI TS 102 221 (Release 6 or higher)	Smart cards; UICC – Terminal interface; Physical and logical characteristics, European Telecommunications Standards Institute Technical Committee Smart Card Platform (TC SCP), 2004	[TS 102 221]
ETSI TS 102 225 (Release 6 or higher)	Smart cards; Secured packet structure for UICC based applications, European Telecommunications Standards Institute Technical Committee Smart Card Platform (TC SCP), 2004	[TS 102 225]
ETSI TS 102 226 (Release 6 or higher)	Smart cards; Remote APDU structure for UICC based applications, European Telecommunications Standards Institute Technical Committee Smart Card Platform (TC SCP), 2004	[TS 102 226]
ETSI TS 102 241 (Release 6 or higher)	Smart cards; UICC Application Programming Interface (UICC API) for Java Card™, European Telecommunications Standards Institute Technical Committee Smart Card Platform (TC SCP), 2004	[TS 102 241]
FIPS PUB 140-2	Federal Information Processing Standards Publication 140-2: Security Requirements for Cryptographic Modules, May 2001	[FIPS 140-2]
FIPS PUB 180-2	Federal Information Processing Standards Publication 180-2, 2002: Specifications for the Secure Hash Standard: U.S. Department of Commerce, Technology Administration, National Institute of Standards and Technology	[FIPS 180-2]
FIPS PUB 186-4	Federal Information Processing Standards Publication 186-4: Digital Signature Standard (DSS)	[FIPS 186-4]
FIPS PUB 197	Federal Information Processing Standards Publication 197, 2001: Specification for the Advanced Encryption Standard (AES): U.S. Department of Commerce, Technology Administration, National Institute of Standards and Technology	[FIPS 197]

Standard / Specification	Description	Ref
FIPS PUB 198	Federal Information Processing Standards Publication 198, 2002: Standard for the Keyed-Hash Message Authentication Code (HMAC): U.S. Department of Commerce, Technology Administration, National Institute of Standards and Technology	[FIPS 198]
ISO/IEC 7816-3:1997	Identification cards – Integrated circuit(s) cards with contacts – Part 3: Electronic signals and transmission protocols	[ISO 7816-3]
ISO/IEC 7816-4:2005	Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange	[ISO 7816-4]
ISO/IEC 7816-6:2004	Identification cards – Integrated circuit(s) cards with contacts – Part 6: Interindustry data elements	[ISO 7816-6]
ISO/IEC 7816-15:2004	Identification cards – Integrated circuit cards with contacts – Part 15: Cryptographic information application	[ISO 7816-15]
ISO 8731-1:1987	Banking – Approved algorithms for message authentication – Part 1: DEA	[ISO 8731-1]
ISO/IEC 8825-1:2002 | ITU-T Recommendation X.690 (2002)	Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)	[ISO 8825-1]
ISO/IEC 9594-8 | ITU-T Recommendation X.509 (2000)	Information Technology – Open Systems Interconnection – The Directory: Public-Key and Attribute Certificate Frameworks	[X.509]
ISO/IEC 9796-2:2002	Information technology – Security techniques – Digital signature schemes giving message recovery – Part 2: Integer factorization based mechanisms	[ISO 9796-2]
ISO/IEC 9797-1	Information technology – Security techniques – Message Authentication Codes (MACs) – Part 1: Mechanisms using a block cipher	[ISO 9797-1]
ISO/IEC 9899:1999	Programming languages – C	[ISO 9899]
ISO/IEC 10116: 1997	Information technology – Modes of operation of an n-bit block cipher algorithm	[ISO 10116]
ISO/IEC 10118-3: 1998	Information technology – Security techniques – Hash functions – Part 3: Dedicated hash functions	[ISO 10118-3]
ISO/IEC 14443-3:2001	Identification cards – Contactless integrated circuit(s) cards – Proximity cards – Part 3: Initialization and anticollision	[ISO 14443-3]
ISO/IEC 14443-4:2001	Identification cards – Contactless integrated circuit(s) cards – Proximity cards – Part 4: Transmission protocol	[ISO 14443-4]

Standard / Specification	Description	Ref
ISO/IEC 18033-3:2005	Information Technology – Security techniques – Encryption algorithms – Part 3: Block ciphers	[ISO 18033-3]
Java Card™	Go to the following website for Java Card™ documentation: http://java.sun.com/products/javacard	
MAO-DOC-REF-009	MULTOS Guide to Generating Application Load Units, version 2.51	[MAO-DOC-REF-009]
MULTOS™	Go to the following website for MULTOS™ documentation: http://www.multos.com	
NICSS Framework (NICSS-F)	NICSS Prerequisites, First Edition, version 1.20, April 24, 2001, The Next generation IC Card System Study group	[NICSS-F]
NIST SP 800-38A	Recommendation for Block Cipher Modes of Operation: Methods and Techniques, 2001	[800-38A]
NIST SP 800-38B	Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, May 2005	[800-38B]
NIST SP 800-56A Revision 1	Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography, March 2007	[800-56A]
NIST SP 800-57 Part 1 revised	Recommendation for Key Management – Part 1: General (Revised) March, 2007	[800-57-1]
NIST SP 800-90 revised	Recommendation for Random Number Generation Using Deterministic Random Bit Generators, March 2007	[800-90]
NIST SP 800-108	Recommendation for Key Derivation Using Pseudorandom Functions, October 2009	[800-108]
PKCS#1	PKCS #1 v2.1: RSA Cryptography Standard, RSA Laboratories, June 14, 2002.	[PKCS#1]
RFC 4279	Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)	[RFC 4279]
RFC 4785	Pre-Shared Key (PSK) Ciphersuites with NULL Encryption for Transport Layer Security (TLS)	[RFC 4785]
RFC 5487	Pre-Shared Key Cipher Suites for TLS with SHA-256/384 and AES Galois Counter Mode	[RFC 5487]
RFC 5639	Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation	[RFC 5639]

1.4 Terminology and Definitions

The following meanings apply to SHALL, SHOULD, and MAY in this document:

- SHALL indicates that the statement containing the SHALL must be implemented as defined in this Specification. It does not mandate the implementation of the statement.
- SHOULD indicates a recommendation. It is strongly recommended to implement the statement as defined in this Specification.
- MAY indicates an option.

Table 1-2 defines the expressions used within this Specification that use an upper case first letter in each word of the expression. Expressions within this document that use a lower case first letter in each word take the common sense meaning. (Tagged data elements are also given an upper case first letter in each word of their names.)

Table 1-2: Terminology and Definitions

Term	Definition
Application	Instance of an Executable Module after it has been installed.
Application Management System	An off-card application-specific system required to successfully implement an Application Provider's service to a cardholder.
Application Protocol Data Unit (APDU)	Standard communication messaging protocol between a card accepting device and a smart card.
Application Provider	Entity that owns an application and is responsible for the application's behavior.
Application Session	The link between the Application and the external world on a logical channel starting with the selection of the Application and ending when the same or another Application is selected on the logical channel, the logical channel is closed or the Card Session terminates.
Asymmetric Cryptography	A cryptographic technique that uses two related transformations, a public transformation (defined by the Public Key component) and a private transformation (defined by the Private Key component); these two key components have a property so that it is computationally infeasible to discover the Private Key, even if given the Public Key.
Basic Logical Channel	The permanently available interface between the card and an external entity. The Basic Logical Channel is numbered zero.
Card Content	Code and Application information (but not Application data) contained in the card that is under the responsibility of the OPEN; e.g. Executable Load Files, Application instances, etc.
Card Image Number (CIN)	An identifier for a specific GlobalPlatform card.
Card Issuer	Entity that owns the card and is ultimately responsible for the behavior of the card.
Card Management System	An off-card system providing functions to manage various card types and their associated application(s) and specific configurations for cardholders.
Card Manager	Generic term for the card management entities of a GlobalPlatform card; i.e. the OPEN, Issuer Security Domain, and a Cardholder Verification Method services provider.

Term	Definition
Card Recognition Data	Information that tells an external system, in particular a Smart Card Management System (SCMS), how to work with the card (including indicating that this is a GlobalPlatform card).
Card Session	The link between the card and the external world starting at card reset (contact cards), activation (contactless cards), or power on of the card and ending with a subsequent reset (contact cards), deactivation (contactless cards), or power off of the card.
Card Unique Data	Data that uniquely identifies a card being the concatenation of the Issuer Identification Number and Card Image Number.
Cardholder	The end user of a card.
Cardholder Verification Method (CVM)	A method to ensure that the person presenting the card is the person to whom the card was issued.
Certificate	In this Specification, a Certificate refers to a key certificate: the public key and identity of an entity together with some other information, rendered unforgeable by signing with the private key of the certification authority which issued that Certificate.
C-MAC	MAC appended to an APDU command.
Controlling Authority	An entity independent from the Card Issuer and Application Providers, responsible for enforcing specific off-card and on-card security policies. Such a Controlling Authority is represented on-card by a Security Domain which provides specific functionalities supporting the Controlling Authority's security policy.
Current Security Level	A level of security that is to be applied to the current command-response pair in a Secure Channel Protocol using secure messaging. It is set for an individual command (APDU pair): the current incoming command APDU and the next response.
DAP Block	Part of the Load File used for ensuring Load File Data Block verification.
DAP Verification	A mechanism used by a Security Domain to verify that a Load File Data Block is authentic.
Delegated Management	Pre-authorized Card Content changes performed by an approved Application Provider.
Digital Signature	A cryptographic transformation of data that allows the recipient of the data to prove the origin and integrity of the data; it protects the sender and the recipient of the data against forgery by third parties; it also protects the sender against forgery by the recipient.
Executable Load File	Actual on-card container of one or more application's executable code (Executable Modules). It may reside in Immutable Persistent Memory or may be created in Mutable Persistent Memory as the resulting image of a Load File Data Block.
Executable Module	Contains the on-card executable code of a single application present within an Executable Load File.
GlobalPlatform Registry	A container of information related to Card Content management.

Term	Definition
Host	A logical term used to represent the back end systems that support the GlobalPlatform system; hosts perform functions such as authorization and authentication, administration, Post-Issuance application code and data downloading, and transactional processing.
Immutable Persistent Memory	Memory that can only be read.
Issuer Security Domain	The primary on-card entity providing support for the control, security, and communication requirements of the card administrator (typically the Card Issuer).
Key	A cryptographic key stored in a Security Domain. The key is uniquely identified per Security Domain by the two parameters Key Version Number and Key Identifier. A key may consist of one or more key components; e.g. a symmetric key has only one key component while an asymmetric key has several components.
Key Identifier (KID)	One of the two parameters identifying a key. In the context of a cryptographic operation or protocol performed by a Security Domain, the absolute or relative value of the Key Identifier determines the exact function of the key. See also the definition of Key Version Number.
Key Set	A set of keys used together by a Security Domain to perform some cryptographic operation or protocol (e.g. Secure Channel Protocol). See also Secure Channel key set.
Key Version Number (KVN)	One of the two parameters identifying a key. This parameter defines the general purpose of a key; i.e. its applicability for some cryptographic operation or protocol. For example, keys involved in the execution of a Secure Channel Protocol share the same Key Version Number. The term ‘version number’ is only used for historic reasons and should not be interpreted as such in the current version of this specification. See also the definition of Key Identifier.
Life Cycle	The existence of Card Content on a GlobalPlatform card and the various stages of this existence where applicable; or the stages in the life of the card itself.
Life Cycle State	A specific state within the Life Cycle of the card or of Card Content.
Load File	A file transferred to a GlobalPlatform card that contains a Load File Data Block and possibly one or more DAP Blocks.
Load File Data Block	Part of the Load File that contains one or more application(s) or libraries and support information for the application(s) as required by the specific platform.
Load File Data Block Hash	A value providing integrity for the Load File Data Block.
Load File Data Block Signature	A value encompassing the Load File Data Block Hash and providing both integrity and authenticity of the Load File Data Block.
Message Authentication Code (MAC)	A symmetric cryptographic transformation of data that provides data origin authentication and data integrity.
Mutable Persistent Memory	Memory that can be modified.

Term	Definition
OPEN	The central on-card administrator that owns the GlobalPlatform Registry.
Post-Issuance	Phase following the card being issued to the Cardholder.
Pre-Issuance	Phase prior to the card being issued to the Cardholder.
Private Key	The private component of the asymmetric key pair.
Public Key	The public component of the asymmetric key pair.
Receipt	A cryptographic value provided by the card (if so required by the Card Issuer) as proof that a Delegated Management operation has occurred.
Retry Counter	A counter, used in conjunction with the Retry Limit, to determine when attempts to present a CVM value shall be prohibited.
Retry Limit	The maximum number of times an invalid CVM value can be presented prior to the CVM prohibiting further attempts to present a CVM value.
R-MAC	MAC appended to an APDU response.
Runtime Environment	Functionality on a card which provides a secure environment for multiple applications to operate. Its role is complementary to that of the GlobalPlatform Card Manager.
Secure Channel	A communication mechanism between an off-card entity and a card that provides a level of assurance, to one or both entities.
Secure Channel key set	A set of keys used together by a Security Domain to perform a Secure Channel Protocol. Keys belonging to such a key set have the same Key Version Number and consecutive Key Identifiers. The number of keys required within a Secure Channel key set depends on the Secure Channel Protocol.
Secure Channel Protocol	A secure communication protocol and set of security services.
Secure Channel Session	A session, during an Application Session, starting with the Secure Channel initiation and ending with a Secure Channel termination or termination of either the Application Session or Card Session.
Security Domain	Application having the Security Domain privilege. This on-card entity provides support for the control, security, and communication requirements of an off-card entity such as the Card Issuer, an Application Provider, or a Controlling Authority.
Session Security Level	A mandatory minimum level of security to be applied to protected commands in a Secure Channel Protocol using secure messaging. It is established during the initialization of the Secure Channel Session, either explicitly or implicitly.
Supplementary Logical Channel	Up to 19 additional interfaces (other than the permanently available Basic Logical Channel) between the card and an external entity. Each Supplementary Logical Channel is numbered from 1 up to 19.
Supplementary Security Domain	A Security Domain other than the Issuer Security Domain.
Symmetric Cryptography	A cryptographic technique that uses the same secret key for both the originator's and the recipient's transformation.

Term	Definition
Token	A cryptographic value provided by a Card Issuer as proof that a Delegated Management operation has been authorized.
Trust Point	An authority whose public key is trusted by a Security Domain or Off-Card Entity through some undefined mechanism such as a secure process that delivers the public key in a self-signed certificate. A Trust Point's public key is typically the 'highest' public key known to the entity.
UICC	The ICC as defined by ETSI Project Smart Card Platform (EP SCP).
Verification Authority	A Controlling Authority whose responsibility is to enforce control over card contents using the Mandated DAP Verification mechanism.

1.5 Abbreviations and Notations

Table 1-3: Abbreviations and Notations

Abbreviation / Notation	Meaning
AES	Advanced Encryption Standard
AID	Application Identifier
APDU	Application Protocol Data Unit
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ATR	Answer-to-Reset
ATQ	Answer-to-Request (for contactless cards)
BCD	Binary Coded Decimal
BER	Basic Encoding Rules
CAT	Card Application Toolkit; or Cryptographic Authorization Template
CBC	Cipher Block Chaining
CCT	Control Reference Template for Cryptographic Checksum
CIN	Card Image Number / Card Identification Number
CLA	Class byte of the command message
CRT	Control Reference Template
CT	Control Reference Template for Confidentiality
CVM	Cardholder Verification Method
DAP	Data Authentication Pattern
DEK	Data Encryption Key
DER	Distinguished Encoding Rules
DES	Data Encryption Standard
DST	Control Reference Template for Digital Signature

Abbreviation / Notation	Meaning
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
EMV	Europay, MasterCard, and Visa; used to refer to the ICC Specifications for Payment Systems
ENC	Encryption
FCI	File Control Information
HEX	Hexadecimal
HMAC	Keyed-Hash Message Authentication Code
ICC	Integrated Circuit Card
ICV	Initial Chaining Vector
IIN	Issuer Identification Number
INS	Instruction byte of the command message
ISO	International Organization for Standardization
Lc	Exact length of data in a case 3 or case 4 command
Le	Maximum length of data expected in response to a case 2 or case 4 command
LV	Length Value
MAC	Message Authentication Code
MEL	MULTOS Executable Language. The instruction set of the MULTOS™ runtime environment
OID	Object Identifier
P1	Reference control parameter 1
P2	Reference control parameter 2
PIN	Personal Identification Number
PKI	Public Key Infrastructure
RAM	Random Access Memory
RFU	Reserved for Future Use
RID	Registered Application Provider Identifier
ROM	Read-only Memory
RSA	Rivest / Shamir / Adleman asymmetric algorithm
SCP	Secure Channel Protocol; or (ETSI) Smart Card Platform
SW	Status Word
SW1	Status Word One
SW2	Status Word Two
TLV	Tag Length Value

Abbreviation / Notation	Meaning
TP	Trust Point
'xx'	Hexadecimal values are expressed as hexadecimal digits between single quotation marks.
X	A value in a cell of a table whose purpose is described in the ‘meaning’ column of the table
-	A value (0 or 1) in a cell of a table that does not affect the ‘meaning’ given for that row of the table

1.6 Revision History

1.6.1 Open Platform Card Specification v2.0 to Open Platform Card Specification v2.0.1

This section provides a brief summary of the revisions made to the Open Platform Card Specification 2.0 Card Specification in the Open Platform Card Specification 2.0.1'.

Wording and formatting of the specification had been improved.

Anything relating to a specific implementation of Open Platform had been removed from the main body of the specification and was detailed in the appendices.

Anything specific relating to the personalization of Open Platform or Applications had been removed.

The changes relating specifically to the Java Card™ implementation of Open Platform were listed in the beginning of Appendix A – *GlobalPlatform API*.

All the issues identified in the FAQ document dated April-June 1999 and the FAQ documents dated October-November 1999 and relating strictly to Open Platform, had been included in this version. The one caveat to this was the point 3.1.20 of the FAQ document dated October-November 1999; i.e. it is now required that the Security Domain associated with an Application be the same Security Domain used to perform Delegated Management functions for this Application.

The inclusion of Part V described a specific use of security and key management that was not present in Open Platform 2.0.

1.6.2 Major Adjustments in GlobalPlatform Card Specification v2.1

The following major adjustments are the modifications decided by GlobalPlatform Members. All of these modifications are intended to make the GlobalPlatform more usable for a wider number of entities while maintaining backwards compatibility. The minor editing changes and rewording for readability are not listed.

1.6.2.1 Enhancement to DAP Verification Scheme

In the process of implementing GlobalPlatform cards that supported Delegated Management and DAP Verification, it was determined that the method defined in the previous version of Open Platform necessitated the verification of multiple signatures on large blocks of data (Load File and Load File Data Block) that differed only slightly. When multiple DAP Verifications and possibly Delegated Management was being performed simultaneously, multiple hash generations were required to run concurrently. This negatively impacted the performance of the load process.

A new method has been defined in which instead of signing large blocks of data, a hash of the critical large block of data (Load File Data Block) may be generated and this hash signed. In this new method signatures are required on very much smaller blocks of information. Only one hash generation and check is required on the Load File Data Block.

1.6.2.2 Application Reception of Data from Security Domains

The Secure Channel mechanism previously provided by Open Platform only allowed an Application to request services from its associated Security Domain.

A new service is now provided that may be initiated by a Security Domain. It allows data to be passed by the Security Domain associated with an Application to the Application for further processing. The main purpose of this service is to facilitate the personalization of Applications through the Applications' associated Security Domain. A new INSTALL command has been defined to identify the Application to be personalized.

1.6.2.3 Security Domain Association and Extradition

In the previous Open Platform Card Specification an Executable Load File was associated with a Security Domain and all Applications instantiated from an Executable Load file, when installed, were also associated with the same Security Domain. This method was restrictive.

The new scheme provides Application Extradition. Application Extradition allows an Application that is already associated with a Security Domain to be extradited and associated with another Security Domain.

Another benefit provided by this enhancement is that in addition to Executable Load Files and Applications, now applications within Executable Load Files become visible in the GlobalPlatform Registry at the time that the Executable Load File is registered.

In order to avoid confusion between selectable Applications and the applications within an Executable Load File a new term has been introduced: Executable Module. The term Executable Module is intended to identify the one or more applications present within an Executable Load File.

1.6.2.4 Executable Modules

In the previous Open Platform Card Specification, an off-card entity could only retrieve information relating to the Executable Load Files and selectable Applications present on the card. In order to enhance the information returned by the GET STATUS command, an additional set of information will be stored in the GlobalPlatform Registry and returned in the response to the GET STATUS command. This information relates to the Executable Module and it is now possible to also retrieve information relating to application code within an Executable Load File that is available for installation.

1.6.2.5 Card Recognition Data

A concerted effort was made to ensure that there would be a uniform method to determine basic information about a card. The Card Recognition Data and the method for retrieving this data have been included in this version of the GlobalPlatform Card Specification. Information such as: this is a GlobalPlatform card, implemented in a particular way, and with a particular version number is now available.

1.6.2.6 Support of New Secure Channel Protocols

In order to be more accommodating, the method for including additional Secure Channel Protocols has been formalized. While this was allowed in previous versions of the Open Platform Card Specification, only one Secure Channel Protocol was defined. As part of the efforts of GlobalPlatform a formal process for including Secure Channel Protocols is defined. This version of the specification details two Secure Channel Protocols and their selection process. To provide clarity a slight reorganization of the GlobalPlatform Card specification has been done. Part V of the Open Platform 2.0.1' specification has been removed. Part of its content is incorporated into Part III and the rest is moved into the appendices.

1.6.2.7 Cardholder Verification Method Services

Additional services and features regarding the optional CVM have been included. In the previous versions of the Open Platform Card Specification, the CVM provided the possibility for a single PIN to be common across multiple Applications. When the PIN was changed by one Application it became visible to all Applications. However an Application could not check whether the PIN had previously been correctly presented to another Application during the same Card Session. The new Cardholder Verification Method (CVM) services provide the ability to do this check.

1.6.2.8 Card Manager Separation

Historically the Card Manager has been viewed and defined as the major on-card component with no distinction or separation between the various responsibilities encompassed within as well as not clearly defining where the runtime environment ends and the Card Manager begins. A decision has been made to separate the Card Manager into three distinct entities and clearly identify what runtime environment functionality must be within each. While the term Card Manager is still present, it now encompasses the OPEN, the Issuer Security Domain and the Cardholder Verification Method Services provider. This new structure clarifies the responsibilities of the Card Manager and takes into account both Java Card and Windows Powered Smart Card.

1.6.2.9 Windows Powered Smart Card API

The GlobalPlatform API for Windows Powered Smart Card is now included.

1.6.2.10 Java Card API

Taking into account the various new features of the GlobalPlatform a new API providing support for these new, and all relevant existing, features has been specified for Java Card. The previous Open Platform API for Java Card defined in version 2.0.1' is deprecated and remains in this version for backward compatibility.

1.6.2.11 Appendices

The body of the GlobalPlatform Card Specification only contains generic GlobalPlatform descriptions. All information specific to a particular implementation has been positioned in a set of appendices.

1.6.3 Revisions in GlobalPlatform Card Specification v2.1.1

The following modifications correct issues in the previous version and synchronize with recent evolutions of the underlying runtime environment specifications while maintaining full backwards compatibility. The minor editing changes and rewording for readability are not listed.

1.6.3.1 Errata

All intermediate published errata have been incorporated into this version. There is also a small set of errata and precisions that would have been published at around the same time this version is released, and these have also been incorporated directly into this version.

1.6.3.2 Content Removal

A new optional Card Content removal feature has been added that allows an Executable Load File and all its related Applications to be deleted in the same operation.

1.6.3.3 Logical Channels

To meet the emerging needs of some industries and recent evolutions of the underlying runtime environment specifications, logical channel functionality is added to this version of the specification as an optional feature.

1.6.3.4 Additional Secure Channel Protocol Implementation Options

An enhanced mechanism of generating a C-MAC has been added to both Secure Channel Protocol '01' and Secure Channel Protocol '02'. One new implementation option has been added for Secure Channel Protocol '01' and four new implementation options have been added for Secure Channel Protocol '02'. It is recommended that these new implementation options be used.

1.6.4 Major Adjustments in GlobalPlatform Card Specification v2.2

The following major adjustments are modifications decided by GlobalPlatform Members. All of these modifications are intended to make the GlobalPlatform more usable for a wider number of entities while maintaining full backward compatibility with previous versions.

The body of the document has been reorganized to reflect these changes, to arrange it by functions and components as well as to remove excessive duplication.

1.6.4.1 PKI functionality and Card Content Management

Card Content Management may now be performed by relying exclusively on asymmetric cryptography and a Public Key Infrastructure. This has resulted in the need to formalize the process of authentication and establishing ownership, so that the card can apply the relevant security and authorization rules for different off-card entities.

A delete token is added as an option, so that the Card Issuer may have a PK based policy to authorize card content removal.

1.6.4.2 Over-The Air Functionality and Inter-Application Communication

A mechanism for inter-application communication is formalized in terms of a general framework: Trusted Framework, which encompasses both the GlobalPlatform mechanism whereby an Application could receive its personalization data from its Security Domain as well as the SIM Toolkit and CAT frameworks defined for UICC cards by ETSI Project Smart Card Platform specifications. A new privilege – Trusted Path – between an on-card Receiving Entity and an on-card target Application is introduced.

1.6.4.3 Contactless Cards, Implicit Selection and Logical Channels

Support for contactless cards and dual interface cards (contact and contactless) is made more explicit in this version.

With the introduction of a new installation parameter, different Applications can now be implicitly selected on different card I/O interfaces: contact or contactless (and potentially others), and different logical channels.

The Default Selected privilege is redefined as the Card Reset privilege to modify the historical bytes. An Application is able to refuse explicit selection; e.g. because it does not support the current card I/O interface, and allow the (partial) selection process by OPEN to continue. To provide backward compatibility, the privilege confers implicit selectability if it has not been awarded to another Application.

As a further option, support of logical channels is expanded up to 19 supplementary logical channels as defined by the latest version of [ISO 7816-4].

1.6.4.4 Secure Channel Protocols

A new Secure Channel Protocol based on asymmetric cryptography and a Public Key Infrastructure is introduced as Secure Channel Protocol '10'. Secure Channel Protocol '10' is compatible with CEN Workshop Agreement specification CWA 14890-1 ([CWA 14890-1]). It also fulfills the requirements of NICSS Framework Scheme ([NICSS-F]) defined by the Next Generation IC Card System Study Group (NICSS).

This version references the Secure Channel Protocol defined by ETSI Project Smart Card Platform TS 102 225 specification ([TS 102 225]) as GlobalPlatform Secure Channel Protocol '80'.

The number of recommended options for Secure Channel Protocol '02' is reduced to the most commonly used ones. The option indicator for Secure Channel Protocol '02' is defined as a bitmap and allows the support of any other option that was described in version 2.1.1 or Amendment A.

Secure Channel Protocol '01' is now deprecated.

The use of Security Levels associated with a secure channel session and an individual command-response pair have been made more explicit in this version. This version provides improved API support for Secure Channel Protocols. For example, an Application has the ability to increase the security level required for a (sequence of) individual command-response pair(s) using the GlobalPlatform API.

1.6.4.5 Global Services and CVMs

A general framework for dynamic management of card-wide services is introduced whereby one or more Global Services Applications may be present on a card to provide services to other Applications. The Global Services Applications are distinguished by having the Global Service privilege. A new installation parameter allows the on-card registration of service parameters. Service parameters are standardized by GlobalPlatform and can be registered as unique on the card by Global Services Applications using the GlobalPlatform API. GlobalPlatform API extensions allow Applications to request and, if authorized, use the services offered by Global Services Applications.

CVM functions are devolved from the OPEN to separate CVM Applications as Global Services Applications. Each CVM Application can handle one or multiple CVMs. GlobalPlatform API extensions allow CVM Applications to establish whether Applications have the authority to use and/or modify CVMs. Backward compatibility of the GlobalPlatform API is ensured for existing Applications using CVM services.

1.6.4.6 Security Domains, Privileges and Hierarchies

The privileges associated with Security Domains, in particular the Issuer Security Domain, are formalized so that access rights to Card Content Management functionality are more explicit. The Issuer Security Domain now has an explicit set of privileges, including new privileges such as Authorized Management or Token Verification. Other new privileges are introduced to formalize the access rights awarded to Security Domains and Applications such as Global Registry Access, Global Lock and Global Delete.

Security Domain and Application privileges can now be modified dynamically on the card during their lifetime. CVM identifiers are standardized by GlobalPlatform.

Security Domains and OPEN itself can now have their Card Content Management functionality restricted dynamically during their Life Cycle through the use of a new INSTALL command.

Security Domains can now be associated with other Security Domains and so create a hierarchy of Security Domains. Association with Security Domains is made more systematic, so that a hierarchy of control can be established through association and extradition. Multiple hierarchies of Security Domains can be established, including by extraditing a Security Domain to itself and so making it the root of a new hierarchy. Access rights of Security Domains to Card Content Management functionality are expressed in regard to their association and hierarchy.

Card Content Management functionality has been extended to include explicit extradition of Executable Load Files.

The Token Verification privilege and the Receipt Generation privilege have been added.

1.6.4.7 On-Card APIs for Applications

A GlobalPlatform API expressed in the 'C' programming language for MULTOS™ cards is now included.

The GlobalPlatform API has been expanded and enhanced in order to support the new functionality introduced in this version.

References to Windows Powered Smart Card, and its specific API, have been deleted from this version.

The deprecated Open Platform Java Card API has also been removed from this version.

1.6.4.8 Key Management

New optional attributes for cryptographic keys are added to allow for more control, and for the partitioning of keys with different purposes. The new attributes are key usage and key access condition.

New key types are also added to support new cryptographic algorithms.

1.6.4.9 Amendment A

Amendment A to version 2.1.1 has been incorporated into this version. It comprises a set of optional extensions in support of GlobalPlatform Systems Scripting Language Specification ([Scripting Lang]), EMV Card Personalization Specification, and ETSI Project Smart Card Platform [TS 102 225] and [TS 102 226] specifications.

1.6.4.10 Errata and Precisions

All errata and precisions published since the release of version 2.1.1 and Amendment A have been incorporated into this version.

1.6.4.11 Other Major Changes

New installation parameters are added to further card memory management and allow memory reservation.

The Get Data command can now be used to retrieve a list of Applications present on the card.

It is now possible to lock, and subsequently unlock, a Security Domain and all its associated Applications in a single command.

It is now possible to load, install and make selectable an Application in a single combined process.

1.6.5 Minor Adjustments in GlobalPlatform Card Specification v2.2.1

1.6.5.1 Errata and Precisions

All errata and precisions published since the release of version 2.2 have been incorporated into this version. Additionally, relevant errata and precisions recorded during the development of the GlobalPlatform Card UICC Configuration v1.0 and v1.0.1 have been incorporated into this version and the corresponding API.

The GlobalPlatform API specifications (Java Card™ and MULTOS™) have been removed from this document and are now published separately on the GlobalPlatform website.

1.6.5.2 Amendment A, Confidential Card Content Management v1.0 and its Errata and Precision v1.0

Items from GlobalPlatform Card Specification v2.2 Amendment A Confidential Card Content Management v1.0, sections 4.8 and 4.9 are applied to this revision.

1.6.6 Minor Adjustments in GlobalPlatform Card Specification v2.3

This version incorporates a number of core features which were previously defined in Amendments to v2.2. These modifications are intended to make GlobalPlatform specifications more usable for a wider number of entities while maintaining backwards compatibility. Minor editing changes and rewording for readability are not listed.

1.6.6.1 Rules for Card Content Extradition

It is now forbidden to extradite an Executable Load File to a Security Domain having the DAP Verification privilege (see section 9.4.1).

1.6.6.2 DAP Verification Schemes

The DAP scheme based on AES keys as previously defined in [Amd D] is now defined in this revision. The DAP scheme based on ECC keys as previously defined in [Amd E] is now defined in this revision.

1.6.6.3 Token Verification and Receipt Generation Schemes

The token scheme based on DES keys as previously defined in [Amd A] is now defined in this revision. Token and receipt schemes based on AES keys as previously defined in [Amd D] are now defined in this revision. Token and receipt schemes based on ECC keys as previously defined in [Amd E] are now defined in this revision.

1.6.6.4 Key Loading

Both RSA and symmetric key loading via STORE DATA as previously defined in [Amd A] have been incorporated into this revision.

Key loading using an AES DEK via PUT KEY or STORE DATA as previously defined in [Amd D] has been incorporated into this revision.

ECC key loading via PUT KEY or STORE DATA as previously defined in [Amd E] has been incorporated into this revision.

1.6.6.5 Ciphered Load File ICV

Optional ICV to be used when processing a ciphered load file, as previously defined in [Amd E], has been incorporated into this revision.

1.6.6.6 Card Recognition Data and Security Domain Management Data

To cope with some historical choices, two different formats are now defined for Card Recognition Data (resp. Security Domain Management Data). These two formats differ in the usage of tag '64' (see appendix H). It is now recommended for off-card entities interested in such data to implement correct interpretation of both formats.

1.6.6.7 Card Capability Information

Card Capability Information as previously defined in [Amd E] has been incorporated into this revision.

1.6.6.8 Security Domain Manager URL

Security Domain Manager URL as previously defined in [Amd C] has been incorporated into this revision.

1.6.6.9 Specification of Command Chaining and Long TLV Values

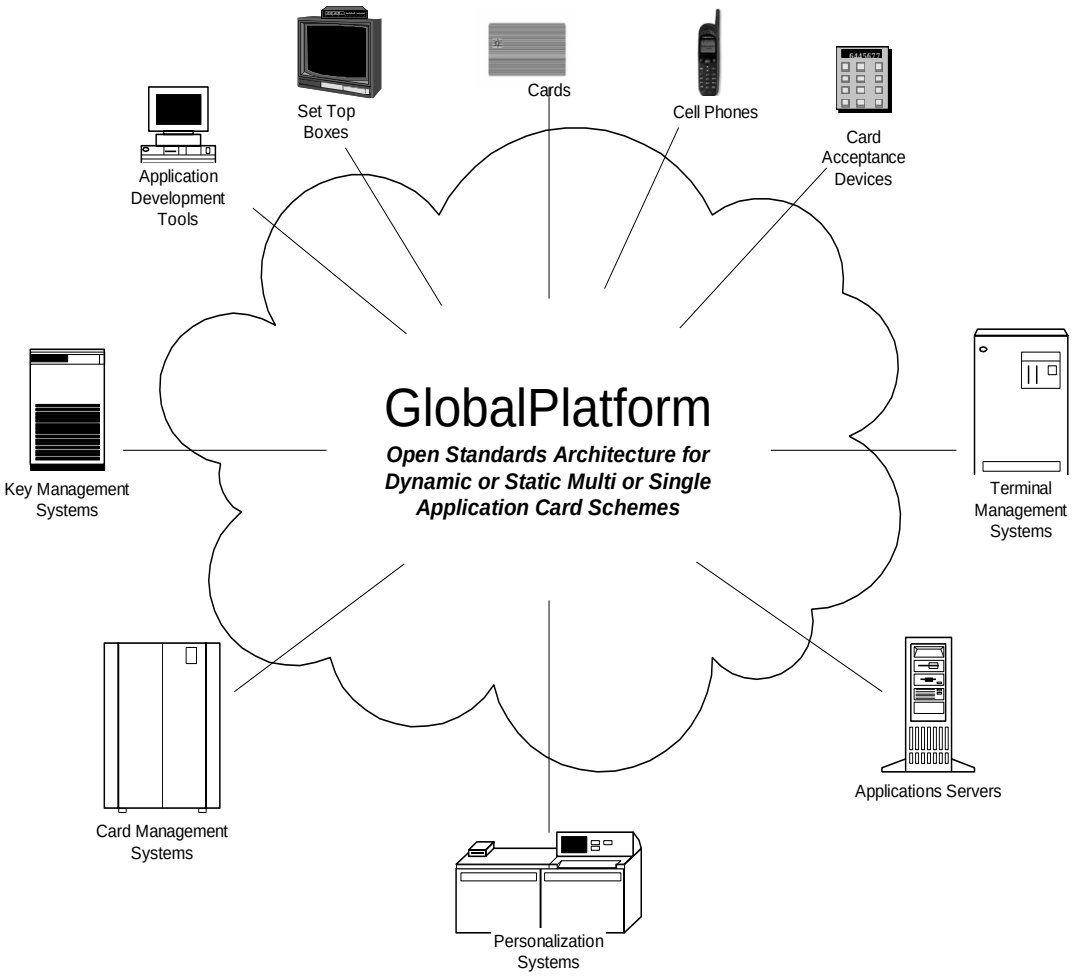
Certificates, digital signatures, and some data fields may be too long to fit in a single command, as a result, command and response chaining mechanisms have been specified. Clarifications to these mechanisms, previously provided in [Amd E], are now defined in this revision.

Part II
Architecture

2 System Architecture

Deploying a large number of chip cards based on dynamic, multi-application technology is not unlike deploying a very large number of workstations in a vast, semi-connected network. These card-based workstations support several different applications at any one time as well as allow for the possibility of updating or deleting those applications and installing new applications at any point in time.

Figure 2-1: GlobalPlatform Architecture



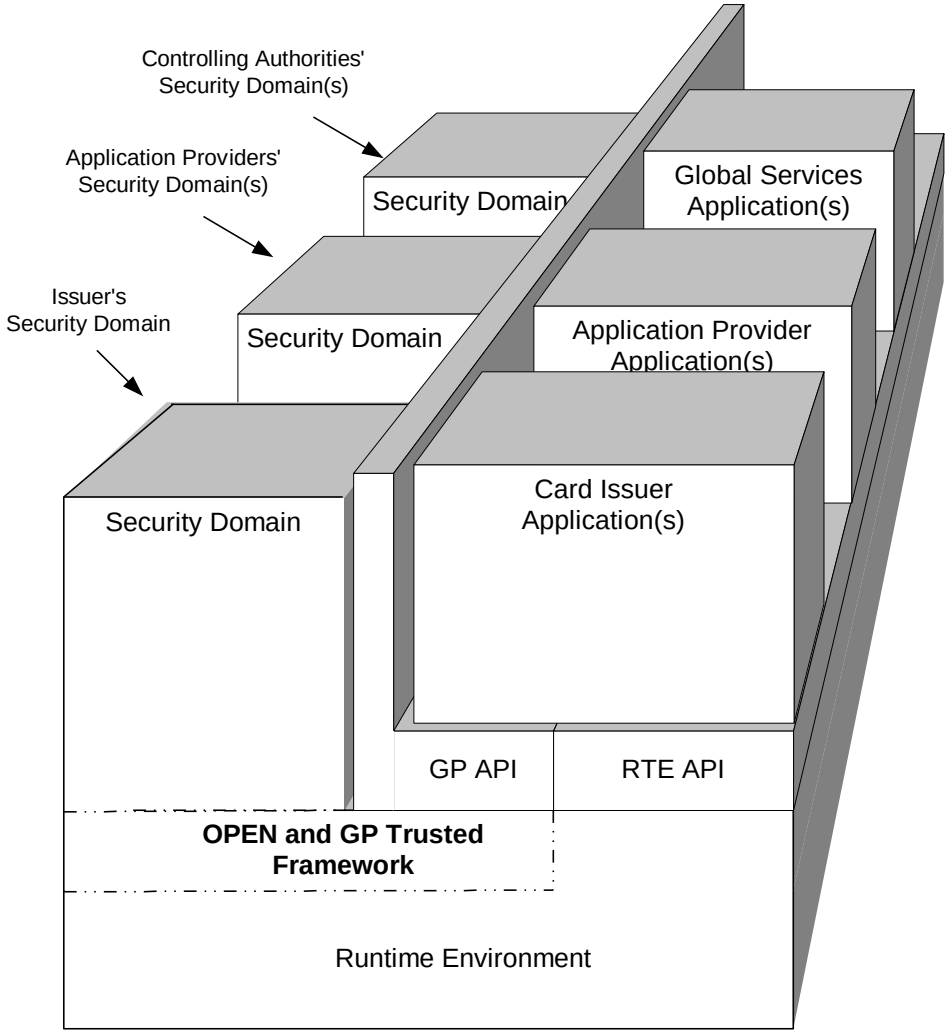
The GlobalPlatform architecture is designed to provide Card Issuers with the system management architecture for managing these smart cards. Although GlobalPlatform is based on the paradigm that there is one single Card Issuer for a card, it offers to the Card Issuer the flexibility for managing an ever-changing array of business partners who may want to run applications on the Card Issuer's cards.

GlobalPlatform gives Card Issuers the power to manage their cards with the ultimate flexibility by enabling them to share control over part of their card with business partners. The ultimate control always rests with the Card Issuer, but through GlobalPlatform, the business partners of a Card Issuer can be allowed to manage their own Applications on the Card Issuer's cards as appropriate.

3 Card Architecture

The GlobalPlatform card architecture is comprised of a number of components that ensure hardware and vendor-neutral interfaces to Applications and off-card management systems. The following figure shows the components in a sample card configuration which includes one or more applications from the Card Issuer; one or more applications from one of the business partners of the Card Issuer, referred to as Application Providers; and one or more applications providing global services (e.g. CVM services) to other applications.

Figure 3-1: GlobalPlatform Card Architecture



All applications shall be implemented in a secure runtime environment that includes a hardware-neutral Application Programming Interface (API) to support application portability. GlobalPlatform does not mandate a specific runtime environment technology. The Card Manager is the primary GlobalPlatform card component that acts as the central administrator for a GlobalPlatform card. Special key and security management applications called Security Domains are created to ensure complete separation of keys between the Card Issuer and multiple other Security Domain providers.

3.1 Security Domains

Security Domains act as the on-card representatives of off-card authorities. There are several types of Security Domains, reflecting the types of off-card entities recognized by a card:

- The Issuer Security Domain is the primary, mandatory on-card representative of the Card Administrator, typically the Card Issuer.
- Supplementary Security Domains are additional, optional on-card representatives of Application Providers or the Card Issuer, or of their agents (e.g. service bureaus).
- Controlling Authority Security Domains are special kinds of Supplementary Security Domains, on-card representatives of Controlling Authorities, if any.

In the main, the above types are simply referred to as Security Domains in this Specification.

Security Domains support security services such as key handling, encryption, decryption, digital signature generation and verification for their providers' (Card Issuer, Application Provider, or Controlling Authority) applications.

Each Security Domain is established on behalf of a Card Issuer, an Application Provider, or a Controlling Authority when these off-card entities require the use of keys that are completely isolated from each other.

3.2 Global Services Applications

One or more Global Services Applications may be present on the card to provide services to other Applications on the card. Examples of such services are Cardholder Verification Method services.

3.3 Runtime Environment

The GlobalPlatform is intended to run on top of any secure, multi-application card runtime environment. This runtime environment is responsible for providing a hardware-neutral API for applications as well as a secure storage and execution space for applications to ensure that each application's code and data can remain separate and secure from other applications on the card. The card's runtime environment is also responsible for providing communication services between the card and off-card entities.

Cards should comply with appropriate standards: [ISO 7816-3], [ISO 7816-4], [ISO 14443-3], and [ISO 14443-4] in terms of announcing options supported in the ATR/ATQ such as the communications protocol, logical channels and command chaining.

3.4 Trusted Framework

GlobalPlatform cards may contain one or more Trusted Frameworks, which provide inter-application communication services between Applications. Trusted Frameworks are not Applications or Security Domains, but have a special status in that they are part of or extensions of the card's run-time environment. They should be assessed for security similarly to the runtime environment's security assessment. See Appendix G – *Trusted Framework Inter-Application Communication* for further details.

3.5 GlobalPlatform Environment (OPEN)

The main responsibilities of the GlobalPlatform Environment (OPEN) are to provide an API to applications, command dispatch, Application selection, (optional) logical channel management, and Card Content management. These functions shall be implemented by the OPEN if the runtime environment does not provide them, or if they are provided by the runtime environment in a way that is not compliant with this Specification.

The OPEN performs the application code loading and related Card Content management and memory management.

The OPEN also manages the installation of applications loaded to the card. The OPEN is responsible for enforcing security principles defined for Card Content management.

Another important function provided by the OPEN is APDU command dispatching and Application selection. When a SELECT command is successfully processed, the OPEN sets the Application referenced in the SELECT command to be the selected Application and subsequent Application commands shall be dispatched to the selected Application.

The availability of logical channels introduces an additional dimension to the card's architecture as multiple Applications may be selected concurrently. The OPEN shall rely on the runtime environment to control whether and when an individual Application can be selected concurrently with itself or another Application. When supporting logical channels, the OPEN shall allow for Applications that have no notion of logical channels as well as those that are multi-selectable. Support of logical channels is optional. Cards may support one or more (up to 19 according to [ISO 7816-4]) Supplementary Logical Channels.

The OPEN owns and uses an internal GlobalPlatform Registry as an information resource for Card Content management. The GlobalPlatform Registry contains information for managing the card, Executable Load Files, Applications, Security Domain associations, and privileges.

3.6 GlobalPlatform API

The GlobalPlatform API provides services to Applications (e.g. Cardholder verification, personalization, or security services). It also provides Card Content management services (e.g. card locking or Application Life Cycle State update) to Applications.

For the specification of the API on a Java Card™, see section A.1.

For the specification of the API on a MULTOS™ card, see section A.2.

3.7 Card Content

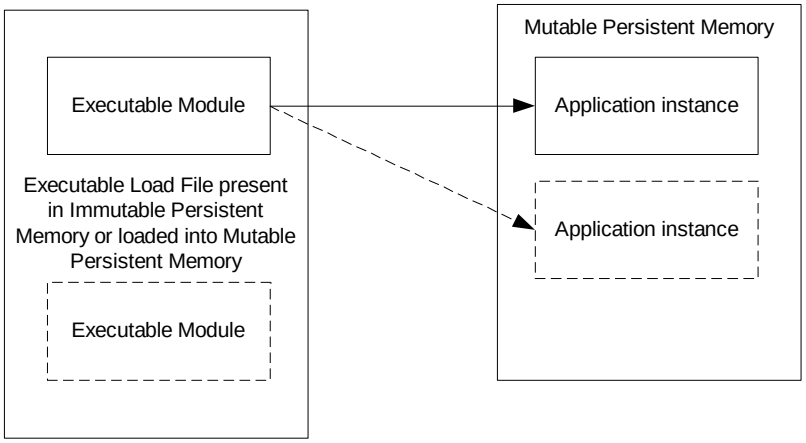
All Card Content, as defined in this specification, is first available on the card in the form of an Executable Load File. An Executable Load File can either exist in:

- Immutable Persistent Memory in which case it is loaded during the manufacturing stage and cannot be altered (except being disabled); or
- Mutable Persistent Memory in which case it can be loaded, or removed during Pre-Issuance or Post-Issuance.

Each Executable Load File may contain one or multiple Executable Modules, being application code. The installation of an Application creates an instance from an Executable Module plus possibly Application data within Mutable Persistent Memory. Any Application instance and its related data can be removed. A GlobalPlatform card is intended to support multiple Executable Load Files and multiple Executable Modules and as such multiple Applications may co-exist on a GlobalPlatform card. Note that the foregoing description assumes that Executable Modules will be present in the Executable Load File; however, their presence is optional and depends on the requirements of the Runtime Environment.

The following figure represents the relationship between an Executable Load File, an Executable Module (in the case where Executable Modules are present) and an Application.

Figure 3-2: Card Content Relationships



3.8 Card Manager

The Card Manager, as the central administrator of the card, assumes multiple responsibilities.

The Card Manager can be viewed as three entities:

- The GlobalPlatform Environment (OPEN)
- The Issuer Security Domain
- Cardholder Verification Method Services

4 Security Architecture

Well-designed security architectures are crucial to protecting the structure and function of cards within the GlobalPlatform system.

This chapter outlines:

- The security goals behind the architecture
- The specific responsibilities of the Card Issuer as the owner of the card
- The Application Providers as the owners of the Applications
- The Controlling Authority
- The security requirements for the on-card components
- The cryptographic support provided by GlobalPlatform

4.1 Goals

The primary goal of the GlobalPlatform is to ensure the security and integrity of the card's components for the life of the card. These components are:

- The runtime environment
- The OPEN
- The Issuer Security Domain
- Supplementary Security Domains
- The Applications

To ensure card security and integrity, the GlobalPlatform is designed to support a range of secure mechanisms for:

- Data integrity
- Resource availability
- Confidentiality
- Authentication

The choice of security policy and cryptography is assumed to be industry and product specific.

Because the cards are only part of a larger card system involving multiple parties and off-card components, the GlobalPlatform also relies upon non-cryptographic, procedural means of protection, such as code testing and verification, physical security, and secure key handling. However, these aspects are out of scope for this card specification.

4.2 Security Responsibilities and Requirements

4.2.1 Card Issuer's Security Responsibilities

The Card Issuer is responsible for:

- Generating and loading the Issuer Security Domain keys
- Enforcing standards and policies for Application Providers governing all aspects of Applications to be provided to the Card Issuer or operated on the Card Issuer's cards
- Working with Application Providers to create and initialize Security Domains other than the Issuer Security Domain
- Determining policy with regards to card and Application Life Cycle management, velocity checking levels, privileges, and other security parameters
- Managing the application code loading and installing both on a Pre-Issuance and Post-Issuance basis
- Cryptographically authorizing load, install, and extradition to be performed by Application Providers

4.2.2 Application Provider's Security Responsibilities

The Application Provider is responsible for:

- Generating the keys for its own Security Domains or obtaining Security Domain keys from a trusted third party
- Working with the Card Issuer to load generated keys into the Application Provider's Security Domain
- Providing applications that meet the Card Issuer's security standards and policies
- Providing load file data block signatures according to its own security policy for integrity and source authenticity
- Obtaining pre-authorization for load, install, and extradition from the Card Issuer
- Returning receipts for load, install, delete, and extradition, according to the Card Issuer's policy

4.2.3 Controlling Authority's Security Responsibilities

A Controlling Authority is responsible for:

- Generating the keys for its own Security Domain or obtaining Security Domain keys from a trusted third party
- Working with the Card Issuer to load generated keys into the Controlling Authority's Security Domain
- Providing signatures and/or certificates to other off-card entities according to its own security policy

4.2.4 On-Card Components' Security Requirements

4.2.4.1 Runtime Environment Security Requirements

The runtime environment is responsible for:

- Providing an interface to all Applications that ensures that the runtime environment security mechanisms cannot be bypassed, deactivated, corrupted, or otherwise circumvented
- Performing secure memory management to ensure that:
  - Each application's code and data (including transient session data) as well as the runtime environment itself and its data (including transient session data) is protected from unauthorized access from within the card.
  - When more than one logical channel is supported, each concurrently selected Application's code and data (including transient session data) as well as the runtime environment itself and its data (including transient session data) is protected from unauthorized access from within the card.
  - The previous content of the memory is not accessible when that memory is reused.
  - The memory recovery process is secure and consistent in case of a loss of power or withdrawal of the card from the card reader while an operation is in progress.
- Providing communication services with off-card entities that ensures the proper transmission (according to the specific communication protocol rules) of unaltered command and response messages

(See the appropriate runtime environment documentation for more details).

4.2.4.2 Trusted Framework Requirements

Each Trusted Framework present on the card shall:

- Check the application access rules of the inter-acting Applications according to their respective privileges.
- Enforce the Trusted Framework security rules for inter-application communication, including the rules; defined in Appendix G.
- Ensure that incoming messages are properly routed unaltered to their intended destinations.
- Ensure that any response messages are properly returned unaltered (except for any cryptographic protection) to the original receiver of the incoming message.

4.2.4.3 OPEN Security Requirements

The OPEN shall:

- Provide an interface to all Applications that ensures that the GlobalPlatform security mechanism cannot be bypassed, deactivated, corrupted, or otherwise circumvented.
- Check application access rules according to the Applications' privileges.
- Manage card and Application Life Cycle (see Chapter 5 – *Life Cycle Models*).
- Ensure that the Card Content changes are authorized by the Card Issuer.
- Ensure that application code has been signed by the Verification Authority represented on the card (if any).

- Ensure that application code has been signed by Application Providers represented on the card, if required.

4.2.4.4 Security Domain Security Requirements

Security Domains enforce the security policies of their off-card Security Domain Provider.

When applicable a Security Domain shall:

- Communicate with off-card entities in accordance with its Security Domain Provider's security policy in Pre-Issuance and Post-Issuance.
- Manage on-card data securely.
- Provide cryptographic protection services for its own Applications during their personalization and optionally during their subsequent operation.
- Request the OPEN to load, install, extradite, and delete card content.
- Return to the off-card entity any receipt for load, install, extradition, and delete.
- Verify the authorization for Card Content changes initiated by an off-card authority.
- Generate receipts for load, install, extradition, and delete.
- Verify the load file data block signature when requested by the OPEN.

4.2.4.5 Global Services Application Security Requirements

A Global Services Application shall:

- Be able to provide services to other Applications, such as CVM services.
- Hold the Global Services application-related data securely.
- Perform internal security measures as required by the service.

4.2.4.6 Application Security Requirements

Applications should:

- Expose only data and resources that are necessary for proper application functionality.
- Perform internal security measures required by the Application Provider.

4.2.5 Back-End System Security Requirements

Despite the best efforts of the card and the loading processes to provide a stable and secure environment, these components alone cannot ensure total security. The back-end systems (multiple back-end systems may exist for a single card), which communicate with the cards, perform the verifications, and manage the off-card key databases, also shall be trusted. Responsible personnel, secure operating systems, system security policies, and audit procedures are all essential components that secure the back-end systems. These requirements are beyond the scope of this Specification. Information on GlobalPlatform's off-card requirements relating to card management can be found in the GlobalPlatform Key Management System Functional Requirements ([KMS Req]), GlobalPlatform Smart Card Management System Functional Requirements ([SCMS Req]), and GlobalPlatform Messaging Specification ([Messaging]).

4.3 Cryptographic Support

One of the major requirements for a GlobalPlatform card is the ability to provide a minimum level of cryptographic functionality. This cryptography is, for example, used for the generation of signatures, and is available for use by the Applications present on the card.

The Issuer Security Domain shall implement one Secure Channel Protocol. A Security Domain other than the Issuer Security Domain shall implement [at least] one Secure Channel Protocol. A GlobalPlatform card should support symmetric cryptography such as the Data Encryption Standard (DES) algorithm or the Advanced Encryption Standard (AES) algorithm. A GlobalPlatform card may also support asymmetric cryptography such as the Rivest / Shamir / Adleman (RSA) algorithm or Elliptic Curve Cryptography (ECC).

The following cryptographic services are described in this section:

- Integrity and authentication
- Secure messaging

When present, services to encrypt and decrypt any pattern of data using these algorithms shall be available to Applications.

It is the responsibility of the Card Issuer and/or Controlling Authorities to set up the appropriate off-card procedures to comply with the governmental restrictions upon cryptography. Features to disable or restrict cryptography usage by Applications on a card are beyond the scope of this Specification.

4.3.1 Secure Card Content Management

The concepts of integrity and authentication represent an additional value associated with a message or a block of data.

The purpose of this additional value is to provide a method of verifying the source and/or the integrity of particular block of code or data.

The following describes the different usages of integrity and authentication for Card Content management in this Specification.

4.3.1.1 Load File Data Block Hash

The Load File Data Block Hash is intended to verify the integrity of a complete Load File Data Block when loaded to a GlobalPlatform card.

The Load File Data Block Hash is used in the computation of:

- The Load File Data Block Signature (see section 4.3.1.2 – *Load File Data Block Signature*)
- The Load Token (see section 4.3.1.3 – *Delegated Management Tokens*)

4.3.1.2 Load File Data Block Signature (DAP)

The Load File Data Block Signature is an authentication value generated by an off-card entity (an Application Provider or a Verification Authority). This is the signature of the Load File Data Block Hash and is included in the DAP Block of the Load File. One or more DAP Blocks may be included in a Load File.

When present during the loading of a Load File to the card, each signature shall be verified by the appropriate Security Domain. The verification operation is referred to as Data Authentication Pattern (DAP) Verification.

4.3.1.3 Delegated Management Tokens

Delegated Management Tokens are signatures of one or more Delegated Management functions (loading, installing, extraditing and deleting) generated by the Card Issuer and used to provide the Card Issuer the control over these Card Content changes. Tokens shall be verified by the appropriate Security Domain.

4.3.1.4 Receipts

The appropriate Security Domain may generate Receipts during Delegated Management. A Receipt is proof that an Application Provider has modified the Card Content.

4.3.2 Secure Communication

A GlobalPlatform card may provide security services related to information exchanged between the card and an off-card entity. The security level of the communication with an off-card entity does not necessarily apply to each individual message being transmitted but can only apply to the environment and/or context in which messages are transmitted. The concept of the Life Cycle of the card (see section 5.1 – *Card Life Cycle*) may be used to determine the security level of the communication between the card and an off-card entity.

The choice of cryptographic algorithms for secure communication is assumed to be industry and product specific.

A GlobalPlatform card offers the following security services associated with messages and defined within a Secure Channel Protocol (see Chapter 10 – *Secure Communication*):

- **Entity authentication** – In which the card or the off-card entity proves its authenticity to the other entity through a cryptographic exchange
- **Integrity and authentication** – In which the receiving entity (the card or off-card entity) ensures that the data being received from the sending entity (respectively the off-card entity or card) actually came from an authenticated entity in the correct sequence and has not been altered
- **Confidentiality** – In which data being transmitted from the sending entity (the off-card entity or card) to the receiving entity (respectively the card or off-card entity) is not viewable by an unauthenticated entity

Authentication of the off-card entity combined with the card Life Cycle State allows the card to assume its environment and/or context.

Part III
Implementation

5 Life Cycle Models

The GlobalPlatform defines Life Cycle models to control the functionality and security of the following GlobalPlatform components:

- Card
- Executable Load Files
- Executable Modules
- Security Domains
- Applications

The OPEN owns and maintains the Life Cycle information within the GlobalPlatform Registry and manages the requested state transitions.

The Life Cycle models of each component are presented in this chapter.

5.1 Card Life Cycle

The OPEN is responsible for maintaining the overall security and administration of the card and its content. As the OPEN plays this supervisory role over the entire card, its life cycle can be thought of as the life cycle of the card and is referred to as the card Life Cycle in the subsequent sections.

From a GlobalPlatform perspective, the card Life Cycle begins with the state OP_READY. Although a cards life includes activities prior to the initial card Life Cycle State, these activities are considered card implementation specific and are beyond the scope of this Specification.

The end of the card Life Cycle is the state TERMINATED.

The Issuer Security Domain inherits the card Life Cycle State.

5.1.1 Card Life Cycle States

The following card Life Cycle States shall apply:

- OP_READY
- INITIALIZED
- SECURED
- CARD_LOCKED
- TERMINATED

The card Life Cycle States OP_READY and INITIALIZED are intended for use during the Pre-Issuance phases of the card's life.

The states SECURED, CARD_LOCKED, and TERMINATED are intended for use during the Post-Issuance phase of the card although it is possible to terminate the card at any point during its life.

5.1.1.1 Card Life Cycle State OP_READY

The state OP_READY indicates that the runtime environment shall be available and the Issuer Security Domain, acting as the selected Application, shall be ready to receive, execute and respond to APDU commands.

The following functionality shall be present when the card is in the state OP_READY:

- The runtime environment shall be ready for execution.
- The OPEN shall be ready for execution.
- The Issuer Security Domain shall be the implicitly selected Application for all card interfaces.
- Executable Load Files that were included in Immutable Persistent Memory shall be registered in the GlobalPlatform Registry.
- An initial key shall be available within the Issuer Security Domain.

The card shall be capable of Card Content changes, the loading of the Load Files containing applications not already present in the card may occur.

The installation, from Executable Load Files, of any Application may occur.

Additionally, if any personalization information is available at this stage, Applications may be personalized.

The OP_READY state may be used by an off-card entity to perform the following actions:

- Supplementary Security Domains may be loaded and/or installed.
- The Security Domain keys may be inserted in order to maintain a cryptographic key separation from the Issuer Security Domain keys.

5.1.1.2 Card Life Cycle State INITIALIZED

The state INITIALIZED is an administrative card production state. The state transition from OP_READY to INITIALIZED is irreversible. Its functionality is beyond the scope of this Specification. This state may be used to indicate that some initial data has been populated (e.g. Issuer Security Domain keys and/or data) but that the card is not yet ready to be issued to the Cardholder.

5.1.1.3 Card Life Cycle State SECURED

The state SECURED is the intended operating card Life Cycle State in Post-Issuance. This state may be used by Security Domains and Applications to enforce their respective security policies. The state transition from INITIALIZED to SECURED is irreversible.

The SECURED state should be used to indicate to off-card entities that the Issuer Security Domain contains all necessary keys and security elements for full functionality.

5.1.1.4 Card Life Cycle State CARD_LOCKED

The card Life Cycle state CARD_LOCKED is present to provide the capability to disable the selection of Security Domain and Applications. The card Life Cycle state transition from SECURED to CARD_LOCKED is reversible.

Setting the card to this state means that the card shall only allow selection of the application with the Final Application privilege.

Card Content changes including any type of data management (specifically Security Domain keys and data) are not allowed in this state.

Either the OPEN, or a Security Domain with Card Lock privilege, or an Application with Card Lock privilege (see section 6.6 – *Privileges*), may initiate the transition from the state SECURED to the state CARD_LOCKED.

5.1.1.5 Card Life Cycle State TERMINATED

The state TERMINATED signals the end of the card Life Cycle and the card. The state transition from any other state to TERMINATED is irreversible.

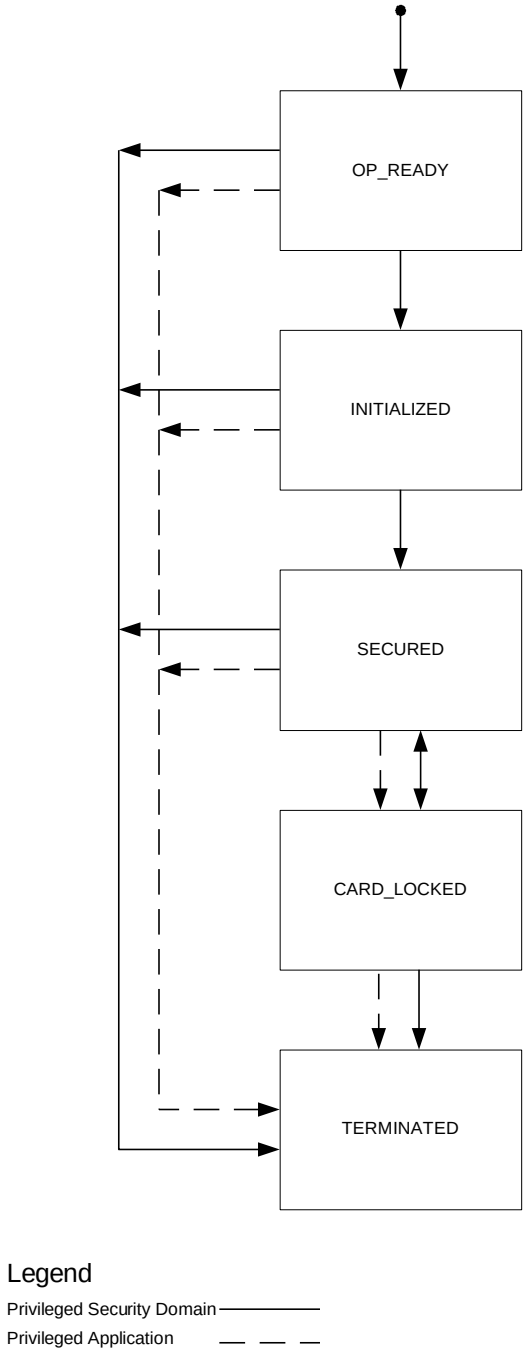
The state TERMINATED shall be used to permanently disable all card functionality with respect to any card content management and any life cycle changes. This card state is intended as a mechanism for an Application to logically 'destroy' the card for such reasons as the detection of a severe security threat or expiration of the card. If a Security Domain has the Final Application privilege only the GET DATA command shall be processed, all other commands defined in this specification shall be disabled and shall return an error. If an application has the Final Application privilege its command processing is subject to issuer policy.

The OPEN itself, or a Security Domain with Card Terminate privilege, or an Application with Card Terminate privilege (see section 6.6 – *Privileges*), may initiate the transition from any of the previous states to the state TERMINATED.

5.1.2 Card Life Cycle State Transitions

The following figure illustrates the state transition diagram for the card Life Cycle. This can typically be viewed as a sequential process with certain possibilities for reversing a state transition or skipping states.

Figure 5-1: Card Life Cycle State Transitions



5.2 Executable Load File/ Executable Module Life Cycle

An Executable Load File is the actual on-card container of one or more application's executable code (Executable Modules). It may reside in Immutable Persistent Memory or may be created in Mutable Persistent Memory as the resulting image of a Load File Data Block. The format in which the Executable Load File is stored on the card is beyond the scope of this Specification.

The OPEN owns and maintains the Executable Load File Life Cycle information within the GlobalPlatform Registry.

5.2.1 Executable Load File Life Cycle

The Executable Load File Life Cycle can only have one state.

5.2.1.1 Executable Load Life Cycle LOADED

The OPEN shall consider all Executable Load Files present in the card in Immutable Persistent Memory or Mutable Persistent Memory to be in the state LOADED. An Executable Load File transferred to the card through a Load File shall become an entry in the GlobalPlatform Registry following the successful completion of the load process. Executable Load Files present in Immutable Persistent Memory shall automatically have entries within the GlobalPlatform Registry and initially be associated with the Issuer's Security Domain.

5.2.1.2 Executable Load File Deletion

The OPEN may receive a request to delete an Executable Load File. If the Executable Load File cannot be physically deleted (e.g., because it is stored in Immutable Persistent Memory), the following behavior shall apply except that the actual space cannot be reclaimed.

The space previously used to store a physically deleted Executable Load File is reclaimed and may be reused. The entries within the GlobalPlatform Registry of the Executable Load File and each Executable Module within the Executable Load File shall no longer be available, and the OPEN is not required to maintain a record of the deleted Executable Load File's or Executable Module's previous existence.

If the received request is also intended to delete each of the Applications instantiated from the Executable Modules within this Executable Load File, then for each of these Applications the behavior described in section 5.3.1.4 – *Application Deletion* or section 5.3.2.5 – *Security Domain Deletion* shall occur.

5.2.2 Executable Module Life Cycle

The Executable Module Life Cycle is linked to the Executable Load File Life Cycle.

5.3 Application and Security Domain Life Cycle

The Life Cycle of the Application or Security Domain begins when the application is instantiated from an Executable Module. The Life Cycle reflects states that are controlled by the OPEN and states that are controlled directly by the Application.

The Application becomes an entry in the GlobalPlatform Registry and the OPEN sets the Application Life Cycle State to the initial state of INSTALLED during the Application installation process. The OPEN is also responsible for making the Application available for selection by setting its Life Cycle State to SELECTABLE upon request during the Application installation process.

Once an Application or Security Domain is available for selection, it takes control of managing its own Life Cycle. The definition of these state transitions is Application or Security Domain dependent and not controlled by the OPEN.

At any point in the Application or Security Domain Life Cycle, the OPEN may take control for security protection by setting the Life Cycle State to LOCKED. The OPEN also controls the deletion of an Application from the card.

5.3.1 Application Life Cycle States

This Specification defines the following Application Life Cycle States:

INSTALLED

SELECTABLE

LOCKED

In addition to these Application Life Cycle States, the Application may define its own Application dependent states.

Once the Application reaches the SELECTABLE state, it is responsible for managing the next steps of its own Life Cycle. It may use any Application specific states as long as these do not conflict with the states already defined by GlobalPlatform. The OPEN may not perform these transitions without instruction from the Application and the Application is responsible for defining state transitions and ensuring that these transitioning rules are respected.

5.3.1.1 Application Life Cycle State INSTALLED

The state INSTALLED means that the Application executable code has been properly linked and that any necessary memory allocation has taken place. The Application becomes an entry in the GlobalPlatform Registry and this entry is accessible to off-card entities authenticated by the associated Security Domain. The Application is not yet selectable. The installation process is not intended to incorporate personalization of the Application, which may occur as a separate step.

5.3.1.2 Application Life Cycle State SELECTABLE

The state SELECTABLE means that the Application is able to receive commands from off-card entities. The state transition from INSTALLED to SELECTABLE is irreversible. The Application shall be properly installed and functional before it may be set to the state SELECTABLE. The transition to SELECTABLE may be combined with the Application installation process.

The behavior of the Application in the state SELECTABLE is beyond the scope of this Specification.

5.3.1.3 Application Life Cycle State LOCKED

The OPEN, the Application itself, the Application's associated Security Domain, an Application with the Global Lock privilege, or a Security Domain with the Global Lock privilege uses the state LOCKED as a security management control to prevent the selection, and therefore the execution, of the Application.

If the OPEN detects a threat from within the card and determines that the threat is associated with a particular Application, that Application may be prevented from further selection by the OPEN setting the state to LOCKED.

Alternatively, the off-card entity may determine that a particular Application on the card needs to be locked for a business or security reason and may initiate the Application Life Cycle transition via the OPEN.

Once the state is LOCKED, only the Application's associated Security Domain, an Application with Global Lock privilege or a Security Domain with Global Lock privilege is allowed to unlock the Application. The OPEN shall ensure that the Application Life Cycle returns to its previous state.

5.3.1.4 Application Deletion

At any point in the Application Life Cycle, the OPEN may receive a request to delete an Application.

The space previously used to store a physically deleted Application is reclaimed and may be reused. The entry within the GlobalPlatform Registry shall no longer be available, and the OPEN is not required to maintain a record of the deleted Application's previous existence.

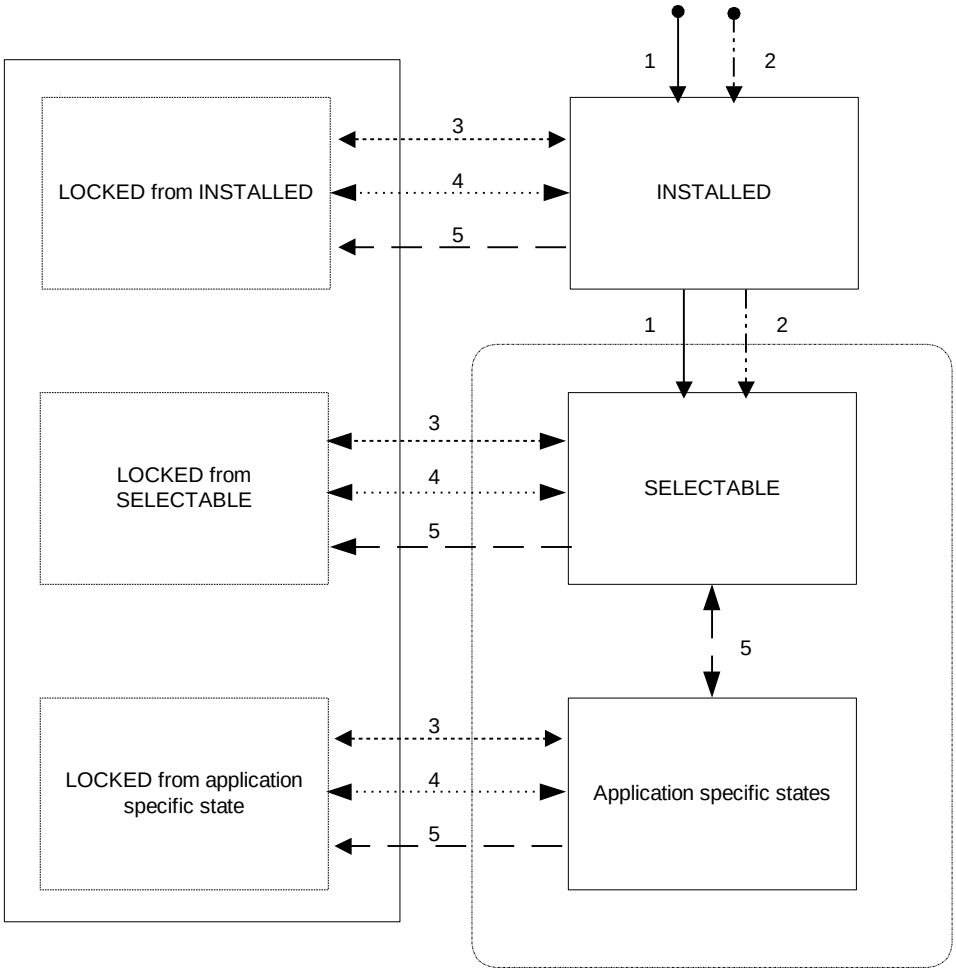
5.3.1.5 Application Specific Life Cycle States

These states are Application specific. The behavior of the Application, while in these states, is determined by the Application itself and is beyond the scope of this Specification. The OPEN does not enforce any control on Application specific Life Cycle State transitions.

5.3.1.6 Application Life Cycle State Transitions

The following figure illustrates the state transition diagram for the Application Life Cycle. This can typically be viewed as a sequential process with certain possibilities for reversing a state transition or skipping states.

Figure 5-2: Application Life Cycle State Transitions



Legend

1. A Security Domain with Authorized Management privilege
2. A Security Domain with Delegated Management privilege
3. The associated Security Domain
4. A Security Domain or Application with Global Lock privilege
5. The Application itself

5.3.2 Security Domain Life Cycle States

This Specification defines the following states applicable to a Security Domain:

1. INSTALLED
2. SELECTABLE
3. PERSONALIZED
4. LOCKED

There are no proprietary Security Domain Life Cycle States.

5.3.2.1 Security Domain Life Cycle State INSTALLED

The state INSTALLED means that the Security Domain becomes an entry in the GlobalPlatform Registry and this entry is accessible to off-card entities authenticated by the associated Security Domain. The Security Domain is not yet available for selection. It cannot be associated with Executable Load Files or Applications yet and therefore its Security Domain services are not available to Applications.

5.3.2.2 Security Domain Life Cycle State SELECTABLE

The state SELECTABLE means that the Security Domain is able to receive commands (specifically personalization commands) from off-card entities. As they still do not have keys, the Security Domains cannot be associated with Executable Load Files or Applications and therefore their services are not available to Applications when they are in this state. The state transition from INSTALLED to SELECTABLE is irreversible. The transition to SELECTABLE may be combined with the Security Domain installation process.

5.3.2.3 Security Domain Life Cycle State PERSONALIZED

The definition of what is required for a Security Domain to transition to the state PERSONALIZED is Security Domain dependent but is intended to indicate that the Security Domain has all the necessary personalization data and keys for full runtime functionality (i.e. usable in its intended environment). The transition from SELECTABLE to PERSONALIZED (initiated by the Security Domain itself) is irreversible.

In the state PERSONALIZED, the Security Domain may be associated with Applications and its services become available to these associated Applications.

5.3.2.4 Security Domain Life Cycle State LOCKED

The OPEN, the Security Domain itself, the Security Domain's associated Security Domain (if any), an Application with the Global Lock privilege or a Security Domain with the Global Lock privilege uses the state LOCKED as a security management control to prevent the selection of the Security Domain.

If the OPEN detects a threat from within the card and determines that the threat is associated with a particular Security Domain, that Security Domain may be prevented from further selection by the OPEN setting the Security Domain's Life Cycle State to LOCKED.

Alternatively, the off-card entity may determine that a particular Security Domain on the card needs to be locked for a business or security reason and may initiate the state transition via the OPEN.

In this state, the Security Domain is prevented from being used for Delegated Management if applicable. Locking a Security Domain prevents this Security Domain from being associated with new Executable Load Files or Applications. In this state DAP verification, extradition and access to that Security Domain's services shall fail. In summary, if a Security Domain is in the lifecycle state LOCKED, it shall reject all received commands.

Once the Life Cycle State is LOCKED, only the Security Domain's associated Security Domain (if any), an Application with Global Lock privilege or a Security Domain with Global Lock privilege is allowed to unlock the Security Domain. The OPEN shall ensure that the Security Domain's Life Cycle returns to its previous state.

5.3.2.5 Security Domain Deletion

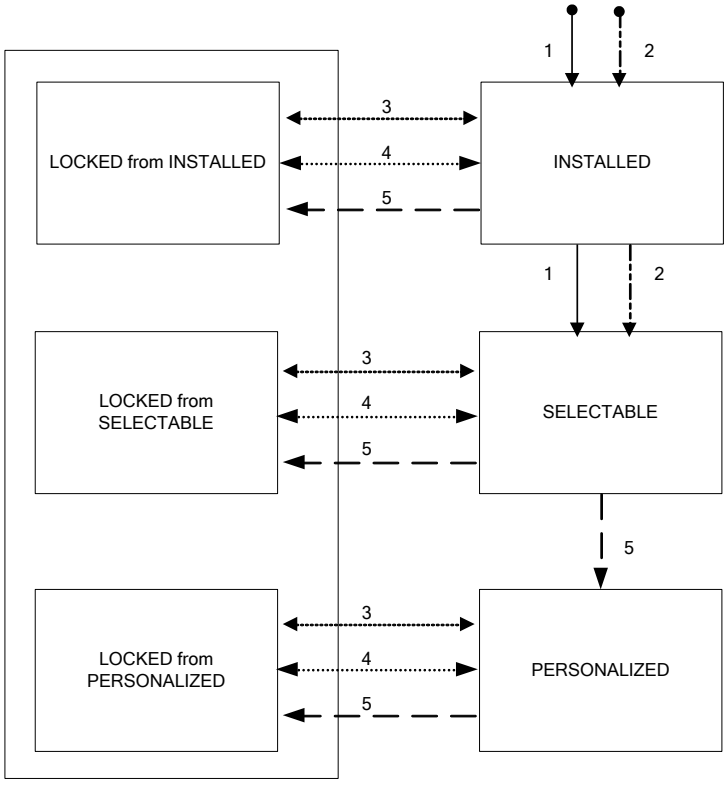
At any point in the Security Domain Life Cycle, the OPEN may receive a request to delete a Security Domain.

The space previously used to store a physically deleted Security Domain is reclaimed and may be reused. The entry within the GlobalPlatform Registry shall no longer be available, and the OPEN is not required to maintain a record of the deleted Security Domain's previous existence.

5.3.2.6 Security Domain Life Cycle State Transitions

The following figure illustrates the state transition diagram for the Security Domain Life Cycle. This can typically be viewed as a sequential process with certain possibilities for reversing a state transition or skipping states.

Figure 5-3: Security Domain Life Cycle State Transitions



Legend

1. A Security Domain with Authorized Management privilege
2. A Security Domain with Delegated Management privilege
3. The associated Security Domain
4. A Security Domain or Application with Global Lock privilege
5. The Security Domain itself

5.4 Sample Life Cycle Illustration

This section provides a description of a sample GlobalPlatform card and its Life Cycle transitions from the card's creation to the time it is terminated. It also shows the status of several Executable Load Files, Executable Modules, and Applications and their relationship with the card Life Cycle. Figure 5-4 illustrates these sample Life Cycle States:

Application A: Application code is present as an Executable Module within an Executable Load File in Mutable Persistent Memory when the card is manufactured. It is installed in an implementation specific manner. It is used throughout the card's life until the card is terminated and as long as the card is not in a CARD_LOCKED state.

Application B: Application code is present as an Executable Module within an Executable Load File in Immutable Persistent Memory when the chip is manufactured. It is installed prior to the card being initialized. Application B, along with its Executable Load File, is deleted some time during the card's life before the card is terminated. Because the Executable Load File for Application B is stored in Immutable Persistent Memory, it cannot be physically deleted from the card.

Application C: Application code is present as an Executable Module within an Executable Load File and is loaded in an implementation specific manner. The Application is installed in Post-Issuance while the card is in the Life Cycle State SECURED. The Application is used for some time and then deleted along with its Executable Load File before the card is terminated. Because the Application and its Executable Load File are stored in Mutable Persistent Memory, the Application and associated Executable Load File, along with all its Executable Modules, are purged from Mutable Persistent Memory and the memory space reclaimed for reuse.

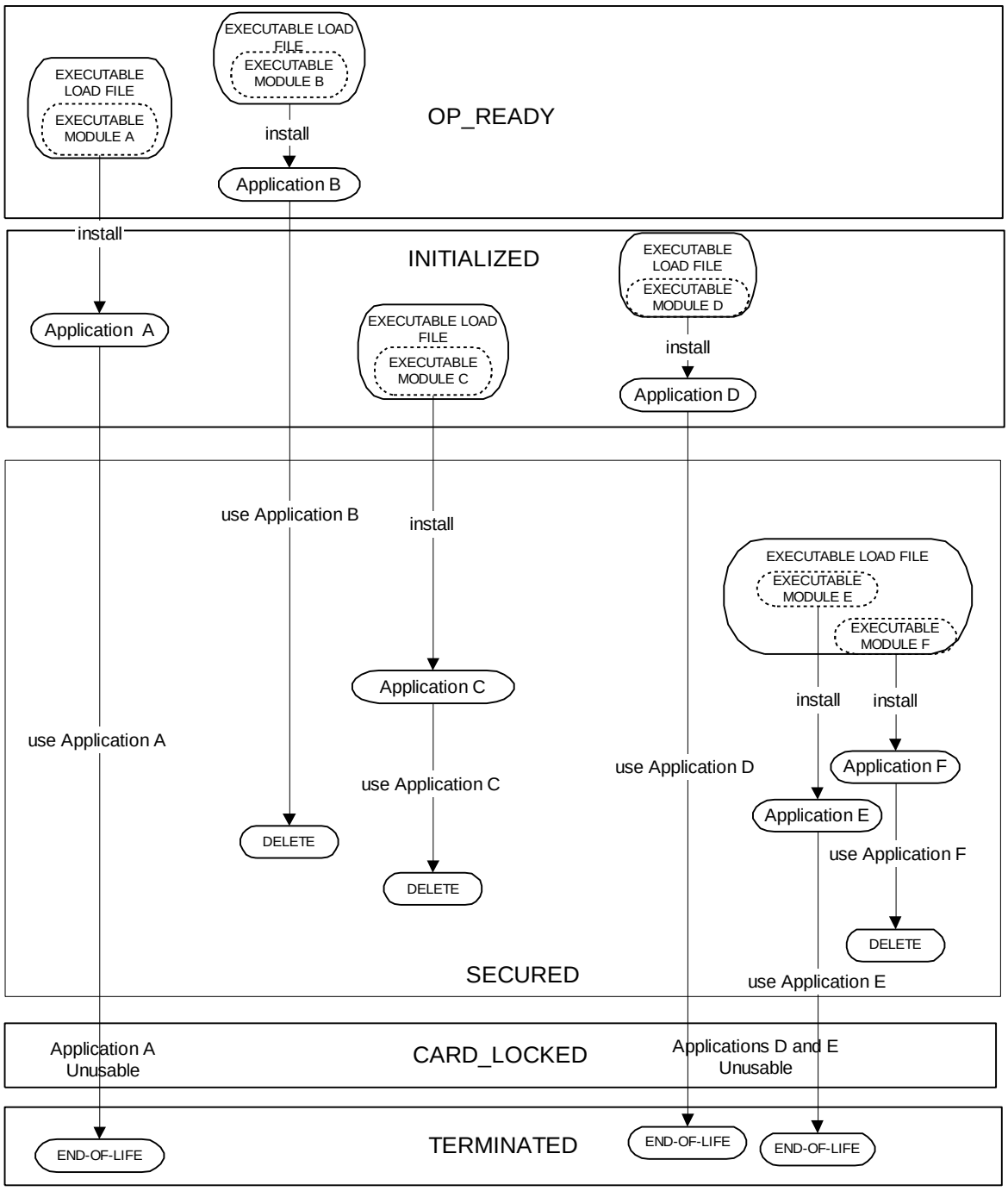
Application D: Application code is present as an Executable Module within an Executable Load File and is loaded in an implementation specific manner. It is used during the full lifetime of the card until the card is terminated and as long as the card Life Cycle State is not CARD_LOCKED.

Application E: Application code is present as an Executable Module within an Executable Load File that is loaded and installed in Post-Issuance while in the card Life Cycle State SECURED. Application E is used until the card is terminated and as long as the card Life Cycle State is not CARD_LOCKED.

Application F: Application code is present as an Executable Module within the same Executable Load File as Application E. It is loaded and installed in Post-Issuance while in the card Life Cycle State SECURED. Application F is deleted some time during the card's lifetime.

Note that the following diagram is not intended to be a comprehensive description of what is permitted in each card Life Cycle State.

Figure 5-4: Sample Card Life Cycle and Application Life Cycles



6 GlobalPlatform Environment (OPEN)

6.1 Overview

The GlobalPlatform Environment (OPEN) supports the following functions if the underlying runtime environment does not support them:

- **Command Dispatch**

  Application and Security Domain selection

  (Optional) Logical channel management

  Command dispatching

- **Card Content Management**

  Content verification

  Content loading

  Content installation

  Content removal

  Access control rules for card content management

- **Security Management**

  Security Domain locking and unlocking

  Application locking and unlocking

  Card locking and unlocking

  Card termination

  Privilege usage

  Security Domain privilege usage

  Tracing and event logging

- **GlobalPlatform Trusted Framework**

  Secure inter-application communication

The OPEN architecture may be illustrated as a collection of system functions built upon a GlobalPlatform Registry. The GlobalPlatform Registry is a data store required to support the various system functions of the GlobalPlatform.

The following are some examples of the use of the GlobalPlatform Registry by the OPEN.

Command Dispatch:

- Determine if an Application or Security Domain is present and available to respond to a SELECT command.
- When supported, manage logical channels.
- Dispatch commands to the selected Application for command processing.

Card Content Management:

- Store state and management information about newly loaded Executable Load Files, Executable Modules, and installed Applications.
- Store the Security Domain to be associated with Executable Load Files being loaded.
- Store the privileges and associated Security Domains of the Applications being installed.
- Identify an Application's associated Security Domain to provide access for that Application to its Security Domain services.

Security Management:

- Allow the audit of Card Content by retrieving status information related to any Application present on the card.
- Verify the integrity and request verification of the authenticity for Executable Load Files.
- Verify that resource restrictions are respected during loading and installing of new content and during Application runtime execution.
- Verify an Application's or a Security Domain's accessibility to functionality that requires privileges.

6.2 OPEN Services

Section A.1 – *GlobalPlatform on a Java Card* provides the Java Card™ implementation of the following interfaces.

Section A.2 – *GlobalPlatform on MULTOS*™ provides the MULTOS™ implementation of the following interfaces.

Applications and Security Domains may access and/or modify some content known or managed by the OPEN. Depending on the relevant privileges of the requesting entity, the following services shall be provided by the OPEN:

- Retrieving the Application's own Life Cycle State stored by the OPEN in the GlobalPlatform Registry
- Retrieving the card Life Cycle State
- Obtaining access to the services of the Security Domain associated with the Application
- Transitioning the card Life Cycle State to CARD_LOCKED
- For contact cards according to [ISO 7816-4] and for contactless cards Type A according to [ISO 14443-3], setting the content of the historical bytes
- Transitioning the Application's own Application Life Cycle State stored by the OPEN in the GlobalPlatform Registry
- Transitioning the card Life Cycle State to TERMINATED
- Obtaining access to GlobalPlatform Registry information
- Obtaining access to Global Services Applications

6.3 Command Dispatch

The commands received by a GlobalPlatform card shall be processed by the OPEN or dispatched to the selected Application for processing.

The SELECT [by name] command is processed by the OPEN.

One option of making the Issuer Security Domain the selected Application, is to specify its AID in a SELECT command with the [first or only occurrence] option set. As another option for making the Issuer Security Domain the selected Application, the SELECT command could contain no data in which case the AID of the Issuer Security Domain would be discovered by the off-card entity in the response to the SELECT command.

The processing of the MANAGE CHANNEL command is dependent on the capabilities of the card:

- If the card is aware of logical channels but only supports the Basic Logical Channel, the OPEN shall respond with the appropriate error.
- If the card is aware of logical channels and supports at least one Supplementary Logical Channel, the OPEN shall process the command.
- If the card only supports the Basic Logical Channel and has no concept of logical channel support, the command shall be dispatched to the selected Application for processing.

Any other type of command received shall be dispatched to the currently selected Application.

Commands are either received on the Basic Logical Channel (logical channel number zero) or on a Supplementary Logical Channel (logical channel number other than zero). In compliance with [ISO 7816-4], logical channel information shall be indicated in the class byte of the APDU command header. For all commands, if the command indicates a Supplementary Logical Channel that is not opened then:

- If the card only supports the Basic Logical Channel and has no concept of logical channel support, the command shall be dispatched to the selected Application for processing.
- If the card is aware of logical channels, the OPEN shall respond with the appropriate error. (This requirement may exclude the SELECT command, if the card supports opening of a logical channel using the SELECT command.)

For commands that are dispatched to an Application, it is the responsibility of the Application to correctly reject commands that it does not recognize, expect or cannot process.

The way in which an Application exhibits its multi-selection capabilities can vary according to the underlying platform and is beyond the scope of this Specification.

The Issuer Security Domain shall have no multi-selection restrictions on cards that support multiple logical channels; i.e. it shall be capable of being selected across multiple logical channels.

6.4 Logical Channels and Application Selection

6.4.1 Implicit Selection Assignment

The following rules apply to the assignment of implicit selection to an Application:

- The Issuer Security Domain is by default the implicitly selectable Application on all logical channels of all card I/O interfaces supported by the card. Implicit selection on a specific logical channel of a specific card I/O interface may be assigned only if the Issuer Security Domain is the implicitly selectable Application on that logical channel of that card I/O interface and no other Locked Application is registered as implicitly selectable for the same logical channel and card I/O interface (see section 9.6.2 – *Application Locking and Unlocking*).
- An Application installed or made selectable with a specific Implicit Selection parameter is registered in the GlobalPlatform Registry as the implicitly selectable Application on the logical channel(s) of the card I/O interface(s) indicated in the parameter if no other Application (other than the Issuer Security Domain) is already registered as implicitly selectable on that logical channel(s) of that card I/O interface(s).
- An Application installed and made selectable with the Card Reset privilege and without Implicit Selection parameter is registered in the GlobalPlatform Registry as the implicitly selectable Application on the Basic Logical Channel for all card I/O interfaces supported by the card if no other Application (other than the Issuer Security Domain) is already registered as implicitly selectable on the Basic Logical Channel of any card I/O interface.
- If an Application implicitly selectable on specific logical channel(s) of specific card I/O interface(s) is deleted, the Issuer Security Domain becomes the implicitly selectable Application on that logical channel(s) of that card I/O interface(s).

The OPEN shall use the implicit selection registration of each Application for controlling the following runtime behavioral requirements:

- Identifying the Application which is implicitly selectable on the Basic Logical Channel of the current card I/O interface during the card reset or activation sequence
- Identifying the Application which is implicitly selectable when opening on the current card I/O interface a new Supplementary Logical Channel from the Basic Logical Channel

6.4.2 Basic Logical Channel

The Basic Logical Channel is the permanently available interface to a GlobalPlatform card. This Basic Logical Channel shall be supported by the OPEN.

6.4.2.1 Application Selection on Basic Logical Channel

The OPEN shall support Application selection on the Basic Logical Channel via two processes:

- Implicit Selection following the card reset (see [ISO 7816-3] for contact cards) or activation sequence (see [ISO 14443-3] for contactless cards)
- Explicit Selection through the SELECT [by name] command

The OPEN may also support additional selection processes.

Partial AID selection as defined in section 6.4.2.1.2 – *Explicit Selection*, shall be supported. (Partial AID selection does not require knowledge of the full AID by the off-card entity.) As multiple Applications on the card may have the same partial AID, it is required that a method exists to select all Applications matching the partial AID.

6.4.2.1.1 Implicit Selection on Basic Logical Channel

Once the card session has been established (for contact cards according to [ISO 7816-4] after Answer-to-Reset, or after the activation sequence for contactless cards according to [ISO 14443-3]), and before the first command is issued to the card, the Application defined as implicitly selectable on the Basic Logical Channel and for that card I/O interface shall become the selected Application on the Basic Logical Channel for that card I/O interface.

Runtime Behavior

The following requirements apply for the OPEN for the implicit Application selection process:

- If the card is in the Life Cycle State CARD_LOCKED or TERMINATED, the Application with the Final Application privilege is the selected Application on the Basic Logical Channel and the OPEN shall not attempt to identify the implicitly selectable Application.
- In all other cases the OPEN shall search the GlobalPlatform Registry for an Application that is marked as implicitly selectable on the Basic Logical Channel for the current card I/O interface (e.g. contact or contactless), and if this Application is not in the Life Cycle State LOCKED, it shall become the selected Application on the Basic Logical Channel. If this is an Application in the Life Cycle State LOCKED, the Application with the Final Application privilege shall become the selected Application on the Basic Logical Channel.

6.4.2.1.2 Explicit Selection on Basic Logical Channel

At any time during a Card Session the OPEN may receive a request to select an Application on the Basic Logical Channel (SELECT [by name] [first or only occurrence] command). The OPEN shall determine if the requested AID matches or partially matches an entry within the GlobalPlatform Registry and whether this entry is selectable.

At any time during a Card Session that has already contained a SELECT [by name] [first or only occurrence] command, the OPEN may receive a request to select a next Application (SELECT [by name] [next occurrence] command) on the Basic Logical Channel. The OPEN shall determine if the requested AID matches or partially matches another entry within the GlobalPlatform Registry and whether this entry is selectable.

For both the SELECT [by name] [first or only occurrence] command and the SELECT [by name] [next occurrence] command, an Application becomes the selected Application on the Basic Logical Channel if all of the following are true:

- The requested AID matches (fully or partially) the Application's AID.
- The Application being selected is in the correct Life Cycle State.
- The Application has no restrictions due to multi-selection, and supports the current card interface.

Runtime Behavior

The following requirements apply to the OPEN in the explicit Application selection (SELECT [by name]) process on the Basic Logical Channel (This behavior does not apply if the card Life Cycle State is TERMINATED):

- In the card Life Cycle State CARD_LOCKED:
  - If the Application being selected has the Final Application privilege, this Application is re-selected and a warning is returned to the off-card entity.
  - If any other Application is being selected, the Application with the Final Application privilege remains selected and an error is returned to the off- card entity.
- If a SELECT [by name] [first or only occurrence] or SELECT [by name] [next occurrence] is received and the data field of the command message is not present, the Issuer Security Domain shall become the currently selected Application and the SELECT command is dispatched to the Issuer Security Domain.
- If a SELECT [by name] [first or only occurrence] is received, the search always begins from the start of the GlobalPlatform Registry.
- If a SELECT [by name] [next occurrence] is received, the search always begins from the entry following the currently selected Application on the Basic Logical Channel in the GlobalPlatform Registry.
- If a full or partial match is found and this Application is in the Life Cycle State INSTALLED, continue searching through the GlobalPlatform Registry for a subsequent full or partial match. If no subsequent full or partial match is found, the OPEN shall return the appropriate error to the off-card entity.
- If a full or partial match is found and this Application is in the Life Cycle State LOCKED, continue searching through the GlobalPlatform Registry for a subsequent full or partial match. In the eventuality that this locked Application is already currently selected on the Basic Logical Channel, the OPEN shall terminate this Application Session. If no subsequent full or partial match is found, the OPEN shall return the appropriate error to the off-card entity.
- If a full or partial match is found and this Application cannot be selected due to a multi-selection restriction or because the Application refuses selection (e.g. because it does not support the current card interface), continue searching through the GlobalPlatform Registry for a subsequent full or partial match. If no subsequent full or partial match is found, the OPEN shall return the appropriate error to the off-card entity.
- If a full or partial match is found and this Application is selectable (i.e. in the correct Life Cycle State and has no multi-selection restrictions), then it shall become the currently selected Application on the Basic Logical Channel and the SELECT [by name] command, whether [first or only occurrence] or [next occurrence], shall be processed according to the requirements of the runtime environment (e.g. dispatched to the Application).
- If no full or partial match is found at all, the currently selected Application on the Basic Logical Channel shall remain the selected Application and

- If the SELECT [by name] command has the [first or only occurrence] parameter set, the SELECT command is dispatched to the Application.
- If the SELECT [by name] command has the [next occurrence] parameter set, the OPEN shall return the appropriate error to the off-card entity.

- In the eventuality that the current Application Session has been terminated and no subsequent full or partial match is found, the OPEN shall make an attempt to select the Application that is marked as implicitly selectable on the Basic Logical Channel for the current card interface.

6.4.2.2 Logical Channel Management on Basic Logical Channel

At any time during a Card Session the OPEN may receive a request on the Basic Logical Channel to either open or close a Supplementary Logical Channel.

If the card only supports the Basic Logical Channel and has no concept of logical channel support, the MANAGE CHANNEL command is dispatched to the currently selected Application. In this case, when a Security Domain is the currently selected Application, the command shall be rejected.

On cards that support logical channels, if a MANAGE CHANNEL [open] is received:

- If an Application is designated as implicitly selectable on the new Supplementary Logical Channel for the current card interface, that Application is implicitly selected on the newly opened Supplementary Logical Channel and runtime behavior requirements apply.
- Otherwise the Application designated as implicitly selectable on the Basic Logical Channel for this card interface is implicitly selected on the newly opened Supplementary Logical Channel and runtime behavior requirements apply.

On cards that support logical channels, if a MANAGE CHANNEL [close] is received, terminate the Application Session currently selected on the Supplementary Logical Channel indicated by the command and then close that logical channel. The Basic Logical Channel can never be closed.

Runtime Behavior

On receipt of a MANAGE CHANNEL [open] command, the following requirements apply:

- If the card is in the Life Cycle State CARD_LOCKED or TERMINATED, return the appropriate error.
- If the number of logical channels supported by the OPEN is not sufficient to open a new Supplementary Logical Channel, return the appropriate error.
- The OPEN shall search the GlobalPlatform Registry for the Application that supports the current card interface and that is marked as implicitly selectable on the new Supplementary Logical Channel (or failing that, on the Basic Logical Channel) and:
  - If this is an Application in the Life Cycle State LOCKED, the Application with the Final Application privilege shall become the selected Application on the Supplementary Logical Channel.
  - If this Application cannot be selected due to a multi-selection restriction, the new logical channel shall not be opened and the OPEN shall return the appropriate error.
  - Otherwise, the Supplementary Logical Channel is opened and this Application shall become the selected Application on the Supplementary Logical Channel.

6.4.2.3 Application Command Dispatch on Basic Logical Channel

Once an Application becomes the selected Application on the Basic Logical Channel, the responsibility for subsequent command dispatching still rests with the OPEN.

Processing SELECT [by name] commands and runtime behavior requirements for OPEN are described in section 6.4.2.1.2 – *Explicit Selection*.

On cards that are aware of logical channels, the MANAGE CHANNEL commands are only processed by the OPEN and are not dispatched to an Application.

All other commands (including the MANAGE CHANNEL commands on cards that are not aware of logical channels or SELECT commands not described in section 6.4.2.1.2 – *Explicit Selection*) are immediately dispatched to the Application currently selected on the Basic Logical Channel. The processing of the command by the Application is beyond the scope of this Specification.

6.4.3 Supplementary Logical Channel

A Supplementary Logical Channel, if supported, allows an Application to be selected simultaneously to the Applications selected on other logical channels.

6.4.3.1 Application Selection on Supplementary Logical Channel

The OPEN shall support Application selection on an available Supplementary Logical Channel via two processes:

- Implicit Selection following a successful MANAGE CHANNEL [open] command
- Explicit Selection through the SELECT [by name] command

The OPEN may also support additional selection processes.

Partial AID selection as defined in section 6.4.3.1.2 – *Explicit Selection*, shall be supported on Supplementary Logical Channels.

6.4.3.1.1 Implicit Selection on Supplementary Logical Channel

Depending on whether a Supplementary Logical Channel is being opened from the Basic Logical Channel or from another Supplementary Logical Channel, the behavior of implicit selection differs.

Refer to section 6.4.2.2 – *Logical Channel Management* for the behavior of implicit selection initiated from the Basic Logical Channel.

Refer to section 6.4.3.2 – *Logical Channel Management* for the behavior of implicit selection initiated from a Supplementary Logical Channel.

6.4.3.1.2 Explicit Selection on Supplementary Logical Channel

At any time on an open Supplementary Logical Channel, the OPEN may receive a request to select an Application on this Supplementary Logical Channel (SELECT [by name] [first or only occurrence] command). The OPEN shall determine if the requested AID matches or partially matches an entry within the GlobalPlatform Registry and whether this entry is selectable.

At any time on an open Supplementary Logical Channel that has already contained a SELECT [by name] [first or only occurrence] command since the Supplementary Logical Channel was last opened, the OPEN may receive a request to select a next Application (SELECT [by name] [next occurrence] command) on this Supplementary Logical Channel. The OPEN shall determine if the requested AID matches or partially matches another entry within the GlobalPlatform Registry and whether this entry is selectable.

For both the SELECT [by name] [first or only occurrence] command and the SELECT [by name] [next occurrence] command, an Application becomes the selected Application on the Supplementary Logical Channel if all of the following are true:

- The requested AID matches (fully or partially) the Application's AID.
- The Application being selected is in the correct Life Cycle State.
- The Application has no restrictions due to multi-selection, and supports the current card interface.

Runtime Behavior

The following requirements apply to the OPEN in the explicit Application selection (SELECT [by name]) process on a Supplementary Logical Channel:

- If the card is in the Life Cycle State CARD_LOCKED or TERMINATED:
  - Close the Supplementary Logical Channel, if currently open.

- Return the appropriate error.

- If a SELECT [by name] [first or only occurrence] or SELECT [by name] [next occurrence] is received and the data field of the command message is not present, the Issuer Security Domain shall become the currently selected Application and the SELECT command is dispatched to the Issuer Security Domain.
- If a SELECT [by name] [first or only occurrence] is received, the search always begins from the start of the GlobalPlatform Registry.
- If a SELECT [by name] [next occurrence] is received, the search always begins from the entry following the currently selected Application on this Supplementary Logical Channel in the GlobalPlatform Registry.
- If a full or partial match is found and this Application is in the Life Cycle State INSTALLED, continue searching through the GlobalPlatform Registry for a subsequent full or partial match. If no subsequent full or partial match is found, the OPEN shall return the appropriate error to the off-card entity.
- If a full or partial match is found and this Application is in the Life Cycle State LOCKED, continue searching through the GlobalPlatform Registry for a subsequent full or partial match. In the eventuality that this locked Application is already currently selected on the same Supplementary Logical Channel, the OPEN shall terminate this Application Session. If no subsequent full or partial match is found, the OPEN shall return the appropriate error to the off-card entity.
- If a full or partial match is found and this Application cannot be selected due to a multi-selection restriction or because the Application refuses selection (e.g. because it does not support the current card interface), continue searching through the GlobalPlatform Registry for a subsequent full or partial match. If no subsequent full or partial match is found, the OPEN shall return the appropriate error to the off-card entity.
- If a full or partial match is found and this Application is selectable (i.e. in the correct Life Cycle State and has no multi-selection restrictions), then it shall become the currently selected Application on this Supplementary Logical Channel and the SELECT [by name] command, whether [first or only occurrence] or [next occurrence], shall be processed according to the requirements of the runtime environment (e.g. dispatched to the Application).
- If no full or partial match is found at all, the currently selected Application on the Supplementary Logical Channel shall remain the selected Application and
  - If the SELECT [by name] command has the [first or only occurrence] parameter set, the SELECT command is dispatched to the Application.
  - If the SELECT [by name] command has the [next occurrence] parameter set, the OPEN shall return the appropriate error to the off-card entity.

6.4.3.2 Logical Channel Management on Supplementary Logical Channel

At any time on an open Supplementary Logical Channel the OPEN may receive a request to either open or close a Supplementary Logical Channel.

If a MANAGE CHANNEL [open] is received and the Application selected on the original Supplementary Logical Channel has no multi-selection restrictions, this Application becomes the selected Application on the newly opened Supplementary Logical Channel.

If a MANAGE CHANNEL [close] is received, terminate the Application Session currently selected on the Supplementary Logical Channel indicated by the command and then close that logical channel. The Basic Logical Channel can never be closed.

Runtime Behavior

On receipt of a MANAGE CHANNEL [open] command, the following requirements apply:

- If the card is in the Life Cycle State CARD_LOCKED or TERMINATED, return the appropriate error.
- If the number of logical channels supported by the OPEN is not sufficient to open a new Supplementary Logical Channel, return the appropriate error.
- If the Application currently selected on the original Supplementary Logical Channel cannot be selected on the new Supplementary Logical Channel due to a multi-selection restriction, the new logical channel shall not be opened and the OPEN shall return the appropriate error.
- Otherwise, the Supplementary Logical Channel indicated by the command is opened and the Application currently selected on the original Supplementary Logical Channel shall become the selected Application on the newly opened Supplementary Logical Channel.

6.4.3.3 Application Command Dispatch on Supplementary Logical Channel

Once an Application becomes the selected Application on a Supplementary Logical Channel, the responsibility for subsequent command dispatching still rests with the OPEN.

Processing SELECT [by name] commands and runtime behavior requirements for OPEN are described in section 6.4.3.1.2 – *Explicit Selection*.

The MANAGE CHANNEL commands are only processed by the OPEN and are not dispatched to an Application.

All other commands (including SELECT commands not described in section 6.4.3.1.2 – *Explicit Selection*) are immediately dispatched to the Application currently selected on the Supplementary Logical Channel. The processing of the command by the Application is beyond the scope of this Specification.

6.5 GlobalPlatform Registry

The GlobalPlatform Registry is used to:

- Store card management information
- Store relevant application management information (e.g., AID, associated Security Domain and Privileges)
- Support card resource management data
- Store Application Life Cycle information
- Store card Life Cycle information
- Track any counters associated with logs

The contents of the GlobalPlatform Registry may be updated in response to:

- An internal OPEN invoked action
- An authorized Application invoked action

All Applications including all Security Domains, and all Executable Load Files, shall have an entry in the GlobalPlatform Registry.

There is no mandatory format for the storage of these data elements. However, format requirements do exist for the handling of the data elements via APDU commands and GlobalPlatform services available to Applications.

6.5.1 Application/Executable Load File/Executable Module Data Elements

6.5.1.1 Application/Executable Load File/Executable Module AID

Each Executable Load File or Executable Module is associated with an AID that shall be unique on the card.

An Application AID may be the same as that of an Executable Module but may not be the same as that of an Executable Load File or the same as another Application already present in the GlobalPlatform Registry.

This AID may be specified in a SELECT command to select the Application. It is not possible to select Executable Load Files or Executable Modules.

6.5.1.2 Application/Executable Load File/Executable Module Life Cycle

The Application Life Cycle State data element contains the current Life Cycle of the Application, Executable Load File or Executable Module.

6.5.1.3 Memory Resource Management Parameters

Resource management data elements contain information about the memory resources that are available to an Application. They are system-specific values and are used as a control mechanism by the OPEN to manage memory resource allocation as described in section 9.7 – *Memory Resource Management*.

When additional resources are requested by an Application, the OPEN shall validate the request against the value of this data element in the GlobalPlatform Registry.

6.5.1.4 Privileges

The Privileges data element indicates the privileges for each Application. Privileges are defined in section 6.6.

6.5.1.5 Implicit Selection Parameter

The implicit selection parameter indicates whether an Application is implicitly selectable on a specific logical channel of a specific card interface; see section 11.1.7 and Table 11-50.

6.5.1.6 Associated Security Domain AID

All Executable Load Files and Applications, including Security Domains, are associated with a Security Domain, whose AID is given in the registry. This AID is the AID of the Security Domain directly associated. A Security Domain also has an associated Security Domain: either another Security Domain or itself, hence describing an association hierarchy of Executable Load Files / Applications and Security Domains.

6.5.1.7 Application Provider ID

The Registry holds the identifier of the Application Provider: the owner of the Application or Executable Load File when explicitly provided during the load or installation process.

An on-card entity may use this information to enforce security policies. Establishing whether an off-card entity is also the owner of the on-card entity being managed is addressed in more detail in section 10.4 – *Entity Authentication*.

6.5.1.8 Service Parameter

A Service Parameter identifies the service offered by the Global Services Application. The Global Services Application may register this service as a unique Global Service as described in section 8.1.1 – *Registering Global Services*. More than one service, hence more than one Service Parameter, may be registered for a Global Services Application.

6.5.2 Card-Wide Data

The card Life Cycle State is stored in the GlobalPlatform Registry similarly to Application information. Any restriction of the Card Content Management functionality of OPEN is stored in the GlobalPlatform Registry.

6.6 Privileges

6.6.1 Privilege Definition

The following table defines Security Domain and Application privileges:

Table 6-1: Privileges

Privilege Number	Privilege	Description	Notes
0	Security Domain	Application is a Security Domain.	Distinguishes a Security Domain from a 'normal' Application.
1	DAP Verification	Application is capable of verifying a DAP; Security Domain privilege shall also be set.	For details, see section 9.2.1.
2	Delegated Management	Application is capable of Delegated Card Content Management: Security Domain privilege shall also be set.	For details, see section 9.1.3.3.
3	Card Lock	Application has the privilege to lock the card.	For details, see section 9.6.3.
4	Card Terminate	Application has the privilege to terminate the card.	For details, see section 9.6.4.
5	Card Reset	Application has the privilege to modify historical bytes. This privilege was previously labeled 'Default Selected'.	For details, see section 6.6.2.
6	CVM Management	Application has the privilege to manage a shared CVM of a CVM Application.	For details, see section 8.2.1.
7	Mandated DAP Verification	Application is capable of and requires the verification of a DAP for all load operations: Security Domain privilege and DAP Verification privilege shall also be set.	For details, see section 9.2.1.
8	Trusted Path	Application is a Trusted Path for inter-application communication.	For details, see section 6.7.
9	Authorized Management	Application is capable of Card Content Management; Security Domain privilege shall also be set.	For details, see section 9.1.3.2.
10	Token Verification	Application is capable of verifying a token for Delegated Card Content Management.	For details, see sections 9.1.3.1 and 9.2.3.
11	Global Delete	Application may delete any Card Content.	For details, see sections 9.1.3.4 and 9.5.
12	Global Lock	Application may lock or unlock any Application.	For details, see sections 9.1.3.5 and 9.6.2.

Privilege Number	Privilege	Description	Notes
13	Global Registry	Application may access any entry in the GlobalPlatform Registry.	For details, see section 9.6.5.
14	Final Application	The only Application selectable in card Life Cycle State CARD_LOCKED and TERMINATED.	For details, see section 9.6.4.
15	Global Service	Application provides services to other Applications on the card.	For details, see section 8.1.1.
16	Receipt Generation	Application is capable of generating a receipt for Delegated Card Content Management.	For details, see section 9.1.3.6.
17	Ciphered Load File Data Block	The Security Domain requires that the Load File being associated to it is to be loaded ciphered.	For details, see section 9.1.3.7.
18	Contactless Activation	Application is capable of activating and deactivating other Applications on the contactless interface.	For details, see [Amd C] section 7.1.
19	Contactless Self-Activation	Application is capable of activating itself on the contactless interface without a prior request to the Application with the Contactless Activation privilege.	For details, see [Amd C] section 7.2.

6.6.2 Privilege Assignment

The following rules apply to the assignment of Privileges:

- Only one Application or Security Domain in the card may be set with the Card Reset privilege at a time (e.g. the Issuer Security Domain, a current legacy Application or an Application that requires specific behavior with regards to logical channels).
- Once the Card Reset privilege has been assigned to an Application, the privilege can be reassigned to a new Application either by deleting the Application which has the privilege, or by revoking its privilege.
- The Card Reset privilege is by default assigned to the Issuer Security Domain. It may be reassigned only if the Issuer Security Domain has the Card Reset privilege.
- If the application with the Card Reset privilege is deleted, the privilege is reassigned to the Issuer Security Domain.
- The Final Application privilege is by default assigned to the Issuer Security Domain. It may be reassigned only if the Issuer Security Domain has the Final Application privilege.
- Only one Application or Security Domain in the card may be set with the Final Application privilege at a time (e.g. the Issuer Security Domain, a current legacy Application or an Application that requires specific behavior with regards to logical channels).
- Once the Final Application privilege has been assigned to an Application, the privilege can be reassigned to a new Application either by deleting the Application which has the privilege, or by revoking its privilege.

- If the application with the Final Application privilege is deleted, the Issuer Security Domain becomes the Application with Final Application privilege.
- Authorized Management and Delegated Management privileges are mutually exclusive.
- Token Verification and Delegated Management privileges are mutually exclusive.
- Receipt Generation and Delegated Management privileges are mutually exclusive.
- Otherwise, the privileges are not mutually exclusive; therefore, one or more privileges may be marked as set for an Application.

In the OP_READY card Life Cycle State, the Issuer Security Domain shall initially have the following set of privileges clearly identifying its functionality: Security Domain, Authorized Management, Global Registry, Global Lock, Global Delete, Token Verification, Card Lock, Card Terminate, Trusted Path, CVM Management, Card Reset, Final Application and Receipt Generation.

Whether the privileges of an Application or Security Domain may be updated after its installation shall be defined in configuration documents. If allowed, such an update may be performed using the INSTALL [for registry update] command (see section 11.5.2.3.5).

For backward compatibility, where a card supports only privileges 0-7, the following assumptions shall apply for the remaining privileges:

Table 6-2: Privilege Defaults

Privilege Number	Privilege	Issuer Security Domain	Supplementary Security Domain	Other Application
8	Trusted Path	yes	yes	no
9	Authorized Management	yes	no	no
10	Token Verification	yes	no	no
11	Global Delete	yes	no	no
12	Global Lock	yes	no	no
13	Global Registry	yes	no	no
14	Final Application	yes	no	no
15	Global Service	no	no	no
16	Receipt Generation	yes	no	no
17	Ciphered Load File Data Block	no	no	no
18	Contactless Activation	no	no	no
19	Contactless Self-Activation	no	no	no

Several use cases can be considered for the assignment of privileges to Security Domains, depending on the business relationships between the Card Issuer and other off-card entities. The following table shows four example use cases.

Table 6-3: Privilege Assignment Example Use Cases

		Use case			
		A	B	C	D
	Authorized Management Privilege	✓	✓	✓	

		Use case			
		A	**B**	**C**	**D**
Issuer Security Domain	Delegated Management Privilege				✓
	Token Verification Privilege	✓	✓		✓
	Receipt Generation Privilege	✓	✓		✓
Supplementary Security Domain	Authorized Management Privilege				
	Delegated Management Privilege		✓		✓
	Token Verification Privilege			✓	
	Receipt Generation Privilege			✓	

6.6.3 Privilege Management

Runtime Behavior

The OPEN shall use the Privileges data element in conjunction with the association hierarchy (see section 7.2 – *Security Domain Association*) of the Applications for controlling the following runtime behavioral requirements:

- Ensuring Token verification when required (see section 9.3.2 – *Card Content Loading*)
- Ensuring Receipt generation when required (see section 9.3.2 – *Card Content Loading*)
- Ensuring DAP verification when required (see section 9.3.2 – *Card Content Loading*)
- Checking for the validity of a request to change Card Contents: load, installation, extradition, registry update, or deletion
- Checking for the validity of a request to lock or unlock the card
- Checking for the validity of a request to lock or unlock an Application, including a Security Domain
- Checking for the validity of a request to terminate the card
- Checking for the validity of a request to communicate with another on-card Application
- Checking for the validity of a request to access the GlobalPlatform Registry entry of another Application, including a Security Domain
- Identifying the Application with Final Application privilege to be default selected when the card is in CARD_LOCKED or TERMINATED state

The OPEN shall use the Privileges data element and the Implicit Selection parameter for each Application (where present) for controlling the following runtime behavioral requirements:

- Checking for the validity of a request to modify historical bytes; for dual interface (contact and contactless) cards, the modification applies to the historical bytes of the card interface(s) for which the requesting on-card Application is registered as implicitly selectable; this feature should preferably be used on cards that support the Basic Logical Channel only.

6.7 The GlobalPlatform Trusted Framework

This section describes a mechanism through which APDU commands received by a Security Domain may be securely forwarded to one of its associated Applications, through a so-called Trusted Framework. Another example of Trusted Framework is described in Appendix G.

An APDU command is received by the Application's Security Domain (the Receiving Entity), which may be the Issuer Security Domain or a Supplementary Security Domain, and is unwrapped by the Security Domain before being passed on to the GlobalPlatform Trusted Framework. The objective is that the GlobalPlatform Trusted Framework forwards the unwrapped command to the Target Application indicated by the Receiving Entity.

The Target Application may use cryptographic services of the Security Domain available through the Secure Channel Session established by the Security Domain (e.g. sensitive data decryption). The Current Security Level viewed by the Target Application may differ from the Security Domain view depending on the Secure Channel Protocol and the authenticated off-card entity's Application Provider ID (e.g. ANY_AUTHENTICATED for the Target Application and AUTHENTICATED for the Security Domain).

Runtime Behavior

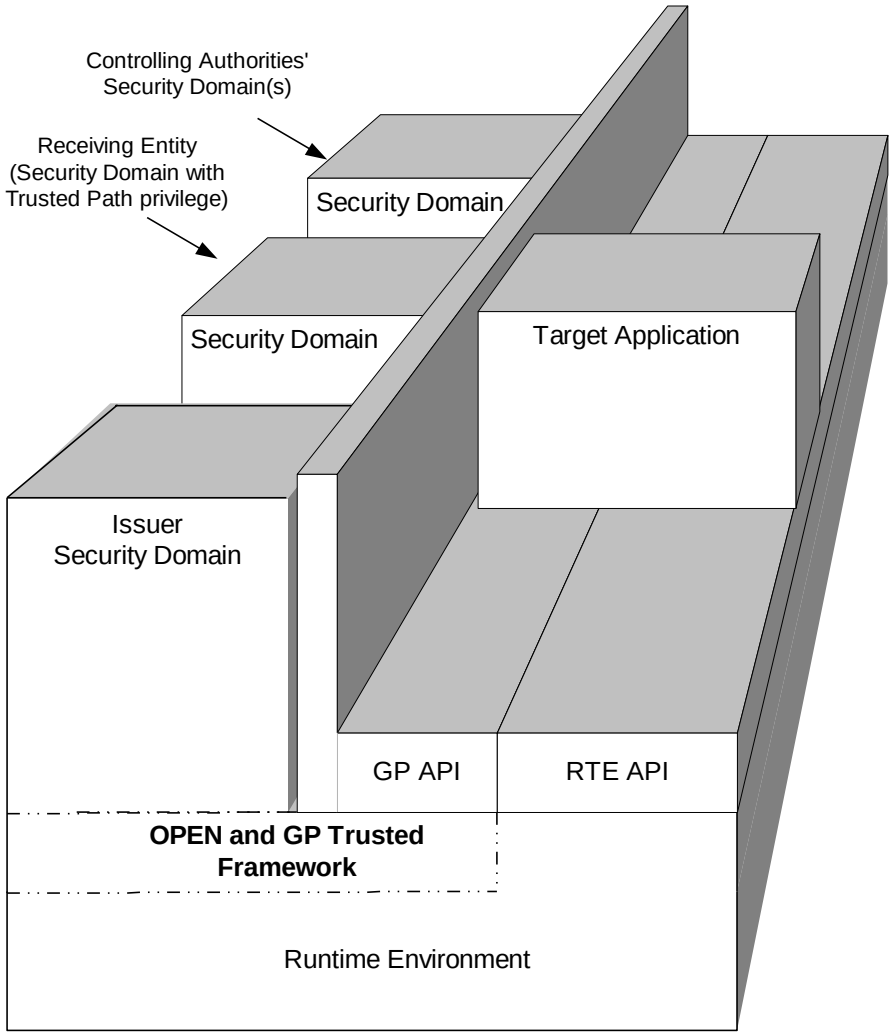
The GlobalPlatform Trusted Framework shall check that:

- The Receiving Entity has the Trusted Path privilege.
- The Receiving Entity is a Security Domain.
- The Target Application exists in the GlobalPlatform Registry and has enabled its 'Process Data' entry point.
- The Target Application has no multi-selection restrictions if it is already selected on another logical channel.
- The Target Application is associated with the currently selected Receiving Entity Security Domain.

If all checks are passed, the GlobalPlatform Trusted Framework passes the command to the Target Application through its GlobalPlatform Application interface.

The GlobalPlatform Trusted Framework and the Applications involved are shown in the following diagram. The process as used in personalization is described in more detail in section 7.3.2 – Security Domain Support for Application Personalization.

Figure 6-1: GlobalPlatform Trusted Framework Roles



7 Security Domains

7.1 General Description

Security Domains are privileged Applications. They hold cryptographic keys which can be used to support Secure Channel Protocol operations and/or to authorize card content management functions.

Each Application and each Executable Load File is associated with a Security Domain. An Application can use the cryptographic services of its associated Security Domain. It is possible to associate Applications owned by one authority with the Security Domain of another authority.

All cards have one mandatory Security Domain: the Issuer Security Domain. A card that supports multiple Security Domains can allow an Application Provider, through its own Security Domain, to manage its own Applications and provide cryptographic services using keys that are completely separate from, and not under the control of, the Card Issuer.

Key Separation

Security Domains are responsible for their own key management. This ensures that Applications and data from different Application Providers may coexist on the same card without violating the privacy and integrity of each Application Provider.

Application Services

The keys and associated cryptography for all Security Domains may be used for:

- Personalization Support: Secure communication support during personalization of an Application Provider's Applications;
- Runtime Messaging Support: secure communication support during runtime for an Application that does not contain its own secure messaging keys.

7.1.1 Issuer Security Domain

The Issuer Security Domain primarily operates as any Security Domain, but has some special characteristics that distinguish it from any other Security Domain:

- It is the first Application installed on a card. GlobalPlatform does not mandate that the Issuer Security Domain be loaded or installed in the same manner as Applications. The Issuer Security Domain, while viewed by the GlobalPlatform Registry as an Application, has implementation specific behavior relating to how it becomes an active entity on the card;
- It does not have a Security Domain Life Cycle State because it inherits the card Life Cycle State;
- It takes on the Card Reset privilege if the Application with that privilege is removed;
- It becomes the implicitly selected Application if the Application implicitly selectable on the same logical channel of the same card I/O interface is removed;
- It becomes the implicitly selected Application for a card interface on a logical channel if the Application that is implicitly selectable on that logical channel for that card interface is removed;
- It may be selected by use of the SELECT command with no command data field;
- It takes on the Final Application privilege if the Application with that privilege is removed.

7.2 Security Domain Association

Applications (including Security Domains) may use the services of their associated Security Domain to provide Secure Channel Sessions and other cryptographic services. An Application need not have any prior knowledge of its Security Domain AID: the GlobalPlatform Registry contains this information and the OPEN supplies the Application with a reference to its associated Security Domain. The Application should not store this reference for future use because its associated Security Domain may change as a result of extradition.

Extradition is the means by which an Application is associated with a different Security Domain.

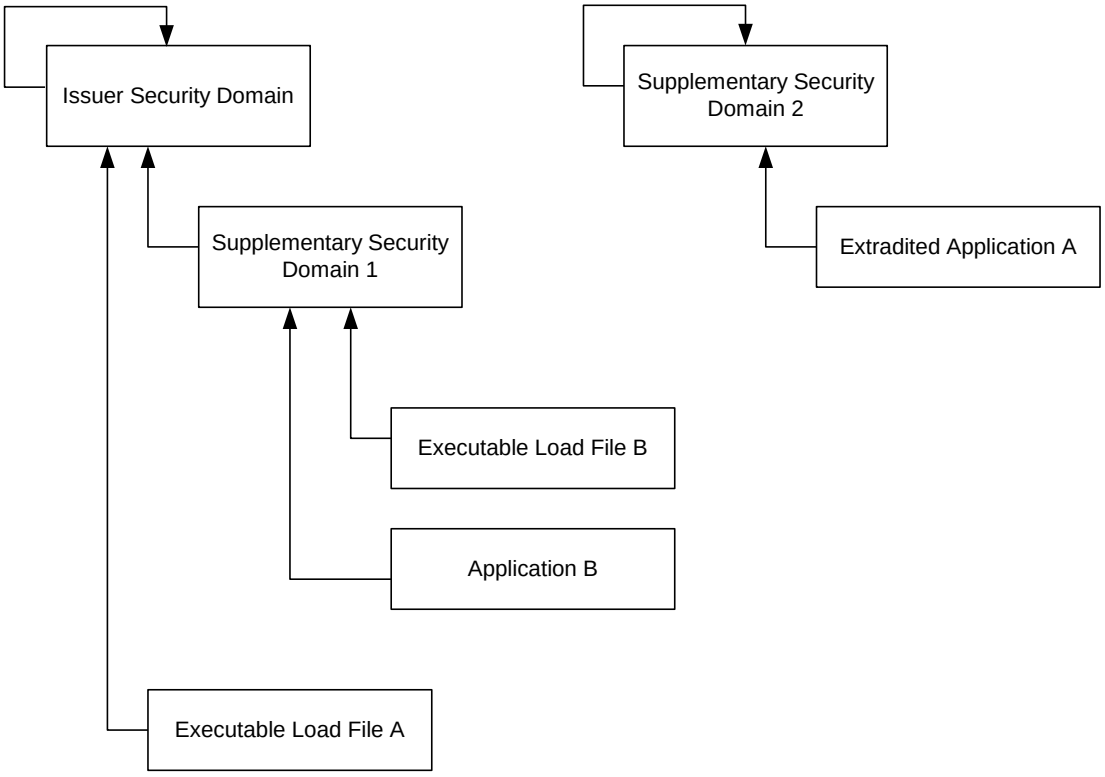
An Executable Load File is initially associated with the Security Domain which loads it, but it may be extradited immediately (implicit extradition if indicated during the load process) or subsequently (explicit extradition with the INSTALL [for extradition] command) to a different Security Domain.

The Issuer Security Domain is effectively associated with itself, its establishment on the card is not defined by GlobalPlatform, and it is not subject to extradition.

Through extradition, a Security Domain may be associated with itself. This means that if it calls on the services of its associated Security Domain, it will be using its own services.

As a result, there are one or more hierarchies of association on a card. The root of each hierarchy is a Security Domain that is associated with itself. The following diagram shows two hierarchies: one starting from the Issuer Security Domain, the other from a Security Domain which has been extradited to itself; arrows indicate ‘is associated with’.

Figure 7-1: Example of Security Domain Hierarchies



A Security Domain associated with another Application is said to be directly associated with it. Another Security Domain higher up the hierarchy is said to be indirectly associated with it.

A sub-hierarchy of a Security Domain includes the Security Domain itself and all of the Security Domain's Applications and Executable Load Files below it in the same hierarchy.

7.3 Security Domain Services

7.3.1 Security Domain Support for Secure Messaging

Applications can access the Secure Messaging services of their associated Security Domain. By using these services, an Application may rely on cryptographic support from the Security Domain to ensure confidentiality and integrity during personalization and runtime, instead of loading its own secure messaging keys and/or implement its own secure messaging protocol. The Security Domain services defined in this specification are generic and shall encompass the following possibilities.

- Initiating a Secure Channel Session upon successful verification of an off-card entity;
- Unwrapping a command received within a Secure Channel Session by verifying its integrity and/or decrypting the original data in the case of confidentiality;
- Controlling the sequence of APDU commands;
- Decrypting a secret data block;
- Setting the security level: integrity and/or confidentiality, to apply to the next incoming command and/or next outgoing response;
- Closing a Secure Channel Session upon request and destroying any secret(s) relating to that Secure Channel Session.

Depending on the specific Secure Channel Protocol supported, the Security Domain services may also encompass the possibility of:

- Wrapping a response sent within a Secure Channel Session by adding integrity and/or encrypting the original data in the case of confidentiality;
- Encrypting a secret data block;
- Controlling the sequence of APDU responses.

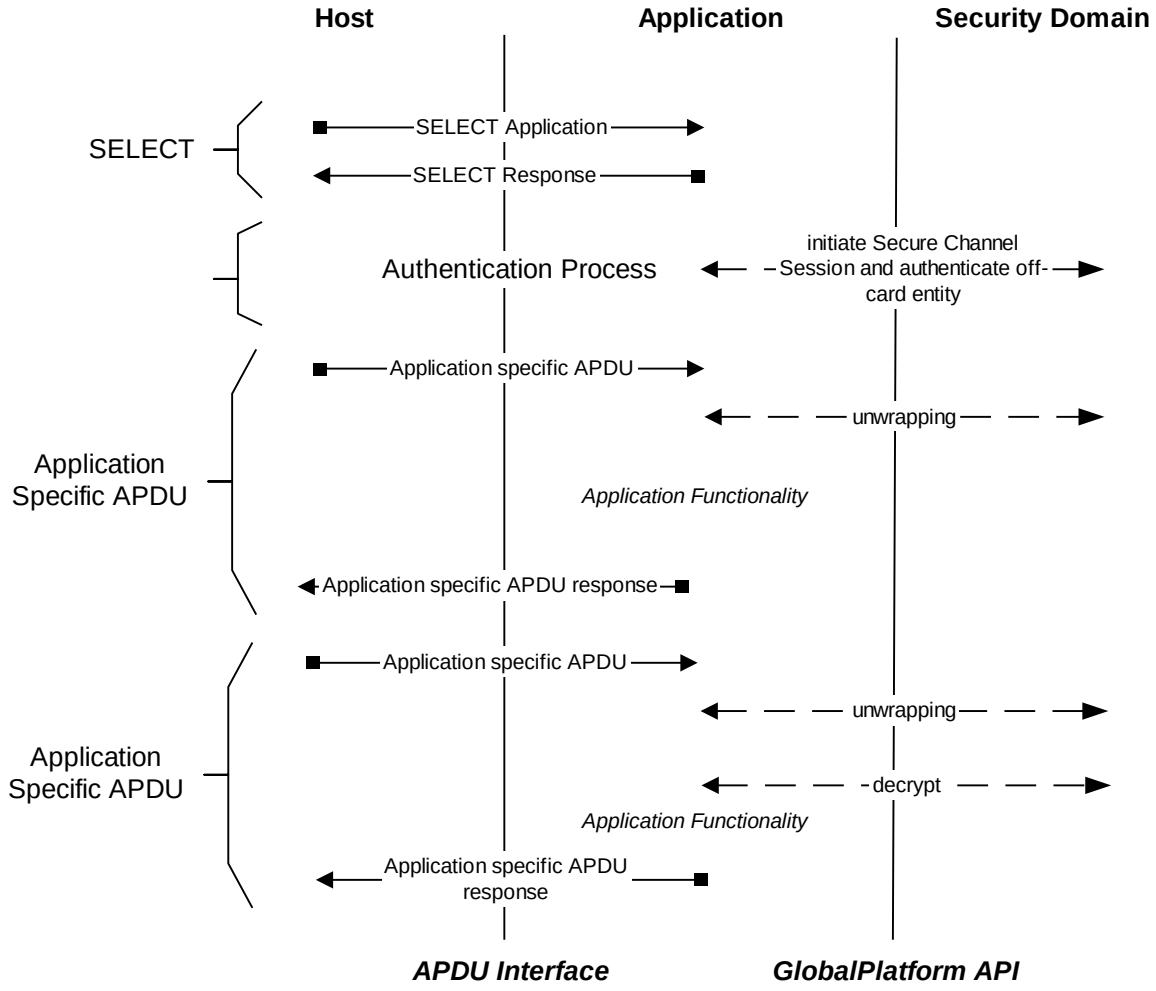
A Security Domain may support the management of multiple Secure Channel Sessions concurrently (i.e. Applications selected on multiple logical channels each initiating a Secure Channel) or may limit itself to only managing one Secure Channel Session immaterial of the number of concurrently selected Applications that attempt to use its services. If a Security Domain does support the management of multiple Secure Channel Sessions concurrently, it shall be able to differentiate between the multiple Secure Channel Sessions and their respective logical channels. If a Security Domain does not support the management of multiple Secure Channel Sessions concurrently, when a request is made to open a Secure Channel Session on a logical channel different from the current Secure Channel Session, the Security Domain rejects this request to initiate a new Secure Channel Session.

If a request is made by an Application to use the services of its associated Security Domain on a logical channel different from the one on which its Secure Channel Session was opened, the Security Domain shall reject this request.

Runtime Messaging Flow

The following diagram is an example of an Application using the services of its associated Security Domain:

Figure 7-2: Runtime Messaging Flow



7.3.2 Security Domain Support for Application Personalization

The Security Domain can receive a STORE DATA command destined to one of its associated Applications. The Security Domain unwraps this command according to the Current Security Level of the current Secure Channel Session and prior to the command being forwarded to the Application. This pre-processing leaves as is the data structures of the command message as well as the eventual encryption of the data value fields of these data structures.

The command is forwarded to the Application by the GlobalPlatform Trusted Framework which handles inter-application communication between Security Domains and Applications, as described in section 6.7.

An Application can use the secure communication and key decryption services of its associated Security Domain to manage the secure loading of its personalization data. This can be achieved in two ways: one being to use the runtime messaging support described in section 7.3.1 – Security Domain Support for Secure Messaging, the other being to use the Security Domain's ability to access the Application which is the method described below.

The Application may be personalized without being the currently selected Application. Rather the Security Domain is the selected Application and the Security Domain receives commands on behalf of the Application. The Security Domain would pre-process the personalization (STORE DATA) command prior to forwarding the command to the Application via the GlobalPlatform Trusted Framework.

Using this method on a card that supports multiple logical channels differs slightly only in that the Application could have a multi-selection restriction in which case personalization would fail. If an Application does not support this method, the Security Domain shall reject any STORE DATA command destined to this Application.

Runtime Behavior

On receipt of an INSTALL [for personalization] command and subsequent STORE DATA commands, the Security Domain performing the personalization of the Application shall:

- Apply its own secure communication policy. (Note: A minimum security level is defined in Table 11-2.)
- Check that the off-card entity is authenticated as the Application Provider.
- Pre-process subsequent STORE DATA commands according to the Current Security Level of the Secure Channel Session and prior to the command being forwarded to the target Application.

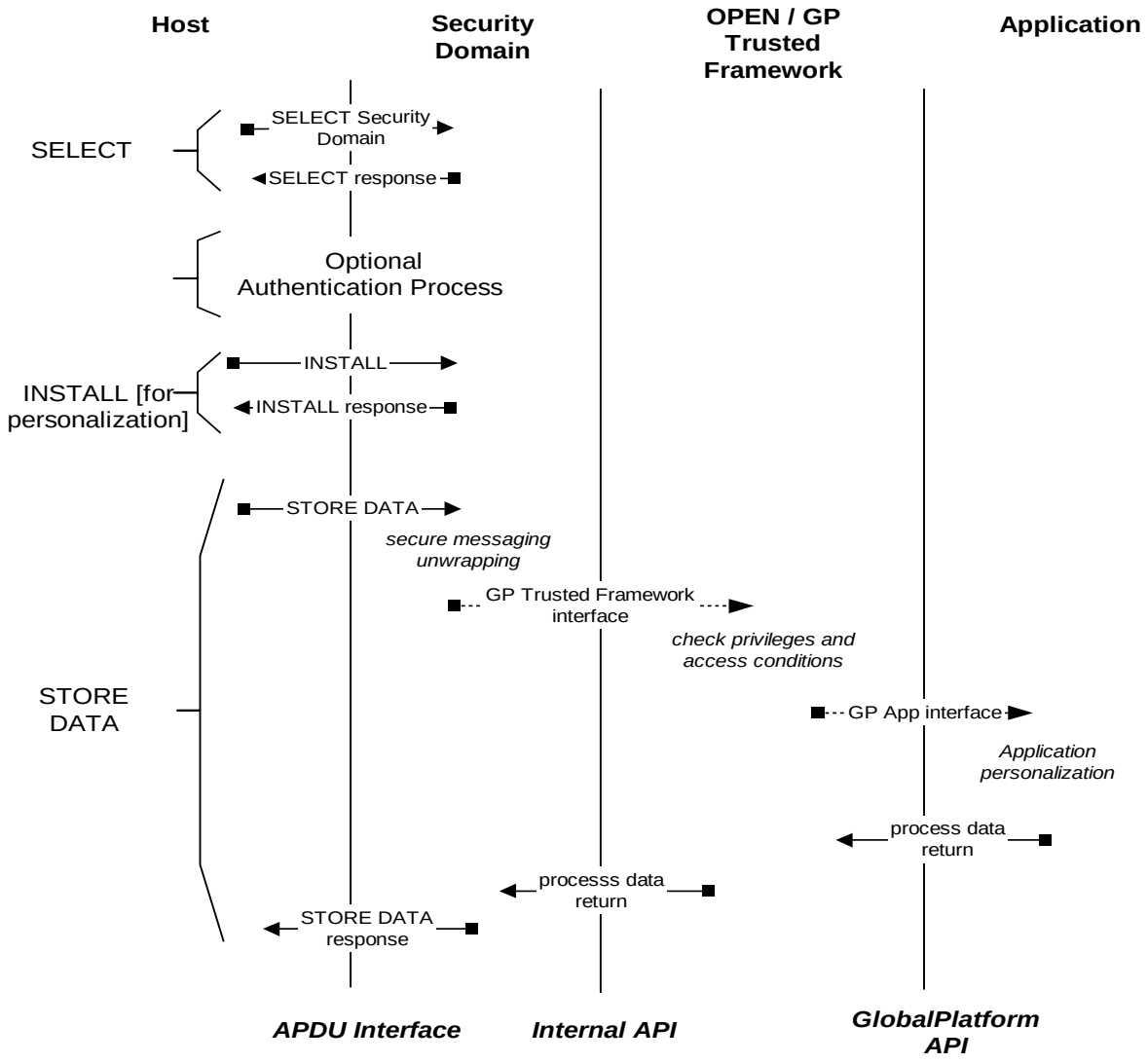
On receipt of a request to forward commands to an Application, the OPEN shall:

- Check that the card Life Cycle State is not CARD_LOCKED or TERMINATED.
- Check that OPEN and the requesting on-card entity have no restrictions for personalization (see Table 11-53).
- Apply the GlobalPlatform Trusted Framework runtime requirements defined in section 6.7.

Personalization Flow

The following diagram is an example of an Application receiving data from its associated Security Domain during personalization.

Figure 7-3: Application Personalization through Associated Security Domain



7.4 Security Domain Data

7.4.1 Issuer Security Domain

The Issuer Security Domain shall be able to handle the following data (available from the card using the GET DATA command):

- Issuer Identification Number (IIN)
- Card Image Number (CIN)
- Card Recognition Data
- Other Card Issuer's proprietary data

Optionally, the Issuer Security Domain shall also be able to handle the following data (available from the card using the GET DATA command):

- Card Capability Information

7.4.1.1 Issuer Identification Number

The Issuer Identification Number (IIN) may be used by an off-card entity to associate the card with a specific Card Management System. The IIN typically contains the ISO 7812 defined identification of the Issuer and is carried by the [ISO 7816-6] tag '42'. The IIN data element is of variable length.

7.4.1.2 Card Image Number

The Card Image Number (CIN) may be used by a Card Management System to uniquely identify a card within its card base. It is a unique value, carried by the ISO/IEC 7816 defined tag '45' (Card Issuer's Data), which is assigned by the Card Issuer (identified itself by the IIN). The CIN data element is of variable length.

7.4.1.3 Card Recognition Data

Card Management Systems need information about a card before they can start to interact with it. This includes the kind of card it is and what Secure Channel Protocol it supports.

Card Recognition Data is the mechanism for providing information about a card, and thus avoiding the vagaries of trial-and-error. See section H.2 for details.

Card Recognition Data shall be present and contained in a data template (tag '73'). This template shall in turn be contained in the 'Card Data' data object (tag '66'), as defined in [ISO 7816-6]. Card Data can be retrieved using the GET DATA command. (For contact cards according to [ISO 7816-4], Card Data may also be accessed through the ATR, if a suitable 'command to perform' is included in the ATR historical bytes.)

Note that the information provided in Card Recognition Data should be enough to enable initial communication with the card without resorting to trial and error. Information not essential to this purpose should be supplied during subsequent interaction with the card.

There is no specific requirement for Card Recognition Data to be updated dynamically by the card, but additional dynamic data objects are not precluded.

7.4.1.4 Card Capability Information

This data is optionally available. Card Capability Information provides information complementary to Card Recognition Data about the cipher suites actually supported by the card. See section H.4 for details.

If available, Card Capability Information shall be contained in a data template (tag '67') and shall be retrieved using the GET DATA command.

The implementation is not responsible for enforcing consistency between Card Capability Information and the functionality actually supported by the card.

7.4.2 Supplementary Security Domains

Security Domains other than the Issuer Security Domain may handle their own identification data:

- A Security Domain Provider Identification Number (SIN)
- A Security Domain Image Number
- Security Domain Management Data
- Security Domain Manager URL
- Other Application Provider proprietary data

When present, these data shall be available from the Security Domain using the GET DATA command.

7.4.2.1 Security Domain Provider Identification Number

The Security Domain Provider Identification Number (SIN) may be used by an off-card entity to associate the Security Domain with a specific Card Management System. It is an IIN, typically contains the ISO 7812 defined identification of the Security Domain provider, and is carried by the [ISO 7816-6] tag '42'. The SIN data element is of variable length.

7.4.2.2 A Security Domain Image Number

The Security Domain Image Number may be used by an Application Management System to uniquely identify an instance of a Security Domain on a card. If used it is a unique value, carried by the ISO/IEC 7816 defined tag '45'.

7.4.2.3 Security Domain Management Data

Application Management Systems need information about a Security Domain before they can start to interact with it. This includes the kind of Security Domain it is and what Secure Channel Protocol it supports.

Security Domain Management Data is the mechanism for providing information about a Security Domain, and thus avoiding the vagaries of trial-and-error. See section H.3 for details.

Security Domain Management Data (tag '73') shall be returned in the response to the SELECT command when present. Security Domain Management Data shall also be returned in response to a GET DATA command with tag '66'.

Note that the information provided in Security Domain Management Data should be enough to enable initial communication with the Security Domain without resorting to trial and error.

There is no specific requirement for Security Domain Management Data to be updated dynamically by the card, but additional dynamic data objects are not precluded.

7.4.2.4 Security Domain Manager URL

The Security Domain Manager URL (tag '5F50') provides an internet link to the manager of the Security Domain. The content and coding of this data object are defined in the GlobalPlatform System Protocol Discovery Mechanism Specification [GP SPDM].

7.5 Security Domain Keys

7.5.1 Key Information

See appendix D, E, and F and the Amendments for details about the keys used with each Secure Channel Protocol. See section C.1 for details about token, receipt, and DAP keys.

Keys have the following attributes:

- Key Version Number and Key Identifier
  - There is no restriction and no pre-defined order in assigning Key Version Numbers and Key Identifiers to keys, including non-sequential Key Version Numbers and Key Identifiers within the same Security Domain.
  - Key Version Numbers and/or Key Identifiers may be associated to specific Key Usages in specific configuration documents.
- A key type, which associates a specific key with one and only one cryptographic algorithm
- A length, for cryptographic algorithms supporting several key (or key component) lengths
- Access conditions, to control and segment access to keys

A key may consist of one or more key components; e.g. a symmetric key has only one key component while an asymmetric key has several components. All key components share the same Key Version Number and Key Identifier.

The combination of a Key Identifier and a Key Version Number identifies unambiguously a key within Security Domain. The key type identifies the cryptographic algorithm and key component. Identifying unambiguously a key and an algorithm within an entity prevent the misuse of cryptographic functionality.

An off-card entity may obtain information on Security Domain key(s) with a GET DATA command for Key Information Template (tag 'E0').

A Security Domain manages keys as follows:

- A Key Identifier and Key Version Number uniquely reference each key within the Security Domain. In other words, each combination Key Identifier / Key Version Number identifies a unique key within that entity.
- Adding a key is equivalent to allocating a new key with a new value, a new Key Identifier, and a new Key Version Number.
- Replacing a key involves updating the key with a new value and possibly a new Key Version Number. The Key Identifier remains the same. The previous key shall no longer be available.

The off-card key management system shall be aware of the scheme used to identify keys held by the Security Domain. Key Identifiers and Key Version Numbers may have arbitrary values (e.g. not being sequential, not starting with '01') and these values may vary from one key management scheme to the next. The assignment of specific Key Version Numbers and Key Identifiers to a specific cryptographic usage is defined in configuration documents.

Any velocity checking related to a particular keys usage; e.g. key try counters and limits are dependent on the Security Domain provider’s security policy.

The Security Domain shall store all the Key Information supplied in the PUT KEY command in association with each key.

7.5.2 Key Access Conditions

The Access Conditions assigned to a Security Domain key are either:

- Owner only: the Security Domain itself
- Authorized users other than the owner; e.g. the Security Domain's associated Applications
- Any authorized users, including the owner; e.g. the Security Domain and its associated Applications

This version of the Specification defines the following Access Conditions for Security Domain keys, coded on one byte as follows:

- '00': any authorized user, including the owner; this is the Access Condition by default for Secure Channel Protocol Keys, when not explicitly provided in the PUT KEY command;
- '01': the owner only; this is the Access Condition by default for Token and DAP Keys, when not explicitly provided in the PUT KEY command;
- '02': authorized users other than the owner;
- '03' to '7F': Reserved for Future Use by GlobalPlatform;
- '80' to 'FE': Reserved for proprietary use;
- 'FF': not available.

The access control rules applicable to any Security Domain key are enforced as follows:

- In order to use any Security Domain's cryptographic service, the Application requests to the OPEN the reference of a Secure Channel interface; the OPEN identifies the Security Domain associated with the Application and provides the corresponding Secure Channel interface reference to the Application (GlobalPlatform Registry acting as an access control list);
- The Application requests cryptographic services to a Security Domain, via its Secure Channel interface; since the OPEN ensures access to associated Applications only, the Security Domain controls are simplified to enforce the Access Conditions applicable to each of its keys (e.g. rejecting requests for accessing a key with an Access Condition set to '01').

7.6 Data and Key Management

These services relate to the storing of cryptographic keys and data on the card.

Runtime Behavior

On receipt of a data/key management request, the corresponding Security Domain shall manage the data/key according to its own access control rules.

The card Life Cycle State shall not be CARD_LOCKED or TERMINATED.

On receipt of a DELETE [key], PUT KEY, or STORE DATA command, the Security Domain performing the data or key management shall apply its own secure communication policy. (Note: A minimum security level is defined in Table 11-2.) The Security Domain provider may apply its own key management policy regarding deletion of keys.

8 Global Platform Services

8.1 Global Services Applications

One or more Global Services Applications may be present on the card to provide services to other Applications on the card. Global Services Applications are distinguished from other Applications through a specific privilege: Global Service.

8.1.1 Registering Global Services

During the installation of an Application (see section 9.3.6 – *Card Content Installation*) or during the registry update for an already installed Application (see section 9.4 – *Content Extradition and Registry Update*), an Application may be assigned the Global Service privilege and optionally Global Service Parameters comprising one (or more) service name(s). When present, each service name identifies a service family and optionally a specific service identifier within that service family, identifying which service(s) the Global Services Application offers to other Applications. When present, the Global Service Parameters shall be recorded in the GlobalPlatform Registry for the Application. Uniqueness on the card of this (these) service name(s) is not checked so that more than one Global Services Application may offer similar services.

In addition, a Global Services Application may explicitly register one or more Global Service(s) with unique service name(s) using the GlobalPlatform API. OPEN is responsible for ensuring the uniqueness of each service name registered by Global Services Applications. Such services are known as uniquely registered Global Services.

Runtime Behavior

The following runtime behavior requirements apply to the OPEN during the registration process of a Global Service with a unique service name. On receipt of unique service registration request, the OPEN shall:

- Check that the requesting on-card entity has the Global Service privilege.
- If one (or more) service name(s) are recorded for that on-card entity (as provided in the INSTALL command), check that the requested service name matches exactly with (one of) the service name(s) recorded for that on-card entity, or belongs to the same service family if the recorded service name(s) only identifies(y) service family(ies). If no service name is recorded for that on-card entity, any service name may be registered.
- Check that the requested service name is not registered within the GlobalPlatform Registry for another on-card entity.
- Register accordingly as unique the requested service name.

The following runtime behavior requirements apply to the OPEN during the deregistration process of a Global Service with a unique service name. On receipt of service deregistration request, the OPEN shall:

- Check that the requesting on-card entity has the Global Service privilege;
- Check that the requested service name is registered in the GlobalPlatform Registry entry of the requesting on-card entity;
- Deregister accordingly the requested service name as being unique.

8.1.2 Application Access to Global Services

GlobalPlatform API is the interface Applications use to access Global Services. An Application may access a uniquely registered Global Service (using a service name only) or a specific Global Services Application (using a service name and an AID).

Runtime Behavior

The following runtime behavior requirements apply to the OPEN during the access of a uniquely registered Global Service or a specific Global Services Application. On receipt of service access request, the OPEN shall:

- If the request indicates a specific service name without any associated AID, check that the requested service name matches exactly with (one of) the service name(s) uniquely registered, or belongs to the same service family uniquely registered.
- If the request indicates a specific AID, check that the on-card entity identified in the request has the Global Service privilege, and that the requested service name matches exactly with (one of) the service name(s) recorded for that on-card entity, or belongs to (one of) the same service family(ies) recorded for that on-card entity.
- Identify the corresponding Global Services Application.
- Obtain the GlobalPlatform Service interface of the corresponding Global Services Application and forward it to the requesting on-card entity.

The requesting on-card entity can then directly invoke the GlobalPlatform Service interface of the corresponding Global Services Application and authenticate itself in order to obtain the requested service.

8.1.3 Global Service Parameters

Global Service Parameters may be defined for an Application and list one or more service names. Each service name is coded on two bytes. The first byte identifies the service family, the second the service id within that family. The following values are assigned to the service family identifier:

- '00' to '7F' – Reserved for proprietary use and not registered by GlobalPlatform
- '80' to '9F' – Reserved for use by GlobalPlatform
  - '80' – Not available
  - '81' – GlobalPlatform Secure Channel
  - '82' – GlobalPlatform CVM
  - '83' – Authority Service (CASD)
  - '84' – HTTP Administration
  - '85' – HTTP Report
- 'A0' to 'FE' – Reserved for use by individual schemes registered by GlobalPlatform
  - 'A0' – USSM
- 'FF' – Not available

The values assigned to the service id within a service family are specific to each service family.

- An Application shall use a service id value of '00' to indicate that it wants to retrieve a Global Service of the specified service family but does not care about the exact service id value. In this case, the OPEN may return any Global Service registered for that service family (the search strategy is implementation specific). An Application may register a Global Service with a service id value of '00'.
- For the GlobalPlatform Secure Channel family, the Secure Channel Protocol identifiers (see 10.7 – *Secure Channel Protocol Identifier*) are the assigned service id values for that family.
- For the GlobalPlatform CVM family, the CVM identifier values (see section 8.2.2.1 – *Registering CVM*) are the assigned service id values for that family.

8.2 CVM Application

The CVM Application, if present on a card, provides a mechanism for a Cardholder Verification Method (CVM), including velocity checking, that may be used by all Applications on the card. In this version of the Specification there is one CVM standardized by GlobalPlatform: the global Personal Identification Number (global PIN).

The CVM verification services may be accessed by any Application. CVM management services shall only be accessible to privileged on-card Applications. The CVM application may also offer CVM management services to suitably authenticated off-card entities through an APDU interface that is beyond the scope of this specification.

A CVM Application may support multiple CVMs, and there may be more than one CVM Application present on a card. Each CVM shall have a unique CVM identifier, which shall be unique across the whole card. The standardized CVM (global PIN), if present on a card, shall be given the CVM identifier of '11'.

For each CVM supported, the CVM Application shall:

- Hold securely CVM management data: CVM value, CVM State, CVM Retry Limit, and CVM Retry Counter.
- Perform CVM-specific risk management, such as internal velocity checks on the CVM to prevent card and Application access violations.
- Implement one or more CVM interfaces.
- Request the OPEN to register the CVM identifier if that CVM is unique.
- Provide the CVM services to Applications.
- Manage the CVM state.
- Manage the CVM Retry Limit and CVM Retry Counter.

Depending on the Issuer policy, a value for the Retry Limit may be set by default.

8.2.1 Application Access to CVM Services

The following CVM services shall be provided by a CVM Application to other on-card Applications:

- Retrieving the CVM state (e.g. to determine if the CVM value has been submitted, verified or blocked);
- Retrieving the number of remaining times the CVM value can be incorrectly presented prior to the CVM being blocked;
- Setting a new value for the CVM value. This depends on the requesting Application having the CVM Management privilege;
- Verifying the content of an incoming CVM value by comparing the incoming CVM value to the stored CVM value;
- Setting the maximum number of times the CVM value can be incorrectly presented prior to the CVM being blocked. This depends on the requesting Application having the CVM Management privilege.

8.2.2 CVM Management

8.2.2.1 Registering CVM

In order to offer unique CVM services that are accessible by other Applications, a CVM Application shall register with the OPEN the CVM identifier(s) for which it offers services. The OPEN shall ensure that the registered CVM identifiers are unique on the whole card. Reuse is only possible if the CVM identifier is deregistered or the corresponding CVM Application is deleted.

The following values are assigned to CVM identifiers:

- '00' – Not available
- '01' to '7F' – Reserved for use by GlobalPlatform
  - '01' to '10' – PIN as defined in ETSI TS 102 221 specification ([TS 102 221])
  - '11' – Global PIN
  - '12' to '1F' – PIN as defined in [TS 102 221]
- '80' to 'EF' – Reserved for use by individual schemes registered by GlobalPlatform
  - '81' to '9F' – PIN as defined in [TS 102 221]
- 'F0' to 'FF' – Reserved for proprietary use and not registered by GlobalPlatform

8.2.2.2 CVM States

The CVM state may be used by a CVM Application to assist in managing CVM services. The non-atomic states of the CVM may be seen within a Card Session. The CVM state, the Retry Limit, and the Retry Counter are closely related. All CVM state transitions are immediately visible to the Application that caused the transition as well as to any Applications that may be selected on other logical channels.

The CVM states are:

- ACTIVE
- INVALID_SUBMISSION
- VALIDATED
- BLOCKED

8.2.2.2.1 CVM State ACTIVE

The CVM state shall first become ACTIVE when both the CVM value and the Retry Limit are set. The CVM state may also transition back to ACTIVE from any other CVM state if a privileged Application unblocks the CVM or changes the CVM value. Changing the CVM value shall also reset the Retry Counter. At the end of a Card Session the CVM state shall transition back to ACTIVE, except if the CVM state transitioned to the CVM state BLOCKED during the Card Session. The end of the Card Session shall not reset the Retry Counter.

8.2.2.2.2 CVM State INVALID_SUBMISSION

During the CVM verification, the CVM state shall transition to INVALID_SUBMISSION if the CVM verification fails. The Retry Counter shall be updated.

The CVM state shall remain INVALID_SUBMISSION until:

- The Card Session ends;
- A valid CVM verification is performed;
- An Application resets the CVM state;
- A privileged Application either blocks the CVM or changes the CVM value or CVM Retry Limit;
- Subsequent CVM verification(s) fail causing the CVM state to transition to BLOCKED.

8.2.2.2.3 CVM State VALIDATED

During CVM verification, the CVM state shall transition to VALIDATED and the Retry Counter shall be reset if the CVM verification is successful.

The CVM state shall remain VALIDATED until:

- The Card Session ends;
- An invalid CVM verification is performed;
- An Application resets the CVM state;
- A privileged Application either blocks the CVM, or changes the CVM value or CVM Retry Limit.

8.2.2.2.4 CVM State BLOCKED

During CVM verification, if the CVM verification fails and the Retry Limit has been reached, the CVM state shall transition to BLOCKED. The CVM state may also transition to BLOCKED if a privileged Application initiates this transition. The BLOCKED state shall not transition when the Card Session ends. The CVM state may only transition from the BLOCKED state back to the ACTIVE state on instruction from a privileged Application, which either resets (unblocks) the CVM state or changes the CVM value or CVM Retry Limit. The CVM verification shall always fail in the BLOCKED state.

8.2.2.3 CVM Format

The following formats are defined for the CVM value:

- Format BCD includes only numerical digits, coded on a nibble (4 bits), left justified, and eventually padded on the right with an 'F' nibble if necessary (i.e. the number of digits is odd);
- Format ASCII includes all displayable characters (alphabetic, numerical, and special) and space (i.e. format ASCII ranges from '20' to '7E'), coded on one byte and left justified.
- Format HEX is equivalent to a transparent mode (“as is”) and includes all binary values coded on one byte.

The following rules apply for the CVM format conversion:

- No conversion from and to HEX format is valid;
- Conversion from BCD format to ASCII format is valid: the numeric nibbles are expanded to the corresponding characters coded on one byte and the padding nibble 'F' is deleted (if present);
- Conversion from ASCII format to BCD format is valid for numeric characters only: the numeric characters coded on one byte are converted to numeric nibbles, padded together in bytes, and a padding nibble 'F' is added on the right if necessary.

The internal format for storing the CVM value by a CVM Application is implementation dependent and shall be transparent to any other on-card Application that uses CVM services. It shall not preclude any format used by Applications requesting CVM services.

9 Card and Application Management

9.1 Card Content Management

9.1.1 Overview

Card Content management on a GlobalPlatform card is the capability for the loading, installation, extradition, registry update and removal of Card Content. GlobalPlatform is designed for providing maximum flexibility to the Card Issuer and its business partners regarding Card Content management. The design of the GlobalPlatform takes into account the possibility that the Card Issuer may not necessarily want to manage all Card Content changes, especially when the Card Content does not belong to the Card Issuer.

Thus the Card Issuer may delegate Card Content management to an Application Provider with or without authorization:

- It may authorize all Card Content management operations performed by an Application Provider;
- It may authorize an Application Provider to have full control of its Card Content;
- It may authorize an Application Provider to isolate its own Security Domain(s) and Application(s) from other Application Providers, and potentially from the Card Issuer itself.

Card Content changes are permitted according to the privileges that have been assigned to the various Security Domains on the card.

The following sections describe the OPEN and Security Domain requirements to support the operation and authorization of Card Content management.

9.1.2 OPEN Requirements

The OPEN:

- Performs the physical loading and installation;
- May prohibit more than one Card Content management operation occurring concurrently;
- Prohibits Card Content management in the card Life Cycle States CARD_LOCKED or TERMINATED.

9.1.3 Security Domain Requirements

The Security Domain through which Card Content management is performed applies its own secure communication policy, which should be checked for consistency with any Card Issuer policies before the Security Domain is installed.

Card Content management involves a Security Domain performing loading, installation, extradition, update to GlobalPlatform Registry and content removal operations.

9.1.3.1 Security Domain with Token Verification Privilege

The privilege allows a Security Domain Provider, to authorize Card Content management operations. Within a sub-hierarchy of Security Domains starting from the Security Domain with the Authorized Management Privilege, the Security Domain having the Token Verification privilege controls such authorization. A Security Domain with Token Verification privilege requires the knowledge of keys and algorithms used for Tokens.

Note: Typically, the Token Verification privilege is assigned to a Security Domain with the Authorized Management privilege. The Token Verification privilege does not provide Card Content Management capability.

In this version of the specification, it is assumed that within a sub-hierarchy, starting from a Security Domain with the Authorized Management privilege, no more than one Security Domain may have the Token Verification privilege.

9.1.3.2 Security Domain with Authorized Management Privilege

Having a Security Domain with this privilege allows a Security Domain provider to perform Card Content management without authorization (i.e. without token) in the case where the off-card entity is authenticated as the owner (Security Domain Provider) of the Security Domain. In that case the Security Domain that has Token Verification privilege is not involved. However, a Token is still required (as for Delegated Management; see section 9.1.3.3) if the off-card entity is authenticated but is not the Security Domain Provider (see ANY_AUTHENTICATED security level in section 10.4 – *Entity Authentication*).

9.1.3.3 Security Domain with Delegated Management Privilege.

The Delegated Management privilege allows an Application Provider's Security Domain with this privilege to perform:

- Delegated loading;
- Delegated installation and make selectable;
- Delegated extradition;
- Delegated update to the GlobalPlatform Registry;
- Delegated deletion.

The privilege allows an Application Provider to manage Card Content with authorization. Within a sub-hierarchy of Security Domains starting from the Security Domain with the Authorized Management privilege, the descendant Security Domain having the Token Verification privilege controls such authorization. In this version of the specification, it is assumed that the Security Domain with the Delegated Management privilege performs card content operations only within the sub-hierarchy starting from a Security Domain with the Authorized Management privilege.

Delegated Management is not a mandated feature of a GlobalPlatform card and is only necessary for Card Issuers that choose to offer this flexibility. In order to achieve it, close co-operation is required between the OPEN and the Security Domains.

9.1.3.4 Security Domain with Global Delete Privilege

This privilege provides the capability to remove any Executable Load File or Application from the card even if the Executable Load File or Application does not belong to this Security Domain.

9.1.3.5 Security Domain with Global Lock Privilege

This privilege provides the right to initiate the locking and unlocking of any Application on the card, independent of its Security Domain association and hierarchy. It also provides the capability to restrict the Card Content Management functionality of OPEN.

9.1.3.6 Security Domain with Receipt Generation Privilege

This privilege allows a Security Domain Provider, to provide a confirmation for a delegated card content management operation. Within a sub-hierarchy of Security Domains starting from the Security Domain with the Authorized Management privilege, the descendant Security Domain with Receipt Generation privilege shall generate the receipt. It requires the knowledge of keys and algorithms used for Receipt generation. It shall also keep track of a Confirmation Counter that is incremented when generating each Receipt. When reaching its maximum value, the Confirmation Counter shall not be reset to zero. Security Domains are not required to support counter values beyond 32767.

Note that this privilege does not provide Card Content management capability.

In this version of the specification, it is assumed that within a sub-hierarchy, starting from a Security Domain with the Authorized Management privilege, no more than one Security Domain may have the Receipt Generation privilege.

9.1.3.7 Ciphered Load File Data Block Privilege

This privilege allows a Security Domain Provider to require that the Load File Data Block being associated to it shall be ciphered.

9.2 Authorizing and Controlling Card Content

The following section details the authorization and control features that may be used during Card Content loading and installation. The responsibility of ensuring that these controls are present when required rests with the OPEN.

9.2.1 DAP Verification

An Application Provider may require that their Application code to be loaded on the card shall be checked for integrity and authenticity. The DAP Verification privilege of the Application Provider's Security Domain detailed in this Specification provides this service on behalf of an Application Provider.

A Verification Authority may require that all Application code to be loaded onto the card shall be checked for integrity and authenticity. The Mandated DAP Verification privilege of the Verification Authority's Security Domain detailed in this Specification provides this service on behalf of the Verification Authority.

The key and algorithm to be used for DAP Verification or Mandated DAP Verification are implicitly known by the corresponding Security Domain.

Flow control and runtime behavior as described in section 9.3.5 – *Card Content Loading Process* applies; more detail on DAP Blocks is provided in section C.3.

9.2.2 Load File Data Block Hash

The Load File Data Block Hash is an integrity check across the whole Load File Data Block to be transferred to the card and is present as a field in the INSTALL [for load] command; more detail is provided in section C.2. The Load File Data Block Hash is mandatory when a Token or DAP Block is present in a Load File, and is optional otherwise. If a Load File Data Block Hash is present in a Load File, then it shall be checked.

See section B.5 for more details on the hash algorithms.

Flow control and runtime behavior as described in section 9.3.5 – *Card Content Loading Process* applies.

9.2.3 Tokens

Tokens relate specifically to Delegated Management, and to Authorized Management where the off-card entity is not the Security Domain Provider. They are not allowed in other cases. More detail is provided in section C.4. The entity owning the Security Domain with Token Verification Privilege provides a Token to the Security Domain Provider performing the content management function. During the processing of the content management function the token is verified on-card by the Security Domain with Token Verification Privilege.

9.3 Card Content Loading, Installation and Make Selectable

9.3.1 Overview

The GlobalPlatform Card Content loading process is designed to allow the addition of code to Mutable Persistent Memory in the card. Card Content loading may be prohibited if a load process is already in progress on another logical channel.

The GlobalPlatform Card Content installation process is designed to allow the Card Issuer to make previously loaded application code executable on the card.

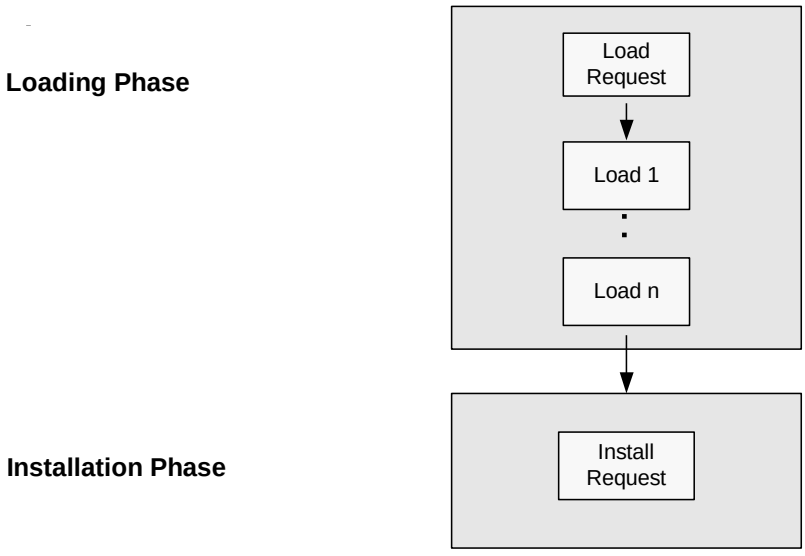
Card Content installation is either performed simultaneously with the load process, immediately following the load process or at a later time.

The Load File Data Block contains the information required in order to create an Executable Load File. The internal organization of the Load File Data Block is beyond the scope of this Specification. The Java Card™ CAP file definition and MULTOS™ Application Load Unit are examples of an expected Load File Data Block. The Load File Data Block may also contain information on the Load File Data Block attributes such as its name, version number and size. The Card Content loading and installation process may include implementation specific linking and actual verification of the executable code. Additional authentication data may also be present in the Load File.

Upon the successful completion of the Card Content loading, an Executable Load File shall be present on the card and the OPEN shall create an entry in the GlobalPlatform Registry for the Executable Load File. The OPEN shall also create an entry in the GlobalPlatform Registry for each Executable Module present within an Executable Load File. However, Executable Modules are not yet ready for execution. Applications may then be installed. Installing an Application results in another entry in the GlobalPlatform Registry.

The following figure details the two possible phases of the Card Content loading and installation.

Figure 9-1: Loading and Installation Process



9.3.2 Card Content Loading

When requested to do so by the OPEN, the Security Domain with Token Verification privilege shall verify the transmission of the Load File from the off-card entity to the card, and when applicable, Security Domains with the relevant privilege shall verify the integrity of the Load File Data Block before the OPEN commits the new content to memory.

A Security Domain with Authorized Management or Delegated Management privilege may load an Executable Load File to any Security Domain. The Executable Load File is subject to acceptance by the receiving Security Domain where applicable.

The load process comprises an INSTALL [for load] command and one or more LOAD commands all of which are processed by the Security Domain. The Security Domain then passes the load request and Load File information to the OPEN for additional verification and processing.

The Load Token allows the OPEN, via the Security Domain with Token Verification privilege within the same sub-hierarchy as the Security Domain performing the load, to ensure that the Token authorized the load process and the loading of the content of the Load File Data Block.

The Load File Data Block Hash links the Token to the actual Load File Data Block.

The response to the last LOAD command identifies the end of the load process. Following the completion of the load process, an optional Load Receipt is returned to the Security Domain performing the Delegated Management operation and shall be transmitted by the Security Domain to the off-card entity.

The Application Provider may then forward the Load Receipt to the corresponding off-card entity as a proof that the loading process was successfully performed. The purpose of the optional Load Receipt is to assist the Application Provider in keeping its Card Management System synchronized with its on-card application base.

9.3.3 Card Content Installation

A Security Domain with Authorized Management or Delegated Management privilege may install an Application. Upon successful installation,

- If the newly installed Application is a Security Domain, then it is associated with the Security Domain that performed the install operation.
- Otherwise, the newly installed Application is associated with the Security Domain associated with the Executable Load File it was created from.

The installation process comprises an INSTALL [for install] command processed by the Security Domain. The Security Domain then passes the install request to the OPEN for additional verification and processing.

The Install Token allows the OPEN, via the Security Domain with Token Verification privilege within the same sub-hierarchy as the Security Domain performing the installation, to ensure that the Token authorized the installation process.

The response to the INSTALL [for install] command identifies the end of the installation process. Following the completion of the installation process, an optional Install Receipt is returned to the Security Domain performing the Delegated Management operation and shall be transmitted by the Security Domain to the off-card entity.

The Application Provider may then forward the Install Receipt to the corresponding off-card entity as a proof that the installation process was successfully performed. The purpose of the optional Install Receipt is to assist the Application Provider in keeping its Card Management System synchronized with its on-card application base.

9.3.4 Card Content Combined Loading, Installation and Make Selectable

When requested to do so by the OPEN, the Security Domain with Token Verification privilege shall verify the transmission of the Load File from the off-card entity to the card, and when applicable, Security Domains with the relevant privilege shall verify the integrity of the Load File Data Block before the OPEN commits the new content to memory.

A Security Domain with Delegated Management privilege may load an Executable Load File to any Security Domain, install an Application from the Executable Load File and make the Application selectable.

The combined load, install and make selectable process comprises a first INSTALL [for load, install and make selectable] command, one or more LOAD commands and a last INSTALL [for load, install and make selectable] command all of which are processed by the Security Domain. The Executable Load File is subject to acceptance by the receiving Security Domain where applicable.

The combined Load, Install and Make Selectable Token allows the OPEN, via the Security Domain with Token Verification privilege within the same sub-hierarchy as the Security Domain performing the combined Load, Install and Make Selectable, to ensure that the Token authorized the load process and the loading of the content of the Load File Data Block as well as the installation of an Application from the previously loaded Executable Load File.

The response to the last INSTALL [for load, install and make selectable] command identifies the end of the combined load and install process. Following the completion of the load and install process, an optional combined Load, Install and Make Selectable Receipt is returned to the Security Domain performing the Delegated Management operation and shall be transmitted by the Security Domain to the off-card entity.

The Application Provider may then forward the combined Load, Install and Make Selectable Receipt to the corresponding off-card entity as a proof that the loading process was successfully performed. The purpose of the optional combined Load, Install and Make Selectable Receipt is to assist the Application Provider in keeping its Card Management System synchronized with its on-card application base.

The response to the last INSTALL [for load, install and make selectable] command completes the combined load, install and make selectable process.

9.3.5 Card Content Loading Process

The phases in Figure 9-1 use a combination of multiple occurrences of two different APDU commands (INSTALL and LOAD). The following sequence of APDU commands apply to the loading:

- An INSTALL [for load] command serves as the load request for loading. The INSTALL [for load] command data field details the requirements regarding a Load File;
- Multiple LOAD commands are then used to transport the Load File in blocks according to the size of the file and the communications buffer size of the card;
- Each INSTALL or LOAD command is processed by the receiving Security Domain before forwarding the load request and Load File to the OPEN for processing.

The following runtime behavior requirements apply during the content loading process.

Load Request Runtime Behavior

On receipt of an INSTALL [for load] command, the Security Domain performing the load shall:

- Apply its own secure communication policy. (Note: A minimum security level is defined in Table 11-2.)
- Apply its own security policy; e.g. check that its Life Cycle State is PERSONALIZED (only applicable to a Security Domain other than the Issuer Security Domain).

- If a Load File Data Block Hash is present in the INSTALL [for load] command, request the OPEN to initiate the hash verification of the subsequent Load File Data Block.
- If the Security Domain performing the load has Delegated Management privilege, check that a Load Token is present in the INSTALL [for load] command.
- If the Security Domain performing the load has the Authorized Management privilege and the off-card entity at the origin of the load request is not authenticated as its Security Domain Provider (see section 10.4 – *Entity Authentication*), check that a Load Token is present in the INSTALL [for load] command.
- If a Token is present in the INSTALL [for load] command, request the OPEN to obtain verification of the Load Token.
- If the Application Provider identifier is present in the load request, request the OPEN to save this in the GlobalPlatform Registry for the Executable Load File.

On receipt of a load request (arising from an INSTALL [for load] command), the OPEN shall:

- Check that the card Life Cycle State is not CARD LOCKED or TERMINATED.
- Check that OPEN and the requesting on-card entity have no restriction for load.
- Check that the requesting on-card entity is a Security Domain with Delegated Management or Authorized Management privilege.
- Check that the AID of the Load File is not already present in the GlobalPlatform Registry as an Executable Load File or Application.
- If an associated Security Domain AID is present and is not the Security Domain performing the load, check that this AID exists within the GlobalPlatform Registry and is registered with the Security Domain privilege. As this equates to the extradition of the Load File, check that the associated Security Domain accepts this extradition. If the associated Security Domain has the Ciphered Load File Data Block Privilege, the OPEN shall check that the Load File Data Block is sent ciphered (i.e. with the tag 'D4'). If no associated Security Domain AID is indicated, the Security Domain performing the load is by default the associated Security Domain.
- At the request of the Security Domain performing the load, request the Security Domain with Token Verification privilege to authorize the load request (e.g. to verify the Load Token).

At the request of OPEN, the associated Security Domain shall:

- Apply the Security Domain Provider's policy to accept or reject this load.
- Apply its own security policy; e.g. check that its Life Cycle State is PERSONALIZED (only applicable to a Security Domain other than the Issuer Security Domain).

At the request of OPEN, a Security Domain with Token Verification privilege shall:

- Verify the Load Token.

At the request of OPEN, the Security Domain to which the new Executable Load File shall be extradited (if any) shall:

- Apply its own policy to accept or reject this extradition.
- Apply its own security policy; e.g. check that its Life Cycle State is PERSONALIZED (only applicable to a Security Domain other than the Issuer Security Domain).

Load Phase Runtime Behavior

On receipt of the LOAD commands, the Security Domain performing the load shall:

- Apply its own secure communication policy. (Note: A minimum security level is defined in Table 11-2.)

- Discover whether any Security Domain has the Mandated DAP Verification privilege and is in the PERSONALIZED state, and if so:
  - Ensure that the required authentication data (DAP Block identifying the above Security Domain) is present in the Load File
- Check if the associated Security Domain has the DAP Verification privilege and if so:
  - Ensure that the required authentication data (DAP Block identifying the associated Security Domain) is present in the Load File;
- If authentication data (one or more DAP Blocks) is present in the Load File:
  - Ensure that a Load File Data Block Hash was received during the load request process;
  - Extract the authentication data (one or more DAP Blocks) from the Load File;
  - For each DAP Block of the Load File, request the OPEN to obtain verification of the DAP by the Security Domain indicated in the DAP Block.

On receipt of the Load File the OPEN shall:

- Verify the resource requirements of the Load File (see section 9.7 – *Memory Resource Management*) and that sufficient card resources are available;
- Check that each DAP verification request from the Security Domain performing the load relates to a Security Domain present in the GlobalPlatform Registry with DAP or Mandated DAP Verification privilege and if so request the Security Domain to verify the DAP;
- Compute the hash of the Load File Data Block when verification of a DAP Block or Load Token is requested.

At the request of OPEN, the Security Domain(s) verifying the DAP(s) shall:

- Verify that the DAP matches with the Load File Data Block Hash received in the load request.

Load Completion Runtime Behavior

On receipt of the last LOAD command, the Security Domain performing the load shall:

- Apply its own secure communication policy. (Note: A minimum security level is defined in Table 11-2.)
- Request the OPEN to obtain a Load Receipt.

On completion of the load process the OPEN shall:

- At the request of the Security Domain performing the load, verify the Load File Data Block Hash received in the load request;
- Check in the GlobalPlatform Registry if any Security Domain has the Mandated DAP Verification privilege and is in the PERSONALIZED state, and if so:
  - Ensure that the above Security Domain has successfully verified a DAP;
- Check in the GlobalPlatform Registry if the associated Security Domain has the DAP Verification privilege and if so:
  - Ensure that the associated Security Domain has successfully verified a DAP;
- If one or more DAP verifications were performed, verify the Load File Data Block Hash received in the load request;
- If the Security Domain performing the load has Delegated Management privilege, ensure that the Security Domain with Token Verification privilege has successfully verified a Token;
- If a Load Token was verified, verify the Load File Data Block Hash received in the load request;

- Create an Executable Load File using the Load File Data Block;
- Create an entry in the GlobalPlatform Registry for the Executable Load File indicating its associated Security Domain;
- Create an entry for each Executable Module within the Executable Load File in the GlobalPlatform Registry. This shall include the Application Provider identifier if requested by the Security Domain in the load request. The associated Security Domain for each Executable Module shall be the same as the associated Security Domain for the Executable Load File;
- At the request of the Security Domain performing the load, request the Security Domain with Receipt Generation privilege to generate a Load Receipt.

At the request of OPEN, the Security Domain with Receipt Generation privilege shall:

- Apply the issuer's policy to generate or not a Load Receipt.

If, at any stage, the OPEN determines that card resources are insufficient for the loading process or that any verification step has failed, the OPEN shall terminate the loading process, shall return the appropriate error and shall reclaim any memory allocated to the load process.

9.3.6 Card Content Installation Process

An INSTALL [for install] command is then used to request the installation of an application. The receiving Security Domain processes the INSTALL command before forwarding the install request to the OPEN for processing.

The installation internal processing is beyond the scope of this Specification. However, it is assumed that the installation process includes the creation of instances and allocation of Application data memory.

Following the installation, the OPEN shall register additional information in the GlobalPlatform Registry regarding the Application Life Cycle State, Security Domain association, and Privileges.

Applications inherit the associated Security Domain of the Executable Load File from which they are installed. They may be extradited to another Security Domain.

The following runtime behavior requirements apply during the content installation process:

Runtime Behavior (Installation)

On receipt of the INSTALL [for install] command, the Security Domain performing the installation shall:

- Apply its own secure communication policy. (Note: A minimum security level is defined in Table 11-2.)
- Apply its own security policy; e.g. check that its Life Cycle State is PERSONALIZED (only applicable to a Security Domain other than the Issuer Security Domain);
- If the Security Domain performing the installation has the Authorized Management privilege and the off-card entity at the origin of the installation request is not authenticated as its Security Domain Provider (see section 10.4 – *Entity Authentication*), check that an Install Token is present in the INSTALL [for install] command.
- If a Token is present in the INSTALL [for install] command, request the OPEN to obtain verification of the Install Token.
- If the Application Provider identifier is present in the install request, request the OPEN to save this in the GlobalPlatform Registry for the Application.
- Request the OPEN to obtain an Install Receipt.

On receipt of the install request, the OPEN shall:

- Check that the card Life Cycle State is not CARD_LOCKED or TERMINATED;

- Check that OPEN and the requesting on-card entity have no restriction for installation;
- Check that the requesting on-card entity is a Security Domain with Delegated Management or Authorized Management privilege;
- Check that the Executable Module AID is present in the GlobalPlatform Registry;
- Check that the Application AID (for future selection of the Application) is not already present in the GlobalPlatform Registry as an Application or Executable Load File;
- If the Security Domain performing the installation is not the Security Domain associated with the Executable Load File from which the Application is being installed, check that the Security Domain associated with the Executable Load File accepts this installation;
- At the request of the Security Domain performing the installation, request the Security Domain with Token Verification privilege to authorize the installation (e.g. to verify the Install Token);
- Verify the resource requirements indicated for the Application (see section 9.7 – *Memory Resource Management*) and that sufficient card resources are available;
- Perform the installation of the Application according to the underlying runtime environment requirements;
- If the Security Domain performing the installation has Delegated Management privilege, ensure that the Security Domain with Token Verification privilege has successfully verified a Token;
- Create an Application from the Executable Module;
- Ensure that the Application, depending on the underlying runtime environment, has the knowledge of its AID, its Privileges and its Install Parameters;
- Create an entry in the GlobalPlatform Registry for the Application indicating its associated Security Domain, Life Cycle State, Privileges and, when present in the install request, Implicit Selection, Service and Memory Resource Management parameters; and including the Application Provider identifier if supplied by the Security Domain in the installation request;
- At the request of the Security Domain performing the installation, request the Security Domain with Receipt Generation privilege to generate an Install Receipt.

At the request of OPEN, the associated Security Domain shall:

- Apply the Security Domain Provider's policy to accept or reject this installation;
- Apply its own security policy; e.g. check that its Life Cycle State is PERSONALIZED (only applicable to a Security Domain other than the Issuer Security Domain).

At the request of OPEN, the Security Domain with Token Verification privilege shall:

- Verify the Install Token.

At the request of OPEN, the Security Domain with Receipt Generation privilege shall:

- Apply the issuer's policy to generate or not an Install Receipt.

If the OPEN determines that card resources are insufficient for installing the Application or the runtime environment does not currently allow the installation, the OPEN shall terminate the installation process, return the appropriate error and reclaim any memory allocated to the install process.

9.3.7 Card Content Make Selectable Process

An INSTALL [for make selectable] command is used to request to make selectable a previously installed application. The receiving Security Domain processes the INSTALL command before forwarding the make selectable request to the OPEN for processing.

The make selectable internal processing is beyond the scope of this Specification.

Following the make selectable, the OPEN shall register additional information in the GlobalPlatform Registry regarding the Application Life Cycle State and Privileges.

The following runtime behavior requirements apply during the content make selectable process:

Runtime Behavior (Make Selectable)

On receipt of the INSTALL [for make selectable] command, the Security Domain making the Application selectable shall:

- Apply its own secure communication policy. (Note: A minimum security level is defined in Table 11-2.)
- Apply its own security policy; e.g. check that its Life Cycle State is PERSONALIZED (only applicable to a Security Domain other than the Issuer Security Domain);
- If the Security Domain making the Application selectable has the Authorized Management privilege and the off-card entity at the origin of the installation request is not authenticated as its Security Domain Provider (see section 10.4 – *Entity Authentication*), check that a Make Selectable Token is present in the INSTALL [for make selectable] command.
- If a Token is present in the INSTALL [for make selectable] command, request the OPEN to obtain verification of the Make Selectable Token.
- If the Application Provider identifier is present in the make selectable request, request the OPEN to save this in the GlobalPlatform Registry for the Application.
- Request the OPEN to obtain a Make Selectable Receipt.

On receipt of the make selectable request, the OPEN shall:

- Check that the card Life Cycle State is not CARD_LOCKED or TERMINATED;
- Check that OPEN and the requesting on-card entity have no restriction for make selectable;
- Check that the requesting on-card entity is a Security Domain with Delegated Management or Authorized Management privilege;
- Check that the Application AID is present in the GlobalPlatform Registry;
- If the Security Domain making the Application selectable is not the associated Security Domain of the Application, check that the Security Domain associated with the Application accepts to make it selectable;
- At the request of the Security Domain making the Application selectable, request the Security Domain with Token Verification privilege to authorize the make selectable (e.g. to verify the Make Selectable Token);
- Make the Application selectable according to the underlying runtime environment requirements;
- If the Security Domain making the Application selectable has Delegated Management privilege, ensure that the Security Domain with Token Verification privilege has successfully verified a Token;
- Update accordingly the GlobalPlatform Registry entry for the Application (e.g. Privileges, Implicit Selection parameters);
- At the request of the Security Domain making the Application selectable, request the Security Domain with Receipt Generation privilege to generate a Make Selectable Receipt.

At the request of OPEN, the associated Security Domain shall:

- Apply the Security Domain Provider’s policy to accept or reject making this Application selectable;

- Apply its own security policy; e.g. check that its Life Cycle State is PERSONALIZED (only applicable to a Security Domain other than the Issuer Security Domain).

At the request of OPEN, the Security Domain with Token Verification privilege shall:

- Verify the Make Selectable Token.

At the request of OPEN, the Security Domain with Receipt Generation privilege shall:

- Apply the issuer's policy to generate or not a Make Selectable Receipt.

9.3.8 Card Content Combined Loading, Installation and Make Selectable Process

The phases in Figure 9-1 are combined into a single process that uses a combination of multiple occurrences of two different APDU commands (INSTALL and LOAD). The following sequence of APDU commands apply to the loading:

A first INSTALL [for load, install and make selectable] command serves as the combined load and install request for loading and installation. The INSTALL [for load, install and make selectable] command data field details the requirements regarding a Load File.

Multiple LOAD commands are then used to transport the Load File in blocks according to the size of the file and the communications buffer size of the card.

A last INSTALL [for load, install and make selectable] command serves as the combined load and install commit for loading and installation. The last INSTALL [for load, install and make selectable] command data field details the requirements regarding the Application being installed.

Each INSTALL or LOAD command is processed by the receiving Security Domain before forwarding the load request and Load File Data Block to the OPEN for processing.

The following runtime behavior requirements apply during the content combined loading and installation process.

Combined Load, Install and Make Selectable Request Runtime Behavior

On receipt of an INSTALL [for load, install and make selectable] command, the Security Domain performing the combined load and install shall:

- Apply its own secure communication policy. (Note: A minimum security level is defined in Table 11-2.)
- Apply its own security policy; e.g. check that its Life Cycle State is PERSONALIZED (only applicable to a Security Domain other than the Issuer Security Domain);
- If a Load File Data Block Hash is present in the INSTALL [for load, install and make selectable] command, request the OPEN to initiate the hash verification of the subsequent Load File Data Block;
- If the Application Provider identifier is present in the load request, request the OPEN to save this in the GlobalPlatform Registry for the Executable Load File.

On receipt of a combined load and installation request (arising from an INSTALL [for load, install and make selectable] command), the OPEN shall:

- Check that the card Life Cycle State is not CARD LOCKED or TERMINATED;
- Check that OPEN and the requesting on-card entity have no restriction for load, installation and make selectable;
- Check that the requesting on-card entity is a Security Domain with Delegated Management or Authorized Management privilege;
- Check that the AID of the Load File is not already present in the GlobalPlatform Registry as an Executable Load File or Application;

- If an associated Security Domain AID is present, and is not the Security Domain performing the combined load, install and make selectable, check that this AID exists within the GlobalPlatform Registry and is registered with the Security Domain privilege. As this equates to the extradition of the Load File, check that the associated Security Domain accepts this extradition and combined load, install and make selectable operation. If no associated Security Domain AID is indicated, the Security Domain performing the load is by default the associated Security Domain.

At the request of OPEN, the associated Security Domain shall:

- Apply the Security Domain Provider's policy to accept or reject this combined load, install and make selectable operation;
- Apply its own security policy; e.g. check that its Life Cycle State is PERSONALIZED (only applicable to a Security Domain other than the Issuer Security Domain).

At the request of OPEN, the Security Domain to which the new Executable Load File shall be extradited (if any) shall:

- Apply its own policy to accept or reject this extradition;
- Apply its own security policy; e.g. check that its Life Cycle State is PERSONALIZED (only applicable to a Security Domain other than the Issuer Security Domain).

Load Phase Runtime Behavior

On receipt of the LOAD commands, the Security Domain performing the load shall:

- Apply its own secure communication policy. (Note: A minimum security level is defined in Table 11-2.)
- Discover whether any Security Domain has the Mandated DAP Verification privilege and is in the PERSONALIZED state, and if so:
  - Ensure that the required authentication data (DAP Block identifying the above Security Domain) is present in the Load File.
- Check if the associated Security Domain has the DAP Verification privilege and if so:
  - Ensure that the required authentication data (DAP Block identifying the associated Security Domain) is present in the Load File.
- If authentication data (one or more DAP Blocks) is present in the Load File:
  - Ensure that a Load File Data Block Hash was received during the combined load, install and make selectable request process;
  - Extract the authentication data (one or more DAP Blocks) from the Load File;
  - For each DAP Block of the Load File, request the OPEN to obtain verification of the DAP by the Security Domain indicated in the DAP Block.

On receipt of the Load File the OPEN shall:

- Verify the resource requirements of the Load File (see section 9.7 – *Memory Resource Management*) and that sufficient card resources are available;
- Check that each DAP verification request from the Security Domain performing the load relates to a Security Domain present in the GlobalPlatform Registry with DAP or Mandated DAP Verification privilege and if so request the Security Domain to verify the DAP;
- Compute the hash of the Load File Data Block when verification of a DAP Block or a combined Load, Install and Make Selectable Token is requested;

At the request of OPEN, the Security Domain(s) verifying the DAP(s) shall:

- Verify that the DAP matches with the Load File Data Block Hash received in the load request.

On completion of the load process the OPEN shall:

- At the request of the Security Domain performing the load, verify the Load File Data Block Hash received in the combined load, install and make selectable request;
- Check in the GlobalPlatform Registry if any Security Domain has the Mandated DAP Verification privilege and is in the PERSONALIZED state, and if so:
  - Ensure that the above Security Domain has successfully verified a DAP;
- Check in the GlobalPlatform Registry if the associated Security Domain has the DAP Verification privilege and if so:
  - Ensure that the associated Security Domain has successfully verified a DAP;
- If one or more DAP verifications were performed, verify the Load File Data Block Hash received in the load request.

Combined Load, Install and Make Selectable Completion Runtime Behavior

On receipt of the last INSTALL [for load, install and make selectable] command, the Security Domain performing the combined load and install shall:

- Apply its own secure communication policy. (Note: A minimum security level is defined in Table 11-2.)
- If the Security Domain performing the installation has the Authorized Management privilege and the off-card entity at the origin of the installation request is not authenticated as its Security Domain Provider (see section 10.4 – *Entity Authentication*), check that a Load, Install and Make Selectable Token is present in the INSTALL [for load, install and make selectable] command;
- If a Token is present in the INSTALL [for load, install and make selectable] command, request the OPEN to obtain verification of the Load, Install and Make Selectable Token;
- If the Application Provider identifier is present in the combined load, install and make selectable request, request the OPEN to save this in the GlobalPlatform Registry for the Executable Load File and the Application;
- Request the OPEN to obtain a combined Load, Install and Make Selectable Receipt.

On completion of the load, install and make selectable process the OPEN shall:

- If the Security Domain performing the combined load, install and make selectable has Delegated Management privilege, ensure that the Security Domain with Token Verification privilege has successfully verified a Token;
- Create an Executable Load File using the Load File Data Block;
- Create an entry in the GlobalPlatform Registry for the Executable Load File indicating its associated Security Domain;
- Create an entry for each Executable Module within the Executable Load File in the GlobalPlatform Registry. This shall include the Application Provider identifier if requested by the Security Domain in the combined load, install and make selectable request. The associated Security Domain for each Executable Module shall be the same as the associated Security Domain for the Executable Load File;
- Check that the Application AID (for future selection of the Application) is not already present in the GlobalPlatform Registry as an Application or Executable Load File;
- Perform the installation of the Application according to the underlying runtime environment requirements;
- Create an Application from the Executable Module;

- Ensure that the Application, depending on the underlying runtime environment, has the knowledge of its AID, its Privileges and its Install Parameters;
- Create an entry in the GlobalPlatform Registry for the Application indicating its associated Security Domain, Life Cycle State, Privileges and, when present in the combined load, install and make selectable request, Implicit Selection, Service and Memory Resource Management parameters; and including the Application Provider identifier if supplied by the Security Domain in the combined load, install and make selectable request;
- At the request of the Security Domain performing the combined load, install and make selectable, request the Security Domain with Receipt Generation privilege to generate a Load, Install and Make selectable Receipt;
- Verify the resource requirements indicated for the Application (see section 9.7 – *Memory Resource Management*) and that sufficient card resources are available.

At the request of OPEN, the Security Domain with Token Verification privilege shall:

- Verify the combined Load, Install and Make Selectable Token.

At the request of OPEN, the Security Domain with Receipt Generation privilege shall:

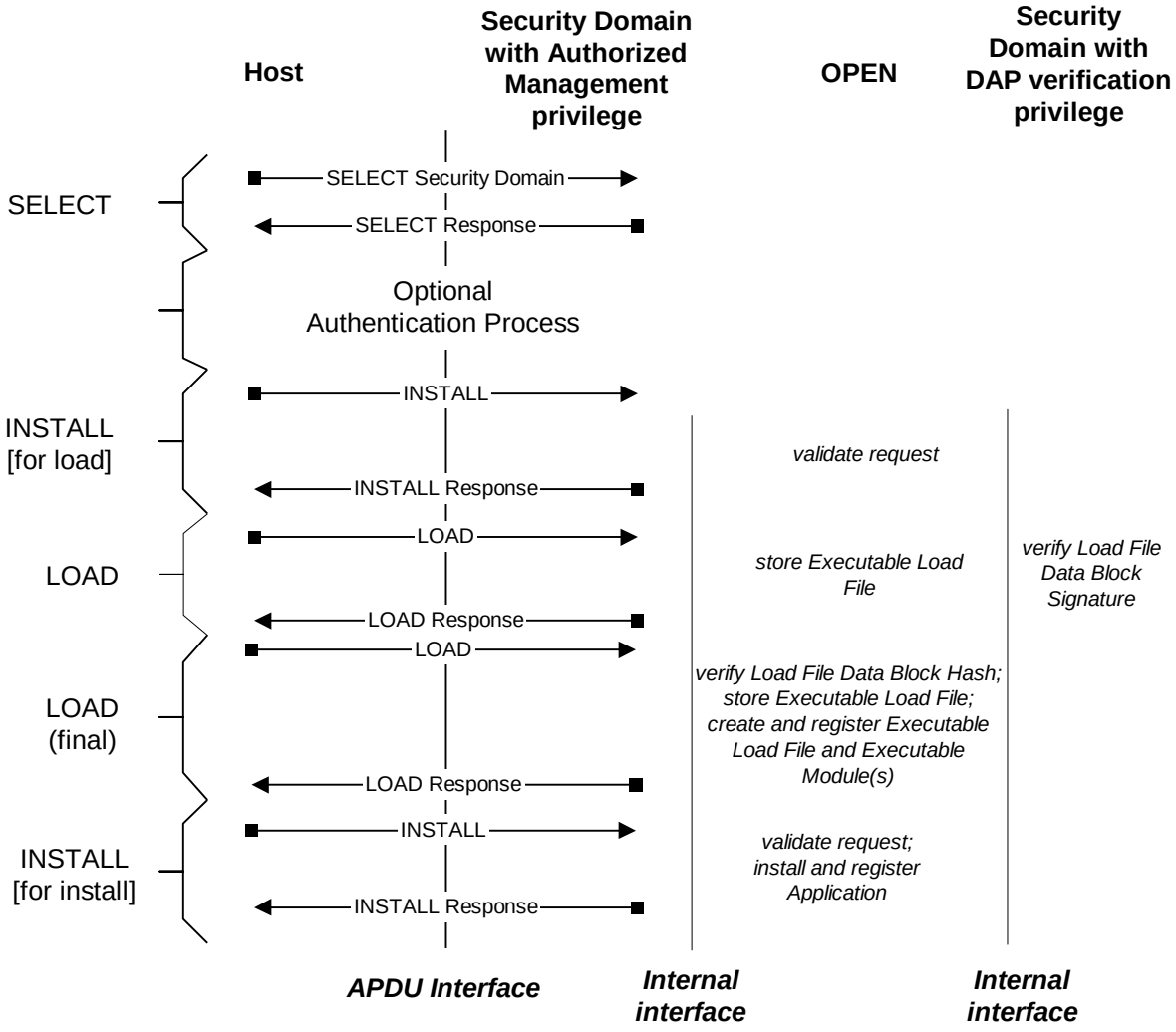
- Apply the issuer's policy to generate or not a combined Load, Install and Make Selectable Receipt.

If, at any stage, the OPEN determines that card resources are insufficient for the loading process or that any verification step has failed, the OPEN shall terminate the loading process, shall return the appropriate error and shall reclaim any memory allocated to the load process.

9.3.9 Examples of Loading and Installation Flow

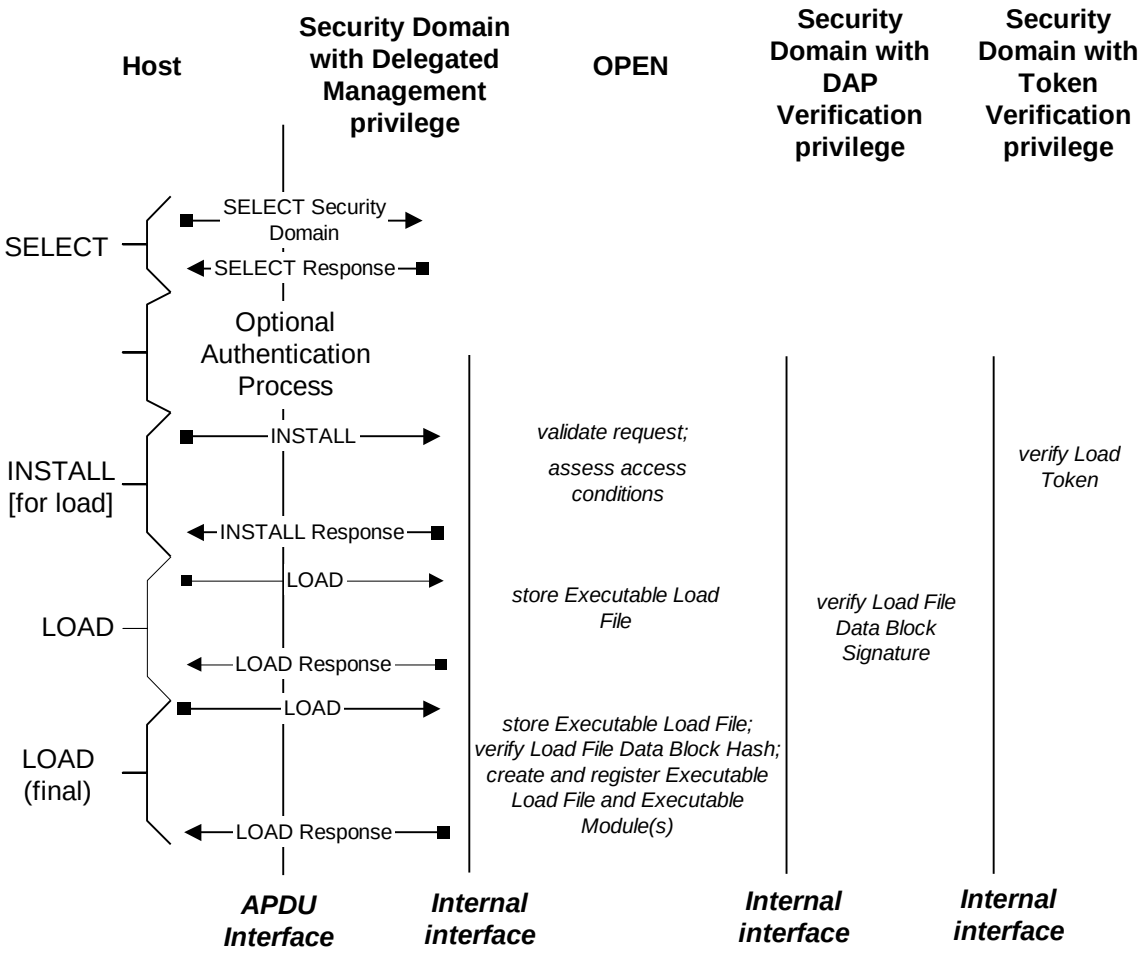
The following diagram is an example of the loading and installing of an Application to a GlobalPlatform card. In this example loading and installation are performed by a Security Domain with Authorized Management privilege. The Load File is loaded on the card and stored in memory as an Executable Load File. The installation phase occurs immediately following the loading phase.

Figure 9-2: Load and Installation Flow Diagram



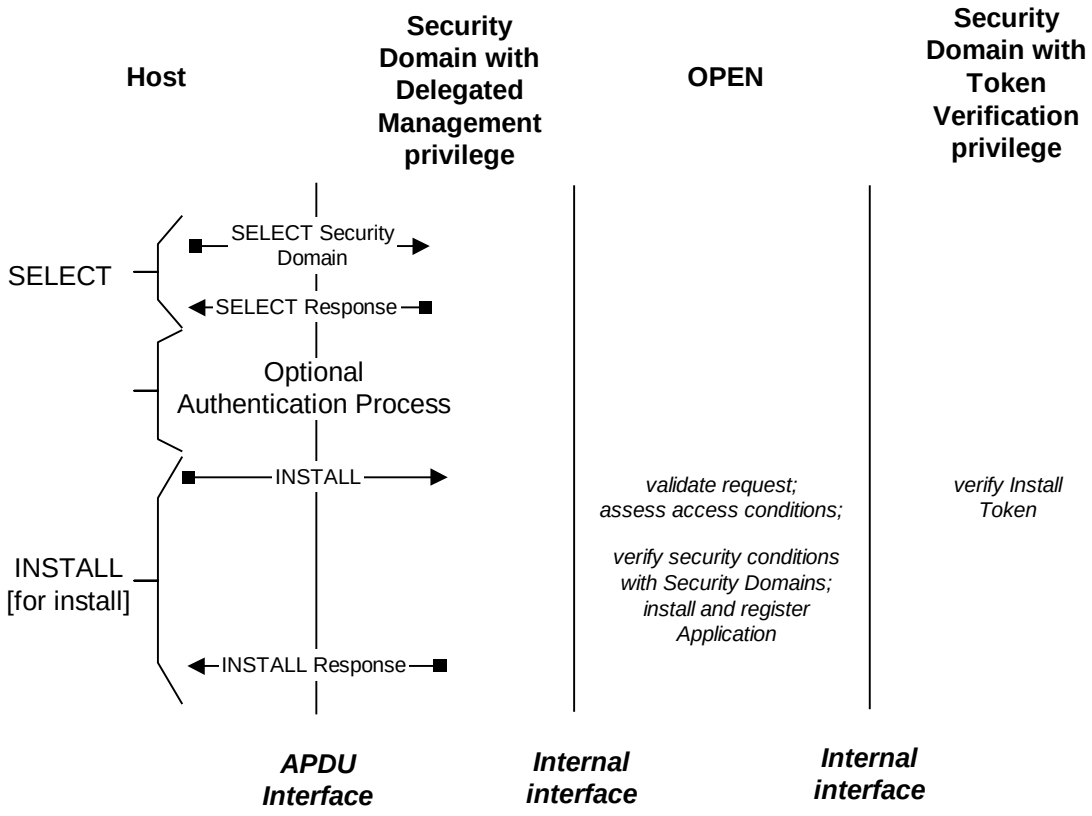
The following diagram is an example of the loading of an Application to a GlobalPlatform card. In this example loading is performed by a Security Domain with Delegated Management privilege.

Figure 9-3: Load Flow Diagram



The following diagram is an example of installing an Application from an Executable Load File already present on the card. In this example loading and installation are performed by a Security Domain with Delegated Management privilege.

Figure 9-4: Install Flow Diagram



9.4 Content Extradition and Registry Update

9.4.1 Content Extradition

The GlobalPlatform Card Content extradition process is designed to allow a previously installed Application or a previously loaded Executable Load File to be associated with a different Security Domain.

The extradition may apply at any time during the Application Life Cycle. Extradition may apply for any Security Domain (in the PERSONALIZED state) or the Issuer Security Domain (in any card Life Cycle State other than CARD_LOCKED and TERMINATED), that accepts the extradited Application.

The extradition process is triggered by an INSTALL [for extradition] command. The Security Domain processing the command then passes the extradition request to the OPEN for additional verification and processing. Using the INSTALL [for extradition] command, it is forbidden to extradite an Executable Load File to a Security Domain having the DAP Verification privilege.

If the Security Domain processing the INSTALL [for extradition] command has the Delegated Management privilege, then an Extradition Token must be present in the command. In this case, the Extradition Token allows the OPEN, via the Security Domain with Token Verification privilege within the same sub-hierarchy as the Security Domain performing the extradition, to ensure that the token authorized the extradition process. Following the completion of the extradition process, an optional Extradition Receipt is returned to the Security Domain performing the Delegated Management operation and shall be transmitted by the Security Domain to the off-card entity. The Application Provider may then forward the Extradition Receipt to the corresponding off-card entity as a proof that the extradition process was successfully performed. The purpose of the optional Extradition Receipt is to assist the Application Provider in keeping its Card Management System synchronized with its on-card application base.

The following runtime behavior requirements apply during the Card Content extradition process.

Runtime Behavior

On receipt of the INSTALL [for extradition] command, the Security Domain performing the extradition shall:

- Apply its own secure communication policy. (Note: A minimum security level is defined in Table 11-2.)
- Apply its own security policy; e.g. check that its Life Cycle State is PERSONALIZED (only applicable to a Security Domain other than the Issuer Security Domain);
- If the Security Domain performing the extradition has the Authorized Management privilege and the off-card entity at the origin of the extradition request is not authenticated as its Security Domain Provider (see section 10.4 – *Entity Authentication*), check that an Extradition Token is present in the INSTALL [for extradition] command;
- If a Token is present in the INSTALL [for extradition] command, request the OPEN to obtain verification of the Extradition Token;
- Request the OPEN to obtain an Extradition Receipt.

On receipt of an extradition request, the OPEN shall:

- Check that the card Life Cycle State is not CARD_LOCKED or TERMINATED;
- Check that OPEN and the requesting on-card entity have no restriction for extradition;
- Determine if the Application or Executable Load File being extradited exists within the GlobalPlatform Registry;
- Check that the requesting on-card entity is a Security Domain with Delegated Management or Authorized Management privilege;
- Check that an on-card entity with the same AID as the Security Domain to which the Application or Executable Load File is being extradited exists within the GlobalPlatform Registry, and that this on-card entity has the Security Domain privilege;
- Check that the Security Domain to which an Executable Load File is being extradited does not have the DAP Verification privilege;
- If the Security Domain performing the extradition has the Delegated Management privilege, ensure that the Security Domain with Token Verification privilege has successfully verified a Token;
- Update accordingly the GlobalPlatform Registry entry for the Application or Executable Load File;
- At the request of the Security Domain performing the extradition, request the Security Domain with Receipt Generation privilege to generate an Extradition Receipt.

At the request of OPEN, the currently associated Security Domain shall:

- Apply the Security Domain Provider’s policy to accept or reject this extradition request;

- Apply its own security policy; e.g. check that its Life Cycle State is PERSONALIZED (only applicable to a Security Domain other than the Issuer Security Domain).

At the request of OPEN, the Security Domain with Token Verification privilege shall:

- Verify the Extradition Token.

At the request of OPEN, the Security Domain with Receipt Generation privilege shall:

- Apply the issuer's policy to generate or not an Extradition Receipt.

At the request of OPEN, the Security Domain accepting the explicit extradition shall:

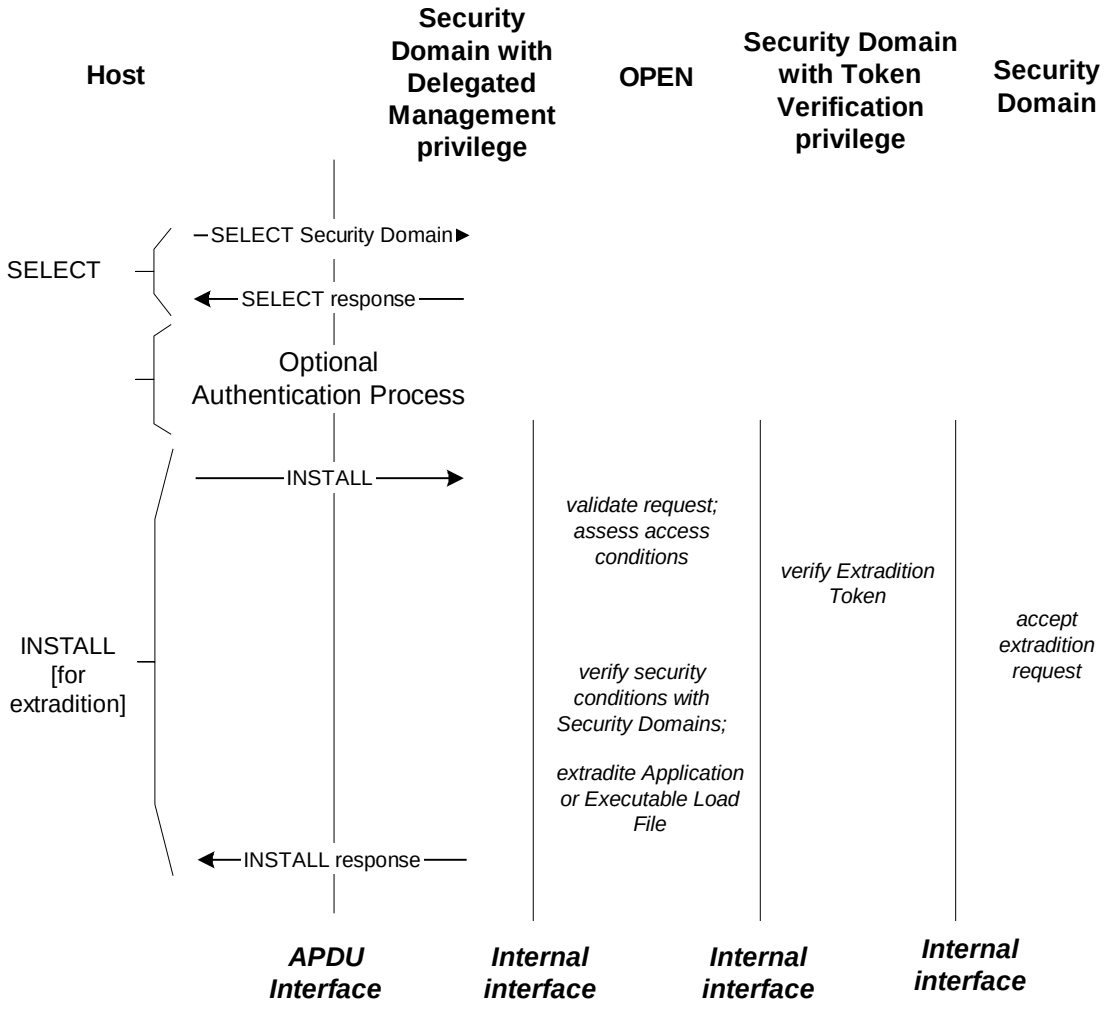
- Apply the Security Domain Provider's policy to accept or reject this Card Content extradition;
- Apply its own security policy; e.g. check that its Life Cycle State is PERSONALIZED (only applicable to a Security Domain other than the Issuer Security Domain).

The Security Domain requesting the extradition does not need to be associated with the Application or Load File being extradited. The Security Domain initially associated with the Application may accept or reject the extradition as detailed in a configuration.

Extradition Flow

The following figure is an example of extradition, and shows delegated extradition:

Figure 9-5: Delegated Extradition Flow



9.4.2 Registry Update

9.4.2.1 Generic Registry Update

The registry update process allows GlobalPlatform Registry data associated with an Application, such as Privileges and Implicit Selection parameters, to be modified. This process also allows the restricting of Card Content Management functionality of a specific Security Domain or OPEN itself (i.e. of all existing Security Domains present on the card and any eventual Security Domain installed afterwards).

The registry update may apply at any time during the Application Life Cycle or card Life Cycle (other than CARD_LOCKED or TERMINATED).

The registry update process comprises one or more INSTALL [for registry update] commands processed by the receiving Security Domain. To restrict Card Content Management functionality of OPEN, no Application AID is provided in the INSTALL [for registry update] command. The Security Domain then passes the registry update request to the OPEN for additional verification and processing.

The Registry Update Token allows the OPEN, via the Security Domain with Token Verification privilege within the same sub-hierarchy as the Security Domain performing the Registry Update, to ensure that the token authorizes the update of the GlobalPlatform Registry.

The response to the INSTALL [for registry update] command identifies the end of the registry update process. Following the completion of the registry update process, an optional Registry Update Receipt is returned to the Security Domain performing the Delegated Management operation and shall be transmitted by the Security Domain to the off-card entity.

The Application Provider may then forward the Registry Update Receipt to the corresponding off-card entity as a proof that the registry update process was successfully performed. The purpose of the optional Registry Update Receipt is to assist the Application Provider in keeping its Card Management System synchronized with its on-card application base.

The following runtime behavior requirements apply during the registry update process.

Runtime Behavior

On receipt of the INSTALL [for registry update] command, the Security Domain performing the registry update shall:

- Apply its own secure communication policy. (Note: A minimum security level is defined in Table 11-2.)
- Apply its own security policy; e.g. check that its Life Cycle State is PERSONALIZED (only applicable to a Security Domain other than the Issuer Security Domain);
- If the Security Domain performing the registry update has the Authorized Management privilege and the off-card entity at the origin of the registry update request is not authenticated as its Security Domain Provider (see section 10.4 – *Entity Authentication*), check that a Registry Update Token is present in the INSTALL [for registry update] command;
- If a Token is present in the INSTALL [for registry update] command, request the OPEN to obtain verification of the Registry Update Token;
- Request the OPEN to obtain a Registry Update Receipt.

On receipt of a registry update request, the OPEN shall:

- Check that the card Life Cycle State is not CARD_LOCKED or TERMINATED;
- Check that OPEN and the requesting on-card entity have no restriction for registry update;
- When updating an Application, determine if the Application being updated exists within the GlobalPlatform Registry;
- Check that the requesting on-card entity is a Security Domain with Delegated Management or Authorized Management privilege;
- When restricting functionality of OPEN, check that the requesting on-card entity is a Security Domain with Global Lock privilege;
- If the Security Domain requesting the registry update is not the associated Security Domain of the Application, check that the Security Domain associated with the Application accepts this registry update;
- If the Security Domain performing the registry update has the Delegated Management privilege, ensure that the Security Domain with Token Verification privilege has successfully verified a Token;
- Update accordingly the GlobalPlatform Registry entry for the Application being updated;
- At the request of the Security Domain performing the registry update, request the Security Domain with Receipt Generation privilege to generate a Registry Update Receipt.

At the request of OPEN, the associated Security Domain shall:

- Apply the Security Domain Provider’s policy to accept or reject this registry update;

- Apply its own security policy; e.g. check that its Life Cycle State is PERSONALIZED (only applicable to a Security Domain other than the Issuer Security Domain).

At the request of OPEN, the Security Domain with Token Verification privilege shall:

- Verify the Registry Update Token.

At the request of OPEN, the Security Domain with Receipt Generation privilege shall:

- Apply the issuer's policy to generate or not a Registry Update Receipt.

9.4.2.2 Extradition using Registry Update

An extradition may be performed using the registry update process. The simultaneous extradition and registry update process is achieved by an appropriately formed INSTALL [for registry update] command.

The runtime behavior requirements for an extradition using registry update are the sum of the runtime behavior requirements for general registry update as well as the additional runtime behavior requirements specified below.

If a Token is required the Registry Update Token is used, and if a Receipt is to be returned the Registry Update Receipt is used; both of which allow for extradition as well as for registry update.

Additional Runtime Behavior

At the request of OPEN, the currently associated Security Domain shall:

- Apply the Security Domain Provider's policy to accept or reject this extradition request;
- Apply its own security policy; e.g. check that its Life Cycle State is PERSONALIZED (only applicable to a Security Domain other than the Issuer Security Domain).

At the request of the OPEN, the Security Domain accepting the explicit extradition shall:

- Apply the Security Domain Provider's policy to accept or reject this Card Content extradition;
- Apply its own security policy; e.g. check that its Lifecycle State is PERSONALIZED (only applicable to a Security Domain other than the Issuer Security Domain).

9.5 Content Removal

This section defines the content removal process that enables a flexible management of the Mutable Persistent Memory space of the cards through the removal of Applications and/or Executable Load Files. Only code and data not referenced by another entity on the card may be deleted. Code and data that is currently being accessed by the runtime environment is deemed to be referenced by another entity. If the Application is selected on another logical channel this Application cannot be deleted.

The DELETE command (see section 11.2 – *DELETE Command*) allows for the actual removal of content from Mutable Persistent Memory and the logical removal of content from Immutable Persistent Memory. The DELETE command is processed by the receiving Security Domain before forwarding the removal request to the OPEN for processing.

Depending on the memory location of the removed Applications and/or Executable Load File, the OPEN shall perform the different actions detailed in the following sections. When the Application or Executable Load File has been successfully removed, its contents in the memory shall no longer be accessible.

A Security Domain with Global Delete privilege has the privilege to delete any Application or Executable Load File from the card regardless of which Security Domain the Application or Executable Load File is associated with.

The Delete Token allows the OPEN, via the Security Domain with the Token Verification privilege within the same sub-hierarchy as the Security Domain performing the Delete, to ensure that the token authorizes the deletion of the associated GlobalPlatform Registry entries.

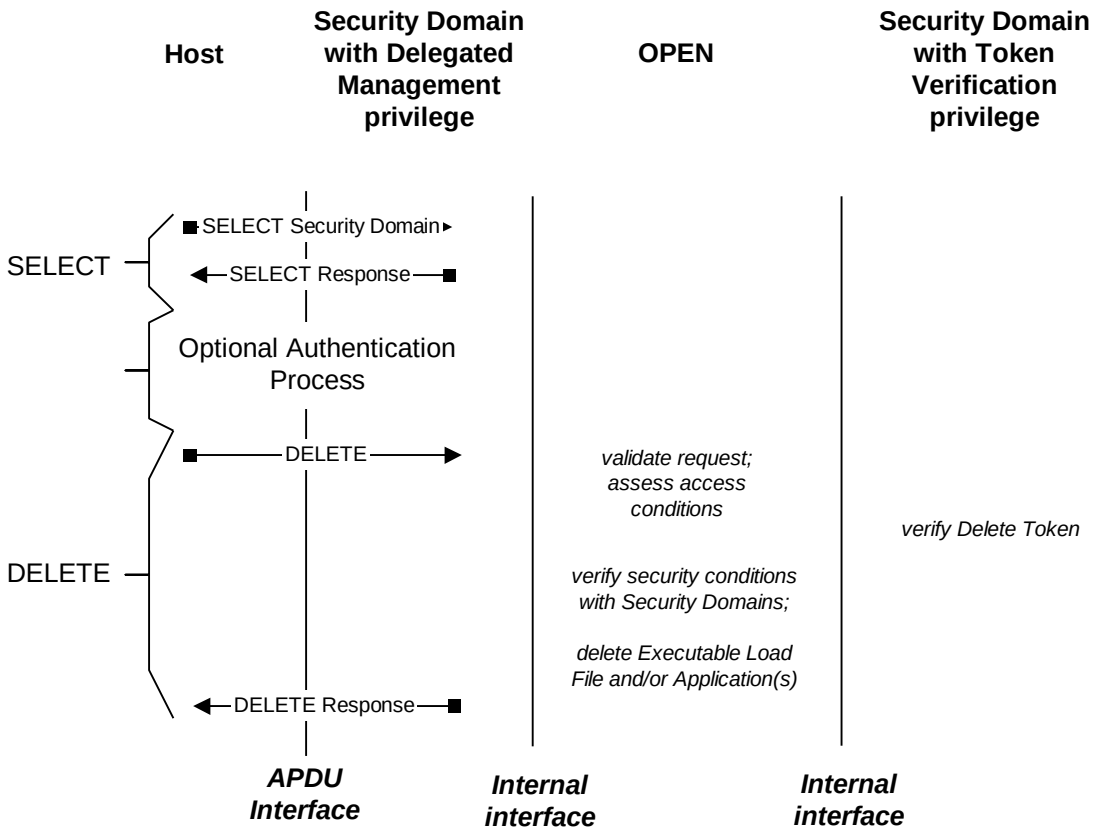
The response to the DELETE command identifies the end of the deletion process. Following the completion of the deletion process, an optional Delete Receipt is returned to the Security Domain performing the Delegated Management operation and shall be transmitted by the Security Domain to the off-card entity.

The Application Provider may then forward the Delete Receipt to the corresponding off-card entity as a proof that the deletion process was successfully performed. The purpose of the optional Delete Receipt is to assist the Application Provider in keeping its Card Management System synchronized with its on-card application base.

Deletion Flow

The following figure is an example of deleting an Application, an Executable Load File, or both:

Figure 9-6: Content Deletion Flow



9.5.1 Application Removal

Application removal may involve the removal of Application instances as well as any Application data associated with the Application.

The following runtime behavior requirements apply during the Application removal process.

Runtime Behavior

On receipt of an Application deletion request (DELETE command), the Security Domain performing the deletion shall:

- Apply its own secure communication policy. (Note: A minimum security level is defined in Table 11-2.)
- Apply its own security policy; e.g. check that its Life Cycle State is PERSONALIZED (applicable to a Security Domain other than the Issuer Security Domain);
- If the Security Domain performing the deletion has the Authorized Management privilege and the off-card entity at the origin of the delete request is not authenticated as its Security Domain Provider (see section 10.4 – *Entity Authentication*), check that a Delete Token is present in the DELETE command;

- If a Token is present in the DELETE command, request the OPEN to obtain verification of the Delete Token;
- Request the OPEN to obtain a Delete Receipt.

On receipt of an Application deletion request, the OPEN shall:

- Check that the card Life Cycle State is not CARD_LOCKED or TERMINATED;
- Check that OPEN and the requesting on-card entity have no restriction for deletion;
- Determine that the Application being deleted has an entry within the GlobalPlatform Registry;
- Check that the requesting on-card entity is a Security Domain with Delegated Management or Authorized Management privilege;
- Check that the Security Domain performing the deletion is the associated Security Domain of the Application being deleted, or that it has Global Delete privilege. Otherwise (i.e. if the Security Domain performing the deletion is not associated with the Application being deleted and does not have the Global Delete privilege), check that the Security Domain associated with the Application accepts this deletion;
- At the request of the Security Domain performing the deletion, request the Security Domain with Token Verification privilege to verify the Delete Token;
- Determine that the Application is not currently selected on another logical channel;
- Determine that no other Applications present in the card make references to this Application;
- Determine that no other Applications present in the card make references to any data within this Application;
- If a Security Domain is being deleted, determine that none of the Applications or Executable Load Files present in the card are associated with this Security Domain;
- If the Security Domain performing the deletion has the Delegated Management privilege, ensure that the Security Domain with Token Verification privilege has successfully verified a token;
- Remove the entry for the Application from within the GlobalPlatform Registry;
- Re-assign the Application's privileges (if any) to the Issuer Security Domain as defined in section 6.6.2 – Privilege Assignment;
- If the Application is implicitly selectable, re-assign implicit selection to the Issuer Security Domain as defined in section 6.4.1 – *Implicit Selection Assignment*;
- Release and mark as available any Mutable Persistent Memory and when supported, apply the memory resource management rules described in section 9.7 – *Memory Resource Management*;
- At the request of the Security Domain performing the deletion, request the Security Domain with Receipt Generation privilege to generate a Delete Receipt.

If the OPEN determines that any of the above verification steps have failed, the OPEN shall not initiate the delete process and shall inform the Security Domain to return the appropriate response. Once this delete process begins it shall complete in the current Card Session or, in the event of an interruption, at least the updates to the GlobalPlatform Registry shall complete in a subsequent Card Session.

At the request of OPEN, the associated Security Domain shall:

- Apply the Security Domain Provider's security policy to accept or reject this Application removal;
- Apply its own security policy e.g. check that its Life Cycle State is PERSONALIZED (only applicable to a Security Domain other than the Issuer Security Domain).

At the request of OPEN, the Security Domain with Token Verification privilege shall:

- Apply the issuer's policy to accept or reject a deletion authorization request without the presence of a Delete Token;
- Verify the Delete Token.

At the request of OPEN, the Security Domain with Receipt Generation privilege shall:

- Apply the issuer's policy to generate or not a Delete Receipt.

9.5.2 Executable Load File Removal

An Executable Load File contains Executable Modules. The removal applies to the entire Executable Load File. Physical removal may occur in Mutable Persistent Memory while only logical removal is possible in Immutable Persistent Memory.

This version of the Specification does not cover the removal of a specific Executable Module within an Executable Load File.

The following runtime behavior requirements apply during the Executable Load File removal process.

Runtime Behavior

On receipt of an Executable Load File deletion request (DELETE command), the Security Domain performing the deletion shall:

- Apply its own secure communication policy. (Note: A minimum security level is defined in Table 11-2.)
- Apply its own security policy; e.g. check that its Life Cycle State is PERSONALIZED (only applicable to a Security Domain other than the Issuer Security Domain);
- If the Security Domain performing the deletion has the Authorized Management privilege and the off-card entity at the origin of the delete request is not authenticated as its Security Domain Provider (see section 10.4 – *Entity Authentication*), check that a Delete Token is present in the DELETE command;
- If a Token is present in the DELETE command, request the OPEN to obtain verification of the Delete Token;
- Request the OPEN to obtain a Delete Receipt.

On receipt of an Executable Load File removal process deletion request, the OPEN shall:

- Check that the card Life Cycle State is not CARD_LOCKED or TERMINATED;
- Check that OPEN and the requesting on-card entity have no restriction for deletion;
- Check that the requesting on-card entity is a Security Domain with Delegated Management or Authorized Management privilege;
- Check that the Security Domain performing the deletion is the associated Security Domain of the Executable Load File being deleted, or that it has Global Delete privilege. Otherwise (i.e. if the Security Domain performing the deletion is not associated with the Executable Load File being deleted and does not have the Global Delete privilege) check that the Security Domain associated with the Executable Load File accepts this deletion;
- At the request of the Security Domain performing the deletion, request the Security Domain with Token Verification privilege to verify the Delete Token;
- Determine that the Executable Load File being deleted has an entry within the GlobalPlatform Registry;
- Determine that no other Applications and no other Executable Load Files present in the card maintain references to this Executable Load File;

- If the Security Domain performing the deletion has the Delegated Management privilege, ensure that the Security Domain with Token Verification privilege has successfully verified a Token;
- Remove the entry for the Executable Load File and the entries for any Executable Modules present in the Executable Load File from within the GlobalPlatform Registry;
- Release and mark as available any Mutable Persistent Memory and when supported, apply the memory resource management rules described in section 9.7 – *Memory Resource Management*;
- At the request of the Security Domain performing the deletion, request the Security Domain with Receipt Generation privilege to generate a Delete Receipt.

If the OPEN determines that any of the above verification steps have failed, the OPEN shall not initiate the delete process and shall inform the Security Domain to return the appropriate response. Once this delete process begins it shall all complete in the current Card Session or, in the event of an interruption, at least the updates to the GlobalPlatform Registry shall complete in a subsequent Card Session.

Only Mutable Persistent Memory is released and marked as available. Executable Load Files contained in Immutable Persistent Memory cannot be deleted but the entry for the Executable Load File and the entries for the Executable Modules present in the Executable Load File shall be deleted from the GlobalPlatform Registry.

At the request of OPEN, the associated Security Domain shall:

- Apply the Security Domain Provider's policy to accept or reject this Executable Load File removal;
- Apply its own security policy; e.g. check that its Life Cycle State is PERSONALIZED (only applicable to a Security Domain other than the Issuer Security Domain).

At the request of OPEN, the Security Domain with Token Verification privilege shall:

- Apply the issuer's policy to accept or reject a deletion authorization request without the presence of a Delete Token;
- Verify the Delete Token.

At the request of OPEN, the Security Domain with Receipt Generation privilege shall:

- Apply the issuer's policy to generate or not a Delete Receipt.

9.5.3 Executable Load File and related Application Removal

An Executable Load File contains Executable Modules from which Applications have been installed. This optional feature removes the Executable Load File as well as all these related Applications. Physical removal may occur in Mutable Persistent Memory while only logical removal is possible in Immutable Persistent Memory.

When supported by the card, the following runtime behavior requirements apply during the Executable Load File and related Application removal process.

Runtime Behavior

On receipt of an Executable Load File deletion request (DELETE command), the Security Domain performing the deletion shall:

- Apply its own secure communication policy. (Note: A minimum security level is defined in Table 11-2.)
- Apply its own security policy; e.g. check that its Life Cycle State is PERSONALIZED (only applicable to a Security Domain other than the Issuer Security Domain);

- If the Security Domain performing the deletion has the Authorized Management privilege and the off-card entity at the origin of the delete request is not authenticated as its Security Domain Provider (see section 10.4 – *Entity Authentication*), check that a Delete Token is present in the DELETE command;
- If a Token is present in the DELETE command, request the OPEN to obtain verification of the Delete Token;
- Request the OPEN to obtain a Delete Receipt.

On receipt of a request to remove an Executable Load File and its related Applications, the OPEN shall:

- Check that the card Life Cycle State is not CARD_LOCKED or TERMINATED;
- Check that OPEN and the requesting on-card entity have no restriction for deletion;
- Check that the requesting on-card entity is a Security Domain with Delegated Management or Authorized Management privilege;
- Check that the Security Domain performing the deletion is the associated Security Domain of each of the related Applications being deleted, or that it has the Global Delete privilege. Otherwise (i.e. if the Security Domain performing the deletion is not associated with one or more of the Applications being deleted and does not have the Global Delete privilege) check in each case that the Security Domain associated with the related Application accepts this deletion;
- Check that the Security Domain performing the deletion is the associated Security Domain of the Executable Load File being deleted, or that it has the Global Delete privilege. Otherwise (i.e. if the Security Domain performing the deletion is not associated with the Executable Load File being deleted and does not have the Global Delete privilege) check that the Security Domain associated with the Executable Load File accepts this deletion;
- At the request of the Security Domain performing the deletion, request the Security Domain with Token Verification privilege to verify the Delete Token;
- If the Security Domain performing the deletion has the Delegated Management privilege, ensure that the Security Domain with Token Verification privilege has successfully verified a Token;
- Determine that the Executable Load File and Applications being deleted have entries within the GlobalPlatform Registry;
- Locate each Application installed from an Executable Module within this Executable Load File and for each Application:
  - Determine that the Application is not currently selected on another logical channel;
  - Determine that no other non-related Applications present in the card make reference to this Application;
  - Determine that no other non-related Applications present in the card maintain references to any data within this Application;
  - If a Security Domain is being deleted, determine that none of the non-related Applications present in the card are associated with this Security Domain;
- Determine that no other Applications and no other Executable Load Files present in the card maintain references to this Executable Load File;
- Remove the entry for the Executable Load File and the entries for any Executable Modules present in the Executable Load File from within the GlobalPlatform Registry;
- Remove the entry for each related Application within the GlobalPlatform Registry;

- Re-assign the privileges of all related Applications (if any) to the Issuer Security Domain as defined in section 6.6.2 – Privilege Assignment;
- If any related Application is implicitly selectable, re-assign implicit selection to the Issuer Security Domain as defined in section 6.4.1 – *Implicit Selection Assignment*;
- Release and mark as available any Mutable Persistent Memory and when supported, apply the memory resource management rules described in section 9.7 – *Memory Resource Management*;
- At the request of the Security Domain performing the deletion, request the Security Domain with Receipt Generation privilege to generate a Delete Receipt.

If the OPEN determines that any of the above verification steps have failed, the OPEN shall not initiate the delete process and shall inform the Security Domain to return the appropriate response. Once the delete processes begin they shall all complete in the current Card Session or, in the event of an interruption, at least the updates to the GlobalPlatform Registry shall complete in a subsequent Card Session.

Only Mutable Persistent Memory is released and marked as available. Executable Load Files contained in Immutable Persistent Memory cannot be deleted but the entry for the Executable Load File and the entries for the Executable Modules present in the Executable Load File shall be deleted from the GlobalPlatform Registry.

At the request of OPEN, the associated Security Domain shall:

- Apply the Security Domain Provider's policy to accept or reject this Executable Load File and related Application removal;
- Apply its own security policy; e.g. check that its Life Cycle State is PERSONALIZED (only applicable to a Security Domain other than the Issuer Security Domain).

At the request of OPEN, the Security Domain with Token Verification privilege shall:

- Apply the issuer's policy to accept or reject a deletion authorization request without the presence of a Delete Token;
- Verify the Delete Token.

At the request of OPEN, the Security Domain with Receipt Generation privilege shall:

- Apply the issuer's policy to generate or not a Delete Receipt.

9.6 Security Management

9.6.1 Life Cycle Management

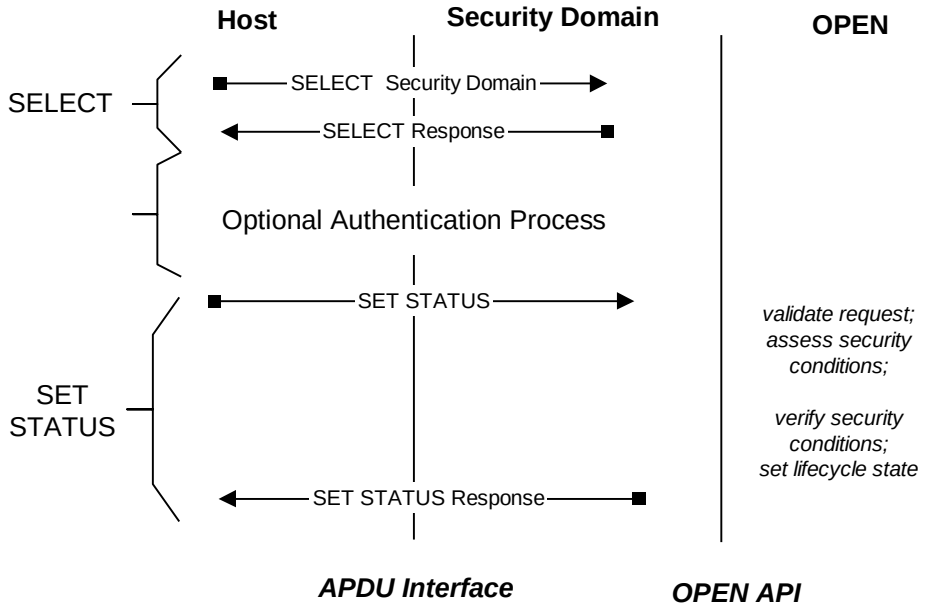
The following set of services is provided to manage the Life Cycles of on-card entities:

- Application locking and unlocking;
- Card locking and unlocking;
- Card termination;
- Application Life Cycle interrogation;
- Card Life Cycle interrogation.

Life Cycle Management Flow

The following figure shows the flow that is common to all the Life Cycle management services:

Figure 9-7: Life Cycle Management Flow



9.6.2 Application Locking and Unlocking

The card has a mechanism to disable and subsequently re-enable the continued execution status of an on-card Application. This mechanism may be invoked from within the OPEN based on exceptions handled by the OPEN or from the use of externally invoked commands. An Application with Global Lock privilege, the Application itself or a directly or indirectly associated Security Domain are the only entities that may initiate the locking of an Application. Only an Application with Global Lock privilege or a directly or indirectly associated Security Domain may initiate the unlocking of an Application.

Runtime Behavior

On receipt of a SET STATUS command, the Security Domain performing the lifecycle management shall:

- Apply its own secure communication policy. (Note: A minimum security level is defined in Table 11-2.)

- Check that the off-card entity at the origin of the lock/unlock request is authenticated as the Security Domain Provider (see section 10.4 – *Entity Authentication*).

On receipt of a request to lock or unlock an Application, the OPEN shall:

- Check that the requesting on-card entity is the Application itself, or the Security Domain directly or indirectly associated with the Application being locked or unlocked, or that the requesting entity has the Global Lock privilege;
- Check that the Application is not requesting to unlock itself;
- Check that an entry for the Application being locked or unlocked is present in the GlobalPlatform Registry and does not have the Final Application privilege;
- For locking, set the Application Life Cycle State to LOCKED;
- For unlocking, reverse the Application Life Cycle State to its previous state;
- For locking, keep a record of the previous Application Life Cycle State to ensure that the Application Life Cycle State can be transitioned back to and only to this previous state.

An Application being locked that is currently selected on a logical channel shall remain the currently selected Application on that logical channel until the end of the Application Session though its Application Life Cycle State is set to LOCKED.

Once the Application Life Cycle State is set to LOCKED, the Application shall not be selectable implicitly or explicitly on any logical channel.

If the locked Application is implicitly selectable on any card interface(s) and logical channel(s), the application with Final Application privilege will be implicitly selected for those card interface(s) and logical channel(s).

9.6.3 Card Locking and Unlocking

Once the card Life Cycle State is CARD_LOCKED, only the Application having the Final Application privilege may be selected. Due to the severity of this action, the card locking shall only be allowed under the most stringent conditions and shall only be performed with the proper security mechanisms and authorizations.

Card locking requests may originate from two sources:

Internal source — either the OPEN, the Issuer Security Domain, or a privileged Application may initiate card locking requests from within the card. Internal requests are likely to occur in response to certain card runtime situations. For example, the OPEN or an Application with the necessary privilege may initiate the card locking process as a response to a perceived security threat based on data collected from velocity checking data.

Off-card source — explicit card locking requests may be initiated by APDU commands sent by the Card Issuer to the Issuer Security Domain or by an authorized representative to an Application with the Card Lock Privilege.

Runtime Behavior

On receipt of a SET STATUS command, the Security Domain performing the lifecycle management shall:

- Apply its own secure communication policy. (Note: A minimum security level is defined in Table 11-2.)
- Check that the off-card entity at the origin of the lock/unlock request is authenticated as the Security Domain Provider (see section 10.4 – *Entity Authentication*).

On receipt of a request to lock the card, the OPEN shall:

- Check that the requesting on-card entity has the required Card Lock privilege;
- Check that the current card Life Cycle State is SECURED;

- Set the card Life Cycle State to CARD_LOCKED.

On receipt of a request to unlock the card, the OPEN shall:

- Check that the requesting on-card entity has the required Card Lock privilege;
- Reverse the card Life Cycle State to SECURED.

Applications that are currently selected on any logical channel shall remain the currently selected Applications on their respective logical channels until the end of their respective Application Sessions. As soon as the current Application Session on a Supplementary Logical Channel ends, the logical channel on which the Application was selected is closed. Once all the Supplementary Logical Channels are closed the Basic Logical Channel becomes the only available interface. As soon as the current Application Session on the Basic Logical Channel ends, the Application with the Final Application privilege is the only selectable Application.

Attempts to modify Card Content are denied from the instant the card transitions to the CARD_LOCKED Life Cycle State.

9.6.4 Card Termination

In the card Life Cycle State TERMINATED, all communication to the card are directed to the application with the Final Application privilege. If the Issuer Security Domain is the application with this privilege all commands except the GET DATA command processed by the Issuer Security Domain shall be disabled. Due to the severity of this action, the card termination shall only be allowed under the most stringent conditions and shall only be performed with the proper security mechanisms and authorizations.

The decision to terminate a card may either be triggered by an internal card event that violates a policy of the Card Issuer or may be invoked externally:

Internal source — either the OPEN, the Issuer Security Domain, or a privileged Application may initiate a card termination request from within the card. An internal card termination request is likely to occur in response to certain card runtime situations.

Off-card source — explicit card termination requests can be initiated by APDU commands sent by the Card Issuer to the Issuer Security Domain or by an authorized representative to an Application with the Card Terminate privilege.

Runtime Behavior

On receipt of a SET STATUS command, the Security Domain performing the lifecycle management shall:

- Apply its own secure communication policy. (Note: A minimum security level is defined in Table 11-2.)
- Check that the off-card entity at the origin of the termination request is authenticated as the Security Domain Provider (see section 10.4 – *Entity Authentication*).

On receipt of a request to terminate the card, the OPEN shall:

- Check that the requesting on-card entity has the Card Terminate privilege;
- Set the card Life Cycle State to TERMINATED from its current state.

Applications that are currently selected on any logical channel shall remain the currently selected Applications on their respective logical channels until the end of their respective Application Sessions. As soon as the current Application Session on a Supplementary Logical Channel ends, the logical channel on which the Application was selected is closed. Once all the Supplementary Logical Channels are closed the Basic Logical Channel becomes the only available interface. As soon as the current Application Session on the Basic Logical Channel ends, no Applications are selectable, the Application with Final Application privilege is always implicitly selected on the Basic Logical Channel.

As the transition to the TERMINATED Life Cycle State can be due to the detection of a severe security threat, the security policy of the Issuer may require additional behavior such as closing the current Card Session by performing an internal card reset.

Card Content management requests are denied from the instant the card transitions to the TERMINATED Life Cycle State. Depending on the security policy of the Issuer, other operations may be allowed to continue during the remaining current Application Sessions.

9.6.5 Application Status Interrogation

The status of an Application (or a Security Domain): its Life Cycle State, Privileges and other parameters registered in the GlobalPlatform Registry, may be accessed by suitably authorized entities.

Runtime Behavior

On receipt of a GET STATUS command, the Security Domain performing the lifecycle interrogation shall:

- Apply its own secure communication policy. (Note: A minimum security level is defined in Table 11-2.)
- Check that the off-card entity at the origin of the request is authenticated as the Security Domain Provider (see section 10.4 – *Entity Authentication*).

On receipt of a request to interrogate the status of an Application, the OPEN shall:

- Check that the requesting on-card entity is the entity itself being interrogated, a Security Domain directly or indirectly associated with the Application being interrogated, or that the requesting entity has the Global Registry privilege

9.6.6 Card Status Interrogation

The status of the card and the Issuer Security Domain: its AID, Privileges and other parameters registered in the GlobalPlatform Registry, may be accessed by suitably authorized entities.

Runtime Behavior

On receipt of a GET STATUS command, the Security Domain performing the lifecycle interrogation shall:

- Apply its own secure communication policy. (Note: A minimum security level is defined in Table 11-2.)
- Check that the off-card entity at the origin of the request is authenticated as the Security Domain Provider (see section 10.4 – *Entity Authentication*).

On receipt of a request to interrogate the status of the card and the Issuer Security Domain, the OPEN shall:

- Check that the requesting on-card entity is the Issuer Security Domain or a Security Domain with the Global Registry Privilege.

9.6.7 Operational Velocity Checking

The OPEN shall be implemented with a velocity checking security mechanism. For the purpose of this document, ‘velocity checking’ is defined as the active monitoring, handling and management of security sensitive activities on the card.

These activities may include, but are not limited to the following:

- Content installation;
- Card exceptions;
- Application exceptions.

Typically, velocity checking is used to counter security attacks that use repeated attempts in their schemes. These attempts can be from internal (on-card) or external (off-card) entities. Velocity checking is implemented to track the number of consecutive failures in each of the related management and security events.

9.6.7.1 Content Loading and Installation

The OPEN may keep track of the number of unsuccessful consecutive attempts of the Card Content load and installation process by a particular Application and the total number of such attempts by all Applications. Actions may include such defensive measures as the locking or termination of the card.

9.6.7.2 Exceptions

Velocity checking may be implemented in cases in which the OPEN generates exceptions. However, it does not have to be implemented such that each individual exception is handled separately. A trace buffer and event log may be used to complement velocity checking.

For example, an implementation of a Security Domain may enable velocity checking to enforce strict APDU command sequencing for card and application management operations (e.g., Card Content loading and installation). The OPEN may also enable velocity checking against repeated failed attempts by an Application to allocate additional memory beyond its allowed limit that is stored in the GlobalPlatform Registry. The OPEN may choose to lock an Application that is exhibiting such behavior.

9.6.8 Tracing and Event Logging

Tracing and event logging functions may be enabled. These functions shall be implemented according to a Card Issuer's policy requirements.

9.7 Memory Resource Management

Memory resource management is an optional feature of a GlobalPlatform card. It consists of the management by OPEN of counters associated with Executable Load Files or Applications regarding their individual allocation of memory resources, as defined in this specification. It applies to both persistent and volatile memory. Memory resource management data elements describe memory usage requirements applicable to each type of memory: volatile or persistent; and for each type of storage: code or data. Memory requirements may be defined for each Executable Load File and Application at the time of their load and installation. The OPEN, in conjunction with the Runtime Environment, is responsible for managing the memory resources of the card; including the security requirements defined in section 4.2.4.1 – *Runtime Environment Security Requirements*.

A memory resource management data element indicates an amount of memory resources in bytes. In the INSTALL [for load] command, the System Parameters may include memory requirements specifying for each type of memory a Minimum Memory. In the INSTALL [for install] command, the System Parameters may include memory requirements specifying for each type of memory a Memory Quota, or Reserved Memory, or both.

When memory resource management is supported, the OPEN shall assign, if currently available, each type of memory resources as described in the corresponding memory resource management data element. Memory resources shall be assigned according to the following rules:

- Assigning Minimum Memory to an Executable Load File (and its subsequent Application instance) shall not reduce the memory resources currently available on the card;
- Assigning Memory Quota to an Application shall not reduce the memory resources currently available on the card;

- The value of Memory Quota assigned to an Application shall not be less than the value of its Reserved Memory;
- Assigning Reserved Memory to an Executable Load File / Application shall reduce the memory resources currently available on the card at the time of its successful load / installation.

When memory resource management is supported, the OPEN shall manage available memory resources for each type of memory according to the following rules:

- At the time of the load's request (INSTALL [for load] command), the Minimum Memory requirements shall be currently available on the card;
- At the time of the installation of an Application (INSTALL [for install] command), the amount of memory effectively allocated to that Application shall first be charged against the Reserved Memory of that Application until it is entirely exhausted. When the Reserved Memory (if any) is exhausted, the amount of allocated memory shall be charged against that Application's Memory Quota and shall reduce the memory resources currently available on the card. When either the Memory Quota is exceeded or the memory resources currently available on the card are exhausted, the installation of the Application shall fail;
- During the lifetime of an Application, the amount of memory effectively allocated to the creation of data shall first be charged against the Reserved Memory of that Application, until it is entirely exhausted. When the Reserved Memory (if any) is exhausted, the amount of allocated memory shall be charged against that Application's Memory Quota and shall reduce the memory resources currently available on the card. When either the Memory Quota is exceeded or the memory resources currently available on the card are exhausted, the resource allocation shall fail;
- The successful removal of an Application (DELETE command) shall augment the memory resources available on the card by the amount of memory effectively released and any unused part of the Reserved Memory (if any) of that Application;
- The results of reporting memory resources are implementation dependent;
- The amount of memory effectively released by the deletion of data shall be reallocated to the Reserved Memory and Memory Quota assigned to that Application. If no Reserved Memory was assigned to that Application, the amount of released memory shall augment the memory resources available on the card.

10 Secure Communication

The GlobalPlatform documentation broadly defines the notion of secure communication over and above that defined in ISO standards. Specific mention is made of secure messaging for APDU commands; an optional authentication process is implied in most of the flow diagrams and Application access to Security Domain services implies that the ability to create a Secure Channel Session exists. This chapter defines the general rules that apply to all Secure Channel Protocols. Applications that utilize the services of Security Domains can use the Secure Channel Protocol supported by their associated Security Domains.

These protocols provide a means by which a Security Domain or an Application may communicate with an off-card entity within a logically secure environment.

Logical channels facilitate the possibility of more than one of the above off-card entities communicating concurrently with multiple Applications on the card, each within its own logically secure environment. It is the responsibility of the Security Domain provider to define whether this is possible or not.

10.1 Secure Channel

The Secure Channel provides a secure communication channel between a card and an off-card entity during an Application Session.

A Secure Channel Session is divided into three sequential phases:

- **Secure Channel Initiation** – when the on-card Application and the off-card entity have exchanged sufficient information enabling them to perform the required cryptographic functions. The Secure Channel Session initiation always includes the authentication of the off-card entity by the on-card Application;
- **Secure Channel Operation** – when the on-card Application and the off-card entity exchange data within the cryptographic protection of the Secure Channel Session. The Secure Channel services offered may vary from one Secure Channel Protocol to the other;
- **Secure Channel Termination** – when either the on-card Application or the off-card entity determines that no further communication is required or allowed via an established Secure Channel Session.

The three levels of security provided by the Secure Channel are defined in section 4.3.2 – *Secure Communication*.

A further level of security applies to sensitive data (e.g. secret keys) that shall always be transmitted as confidential data.

10.2 Explicit / Implicit Secure Channel

A Secure Channel Session is open after a successful initiation, including the authentication of the off-card entity by the on-card Application. A Secure Channel Session is terminated after the last successful communication within the same Secure Channel Session.

10.2.1 Explicit Secure Channel Initiation

Initiating a Secure Channel Session may be achieved by the off-card entity using appropriate APDU command(s) or by the on-card Application using the appropriate API. This method is the explicit Secure Channel creation. Appropriate APDU commands allow the off-card entity to instruct the card what level of security is required for the current Secure Channel Session (integrity and/or confidentiality). They also give the off-card entity the possibility of selecting the keys to be used.

10.2.2 Implicit Secure Channel Initiation

The Secure Channel Session is initiated by the on-card Secure Channel Protocol handler, directly or through an API, when receiving the first APDU command that contains a cryptographic protection. The Secure Channel Protocol handler implicitly knows the required level of security. The Secure Channel Protocol handler may implicitly know which keys are to be used or may have been previously instructed by the off-card entity using appropriate APDU command(s) prior to initiating the Secure Channel Session.

10.2.3 Secure Channel Termination

Termination of the Secure Channel Session can be achieved by the on-card Application using the appropriate API or by the off-card entity using the appropriate APDU command.

Secure Channel termination causes all session data to be reset and all ICVs and session keys to be erased. The Current Security Level is set to NO_SECURITY and the Session Security Level is reset.

Termination of the Secure Channel Session occurs when one of the following conditions is met:

- The on-card Application Session is terminated - for instance when another Application is selected (i.e. the Application Session ends) on the same logical channel of the same card I/O interface. It may be the responsibility of the Application to initiate the termination of the Secure Channel Session when this occurs;
- The associated logical channel is explicitly closed;
- The card is reset (see [ISO 7816-3] for contact cards), deactivated (see [ISO 14443-3] for contactless cards) or powered off: i.e. the Card Session ends.

A Secure Channel Session is aborted but not terminated in the following cases:

- The on-card Application receives the first APDU command with an erroneous cryptographic protection;
- The on-card Application receives an APDU command without the required cryptographic protection set up during Secure Channel Session initiation;

If a Secure Channel Session is aborted, the Current Security Level is set to NO_SECURITY, the Session Security Level is not reset and the error condition persists until the Secure Channel Session is terminated.

A Secure Channel Session on a logical channel is never terminated in favor of a new Secure Channel Session being initiated on another logical channel.

10.3 Direct / Indirect Handling of a Secure Channel Protocol

There are two ways for the on-card Application to use a Secure Channel Protocol.

- **Direct** – The Application owns its Secure Channel key set and fully implements the protocol: an example is Security Domains.
- **Indirect** – The Application uses the Security Domain services to handle the Secure Channel Protocol. This allows the Application using these services to be coded independently from the Secure Channel Protocol supported by the card.

10.4 Entity Authentication

Off-card entity authentication is achieved through the process of initiating a Secure Channel Session and provides assurance to the card that it is communicating with an authenticated entity. If any step in the authentication process fails, the process shall be restarted.

Authentication of the off-card entity is only provided for a limited time, a Secure Channel Session, and is valid only for the messages within that Secure Channel. This Secure Channel Session relates to the establishment and termination of a Secure Channel. If the Secure Channel Session is closed for any reason the off-card entity shall no longer be considered authenticated.

10.4.1 Authentication with Symmetric Cryptography

With a symmetric key Secure Channel Protocol (e.g. SCP01, SCP02, or SCP03), an authenticated off-card entity is the entity which knows the secret Secure Channel keys needed to initiate a Secure Channel Session. The card cannot distinguish between the actual Application Provider of the Security Domain and one of its agents to whom a (set of) secret Secure Channel key(s) was provided. In this case, the card cannot distinguish between the Security Domain provider and the provider of one of its associated Applications. Both are assumed to be the same entity: the Application Provider.

When the Secure Channel initiation process successfully authenticates the Application Provider, the Current Security Level shall be updated with the AUTHENTICATED indicator (see section 10.6 – *Security Levels*). The Application is informed whether that unique entity, the Application Provider, has been successfully authenticated or not by examining the presence of the AUTHENTICATED in the Current Security Level.

10.4.2 Authentication with Asymmetric Cryptography

With an asymmetric key Secure Channel Protocol (e.g. SCP10), any off-card entity that owns a pair of asymmetric keys and obtained a certificate for its public key certified by an authority recognized by the Security Domain can be successfully authenticated by that Security Domain. An authenticated off-card entity is the subject identified in the off-card entity's public key certificate verified during the Secure Channel Initiation. The card can distinguish between the actual Application Provider of the Security Domain and any other off-card entity using the off-card entity public key certificate's subject identifier.

In this case, the card can distinguish between the Security Domain's provider, the provider of one of its associated Applications, and any other off-card entity. The three are not necessarily the same entity: the Application Provider Id of each on-card entity is registered in the GlobalPlatform Registry at the time of the load or installation of that on-card entity and cannot be changed subsequently.

When the Secure Channel initiation process successfully authenticates the Application Provider, the Current Security Level shall be updated with the AUTHENTICATED indicator (see section 10.6 – *Security Levels*). When the Secure Channel initiation process successfully authenticates another off-card entity, including the Security Domain's provider, or if the Application Provider Id is not registered, the Current Security Level shall be updated with the ANY_AUTHENTICATED indicator. The Application is informed of these states by examining the presence of the AUTHENTICATED and ANY_AUTHENTICATED indicators in the Current Security Level.

The Current Security Level viewed by the Target Application may differ from the Security Domain's view depending on the Secure Channel Protocol and the authenticated off-card entity's Application Provider Id (e.g. ANY_AUTHENTICATED for the Target Application and AUTHENTICATED for the Security Domain).

10.5 Secure Messaging

Secure messaging allows a sending entity to add confidentiality and/or integrity and authentication to the composition of an APDU message prior to the message being transmitted to a receiving entity:

- Integrity of an APDU command sent to the card or confidentiality of the command message and integrity of an APDU command sent to the card;
- Integrity of the sequence of APDU commands sent to the card within a Secure Channel Session;
- Optionally, depending on the Secure Channel Protocol, confidentiality and/or integrity of an APDU response message sent by the card, and integrity of the sequence of responses within a Secure Channel Session.

In the context of secure messaging, message integrity also provides data origin authentication.

10.6 Security Levels

A Secure Channel Protocol operates according to the Security Level that is established. This is done at two levels:

- A mandatory minimum Session Security Level is set at the initialization of the Secure Channel Session either explicitly or implicitly;
- A different Current Security Level may be set for an individual command or an individual response.

These Security Levels establish the minimum acceptable secure messaging protection that must be applied to protected messages, either for the whole session or for a specific command-response pair.

The following table shows the coding of Current Security Level:

Table 10-1: Current Security Level

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1	0	-	-	-	-	-	-	AUTHENTICATED
0	1	-	-	-	-	-	-	ANY_AUTHENTICATED
-	-	-	-	-	-	1	-	C_DECRYPTION
-	-	-	-	-	-	-	1	C_MAC
-	-	1	-	-	-	-	-	R_ENCRYPTION
-	-	-	1	-	-	-	-	R_MAC
-	-	-	-	X	X	-	-	RFU
0	0	0	0	0	0	0	0	NO_SECURITY_LEVEL

Note that the Current Security Level cannot have its AUTHENTICATED and ANY_AUTHENTICATED indicators set simultaneously.

The rules for handling Security Levels are defined individually for each Secure Channel Protocol.

10.7 Secure Channel Protocol Identifier

The Secure Channel Protocol identifies which particular secure communication protocol and set of security services are implemented in a Security Domain. The following values are assigned to the Secure Channel Protocol identifier:

- '**00**' – Not available
- '**01**' to '**7F**' – Reserved for use by GlobalPlatform
  - '**01**' – Deprecated Secure Channel Protocol 1 defined in Appendix D of this document
  - '**02**' – Secure Channel Protocol 2 defined in Appendix E of this document
  - '**03**' – Secure Channel Protocol 3 defined in [Amd D]
  - '**10**' – Secure Channel Protocol '10' defined in Appendix F of this document
  - '**11**' – Secure Channel Protocol '11' defined in [Amd F]
- '**80' to 'EF**' – Reserved for use by individual schemes registered by GlobalPlatform
  - '**80**' – Secure Channel Protocol '80' defined in [TS 102 225] and [TS 102 226]
  - '**81**' – Secure Channel Protocol '81' defined in [Amd B] based on HTTP and Pre-Shared Key TLS protocols
- '**F0**' to '**FF**' – Reserved for proprietary use and not registered by GlobalPlatform

The deprecated symmetric key Secure Channel Protocol '01' is backward compatible with the Open Platform Card Specification version 2.0.1'; it is deprecated in this version of the Specification.

The symmetric key Secure Channel Protocol '02' includes services in addition to those provided by Secure Channel Protocol '01' as well as optimizing the operation of some services compared to the deprecated Secure Channel Protocol '01'.

The symmetric key Secure Channel Protocol '03' includes services similar to Secure Channel Protocol '02', however, it uses AES rather than DES cryptography.

The Secure Channel Protocol '80' supports the Over The Air security scheme defined in [TS 102 225].

The Secure Channel Protocol '81' supports an Over The Air security scheme based on the usage of both HTTP and Pre-Shared Key TLS protocols.

The asymmetric key Secure Channel Protocol '10' offers authentication services using an RSA-based Public Key Infrastructure (PKI) and secure messaging protection of commands and responses using symmetric cryptography.

The asymmetric key Secure Channel Protocol '11' offers authentication services using an ECC-based Public Key Infrastructure (PKI) and secure messaging protection of commands and responses based on SCP '03'.

Part IV
APDU Command Reference

11 APDU Command Reference

This chapter details the GlobalPlatform APDU commands that may be implemented. The commands are listed alphabetically.

The following table summarizes the APDU commands that shall be supported by a Security Domain with Authorized Management privilege, a Security Domain with Delegated Management privilege and the requirements for support of these APDU commands by other Security Domains. When logical channels are supported, the MANAGE CHANNEL command is only processed by the OPEN and no Security Domain support is required for this command.

Table 11-1: Authorized GlobalPlatform Commands per Card Life Cycle State

Command	OP_READY			INITIALIZED			SECURED			CARD_ LOCKED	TERMINATED
	AM SD	DM SD	SD	AM SD	DM SD	SD	AM SD	DM SD	SD	SD	SD
DELETE Executable Load File											
DELETE Executable Load File and related Application(s)											
DELETE Application	✓			✓			✓				
DELETE Key											
GET DATA	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
GET STATUS	✓			✓			✓			✓	
INSTALL [for load]											
INSTALL [for install]											
INSTALL [for load, install and make selectable]											
INSTALL [for install and make selectable]	✓	✓		✓	✓		✓	✓			
INSTALL [for make selectable]											
INSTALL [for extradition]											
INSTALL [for registry update]											
INSTALL [for personalization]											
LOAD											
PUT KEY	✓			✓			✓				
SELECT	✓	✓	✓	✓	✓	✓	✓	✓	✓	See Note 1	
SET STATUS	✓			✓			✓			✓	
STORE DATA	✓			✓			✓				

Legend of Table 11-1:

AM SD	Security Domain with Authorized Management privilege
DM SD	Security Domain with Delegated Management privilege
SD	Other Security Domain
✓	Support required
Blank cell	Support optional
Striped cell	Support prohibited

Note 1: If an SD does have the Final Application privilege, it may be selected and process the SELECT command in the CARD_LOCKED life cycle state. Otherwise, it may not be selected; however, it may be able to process commands received and internally forwarded to it through a trusted framework (see Appendix G).

The following table summarizes the minimum security requirements for the APDU commands.

Table 11-2: Minimum Security Level for GlobalPlatform Commands

Command	Minimum Security Level
DELETE	AUTHENTICATED
GET DATA	None
GET STATUS	AUTHENTICATED
INSTALL	AUTHENTICATED
LOAD	AUTHENTICATED
MANAGE CHANNEL	Not applicable
PUT KEY	AUTHENTICATED
SELECT	Not applicable
SET STATUS	AUTHENTICATED
STORE DATA	AUTHENTICATED

Note: Higher minimum security levels may be defined for these commands in some contexts (e.g. usage of a SCP80 secure channel).

11.1 General Coding Rules

The use of 'RFU' and '-' (dash) in tables is as defined in Table 1-3. If a bit is shown as RFU the bit should be set to zero by the off-card entity and the card shall ignore the value.

11.1.1 Life Cycle State Coding

The Executable Load File Life Cycle is coded on one byte as described in the following table:

Table 11-3: Executable Load File Life Cycle Coding

b8	b7	b6	b5	b4	b3	B2	b1	Meaning
0	0	0	0	0	0	0	1	LOADED

The Application Life Cycle has a bit-oriented coded value on one byte as described in the following table.

Note: The application has free use of bits b4-b7, and the coding of these bits is beyond the scope of this specification.

Table 11-4: Application Life Cycle Coding

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0	0	0	0	0	1	1	INSTALLED
0	0	0	0	0	1	1	1	SELECTABLE
0	X	X	X	X	1	1	1	Application Specific State
1	-	-	-	-	-	1	1	LOCKED

The Security Domain Life Cycle has a bit-oriented coded value on one byte as described in the following table:

Table 11-5: Security Domain Life Cycle Coding

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0	0	0	0	0	1	1	INSTALLED
0	0	0	0	0	1	1	1	SELECTABLE
0	0	0	0	1	1	1	1	PERSONALIZED
1	0	0	0	-	-	1	1	LOCKED

The allowed Application and Security Domain Life Cycle transitions are described in Chapter 5 – *Life Cycle Models*.

The Issuer Security Domain inherits the card Life Cycle State that has a bit-oriented coded value on one byte as described in the following table:

Table 11-6: Card Life Cycle Coding

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0	0	0	0	0	0	1	OP_READY
0	0	0	0	0	1	1	1	INITIALIZED
0	0	0	0	1	1	1	1	SECURED
0	1	1	1	1	1	1	1	CARD_LOCKED
1	1	1	1	1	1	1	1	TERMINATED

The allowed card Life Cycle transitions are described in Chapter 5 – *Life Cycle Models*.

11.1.2 Privileges Coding

Privileges are coded on three bytes as described in the following tables (see section 6.6 – *Privileges* for additional details):

Table 11-7: Privileges (Byte 1)

b8	b7	b6	b5	b4	b3	b2	b1	Meaning	Privilege Number
1	-	-	-	-	-	-	-	Security Domain	0
1	1	-	-	-	-	-	0	DAP Verification	1
1	-	1	-	-	-	-	-	Delegated Management	2
-	-	-	1	-	-	-	-	Card Lock	3
-	-	-	-	1	-	-	-	Card Terminate	4
-	-	-	-	-	1	-	-	Card Reset	5
-	-	-	-	-	-	1	-	CVM Management	6
1	1	-	-	-	-	-	1	Mandated DAP Verification	7

Table 11-8: Privileges (Byte 2)

b8	b7	b6	b5	b4	b3	b2	b1	Meaning	Privilege Number
1	-	-	-	-	-	-	-	Trusted Path	8
-	1	-	-	-	-	-	-	Authorized Management	9
-	-	1	-	-	-	-	-	Token Management	10
-	-	-	1	-	-	-	-	Global Delete	11
-	-	-	-	1	-	-	-	Global Lock	12
-	-	-	-	-	1	-	-	Global Registry	13
-	-	-	-	-	-	1	-	Final Application	14
-	-	-	-	-	-	-	1	Global Service	15

Table 11-9: Privileges (Byte 3)

b8	b7	b6	b5	b4	b3	b2	b1	Meaning	Privilege Number
1	-	-	-	-	-	-	-	Receipt Generation	16
-	1	-	-	-	-	-	-	Ciphered Load File Data Block	17
-	-	1	-	-	-	-	-	Contactless Activation	18
-	-	-	1	-	-	-	-	Contactless Self-Activation	19
-	-	-	-	X	X	X	X	RFU	-

11.1.3 General Error Conditions

The following table describes the error conditions that may be returned by any command:

Table 11-10: General Error Conditions

SW1	SW2	Meaning
'64'	'00'	No specific diagnosis
'67'	'00'	Wrong length in Lc
'68'	'81'	Logical channel not supported or is not active
'69'	'82'	Security status not satisfied
'69'	'85'	Conditions of use not satisfied
'6A'	'86'	Incorrect P1 P2
'6D'	'00'	Invalid instruction
'6E'	'00'	Invalid class

11.1.4 Class Byte Coding

The class byte of all commands shall conform to [ISO 7816-4] and shall be coded according to section 11.1.4.1 for commands addressed to the Basic Logical Channel and Supplementary Logical Channels 1, 2, and 3. For cards implementing four or more Supplementary Logical Channels the class byte of all commands addressed to Supplementary Logical Channel four to nineteen shall be coded according to section 11.1.4.2.

11.1.4.1 First Interindustry Class Byte Coding

The following table describes the first interindustry class byte coding:

Table 11-11: CLA Byte Coding

b8	b7	b6	b5(*)	b4	b3	b2	b1	Meaning
0	0	0	0	-	-	-	-	Command defined in ISO/IEC 7816
1	0	0	0	-	-	-	-	GlobalPlatform command
-	0	0	0	0	0	-	-	No secure messaging
-	0	0	0	0	1	-	-	Secure messaging – GlobalPlatform proprietary
-	0	0	0	1	0	-	-	Secure messaging – ISO/IEC 7816 standard, command header not processed (no C-MAC)
-	0	0	0	1	1	-	-	Secure messaging – ISO/IEC 7816 standard, command header authenticated (C-MAC)
-	0	0	0	-	-	X	X	Logical channel number

Class byte bits b1 and b2 may be set to define the required logical channel number according to ISO/IEC 7816:

- A class byte with bits b1 and b2 set to 00 indicates receipt of the command on the Basic Logical Channel;

- A class byte with bits b1 and b2 set to 01 (1), 10 (2), or 11 (3) indicates receipt of the command on a Supplementary Logical Channel.

Class byte bits b4 and b3 may be set to define the required secure messaging according to [ISO 7816-4]:

- A class byte with bits b4 and b3 set to 00 indicates no secure messaging;
- A class byte with bits b4 and b3 set to 01 indicates secure messaging with GlobalPlatform format;
- A class byte with bits b4 and b3 set to 11 or 10 indicates secure messaging with [ISO 7816-4] format.

(*) Note: class byte bit b5 is always 0 except if it is coded for command chaining as defined in Appendix F.

11.1.4.2 Further Interindustry Class Byte Coding

The following table describes the further interindustry class byte coding.

Table 11-12: CLA Byte Coding

b8	b7	b6	b5(*)	b4	b3	b2	b1	Meaning
0	1	-	0	-	-	-	-	Command defined in ISO/IEC 7816
1	1	-	0	-	-	-	-	GlobalPlatform command
-	1	0	0	-	-	-	-	No secure messaging
-	1	1	0	-	-	-	-	Secure messaging – ISO/IEC 7816 or GlobalPlatform proprietary
-	1	-	0	X	X	X	X	Logical channel number

Class byte bits b1 to b4 may be set to define the required logical channel number according to ISO/IEC 7816;

- A class byte with bit b7 set to 1 indicates in bits b4 to b1 (values 0000 to 1111) receipt of the command on Supplementary Logical Channel 4 to 19.

Class byte bit b6 may be set to define the required secure messaging according to [ISO 7816-4] or proprietary GlobalPlatform format:

- A class byte with bit b6 set to 0 indicates no secure messaging;
- A class byte with bit b6 set to 1 indicates secure messaging with GlobalPlatform format or with [ISO 7816-4] format.

11.1.5 APDU Message and Data Length

All GlobalPlatform APDU messages and data elements are counted in bytes. All GlobalPlatform APDU commands use ISO/IEC 7816 short message lengths; i.e. Lc and Le bytes coded on one byte.

All GlobalPlatform APDU command messages (excluding the APDU header) are limited to 255 bytes in length. All GlobalPlatform APDU commands that expect response data have their Le byte set to zero, indicating that all available response data shall be returned. According to [ISO 7816-4], all GlobalPlatform responses returned in APDU response messages shall have a maximum length of 256 bytes of response data.

All length fields of GlobalPlatform messages and data objects defined in this specification, except for Lc and Le, are coded as defined for ASN.1 BER-TLV (see [ISO 8825-1]): 1 byte for a length up to 127, 2 bytes for a length up to 255, and 3 bytes for a length up to 65535, except where otherwise stated.

This version of the Specification is written as though most command functions can be handled with a single APDU command and response pair. This is done in order to simplify the description, and is not intended to preclude the use of multiple commands to transfer a long message where such a mechanism is indicated: such as (for GlobalPlatform chained commands) the 'more commands' bit in the P1 byte and status bytes controlling the return of additional data; and (for ISO commands) command and response chaining.

11.1.5.1 GlobalPlatform Command Chaining

The following rules shall apply to commands which support the 'more commands' bit chaining mechanism (DELETE, INSTALL and PUT KEY commands):

- All the commands of the sequence shall directly follow each other and have the same command header, except for Lc and the chaining information in P1. If this is not the case, a response of '6985' shall be returned and the chaining session shall be aborted.
- P1.b8 indicates if the command is expected to be followed by the next command of the chain or if the command is the last of a sequence of chained commands.
- The overall length supported with chaining is implementation dependent. If chaining is not supported or if the chaining limit has been reached, then a response of '6A86' shall be returned.
- The data fields of the sequence of commands shall be concatenated on the card to form a full set of data that may be consistently analyzed and processed. At any moment during the reception of the sequence of commands, an appropriate error code may be returned if an error is detected in the data field of a command, in which case the chaining session shall be aborted. However, an implementation is not required to check or process data before the entire sequence of commands has been received.
- For all commands of the sequence except for the last one, a response of '9000' with no additional response data shall be returned if no error has been detected (in particular, if the Current Security Level indicates R_MAC, no R-MAC shall be returned; see section 11.1.5.2). The regular response shall only be returned upon reception of the last command of the sequence.
- Security shall be applied to the command data before segmenting it into several commands. If a command includes a checksum (e.g. C-MAC) and the command data without the checksum exceeds 255 bytes, the calculation of the checksum shall be performed across the non-segmented command which is defined as follows:
  - Reference control parameters P1 and P2 shall be set as for the last command of the sequence of chained commands.
  - 'Length of the following data fields' shall indicate the length of the data following. It shall be coded on 3 bytes with values from '00 01 00' to '00 FF FF'.
  - This shall be followed by the concatenation of the data fields of all commands of the chained sequence.

  The previous statement has an impact on how Secure Channel Protocols (e.g. SCP02 or SCP03) are applied to such a sequence of chained commands. In this case, the C-MAC shall be computed over the non-segmented command and shall be appended only at the end of the last command of the sequence. In the same manner, command data field encryption/decryption (if requested) shall be performed over the non-segmented command data field.

11.1.5.2 Response Chaining

For any command where the response exceeds 256 bytes, the chaining mechanism defined in [ISO 7816-4], using the '61xx' status word and multiple GET RESPONSE commands, should be used. This has an impact on how Secure Channel Protocols (e.g. SCP02 or SCP03) shall be applied to response data. In this case, the R-MAC (if requested) shall be computed over the non-segmented response data and shall be appended only to the response of the last GET RESPONSE command of the sequence. In the same manner, response data field encryption (if requested) shall be performed over the non-segmented response data field.

11.1.6 Confirmations in Response Messages

The confirmation shall be structured according to the following table:

Table 11-13: Confirmation Structure

Length	Data Element	Presence
1-2	Length of Receipt ('00' - '7F' or '81 80' - '81 FF')	Mandatory
0-n	Receipt	Conditional
5-n	Confirmation Data	Mandatory

The length field for the Receipt is coded according to ASN.1 BER-TLV (see [ISO 8825-1]).

Confirmation Data is structured according to the following table:

Table 11-14: Confirmation Data

Length	Data Element	Presence
1	Length of Confirmation Counter	Mandatory
2	Confirmation Counter	Mandatory
1	Length of Card Unique Data	Mandatory
1-n	SD Unique Data	Mandatory
1	Length of Token identifier	Conditional
0-n	Token identifier	Conditional
1	Length of Token Data digest	Optional
0-n	Token Data digest	Optional

The presence of Token identifier depends on whether a Token identifier was included in the original Token. The presence of the Token Data digest is dependent on the policy of the Security Domain with Receipt Generation privilege.

Security Domain unique data is the concatenation without delimiters of the Security Domain Provider Identification Number ('42') and the Security Domain Image Number (SDIN, tag '45') of the Security Domain generating the confirmation (with the Receipt Generation Privilege).

11.1.7 Implicit Selection Parameter Coding

The Implicit Selection parameter (tag 'CF') indicates that this Application is to be implicitly selected on one (or more) specific card interface(s) for a specific logical channel, see section 6.6.3 – *Privilege Management*. Setting both bits 8 and 7 to zero indicates that this Application is to be selectable only explicitly with a SELECT [by name] command and the logical channel number shall be ignored by the card. The value field is coded on one byte as follows:

Table 11-15: Implicit Selection Parameter

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1	-	-	-	-	-	-	-	Contactless I/O
-	1	-	-	-	-	-	-	Contact I/O
-	-	X	-	-	-	-	-	RFU
-	-	-	X	X	X	X	X	Logical channel number (0 to 19)

11.1.8 Key Type Coding

Key type may be coded on one or two bytes. When coded on two bytes, the first byte shall be set to 'FF' to indicate an extended format of the key data field. The second or only byte shall be coded according to the following table:

Table 11-16: Key Type Coding

Value	Meaning
'00' - '7F'	Reserved for private use
'80'	DES – mode (ECB/CBC) implicitly known
'81'	Reserved for historical reasons
'82'	Reserved for historical reasons
'83'	Reserved for historical reasons
'84'	Reserved for historical reasons
'85'	Pre-Shared Key for Transport Layer Security
'86' - '87'	RFU (symmetric algorithms)
'88'	AES (16, 24, or 32 long keys)
'89' - '8F'	RFU (symmetric algorithms)
'90'	HMAC-SHA1 – length of HMAC is implicitly known
'91'	HMAC-SHA1-160 – length of HMAC is 160 bits
'92' - '9F'	RFU (symmetric algorithms)
'A0'	RSA Public Key - public exponent e component (clear text)
'A1'	RSA Public Key - modulus N component (clear text)
'A2'	RSA Private Key - modulus N component
'A3'	RSA Private Key - private exponent d component

Value	Meaning
'A4'	RSA Private Key - Chinese Remainder P component
'A5'	RSA Private Key - Chinese Remainder Q component
'A6'	RSA Private Key - Chinese Remainder PQ component ($q^{-1} \bmod p$)
'A7'	RSA Private Key - Chinese Remainder DP1 component ($d \bmod (p-1)$)
'A8'	RSA Private Key - Chinese Remainder DQ1 component ($d \bmod (q-1)$)
'A9' - 'AF'	RFU (asymmetric algorithms)
'B0'	ECC public key
'B1'	ECC private key
'B2'	ECC field parameter P (field specification)
'B3'	ECC field parameter A (first coefficient)
'B4'	ECC field parameter B (second coefficient)
'B5'	ECC field parameter G (generator)
'B6'	ECC field parameter N (order of generator)
'B7'	ECC field parameter k (cofactor of order of generator)
'B8' - 'EF'	RFU (asymmetric algorithms)
'F0'	ECC key parameters reference
'F1' - 'FE'	RFU (asymmetric algorithms)
'FF'	Extended format (usage defined for specific APDU commands; e.g. PUT KEY)

11.1.9 Key Usage Qualifier Coding

The values for the Key Usage Qualifier parameter may be coded on 1 or 2 bytes according to the following tables:

Table 11-17: Key Usage Qualifier (1st Byte)

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1	-	-	-	-	-	-	-	Verification (DST, CCT, CAT), Encipherment (CT)
-	1	-	-	-	-	-	-	Computation (DST, CCT, CAT), Decipherment (CT)
-	-	1	-	-	-	-	-	Secure messaging in response data fields (CT, CCT)
-	-	-	1	-	-	-	-	Secure messaging in command data fields (CT, CCT)
-	-	-	-	1	-	-	-	Confidentiality (CT)
-	-	-	-	-	1	-	-	Cryptographic Checksum (CCT)
-	-	-	-		-	1	-	Digital Signature (DST)
-	-	-	-	-	-	-	1	Cryptographic Authorization (CAT)

Table 11-18: Key Usage Qualifier (2nd Byte)

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1	-	-	-	-	-	-	-	Key Agreement (KAT)
-	-	-	-	-	-	-	-	(RFU)

If only the first byte is provided or available, then the second byte shall be assumed to have a value of '00'.

The following values are used by GlobalPlatform. Values are independent of the cryptographic algorithm used.

- C-MAC = '14', R-MAC = '24', C-MAC + R-MAC = '34'
- C-ENC = '18', R-ENC = '28', C-ENC + R-ENC = '38'
- C-DEK = '48', R-DEK = '88', C-DEK + R-DEK = 'C8'
- PK_SD_AUT = '82'
- SK_SD_AUT = '42'
- Token = '81'
- Receipt = '44'
- DAP = '84'
- PK_SD_AUT + Token = '83'
- SK_SD_AUT + Receipt = '43'
- PK_SD_AUT + DAP = '86'
- PK_SD_AUT + Token + DAP = '87'

11.1.10 Key Access Coding

The values for the Key Access parameter are set according to the following table (see section 7.5.2 – *Key Access Conditions* for details):

Table 11-19: Key Access

Value	Description
'00'	The key may be used by the Security Domain and any associated Application
'01'	The key may only be used by the Security Domain
'02'	The key may be used by any Application associated with the Security Domain but not by the Security Domain itself
'03' - '1F'	RFU
'20' - 'FE'	Proprietary usage
'FF'	Not available

11.1.11 Tag Coding

All tags of GlobalPlatform data objects are coded as defined by ASN.1 BER-TLV rules (see [ISO 8825-1]).

For Security Domain data objects present in GlobalPlatform APDU messages, BER-TLV tags are coded according to the following rules:

- '00' to '7E' – Reserved for use by ISO/IEC;
- '80' to '9E' and 'A0' to 'BE' – Reserved for use depending on the context (see note);
- 'C0' to 'DD' and 'E0' to 'FD' – Reserved for use by GlobalPlatform or individual schemes registered by GlobalPlatform;
  - 'CA' and 'EA' – Reserved for use by [TS 102 226];
- 'DE' and 'FE' – Reserved for proprietary use and not registered by GlobalPlatform;
- '1F 1F' to '7F 7F' – Reserved for use by ISO/IEC;
- '9F 1F' to '9F 7F' and 'BF 1F' to 'BF 7F' – Reserved for use depending on the context (see note);
- 'DF 1F' to 'DF 7F' and 'FF 1F' to 'FF 7F' – Reserved for use by GlobalPlatform or individual schemes registered by GlobalPlatform;
  - 'FF 1F' to 'FF 3F' – Reserved for use by [TS 102 226].

Note: Context dependent tags are assigned by the authority defining the context; e.g. ISO/IEC for data objects embedded in other ISO/IEC data objects or GlobalPlatform for data objects embedded in other GlobalPlatform data objects.

Tag allocation for any data objects embedded in the constructed tag 'FE' (reserved for proprietary use) is beyond the scope of this specification. Applying BER-TLV rules in this case is recommended.

11.1.12 Data Grouping Identifier (DGI) Coding

This specification (and amendments) defines specific DGIs for Security Domains. The use of DGIs by other Applications is out of scope. Any DGI shall be coded on two bytes in binary format, followed by a length indicator coded as defined in [Scripting Lang] Annex B.

11.2 DELETE Command

11.2.1 Definition and Scope

The DELETE command is used to delete a uniquely identifiable object such as an Executable Load File, an Application, an Executable Load File and its related Applications or a key. In order to delete an object, the object shall be uniquely identifiable by the selected Application.

11.2.2 Command Message

The DELETE command message is coded according to the following table:

Table 11-20: DELETE Command Message

Code	Value	Meaning
CLA	'80' - '8F', 'C0' - 'CF', or 'E0' - 'EF'	See section 11.1.4
INS	'E4'	DELETE
P1	'xx'	Reference control parameter P1
P2	'xx'	Reference control parameter P2
Lc	'xx'	Length of data field
Data	'xxxx…'	TLV coded objects (and C-MAC if present)
Le	'00'	

11.2.2.1 Reference Control Parameter P1

Reference control parameter P1 allows for command data to be longer than 255 bytes and to be segmented into components of arbitrary size and transmitted in a series of DELETE commands. P1 indicates whether the command data is one of a sequence of components or the last (or only) component.

Table 11-21: DELETE Reference Control Parameter P1

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	-	-	-	-	-	-	-	Last (or only) command
1	-	-	-	-	-	-	-	More DELETE commands
-	X	X	X	X	X	X	X	RFU

11.2.2.2 Reference Control Parameter P2

The reference control parameter P2 indicates whether the object in the data field is being deleted or whether the object in the data field and its related objects are being deleted. It shall be coded according to the following table:

Table 11-22: DELETE Reference Control Parameter P2

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	-	-	-	-	-	-	-	Delete object
1	-	-	-	-	-	-	-	Delete object and related object
-	X	X	X	X	X	X	X	RFU

11.2.2.3 Data Field Sent in the Command Message

The data field of the DELETE command message shall contain the TLV coded name(s) of the object to be deleted.

11.2.2.3.1 Delete [card content] Data Field

The Application or Executable Load File AID may be followed by a Control Reference Template for Digital Signature and a Delete Token, as follows:

Table 11-23: Delete [card content] Command Data Field

Tag	Length	Name	Presence
'4F'	5-16	Executable Load File or Application AID	Mandatory
'B6'	Variable	Control Reference Template for Digital Signature	Conditional
'42'	1-n	Identification Number of the Security Domain with the Token Verification privilege	Optional
'45'	1-n	Image number of the Security Domain with the Token Verification privilege	Optional
'5F20'	1-n	Application Provider identifier	Optional
'93'	1-n	Token identifier/number (digital signature counter)	Conditional
'9E'	1-n	Delete Token	Conditional

The identity of the Application or Executable Load File to delete shall be specified using the tag for an AID ('4F') followed by a length and the AID of the Application or Executable Load File. When simultaneously deleting an Executable Load File and all its related Applications, only the identity of the Executable Load File shall be provided.

The presence of the Control Reference Template for Digital Signature and Delete Token depends on the Card Issuer's policy. The presence of the data objects in tag 'B6' and the sub-tags '42', '45', '5F20', and '93' are strongly recommended when using SCP10.

The length fields for the Control Reference Template for Digital Signature and the Delete Token are coded according to ASN.1 BER-TLV (see [ISO 8825-1]).

11.2.2.3.2 Delete (Key) Data Field

To delete one or more keys, the Key Version Number and/or the Key Identifier are included in the command data field as follows:

Table 11-24: DELETE [key] Command Data Field

Tag	Length	Meaning	Presence
'D0'	1	Key Identifier	Conditional
'D2'	1	Key Version Number	Conditional

A single key is deleted when both the Key Identifier ('D0') and the Key Version Number ('D2') are provided in the DELETE command message data field. Multiple keys may be deleted if one of these values is omitted (i.e. all keys with the specified Key Identifier or Key Version Number). The options available for omitting these values are conditional on the Issuer's policy.

11.2.3 Response Message

A data field shall always be returned in the response message. The content of the data field is only relevant in the case of Delegated Management; i.e. if a Security Domain with the Delegated Management privilege is deleting an Application or Executable Load File, a Receipt may be present in the data field depending on the security policy of the Card Issuer.

11.2.3.1 Data Field Returned in the Response Message

For a DELETE [key] command, a single byte of '00' shall be returned indicating that no additional data is present.

For a DELETE [card content] command being issued to a Security Domain with the Delegated Management privilege, the data field may contain the confirmation of the delete procedure. The overall length of the response message shall not exceed 256 bytes.

The following table describes the structure of the DELETE response data field. The length field for the Delete Confirmation is coded according to ASN.1 BER-TLV (see [ISO 8825-1]).

Table 11-25: DELETE Response Data Field

Name	Length	Value	Presence
Length of delete confirmation	1-2	'00' - '7F' or '81 80' - '81 FF'	Mandatory
Delete confirmation	0-n	'xxxx...' – see section 11.1.6	Conditional

11.2.3.2 Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

This command may return either a general error condition as listed in section 11.1.3 – *General Error Conditions* or one of the following error conditions.

Table 11-26: DELETE Error Conditions

SW1	SW2	Meaning
'65'	'81'	Memory failure
'6A'	'88'	Referenced data not found
'6A'	'82'	Application not found
'6A'	'80'	Incorrect values in command data

11.3 GET DATA Command

11.3.1 Definition and Scope

The GET DATA command is used to retrieve either a single data object, which may be constructed, or a set of data objects. Reference control parameters P1 and P2 coding is used to define the specific data object tag. The data object may contain information pertaining to a key.

11.3.2 Command Message

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

Table 11-27: GET DATA Command Message

Code	Value	Meaning
CLA	'00' - '0F', '40' - '4F', '60' - '6F', '80' - '8F', 'C0' - 'CF', or 'E0' - 'EF'	See section 11.1.4
INS	'CA' or 'CB'	GET DATA If CLA = '00' - '0F', '40' - '4F', '60' - '6F', even or odd instruction code 'CA' or 'CB' If CLA = '80' - '8F', 'C0' - 'CF' or 'E0' - 'EF', even instruction code 'CA'
P1	'xx'	'00' or high order tag value
P2	'xx'	Low order tag value
Lc	'xx'	Not present if no command data, or length of data field.
Data	'xxxx…'	Tag list and/or C-MAC if present.
Le	'00'	

According to [ISO 7816-4], the odd instruction code 'CB' in conjunction with a class byte indicating an ISO command (CLA set to '00' - '0F', '40' - '4F' or '60' - '6F') is used to retrieve the contents of a file. It may be used to retrieve the list of Applications on the card (P1-P2 set to '2F 00').

11.3.2.1 Parameter P1 and P2

Parameters P1 and P2 define the tag of the data object to be read.

The value '2F 00' indicates a request to obtain details of Applications on the card, as defined in [ISO 7816-4]. The instruction code shall be set to 'CA' if the class byte indicates a GlobalPlatform command (CLA set to '80' - '8F', 'C0' - 'CF' or 'E0' - 'EF') or 'CB' if the class byte indicates an ISO command (CLA set to '00' - '0F', '40' - '4F' or '60' - '6F').

Security Domains shall support at least the following data object tags:

- Tag '42': Issuer Identification Number (or Security Domain Provider Identification Number)
- Tag '45': Card Image Number (or Security Domain Image Number)
- Tag '66': Card Data (or Security Domain Management Data)
- Tag 'E0': Key Information Template

The Issuer Security Domain may support the following data object tags:

- Tag '67': Card Capability Information

Security Domains may support the following data object tags:

- Tag 'D3': Current Security Level
- Tag '2F00': List of Applications associated with the Security Domain, or every application on the card if the Security Domain has Global Registry Privilege
- Tag 'FF21': Extended Card Resources Information available for Card Content Management, as defined in [TS 102 226]
- Tag '5F50': Security Domain Manager URL

A Security Domain with Receipt Generation privilege shall support the following additional data object tag:

- Tag 'C2': Confirmation Counter

A Security Domain supporting Secure Channel Protocol '02', or Secure Channel Protocol '03' in predictable card challenge mode (see [Amd D]), shall support the following data object tag:

- Tag 'C1': Sequence Counter of the default Key Version Number

For a Security Domain supporting only Secure Channel Protocol '01', an attempt to retrieve the Sequence Counter of the default Key Version Number (tag 'C1') shall be rejected.

Other tags as defined in section 11.1.11 – *Tag Coding* may be available for data objects supported by a Security Domain.

11.3.2.2 Data Field Sent in the Command Message

The data field of the GET DATA command message shall be empty unless a tag list and/or a MAC is required. For retrieving a list of applications present on the card (P1-P2 set to '2F 00') a tag list shall be present and coded as '5C 00'.

11.3.3 Response Message

11.3.3.1 Data Field Returned in the Response Message

When the class byte indicates a GlobalPlatform proprietary command (bit b8 = 1), the GET DATA response data field shall contain the TLV coded data object referred to in reference control parameters P1 and P2 of the command message, except where P1-P2 are set to '2F 00'.

Where the class byte indicates an ISO command (bit b8 = 0), the GET DATA response data field shall contain only the value of the TLV coded data object referred to in reference control parameters P1 and P2 of the command message, except where P1-P2 are set to '2F 00'.

11.3.3.1.1 Key Information Template ('E0')

When retrieving key information for the currently selected Application, the key information is returned in the template 'E0' where each Key Information Data data object is introduced by the tag 'C0'. Its structure depends on the value of Key Type. If the Key Type is coded on one byte (value other than 'FF'), the Key Information Data is structured as follows:

Table 11-28: Key Information Data Structure – Basic

Name	Length	Value	Presence
Key Identifier	1	Key Identifier value	Mandatory
Key Version Number	1	Key Version Number value	Mandatory
Key type of first (or only) key component	1	'00' - 'FE', see section 11.1.8 – *Key Type Coding*	Mandatory
Length of first (or only) key component	1	'01' - 'FF'	Mandatory
…	…	…	
Key type of last key component	1	'00' - 'FE', see section 11.1.8 – *Key Type Coding*	Conditional
Length of last key component	1	'01' - 'FF'	Conditional

If the Key Type is coded on two bytes (first byte = 'FF'), the Key Information Data is structured as follows:

Table 11-29: Key Information Data Structure – Extended

Name	Length	Value	Presence
Key Identifier	1	Key Identifier value	Mandatory
Key Version Number	1	Key Version Number value	Mandatory
Key type of first (or only) key component	2	'FF xx' – see section 11.1.8 – *Key Type Coding*	Mandatory
Length of first (or only) key component	2	'00 01' - '7F FF'	Mandatory
…	…	…	
Key type of last key component (if any)	2	'FF xx' – see section 11.1.8 – *Key Type Coding*	Conditional
Length of last key component (if any)	2	'00 01' - '7F FF'	Conditional
Length of Key Usage	1	'00' - '01'	Mandatory
Key usage	0 or 1	'xx' – see section 11.1.9	Conditional
Length of Key Access	1	'00' - '01'	Mandatory
Key Access	0 or 1	'xx' – see section 11.1.10	Conditional

For the Extended Key Information Data Structure,

- If Key Usage was not given at the time of key loading (e.g. PUT KEY Format 1), then Length of Key Usage shall be set to 0 or the Key Usage value shall conform to the predefined usages described in section 11.1.9.
- If Key Access was not given at the time of key loading (e.g. PUT KEY Format 1), then Length of Key Access shall be set to 0 or the Key Access value shall be set to '00' (access allowed for the Security Domain and any associated Applications) for a Secure Channel Key, and '01' (access allowed for the Security Domain only) for any other key.

The choice of using either the Basic or Extended Key Information Data Structure is implementation-dependent. Using the Basic Key Information Data Structure remains possible even in the presence of key sizes greater than 255 bytes: in this case, the indicated length shall be set to '00' (meaning ‘greater than or equal to 256 bytes’).

Additional rules for ECC public and private keys

Additional rules apply to the format of Key Information Data for ECC public and private keys:

- For key type 'B0' (ECC Public Key), the length of the component shall have one of the following values, excluding coding identifier '04':
  - '40' (ECC 256)
  - '60' (ECC 384)
  - '80' (ECC 512)
  - '84' (ECC 521)
- For key type 'B1' (ECC Private Key), the length of the component shall have one of the following values:
  - '20' (ECC 256)
  - '30' (ECC 384)
  - '40' (ECC 512)
  - '42' (ECC 521)
- If the key references preloaded ECC Curve Parameters (see section B.4.1), information about key type 'F0' (Key Parameter Reference) shall be provided (see below) but no information about individual Curve Parameters shall be present. If the key was loaded together with its own specific Curve Parameters, information about Curve Parameters shall not be present.
- For key type 'F0' (Key Parameter Reference), the length of the component is replaced by the value of the Key Parameter Reference (coded on 1 or 2 bytes).

11.3.3.1.2 Extended Card Resources ('FF21')

When retrieving Extended Card Resources Information with tag 'FF21', the response shall be coded as defined in [TS 102 226].

11.3.3.1.3 List of Applications ('2F00')

When retrieving the list of Applications present on the card, the response shall be coded as a series of Application Template data objects (tag '61') as defined in [ISO 7816-4]. An example of the response is as follows:

Table 11-30: List of On-Card Applications

Tag	Length	Name	Presence
'61'	7-n	Application Template	Mandatory
'4F'	5-16	Application AID	Mandatory
	...	...	
'61'	7-n	Application Template	Mandatory
'4F'	5-16	Application AID	Mandatory
	...	...	

11.3.3.2 Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

This command may return either a general error condition as listed in section 11.1.3 – *General Error Conditions* or the following error condition.

Table 11-31: GET DATA Error Conditions

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

11.4 GET STATUS Command

11.4.1 Definition and Scope

The GET STATUS command is used to retrieve Issuer Security Domain, Executable Load File, Executable Module, Application or Security Domain Life Cycle status information according to a given match/search criteria.

11.4.2 Command Message

The GET STATUS command message shall be coded according to the following table.

Table 11-32: GET STATUS Command Message

Code	Value	Meaning
CLA	'80' - '8F', 'C0' - 'CF', or 'E0' - 'EF'	See section 11.1.4
INS	'F2'	GET STATUS
P1	'xx'	Reference control parameter P1
P2	'xx'	Reference control parameter P2
Lc	'xx'	Length of data field
Data	'xx...'	Search criteria (and C-MAC if present)
Le	'00'	

11.4.2.1 Reference Control Parameter P1

Reference control parameter P1 is used to select a subset of statuses to be included in the response message. It is coded as follows:

Table 11-33: GET STATUS Reference Control Parameter P1

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1	-	-	-	-	-	-	-	Issuer Security Domain
-	1	-	-	-	-	-	-	Applications, including Security Domains
-	-	1	-	-	-	-	-	Executable Load Files
-	-	-	1	-	-	-	-	Executable Load Files and Executable Modules
-	-	-	-	X	X	X	X	RFU

The following values of the reference control parameter shall be supported:

'80' – Issuer Security Domain only. In this case the search criteria is ignored and the Issuer Security Domain information is returned. Only the ISD or an SD with the Global Registry privilege may accept this value.

'40' – Applications and Supplementary Security Domains only. Security Domains and Applications may be differentiated within response data based on their privileges.

'20' – Executable Load Files only.

'10' – Executable Load Files and their Executable Modules only.

11.4.2.2 Reference Control Parameter P2

The reference control parameter P2 controls the number of consecutive GET STATUS command and indicates the format of the response message. It shall be coded according to the following table.

Table 11-34: GET STATUS Reference Control Parameter P2

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
X	X	X	X	X	X	-	-	RFU
-	-	-	-	-	-	-	0	Get first or all occurrence(s)
-	-	-	-	-	-	-	1	Get next occurrence(s)
-	-	-	-	-	-	0	-	Deprecated
-	-	-	-	-	-	1	-	Response data structure according to Table 11-36 and 37

A GET STATUS [get next occurrence] command shall be rejected if no prior GET STATUS [get first or all occurrence(s)] was received within the current Application Session.

When retrieving the status of the Issuer Security Domain only, the get next occurrence indicator shall be rejected.

If P2.b2=0, then the behavior of the card is out of scope. The P2.b2=0 option relates to a response format described in older versions of this specification, which usage is now deprecated. The command may be rejected with an error or response data may be returned in a format which is now out of scope.

11.4.2.3 Data Field Sent in the Command Message

The data field is structured as follows:

Table 11-35: GET STATUS Command Data Field

Tag	Length	Name	Presence
'4F'	0-16	Application AID	Mandatory
'xx' or 'xxxx'	0-n	Other search criteria	Optional
...	...	...	...
'5C'	1-n	Tag list	Optional

The GET STATUS command message data field shall contain at least one TLV coded search qualifier: the AID (tag '4F'). It shall be possible to search for all the occurrences that match the selection criteria according to the reference control parameter P1 using a search criteria of '4F' '00'.

It shall be possible to search for all occurrences with the same RID.

Other search criteria may be added. In such cases, the additional search criteria shall be TLV coded and appended after the mandatory search criterion AID (tag '4F'). When not supported by the card, additional search criteria shall be ignored.

The search is limited to the Executable Load Files, Applications and Security Domains that are directly or indirectly associated with the on-card entity receiving the command. When the on-card entity receiving the command has the Global Registry privilege, the search applies to all Executable Load Files, Applications and Security Domains registered in the GlobalPlatform Registry.

The tag list (tag '5C') indicates to the card how to construct the response data for each on-card entity matching the search criteria. Its value field contains a concatenation of tags (without delimitation) indicating the data objects to include in the response. If an on-card entity that matches the search criteria is found in GlobalPlatform Registry without the corresponding data object, an error status may be returned or the requested data object may simply be omitted for that on-card entity. When not supported by the card, the tag list shall be ignored.

11.4.3 Response Message

11.4.3.1 Data Field Returned in the Response Message

If P2.b2=0 then the behavior of the card is out of scope.

If P2.b2=1 then, based on the search criteria present in the command data field and the selection criteria of reference control parameter P1 and P2, multiple occurrences of the data structures described in Table 11-36 and 37 may be returned.

Table 11-36: GlobalPlatform Application Data (TLV)

Tag	Length	Name
'E3'	Variable	GlobalPlatform Registry related data
'4F'	5-16	AID
'9F70'	1	Life Cycle State
'C5'	3	Privileges (byte 1 – byte 2 – byte 3); see section 11.1.2
'CF'	1	First Implicit Selection Parameter; see section 11.1.7
…	…	…
'CF'	1	Last Implicit Selection Parameter; see section 11.1.7
'C4'	1-n	Application's Executable Load File AID
'CC'	1-n	Associated Security Domain's AID

Table 11-37: GlobalPlatform Executable Load File Data (TLV)

Tag	Length	Name
'E3'	Variable	GlobalPlatform Registry related data
'4F'	5-16	AID
'9F70'	1	Life Cycle State
'CE'	1-n	Executable Load File Version Number (see Note 1)
'84'	1-n	First Executable Module AID (see Note 2)
…	...	…
'84'	1-n	Last Executable Module AID (see Note 2)
'CC'	1-n	Associated Security Domain's AID

Note 1: The Executable Load File Version Number format and contents depend on the format of the Load File. On a Java Card based card, this is a 2-byte version number reflecting the major and minor version attributes (in this order) of the Java Card CAP file.

Note 2: The Executable Modules (tag '84') shall be present if reference control parameter P1 indicates Executable Load Files and their Executable Modules only (P1.b5=1) and if the Runtime Environment supports Executable Modules. Of course, tag '84' is not present for an Executable Load File that does not contain any Executable Modules.

If no tag list (tag '5C') is present in the command data field or if the card does not support tag lists, then the content of template 'E3' shall be built according to the following rules:

- For the Issuer Security Domain (P1.b1=1) and for Applications (including Security Domains) (P1.b2=1), the AID (tag '4F'), the Life Cycle State (tag '9F70') and Privileges (tag 'C5') shall be present;
- For Executable Load Files, the AID (tag '4F') and the Life Cycle State (tag '9F70') shall be present, and
- Executable Modules (tag '84') shall be present according to Note 2 above.

If a tag list (tag '5C') is present in the command data field and tag lists are supported by the card, then the response shall only contain the data objects whose tags are listed in the tag list, the order of the data objects within template 'E3' being arbitrary.

11.4.3.2 Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

The command may return the following warning condition:

Table 11-38: GET STATUS Warning Condition

SW1	SW2	Meaning
'63'	'10'	More data available

Following status '63 10' a subsequent GET STATUS [get next occurrence(s)] may be issued to retrieve additional data.

This command may return either a general error condition as listed in section 11.1.3 – *General Error Conditions* or one of the following error conditions.

Table 11-39: GET STATUS Error Conditions

SW1	SW2	Meaning
'6A'	'88'	Referenced data not found
'6A'	'80'	Incorrect values in command data

11.5 INSTALL Command

11.5.1 Definition and Scope

The INSTALL command is issued to a Security Domain to initiate or perform the various steps required for Card Content management.

11.5.2 Command Message

The INSTALL command message shall be coded according to the following table.

Table 11-40: INSTALL Command Message

Code	Value	Meaning
CLA	'80' - '8F', 'C0' - 'CF', or 'E0' - 'EF'	See section 11.1.4
INS	'E6'	INSTALL
P1	'xx'	Reference control parameter P1
P2	'00', '01', or '03'	Reference control parameter P2
Lc	'xx'	Length of data field
Data	'xxxx…'	Install data (and C-MAC if present)
Le	'00'	

11.5.2.1 Reference Control Parameter P1

The reference control parameter P1 of the INSTALL command defines the specific role of the INSTALL command and also allows for command data to be longer than 255 bytes and to be segmented into arbitrary components and transmitted in a series of INSTALL commands. It is coded according to the following table.

Table 11-41: INSTALL Command Reference Control Parameter P1

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	-	-	-	-	-	-	-	Last (or only) command
1	-	-	-	-	-	-	-	More INSTALL commands
-	1	0	0	0	0	0	0	For registry update
-	0	1	0	0	0	0	0	For personalization
-	0	0	1	0	0	0	0	For extradition
-	0	0	0	1	-	-	0	For make selectable
-	0	0	0	-	1	-	0	For install
-	0	0	0	-	-	1	0	For load

Bits b8 to b1 shall be coded as follows:

b8 = 1 indicates that the command data is one of a sequence of components (and not the last), **b8 = 0** indicates that it is the last (or only) component.

b7 = 1 indicates that the GlobalPlatform Registry is to be updated, or that functions are to be restricted.

b6 = 1 indicates that the currently selected Security Domain shall personalize one of its associated Applications and a subsequent STORE DATA command is to be expected

b5 = 1 indicates that the Application shall be extradited.

b4 = 1 indicates that the Application shall be made selectable. This applies to an Application being installed or an Application that is already installed.

b3 = 1 indicates that the Application shall be installed.

b2 = 1 indicates that the Load File shall be loaded. A subsequent LOAD command is to be expected.

A combination of the [for install] and [for make selectable] options may apply. A combination of the [for load], [for install] and [for make selectable] options may also apply.

11.5.2.2 Reference Control Parameter P2

The reference control parameter P2 shall be set as follows:

'**00**' indicates that no information is provided.

'**01**' indicates the beginning of the combined Load, Install and Make Selectable process.

'**03**' indicates the end of the combined Load, Install and Make Selectable process.

11.5.2.3 Data Field Sent in the Command Message

The data field of the command message contains LV coded data. The LV coded data is represented without delimiters.

11.5.2.3.1 Data Field for INSTALL [for load]

The following table details the INSTALL [for load] command data field. It also applies to the first combined INSTALL [for load, install and make selectable] command data field.

Table 11-42: INSTALL [for load] Command Data Field

Name	Length	Value	Presence
Length of Load File AID	1	'05' - '10'	Mandatory
Load File AID	5-16	'xxxx...'	Mandatory
Length of Security Domain AID	1	'00' or '05' - '10'	Mandatory
Security Domain AID	0 or 5-16	'xxxx...'	Conditional
Length of Load File Data Block Hash	1	'00' - '7F'	Mandatory
Load File Data Block Hash	0-n	'xxxx...' – see section C.2	Conditional
Length of Load Parameters field	1-3	'00' - '80', or '81 80' - '81 FF', or '82 01 00' - '82 FF FF'	Mandatory
Load Parameters field	0-n	'xxxx...' – see section 11.5.2.3.7	Conditional
Length of Load Token	1-3	'00' - '80', or '81 80' - '81 FF', or '82 01 00' - '82 FF FF'	Mandatory
Load Token	0-n	'xxxx...' – see section C.4.1	Conditional

The Load Token is mandatory for Delegated Management (except for the combined [for load, install and make selectable] command), and for Authorized Management if the off-card entity is not the Security Domain Provider. In all other cases an error status shall be returned if a Load Token is present. The length field for a Load Token is as defined for ASN.1 BER-TLV (see [ISO 8825-1]) except that the length 128 may also be coded on one byte as '80'.

The Load File Data Block Hash is mandatory if a Load Token is present, if the Load File contains one or more DAP Blocks, or if the Load File Data Block is ciphered. In all other cases, the Load File Data Block Hash is optional and may be verified by the card. See section C.2 for details on acceptable Load File Data Block Hash algorithms.

The Load File AID and the Load Parameters shall be consistent with the information contained in the Load File Data Block (if any).

11.5.2.3.2 Data Field for INSTALL [for install]

The following table details the INSTALL [for install] command data field. It also applies to the final combined [for load, install and make selectable] command data field.

Table 11-43: INSTALL [for install] Command Data Field

Name	Length	Value	Presence
Length of Executable Load File AID	1	'00' or '05' - '10'	Mandatory
Executable Load File AID	0 or 5-16	'xxxx...'	Conditional
Length of Executable Module AID	1	'00' or '05' - '10'	Mandatory
Executable Module AID	0 or 5-16	'xxxx...'	Conditional
Length of Application AID	1	'05' - '10'	Mandatory
Application AID	5-16	'xxxx...'	Mandatory
Length of Privileges	1	'01', '03'	Mandatory
Privileges	1, 3	byte 1 – byte 2 – byte 3; see section 11.1.2	Mandatory
Length of Install Parameters field	1-3	'00' - '80', or '81 80' - '81 FF', or '82 01 00' - '82 FF FF'	Mandatory
Install Parameters field	2-n	'xxxx' – see section 11.5.2.3.7	Mandatory
Length of Install Token	1-3	'00' - '80', or '81 80' - '81 FF', or '82 01 00' - '82 FF FF'	Mandatory
Install Token	0-n	'xxxx...' – see section C.4.2 or C.4.7	Conditional

The Install Token is mandatory for Delegated Management and for Authorized Management if the off-card entity is not the Security Domain Provider. In all other cases an error status shall be returned if an Install Token is present. The length field for the Install Token is formatted as defined by ASN.1 BER-TLV (see [ISO 8825-1]) except that the length 128 may also be coded on one byte as '80'.

The presence of the Executable Load File AID is optional for the combined Load, Install and Make Selectable command. If present it shall match the Load File AID of the first combined Load, Install and Make Selectable command.

The Executable Module AID is the AID of the Executable Module previously loaded. The presence of the Executable Module depends on the requirements of the Run Time Environment.

GlobalPlatform cards use the instance AID to indicate the AID with which the installed Application will be selected.

The presence of the Privileges is required. If an Application is only being installed and not made selectable with the same INSTALL command the Card Reset privilege cannot be set.

The instance AID, the Privileges and the Application Specific Parameters shall be made known to the Application.

Note for backward compatibility: when receiving Privileges coded on one byte for any Application or Security Domain, the OPEN shall assign the second byte with the default values defined in section 6.6.2.

11.5.2.3.3 Data Field for INSTALL [for make selectable]

The following table details the INSTALL [for make selectable] command data field.

Table 11-44: INSTALL [for make selectable] Command Data Field

Name	Length	Value	Presence
Length of data	1	'00'	Mandatory
Length of data	1	'00'	Mandatory
Length of Application AID	1	'05' - '10'	Mandatory
Application AID	5-16	'xxxx...'	Mandatory
Length of Privileges	1	'01', '03'	Mandatory
Privileges	1, 3	byte 1 – byte 2 – byte 3; see section 11.1.2	Mandatory
Length of Make Selectable Parameters field	1-3	'00' - '80', or '81 80' - '81 FF', or '82 01 00' - '82 FF FF'	Mandatory
Make Selectable Parameters field	0-n	'xxxx' – see section 11.5.2.3.7	Conditional
Length of Make Selectable Token	1-3	'00' - '80', or '81 80' - '81 FF', or '82 01 00' - '82 FF FF'	Mandatory
Make Selectable Token	0-n	'xxxx...' – see section C.4.3	Conditional

If the Card Reset privilege is set in the Privileges field, the GlobalPlatform Registry shall be updated according to the rules defined in section 6.6 – *Privileges*. Any other privilege set in the Privileges field, regardless of its length, shall be ignored by the card.

The Make Selectable Token is mandatory for Delegated Management, and for Authorized Management if the off-card entity is not the Security Domain Provider. In all other cases an error status shall be returned if a Make Selectable Token is present. The length field for the Make Selectable Token is as defined for ASN.1 BER-TLV (see [ISO 8825-1]) except that the length 128 may also be coded on one byte as '80'.

11.5.2.3.4 Data Field for INSTALL [for extradition]

The following table details the INSTALL [for extradition] command data field.

Table 11-45: INSTALL [for extradition] Command Data Field

Name	Length	Value	Presence
Length of Security Domain AID	1	'05' - '10'	Mandatory
Security Domain AID	5-16	'xxxx...'	Mandatory
Length of data	1	'00'	Mandatory
Length of Application or Executable Load File AID	1	'05' - '10'	Mandatory
Application or Executable Load File AID	5-16	'xxxx...'	Mandatory
Length	1	'00'	Mandatory
Length of Extradition Parameters field	1-3	'00' - '80', or '81 80' - '81 FF', or '82 01 00' - '82 FF FF'	Mandatory
Extradition Parameters field	0-n	'xxxx' – see section 11.5.2.3.7	Conditional
Length of Extradition Token	1-3	'00' - '80', or '81 80' - '81 FF', or '82 01 00' - '82 FF FF'	Mandatory
Extradition Token	0-n	'xxxx...' – see section C.4.4	Conditional

The Security Domain AID indicates to which Security Domain this Application or Executable Load File is being extradited. The Security Domain with which this Application or Executable Load File is currently associated is the currently selected Application.

The Extradition Token is mandatory for Delegated Management, and for Authorized Management if the off-card entity is not the Security Domain Provider. In all other cases an error status shall be returned if an Extradition Token is present. The length field for the Extradition Token is as defined for ASN.1 BER-TLV (see [ISO 8825-1]) except that the length 128 may also be coded on one byte as '80'.

11.5.2.3.5 Data Field for INSTALL [for registry update]

The following table details the INSTALL [for registry update] command data field.

Table 11-46: INSTALL [for registry update] Command Data Field

Name	Length	Value	Presence
Length of Security Domain AID	1	'00' or '05' - '10'	Mandatory
Security Domain AID	0 or 5-16	'xxxx...'	Conditional
Length of data	1	'00'	Mandatory
Length of Application AID	1	'00' or '05' - '10'	Mandatory
Application AID	0 or 5-16	'xxxx...'	Conditional
Length of Privileges	1	'00', '01', '03'	Mandatory
Privileges	0, 1, 3	byte 1 – byte 2 – byte 3; see section 11.1.2	Conditional
Length of Registry Update Parameters field	1-3	'00' - '80', or '81 80' - '81 FF', or '82 01 00' - '82 FF FF'	Mandatory
Registry Update Parameters field	0-n	'xxxx' – see section 11.5.2.3.7	Conditional
Length of Registry Update Token	1-3	'00' - '80', or '81 80' - '81 FF', or '82 01 00' - '82 FF FF'	Mandatory
Registry Update Token	0-n	'xxxx...' – see section C.4.5	Conditional

The Security Domain AID, if present, indicates to which Security Domain this Application is being extradited. The Security Domain with which this Application is currently associated is the currently selected Application. The Application AID shall be present except when modifying the functionality of the OPEN.

For updating or revoking privileges the Privileges field shall be present. For updating implicit selection parameters or service parameters the corresponding tag within the Registry Update Parameters field shall be present – see section 11.5.2.3.7.

The Registry Update Token is mandatory for Delegated Management, and for Authorized Management if the off-card entity is not the Security Domain Provider. In all other cases an error status shall be returned if a Registry Update Token is present. The length field for the Registry Update Token is as defined for ASN.1 BER-TLV (see [ISO 8825-1]) except that the length 128 may also be coded on one byte as '80'.

All modifications requested in the INSTALL [for registry update] command data field (e.g. registry updates and extradition) shall either all succeed or no update shall occur if any one of the updates would fail for any reason.

11.5.2.3.6 Data Field for INSTALL [for personalization]

The following table details the INSTALL [for personalization] command data field.

Table 11-47: INSTALL [for personalization] Command Data Field

Name	Length	Value	Presence
Length of data	1	'00'	Mandatory
Length of data	1	'00'	Mandatory
Length of Application AID	1	'05' - '10'	Mandatory
Application AID	5-16	'xxxx...'	Mandatory
Length of data	1	'00'	Mandatory
Length of data	1	'00'	Mandatory
Length of data	1	'00'	Mandatory

11.5.2.3.7 INSTALL Command Parameters

The Load and Install Parameters fields are TLV structured values including optional System Specific Parameters and Application Specific Parameters. While the presence of the System Specific Parameters is optional for both loading and installation, even if they are present, it is not required that the system take heed of these; i.e. their presence shall be anticipated but the content may be ignored.

In all cases, the presence of the data objects in tag 'B6' and the sub-tags '42', '45', '5F20', and '93' is strongly recommended when using asymmetric Secure Channel Protocol '10' and '11'.

The following table identifies the possible tags for use in the Load Parameters field of the INSTALL [for load] command:

Table 11-48: Load Parameter Tags

Tag	Length	Name	Presence
'EF'	0-n	System Specific Parameters	Optional
'C6'	2 or 4	Non-volatile code Minimum Memory requirement	Optional
'C7'	2 or 4	Volatile data Minimum Memory requirement	Optional
'C8'	2 or 4	Non-volatile data Minimum Memory requirement	Optional
'CD'	1	Load File Data Block format id	Optional
'DD'	1-n	Load File Data Block Parameters	Conditional
'B6'	0-n	Control Reference Template for Digital Signature (Token)	Conditional
'42'	1-n	Identification Number of the Security Domain with the Token Verification privilege	Optional
'45'	1-n	Image Number of the Security Domain with the Token Verification privilege	Optional
'5F20'	1-n	Application Provider identifier	Optional
'93'	1-n	Token identifier/number (digital signature counter)	Optional

Some of the data objects in Load Parameters relate to memory management, which is an optional feature of a card. A Memory Management data object is a data object that represents an amount of memory resources counted in bytes. The minimum memory requirements (tags 'C6', 'C7', and 'C8') are coded as a 2-byte integer for values up to 32767 and a 4-byte integer above 32767 – see section 9.7 – *Memory Resource Management*. If both tags 'C6' and 'C8' are present and the implementation does not make any distinction between Non-Volatile Code and Non-Volatile Data Memory then the required minimum shall be the sum of both values.

When present, the Provider id (tag '5F20') shall be stored in the GlobalPlatform Registry for the new Executable Load File.

The presence of Load File Data Block Parameters depends on the value of Load File Data Block format id. These tags are reserved for implementations supporting proprietary Load File Data Block formats and may be ignored by other implementations. In particular, these tags are not intended to be used when the Load File Data Block is encoded as defined in the *Java Card™ 2.2.x Virtual Machine Specification*.

The following table identifies the possible tags for use in the Install Parameters field of the INSTALL [for install] command:

Table 11-49: Install Parameter Tags

Tag	Length	Name	Presence
'C9'	0-n	Application Specific Parameters	Mandatory
'EF'	0-n	System Specific Parameters	Optional
'C7'	2 or 4	Volatile Memory Quota	Optional
'C8'	2 or 4	Non-volatile Memory Quota	Optional
'CB'	2-n	Global Service Parameters	Optional
'D7'	2 or 4	Volatile Reserved Memory	Optional
'D8'	2 or 4	Non-volatile Reserved Memory	Optional
'CA'	1-n	[TS 102 226] specific parameter	Optional
'CF'	1	Implicit selection parameter	See note below
'EA'	0-n	[TS 102 226] specific template	Optional
'B6'	0-n	Control Reference Template for Digital Signature (Token)	Conditional
'42'	1-n	Identification Number of the Security Domain with the Token Verification privilege	Optional
'45'	1-n	Image Number of the Security Domain with the Token Verification privilege	Optional
'5F20'	1-n	Application Provider identifier	Optional
'93'	1-n	Token identifier/number (digital signature counter)	Optional

Note: Tag 'CF' may be present in Install Parameters for the combined INSTALL [for install and make selectable] command and in the final combined INSTALL [for load, install and make selectable] command. Tag 'CF' may occur in multiple INSTALL commands, appending to the collection of implicit selection parameters for the Application. Implicit Selection parameters cannot be deleted.

Some of the data objects in Install Parameters relate to memory management, which is an optional feature of a card. A Memory Management data object is a data object that represents an amount of memory resources counted in bytes. The memory quota values (tags 'C7' and 'C8') are coded as a 2-byte integer for values up to 32767 and a 4-byte integer above 32767 – see section 9.7 – *Memory Resource Management*.

When present, the Provider id (tag '5F20') shall be stored in the GlobalPlatform Registry for the newly installed Application.

When present in the INSTALL [for install] command, the information contained in tag 'C9' (length + data) shall be passed to the Application being installed.

If the Application being installed is a Security Domain, the information contained in tag 'C9' shall be BER-TLV encoded. Tags ranging from '70' to '7E' and from 'F0' to 'FE' are reserved for proprietary usage. All other tags are reserved by GlobalPlatform for usage in configuration documents.

When present in the INSTALL [for install] command, the information contained in tag 'CB' shall comprise one or more two-byte service parameters as defined in section 8.1.3 – *Global Service Parameters*.

The following table identifies the possible tags for use in the Make Selectable Parameters field of the INSTALL [for make selectable] command:

Table 11-50: Make Selectable Parameter Tags

Tag	Length	Name	Presence
'EF'	0-n	System Specific Parameters	Optional
'CF'	1	Implicit selection parameter	Optional
'B6'	0-n	Control Reference Template for Digital Signature (Token)	Conditional
'42'	1-n	Identification Number of the Security Domain with the Token Verification privilege	Optional
'45'	1-n	Image Number of the Security Domain with the Token Verification privilege	Optional
'5F20'	1-n	Application Provider identifier	Optional
'93'	1-n	Token identifier/number (digital signature counter)	Optional

The following table identifies the possible tags for use in the Extradition Parameters field of the INSTALL [for extradition] command:

Table 11-51: Extradition Parameter Tags

Tag	Length	Name	Presence
'B6'	0-n	Control Reference Template for Digital Signature (Token)	Conditional
'42'	1-n	Identification Number of the Security Domain with the Token Verification privilege	Optional
'45'	1-n	Image Number of the Security Domain with the Token Verification privilege	Optional
'5F20'	1-n	Application Provider identifier	Optional
'93'	1-n	Token identifier/number (digital signature counter)	Optional

The following table identifies the possible tags for use in the Registry Update Parameters field of the INSTALL [for registry update] command:

Table 11-52: Registry Update Parameter Tags

Tag	Length	Name	Presence
'EF'	0-n	System Specific Parameters	Optional
'CF'	1	Implicit selection parameter	Optional
'CB'	2-n	Global Service Parameters	Optional
'D9'	1	Restrict Parameter (see Table 11-53)	Conditional
'B6'	0-n	Control Reference Template for Digital Signature (Token)	Conditional
'42'	1-n	Identification Number of the Security Domain with the Token Verification privilege	Optional
'45'	1-n	Image Number of the Security Domain with the Token Verification privilege	Optional
'5F20'	1-n	Application Provider identifier	Optional
'93'	1-n	Token identifier/number (digital signature counter)	Optional

When no Application AID is present in the INSTALL [for registry update] command, the Restrict Parameter (tag 'D9') applies to OPEN. The effects of this function shall be irreversible. When an Application AID is present in the INSTALL [for registry update] command, the Restrict Parameter (tag 'D9') applies to the specific Security Domain identified by the Application AID.

If the Restrict Parameter is present and the card does not implement the option the card shall reject the command.

The Restrict Parameter (tag 'D9') lists the functionalities that shall be disabled. The following table indicates the functionalities that could possibly be disabled.

Table 11-53: Values for Restrict Parameter (Tag 'D9')

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
X	-	-	-	-	-	-	-	RFU
-	1	-	-	-	-	-	-	Registry Update
-	-	1	-	-	-	-	-	Personalization
-	-	-	1	-	-	-	-	Extradition
-	-	-	-	1	-	-	-	Make selectable
-	-	-	-	-	1	-	-	Install
-	-	-	-	-	-	1	-	Load
-	-	-	-	-	-	-	1	Delete

11.5.3 Response Message

A data field shall always be returned in the response message. Confirmation data or a single byte of '00' may be returned.

11.5.3.1 Data Field Returned in the Response Message

If an INSTALL [for load] command or an INSTALL [for personalization] command is being issued, a single byte of '00' shall be returned indicating that no additional data is present.

For INSTALL [for install], INSTALL [for make selectable], INSTALL [for install and make selectable], INSTALL [for extradition], INSTALL [for registry update] and INSTALL [for load, install and make selectable] (with P2 = '03') commands being issued to a Security Domain with the Delegated Management privilege, the data field may contain the confirmation of the install procedure.

The following table describes the structure of the INSTALL response data field. The length field for the Install Confirmation is coded according to ASN.1 BER-TLV (see [ISO 8825-1]).

Table 11-54: INSTALL Response Data Field

Name	Length	Value	Presence
Length of Install Confirmation	1-2	'00' - '7F' or '81 80' - '81 FF'	Mandatory
Install Confirmation	0-n	'xxxx...' – see section 11.1.6	Conditional

11.5.3.2 Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

This command may return either a general error condition as listed in section 11.1.3 – *General Error Conditions* or one of the following error conditions.

Table 11-55: INSTALL Error Conditions

SW1	SW2	Meaning
'65'	'81'	Memory failure
'6A'	'80'	Incorrect parameters in data field
'6A'	'84'	Not enough memory space
'6A'	'88'	Referenced data not found

11.6 LOAD Command

11.6.1 Definition and Scope

This section defines the structure of the Load File transmitted in the LOAD command data field for loading a Load File. The runtime environment internal handling or storage of the Load File is beyond the scope of this Specification.

Multiple LOAD commands may be used to transfer a Load File to the card. The Load File is divided into smaller components for transmission. Each LOAD command shall be numbered starting at '00'. The LOAD command numbering shall be strictly sequential and increments by one. The card shall be informed of the last block of the Load File.

After receiving the last block of the Load File, the card shall execute the internal processes necessary for the Load File and any additional processes identified in the INSTALL [for load] command that preceded the LOAD commands.

11.6.2 Command Message

The LOAD command message is coded according to the following table.

Table 11-56: LOAD Command Message Structure

Code	Value	Meaning
CLA	'80' - '8F', 'C0' - 'CF', or 'E0' - 'EF'	See section 11.1.4
INS	'E8'	LOAD
P1	'xx'	Reference control parameter P1
P2	'xx'	Block number
Lc	'xx'	Length of data field
Data	'xxxx..'	Load data (and C-MAC if present)
Le	'00'	

11.6.2.1 Reference Control Parameter P1

The following table describes the coding of the reference control parameter P1 indicating whether the block contained in the command message is one in a sequence of blocks or the last block in the sequence.

Table 11-57: LOAD Command Reference Control Parameter P1

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	-	-	-	-	-	-	-	More blocks
1	-	-	-	-	-	-	-	Last block
-	X	X	X	X	X	X	X	RFU

11.6.2.2 Reference Control Parameter P2 – Block Number

Reference control parameter P2 contains the block number, and shall be coded sequentially from '00' to 'FF'.

11.6.2.3 Data Field Sent in the Command Message

The data field of the command message contains a portion of the Load File.

A complete GlobalPlatform Load File is structured as defined in the following table:

Table 11-58: Load File Structure

Tag	Length	Name	Presence
'E2'	1-n	DAP Block	Conditional
'4F'	5-16	Security Domain AID	Mandatory
'C3'	1-n	Load File Data Block Signature	Mandatory
:	:	:	
:	:	:	
'E2'	1-n	DAP Block	Conditional
'4F'	5-16	Security Domain AID	Mandatory
'C3'	1-n	Load File Data Block Signature	Mandatory
'C4'	1-n	Load File Data Block	Conditional
'D3'	8 or 16	ICV	Optional
'D4'	1-n	Ciphered Load File Data Block	Conditional

The data objects defined in Table 11-58 shall be coded according to ASN.1 Basic Encoding Rules, and in particular, the length fields (e.g. one byte for length values up to 127).

A DAP Block shall be present within a Load File if the associated Security Domain has the DAP Verification privilege or a Security Domain with the Mandated DAP Verification privilege is present. A properly constructed DAP Block shall contain the Security Domain AID and the Load File Data Block Signature.

A Load File may contain multiple DAP Blocks; each DAP Block being introduced by tag 'E2' and preceding the Load File Data Block.

If the associated Security Domain has the Ciphered Load File Data Block privilege, the Load File Data Block shall be sent encrypted using tag 'D4'. In this case, the encryption of the Load File Data Block shall be performed according to the rules described in section C.6. Otherwise the Load File Data Block shall be sent unencrypted using tag 'C4'.

11.6.3 Response Message

A data field shall always be returned in the response message. The content of the data field is only relevant in the case of Delegated Management; i.e. if a last LOAD command is being issued to a Security Domain with the Delegated Management privilege, a Receipt may be present in the data field depending on the security policy of the Card Issuer.

11.6.3.1 Data Field Returned in the Response Message

If the LOAD command does not contain the last block in the sequence, a single byte of '00' shall be returned indicating that no additional data is present.

If the Issuer Security Domain processes the LOAD command containing the last block in the sequence, a single byte of '00' shall be returned indicating that no additional data is present.

For a LOAD command containing the last block in the sequence being issued to a Security Domain with the Delegated Management privilege, the data field may contain the confirmation of the load procedure. The overall length of the response message shall not exceed 256 bytes.

The following table describes the structure of the LOAD response data field. The length field for the Load Confirmation is coded according to ASN.1 BER-TLV (see [ISO 8825-1]).

Table 11-59: LOAD Response Data Field

Name	Length	Value	Presence
Length of Load Confirmation	1-2	'00' - '7F' or '81 80' - '81 FF'	Mandatory
Load Confirmation	0-n	'xxxx...' – see section 11.1.6	Conditional

11.6.3.2 Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

This command may return either a general error condition as listed in section 11.1.3 – *General Error Conditions* or one of the following error conditions.

Table 11-60: LOAD Error Conditions

SW1	SW2	Meaning
'65'	'81'	Memory failure
'6A'	'84'	Not enough memory space

11.7 MANAGE CHANNEL Command

11.7.1 Definition and Scope

The MANAGE CHANNEL command is processed by the OPEN on cards that are aware of logical channels. It is used to open and close Supplementary Logical Channels. The Basic Logical Channel (channel number zero) can never be closed.

11.7.2 Command Message

The MANAGE CHANNEL command message shall be coded according to the following table.

Table 11-61: MANAGE CHANNEL Command Message

Code	Value	Meaning
CLA	'00' - '03' or '40' - '4F'	See section 11.1.4
INS	'70'	MANAGE CHANNEL
P1	'xx'	Reference control parameter P1
P2	'xx'	Reference control parameter P2
Lc		Not present
Le		'01' if P1 = '00' and not present if P1 = '80'

11.7.2.1 Reference Control Parameter P1

Reference control parameter P1 is used to indicate whether a Supplementary Logical Channel is being opened or closed.

The following values of the reference control parameter may apply:

'80' – Close the Supplementary Logical Channel identified in reference control parameter P2.

'00' – Open the next available Supplementary Logical Channel.

11.7.2.2 Reference Control Parameter P2

If a Supplementary Logical Channel is being closed (reference control parameter P1 is '80'), the reference control parameter P2 identifies the Supplementary Logical Channel to be closed (i.e. '01', '02', or '03'). For a card supporting further interindustry logical channels, the reference control parameter may identify further interindustry Supplementary Logical Channels (i.e. '04' to '13').

If a Supplementary Logical Channel is being opened (reference control parameter P1 is '00'), the reference control parameter P2 indicates that the next available Supplementary Logical Channel is being opened (i.e. '00').

A card supporting logical channel assignment by off-card entities may accept a reference control parameter P2 indicating the Supplementary Logical Channel number to be opened (i.e. '01', '02', or '03'). A card supporting further interindustry logical channel assignment by off-card entities may accept a reference control parameter P2 indicating the further interindustry Supplementary Logical Channel number to be opened (i.e. '04' to '13').

11.7.2.3 Data Field Sent in the Command Message

The data field of the command message is not present.

11.7.3 Response Message

11.7.3.1 Data Field Returned in the Response Message

The data field of the response message is only present if a Supplementary Logical Channel is being opened.

Depending on the number of logical channels supported by the card, the following values of the data field of the response message may apply:

'01', '02', or '03' – Supplementary Logical Channel opened.

For a card supporting further interindustry logical channels, and depending on the number of Supplementary Logical Channels supported by the card, the following values of the data field of the response message may apply:

'04' to '13' – Supplementary Logical Channel opened.

11.7.3.2 Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

The command may return the following warning:

Table 11-62: MANAGE CHANNEL Warning Conditions

SW1	SW2	Meaning
'62'	'00'	Logical Channel already closed

This command may return either a general error condition as listed in section 11.1.3 – *General Error Conditions* or one of the following error conditions.

Table 11-63: MANAGE CHANNEL Error Conditions

SW1	SW2	Meaning
'68'	'82'	Secure messaging not supported
'6A'	'81'	Function not supported e.g. card Life Cycle State is CARD_LOCKED

11.8 PUT KEY Command

11.8.1 Definition and Scope

The PUT KEY command is used to either:

- Replace an existing key with a new key: The new key has the same or a different Key Version Number but the same Key Identifier as the key being replaced;
- Replace multiple existing keys with new keys: The new keys have the same or a different Key Version Number (identical for all new keys) but the same Key Identifiers as the keys being replaced;
- Add a single new key: The new key has a different combination Key Identifier / Key Version Number than that of the existing keys;
- Add multiple new keys: The new keys have different combinations of Key Identifiers / Key Version Number (identical to all new keys) than that of the existing keys;

When the key management operation requires multiple PUT KEY commands, chaining of the multiple PUT KEY commands is recommended to ensure integrity of the operation.

In this version of the Specification the public values of asymmetric keys are presented in clear text.

11.8.2 Command Message

The PUT KEY command message is coded according to the following table:

Table 11-64: PUT KEY Command Message

Code	Value	Meaning
CLA	'80' - '8F', 'C0' - 'CF' or 'E0' - 'EF'	See section 11.1.4
INS	'D8'	PUT KEY
P1	'xx'	Reference control parameter P1
P2	'xx'	Reference control parameter P2
Lc	'xx'	Length of data field
Data	'xxxx..'	Key data (and C-MAC if present)
Le	'00'	

11.8.2.1 Reference Control Parameter P1

Reference control parameter P1 defines a Key Version Number and whether more PUT KEY commands will follow this one.

The Key Version Number identifies a key or group of keys that is 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'.

The reference control parameter P1 of the PUT KEY command message is coded according to the following table:

Table 11-65: PUT KEY Reference Control Parameter P1

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	-	-	-	-	-	-	-	Last (or only) command
1	-	-	-	-	-	-	-	More PUT KEY commands
-	X	X	X	X	X	X	X	Key Version Number

11.8.2.2 Reference Control Parameter P2

Reference control parameter P2 defines a Key Identifier and whether one or multiple keys are contained in the data field.

When one key is contained in the command message data field, reference control parameter P2 indicates the Key Identifier of this key. When multiple keys are contained in the command message data field, reference control parameter P2 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'.

The reference control parameter P2 of the PUT KEY command message is coded according to the following table:

Table 11-66: PUT KEY Reference Control Parameter P2

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	-	-	-	-	-	-	-	Single key
1	-	-	-	-	-	-	-	Multiple keys
-	X	X	X	X	X	X	X	Key Identifier

11.8.2.3 Data Field Sent in the Command Message

The command message data field contains a new Key Version Number (coded from '01' to '7F') followed by one or multiple key data fields as represented in the following diagram (with optionally a second and third key):

Table 11-67: Structure of the PUT KEY Command Data Field

Key version number of keys present in the data field
Key data field (implicit key id P2+0)
Key data field (implicit key id P2+1)
Key data field (implicit key id P2+2)

The Key Version Number field defines either:

- The version number of a new key or group of keys to be created on the card (Key Version Number indicated in P1 is set to zero); or
- The version number of a new key or group of keys that will replace an existing key or group of keys (Key Version Number indicated in P1 is different from zero).

If the data field contains multiple keys, the keys all share the same Key Version Number and the sequence in the command data field reflects the incremental sequence of the Key Identifiers.

11.8.2.3.1 Format of the Key Data Field

The key data field may be formatted according to the Basic Format or the Extended Format, which are described in

Table 11-68 and Table 11-69. In the Basic Format, the Key Type is coded on one byte and has a value other than 'FF'. In the Extended Format, the Key Type is coded on two bytes with a first byte of 'FF'.

In both formats:

- A cryptographic key may be composed of one or several key components. The value of each key component is provided by a Key Component Block. The format of the Key Component Block is described in section 11.8.2.3.2.
- All lengths shall be coded as defined for ASN.1 BER-TLV (see [ISO 8825-1]) except that the length 128 may also be coded on one byte as '80' (for backward compatibility).

Standardized key types being coded with values greater than '7F', the length field for the key check value can be unambiguously detected.

Table 11-68: Key Data Field – Format 1 (Basic Format)

Name	Length	Value	Presence
Key type of first or only key component	1	'00' - 'FE' – see section 11.1.8 – *Key Type Coding*	Mandatory
Length of first or only Key Component Block	1-3	'01' - '80', or '81 80' - '81 FF', or '82 01 00' - '82 FF FF'	Mandatory
First or only Key Component Block	1-n	'xxxx...' (see section 11.8.2.3.2)	Mandatory
…	…	…	…
Key type of last key component (if more than one)	1	'00' - 'FE' – see section 11.1.8 – *Key Type Coding*	Conditional
Length of last Key Component Block	1-3	'01' - '80', or '81 80' - '81 FF', or '82 01 00' - '82 FF FF'	Conditional
Last Key Component Block	1-n	'xxxx...' (see section 11.8.2.3.2)	Conditional
Length of key check value	1	'00' - '7F'	Mandatory
Key check value	0-n	'xxxx...'	Conditional

Table 11-69: Key Data Field – Format 2 (Extended Format)

Name	Length	Value	Presence
Key type of the first or only key component	2	'FF xx' – see section 11.1.8 – *Key Type Coding*	Mandatory
Length of first or only Key Component Block	1-3	'01' - '80', or '81 80' - '81 FF', or '82 01 00' - '82 FF FF'	Mandatory
First or only Key Component Block	1-n	'xxxx...' (see section 11.8.2.3.2)	Mandatory
…	…	…	…
Key type of the last key component (if more than one)	2	'FF xx' – see section 11.1.8 – *Key Type Coding*	Conditional
Length of last Key Component Block	1-3	'01' - '80', or '81 80' - '81 FF', or '82 01 00' - '82 FF FF'	Conditional
Last Key Component Block	1-n	'xxxx…' (see section 11.8.2.3.2)	Conditional
Length of key check value	1	'00' - '7F'	Mandatory
Key check value	0-n	'xxxx...'	Conditional
Length of key usage qualifier	1	'00' - '7F'	Mandatory
Key usage qualifier	0-n	'xxxx...' see section 11.1.9 – *Key Usage Qualifier Coding*	Conditional
Length of key access	1	'00' - '7F'	Mandatory
Key access	0-n	'xxxx...' see section 11.1.10 – *Key Access Coding*	Conditional

11.8.2.3.2 Format of the Key Component Block

For a public key component, the key component value does not need to be encrypted and the Key Component Block only contains the clear-text key component value.

For a secret or private key component, the key component value must be encrypted and the following rules apply:

- If the length of the key component value is not a multiple of the block size of the encryption algorithm (i.e. 8 bytes for DES, 16 bytes for AES), then the key component value must be right-padded (with arbitrary bytes) prior to encryption. In this case, the Key Component Block shall be formatted as described in Table 11-70. This format allows providing the length of the clear-text key component value, which is required for the Security Domain processing the PUT KEY command to recover the key component value after decryption.
- If the length of the key component value is a multiple of the block size of the encryption algorithm (i.e. 8 bytes for DES, 16 bytes for AES), then the key component value does not need to be padded prior to encryption and both clear-text and encrypted key component values have the same length. In this case, two options exists: The Key Component Block may be formatted according to Table 11-70 or may simply contain the encrypted key component value as shown in Table 11-71.

Table 11-70: Format of Key Component Block – Padding Present if Needed

Name	Length	Value
Length (in bytes) of key component value	1-3 bytes	'01' – '80', or '81 80' – '81 FF', or '82 01 00' – '82 FF FF'
Encrypted key component value	Multiple of encryption algorithm block size	'xxxx…'

Table 11-71: Format of Key Component Block – Padding Not Present

Name	Length	Value
Encrypted key component value	Multiple of encryption algorithm block size	'xxxx…'

11.8.2.3.3 Processing Rules

When replacing keys, the new keys shall be presented to the card with the same characteristics as the previous ones: in other words, it is not possible to change the size and the associated cryptographic algorithm of an existing key.

When chaining is used to load or replace a key comprised of more than one component, the subsequent commands must refer to the same Key Identifier and the same Key Version Number as the first PUT KEY command used for the first key component.

If the data field contains multiple keys or key components, the card must handle the multiple keys or key components in an atomic manner. When PUT KEY commands are chained (i.e. bit b8 of P1 set to 1), the card must handle the multiple key components transferred in the chain of PUT KEY commands (until and including the first PUT KEY command with bit b8 of P1 = 0) in an atomic manner.

For a secret or private key component, the key component value is assumed to be encrypted and the following rules apply:

- If the length of the Key Component Block is not a multiple of the block size of the encryption algorithm (i.e. 8 bytes for DES, 16 bytes for AES), then it shall be assumed that the key component value was right-padded prior to encryption and that the Key Component Block was formatted as described in Table 11-70. In this case, the first byte(s) of the Key Component Block provides the actual length of the key component value, which allows recovering the clear-text key component value after decryption of the encrypted key component value and removal of padding bytes.
- If the length of the Key Component Block is a multiple of the block size of the encryption algorithm (i.e. 8 bytes for DES, 16 bytes for AES), then it shall be assumed that no padding bytes were added before encrypting the key component value and that the Key Component Block is only composed of the encrypted key component value (as shown in Table 11-71). In this case, the clear-text key component value is simply recovered by decrypting the Key Component Block.

If present, the Key Check Value shall be verified. For all key types described in section B.6, the Key Check Value shall be present.

Upon successful creation or update of a key, the Key Information Data which may later be returned by the GET DATA command shall be updated.

11.8.2.3.4 Additional Rules for AES Keys

It is recommended to encrypt AES keys using an AES Key-DEK with the same or higher strength.

11.8.2.3.5 Additional Rules for ECC Keys

All ECC key components or field parameters to be loaded shall be contained in one key data field of the PUT KEY command. The command itself may be chained (see section 11.1.5.1).

The key component value shall be formatted as follows:

1. The public key component and the G component (i.e. key types 'B0' and 'B5') shall be formatted using uncompressed encoding as specified in [TR 03111] section 3.2.1, with most significant byte coming first.
2. Other key components shall be encoded as follows: Fixed length related to the byte length of the key, zero-padding added before the most significant bit as needed. This does not apply to the cofactor of order of generator, which shall be encoded with the smallest number of bytes.

No key check mechanism for ECC keys is defined; i.e. the length of the key check value shall be '00'.

For the loading of ECC curve parameters into a security domain, PUT KEY shall contain the following key components:

Table 11-72: Loading of ECC Curve Parameters

Key Type	Key Component	Presence	Data Content Encrypted
'F0'	Key parameter reference	Mandatory	No
'B2'	ECC field parameter P (field specification)	Mandatory	No
'B3'	ECC field parameter A (first coefficient)	Mandatory	No
'B4'	ECC field parameter B (second coefficient)	Mandatory	No
'B5'	ECC field parameter G (generator)	Mandatory	No
'B6'	ECC field parameter N (order of generator)	Mandatory	No
'B7'	ECC field parameter k (cofactor of order of generator)	Optional	No

For the loading of a public or a private ECC key that references global or local ECC curve parameters, PUT KEY shall contain the following key components:

Table 11-73: Loading of ECC Key with Parameter Reference

Key Type	Key Component	Presence	Data Content Encrypted
'B0' or 'B1'	ECC public or private key	Mandatory	No for public / yes for private key
'F0'	Key parameter reference	Mandatory	No

For the loading of a public or a private ECC key with its own ECC curve parameters, PUT KEY shall contain the following key components:

Table 11-74: Loading of ECC Key with Own Parameters

Key Type	Key Component	Presence	Data Content Encrypted
'B0' or 'B1'	ECC public or private key	Mandatory	No for 'B0' / Yes for 'B1'
'B2'	ECC field parameter P (field specification)	Mandatory	No
'B3'	ECC field parameter A (first coefficient)	Mandatory	No
'B4'	ECC field parameter B (second coefficient)	Mandatory	No
'B5'	ECC field parameter G (generator)	Mandatory	No
'B6'	ECC field parameter N (order of generator)	Mandatory	No
'B7'	ECC field parameter k (cofactor of order of generator)	Optional	No

11.8.2.3.6 Additional Rules for RSA Keys

All RSA key components or field parameters to be loaded shall be contained in one key data field of the PUT KEY command. The command itself may be chained (see section 11.1.5.1).

No key check mechanism for RSA keys is defined; i.e. the length of the key check value shall be '00'.

For the loading of a public RSA key, PUT KEY shall contain the following key components:

Table 11-75: Loading of RSA Public Key

Key Type	Key Component	Presence	Data Content Encrypted
'A1'	RSA Key Modulus (clear-text)	Mandatory	No
'A0'	Public Key Exponent	Mandatory	No

For the loading of a private RSA key, PUT KEY shall contain the following key components:

Table 11-76: Loading of RSA Private Key

Key Type	Key Component	Presence	Data Content Encrypted
'A1' or 'A2'	RSA Key Modulus (clear-text or encrypted)	Mandatory	No for 'A1' / Yes for 'A2'
'A3'	Private Key Exponent	Mandatory	Yes

For the loading of a private RSA key in Chinese Remainder Theorem (CRT) format, PUT KEY shall contain the following key components:

Table 11-77: Loading of RSA Private Key in CRT Format

Key Type	Key Component	Presence	Data Content Encrypted
'A4'	RSA Private Key - Chinese Remainder P component	Mandatory	Yes
'A5'	RSA Private Key - Chinese Remainder Q component	Mandatory	Yes
'A6'	RSA Private Key - Chinese Remainder PQ component ($q^{-1} \bmod p$)	Mandatory	Yes
'A7'	RSA Private Key - Chinese Remainder DP1 component ($d \bmod (p-1)$)	Mandatory	Yes
'A8'	RSA Private Key - Chinese Remainder DQ1 component ($d \bmod (q-1)$)	Mandatory	Yes

11.8.3 Response Message

11.8.3.1 Data Field Returned in the Response Message

The data field of the response message contains in clear text the Key Version Number followed by the key check value(s) not preceded by a length, if any, as presented in the command message data field. The personalization server may use the returned Key Version Number and key check value(s) to verify the correct loading of the key(s).

11.8.3.2 Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

This command may return either a general error condition as listed in section 11.1.3 – *General Error Conditions* or one of the following error conditions.

Table 11-78: PUT KEY Error Conditions

SW1	SW2	Meaning
'65'	'81'	Memory failure
'6A'	'80'	Wrong data
'6A'	'84'	Not enough memory space
'6A'	'88'	Referenced data not found
'94'	'84'	Algorithm not supported (DEPRECATED: Use '6A80' instead.)
'94'	'85'	Invalid key check value (DEPRECATED: Use '6982' instead.)
'69'	'82'	Invalid key check value

11.9 SELECT Command

11.9.1 Definition and Scope

The SELECT command is used for selecting an Application. The OPEN only processes SELECT commands indicating the SELECT [by name] option. All options other than SELECT [by name] shall be passed to the currently selected Security Domain or Application on the indicated logical channel.

11.9.2 Command Message

The SELECT command is coded according to the following table:

Table 11-79: SELECT Command Message

Code	Value	Meaning
CLA	'00' - '03' or '40' - '4F'	See section 11.1.4
INS	'A4'	SELECT
P1	'xx'	Reference control parameter P1
P2	'xx'	Reference control parameter P2
Lc	'xx'	Length of AID
Data	'xxxx..'	AID of Application to be selected
Le	'00'	

11.9.2.1 Reference Control Parameter P1

Reference control parameter P1 shall be coded according to the following table.

Table 11-80: SELECT Reference Control Parameter P1

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0	0	0	0	1	0	0	Select by name

A Security Domain or Application may support other values as defined in [ISO 7816-4].

11.9.2.2 Reference Control Parameter P2

Reference control parameter P2 shall be coded according to the following table.

Table 11-81: SELECT Reference Control Parameter P2

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

OPEN or the underlying runtime environment may support other values as defined in [ISO 7816-4].

11.9.2.3 Data Field Sent in the Command Message

The data field of the command shall contain the AID of the Application to be selected. The Lc and data field of the SELECT command may be omitted if the Issuer Security Domain is being selected. In this case, Le shall be set to '00' and the command is a case 2 command according to [ISO 7816-4].

11.9.3 Response Message

11.9.3.1 Data Field Returned in the Response Message

The SELECT response data field consists of information specific to the selected Application.

The coding of the File Control Information for the Issuer Security Domain and Security Domains shall be according to the following table.

Table 11-82: File Control Information

Tag	Description	Presence
'6F'	File Control Information (FCI template)	Mandatory
'84'	Application / file AID	Mandatory
'A5'	Proprietary data	Mandatory
'73'	Security Domain Management Data (see section H.3 for detailed coding)	Optional
'9F6E'	Application production Life Cycle data	Optional
'9F65'	Maximum length of data field in command message	Mandatory

Additional data objects may be returned within the FCI template.

11.9.3.2 Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

The command may return the following warning condition when the Security Domain with the Final Application privilege is being selected.

Table 11-83: SELECT Warning Condition

SW1	SW2	Meaning
'62'	'83'	Card Life Cycle State is CARD_LOCKED

This command may return either a general error condition as listed in section 11.1.3 – *General Error Conditions* or one of the following error conditions.

Table 11-84: SELECT Error Conditions

SW1	SW2	Meaning
'68'	'82'	Secure messaging not supported
'6A'	'81'	Function not supported e.g. card Life Cycle State is CARD_LOCKED
'6A'	'82'	Selected Application / file not found

11.10 SET STATUS Command

11.10.1 Definition and Scope

The SET STATUS command shall be used to modify the card Life Cycle State or the Application Life Cycle State.

11.10.2 Command Message

The SET STATUS command message is coded according to the following table.

Table 11-85: SET STATUS Command Message

Code	Value	Meaning
CLA	'80' - '8F', 'C0' - 'CF', or 'E0' - 'EF'	See section 11.1.4
INS	'F0'	SET STATUS
P1	'xx'	Status type
P2	'xx'	State control
Lc	'xx'	Length of data field
Data	'xxxxx…'	AID of Application (and C-MAC if present)
Le		Not present

11.10.2.1 Reference Control Parameter P1 – Status Type

The status type of the SET STATUS command message indicates if the change in the Life Cycle State applies to the Issuer Security Domain, Supplementary Security Domains or an Application. The status type can also indicate that the command applies to a Security Domain and all its associated Applications: this only applies for transition to, and back from, the LOCKED state. The status type shall be coded according to the following table.

Table 11-86: SET STATUS – Status Type

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1	0	0	-	-	-	-	-	Indicate Issuer Security Domain
0	1	0	-	-	-	-	-	Indicate Application or Supplementary Security Domain
0	1	1	-	-	-	-	-	Indicate Security Domain and its associated Applications
-	-	-	X	X	X	X	X	RFU

Note: When the SET STATUS command applies for a Security Domain and its Associated Applications, then the SET STATUS command applies to the sub-hierarchy of the Security Domain indicated in the command data field. If the SET STATUS is used to lock all Applications, all Applications remain already locked, and no error status is returned. If the SET STATUS is used to unlock all Applications, all Applications already unlocked remain unlocked, and no error status is returned.

11.10.2.2 Reference Control Parameter P2 – State Control

The state control of the SET STATUS command message indicates the state transition required and shall be coded according to section 11.1.1 – *Life Cycle State Coding*.

For the Issuer Security Domain (which inherits the card Life Cycle State), the parameter shall be coded according to Table 11-6 and abide by the transitioning rules as diagrammed in Figure 5-1. A Security Domain with the Card Terminate or the Card Lock privilege is able to terminate or lock/unlock, respectively, the card using the SET STATUS command with P1 set to '80'.

For a Security Domain setting its own Life Cycle State using this command, the only possible transitions are to the PERSONALIZED or LOCKED state: The parameter shall be coded according to Table 11-5.

For an Application setting its own Life Cycle State using this command, the parameter shall be coded according to Table 11-4 and abide by the transitioning rules as depicted in Figure 5-2.

For a Security Domain setting the Life Cycle State of another Application or Supplementary Security Domain, the only possible transitions are to the LOCKED state and subsequently back to the previous state. The only relevant bit of this parameter would therefore be bit b8 (all other bits are ignored):

- b8 = 1 indicates a transition to the LOCKED state;
- b8 = 0 indicates a transition (from LOCKED) back to the previous state.

A request to transition a Security Domain or an Application to its current Life Cycle State shall be rejected.

11.10.2.3 Data Field Sent in the Command Message

The data field shall contain the AID of the target Application or Security Domain for which a Life Cycle change is requested. The on-card entity receiving the command shall be directly or indirectly associated with this target Application or Security Domain or shall have the relevant privilege. If reference control parameter P1 is '80' the content of the command data field shall be ignored. If the command relates to a Security Domain and all its associated Applications, the data field shall contain the AID of the Security Domain.

11.10.3 Response Message

11.10.3.1 Data Field Returned in the Response Message

The data field of the response message shall not be present.

11.10.3.2 Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

This command may return either a general error condition as listed in section 11.1.3 – *General Error Conditions* or one of the following error conditions.

Table 11-87: SET STATUS Error Conditions

SW1	SW2	Meaning
'6A'	'80'	Incorrect values in command data
'6A'	'88'	Referenced data not found

11.11 STORE DATA Command

11.11.1 Definition and Scope

The STORE DATA command is used to transfer data to an Application or the Security Domain processing the command.

The Security Domain determines if the command is intended for itself or an Application depending on a previously received command. If a preceding command was an INSTALL [for personalization] command, the STORE DATA command is destined for an Application.

Multiple STORE DATA commands are used to send data to the Application or Security Domain by breaking the data into smaller components for transmission. The Security Domain shall be informed of the last block.

A personalization session starts when a Security Domain receives a valid INSTALL [for personalization] command designating an Application (implementing either the Application or the Personalization interface) to which the Security Domain shall forward subsequently received STORE DATA commands.

A personalization session ends when:

- The card is reset;
- The Security Domain is deselected (i.e. an Application – another or the same – is selected on the same logical channel);
- The Security Domain is selected on the same or another logical channel;
- The Secure Channel session (if any) established by the Security Domain is reset, possibly by the targeted Application (see conditions triggering Secure Channel Termination described in section 10.2.3);
- The Security Domain receives an INSTALL [for personalization] command (starting a new personalization session for another application);
- The Security Domain receives a STORE DATA command indicating P1.b8=1 (last block);

Any STORE DATA command received out of such a personalization session shall be processed by the Security Domain itself.

11.11.2 Command Message

The STORE DATA command message shall be coded according to the following table.

Table 11-88: STORE DATA Command Message

Code	Value	Meaning
CLA	'80' - '8F', 'C0' - 'CF', or 'E0' - 'EF'	See section 11.1.4
INS	'E2'	STORE DATA
P1	'xx'	Reference control parameter P1
P2	'xx'	Block number
Lc	'xx'	Length of data field
Data	'xxxxx…'	Application data and MAC (if present)
Le		Not present

11.11.2.1 Reference Control Parameter P1

Reference control parameter P1 shall be coded according to the following table.

Table 11-89: STORE DATA Reference Control Parameter P1

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	-	-	-	-	-	-	-	More blocks
1	-	-	-	-	-	-	-	Last block
-	0	0	-	-	-	-	-	No general encryption information or non-encrypted data
-	0	1	-	-	-	-	-	Application dependent encryption of the data
-	1	0	-	-	-	-	-	RFU (encryption indicator)
-	1	1	-	-	-	-	-	Encrypted data
-	-	-	0	0	-	-	-	No general data structure information
-	-	-	0	1	-	-	-	DGI format of the command data field
-	-	-	1	0	-	-	-	BER-TLV format of the command data field
-	-	-	1	1	-	-	-	RFU (data structure information)
-	-	-	-	-	-	-	0	ISO case 3 command (no response data expected)
-	-	-	-	-	-	-	1	ISO case 4 command (response data may be returned)
-	-	-	-	-	X	X	-	RFU

Bits b5 and b4 provide information on the data structure of the command message data field.

- b5 - b4 = 00 indicate that no general information on the data structure is provided; i.e. the data structure is Application dependent;
- b5 - b4 = 01 indicate that the command message data field is coded as one or more DGI structures, according to [Scripting Lang];
- b5 - b4 = 10 indicate that the command message data field is coded as one or more BER-TLV structures, according to ISO 8825.

Bits b7 and b6 provide information on the encryption of the value fields of the data structure present in the command message data field.

- b7 – b6 = 00 indicate that no general information on the data encryption is provided; i.e. the encryption (or non-encryption) of the data is Application dependent, or that the data value fields of all the data structures present in the current command message are not encrypted;
- b7 – b6 = 01 indicate that the encryption (or non-encryption) of the data structure value fields is Application dependent; e.g. when multiple data structures are present in the current command message, some may have encrypted data value fields and other data value fields may be non-encrypted;
- b7 – b6 = 11 indicate that the data value fields of all the data structures present in the current command message are encrypted.

The decryption of application-specific data is the responsibility of the Application.

11.11.2.2 Reference Control Parameter P2 – Block Number

Reference control parameter P2 shall contain the block number coded sequentially from '00' to 'FF'. The Security Domain shall check the sequence of commands and shall reset the block counter when one of the following occurs:

- A personalization session is completed or a new one is starting (see section 11.11.1).
- The Security Domain receives a STORE DATA command indicating P1.b8=1 (last block).
- The Security Domain rejects an erroneous STORE DATA command.
- The Secure Channel session (if any) established by the Security Domain is reset (see conditions triggering Secure Channel Termination, described in section 10.2.3).
- The Security Domain is deselected (i.e. an Application – another or the same – is selected on the same logical channel).
- The Security Domain is selected on the same or another logical channel.
- The card is reset.

11.11.2.3 Data Field Sent in the Command Message

The data field shall contain data in a format expected by the Security Domain or the Application. If the data is intended for an Application, it is sent to the Application as described in section 7.3.2.

Three data structuring modes are defined:

- Application dependent format applies when no information is available on the format of the incoming command data: bits b5-b4 of reference control parameter P1 are set to '00'. In this case, information on the encryption (or non-encryption) of the incoming command data is usually not available (parameter P1 bits b7-b6 set to '00'): the format and eventual encryption of the incoming command data are implicitly known by the Application;
- DGI formatting applies when all data structures that are present in the command data field are formatted as DGI structures (as defined in [Scripting Lang]): bits b5-b4 of reference control parameter P1 are set to '01'. In this case, some information may be available on the encryption (or non-encryption) of the value fields of the DGI data structures: reference control parameter P1 bits b7-b6 are set accordingly;
- BER-TLV formatting applies when all data structures that are present in the command data field are formatted as BER-TLV structures (as defined in ISO 8825): bits b5-b4 of reference control parameter P1 are set to '10'. In this case, some information may be available on the encryption (or non-encryption) of the value fields of the TLV data structures: reference control parameter P1 bits b7-b6 are set accordingly.

If the overall length of the intended command message exceeds 255 bytes, the individual (or group of) data shall be sent in multiple consecutive STORE DATA commands. Whether the data format is a DGI or BER-TLV data structure, the following rules shall apply:

- The data structure length indicators shall always reflect the actual full length of the data structure value field;
- The data structure value field shall be truncated in the STORE DATA command message containing the data structure length indicator (e.g. at the maximum length of the command message);
- The subsequent STORE DATA command shall contain the remainder of the data structure value field (that may be followed by one or more data structure(s) in this same command message) – note: for very large data, more than one subsequent STORE DATA command message may be required for the remainder of the data structure value field;

- The target Application or Security Domain shall use the last data structure length indicator of a STORE DATA command message to determine whether a subsequent STORE DATA command is expected to contain the remainder of the data structure value field.

The Issuer Security Domain shall support at least the following TLV coded data objects:

- Issuer Identification Number (tag '42')
- Card Image Number (tag '45')
- Issuer Security Domain AID (tag '4F')
- Card Data (tag '66')
- Card Capability Information (tag '67')

A Supplementary Security Domain should support the following TLV coded data objects:

- Security Domain Provider Identification Number (tag '42')
- Security Domain Image Number (tag '45')
- Security Domain Management Data (tag '66')
- Security Domain Manager URL (tag '5F50')

When DGI formatting is used, these data objects shall be embedded within template DGI '0070'. Otherwise, these data objects shall be presented in BER-TLV format.

Some data objects are internal data dynamically generated by the Security Domain and cannot be stored by this command. The command shall be rejected if such data objects are presented in DGI '0070' or presented in BER-TLV format. The following is a list (possibly incomplete) of such data objects:

- Tag 'D3': Current Security Level
- Tag '2F00': List of Applications associated with the Security Domain, or every application on the card if the Security Domain has Global Registry Privilege
- Tag 'FF21': Extended Card Resources Information available for Card Content Management, as defined in [TS 102 226]
- Tag 'C2': Confirmation Counter
- Tag 'C1': Sequence Counter of the default Key Version Number

11.11.3 Response Message

11.11.3.1 Data Field Returned in the Response Message

If the P1 parameter indicates that this command is an ISO case 3 command, then:

- If the target Application or Security Domain indicates that response data are available, a response of '6A86' shall be returned.
- The data field of the response message shall not be present.

If the P1 parameter indicates that this command is an ISO case 4 command, then a response data field may be present.

11.11.3.2 Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

This command may return either a general error condition as listed in section 11.1.3 – *General Error Conditions* or one of the following error conditions.

Table 11-90: STORE DATA Error Condition

SW1	SW2	Meaning
'6A'	'80'	Incorrect values in command data
'6A'	'84'	Not enough memory space
'6A'	'88'	Referenced data not found

11.11.4 Key Loading

Using DGI formatting it is possible to load keys using the STORE DATA command.

The encryption and decryption of the DGI's content shall be performed using the Data Encryption session key (DEK session key) and the algorithm supported by the Secure Channel Protocol for sensitive data encryption/decryption.

All encrypted data grouping content defined in this section shall be right-padded with as many arbitrary bytes as needed to reach the required block size (depending on the algorithm used by the DEK session key); e.g. 8 bytes for DES encryption, 16 bytes for AES encryption.

The STORE DATA command shall be coded as a case 3 command.

The Data Group Identifier for the Key Control Reference Template is defined in the following table:

Table 11-91: DGI for Key Information Data

DGI	DGI Length	Data Content	Encrypt
'00B9'	Variable	Key Information Data	No

A key may consist of a number of key components and therefore may require multiple key Control Reference Template (CRT) TLVs to be provided. If the key version, key id, or key usage qualifier provided in a CRT TLV is not consistent with a value already received for the same key then the STORE DATA command shall be rejected.

11.11.4.1 Symmetric Key Scheme

When supporting a symmetric scheme, the Data Grouping Identifiers '00B9' and '8113' shall be used to load/update a secret key.

The key data format is:

1. Most significant byte first
2. When ciphering a key component value additional padding shall be added according to the encryption algorithm used.

The CRT defined in the following table is used to describe the symmetric keys sent in responses/commands from/to the Security Domains.

Table 11-92: Data Content for DGI '00B9' – Symmetric Scheme

Tag	Length	Description	Presence
'B9'	Variable	CRT tag (CT)	Mandatory
'95'	1 or 2	Key Usage Qualifier values according to section 11.1.9	Mandatory
'96'	1	Key Access according to section 11.1.10	Optional
'80'	1	Key Type according to section 11.1.8	Mandatory
'81'	1 or 2	Key Length, in bytes (unsigned integer value)	Mandatory
'82'	1	Key Identifier	Mandatory
'83'	1	Key Version Number	Mandatory
'84'	3	Key check value	Mandatory
'B9'	Variable	CRT tag (CT)	Conditional
…	…	…	…

When the Key Access field is not present, the default Key Access value is '00'.

The Key Check Value shall be coded as described in section B.6.

The decrypted key shall be verified against its associated check value as described in section B.6. If this comparison fails, a response of '6982' shall be returned.

11.11.4.1.1 DGI for a Symmetric Scheme in Secret Key Format

The following Data Grouping Identifier is used to populate a secret key:

Table 11-93: Data Content for DGI '8113'

DGI	Length	Data Content	Encrypt
'8113'	Variable – multiple of 8	Secret Key	Yes

DGI '8113' shall immediately follow DGI '00B9' and shall be repeated once for each key described in DGI '00B9'.

11.11.4.2 Asymmetric Key Schemes

11.11.4.2.1 RSA

When supporting the RSA scheme, either the Data Grouping Identifier's '8112' Private Key Exponent, or the Data Grouping Identifiers '8121' to '8125' RSA Chinese Remainder Theorem (RSACRT) constants shall be used to load/update the private component.

The DGI '0011' is used for the Public Key Exponent. The DGI '0010' is used for the Modulus.

The key data format is:

1. Most significant byte first
2. Fixed length related to the length of the modulus, zero-padding to the left.
3. Exception for the public key exponent which is encoded using the shortest byte representation.

The Control Reference Template is used to describe the keys sent in commands and responses to or from the Security Domains.

11.11.4.2.1.1 DGIs for the RSA Public Key

The data content of the DGI for the Key Control Reference Template for the asymmetric key scheme is defined in the following table.

Table 11-94: Data Content for DGI '00B9' – RSA Public Key

Tag	Length	Description	Presence
'B9'	Variable	CRT tag (CT)	Mandatory
'95'	1 or 2	Key Usage Qualifier values according to section 11.1.9	Mandatory
'96'	1	Key Access according to section 11.1.10	Optional
'80'	1	Key Type = 'A1' Key Modulus	Mandatory
'81'	1 or 2	Key Length, in bytes (unsigned integer value)	Mandatory
'82'	1	Key Identifier	Mandatory
'83'	1	Key Version Number	Mandatory
'B9'	Variable	CRT tag (CT)	Mandatory
'95'	1	Key Usage Qualifier values according to section 11.1.9	Mandatory
'96'	1	Key Access according to section 11.1.10	Optional
'80'	1	Key Type = 'A0' Public Key Exponent	Mandatory
'81'	1 or 2	Key Length, in bytes (unsigned integer value)	Mandatory
'82'	1	Key Identifier	Mandatory
'83'	1	Key Version Number	Mandatory

The following Data Grouping Identifiers shall be used immediately after DGI '00B9' to populate the Key Modulus and Public Key Exponent:

Table 11-95: Data Content for DGIs '0010' and '0011'

DGI	Length	Data Content	Encrypt
'0010'	Variable	Key Modulus	No
'0011'	Variable	Public Key Exponent	No

11.11.4.2.1.2 DGIs for the RSA Private Key, Exponent Format

When the private key exponent format is used to populate asymmetric keys, the following data content shall be included in the DGI '00B9':

Table 11-96: Data Content for DGI '00B9' – RSA Private Key, Exponent Format

Tag	Length	Description	Presence
'B9'	Variable	CRT tag (CT)	Mandatory
'95'	1 or 2	Key Usage Qualifier values according to section 11.1.9	Mandatory
'96'	1	Key Access according to section 11.1.10	Optional
'80'	1	Key Type = 'A1' Key Modulus	Mandatory
'81'	1 or 2	Key Length, in bytes (unsigned integer value)	Mandatory
'82'	1	Key Identifier	Mandatory
'83'	1	Key Version Number	Mandatory
'B9'	Variable	CRT tag (CT)	Mandatory
'95'	1	Key Usage Qualifier values according to section 11.1.9	Mandatory
'96'	1	Key Access according to section 11.1.10	Optional
'80'	1	Key Type = 'A3' Private Key Exponent	Mandatory
'81'	1 or 2	Key Length, in bytes (unsigned integer value)	Mandatory
'82'	1	Key Identifier	Mandatory
'83'	1	Key Version Number	Mandatory

The following Data Grouping Identifiers shall be used immediately after DGI '00B9' to populate the Private Key when the private key exponent format is used:

Table 11-97: Data Content for DGIs '0010' and '8112'

DGI	Length	Data Content	Encrypt
'0010'	Variable	Key Modulus	No
'8112'	Variable	Private Key Exponent	Yes

11.11.4.2.1.3 DGIs for the RSA Private Key, CRT Format

When the Chinese Remainder Theorem (CRT) format is used to populate the RSA asymmetric keys, the following data content shall be included in DGI '00B9':

Table 11-98: Data Content for DGI '00B9' – RSA Private Key, CRT Format

Tag	Length	Description	Presence
'B9'	Variable	CRT tag (CT)	Mandatory
'95'	1 or 2	Key Usage Qualifier values according to section 11.1.9	Mandatory
'96'	1	Key Access according to section 11.1.10	Optional

Tag	Length	Description	Presence
'80'	1	Key Type = 'A6' Private Key q^{-1} mod p	Mandatory
'81'	1 or 2	Key Length, in bytes (unsigned integer value)	Mandatory
'82'	1	Key Identifier	Mandatory
'83'	1	Key Version Number	Mandatory
'B9'	Variable	CRT tag (CT)	Mandatory
'95'	1	Key Usage Qualifier values according to section 11.1.9	Mandatory
'96'	1	Key Access according to section 11.1.10	Optional
'80'	1	Key Type = 'A8' Private Key d mod (q – 1)	Mandatory
'81'	1 or 2	Key Length, in bytes (unsigned integer value)	Mandatory
'82'	1	Key Identifier	Mandatory
'83'	1	Key Version Number	Mandatory
'B9'	Variable	CRT tag (CT)	Mandatory
'95'	1	Key Usage Qualifier values according to section 11.1.9	Mandatory
'96'	1	Key Access according to section 11.1.10	Optional
'80'	1	Key Type = 'A7' Private Key d mod (p – 1)	Mandatory
'81'	1 or 2	Key Length, in bytes (unsigned integer value)	Mandatory
'82'	1	Key Identifier	Mandatory
'83'	1	Key Version Number	Mandatory
'B9'	Variable	CRT tag (CT)	Mandatory
'95'	1	Key Usage Qualifier values according to section 11.1.9	Mandatory
'96'	1	Key Access according to section 11.1.10	Optional
'80'	1	Key Type = 'A5' Private Key prime factor q	Mandatory
'81'	1 or 2	Key Length, in bytes (unsigned integer value)	Mandatory
'82'	1	Key Identifier	Mandatory
'83'	1	Key Version Number	Mandatory
'B9'	Variable	CRT tag (CT)	Mandatory
'95'	1	Key Usage Qualifier values according to section 11.1.9	Mandatory
'96'	1	Key Access according to section 11.1.10	Optional
'80'	1	Key Type = 'A4' Private Key prime factor p	Mandatory
'81'	1 or 2	Key Length, in bytes (unsigned integer value)	Mandatory
'82'	1	Key Identifier	Mandatory
'83'	1	Key Version Number	Mandatory

The following Data Grouping Identifiers shall be used immediately after DGI '00B9' to populate the Private Key when the Chinese Remainder Theorem format is used:

Table 11-99: Data Content for DGIs '8121' through '8125'

DGI	Length	Data Content	Encrypt
'8121'	Variable	RSACRT constant q^{-1} mod p	Yes
'8122'	Variable	RSACRT constant d mod (q – 1)	Yes
'8123'	Variable	RSACRT constant d mod (p – 1)	Yes
'8124'	Variable	RSACRT constant prime factor q	Yes
'8125'	Variable	RSACRT constant prime factor p	Yes

11.11.4.2.2 ECC

When supporting the ECC scheme, DGI '00B9' is used to describe curve parameters and keys sent to a Security Domain. It shall be followed by DGIs providing the values of ECC curve parameters [TR 03111] or keys. Their data format shall be:

1. The public key component and the G component (i.e. key types 'B0' and 'B5') shall be formatted using uncompressed encoding as specified in [TR 03111] section 3.2.1, with most significant byte coming first.
2. Other key components shall be encoded as follows: Fixed length related to the byte length of the key, zero-padding added before the most significant bit as needed. This does not apply to the cofactor of order of generator, which shall be encoded with the smallest number of bytes.

11.11.4.2.2.1 DGIs for the ECC Curve Parameters

The data content of DGI '00B9' (Key Control Reference Template) for ECC curve parameters is defined in the following table:

Table 11-100: Data Content for DGI '00B9' – ECC Curve Parameters

Tag	Length	Description	Presence
'B9'	Variable	CRT tag (CT)	Conditional
'80'	1	Key Type = 'F0' – Key parameter reference	Conditional
'85'	1 or 2	Key parameter reference value (see Table B-2)	Conditional
'B9'	Variable	CRT tag (CT)	Mandatory
'80'	1	Key Type = 'B2' – P (field specification)	Mandatory
'81'	1	Curve parameter length, in bytes (unsigned integer value)	Mandatory
'B9'	Variable	CRT tag (CT)	Mandatory
'80'	1	Key Type = 'B3' – A (first coefficient)	Mandatory
'81'	1	Curve parameter length, in bytes (unsigned integer value)	Mandatory
'B9'	Variable	CRT tag (CT)	Mandatory
'80'	1	Key Type = 'B4' – B (second coefficient)	Mandatory
'81'	1	Curve parameter length, in bytes (unsigned integer value)	Mandatory
'B9'	Variable	CRT tag (CT)	Mandatory
'80'	1	Key Type = 'B5' – G (generator)	Mandatory
'81'	1	Curve parameter length, in bytes (unsigned integer value)	Mandatory
'B9'	Variable	CRT tag (CT)	Mandatory
'80'	1	Key Type = 'B6' – N (order of generator)	Mandatory
'81'	1	Curve parameter length, in bytes (unsigned integer value)	Mandatory
'B9'	Variable	CRT tag (CT)	Optional
'80'	1	Key Type = 'B7' – k (cofactor of order of generator)	Optional
'81'	1	Curve parameter length, in bytes (unsigned integer value)	Optional

When loading ECC curve parameters that are intended to be referenced, the CRT with the Key Parameter Reference shall be present.

When loading ECC curve parameters that are directly related to one key only (see below), the CRT with the Key Parameter Reference shall be absent.

The following DGIs are used to populate the ECC curve parameters and shall immediately follow DGI '00B9':

Table 11-101: Data Grouping Identifiers for ECC Curve Parameters

DGI	Length	Data Content	Presence	Data Content Encrypted
'0030'	Variable	P (field specification)	Mandatory	No
'0031'	Variable	A (first coefficient)	Mandatory	No
'0032'	Variable	B (second coefficient)	Mandatory	No
'0033'	Variable	G (generator)	Mandatory	No
'0034'	Variable	N (order of generator)	Mandatory	No
'0035'	Variable	k (cofactor of order of generator); default value '01'	Optional	No

11.11.4.2.2.2 DGIs for the ECC Public Key

The data content of DGI '00B9' (Key Control Reference Template) for an ECC Public Key is defined in the following table:

Table 11-102: Data Content for DGI '00B9' – ECC Public Key

Tag	Length	Data Element	Presence
'B9'	Variable	CRT tag (CT)	Mandatory
'95'	1 or 2	Key Usage Qualifier values according to section 11.1.9	Mandatory
'96'	1	Key Access according to section 11.1.10	Optional
'80'	1	Key Type = 'B0' – ECC public key	Mandatory
'81'	1 or 2	Key Length in bytes (unsigned integer value)	Mandatory
'82'	1	Key Identifier	Mandatory
'83'	1	Key Version Number	Mandatory
'B9'	Variable	CRT tag (CT)	Conditional
'80'	1	Key Type = 'F0' – Key parameter reference	Conditional
'85'	1 or 2	Key Parameter Reference Value	Conditional

The following Data Grouping Identifier is used to populate an ECC Public Key and shall immediately follow DGI '00B9':

Table 11-103: Data Grouping Identifier for ECC Public Key

DGI	DGI Length	Data Content	Data Content Encrypted
'0036'	Variable	Q (public key)	No

When loading a key that shall use ECC curve parameters already present on the card, the CRT with the Key Parameter Reference shall be present. In case of a local reference, the curve parameters with the given reference found in the SD, or if missing there, found in the closest ascendant SD shall be used.

If the CRT with the Key Parameter Reference is absent, DGI '0036' shall be followed by the DGIs for curve parameters as defined in section 11.11.4.2.2.1. The CRT with the Key Parameter Reference shall also be absent within DGI '00B9' of the Curve Parameters.

11.11.4.2.2.3 DGIs for the ECC Private Key

The data content of DGI '00B9' (Key Control Reference Template) for an ECC Private Key is defined in the following table:

Table 11-104: Data Content for DGI '00B9' – ECC Private Key

Tag	Length	Data Element	Presence
'B9'	Variable	CRT tag (CT)	Mandatory
'95'	1 or 2	Key Usage Qualifier values according to section 11.1.9	Mandatory
'96'	1	Key Access according to section 11.1.10	Optional
'80'	1	Key Type = 'B1' – ECC private key	Mandatory
'81'	1 or 2	Key Length in bytes (unsigned integer value)	Mandatory
'82'	1	Key Identifier	Mandatory
'83'	1	Key Version Number	Mandatory
'B9'	Variable	CRT tag (CT)	Conditional
'80'	1	Key Type = 'F0' – Key parameter reference	Conditional
'85'	1 or 2	Key Parameter Reference Value	Conditional

The following Data Grouping Identifier is used to populate an ECC Private Key and shall immediately follow DGI '00B9':

Table 11-105: Data Grouping Identifier for ECC Private Key

DGI	DGI Length	Data Content	Data Content Encrypted
'8137'	Variable	d (private key)	Yes

When loading a key that shall use ECC curve parameters already present on the card, the CRT with the Key Parameter Reference shall be present. In case of a local reference, the curve parameters with the given reference found in the SD, or if missing there, found in the closest ascendant SD shall be used.

If the CRT with the Key Parameter Reference is absent, DGI '8137' shall be followed by the DGIs for curve parameters as defined in section 11.11.4.2.2.1. The CRT with the Key Parameter Reference shall also be absent within DGI '00B9' of the Curve Parameters

Appendices

A GlobalPlatform API

A.1 GlobalPlatform on a Java Card™

This section contains only the API required for GlobalPlatform 2.2.x Java Cards. Use of the Open Platform 2.0.1' API is still allowed for supporting older versions of Applications but is deprecated. It is defined in version 2.1.1 of this Specification.

The deprecated API and new API both access the same objects where applicable. While this may seem obvious for methods that have the same name across both classes (e.g. `setATRHistBytes()`, `setCardContentState()`, and `getCardContentState()`) it shall also be noted as for example that an application that uses the `update()` method in the new API to change the value of the global PIN will affect the same global PIN of an application that uses the `setPIN()` method in the deprecated API to verify the global PIN.

GlobalPlatform Specific Requirements

In order to ensure the highest level of interoperability of GlobalPlatform implementations, GlobalPlatform also adopts the order defined in section 6.2 of *Java Card™ 2.2.x Virtual Machine Specification*.

Some of the minor modifications to the standard functionality defined in the *Java Card™ 2.1.1 Runtime Environment (JCRE) Specifications* and the *Java Card™ 2.1.1 Application Programming Interface* are no longer relevant with the Java Card™ 2.2.x specifications.

GPRegistryEntry objects shall be implemented as Shareable Interface Objects as defined in section 6.2.4 '*Shareable Interfaces*' of the *Java Card™ 2.2.x Runtime Environment (JCRE) Specification* in order to ensure shared access.

This specification describes mechanisms for a Security Domain to install or delete other Security Domains. This behavior potentially overrides the provisions of the *Java Card™ 2.2.x Runtime Environment (JCRE) Specification* that forbids installation or deletion of an application whose context is already active on the card. Regarding other kinds of applications, the provisions of the JCRE shall apply.

GlobalPlatform Package AID

Each GlobalPlatform package AID will be a concatenation of a RID and a PIX. The AID value of the Java Card Export File for the new GlobalPlatform API (identical for both GlobalPlatform 2.1 and 2.1.1) based on the RID specified in appendix H – *GlobalPlatform Data Values* is 'A00000015100'.

Installation

In section 3.1 – *The Method install* of the *Java Card™ 2.2.x Runtime Environment (JCRE) Specifications*, the parameters passed to the method are defined to be initialization parameters from the contents of the incoming byte array parameter.

This specification expands on this requirement and further defines the content of the Install Parameters. This expansion affects both the implementation of an OPEN and the behavior of a Java Card applet developed for a GlobalPlatform card. It does not affect the definition of the `install` method of the Class Applet of the *Java Card™ 2.2.x Application Programming Interface* specification.

The Install Parameters shall identify the following data, present in the INSTALL [for install] command (see section 11.5.2.3.2 – *Data Field for INSTALL [for install]*):

- The instance AID
- The Privileges

- The Application Specific Parameters[1]

The OPEN is responsible for ensuring that the parameters (`bArray`, `bOffset`, and `bLength`) contain the following information:

The array, `bArray`, shall contain the following consecutive LV coded data:

- Length of the instance AID
- The instance AID
- Length of the Privileges
- The Privileges
- Length of the Application Specific Parameters
- The Application Specific Parameters

The byte, `bOffset`, shall contain an offset within the array pointing to the length of the instance AID.

The byte, `bLength`, shall contain a length indicating the total length of the above-defined data in the array.

The applet is required to utilize the instance AID as a parameter when invoking the `register (byte [] bArray, short bOffset, byte bLength)` method of the Class Applet of the *Java Card™ 2.2.x Application Programming Interface* specification.

T=0 Transmission Protocol

GlobalPlatform cards are intended to be functional in the widest range of environments (i.e. Card Acceptance Devices). Currently the *Java Card™ 2.2.x Runtime Environment (JCRE) Specifications* describe the behavior for case 2 commands (when using the T=0 protocol) in contradiction to EMV 2000. GlobalPlatform mandates that the JCRE shall handle this case of command in accordance with ISO/IEC 7816: An applet receiving a case 2 command builds the response and invokes the appropriate API to output the data. If the data is less than the data expected by the terminal, the OPEN will store the data and output a '6Cxx' response code and wait for the CAD to re-issue the command with the correct length. When the re-issued command is received the JCRE will manage the outputting of the stored data.

Atomicity

Unless otherwise specified all internal persistent objects of the GlobalPlatform API must conform to a transaction in progress.

All operations performed by this API, except the `Application.processData()` method shall be executed atomically. Objects used to enforce the implementation of velocity checking shall not conform to a transaction in progress.

Logical Channels

The following logical channel restrictions apply to Java Card™ 2.2.x (see the *Java Card™ 2.2.x Runtime Environment (JCRE) Specifications* for more detail):

[1] While the APDU command contains Install Parameters representing TLV coded system and Application Specific Parameters, the application only requires knowledge of the Application Specific Parameters i.e. only LV of the TLV coded structure ‘C9’ are present as parameters.

Selection of an Application on a logical channel as defined in section 6.3 – *Command Dispatch* will be unsuccessful if this same Application, or any other Application instantiated from code in the same package from which the Application being selected was instantiated, is currently selected on another logical channel but the application code does not implement the `MultiSelectable` interface. Security Domains shall implement the `MultiSelectable` interface.

Changing context from a Security Domain to an Application as defined in section 7.3.2 – Security Domain Support for Application Personalization will be unsuccessful if this same Application, or any other Application instantiated from code in the same package from which the Application being personalized was instantiated, is currently selected on another logical channel but the application code does not implement the `MultiSelectable` interface.

An Application that has the Card Reset privilege and is intended for a card that supports Supplementary Logical Channels should implement the `MultiSelectable` interface.

GlobalPlatform only defines the assignment of logical channel numbers by the card. Optionally and as defined in Java Card 2.2, a card may also support assignment of logical channel numbers by the terminal.

Cryptographic Algorithms

GlobalPlatform cards supporting RSA cryptography should support key sizes not defined as constants in the Key Builder class. More specifically support for key sizes being a multiple of 4 bytes (32 bits), and being within the allowed key lengths defined by the implementation, should be available.

Level of Trust

The Java Card 2.2.x specifications assume that the RID of the AID of packages, applets and instances will be utilized to ensure a level of trust between these entities. In section 4.2.2 – *AID Usage* of the *Java Card™ 2.2.1 Application Programming Interface* it is defined that the RID of an AID of a component must match the RID of the AID of the package and in the definition of the `register (byte [] bArray, short bOffset, byte bLength)` method of the *Java Card™ 2.2.x Application Programming Interface* specification it is defined that an exception must be thrown if the RID portion of the AID bytes in the `bArray` parameter does not match the RID portion of the Java Card name of the applet.

From a real world implementation point of view, mandating that the RID of the instance AID must be the same as the RID of the component from which it was instantiated, is not practical. GlobalPlatform implementations shall not mandate that there be any link through the AID of an instance to its original package. It does however assume that all applications in the same package share the same level of trust.

Invocation of GlobalPlatform Methods

The Application Programming Interface defined herein is accessible to any Java Card applet developed with the intention of being present on a GlobalPlatform card. One limitation does exist relating to the constructor of the applet and to the `install()` method of the Class Applet of the *Java Card™ 2.2.x Application Programming Interface*. As this specification does not define exactly when the instance of an applet becomes an entry in the card's GlobalPlatform Registry, an applet developer can only assume that this has occurred following the successful completion of the install method. To ensure interoperability, GlobalPlatform API methods that require access to the GlobalPlatform Registry entry of the applet invoking the method, shall not be invoked before registering the applet.

The following is a list of methods that may be invoked from within the constructor or the `install()` method:

- getCardState
- getCVM
- getService

The required behavior of the card in the event that an Application incorrectly invokes a method of the `org.globalplatform.GPSystem` class other than those listed above is undefined. For example, an exception may be thrown and the processing of the `install()` method may be aborted.

Selection

On GlobalPlatform cards, if an error occurs during the processing of the `select()` method or if the `select()` method returns `false` or if the Application cannot be selected because it does not implement the `Multiselectable` interface, the OPEN shall continue searching through the GlobalPlatform Registry for a subsequent full or partial match as defined in section 6.4.2.1.2.

If no Application is selected, the corresponding logical channel remains open with no currently selected Application. Since no Application is currently selected, any subsequent command, other than a MANAGE CHANNEL or SELECT [by name] command, will be rejected. It is expected that the off-card entity will take appropriate action on such an error; e.g. select another Application, close the corresponding logical channel, reset or power off the card.

The Method Summary and Details for the GlobalPlatform Java Card™ API specification is now available in a separate document which may be found on the GlobalPlatform website.

A.2 GlobalPlatform on MULTOS™

The GlobalPlatform on MULTOS™ specification is now available in a separate document which may be found on the GlobalPlatform website.

B Algorithms (Cryptographic and Hashing)

A GlobalPlatform card may support many types of security functions for use by applications. This appendix contains examples of several of the cryptographic algorithms and the hashing method that are possible for GlobalPlatform.

B.1 Data Encryption Standard (DES)

The Data Encryption Standard (DES) is a symmetric cryptographic algorithm that requires the use of the same secret key to encrypt and decrypt data. In its simplest form it uses an 8-byte key to encrypt an 8-byte block of data and the same 8-byte key to decrypt and retrieve the original clear text.

Triple DES uses a compound operation of DES encryption and decryption. Triple DES as used in this specification uses keying option 2 as defined in [ISO 18033-3].

B.1.1 Encryption/Decryption

For encryption two variants are defined.

B.1.1.1 CBC Mode

Triple DES in CBC mode, as defined in [ANSI X9.52] and [ISO 10116], is used with an Initial Chaining Value equal to '00 00 00 00 00 00 00 00'.

B.1.1.2 ECB Mode

Triple DES in ECB mode, as defined in [ANSI X9.52] and [ISO 10116], is used.

B.1.2 MACing

The chaining data encryption methods are defined in [ISO 9797-1].

B.1.2.1 Full Triple DES MAC

The full triple DES MAC is as defined in [ISO 9797-1] as MAC Algorithm 1, with initial transformation 1 and output transformation 1, without truncation, and with Triple DES taking the place of the block cipher.

B.1.2.2 Single DES Plus Final Triple DES MAC

This is also known as the Retail MAC. It is as defined in [ISO 9797-1] as MAC Algorithm 3, with initial transformation 1 and output transformation 3, without truncation, and with DES taking the place of the block cipher.

B.1.3 DES Padding

Unless specified to the contrary, padding prior to performing a DES operation across a block of data is achieved in the following manner:

- Append an '80' to the right of the data block;
- If the resultant data block length is a multiple of 8 bytes, no further padding is required;
- Append binary zeroes to the right of the data block until the data block length is a multiple of 8 bytes.

If the data is being padded with the intention of generating a MAC, the padding is discarded following the DES operation.

B.2 Advanced Encryption Standard (AES)

AES is a symmetric cryptographic algorithm that requires the use of the same secret key to encrypt and decrypt data. In its simplest form it uses a 16-byte key to encrypt a 16-byte block of data and the same 16-byte key to decrypt and retrieve the original clear text.

Two other versions of AES exist that involve 24-byte key and 32-byte key respectively. In these versions, the clear text and the cipher text are still 16-byte long. The different 'flavors' may be referred to as 'AES-128', 'AES-192', and 'AES-256'.

A specification of AES with its different versions may be found in 'Advanced Encryption Standard (AES)' FIPS 197 ([FIPS 197]).

This complies with the FIPS PUB 140-2 ([FIPS 140-2]) Annex A (Symmetric Key Encryption – 1).

B.2.1 Encryption/Decryption

AES in CBC mode is used as specified in NIST 800-38A [800-38A]. The ICV should satisfy the requirements of [800-38A] for unpredictable ICVs.

This complies with [FIPS 140-2] Annex A (Symmetric Key Encryption – 1).

B.2.2 MACing

CMAC as specified in NIST SP 800-38B [800-38B] is used for MAC calculations. The resulting signature is composed of 16 bytes. Note that [800-38B] also specifies the padding to be applied to the input.

This complies with [FIPS 140-2] Annex A (Message authentication – 3).

B.2.3 AES Padding

Unless specified otherwise, padding prior to performing an AES operation across a block of data is achieved in the following manner:

- Append an '80' to the right of the data block;
- If the resultant data block length is a multiple of 16, no further padding is required;
- Append binary zeroes to the right of the data block until the data block length is a multiple of 16.

This padding complies with one of the padding schemes proposed in [800-38A].

B.3 RSA

B.3.1 Scheme 1

The schemes defined in this section shall be used with RSA keys of1024 bits.

B.3.1.1 Signature

The signature scheme is RSASSA-PKCS-v1_5 as defined in PKCS#1 ([PKCS#1]). The hash function is SHA-1

B.3.1.2 Encryption

The encryption scheme is RSAES-PKCS1-v1_5 as defined in PKCS#1 ([PKCS#1]).

B.3.2 Scheme 2

The schemes defined in this section shall be used with RSA keys longer than 1024 bits. It is recommended that the key length is an integer multiple of 32

B.3.2.1 Signature

The following signature scheme shall be used for RSA signatures:

- RSASSA-PSS as defined in [PKCS#1].

The following RSASSA-PSS scheme parameters shall be used for hash algorithm, mask generation function, salt length and trailer field:

- Hash algorithm: SHA-256.
- MGF1 as defined in [PKCS#1] shall be used as mask generation function and the underlying hash function shall be SHA-256.
- PKCS#1 default salt length shall be the octet length of the hash value.
- PKCS#1 defined 'BC' shall be used as trailer field.

B.3.2.2 Encryption

The following RSA encryption scheme shall be used:

- RSAES-OAEP as defined in [PKCS#1].

The following RSA-OAEP scheme parameters shall be used for hash algorithm, mask generation function and (optional) label:

- Hash algorithm: SHA-256.
- MGF1 as defined in [PKCS#1] shall be used as mask generation function and the underlying hash function shall be SHA-256.
- An empty label parameter shall be used.

B.4 Elliptic Curve Cryptography (ECC)

B.4.1 Curve Parameters and Key Lengths

Elliptic Curve Cryptography over prime fields GF(p) shall be used.

Standardized Curve Parameters are recommended to be used. Such Parameters can be found in:

- Digital Signature Standard (DSS) ([FIPS 186-4]), recommended by NIST, or
- Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation [RFC 5639], recommended by BSI.

The following table lists recommended curves for different ECC key lengths.

Table B-1: ECC Key Length and Recommended Curves

ECC Key Length	Curve Specified in [FIPS 186-4]	Curve specified in [RFC 5639]
256	P-256	brainpoolP256r1 brainpoolP256t1
384	P-384	brainpoolP384r1 brainpoolP384t1
512	–	brainpoolP512r1 brainpoolP512t1
521	P-521	–

B.4.2 Preloaded ECC Curve Parameters

A card may have one or multiple sets of preloaded ECC curve parameters. A set of curve parameters is identified by a key parameter reference that is assigned at the time curve parameters are loaded. It is then possible to refer to a set of curve parameters, with a particular key parameter reference, when loading a new public or private ECC key, which avoids transferring curve parameters with each new key.

Key parameter references are coded on 1 or 2 bytes, bit b8 of the first byte indicating a coding on 2 bytes. Two kinds of key parameter references are defined:

- Global key parameter references and associated curve parameters can only be stored by the ISD and, when loading a new key, can be referenced from any Security Domain on the card.
- Local key parameter references and associated curve parameters can be stored by any Security Domain on the card and, when loading a new key, can be referenced from this Security Domain or any Security Domain in its sub-hierarchy.

Table B-2 describes reserved ranges of values for each kind of key parameter reference. A range is reserved for global key parameter references defined by GlobalPlatform (RFU), another range is reserved for proprietary global key parameter references, and another one is reserved for proprietary local key parameter references. In this case, ‘proprietary’ either means that no standard reference has been assigned to it yet by GlobalPlatform, or that a standard set of parameters is loaded, at the discretion of a Security Domain owner, under a local key parameter reference (e.g. because not preloaded by the card issuer).

Table B-2: Key Parameter Reference Values

Curve		Key Parameter Reference Value
P-256	as specified in [FIPS 186-4]	'00'
P-384		'01'
P-521		'02'
brainpoolP256r1	as specified in [RFC 5639]	'03'
brainpoolP256t1		'04'
brainpoolP384r1		'05'
brainpoolP384t1		'06'
brainpoolP512r1		'07'
brainpoolP512t1		'08'
RFU		'09' to '3F' and '80 00' to 'BF FF'
Reserved for proprietary global key parameter references		'40' to '5F' and 'C0 00' to 'DF FF'
Reserved for local key parameter references		'60' to '7F' and 'E0 00' to 'FF FF'

B.4.3 ECDSA

ECDSA (sometimes named EC-DSA) is a signature algorithm specified in ANSI X9.62 [ANSI X9.62] standard.

From an input message M, ECDSA produces a signature (r,s).

ECDSA requires a hash function. Unless defined otherwise, the security strength of the hash function used shall meet or exceed the security strength associated with the order of the key according to [FIPS 186-4].

The following table lists the Hash algorithms that shall be used depending on specific ECC key lengths.

Table B-3: Hash Algorithms for ECDSA

ECC Key Length (in bits)	Hash Algorithm
256-383	SHA-256
384-511	SHA-384
512+	SHA-512

The signature shall be coded in plain format as specified in [TR 03111]; i.e. it is the concatenation of the byte string representation of r and s. Thus the signature will have a fixed length of twice the order length.

The ECDSA signature algorithm requires a random value as input. To protect against attacks, a high quality random number generator is required for the entity generating the signature. Recommendations for appropriate random number generators are given in [TR 02102] and NIST 800-90 [800-90].

B.4.4 ECKA

Elliptic Curve Key Agreement Algorithm (ECKA) shall be performed according to the ElGamal Key Agreement (ECKA-EG) described in [TR 03111] which uses one static and one ephemeral key pair. This scheme is equivalent to the scheme named 'One-pass Diffie-Hellmann, C(1, 1, ECC CDH)' in NIST 800-56A [800-56A] for curves with a cofactor of 1. The recommendations in [800-56A] on the handling of ephemeral keys and of intermediate results (e.g. the shared secret ShS) should be taken into account in an implementation.

The static or ephemeral public key shall be coded in uncompressed format as specified in [TR 03111] section 3.2.1 with most significant byte coming first (hence the value shall start with the coding identifier byte '04').

B.4.5 Key Derivation

The shared secret ShS generated by ECKA-EG is not used directly as a key for cryptographic operations, but as an input to a key derivation process.

A key for calculating a receipt and the key set or the base key of a security domain are derived from an initial secret as defined in [TR 03111] for the 'X9.63 Key Derivation Function'. On request, this key derivation may include additional entropy (a random number DR) generated on the card, which then becomes part of the 'SharedInfo' of the key derivation algorithm.

B.5 Hashing Algorithms

A hash is a one-way digest operation over an arbitrary length of data that returns a fixed length hash value. A hash is not a cryptographic algorithm and it only provides integrity of the data. It does not provide authentication or confidentiality.

B.5.1 Secure Hash Algorithm (SHA-1)

SHA-1 is defined in [ISO 10118-3] and FIPS PUB 180-2 ([FIPS 180-2]).

B.5.2 Secure Hash Algorithm (SHA-256)

SHA-256 is defined in [ISO 10118-3] and FIPS PUB 180-2 ([FIPS 180-2]).

B.5.3 Secure Hash Algorithm (SHA-384)

SHA-384 is defined in [ISO 10118-3] and FIPS PUB 180-2 ([FIPS 180-2]).

B.5.4 Secure Hash Algorithm (SHA-512)

SHA-512 is defined in [ISO 10118-3] and FIPS PUB 180-2 ([FIPS 180-2]).

B.5.5 MULTOS Asymmetric Hash Algorithm

The asymmetric hash algorithm for GlobalPlatform on MULTOS is described in MULTOS document [MAO-DOC-REF-009].

B.6 Key Check Values

When loading a secret key to the card, a key check value shall be calculated as described below.

For a DES key, the key check value is computed by encrypting 8 bytes, each with value '00', with the key to be checked and retaining the 3 highest-order bytes of the encrypted result.

For a AES key, the key check value is computed by encrypting 16 bytes, each with value '01', with the key to be checked and retaining the 3 highest-order bytes of the encrypted result.

C Secure Content Management

This appendix defines the different methods that shall be used to secure the changes and/or modifications to the content of a GlobalPlatform card. At this point in time, these methods are the only ones specified for GlobalPlatform cards. Depending on the implementation of the card, any subset of these methods may be supported.

All Secure Content Management operations utilize creation or verification of a Signature using a key and associated algorithm. The options available are summarized in Table C-1. Although it is possible for a Security Domain to support multiple algorithms, a Security Domain shall process signatures in accordance with the key type and key length of the key loaded for a particular content management operation.

Table C-1: Key Strength Requirements for Signature Schemes

Signature Scheme	Token Verification	Receipt Generation	DAP	Key Length
DES Scheme	DES Key	DES Key	DES Key	128 bits (112 bit key as key value contained on 7 bits of each byte)
AES Scheme	AES Key	AES Key	AES Key	128 bits, 192 bits, or 256 bits
RSA Scheme 1	RSA Public Key	RSA Private Key	RSA Public Key	1024 bits
RSA Scheme 2	RSA Public Key	RSA Private Key	RSA Public Key	> 1024 bits
ECC Scheme	ECC Public Key	ECC Private Key	ECC Public Key	>= 256 bits

It is recommended that RSA keys are a multiple of 32 bits in length above the minimum length.

Table C-2: Key Strength Requirements for Cipher Schemes

Cipher Scheme	Ciphered Load File Data Block	Key Length
DES Scheme	DES Key	128 bits (112 bit key as key value contained on 7 bits of each byte)
AES Scheme	AES Key	128 bits, 192 bits, or 256 bits

C.1 Keys

In order to perform the Card Content management operations described in the subsequent sub-sections different keys are required.

C.1.1 Token and Receipt Keys

If Delegated Management is supported then the Security Domain with Token Verification privilege shall have knowledge of a token verification key and the Security Domain with Receipt Generation privilege shall have knowledge of a receipt generation key.

C.1.1.1 Token Key

If the card supports Delegated Management, the Security Domain with Token Verification privilege shall have an unambiguously identified symmetric key or public asymmetric key to be used to verify a Token. Token verification and the corresponding delegated management operation shall fail if the Token verification key is not present. The list of supported Token Verification keys and their associated algorithms are outlined in Table C-1.

C.1.1.2 Receipt Key

If the card supports Delegated Management, the Security Domain with Receipt Generation privilege shall have an unambiguously identified symmetric key or private asymmetric key to be used to generate Receipts. Receipt generation and the corresponding delegated management operation shall fail if the Receipt generation key is not present. The list of supported Receipt Generation keys and their associated algorithms are outlined in Table C-1.

C.1.2 DAP Verification Keys

If the Security Domain has the DAP Verification privilege, it shall have an unambiguously identified symmetric key or public asymmetric key to be used to verify Load File Data Block signatures. DAP Verification shall fail if the DAP Verification key is not present. The list of supported DAP Verification keys and their associated algorithms are outlined in Table C-1.

C.1.3 Load File Data Block Decryption Keys

If the card supports Ciphered Load File Data Block, the Security Domain with Ciphered Load File Data Block privilege shall have an unambiguously identified symmetric key to be used to decrypt Load File Data Blocks. Decryption of Load File Data Blocks shall fail if the Ciphered Load File Data Block key is not present. The list of supported Ciphered Load File Data Block keys and their associated algorithms are outlined in Table C-2.

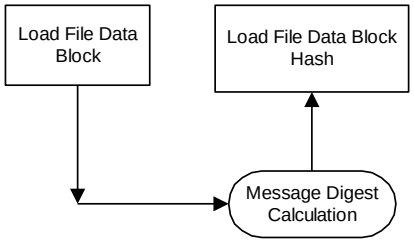
C.2 Load File Data Block Hash (LFDBH)

The Load File Data Block Hash (LFDBH) is a field that is required to be present in the INSTALL [for load] command if Delegated Management loading is occurring, if the Load File contains one or more DAP Blocks, or if the Load File Data Block is encrypted. The purpose of this field is to provide a digest of the Load File Data Block and it is used to ensure that the contents of the Load File Data Block have not been modified in any way.

While the actual hash is not secured in any manner, it is used in other secure operations that ensure that the Load File Data Block has not been modified and a new hash generated for the modified data. The OPEN shall verify the integrity of the Load File Data Block prior to creating an Executable Load File.

The following figure details the calculation of the Load File Data Block Hash.

Figure C-1: Load File Data Block Hash Calculation



Padding of the data is as defined by the hashing algorithm. The digest is generated prior to the Load File Data Block being encapsulated in the TLV structure of the Load File (i.e. excluding tag 'C4' and its length).

Several mechanisms require the presence of a LFDBH: the presence of one or more DAP blocks, the presence of a Load Token or Load File Data Block encryption. For each mechanism, the hash algorithm selected for generating the LFDBH shall comply with the recommendations of NIST SP 800-57-1 [800-57-1] which are recalled in Table C-3.

The LFDBH algorithms supported by the implementation shall be specified in the Card Capability Information (see section 7.4.1.4). The LFDBH algorithm corresponding to the LFDBH present in the INSTALL [for load] command may either be known implicitly or be automatically detected by the OPEN (e.g. based on the length of the LFDBH value). In all cases, the OPEN shall check that this algorithm does fulfill the requirements from all the mechanisms used during a loading operation (see Table C-3).

Table C-3: Hash Selection for LFDBH

Hash Algorithm Used by DAP or Load Token	Algorithm for Load File Data Block Encryption	Acceptable Hash Algorithm(s) for LFDBH
SHA-1	Triple DES with 16 byte key	SHA-1 or SHA-256 or SHA-384 or SHA-512
SHA-256	AES-128	SHA-256 or SHA-384 or SHA-512
SHA-384	AES-192	SHA-384 or SHA-512
SHA-512	AES-256	SHA-512

When Delegated Management is used, the Load Token is a signature of multiple fields including this hash and is proof that the Card Issuer generated the Token for the Load File Data Block linked to the hash.

When DAP Verification is used, a DAP block is a signature of this hash and is proof that the DAP Block was generated by an entity that verified the content of the Load File Data Block linked to the hash.

C.3 Load File Data Block Signature (DAP Verification)

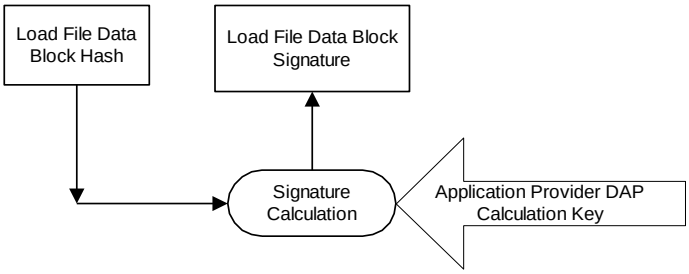
The Load File Data Block Signature provides verification of the Load File Data Block prior to the processing of the actual Load File Data Block. The OPEN shall request the Security Domain linked to the Load File Data Block Signature to verify the signature.

DAP Verification requires the presence of a Security Domain with DAP Verification privilege which is personalized with the appropriate key.

The Load File Data Block Signature is a signature of the Load File Data Block Hash (LFDBH). It is placed in a TLV structured DAP Block along with the AID of the Security Domain responsible for verifying it. DAP Blocks are positioned in the beginning of the Load File.

The following figure details the calculation of the Load File Data Block Signature.

Figure C-2: Load File Data Block Signature Calculation



The following signature algorithms shall be used:

- If the signing key is a DES key, the Load File Data Block Signature is generated according to section B.1.2.2. Prior to signing, padding of the data (LFDBH) is performed as defined in section B.1.3.
- If the signing key is an AES key, the Load File Data Block Signature is generated according to section B.2.2.
- If the signing key is an RSA private key, the Load File Data Block Signature is generated according to section B.3.1.1 or B.3.2.1 depending on the size of the signing key. Notice that the PKCS#1 signature algorithm includes a step where a digest of the input message must be computed. In our context, this digest must not be confused with the input message, the Load File Data Block Hash, which is itself a digest of the Load File Data Block.
- If the signing key is an ECC private key, the Load File Data Block Signature is generated according to section B.4.3.

C.4 Tokens

Tokens are signatures used as proof that an Application Provider has been authorized to perform a Card Content Management operation. Tokens are generated and verified according to this appendix. Tokens may be generated for use on one or multiple cards.

The following signature algorithms shall be used:

- If the signing key is a DES key, the Token is generated according to section B.1.2.2. Prior to signing, padding of the data (LFDBH) is performed as defined in section B.1.3.
- If the signing key is an AES key, the Token is generated according to section B.2.2. The resulting signature is composed of 16 bytes.
- If the signing key is an RSA private key, the Load Token is generated according to section B.3.1.1 or section B.3.2.1 depending on the size of the signing key. Notice that the PKCS#1 signature algorithm includes a step where a digest of the input message must be computed. In our context, this digest must not be confused with the input message, the Load File Data Block Hash, which is itself a digest of the Load File Data Block.
- If the signing key is an ECC private key, the Token is generated according to section B.4.3.

If a command includes a token and the command data without the token exceeds 255 bytes, the token shall be computed across the non-segmented command which is defined as follows:

- Reference control parameters P1 and P2 shall be set as for the last command of the sequence of chained commands.

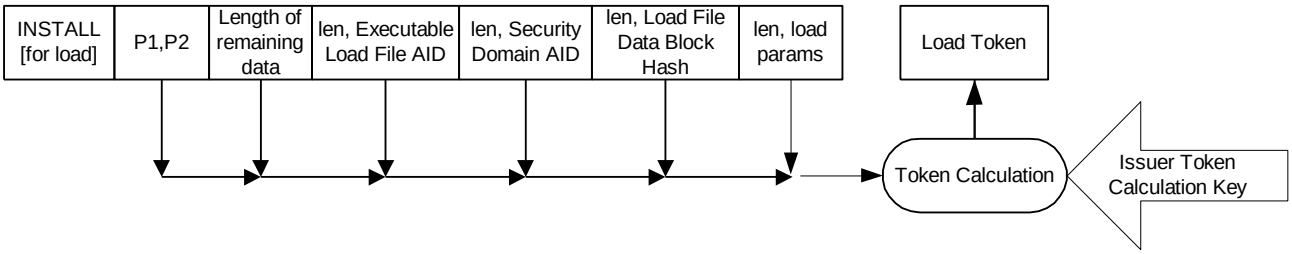
- ‘Length of the following data fields’ shall indicate the length of the data following prior to the token being added. It shall be coded on 3 bytes with values from '00 01 00' to '00 FF FF'.
- This shall be followed by the concatenation of the data fields of all commands of the chained sequence.

C.4.1 Load Token

The Load Token is a signature authorizing the transmission of application code to the card. It allows for the verification of the load request. The OPEN shall request the Security Domain with Token Verification privilege to verify the Load Token.

The following figure shows the Load Token generation.

Figure C-3: Load Token Calculation



Generating a Load Token ensures the following:

- Only the application code (hash of the Load File Data Block) included in this signature may be loaded to the card;
- The AID of the Load File Data Block may only be that included in the signature;
- The Executable Load File (derived from the Load File Data Block) and all Executable Modules within the Executable Load File may only be associated with the Security Domain included in the signature;
- The issuer of the Load Token may only be the Security Domain Provider of the Security Domain with Token Verification Privilege.

The Load Token is a signature of the following data that will be included in the actual INSTALL [for load] command to be transmitted to the card.

Table C-4: Data Elements Included in the Load Token

Name	Length
Reference control parameter P1	1
Reference control parameter P2	1
Length of the following data fields (including the length fields) – see note below	1
Load File AID length	1
Load File AID	5-16
Security Domain AID length	1
Security Domain AID	0 or 5-16
Length of the Load File Data Block Hash	1
Load File Data Block Hash	20
Load parameters field length	1-3
Load parameters field	Variable

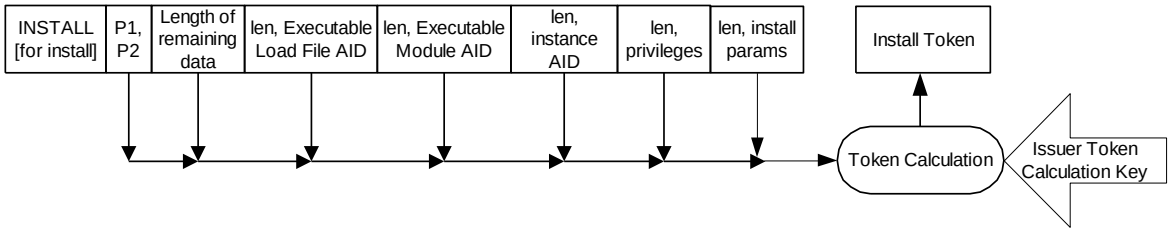
Note that 'Length of the following data fields' is the value of Lc of the INSTALL command, prior to a Token being added. It is coded on 1 byte with a value up to 255.

C.4.2 Install Token

The Install Token allows for the verification of the installation process. The token is a signature authorizing the installation to the card of an Application (Executable Module) contained in a previously loaded file (Executable Load File). The OPEN shall request the Security Domain with Token Verification privilege to verify the Install Token.

The following figure shows the Install Token generation.

Figure C-4: Install Token Calculation



Generating an Install Token ensures the following:

- Only the application (Executable Module) included in this signature and present in the Executable Load File may be installed on the card;
- That only the instance AID included in this signature may be used as the AID to select the instance;
- The Application may only be installed with the privileges included in this signature;
- Only the parameters included in this signature may be used to install the Application;
- The issuer of the Install Token may only be the Security Domain Provider of the Security Domain with Token Verification Privilege.

The Install Token is a signature of the following data:

Table C-5: Input Data for Install Token Computation

Name	Length
Reference control parameter P1	1
Reference control parameter P2	1
Length of the following data fields (including the length fields) – see note below	1
Executable Load File AID length	1
Executable Load File AID	5-16
Executable Module AID length	1
AID within the Executable Load File	5-16
Instance AID length	1
Instance AID	5-16
Privileges length	1
Privileges (byte 1 – byte 2 – byte 3)	1 or 3
Install Parameters field length	1-3
Install Parameters (system and Application) field	Variable

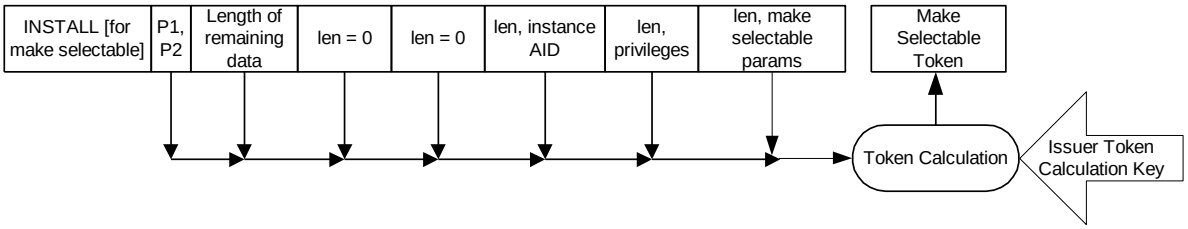
Note that 'Length of the following data fields' is the value of Lc of the INSTALL command excluding the Token and its length. It is coded on 1 byte with a value up to 255.

C.4.3 Make Selectable Token

The Make Selectable Token allows for the verification of the installation for make selectable process. The token is a signature authorizing the making selectable of an Application. The OPEN shall request the Security Domain with Token Verification privilege to verify the Make Selectable Token.

The following figure shows the Make Selectable Token generation.

Figure C-5: Make Selectable Token Calculation



Generating a Make Selectable Token ensures the following:

- Only the Application included in this signature and present in the Executable Load File may be made selectable on the card;

- That only the instance AID included in this signature may be used as the AID to select the instance;
- The Application may only be made selectable with the privileges included in this signature;
- Only the parameters included in this signature may be used to make the Application selectable;
- The issuer of the Make Selectable Token may only be the Security Domain Provider of the Security Domain with Token Verification Privilege.

The Make Selectable Token is a signature of the following:

Table C-6: Input Data for Make Selectable Token Computation

Name	Length
Reference control parameter P1	1
Reference control parameter P2	1
Length of the following data fields (including the length fields) – see note below	1
Executable Load File AID length (= '00')	1
Executable Module AID length (= '00')	1
Instance AID length	1
Instance AID	5-16
Privileges length	1
Privileges (byte 1 – byte 2 – byte 3)	1 or 3
Make Selectable parameters field length	1-3
Make Selectable parameters field	0-n

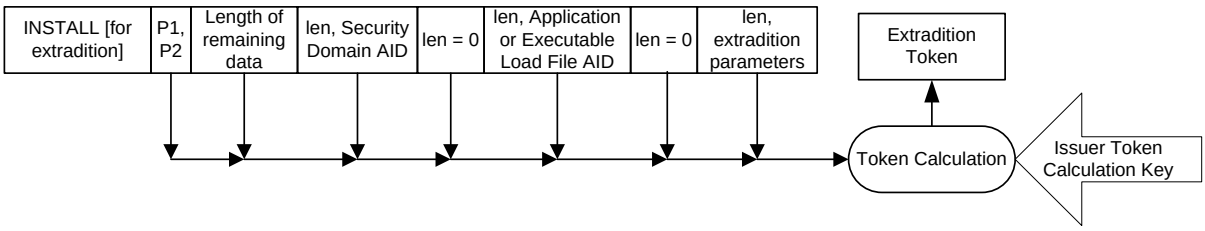
Note that 'Length of the following data fields' is the value of Lc of the INSTALL command excluding the Token and its length. It is coded on 1 byte with a value up to 255.

C.4.4 Extradition Token

The Extradition Token allows for the verification of the extradition process. The token is a signature authorizing the extradition of an Application from one Security Domain to another. The OPEN shall request the Security Domain with Token Verification privilege to verify the Extradition Token.

The following figure shows Extradition Token generation.

Figure C-6: Extradition Token Calculation



Generating an Extradition Token ensures the following:

- Only the Application or Executable Load File included in this signature may be extradited;
- The issuer of the Extradition Token may only be the Security Domain Provider of the Security Domain with Token Verification Privilege.

The Extradition Token is a signature of the following data:

Table C-7: Input Data for Extradition Token Computation

Name	Length	Presence
Reference control parameter P1	1	Mandatory
Reference control parameter P2	1	Mandatory
Length of the following data fields (including the length fields) – see note below	1	Mandatory
Security Domain AID length	1	Mandatory
Security Domain AID	5-16	Conditional
Length = 0	1	Mandatory
Application or Executable Load File AID length	1	Mandatory
Application or Executable Load File AID	5-16	Mandatory
Length = 0	1	Mandatory
Length of Extradition Parameters field	1-3	Mandatory
Extradition Parameters field	0-n	Conditional

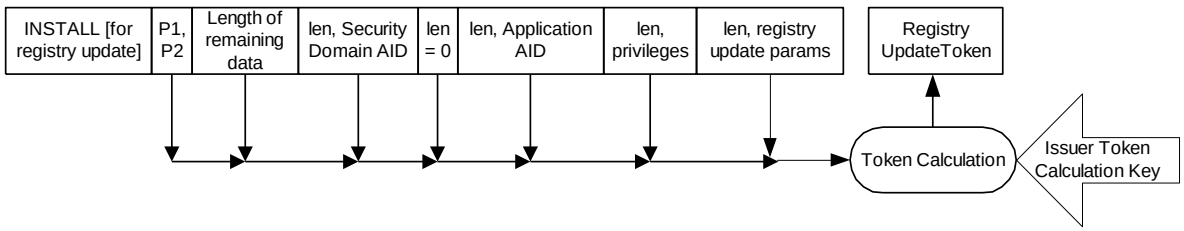
Note that 'Length of the following data fields' is the value of Lc of the INSTALL command excluding the Token and its length. It is coded on 1 byte with a value up to 255.

C.4.5 Registry Update Token

The Registry Update Token allows for the verification of the registry update process. The token is a signature authorizing the updating of the GlobalPlatform Registry data for an Application, and optionally the extradition of an Application from one Security Domain to another. The OPEN shall request the Security Domain with Token Verification privilege to verify the Registry Update Token.

The following figure shows the Registry Update Token generation.

Figure C-7: Registry Update Token Calculation



Generating a Registry Update Token ensures the following:

- Only the Application included in this signature may have its GP Registry entry updated (and be extradited, in the case of extradition using registry update);
- The GP Registry entry of the Application may only be updated with the privileges and/or parameters included in this signature;
- The issuer of the Registry Update Token may only be the Security Domain Provider of the Security Domain with Token Verification Privilege

The Registry Update Token is a signature of the following data:

Table C-8: Input data for Registry Update Token Computation

Name	Length	Presence
Reference control parameter P1	1	Mandatory
Reference control parameter P2	1	Mandatory
Length of the following data fields (including the length fields) – see note below	1	Mandatory
Security Domain AID length	1	Mandatory
Security Domain AID	0 or 5-16	Conditional
Length = 0	1	Mandatory
Application AID length	1	Mandatory
Application AID	5-16	Conditional
Length of Privileges	1	Mandatory
Privileges (byte 1 – byte 2 – byte 3)	0-3	Conditional
Length of Registry Update Parameters	1-3	Mandatory
Registry Update Parameters	0-n	Conditional

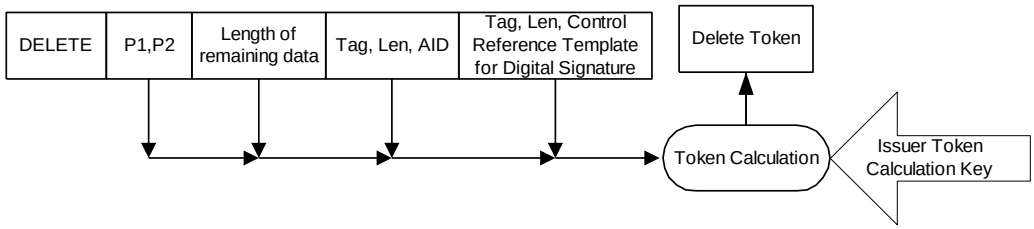
Note that ‘Length of the following data fields’ is the value of Lc of the INSTALL command excluding the Token and its length. It is coded on 1 byte with a value up to 255.

C.4.6 Delete Token

The Delete Token is a signature authorizing the deletion of card content from the card. It allows for the verification of the card content removal process The Token is a signature of selected fields within the DELETE command APDU and is appended to the DELETE command.

The following figure shows the Delete Token generation.

Figure C-8: Delete Token Calculation



Generating a Delete Token ensures the following:

- Only the Application or Executable Load File included in this signature may be deleted;
- The issuer of the Delete Token may only be the Security Domain Provider of the Security Domain with Token Verification Privilege.

The Delete Token is a signature of the following data:

Table C-9: Input Data for Delete Token Computation

Name	Length
Reference control parameter P1	1
Reference control parameter P2	1
Length of the following data fields – see note below	1
Application or Executable Load File AID tag ('4F')	1
AID length	1
Application or Executable Load File AID	5-16
Control Reference Template for Digital Signature tag ('B6')	0 or 1
Control Reference Template for Digital Signature length	0-3
Control Reference Template for Digital Signature (Token) as defined in Table 11-23, comprising (in TLV format) Identification Number of the Security Domain with the Token Verification privilege (tag '42'), Image Number of the Security Domain with the Token Verification privilege (tag '45'), Application Provider id (tag '5F20'), and Token identifier/number (tag '93')	0-n

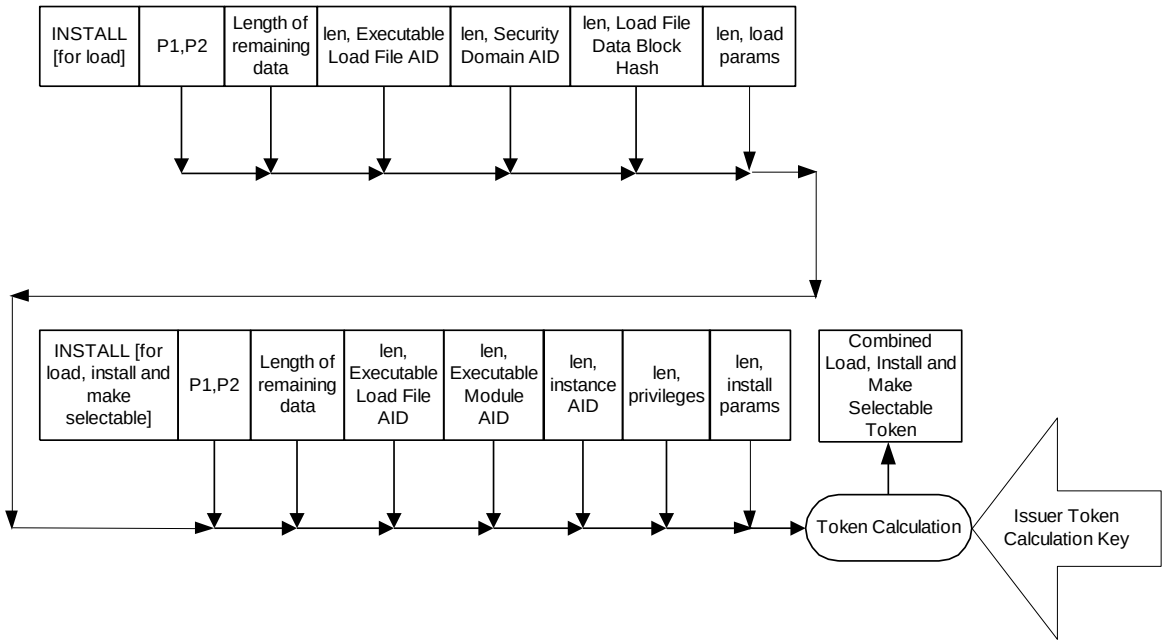
Note that ‘Length of the following data fields’ is the value of Lc of the DELETE command, prior to a Token being added. It is coded on 1 byte with a value up to 255.

C.4.7 Load, Install and Make Selectable Token

The combined Load, Install and Make Selectable Token allows for the verification of the combined loading and installation process. The token is a signature authorizing the combined loading of an Executable Load File and installation to the card of an Application contained in a previously loaded file (Executable Load File). The OPEN shall request the Security Domain with Token Verification privilege to verify the combined Load, Install and Make Selectable Token.

The following figure shows the combined Load, Install and Make Selectable Token generation.

Figure C-9: Load, Install and Make Selectable Token Calculation



Generating a combined Load, Install and Make Selectable Token ensures the following:

- Only the application code included in this signature may be loaded to the card;
- The AID of the Load File Data Block may only be that included in the signature;
- The Executable Load File (derived from the Load File Data Block) and all Executable Modules within the Executable Load File may only be associated with the Security Domain included in the signature;
- Only the application (Executable Module) included in this signature and present in the Executable Load File may be installed on the card;
- That only the instance AID included in this signature may be used as the AID to select the instance;
- The Application may only be installed with the privileges included in this signature;
- Only the parameters included in this signature may be used to install the Application;
- The issuer of the combined Load, Install and Make Selectable Token may only be the Security Domain Provider of the Security Domain with Token Verification Privilege.

The combined Load, Install and Make Selectable Token is a signature of the following data:

Table C-10: Input Data for 'Load, Install and Make Selectable' Token Computation

Name	Length
Reference control parameter P1	1
Reference control parameter P2	1
Length of the following data fields (including the length fields)	1
Load File AID length	1
Load File AID	5-16
Security Domain AID length	1
Security Domain AID	0 or 5-16
Length of the Load File Data Block Hash	1
Load File Data Block Hash	0 or 20
Load parameters field length	1-3
Load parameters field	Variable
Reference control parameter P1	1
Reference control parameter P2	1
Length of the following data fields (including the length fields)	1
Executable Load File AID length	1
Executable Load File AID	0 or 5-16
Executable Module AID length	1
AID within the Executable Load File	0 or 5-16
Instance AID length	1
Instance AID	5-16
Privileges length	1
Privileges (byte 1 – byte 2 – byte 3)	1 or 3
Install Parameters field length	1-3
Install Parameters (system and Application) field	Variable

Note that 'Length of the following data fields' is the value of Lc of the INSTALL command excluding the Token and its length. It is coded on 1 byte with a value up to 255.

C.5 Receipts

Receipts are optional. They are generated by the Security Domain with Receipt Generation privilege, which requires knowledge of the appropriate receipt key. The Receipt is comprised of data appropriate to the function that has been performed, together with Card Unique Data generated by the Security Domain with Receipt Generation privilege. The receipt generation algorithm is determined by the type and length of the receipt generation key.

A Receipt is computed with one of the following methods:

- Symmetric cryptography using a DES receipt key, an ICV of binary zeroes, and the signature method described in section B.1.2.2 – *Single DES Plus Final Triple DES MAC* applied to the receipt data. Prior to generating the signature, the data shall be padded according to section B.1.3
- Asymmetric cryptography using an RSA private key and the signature method described in section B.3.1.1 or B.3.2.1 (depending on the length of the key) applied to the receipt data.
- Symmetric cryptography using an AES receipt key, an ICV of binary zeroes, and the signature method described in section B.2.2 applied to the receipt data.
- Asymmetric cryptography using an ECC private key of and the signature method described in section B.4 applied to the receipt data.

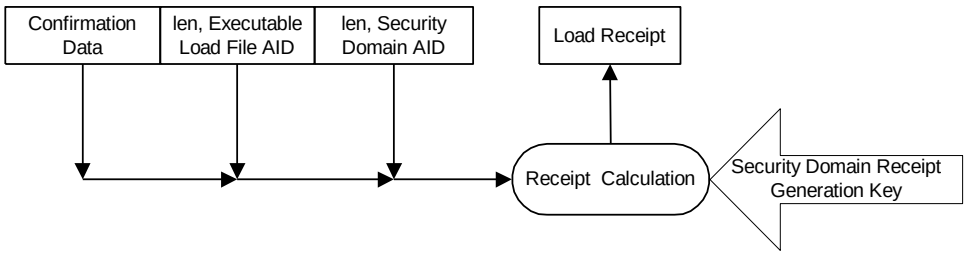
In addition to the appropriate cryptographic key, the Security Domain with Receipt Generation privilege also keeps track of a Confirmation Counter. The Confirmation Counter is a 16-bit value that is incremented by one (1) following each receipt generation. The Confirmation Counter is initialized to zero. When reaching its maximum value, the Confirmation Counter shall not be reset to zero. Cards are not required to support counter values beyond 32767.

C.5.1 Load Receipt

The Load Receipt provides confirmation that a successful load has occurred through the delegated loading process. A Load Receipt may be included in the response message for the last LOAD command of a sequence of LOAD commands.

The following figure details the Load Receipt calculation.

Figure C-10: Load Receipt Calculation



A Load Receipt is a signature of the following data:

Table C-11: Data Elements Included in the Load Receipt

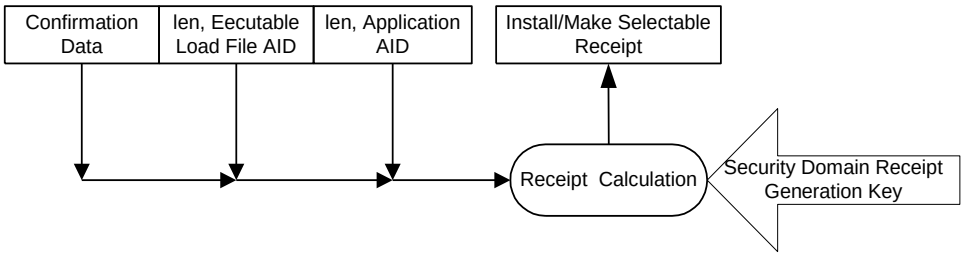
Name	Length
Confirmation Data as defined in section 11.1.6	1-n
Executable Load File AID length	1
Executable Load File AID	5-16
Security Domain AID length	1
Security Domain AID	5-16

C.5.2 Install Receipt and Make Selectable Receipt

The Install/Make Selectable Receipt provides confirmation card that a successful installation/make selectable has occurred through the delegated installation/make selectable process. An Install/Make Selectable Receipt may be returned in the response message to the INSTALL [for install] command, the INSTALL [for make selectable] and the INSTALL [for install and make selectable] command. An Install Receipt is not returned in the response message to the INSTALL [for personalization] command.

The following figure details the Install/Make Selectable Receipt calculation.

Figure C-11: Install/Make Selectable Receipt Calculation



An Install/Make Selectable Receipt is a signature of the following data:

Table C-12: Data Elements Included in the Install/Make Selectable Receipt

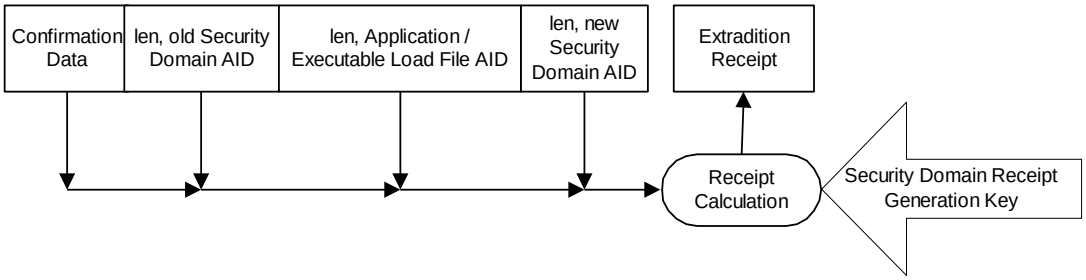
Name	Length
Confirmation Data as defined in section 11.1.6	1-n
Executable Load File AID length	1
Executable Load File AID	5-16
Instance AID length indicator	1
Instance AID	5-16

C.5.3 Extradition Receipt

The Extradition Receipt provides confirmation that the identified old Security Domain extradited the identified Application or Executable Load File to the identified new Security Domain, on a specific card, through the Delegated Management process. An Extradition Receipt may be returned in the response message to the INSTALL [for extradition] command.

The following figure details the Extradition Receipt calculation performed by the Security Domain with Receipt Generation privilege.

Figure C-12: Extradition Receipt Calculation



An Extradition Receipt is a signature of the following data:

Table C-13: Data Elements Included in the Extradition Receipt

Name	Length	Presence
Confirmation Data as defined in section 11.1.6	1-n	Mandatory
AID length indicator	1	Mandatory
Old Security Domain AID	5-16	Mandatory
AID length indicator	1	Mandatory
Application instance or Executable Load File AID	5-16	Mandatory
AID length indicator	1	Mandatory
New Security Domain AID	5-16	Mandatory

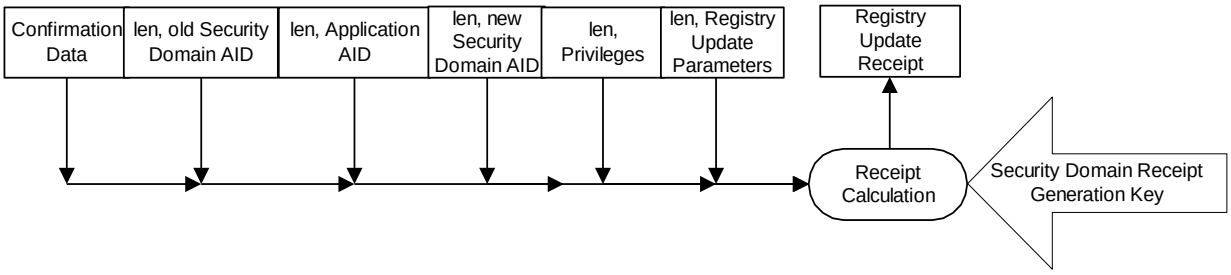
C.5.4 Registry Update Receipt

The Registry Update Receipt provides confirmation that GlobalPlatform Registry data (Privileges and/or Registry Update Parameters) for the identified Application was modified to the specified values on a specific card.

If registry update is used for extradition, the Registry Update Receipt also provides confirmation that the identified old Security Domain extradited the identified Application to the identified new Security Domain, on a specific card, through the Delegated Management process.

The following figure details the Registry Update Receipt calculation performed by the Security Domain with Receipt Generation privilege.

Figure C-13: Registry Update Receipt Calculation



A Registry Update Receipt is a signature of the following data:

Table C-14: Data Elements Included in the Registry Update Receipt

Name	Length	Presence
Confirmation Data as defined in section 11.1.6	1-n	Mandatory
AID length indicator	1	Mandatory
Old Security Domain AID	0 or 5-16	Conditional
AID length indicator	1	Mandatory
Application AID	0 or 5-16	Conditional
AID length indicator	1	Mandatory
New Security Domain AID	0 or 5-16	Conditional
Length of Privileges	1	Mandatory
Privileges (byte 1 – byte 2 – byte 3)	0, 1 or 3	Conditional
Length of Registry Update Parameters	1-3	Mandatory
Registry Update Parameters	0-n	Conditional

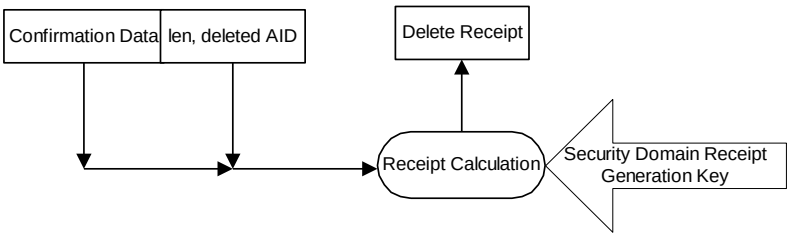
The presence of conditional data depends on what Registry data (Privileges and/or Registry Update Parameters) is being updated and whether registry update is being used for extradition. See sections 11.5.2.3.5 and 11.5.2.3.7 for further details and the format of data. Security Domain AIDs are present for extradition.

C.5.5 Delete Receipt

The Delete Receipt provides proof that the identified Security Domain deleted an Application, an Executable Load File, or an Executable Load File and all its Application instances from a specific card through the delegated deletion process. The Delete Receipt may be returned in the response message to the DELETE [card content] command.

The following figure details the Delete Receipt calculation performed by the Security Domain with Receipt Generation privilege.

Figure C-14: Delete Receipt Calculation



A Delete Receipt is a signature of the following data:

Table C-15: Data Elements Included in the Delete Receipt

Name	Length
Confirmation Data as defined in section 11.1.6	1-n
AID length indicator	1
Application instance or Executable Load File AID	5-16

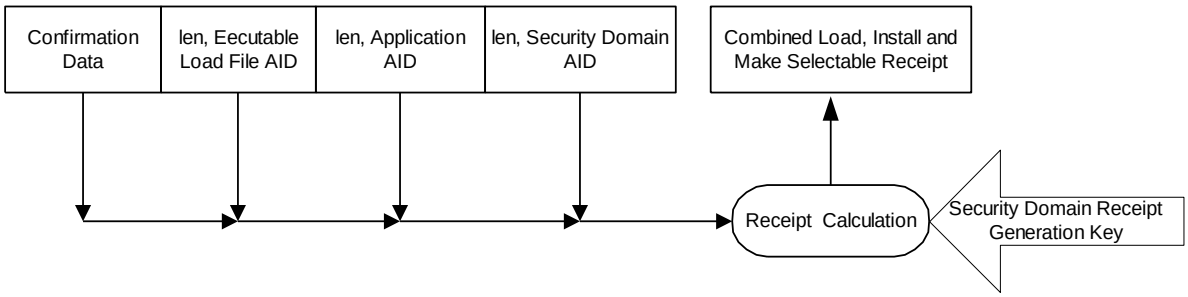
If the function was to delete an Executable Load File and all its Application instances, the receipt returns only the Executable Load File AID.

C.5.6 Combined Load, Install and Make Selectable Receipt

The combined Load, Install and Make Selectable Receipt provides confirmation card that a successful installation has occurred through the delegated installation process. A combined Load, Install and Make Selectable Receipt may be returned in the response message to the INSTALL [for load, install and make selectable] command.

The following figure details the combined Load, Install and Make Selectable Receipt calculation.

Figure C-15: Load, Install and Make Selectable Receipt Calculation



A combined Load, Install and Make Selectable Receipt is a signature of the following data:

Table C-16: Data Elements Included in the Load, Install and Make Selectable Receipt

Name	Length
Confirmation Data as defined in section 11.1.6	1-n
Executable Load File AID length	1
Executable Load File AID	5-16
Instance AID length indicator	1
Instance AID	5-16
Security Domain AID length	1
Security Domain AID	5-16

C.6 Encryption/Decryption of Load File Data Blocks

The Load File Data Block shall be encrypted if the associated Security Domain has the Ciphered Load File Data Block privilege.

The following encryption algorithms shall be used:

- If the encryption key is a DES key, the Load File Data Block shall be encrypted according to section B.1.1.1. Prior to encryption the Load File Data Block shall be padded according to section B.1.3.
- If the encryption key is an AES key, the Load File Data Block shall be encrypted according to section B.2.1. Prior to encryption the Load File Data Block shall be padded according to section B.2.3.
- In both cases, CBC encryption shall be used. If tag 'D3' is missing, a zero ICV shall be used. Else the ICV given in tag 'D3' shall be used which should be a random value. The length of the ICV shall be the block size of the encryption algorithm.

C.7 GlobalPlatform on MULTOS

C.7.1 Keys

In order to perform Card Content management operations a GlobalPlatform on MULTOS smart card requires the following key:

- Issuer's key

The following keys may optionally be supplied to the card during a card content loading operation:

- Application Provider's key
- Application Provider's transport keys

These keys are described in [MAO-DOC-REF-009].

C.7.2 Cryptographic Structures

A GlobalPlatform on MULTOS smart card utilizes the following cryptographic structures:

- Public Key Certificate
- Card Content Management Certificates:
  - Application Load Certificate
  - Application Delete Certificate
- Application Provider's Signature
- Key Transformation Unit

These structures are described in [MAO-DOC-REF-009].

D Secure Channel Protocol '01' (Deprecated)

This Secure Channel Protocol is deprecated, and will be removed from a future version of the Specification. Use of another Secure Channel Protocol, such as Secure Channel Protocol '02' (SCP02), is recommended.

D.1 Secure Communication

D.1.1 SCP01 Secure Channel

The 3 levels of security provided by SCP01 are:

Mutual authentication – in which the card and the off-card entity each prove that they have knowledge of the same secrets;

Integrity and data origin authentication – in which the card ensures that the data being received from the off-card entity actually came from an authenticated off-card entity in the correct sequence and has not been altered;

Confidentiality – in which data being transmitted from the off-card entity to the card is not viewable by an unauthorized entity.

A further level of security applies to sensitive data (e.g. secret keys) that shall always be transmitted as confidential data.

A maximum of 3 Supplementary Logical Channels is supported in SCP01.

Only explicit Secure Channel initiation is supported in SCP01. Both implicit and explicit Secure Channel termination are supported in SCP01.

In SCP01 the card shall support the following implementation options defined by “i” (see Appendix H – *GlobalPlatform Data Values*):

- “i” = '05': Initiation mode explicit, C-MAC on modified APDU, ICV set to zero, no ICV encryption, 3 Secure Channel Keys;
- “i” = '15': Initiation mode explicit, C-MAC on modified APDU, ICV set to zero, ICV encryption, 3 Secure Channel Keys.

Implementation option '15' is an enhancement over implementation option '05' and is therefore recommended.

D.1.2 Mutual Authentication

Mutual authentication is achieved through the process of initiating a Secure Channel and provides assurance to both the card and the off-card entity that they are communicating with an authenticated entity. If any step in the mutual authentication process fails, the process shall be restarted; i.e. new challenges and Secure Channel Session generated.

The Secure Channel is explicitly initiated by the off-card entity using the INITIALIZE UPDATE and EXTERNAL AUTHENTICATE commands defined in this section. The Application may pass the APDU to the Security Domain using the appropriate API e.g. the processSecurity() method of a GlobalPlatform Java Card.

The explicit Secure Channel initiation allows the off-card entity to instruct the card (see section D.4.2 – *EXTERNAL AUTHENTICATE Command*) as to what level of security is required for the current Secure Channel (integrity and/or confidentiality) and apply this level of security to all the subsequent messages exchanged between the card and the off-card entity until the end of the Secure Channel Session. It also gives the off-card entity the possibility of selecting the Key Version Number and Key Identifier to be used (see the INITIALIZE UPDATE command).

Note: The explicit Secure Channel initiation also allows the card to inform the off-card entity what Secure Channel Protocol is supported, using the returned Secure Channel Protocol identifier.

The Secure Channel is always initiated (see section D.4.1 – *INITIALIZE UPDATE Command*) by the off-card entity by passing a ‘host’ challenge (random data unique to this Secure Channel Session) to the card.

The card, on receipt of this challenge, generates its own ‘card’ challenge (again random data unique to this Secure Channel Session).

The card, using the host challenge, the card challenge and its internal static keys, creates new secret Secure Channel session keys and generates a first cryptographic value (card cryptogram) using one of its newly created Secure Channel session keys (see section D.3.1 – *DES Session Keys*).

This card cryptogram along with the card challenge, the Secure Channel Protocol identifier, and other data is transmitted back to the off-card entity.

As the off-card entity should now have all the same information that the card used to generate the card cryptogram, it should be able to generate the same Secure Channel session keys and the same card cryptogram and by performing a comparison, it is able to authenticate the card.

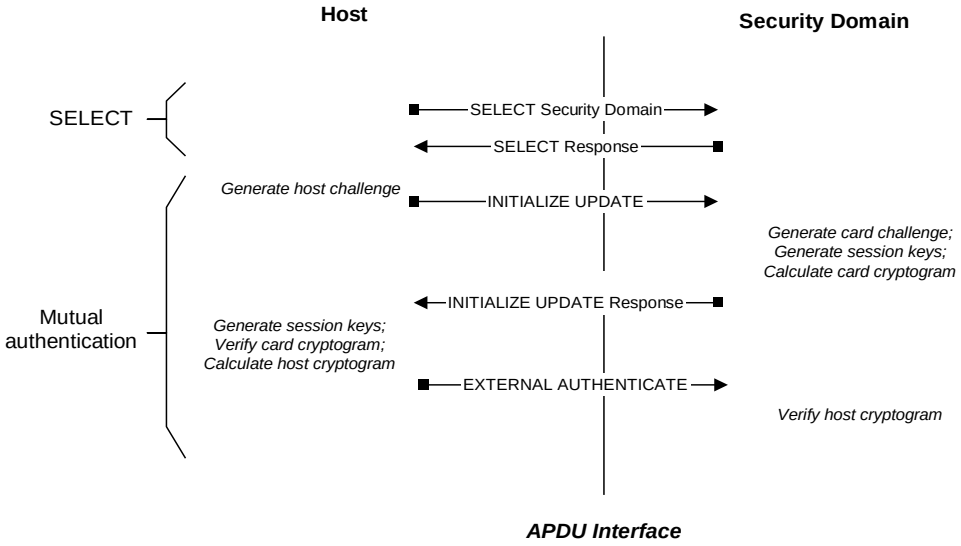
The off-card entity now uses a similar process to create a second cryptographic value (host cryptogram) to be passed back to the card (see section D.4.2 – *EXTERNAL AUTHENTICATE Command*).

As the card has all the same information that the host used to generate the host cryptogram, it should be able to generate the same host cryptogram and, by performing a comparison, it is able to authenticate the off-card entity.

SCP01 Mutual Authentication Flow

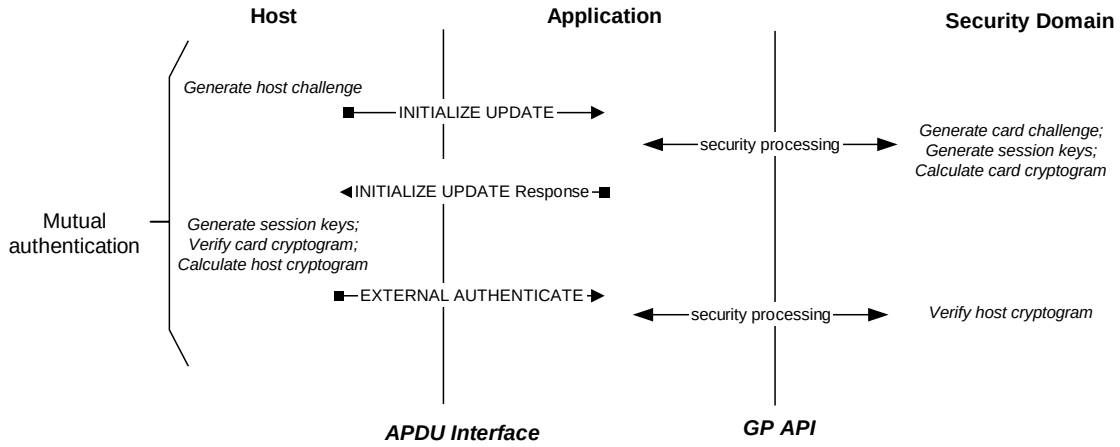
The following flow is an example of mutual authentication between a card and an off-card entity. This flow shows mutual authentication occurring between a Security Domain and an off-card entity.

Figure D-1: Mutual Authentication Flow (Security Domain)



Expanding the authentication process shown in the flow described in section 7.3.1 – Security Domain Support for Secure Messaging, it can be seen how an Application would use the services of an associated Security Domain to achieve mutual authentication.

Figure D-2: Mutual Authentication Flow (using services of Security Domain)



D.1.3 Message Integrity

The C-MAC is generated by applying multiple chained DES operations (using a Secure Channel session key generated during the mutual authentication process) across the header and data field of an APDU command.

The card, on receipt of the message containing a C-MAC, using the same Secure Channel session key, performs the same operation and by comparing its internally generated C-MAC with the C-MAC received from the off-card entity is assured of the integrity of the full command. (If message data confidentiality has also been applied to the message, the C-MAC applies to the message data field before encryption.)

The integrity of the sequence of commands being transmitted to the card is achieved by using the C-MAC from the current command as the (possibly encrypted) Initial Chaining Vector (ICV) for the subsequent command. This ensures the card that all commands in a sequence have been received.

D.1.4 Message Data Confidentiality

The message data field is encrypted by applying multiple chained DES operations (using a Secure Channel session key generated during the mutual authentication process) across the entire data field of the command message to be transmitted to the card, regardless of its contents (clear text data and/or already protected sensitive data).

D.1.5 ICV Encryption

As an enhancement to the C-MAC mechanism, the ICV is encrypted before being applied to the calculation of the next C-MAC. The encryption mechanism used is triple DES with the C-MAC session key. The first ICV of a session, used to generate the C-MAC on the EXTERNAL AUTHENTICATE command, is not encrypted.

D.1.6 Security Level

The Current Security Level of a communication not included in a Secure Channel Session shall be set to NO_SECURITY_LEVEL.

For Secure Channel Protocol '01', the Current Security Level established in a Secure Channel Session is a bitmap combination of the following values: AUTHENTICATED, C_MAC, and C_DECRYPTION. The Current Security Level shall be set as follows:

- NO_SECURITY_LEVEL when a Secure Channel Session is terminated or not yet fully initiated;
- AUTHENTICATED after a successful processing of an EXTERNAL AUTHENTICATE command: AUTHENTICATED shall be cleared once the Secure Channel Session is terminated;
- AUTHENTICATED and C_MAC after a successful processing of an EXTERNAL AUTHENTICATE command with P1 indicating C-MAC (P1 = '01'): AUTHENTICATED and C-MAC shall be cleared once the Secure Channel Session is terminated;
- AUTHENTICATED, C_MAC, and C_DECRYPTION after a successful processing of an EXTERNAL AUTHENTICATE command with P1 indicating C-MAC and command encryption (P1 = '03'): AUTHENTICATED, C_MAC, and C_DECRYPTION shall be cleared once the Secure Channel Session is terminated.

D.1.7 Protocol Rules

In accordance with the general rules described in Chapter 10 – *Secure Communication*, the following protocol rules apply to Secure Channel Protocol '01':

- The successful initiation of a Secure Channel Session shall set the Current Security Level to the security level indicated in the EXTERNAL AUTHENTICATE command: it is at least set to AUTHENTICATED;
- The Current Security Level shall apply to the entire Secure Channel Session unless successfully modified at the request of the Application;
- When the Current Security Level is equal to NO_SECURITY_LEVEL:
    - If the Secure Channel Session was aborted during the same Application Session, the incoming command shall be rejected with a security error;

  - Otherwise no security verification of the incoming command shall be performed. The Application processing the command is responsible to apply its own security rules.
- If a Secure Channel Session is active (i.e. Current Security Level at least set to AUTHENTICATED), the security of the incoming command shall be checked according to the Current Security Level regardless of the command secure messaging indicator:
  - When the security of the command does not match (nor exceeds) the Current Security Level, the command shall be rejected with a security error, the Secure Channel Session aborted and the Current Security Level reset to NO_SECURITY_LEVEL;
  - If a security error is found, the command shall be rejected with a security error, the Secure Channel Session aborted and the Current Security Level reset to NO_SECURITY_LEVEL;
  - In all other cases, the Secure Channel Session shall remain active and the Current Security Level unmodified. The Application is responsible for further processing the command.
- If the Security Domain supports application data encryption and/or decryption, it shall decrypt or encrypt a block of secret data upon request. If the service is not supported or if (one of) the appropriate cryptographic key(s) is not available, the request shall be rejected but the Current Security Level, Session Security Level and Secure Channel Session in operation (if any) shall not be impacted;
- If a Secure Channel Session is aborted, it is still considered not terminated;
- The current Secure Channel Session shall be terminated (if aborted or still open) and the Current Security Level reset to NO_SECURITY_LEVEL on either:
  - Attempt to initiate a new Secure Channel Session (new INITIALIZE UPDATE command);
  - Termination of the Application Session (e.g. new Application selection);
  - Termination of the associated logical channel;
  - Termination of the Card Session (card reset or power off);
  - Explicit termination by the Application (e.g. invoking GlobalPlatform API).

D.2 Cryptographic Keys

Table D-1: Security Domain Secure Channel Keys

Key	Usage	Length	Remark
Secure Channel Encryption Key (S-ENC)	Secure Channel Authentication & Encryption (DES)	16 bytes	Mandatory
Secure Channel Message Authentication Code Key (S-MAC)	Secure Channel MAC Verification (DES)	16 bytes	Mandatory
Data Encryption Key (DEK)	Sensitive Data Encryption and Decryption (DES)	16 bytes	Mandatory

A Security Domain, including the Issuer Security Domain shall have at least one key set containing 3 keys to be used in the initiation and use of a Secure Channel. These keys are all double length DES keys and are the following:

- The Secure Channel encryption key (S-ENC) and the Secure Channel MAC key (S-MAC). These keys are only used to generate Secure Channel session keys during the initiation of a Secure Channel;
- The data encryption key (DEK) for encrypting and decrypting sensitive data; e.g. secret or private keys. This key is a double length DES key and is used as a static key.

D.3 Cryptographic Usage

D.3.1 DES Session Keys

DES session keys shall be generated every time a Secure Channel is initiated and are used in the mutual authentication process. These same session keys may be used for subsequent commands if the Current Security Level indicates that secure messaging is required.

Session keys are generated to ensure that a different set of keys is used for each Secure Channel Session. While this is not required for key encryption operations, it is important for authentication operations, MAC generation and verification and command message encryption and decryption. It is therefore only necessary to create session keys from the static Secure Channel encryption key (S-ENC) and the Secure Channel MAC key (S-MAC).

DES session keys are created using the static Secure Channel encryption key (S-ENC) and the Secure Channel MAC key (S-MAC) and the random host and card challenges. Creating session keys involves 3 steps.

- Generating the session key derivation data. (The same derivation data is used to create both the Secure Channel encryption and Secure Channel MAC session keys.);
- Creating the Secure Channel encryption session key;
- Creating the Secure Channel MAC session key.

The DES operation used to generate these keys is always triple DES in ECB mode.

Figure D-3: Session Key – Step 1 – Generate Derivation Data

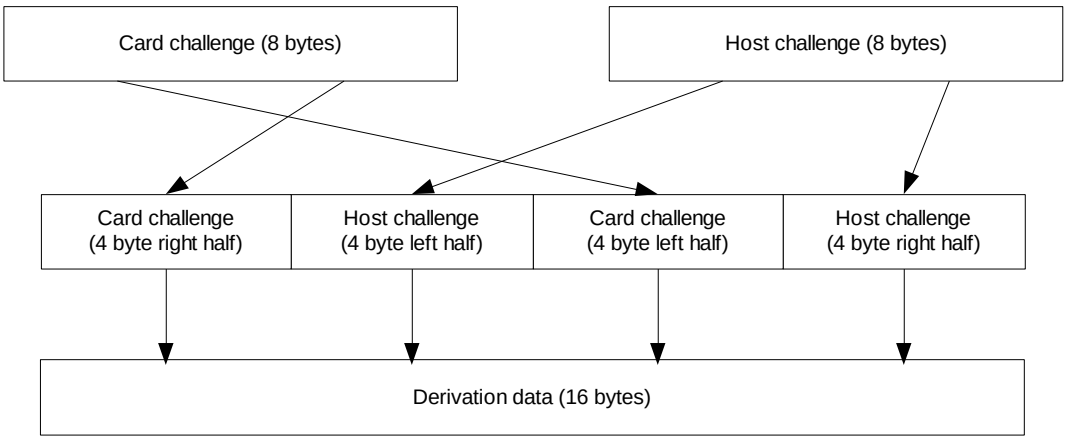


Figure D-4: Session Key – Step 2 – Create S-ENC Session Key

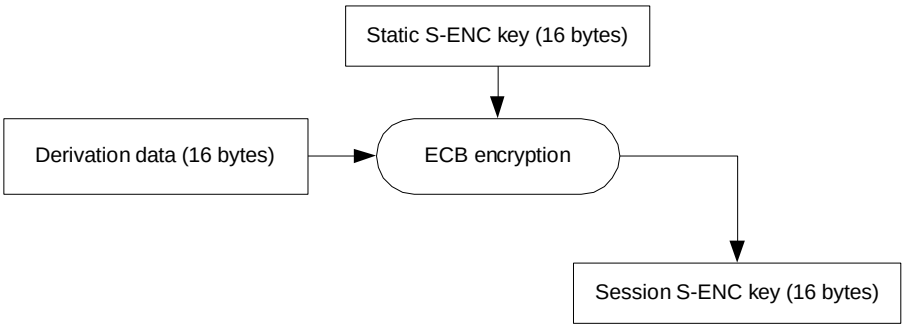
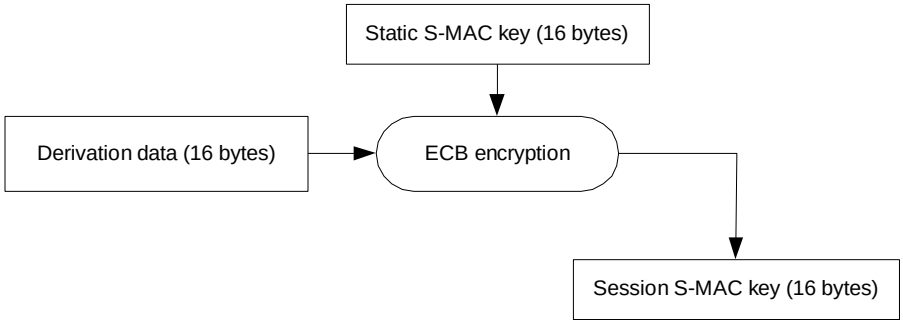


Figure D-5: Session Key – Step 3 – Create S-MAC Session Key



D.3.2 Authentication Cryptograms

Both the card and the off-card entity (Host) generate an authentication cryptogram. The off-card entity verifies the card cryptogram and the card verifies the host cryptogram. Generating or verifying an authentication cryptogram uses the S-ENC session key and the signing method described in section B.1.2.1 – *Full Triple DES*.

D.3.2.1 Card Authentication Cryptogram

The generation and verification of the card cryptogram is performed by concatenating the 8-byte host challenge and 8-byte card challenge resulting in a 16-byte block.

Applying the same padding rules defined in section B.1.3, the data shall be padded with a further 8-byte block ('80 00 00 00 00 00 00 00').

The signature method, using the S-ENC session key and an ICV of binary zeroes, is applied across this 24-byte block and the resulting 8-byte signature is the card cryptogram.

D.3.2.2 Host Authentication Cryptogram

The generation and verification of the host cryptogram is performed by concatenating the 8-byte card challenge and 8-byte host challenge resulting in a 16-byte block.

Applying the same padding rules defined in section B.1.3, the data shall be padded with a further 8-byte block ('80 00 00 00 00 00 00 00').

The signature method, using the S-ENC session key and an ICV of binary zeroes, is applied across this 24-byte block and the resulting 8-byte signature is the host cryptogram.

D.3.3 APDU Command MAC Generation and Verification

The Secure Channel mandates the use of a MAC on the EXTERNAL AUTHENTICATE command. Depending on the Session Security Level defined in the initiation of the Secure Channel, all other commands within the Secure Channel may require secure messaging and as such the use of a C-MAC.

A C-MAC is generated by an off-card entity and applied across the full APDU command being transmitted to the card including the header (5 bytes) and the data field in the command message. (It does not include Le.)

Modification of the APDU command header and padding is required prior to the MAC operation being performed.

The rules for APDU command header modification are as follows:

- The length of the command message (Lc) shall be incremented by 8 to indicate the inclusion of the C-MAC in the data field of the command message;
- The class byte shall be modified to indicate that this APDU command includes secure messaging. This is achieved by setting bit b3 of the class byte. For all the commands defined in this specification, the class byte of commands that contain secure messaging shall be '84'. The C-MAC generation and verification does not reflect the logical channel information included in the class byte of the command: the logical channel number is always assumed to be zero in the generation or verification of the C-MAC;
- The rules for MAC padding are as defined in section B.1.3.

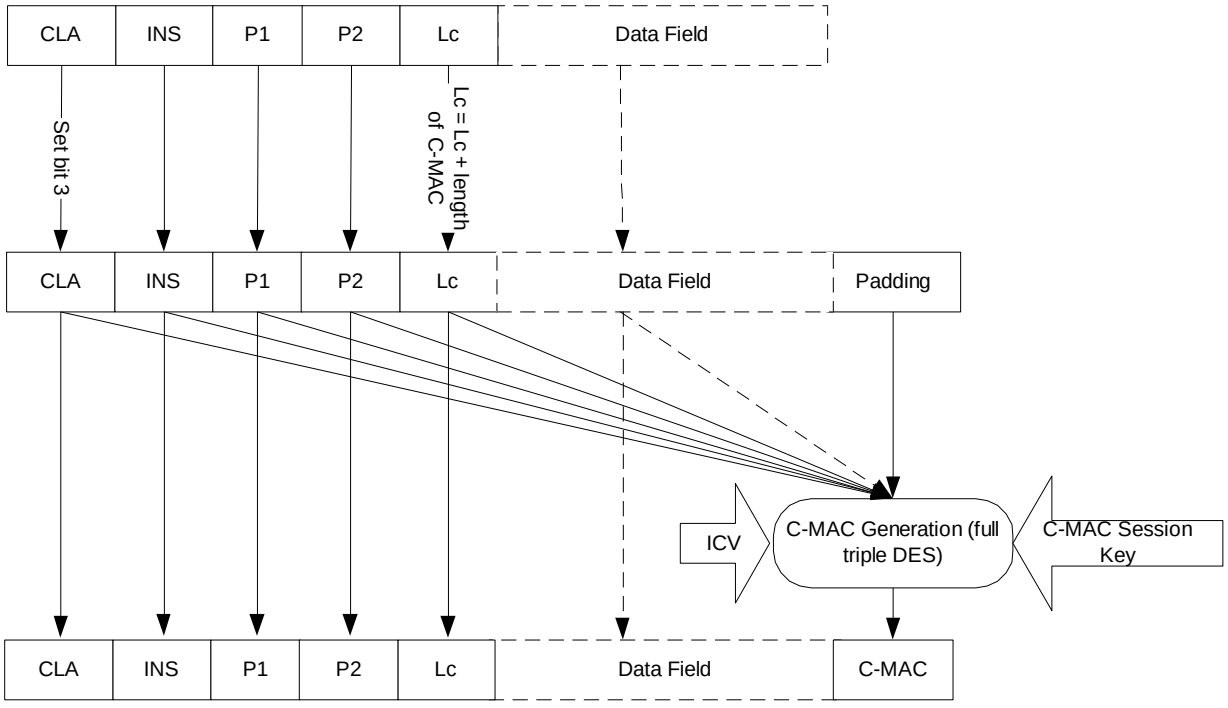
As the ICV is used to chain the commands for command sequence integrity, the value of the ICV depends on which APDU command within the sequence the MAC is being generated for:

- For the EXTERNAL AUTHENTICATE command, the ICV is set to binary zeroes;
- For any command following the EXTERNAL AUTHENTICATE command, the ICV is the C-MAC value successfully verified for the previous command received by the card. (Prior to using the ICV, the ICV can be encrypted as described in section D.1.5 – *ICV Encryption*.)

The signature method defined in section B.1.2.1 – *Full Triple DES*, using the MAC session key and the ICV, is applied across the padded data block and the resulting 8-byte signature is the C-MAC.

The C-MAC is appended at the end of the APDU command message excluding any padding but including the modifications made to the command header (class and Lc).

Figure D-6: APDU Command MAC Generation and Verification



If no other secure messaging is required, the message is now prepared for transmission to the card. The C-MAC shall be retained and used as the ICV for the subsequent C-MAC verification (if any).

The card, in order to verify the C-MAC, shall perform the same padding mechanism to the data and use the same ICV and MAC session key employed by the off-card entity in order to verify the C-MAC. As with the off-card entity, the C-MAC shall be retained and used as the ICV for the subsequent C-MAC verification (if any).

If the Secure Channel Session is occurring on a Supplementary Logical Channel, the logical channel number is added to the class byte of the message after the C-MAC has been generated and prior to the message being transmitted to the card. The card shall ignore any logical channel information in the class byte (i.e. assume the logical channel number is zero) prior to verifying the C-MAC.

D.3.4 APDU Data Field Encryption and Decryption

Depending on the Session Security Level defined in the initiation of the Secure Channel, all subsequent APDU commands within the Secure Channel may require secure messaging and as such the use of a C-MAC (integrity) and encryption (confidentiality).

If confidentiality is required, the off-card entity encrypts the ‘clear text’ data field of the command message being transmitted to the card. (If the APDU command does not originally contain command data, encryption is not performed.) This excludes the header and the C-MAC but includes any data within the data field that has been protected for another purpose e.g. secret or private keys encrypted with the data encryption key (DEK).

Command message encryption and decryption uses the Secure Channel encryption (S-ENC) session key and the full triple DES encryption method described in section B.1.1.1 – *Encryption/Decryption CBC Mode*.

Prior to encrypting the data, the data shall be padded. Unlike the C-MAC, this padding now becomes part of the data field and this necessitates further modification of the Lc value.

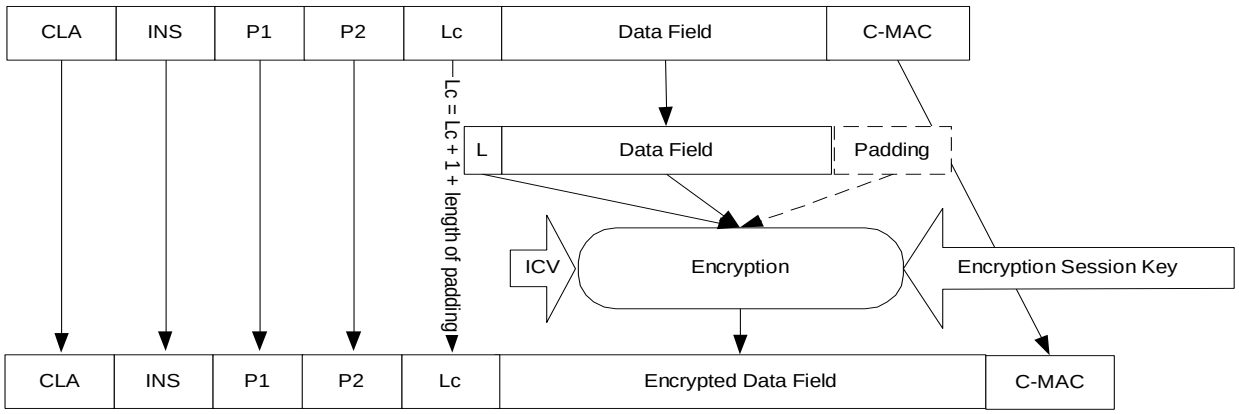
Padding of the data field to be encrypted is performed according to the following rules:

- The length of the original, ‘clear text’, data field is appended to the left of, and becomes part of the command data;
- If the length of the data field is now a multiple of 8, no further padding is required else continue with padding as defined in section B.1.3.

The number of bytes appended to the data field in order to fulfill the above padding shall be added to Lc. This includes the mandatory length byte, the optional '80' and any optional binary zeroes.

The encryption can now be performed across the padded data field using the Secure Channel encryption session key (S-ENC) and the result of each encryption becomes part of the encrypted data field in the command message.

Figure D-7: APDU Data Field Encryption



Note: The ICV and the chaining are only used to link the blocks of the data currently being encrypted. For this reason the ICV for command data encryption is always binary zeroes.

The message is now prepared for transmission to the card.

The card is required to first decrypt the command message and strip off any padding prior to attempting to verify the C-MAC. This decryption uses an ICV of binary zeroes and the same encryption session key employed by the off-card entity. The padding shall be removed and Lc shall be modified to reflect the length prior to encryption; i.e. original clear text data plus C-MAC length.

D.3.5 Key Sensitive Data Encryption and Decryption

Key data encryption is used when transmitting key sensitive data to the card and is over and beyond the security level required for the Secure Channel. For instance all DES keys transmitted to a card should be encrypted.

The key data encryption process uses the static data encryption key and the encryption method described in section B.1.1.2 – *ECB Mode*.

As all DES keys are by their very nature a multiple of 8-byte lengths no padding is required for key encryption operations. Similarly the sensitive data block length shall be constructed as a multiple of 8-byte long block before the data encryption operations: the eventual padding method is application specific.

The encryption is performed across the key sensitive data and the result of each encryption becomes part of the encrypted key data. This encrypted key data becomes part of the 'clear text' data field in the command message.

The on-card decryption of key data is the exact opposite of the above operation; in particular, no padding is removed by the decryption operation.

D.4 Secure Channel APDU Commands

The following table provides the list of minimum security requirements for SCP01 commands.

Table D-2: Minimum Security Requirements for SCP01 Commands

Command	Minimum Security
INITIALIZE UPDATE	None
EXTERNAL AUTHENTICATE	C-MAC

The following table provides the list of SCP01 command support per card Life Cycle State.

Table D-3: SCP01 Command Support per Card Life Cycle State

Command	OP_READY			INITIALIZED			SECURED			CARD_LOCKED	TERMINATED
	AM SD	DM SD	SD	AM SD	DM SD	SD	AM SD	DM SD	SD	SD	SD
INITIALIZE UPDATE	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
EXTERNAL AUTHENTICATE	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	

Legend:

AM SD	Security Domain with Authorized Management privilege.
DM SD	Supplementary Security Domain with Delegated Management privilege.
SD	Supplementary Security Domain without Delegated Management privilege.
SD	Other Security Domain.
✓	Support required.
Blank cell	Support optional.
Striped cell	Support prohibited.

D.4.1 INITIALIZE UPDATE Command

D.4.1.1 Definition and Scope

The INITIALIZE UPDATE command is used to transmit card and Secure Channel Session data between the card and the host. This command initiates the initiation of a Secure Channel Session.

At any time during a current Secure Channel, the INITIALIZE UPDATE command can be issued to the card in order to initiate a new Secure Channel Session.

This INITIALIZE UPDATE command is not to be confused with commands of the same name in legacy applications.

D.4.1.2 Command Message

The INITIALIZE UPDATE command message is coded according to the following table:

Table D-4: INITIALIZE UPDATE Command Message

Code	Value	Meaning
CLA	'80' - '83	See section 11.1.4
INS	'50'	INITIALIZE UPDATE
P1	'xx'	Key Version Number
P2	'xx'	Key Identifier
Lc	'08'	Length of host challenge
Data	'xx xx…'	Host challenge
Le	'00'	

D.4.1.3 Reference Control Parameter P1 – Key Version Number

The Key Version Number defines the Key Version Number within the Security Domain to be used to initiate the Secure Channel Session. If this value is zero, the first available key chosen by the Security Domain will be used.

D.4.1.4 Reference Control Parameter P2 – Key Identifier

The Key Identifier together with the Key Version Number defined in reference control parameter P1 provide a unique reference to the set of keys to be used to initiate the Secure Channel Session.

D.4.1.5 Data Field Sent in the Command Message

The data field of the command message contains 8 bytes of host challenge. This challenge, chosen by the off-card entity, should be unique to this session.

D.4.1.6 Response Message

The data field of the response message shall contain the concatenation without delimiters of the following data elements:

Table D-5: INITIALIZE UPDATE Response Message

Name	Length
Key diversification data	10 bytes
Key information	2 bytes
Card challenge	8 bytes
Card cryptogram	8 bytes

The key diversification data is data typically used by a backend system to derive the card static keys.

The key information includes the Key Version Number and the Secure Channel Protocol identifier, here '01', used in initiating the Secure Channel Session.

The card challenge is an internally generated random number.

The card cryptogram is an authentication cryptogram.

D.4.1.7 Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

This command may return either a general error condition as listed in section 11.1.3 – *General Error Conditions* or the following error condition:

Table D-6: INITIALIZE UPDATE Error Condition

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

D.4.2 EXTERNAL AUTHENTICATE Command

D.4.2.1 Definition and Scope

The EXTERNAL AUTHENTICATE command is used by the card to authenticate the host and to determine the level of security required for all subsequent commands.

A successful execution of the INITIALIZE UPDATE command shall precede this command.

D.4.2.2 Command Message

The EXTERNAL AUTHENTICATE command message is coded according to the following table:

Table D-7: EXTERNAL AUTHENTICATE Command Message

Code	Value	Meaning
CLA	'84' - '87'	See section 11.1.4
INS	'82'	EXTERNAL AUTHENTICATE
P1	'xx'	Security level
P2	'00'	Reference control parameter P2
Lc	'10'	Length of host cryptogram and MAC
Data	'xx xx…'	Host cryptogram and MAC
Le		Not present

D.4.2.3 Reference Control Parameter P1 – Security Level

The reference control parameter P1 defines the level of security for all secure messaging commands following this EXTERNAL AUTHENTICATE command (it does not apply to this command) and within this Secure Channel Session.

Table D-8: EXTERNAL AUTHENTICATE Reference Control Parameter P1

b8	b7	b6	b5	b4	b3	b2	b1	Description
0	0	0	0	0	0	1	1	C-DECRYPTION and C-MAC.
0	0	0	0	0	0	0	1	C-MAC
0	0	0	0	0	0	0	0	No secure messaging expected.

D.4.2.4 Reference Control Parameter P2

The reference control parameter P2 shall always be set to '00'.

D.4.2.5 Data Field Sent in the Command Message

The data field of the command message contains the host cryptogram and the APDU command MAC.

D.4.2.6 Data Field Returned in the Response Message

The data field of the response message is not present.

D.4.2.7 Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

This command may return either a general error condition as listed in section 11.1.3 – *General Error Conditions* or the following error condition:

Table D-9: EXTERNAL AUTHENTICATE Error Condition

SW1	SW2	Meaning
'63'	'00'	Authentication of host cryptogram failed

E Secure Channel Protocol '02'

E.1 Secure Communication

Secure Channel Protocol '02' (SCP02) supports up to 19 Supplementary Logical Channels.

E.1.1 SCP02 Secure Channel

Either the card or the off-card entity may play the role of being the secure sending and receiving entities. SCP02 provides the three followings levels of security:

Entity authentication – in which the card authenticates the off-card entity and the off-card entity may authenticate the card, proving that the off-card entity has knowledge of the same secret(s) as the card;

Integrity and Data origin authentication – in which the receiving entity (the card or off-card entity) ensures that the data being received actually came from an authenticated sending entity (respectively the off-card entity or card) in the correct sequence and has not been altered;

Confidentiality – in which data being transmitted from the sending entity (the off-card entity or card) to the receiving entity (respectively the card or off-card entity) is not viewable by an unauthorized entity.

A further level of security applies to sensitive data (e.g. secret keys) that shall always be transmitted as confidential data.

In SCP02 the “i” parameter (as defined in Appendix H – *GlobalPlatform Data Values*) shall be formed as a bitmap on one byte as follows:

Table E-1: Values of Parameter “i”

b8	b7	b6	b5	b4	b3	b2	b1	Description
Reserved							1	3 Secure Channel Keys
Reserved							0	1 Secure Channel base key
Reserved						1		C-MAC on unmodified APDU
Reserved						0		C-MAC on modified APDU
Reserved					1			Initiation mode explicit
Reserved					0			Initiation mode implicit
Reserved				1				ICV set to MAC over AID
Reserved				0				ICV set to zero
Reserved			1					ICV encryption for C-MAC session
Reserved			0					No ICV encryption
Reserved		1						R-MAC support
Reserved		0						No R-MAC support
Reserved	1							Well-known pseudo-random algorithm (card challenge)
Reserved	0							Unspecified card challenge generation method

Note: “i” is a subidentifier within an object identifier, and bit b8 is reserved for use in the structure of the object identifier according to [ISO 8825-1].

The Security Domain shall support at least one of the following implementation options as defined by “i”:

- “i” = '15': Initiation mode explicit, C-MAC on modified APDU, ICV set to zero, ICV encryption for C-MAC session, 3 Secure Channel Keys, unspecified card challenge generation method, no R-MAC;
- “i” = '1A': Initiation mode implicit, C-MAC on unmodified APDU, ICV set to MAC over AID, ICV encryption for C-MAC session, 1 Secure Channel base key. no R-MAC;
- “i” = '55': Initiation mode explicit, C-MAC on modified APDU, ICV set to zero, ICV encryption for C-MAC session, 3 Secure Channel Keys, well-known pseudo-random algorithm (card challenge), no R-MAC.

E.1.2 Entity Authentication

Off-card entity authentication is achieved through the process of initiating a Secure Channel and provides assurance to the card that it is communicating with an authenticated off-card entity. If any step in the off-card authentication process fails, the process shall be restarted (i.e. new session keys generated).

The Secure Channel initiation and off-card entity authentication implies the creation of session keys derived from card static key(s) using a Secure Channel Sequence Counter maintained by the Security Domain. The Security Domain shall manage one Sequence Counter per Secure Channel base key or Secure Channel keys sharing the same Key Version Number (see section E.2 – *Cryptographic Keys*).

The Sequence Counter is incremented by 1 when and only when the first C-MAC of a secure channel (started implicitly or explicitly) is verified as valid. It is incremented before the processing specific for the command. The Sequence Counter is reset to zero on creation or update of the Secure Channel key(s). When the Secure Channel key set contains more than one key (see section E.2 – *Cryptographic Keys*), the Sequence Counter is reset to zero on the creation or update of any one of the keys of this key set.

Note: the Sequence Counter is not updated for a C-MAC that is not the first of a Secure Channel Session or when a Secure Channel Session is started and the C-MAC is invalid. In this respect, the Sequence Counter keeps track of the number of valid Secure Channel Sessions the corresponding secure messaging key set has experienced so far. When reaching its maximum value, the Sequence Counter shall not be reset to zero. Cards are not required to support counter values beyond 32767.

E.1.2.1 Explicit Secure Channel Initiation

The Secure Channel may be explicitly initiated by the off-card entity using the INITIALIZE UPDATE and EXTERNAL AUTHENTICATE commands. The Application may pass the APDU to the Security Domain using the appropriate API; e.g. the processSecurity() method of a GlobalPlatform Java Card.

The explicit Secure Channel initiation allows the off-card entity to instruct the card (see section E.5.2 – *EXTERNAL AUTHENTICATE Command*) as to what level of security is required for the current Secure Channel (integrity and/or confidentiality) and apply this level of security to all the subsequent messages exchanged between the card and the off-card entity until the end of the session. It also gives the off-card entity the possibility of selecting the Key Version Number to be used (see section E.5.1 – *INITIALIZE UPDATE Command*).

Note: The explicit Secure Channel Session initiation also allows the card to inform the off-card entity what Secure Channel Protocol is supported, using the returned Secure Channel Protocol identifier.

The Secure Channel is always initiated (see section E.5.1 – *INITIALIZE UPDATE Command*) by the off-card entity by passing a ‘host’ challenge (random data unique to this session) to the card.

The card, on receipt of this challenge, generates its own ‘card’ challenge (again random data unique to this session).

The card, using its internal Sequence Counter and static keys, creates new secret session keys and generates a first cryptographic value (card cryptogram) using one of its newly created session keys (see section E.4.1 – *DES Session Keys*).

This card cryptogram along with the Sequence Counter, the card challenge, the Secure Channel Protocol identifier, and other data is transmitted back to the off-card entity.

As the off-card entity should now have all the same information that the card used to generate the card cryptogram, it should be able to generate the same session keys and the same card cryptogram and by performing a comparison, it is able to authenticate the card.

The off-card entity now uses a similar process to create a second cryptographic value (host cryptogram) to be passed back to the card (see section E.5.2 – *EXTERNAL AUTHENTICATE Command*).

As the card has all the same information that the host used to generate the host cryptogram, it should be able to generate the same cryptogram and, by performing a comparison, it is able to authenticate the off-card entity.

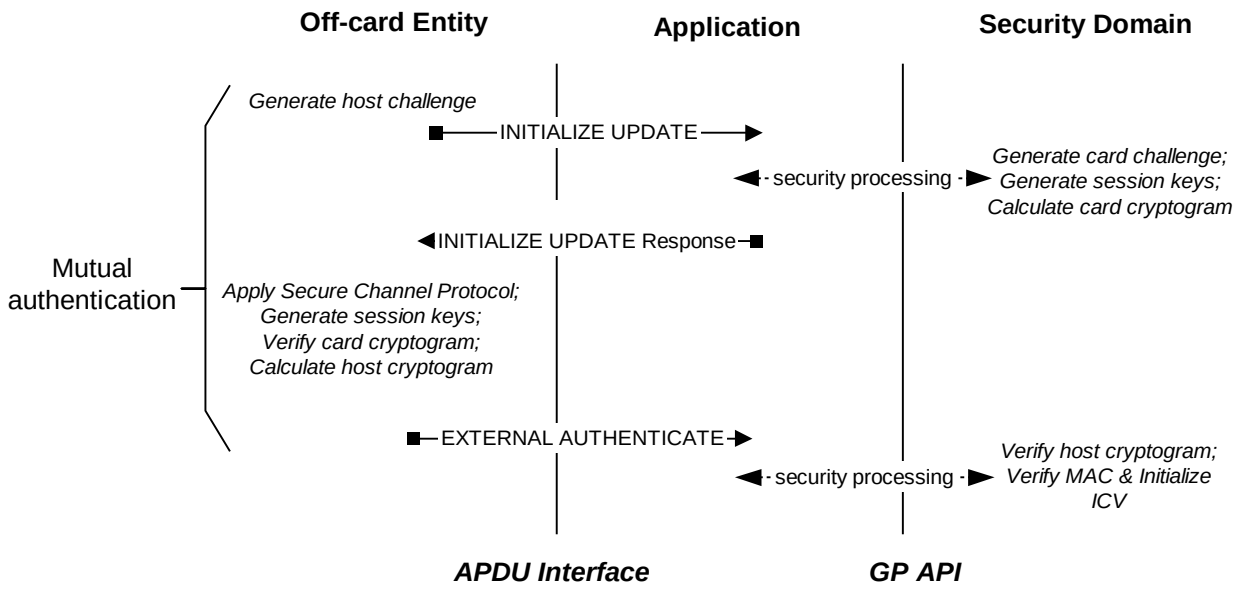
The off-card entity also creates a MAC to be passed back to the card and verified by the card. The verified MAC is used by the card to create the Initial Chaining Vector for the verification of the subsequent C-MAC and/or R-MAC.

When the off-card entity is successfully authenticated, the card increments its internal Secure Channel Sequence Counter.

Explicit Secure Channel Initiation Flow

The following flow is an example of explicit Secure Channel initiation between a card and an off-card entity. Expanding the authentication process shown in the flow described in section 7.3.1 – Security Domain Support for Secure Messaging, it can be seen how an Application would use the services of a Security Domain to achieve the explicit Secure Channel initiation.

Figure E-1: Explicit Secure Channel Initiation Flow



E.1.2.2 Implicit Secure Channel Initiation

The Secure Channel is implicitly initiated when receiving the first APDU command that contains a cryptographic protection (C-MAC). The required level of security is implicitly known by both the card and off-card entity as command message integrity only. The off-card entity implicitly knows which keys are to be used and the current value of the Secure Channel Sequence Counter, or may have previously retrieved the corresponding information using a GET DATA command.

The off-card entity, using the information it knows about the card static keys and its Secure Channel Sequence Counter, creates new secret session keys and generates a C-MAC on an APDU command message.

The Secure Channel is initiated by the off-card entity by appending a C-MAC to an APDU command.

The card, on receipt of this first C-MAC, creates the C-MAC session key using its internal card static key(s) and Secure Channel Sequence Counter.

The card has the same information that the host used to generate the C-MAC. It generates the same MAC and, by performing a comparison, it authenticates the off-card entity.

When the off-card entity is successfully authenticated, the card increments its internal Secure Channel Sequence Counter.

For sensitive data decryption, the card creates the data encryption session key using its internal card static key(s) and the newly incremented value of the Secure Channel Sequence Counter. For R-MAC generation, the card creates the R-MAC session key using its internal card static key(s) and the newly incremented value of the Secure Channel Sequence Counter.

Note: In addition to command message integrity, the off-card entity may, at any moment during the Secure Channel Session, initiate response message integrity using the BEGIN R-MAC SESSION command.

E.1.3 Message Integrity

The MAC is generated by applying multiple chained DES operations (using a session key generated prior to or when opening the Secure Channel) across an APDU message.

A MAC may be generated on the following:

- C-MAC for APDU command messages (generated by the off-card entity);
- R-MAC for APDU response messages (generated by the card).

The receiving entity, on receipt of the message containing a MAC, using the same session key, performs the same operation and by comparing its generated MAC with the MAC received from the sending entity is assured of the integrity of the full command or response. (If message data confidentiality has also been applied to the message, the MAC applies to the message data field before encryption.)

The integrity of the sequence of APDU command or response messages being transmitted to the receiving entity is achieved by using the MAC from the current command or response as the (possibly encrypted) Initial Chaining Vector (ICV) for the subsequent command or response. This ensures the receiving entity that all messages in a sequence have been received. Computing the ICV is detailed in section E.3 – *Cryptographic Algorithms*.

E.1.4 Message Data Confidentiality

The message data field is encrypted by applying multiple chained DES operations (using a session key generated during the Secure Channel initiation process) across the entire data field of either a command message or a response, regardless of its contents (clear text data and/or already protected sensitive data).

E.1.5 Security Level

The Current Security Level of a communication not included in a Secure Channel Session shall be set to NO_SECURITY_LEVEL.

For Secure Channel Protocol '02' with explicit initiation mode, the Current Security Level established in a Secure Channel Session is a bitmap combination of the following values: AUTHENTICATED, C_MAC, R_MAC, and C_DECRYPTION. The Current Security Level shall be set as follows:

- NO_SECURITY_LEVEL when a Secure Channel Session is terminated or not yet fully initiated;
- AUTHENTICATED after a successful processing of an EXTERNAL AUTHENTICATE command: AUTHENTICATED shall be cleared once the Secure Channel Session is terminated;
- C_MAC after a successful processing of an EXTERNAL AUTHENTICATE command with P1 indicating C-MAC (P1 = 'x1' or 'x3'): C-MAC shall be cleared once the Secure Channel Session is terminated. Note that C_MAC is always combined with AUTHENTICATED and simultaneously set and cleared;
- C_DECRYPTION after a successful processing of an EXTERNAL AUTHENTICATE command with P1 indicating Command Encryption (P1 = 'x3'): C_DECRYPTION shall be cleared once the Secure Channel Session is terminated. Note that C_DECRYPTION is always combined with AUTHENTICATED and C_MAC and simultaneously set and cleared;
- R_MAC after a successful processing of an EXTERNAL AUTHENTICATE command with P1 indicating R-MAC (P1 = '1x'): R-MAC shall be cleared once the Secure Channel Session is terminated. Note that in this case R_MAC is always combined with AUTHENTICATED and simultaneously set and cleared. R_MAC may also be combined with C_MAC or C_DECRYPTION (according to the P1 value of the EXTERNAL AUTHENTICATE command) and simultaneously set and cleared;
- R_MAC after a successful processing of a BEGIN R-MAC SESSION command: R-MAC shall be cleared after a successful processing of an END R-MAC SESSION command. Note that in this case R_MAC may be combined with AUTHENTICATED, C_MAC, or C_DECRYPTION depending on the pre-existing Current Security Level of the Secure Channel Session. R_MAC is set and cleared independently of AUTHENTICATED, C_MAC, or C_DECRYPTION.

For Secure Channel Protocol '02' with implicit initiation mode, the Current Security Level established in a Secure Channel Session is a bitmap combination of the following values: AUTHENTICATED, C_MAC, and R_MAC. The Current Security Level shall be set as follows:

- NO_SECURITY_LEVEL when a Secure Channel Session is terminated or not yet initiated;
- AUTHENTICATED and C_MAC after a successful verification of the C- MAC of the first command message with a C-MAC. The Session Security Level shall be set to AUTHENTICATED only. C_MAC shall be cleared after the reception of the first command message without a C-MAC. AUTHENTICATED and C_MAC shall be cleared once the Secure Channel Session is terminated;
- R_MAC after a successful processing of a BEGIN R-MAC SESSION command: R-MAC shall be cleared after a successful processing of an END R-MAC SESSION command.

E.1.6 Protocol Rules

In accordance with the general rules described in Chapter 10 – *Secure Communication*, the following protocol rules apply to Secure Channel Protocol '02' with explicit initiation mode:

- The successful explicit initiation of a Secure Channel Session shall set the Current Security Level to the security level value indicated in the EXTERNAL AUTHENTICATE command: it is at least set to AUTHENTICATED;

- The Current Security Level shall apply to the entire Secure Channel Session unless successfully modified at the request of the Application;
- When the Current Security Level is not set to AUTHENTICATED (i.e. equal to NO_SECURITY_LEVEL or R-MAC only – case of a R-MAC session in progress originally initiated with a BEGIN R-MAC SESSION command):
  - If the Secure Channel Session was aborted during the same Application Session, the incoming command shall be rejected with a security error;
  - Otherwise no security verification of the incoming command shall be performed. The Application processing the command is responsible for applying its own security rules.
- If a Secure Channel Session is active for incoming commands (i.e. Current Security Level at least set to AUTHENTICATED), the security of the incoming command shall be checked according to the Current Security Level regardless of the command secure messaging indicator:
  - When the security of the command does not match (nor exceeds) the Current Security Level, the command shall be rejected with a security error, the Secure Channel Session aborted and the Current Security Level reset to NO_SECURITY_LEVEL;
  - If a security error is found, the command shall be rejected with a security error, the Secure Channel Session aborted and the Current Security Level reset to NO_SECURITY_LEVEL;
  - In all other cases, the Secure Channel Session shall remain active and the Current Security Level unmodified. The Application is responsible for further processing the command.
- If the Security Domain supports application data encryption and/or decryption, it shall decrypt or encrypt a block of secret data upon request. If the service is not supported or if (one of) the appropriate cryptographic key(s) is not available, the request shall be rejected but the Current Security Level, Session Security Level and Secure Channel Session in operation (if any) shall not be impacted;
- If a Secure Channel Session is aborted, it is still considered not terminated;
- The current Secure Channel Session shall be terminated (if aborted or still open) and the Current Security Level reset to NO_SECURITY_LEVEL on either:
  - Attempt to initiate a new Secure Channel Session (new INITIALIZE UPDATE command);
  - Termination of the Application Session (e.g. new Application selection);
  - Termination of the associated logical channel;
  - Termination of the Card Session (card reset or power off);
  - Explicit termination by the Application (e.g. invoking GlobalPlatform API).

In accordance with the general rules described in Chapter 10 – *Secure Communication*, the following protocol rules apply to Secure Channel Protocol '02' with implicit initiation mode:

- The successful implicit initiation of a Secure Channel Session for incoming commands shall set the Current Security Level to AUTHENTICATED and C-MAC (the R-MAC indicator remaining as is);
- The Session Security Level for incoming commands is implicit and set to AUTHENTICATED for the entire Secure Channel Session. It is reset on the normal termination or abort of a Secure Channel Session for incoming commands;
- When the Current Security Level is not set to AUTHENTICATED (i.e. equal to NO_SECURITY_LEVEL or R-MAC only), no Secure Channel Session is active for incoming commands:

- If no secure messaging is indicated in the incoming command, no security verification of the command shall be performed. The Application processing the command is responsible to apply its own security rules;
- If secure messaging is indicated, a new Secure Channel Session initiation for incoming commands shall be attempted and the security of the incoming command verified.

- If a Secure Channel Session is active for incoming commands (i.e. Current Security Level at least set to AUTHENTICATED), the security of the incoming command shall be checked according to the Current Security Level:
  - If no secure messaging is indicated in the command, the Secure Channel Session shall remain active, the Current Security Level set to AUTHENTICATED (the R-MAC indicator remaining as is) and no further security verification of the command performed. The Application processing the command is responsible to apply its own security rules;
  - If secure messaging is indicated and the Current Security Level is set to AUTHENTICATED and C-MAC (ignoring the R-MAC indicator), the security of the incoming command shall be verified:
    - If a security error is found, the command shall be rejected with a security error, the Secure Channel Session aborted and Current Security Level reset to NO_SECURITY_LEVEL or R-MAC only (in case of a R-MAC session in progress);
    - Otherwise, the Secure Channel Session shall remain active and the Current Security Level unmodified. The Application is responsible for further processing the command.
  - If secure messaging is indicated and the Current Security Level is set to AUTHENTICATED (ignoring the R-MAC indicator), the current Secure Channel Session for incoming commands shall be terminated and the Current Security Level reset to NO_SECURITY_LEVEL or R-MAC only (in case of a R-MAC session in progress). A new Secure Channel Session initiation for incoming commands shall be attempted and the security of the incoming command verified.
- The current Secure Channel Session shall be terminated (if aborted or still open) and the Current Security Level reset to NO_SECURITY_LEVEL on either:
  - Termination of the Application Session (e.g. new Application selection);
  - Termination of the associated logical channel;
  - Termination of the Card Session (card reset or power off);
  - Explicit termination by the Application (e.g. invoking GlobalPlatform API).

E.2 Cryptographic Keys

A Security Domain, including the Issuer Security Domain, shall have at least one key set to be used in the initiation and use of a Secure Channel. This key set shall contain at least one double length DES key, the Secure Channel base key. This Secure Channel base key is only used to generate session keys during the initiation of a Secure Channel.

Table E-2: SCP02 – Security Domain Secure Channel Base Key

Key	Usage	Length	Remark
Secure Channel base key	Secure Channel authentication, MAC verification, & sensitive data decryption (DES)	16 bytes	Mandatory

A Security Domain's, including the Issuer Security Domain's, Secure Channel key set may contain 3 double length DES keys: the Secure Channel encryption key (S-ENC), the Secure Channel MAC key (S-MAC), and the data encryption key (DEK) for encrypting and decrypting sensitive data; e.g. secret or private keys. These keys are only used to generate session keys during the initiation of a Secure Channel.

Table E-3: SCP02 – Security Domain Secure Channel Keys

Key	Usage	Length	Remark
Secure Channel encryption key (S-ENC)	Secure Channel authentication & encryption (DES)	16 bytes	Mandatory
Secure Channel message authentication code key (S-MAC)	Secure Channel MAC verification and generation (DES)	16 bytes	Mandatory
Data encryption key (DEK)	Sensitive data encryption and decryption (DES)	16 bytes	Mandatory

E.3 Cryptographic Algorithms

The cryptographic and hashing algorithms described in Appendix B – *Algorithms (Cryptographic and Hashing)* apply to SCP02. This section defines the additional requirements for SCP02.

E.3.1 Cipher Block Chaining (CBC)

The Initial Chaining Vector (ICV) used for chained data encryption in CBC mode is always 8 bytes of binary zero ('00').

The Initial Chaining Vector (ICV) used for chained message integrity in CBC mode is always 8 bytes and initialized during the Secure Channel initiation.

E.3.2 Message Integrity ICV using Explicit Secure Channel Initiation

When using explicit Secure Channel initiation, SCP02 mandates the use of a MAC on the EXTERNAL AUTHENTICATE command.

For the EXTERNAL AUTHENTICATE command MAC verification, the ICV is set to zero.

Once successfully verified, the C-MAC of the EXTERNAL AUTHENTICATE command becomes the ICV for the subsequent C-MAC verification and/or R-MAC generation. Notice that this is true also if the EXTERNAL AUTHENTICATE command does not request R-MAC and R-MAC is subsequently requested by a BEGIN R-MAC SESSION command; i.e. the ICV for R-MAC generation is the C-MAC of the EXTERNAL AUTHENTICATE command that started the Secure Channel session.

E.3.3 Message Integrity ICV using Implicit Secure Channel Initiation

When using implicit Secure Channel Session initiation, the ICV shall be a MAC computed on the AID of the selected Application. The ICV for the first C-MAC calculation of a new Secure Channel Session is calculated as follows:

- Apply reversible padding to the AID of the selected Application as defined in section B.1.3.
- Calculate a MAC as defined in section B.1.2.2 – *Single DES Plus Final Triple DES* with the C-MAC key over the padded Application AID with an ICV value of binary zeroes.

The resulting MAC is the ICV for the first C-MAC of the Secure Channel Session. This ICV makes the Secure Channel Session application specific.

The ICV for the first R-MAC of a Secure Channel Session is calculated the same way, except that the R-MAC key is used in step 2 for the MAC calculation.

E.3.4 ICV Encryption

As an enhancement to the C-MAC mechanism, the ICV is encrypted before being applied to the calculation of the next C-MAC. The encryption mechanism used is single DES with the first half of the Secure Channel C-MAC session key. The first ICV of a session is not encrypted.

E.4 Cryptographic Usage

E.4.1 DES Session Keys

DES session keys are generated every time a Secure Channel is initiated. These session keys may be used for subsequent commands if secure messaging is required.

Session keys are generated to ensure that a different set of keys is used for each secure communication session.

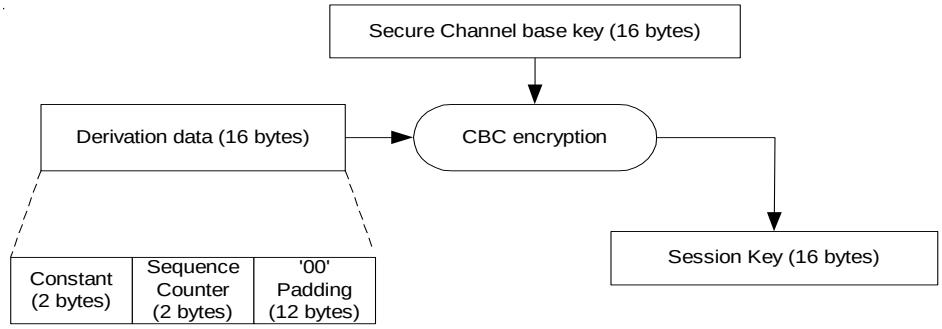
DES session keys are created using the static Secure Channel key(s), a constant, the current value of the Secure Channel Sequence Counter, and a padding of binary zeroes. Creating session keys is performed as follows:

- Generating the Secure Channel C-MAC session keys using the Secure Channel base key or MAC key (S-MAC) and the session keys derivation data with a constant of '0101';
- Generating the Secure Channel R-MAC session keys using the Secure Channel base key or MAC key (S-MAC) and the session keys derivation data with a constant of '0102';
- Generating the Secure Channel encryption session keys using the Secure Channel base key or encryption key (S-ENC) and the session keys derivation data with a constant of '0182';
- Generating the Secure Channel data encryption session keys using the Secure Channel base key or data encryption key (DEK) and the session keys derivation data with a constant of '0181'.

The DES operation used to generate these keys is always triple DES in CBC mode.

R-MAC session keys are generated using the current value of the sequence counter at the time the R-MAC session is initiated. If the R-MAC session starts with the EXTERNAL AUTHENTICATE command, the same sequence counter value is used to generate the R-MAC session key as is used to generate the other session keys for the same Secure Channel Session.

Figure E-2: Create Secure Channel Session Key from the Base Key



E.4.2 Authentication Cryptograms in Explicit Secure Channel Initiation

Both the card and the off-card entity (host) generate an authentication cryptogram. The off-card entity verifies the card cryptogram and the card verifies the host cryptogram. Generating or verifying an authentication cryptogram uses the S-ENC session key and the signing method described in section B.1.2.1 – *Full Triple DES*.

E.4.2.1 Card Authentication Cryptogram

The generation and verification of the card cryptogram is performed by concatenating the 8-byte host challenge, 2-byte Sequence Counter, and 6-byte card challenge resulting in a 16-byte block.

Applying the same padding rules defined in section B.1.3, the data shall be padded with a further 8-byte block ('80 00 00 00 00 00 00 00').

The signature method, using the S-ENC session key and an ICV of binary zeroes, is applied across this 24-byte block and the resulting 8-byte signature is the card cryptogram.

E.4.2.2 Host Authentication Cryptogram

The generation and verification of the host cryptogram is performed by concatenating the 2-byte Sequence Counter, 6-byte card challenge, and 8-byte host challenge resulting in a 16-byte block.

Applying the same padding rules defined in section B.1.3, the data shall be padded with a further 8-byte block ('80 00 00 00 00 00 00 00').

The signature method, using the S-ENC session key and an ICV of binary zeroes, is applied across this 24-byte block and the resulting 8-byte signature is the host cryptogram.

E.4.2.3 Card Challenge

The card challenge is either a random or pseudo-random number that shall be unique to a Secure Channel Session. A pseudo-random card challenge may be generated as follows:

- The AID of the Application requesting to open the Secure Channel is padded according to the padding rules defined in section B.1.3;
- A MAC is calculated across the padded data as defined in section B.1.2.2 – *Single DES Plus Final Triple DES MAC*, using the C-MAC session key and an ICV of binary zeroes;
- The six leftmost bytes of the resultant MAC constitute the card challenge.

E.4.3 Authentication Cryptogram in Implicit Secure Channel Initiation

When using the implicit Secure Channel Session initiation, the authentication cryptogram is the first C-MAC received by the card from the off-card entity.

The rules for C-MAC generation and verification are detailed in the following subsection.

E.4.4 APDU Command C-MAC Generation and Verification

A C-MAC is generated by an off-card entity and applied across the full APDU command being transmitted to the card including the header and the data field in the command message. It does not include Le.

C-MAC generation and verification uses the Secure Channel C-MAC session key, an ICV and the signature method described in section B.1.2.2 – *Single DES Plus Final Triple DES*. (Prior to using the ICV, the ICV can be encrypted as described in section E.3.4 – *ICV Encryption*).

The ICV is used to chain the commands for command sequence integrity; the initial value of the ICV is described in section E.3 – *Cryptographic Algorithms*. For any subsequent command following the first successful C-MAC verification, the ICV is the C-MAC value successfully verified for the previous command received by the card.

The signature method, using the Secure Channel C-MAC session key and the ICV, is applied across the padded command message and the resulting 8-byte signature is the C-MAC. The rules for C-MAC padding are as defined in section B.1.3.

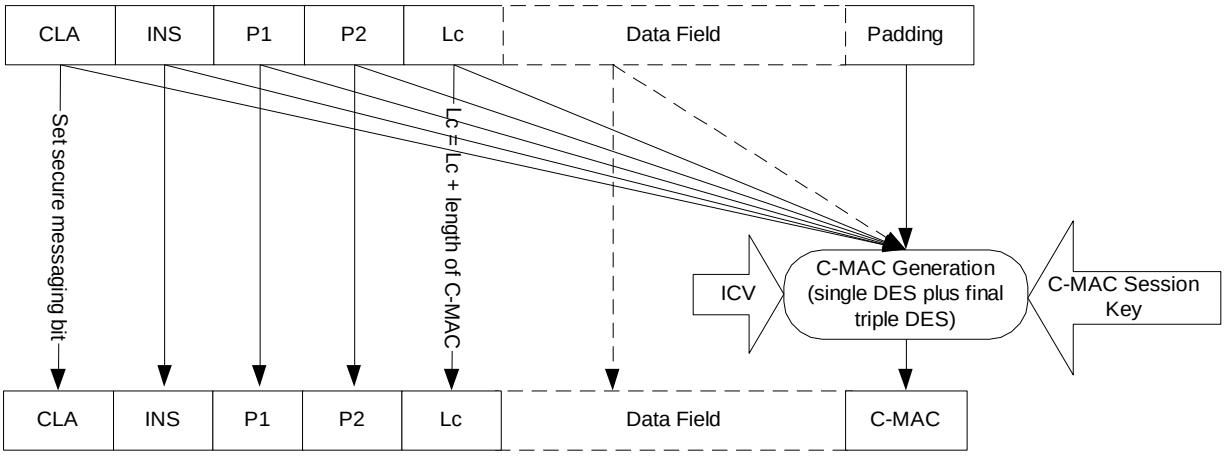
In order to compute a C-MAC, the APDU command shall be prepared as follows:

- Any indication of logical channel number shall be removed from the class byte.
- Padding shall be added to APDU command data.

The two following methods are defined for the C-MAC generation:

- C-MAC generation on unmodified APDU
  - No further modification is required before generating the C-MAC.
  - After generating the C-MAC, the length of the command message (Lc) shall be incremented by 8 to indicate the inclusion of the C-MAC in the data field of the command message.

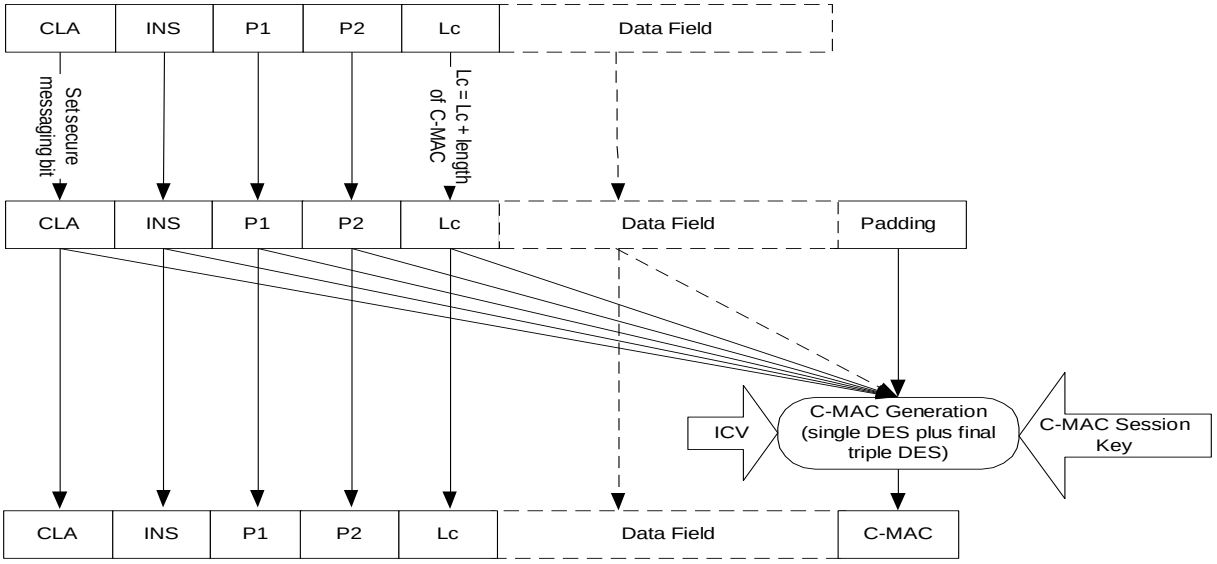
Figure E-3: C-MAC Generation on Unmodified APDU



- C-MAC generation on modified APDU
  - Before generating the C-MAC, the following additional modifications shall be performed:

- The class byte shall have bit 4 set to 0 and bit 3 set to 1 to indicate GlobalPlatform proprietary secure messaging.
- The length of the command message (Lc) shall be incremented by 8 to indicate the inclusion of the C-MAC in the data field of the command message.

Figure E-4: C-MAC Generation on Modified APDU



Any padding shall be removed and the C-MAC shall be appended at the end of the APDU command message. Any modification already made to the command header (class and/or Lc) shall be maintained.

If the command is sent on a logical channel, the logical channel number shall be added to the class byte. If logical channel number 0 to 3 is used, the GlobalPlatform proprietary secure messaging shall be indicated in the class byte by setting bit 3 to 1 – see Table 11-11. If logical channel number 4 to 19 is used, the GlobalPlatform proprietary secure messaging shall be indicated in the class byte setting bit 6 to 1 – see Table 11-12.

If data field encryption is required by the current Security Level, it shall now be performed according to section E.4.6. Otherwise, the message is ready to be transmitted to the card.

The card shall apply the same padding mechanism and use the same ICV and C-MAC session key as the off-card entity in order to verify the C-MAC. The verified C-MAC shall be retained and used as the ICV for any subsequent C-MAC verification, regardless of whether the APDU completed successfully or not; i.e. a verified C-MAC shall never be discarded in favor of a previously verified C-MAC.

E.4.5 APDU Response R-MAC Generation and Verification

When using explicit Secure Channel Session initiation, the off-card entity instructs the card whether R-MAC generation applies to all the subsequent command/response messages. R-MAC computation does not include the EXTERNAL AUTHENTICATE command/response pair and lasts until the end of the Secure Channel Session.

When using explicit or implicit Secure Channel Session initiation, the off-card entity instructs the card to start APDU response message integrity using the BEGIN R-MAC SESSION command (see section E.5.3 – *BEGIN R-MAC SESSION Command*). R-MAC computation includes the BEGIN R-MAC SESSION command/response pair and lasts until reception of the END R-MAC SESSION command (see section E.5.4 – *END R-MAC SESSION Command*).

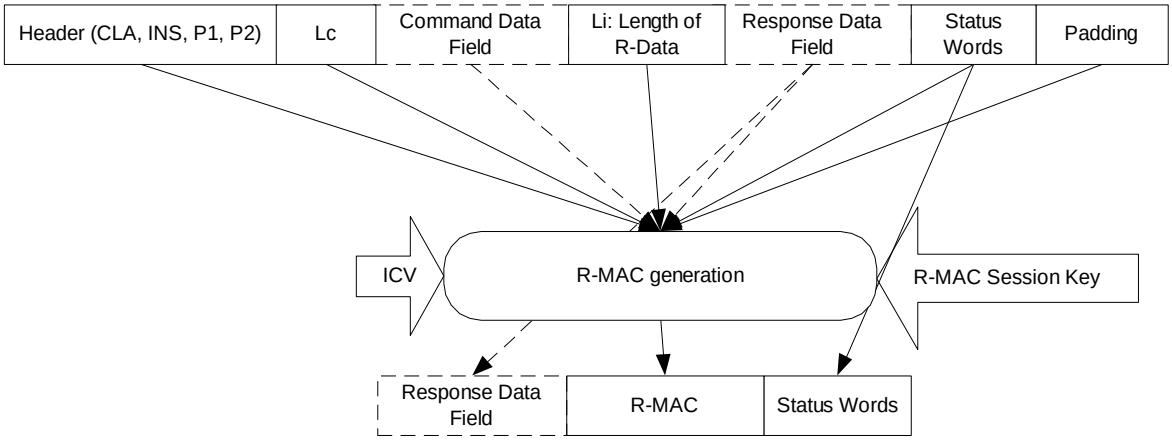
The R-MAC is computed on the following data block:

- The stripped APDU command message; i.e. without any C-MAC and modified command header (the logical channel number is always assumed to be zero). In the case of a case 1 or case 2 command, Lc is always present and set to zero;
- In case of a successful execution or a warning, the response data preceded with a byte Li that codes its length modulo 256. Li is generated by the Security Domain. If there is no response data this byte is present and is set to zero;
- In case of an error, a byte set to '00' indicating no response data;
- The status bytes.

When an R-MAC is returned in response to every command, in the event of case 1 and case 3 commands received by the card, these commands shall be processed respectively as case 2 and case 4 commands with Le set to zero.

The signature method, using the Secure Channel R-MAC session key and the ICV, is applied across the padded data block and the resulting 8-byte signature is the R-MAC. The rules for R-MAC padding are as defined in section B.1.3.

Figure E-5: R-MAC Generation



The R-MAC can be retrieved from the card by sending an END R-MAC SESSION command. The END R-MAC SESSION command allows the off-card entity to instruct the card to either terminate or continue the R-MAC session. The END R-MAC SESSION command is not included in the R-MAC computation.

The off-card entity, in order to verify the R-MAC, shall perform the same padding mechanism to the command/response pair and use the same ICV and R-MAC session key employed by the card in order to verify the R-MAC. The generated R-MAC shall be retained and used as the ICV for any subsequent R-MAC generation. (This is true regardless of whether the APDU command completed successfully or not.)

If the Secure Channel Session is occurring on a Supplementary Logical Channel, the logical channel information is not included in any of the R-MAC computations. The R-MAC is generated with a logical channel number assumed to be zero and the off-card entity is either not aware of, or removes, any logical channel information from the class byte of the corresponding command when verifying the R-MAC.

E.4.6 APDU Command Data Field Encryption and Decryption

Depending on the Session Security Level defined in the explicit initiation of the Secure Channel, all subsequent APDU commands within the Secure Channel may require secure messaging and as such the use of a C-MAC (integrity) and encryption (confidentiality).

Note that, with an implicit initiation of the Secure Channel, command message data field encryption does not apply.

If confidentiality is required, the off-card entity encrypts the data field of the command message being transmitted to the card. This excludes the header and the C-MAC but includes any data within the data field that has been protected for another purpose e.g. secret or private keys encrypted with the data encryption session key.

Command message encryption and decryption uses the Secure Channel encryption session key and the encryption method described in section B.1.1.1 – *Encryption/Decryption CBC Mode*.

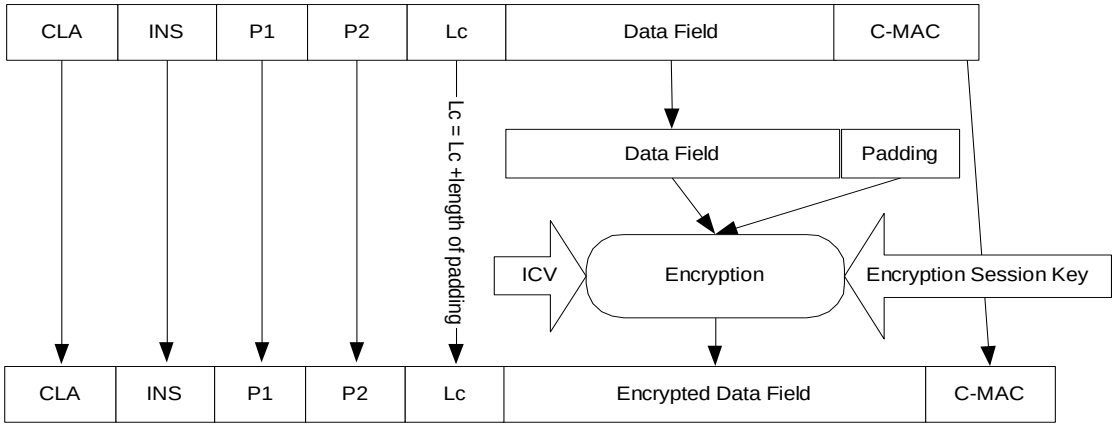
Prior to encrypting the data, the data shall be padded. Unlike the C-MAC, this padding now becomes part of the data field and this necessitates further modification of the Lc value.

Padding of the data field to be encrypted is performed according to section B.1.3.

The number of bytes appended to the data field in order to fulfill the above padding shall be added to Lc. This includes the mandatory '80' and any optional binary zeroes.

The encryption can now be performed across the padded data field using the Secure Channel encryption session key and the result of each encryption becomes part of the encrypted data field in the command message.

Figure E-6: APDU Command Data Field Encryption



Note: The ICV and the chaining are only used to link the blocks of the data currently being encrypted. For this reason the ICV for command data encryption is always binary zeroes.

The message is now prepared for transmission to the card. The card is required to first decrypt the command message and strip off any padding prior to attempting to verify the C-MAC. This decryption uses an ICV of binary zeroes and the same encryption session key employed by the off-card entity. The padding shall be removed and Lc shall be modified to reflect the length prior to encryption; i.e. original clear text data plus C-MAC length.

E.4.7 Sensitive Data Encryption and Decryption

Data encryption is used when transmitting sensitive data to the card and is over and beyond the Current Security Level required for the Secure Channel; For instance all DES keys transmitted to a card (e.g. in a PUT KEY command) should be encrypted.

The data encryption process uses the data encryption session key and the encryption method described in section B.1.1.2 – *Encryption/Decryption ECB Mode* when using explicit initiation of the Secure Channel and section B.1.1.1 – *Encryption/Decryption CBC Mode* when using implicit initiation of the Secure Channel.

As all DES keys are by their very nature a multiple of 8-byte lengths no padding is required for key encryption operations. Similarly the sensitive data block length shall be constructed as a multiple of 8-byte long block before the data encryption operations: the eventual padding method is application specific.

The encryption is performed across the sensitive data and the result of each encryption becomes part of the encrypted data. This encrypted data becomes part of the ‘clear text’ data field in the command message.

The on-card decryption of key data, is the exact opposite of the above operation: in particular, no padding is removed by the decryption operation.

E.5 Secure Channel APDU Commands

Table E-4: SCP02 Command Support

Command	Secure Channel Initiation	
	Explicit	**Implicit**
INITIALIZE UPDATE	✓	
EXTERNAL AUTHENTICATE	✓	
BEGIN R-MAC SESSION		
END R-MAC SESSION		

The following table summarizes the minimum security requirements for the APDU commands.

Table E-5: Minimum Security Requirements for SCP02 Commands

Command	Minimum Security	
	Explicit	**Implicit**
INITIALIZE UPDATE	None	Dependent on the Issuer security policy
EXTERNAL AUTHENTICATE	C-MAC	
BEGIN R-MAC SESSION	Secure Channel initiation	
END R-MAC SESSION	Secure Channel initiation	

Note that Table E-4 shall be used in conjunction with Table E-6 to determine SCP02 command support requirements. The following table provides the list of SCP02 command support per card Life Cycle State.

Table E-6: SCP02 Command Support per card Life Cycle State

Command	OP_READY			INITIALIZED			SECURED			CARD_LOCKED	TERMINATED
	AM SD	DM SD	SD	AM SD	DM SD	SD	AM SD	DM SD	SD	SD	SD
INITIALIZE UPDATE	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
EXTERNAL AUTHENTICATE	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
BEGIN R-MAC SESSION											
END R-MAC SESSION											

Legend of Table E-4 and Table E-6:

AM SD	Security Domain with Authorized Management privilege.
DM SD	Supplementary Security Domain with Delegated Management privilege.
SD	Other Security Domain.
✓	Support required.
Blank cell	Support optional.
Striped cell	Support prohibited.

E.5.1 INITIALIZE UPDATE Command

E.5.1.1 Definition and Scope

The INITIALIZE UPDATE command is used, during explicit initiation of a Secure Channel, to transmit card and session data between the card and the host. This command initiates the initiation of a Secure Channel Session.

At any time during a current Secure Channel, the INITIALIZE UPDATE command can be issued to the card in order to initiate a new Secure Channel Session.

This INITIALIZE UPDATE command is not to be confused with commands of the same name in legacy applications.

E.5.1.2 Command Message

The INITIALIZE UPDATE command message is coded according to the following table:

Table E-7: INITIALIZE UPDATE Command Message

Code	Value	Meaning
CLA	'80' - '83' or 'C0' - 'CF'	See section 11.1.4
INS	'50'	INITIALIZE UPDATE
P1	'xx'	Key Version Number
P2	'00'	Reference control parameter P2
Lc	'08'	Length of host challenge
Data	'xx xx…'	Host challenge
Le	'00'	

E.5.1.3 Reference Control Parameter P1 – Key Version Number

The Key Version Number defines the Key Version Number within the Security Domain to be used to initiate the Secure Channel Session. If this value is zero, the first available key chosen by the Security Domain will be used.

E.5.1.4 Reference Control Parameter P2

The reference control parameter P2 shall always be set to '00'.

E.5.1.5 Data Field Sent in the Command Message

The data field of the command message contains 8 bytes of host challenge. This challenge, chosen by the off-card entity, should be unique to this session.

E.5.1.6 Response Message

The data field of the response message contains the concatenation without delimiters of the following data elements:

Table E-8: INITIALIZE UPDATE Response Message

Name	Length
Key diversification data	10 bytes
Key information	2 bytes
Sequence Counter	2 bytes
Card challenge	6 bytes
Card cryptogram	8 bytes

The key diversification data is data typically used by a backend system to derive the card static keys.

The key information includes the Key Version Number and the Secure Channel Protocol identifier, here '02', used in initiating the Secure Channel Session.

The Sequence Counter is an internal incremental counter used for creating session keys.

The card challenge is an internally generated random number.

The card cryptogram is an authentication cryptogram.

E.5.1.7 Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

This command may return either a general error condition as listed in section 11.1.3 – *General Error Conditions* or the following error condition:

Table E-9: INITIALIZE UPDATE Error Condition

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

E.5.2 EXTERNAL AUTHENTICATE Command

E.5.2.1 Definition and Scope

The EXTERNAL AUTHENTICATE command is used by the card, during explicit initiation of a Secure Channel, to authenticate the host and to determine the level of security required for all subsequent commands.

A successful execution of the INITIALIZE UPDATE command shall precede this command.

E.5.2.2 Command Message

The EXTERNAL AUTHENTICATE command message is coded according to the following table.

Table E-10: EXTERNAL AUTHENTICATE Command Message

Code	Value	Meaning
CLA	'84' - '87' or 'E0' - 'EF'	See section 11.1.4
INS	'82'	EXTERNAL AUTHENTICATE
P1	'xx'	Security level
P2	'00'	Reference control parameter P2
Lc	'10'	Length of host cryptogram and MAC
Data	'xx xx…'	Host cryptogram and MAC
Le		Not present

E.5.2.3 Reference Control Parameter P1 – Security Level

The reference control parameter P1 defines the level of security for all secure messaging commands following this EXTERNAL AUTHENTICATE command (it does not apply to this command) and within this Secure Channel Session.

Table E-11: EXTERNAL AUTHENTICATE Reference Control Parameter P1

b8	b7	b6	b5	b4	b3	b2	b1	Description
0	0	1	1	0	0	1	1	RFU
0	0	1	1	0	0	0	1	RFU
0	0	1	1	0	0	0	0	RFU
0	0	0	1	0	0	1	1	C-DECRYPTION, C-MAC, and R-MAC
0	0	0	1	0	0	0	1	C-MAC and R-MAC
0	0	0	1	0	0	0	0	R-MAC
0	0	0	0	0	0	1	1	C-DECRYPTION and C-MAC
0	0	0	0	0	0	0	1	C-MAC
0	0	0	0	0	0	0	0	No secure messaging expected.

E.5.2.4 Reference Control Parameter P2

The reference control parameter P2 shall always be set to '00'.

E.5.2.5 Data Field Sent in the Command Message

The data field of the command message contains the host cryptogram and the C-MAC.

E.5.2.6 Data Field Returned in the Response Message

The data field of the response message is not present.

E.5.2.7 Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

This command may return either a general error condition as listed in section 11.1.3 – *General Error Conditions* or the following warning condition:

Table E-12: EXTERNAL AUTHENTICATE Warning Code

SW1	SW2	Meaning
'63'	'00'	Authentication of host cryptogram failed

E.5.3 BEGIN R-MAC SESSION Command

E.5.3.1 Definition and Scope

The BEGIN R-MAC SESSION command is used to initiate a Secure Channel Session for APDU response message integrity. At any time, the BEGIN R-MAC SESSION command may be issued to the card in order to initiate a R-MAC session.

E.5.3.2 Command Message

The BEGIN R-MAC SESSION command message is coded according to the following table:

Table E-13: BEGIN R-MAC SESSION Command Message

Code	Value	Meaning
CLA	'80' - '87', 'C0' - 'CF' or 'E0' - 'EF'	See section 11.1.4
INS	'7A'	BEGIN R-MAC SESSION
P1	'xx'	Reference control parameter P1
P2	'01'	Reference control parameter P2
Lc	'xx'	Length of data field, if any
Data	'xx xx…'	BEGIN R-MAC SESSION data and C-MAC, if needed
Le		Not present

E.5.3.3 Reference Control Parameter P1

The reference control parameter P1 defines the level of security for all subsequent APDU response messages following this BEGIN R-MAC SESSION command (it does not apply to this command).

Table E-14: BEGIN R-MAC SESSION Reference Control Parameter P1

b8	b7	b6	b5	b4	b3	b2	b1	Description
0	0	1	1	0	0	0	0	Response Encryption and R-MAC (RFU)
0	0	0	1	0	0	0	0	R-MAC
0	0	0	0	0	0	0	0	No secure messaging expected

When P1 is set to '10' each APDU response message during the R-MAC session includes a R-MAC.

When P1 is set to '00' only the END R-MAC SESSION response message will contain a R-MAC.

E.5.3.4 Reference Control Parameter P2

The reference control parameter P2 defines the beginning of the session for APDU response message integrity.

Table E-15: BEGIN R-MAC SESSION Reference Control Parameter P2

b8	b7	b6	b5	b4	b3	b2	b1	Description
0	0	0	0	0	0	0	1	Begin R-MAC session

E.5.3.5 Data Field Sent in the Command Message

The data field of the BEGIN R-MAC SESSION contains an LV coded ‘data’ element and optionally a C–MAC. The card does not interpret the ‘data’. However since it is included in R-MAC calculation, this gives the off-card entity the possibility to include a challenge in the R-MAC.

The following table details the BEGIN R-MAC SESSION data field:

Table E-16: BEGIN R-MAC SESSION Command Data Field

Length	Name	Presence
1	Length of data	Mandatory
0-24	data	Conditional

E.5.3.6 Data Field Returned in the Response Message

The data field of the response message is not present.

E.5.3.7 Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

This command may return either a general error condition as listed in section 11.1.3 – *General Error Conditions* or the following error condition:

Table E-17: BEGIN R-MAC SESSION Error Conditions

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

E.5.4 END R-MAC SESSION Command

E.5.4.1 Definition and Scope

The END R-MAC SESSION command is used to terminate a Secure Channel Session for APDU response message integrity or to retrieve the current R-MAC without ending the R-MAC Session. The END R-MAC SESSION command may be issued to the card at any time during an R-MAC session.

E.5.4.2 Command Message

The END R-MAC SESSION command message is coded according to the following table:

Table E-18: END R-MAC SESSION Command Message

Code	Value	Meaning
CLA	'80' - '87', 'C0' - 'CF' or 'E0' - 'EF'	See section 11.1.4
INS	'78'	END R-MAC SESSION
P1	'00'	Reference control parameter P1
P2	'01' or '03'	Reference control parameter P2
Lc	'xx'	Length of data field, if any
Data	'xx xx…'	C-MAC, if needed
Le	'00'	

E.5.4.3 Reference Control Parameter P1

Reference control parameter P1 shall always be set to '00'.

E.5.4.4 Reference Control Parameter P2

The reference control parameter P2 is coded according to the following table:

Table E-19: END R-MAC SESSION Reference Control Parameter P2

b8	b7	b6	b5	b4	b3	b2	b1	Description
0	0	0	0	0	0	1	1	End R-MAC session & return R-MAC
0	0	0	0	0	0	0	1	Return R-MAC

E.5.4.5 Data Field Sent in the Command Message

The data field of the command message may optionally contain a C-MAC.

E.5.4.6 Data Field Returned in the Response Message

The data field of response message contains the R-MAC of the current R-MAC session.

E.5.4.7 Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

This command may return either a general error condition as listed in section 11.1.3 – *General Error Conditions* or the following error condition:

Table E-20: END R-MAC SESSION Error Conditions

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

F Secure Channel Protocol '10'

F.1 Secure Communication

Secure Channel Protocol '10' (SCP10) supports up to 19 Supplementary Logical Channels.

F.1.1 SCP10 Secure Channel

SCP10 is a Secure Channel Protocol based on asymmetric cryptography and Public Key infrastructure (PKI). The Secure Channel Protocol operates between a Security Domain on the card (or the Issuer Security Domain) and an Off-Card Entity (OCE) which may be the Application Provider (or Card Issuer) or another party.

F.1.1.1 Synchronous Mode

The synchronous implementation of SCP10 provides the following levels of security:

Authentication - in which the Security Domain authenticates the Off-Card Entity and the Off-Card Entity may authenticate the Security Domain. There are two aspects to this: Key Authentication which involves establishing mutual trust in each other's public key (PK); and Entity Authentication which involves establishing that the other party in the Secure Channel Session is the authentic owner of that public key.

Integrity and data origin authentication – in which the Security Domain and the Off-Card Entity ensure that the data being received from the other entity actually came from its claimed source in the correct sequence and has not been altered.

Confidentiality – in which confidential data is not viewable by an unauthorized entity.

A further level of security applies to specific sensitive data (e.g. cryptographic keys) which may be encrypted using a mechanism that is not part of the Secure Channel Session security.

F.1.1.2 Asynchronous Mode

The asynchronous mode allows off-line pre-packaging of encrypted and signed content and provides the following levels of security:

Integrity and data origin authentication – in which the Security Domain confirms the public-key-signed signature of the supplied content. This enables that the content being received from the other entity actually came from its claimed source in the correct sequence and has not been altered.

Confidentiality – in which confidential data is not viewable by an unauthorized entity. Confidential content is encrypted with the public key of the desired Security Domain.

F.1.2 Initiating a Secure Channel

The steps involved in initiating a Secure Channel Protocol are as follows:

1. **Certificate Verification (optional)** – the Security Domain and the Off-Card Entity (OCE) may need to traverse and verify a chain of certificates from established trust points down to each other's public key. The extent to which this is needed depends on what keys are currently validated by each party: the null condition is where they have both already validated each other's public key, in which case no explicit Certificate Verification is necessary;
2. **Entity Authentication** – the Security Domain and optionally the Off-Card Entity further check the authenticity of the other party by verifying the signature of a challenge sent to the other party;
3. **Session Key Establishment** – the two parties establish symmetric session keys for subsequent secure messaging.

Only explicit Secure Channel initiation is supported in SCP10. There is no specific command to terminate a session.

Some of the variations on the Secure Channel Protocol supported by the card or the Security Domain are announced in the parameter "i" in Card Recognition Data or Security Domain Management Data. See Appendix H – *GlobalPlatform Data Values* for details of Card Recognition Data and Security Domain Management Data. The value "i" is coded on one byte as a bitmap as follows:

Table F-1: Values of Parameter "i"

b8	b7	b6	b5	b4	b3	b2	b1	Description
Not available	0	0	0	0	0	-	0	Key Transport
Not available	0	0	0	0	0	-	1	Key Agreement
Not available	0	0	0	0	0	0	-	Signature with message recovery
Not available	0	0	0	0	0	1	-	Signature without message recovery

Note: "i" is a subidentifier within an object identifier, and bit b8 is reserved for use in the structure of the object identifier according to [ISO 8825-1].

Key transport and key agreement relate to the process of establishing session keys for the Secure Channel Session.

- With **key agreement** the Security Domain and the Off-Card Entity exchange secret values when the Secure Channel is being initiated, and session keys are then derived from those secrets using an algorithm known to both the Off-Card Entity and the Security Domain;
- With **key transport** the Security Domain receives session keys to be used for the Secure Channel Session from the Off-Card Entity during Secure Channel initiation.

Signature with message recovery and signature without message recovery refer to the signature scheme used for digital signatures in data messages during Entity Authentication. (Note that these mechanisms apply also with the signatures on digital certificates, but the value "i" does not refer to this.)

- With **signature with message recovery**, part or all of the message data that is signed is contained in the signature block and is recovered during the process of verifying the signature. Signature with message recovery is standardized in [ISO 9796-2];
- With **signature without message recovery**, the signature does not contain any part of the message data that is signed, but comprises an appendix to the complete message. Such a scheme is also known as a signature scheme 'with appendix'. Signature without message recovery is standardized in [PKCS#1], and is also used in ITU Recommendation X.509 ([X.509]).

The Security Domain may support any combination of values of parameter "i", but shall support at least one of the following options as defined by "i":

- "i" = '01': Signature with recovery, key agreement;
- "i" = '02': Signature without recovery, key transport.

There is no requirement for a Security Domain to support more than one certificate format.

F.1.3 Certificate Verification

F.1.3.1 Overview

The Security Domain verifies a certificate of the Off-Card Entity to establish the validity of its public key. A certificate must be issued by the Trust Point for External Authentication (TP_EX) or a subordinate key authority of the TP_EX. On the other hand, the Off-Card Entity establishes the validity of the Security Domain's public key by verifying a certificate issued by the Key Authority of the Security Domain. The Security Domain shall hold a single (default) public key of the Trust Point for External Authentication (PK.TP_EX.AUT) and may also hold the validated public keys of other authorities in a certificate chain.

The Off-Card Entity may know implicitly what other public keys have already been validated by the Security Domain, and can use this knowledge to reduce the number of certificates the Security Domain is asked to verify, potentially down to zero.

By default the Security Domain expects the first certificate presented for verification to be signed by the PK.TP_EX.AUT, and each subsequent certificate to be signed by the public key validated with the previous certificate. The Off-Card Entity may request the Security Domain to use a different default initial public key by indicating it with a MANAGE SECURITY ENVIRONMENT command.

The Off-Card Entity establishes the validity of the Security Domain's public key by checking a (chain of) certificate(s).

The minimum set of keys and certificates to be stored by a Security Domain that supports asymmetric Secure Channel Protocol '10' shall be as follows:

- One Public Key for Trust Point for External Authentication (PK.TP_EX.AUT);
- One Security Domain Private Key (SK.SD.AUT);
- One Security Domain Public Key (PK.SD.AUT); and
- One Security Domain Certificate (CERT.SD.AUT) corresponding to the Security Domain Public Key and signed by the Security Domain Key Authority.

The Security Domain may also hold:

- The Off-Card Entity's Public Key(s) (PK.OCE.AUT) for External Authentication which has been validated;
- The Key Authority's Public Key(s) (PK.KA_EX.AUT) for External Authentication in a valid certificate chain from the Trust Point for External Authentication to the Off-Card Entity; and
- The Key Authority's Certificate(s) for Internal Authentication (CERT.KA_IN.AUT) in a valid certificate chain from the Trust Point for Internal Authentication to the Security Domain.

The minimum set of keys and certificates to be available to an Off-Card Entity (OCE) that wishes to communicate with a specific Security Domain is assumed to be as follows:

- One Public Key for Trust Point for Internal Authentication (PK.TP_IN.AUT));
- One Off-Card Entity Public Key (PK.OCE.AUT);
- One Off-Card Entity Certificate (CERT.OCE.AUT) corresponding to the Off-Card Entity Public Key and signed by the Off-Card Entity's Key Authority;
- The Off-Card Entity Trust Point Certification Public Key (PK.TP_OCE.AUT) that participates in a chain of certificates down to the Security Domain's Public Key.

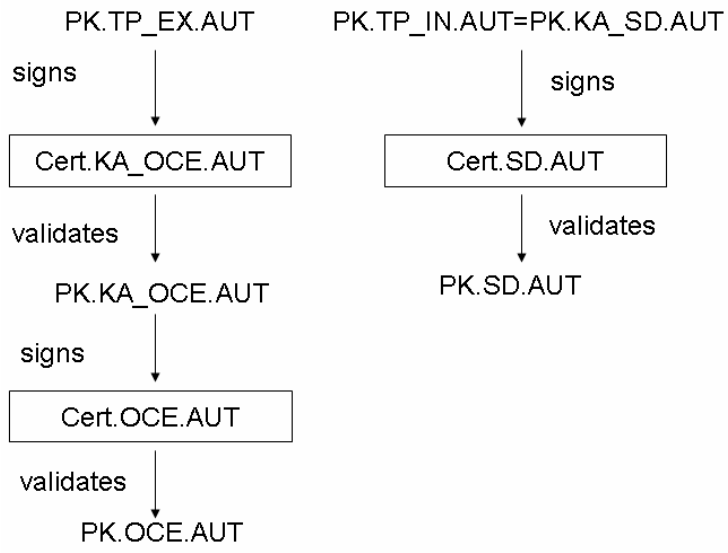
The Off-Card Entity may also hold:

- The Key Authority's Certificate(s) for External Authentication (CERT.KA_EX.AUT) in a valid certificate chain from the Trust Point for External Authentication to the Off-Card Entity.

Note that the Security Domain's owner may be considered to be the Application Provider (for a Security Domain) or the Card Issuer (for the Issuer Security Domain).

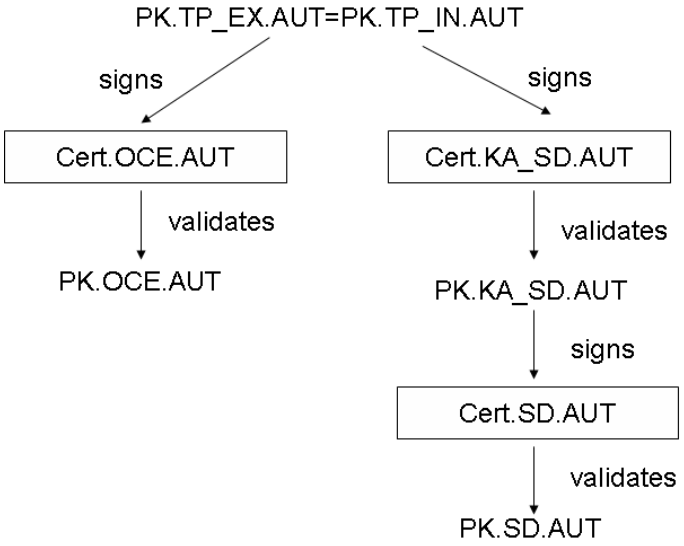
The first example in Figure F-1 is a simple case where the Trust Point for External Authentication (TP_EX) and the Trust Point for Internal Authentication (TP_IN) certify the public key of the Key Authority of the Off-Card Entity and the Security Domain respectively, and the Key Authority of the Off-Card Entity (KA_OCE) certifies the public key of the Off-Card Entity.

Figure F-1: Certificate Chains – Example a



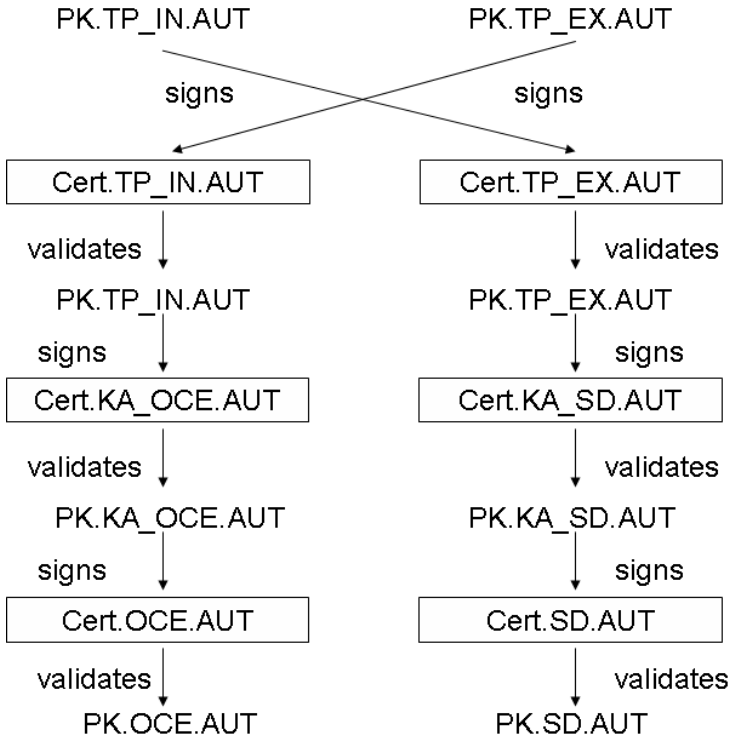
The second example in Figure F-2 illustrates that the Trust Point for External Authentication (TP_EX) and Internal Authentication (TP_IN) directly certifies the public key of both the Off-Card Entity and the Security Domain's Key Authority; and the Security Domain's Key Authority in turn certifies the Security Domain's public key.

Figure F-2: Certificate Chains – Example b



The third example in Figure F-3 illustrates that the Trust Point for External Authentication (TP_EX) and Internal Authentication (TP_IN) cross-certify each other's public keys, and there are two certificate chains down to the Security Domain and OCE public keys.

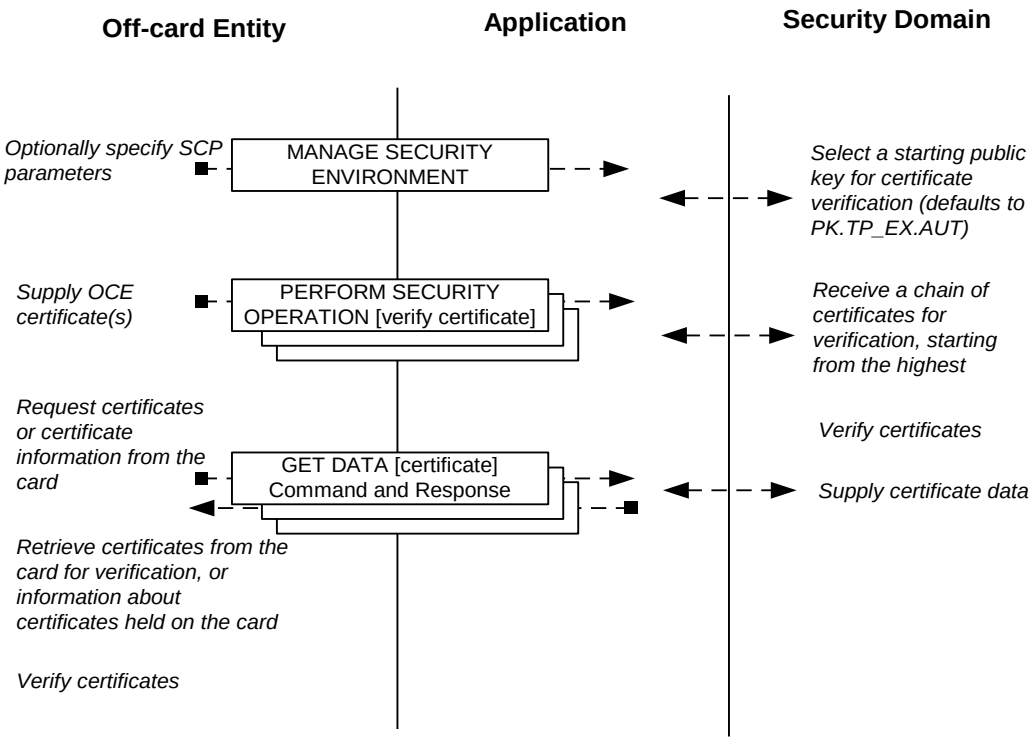
Figure F-3: Certificate Chains – Example c



F.1.3.2 Certificate Verification Process Flow

The following flow is an example of the Certificate Verification stage of Secure Channel initiation. Expanding on the authentication processing shown in the flow described in section F.3.1.1 it can be seen how an Application would use the services of a Security Domain to perform Certificate Verification.

Figure F-4: Certificate Verification Flow



The following commands shall be supported:

- MANAGE SECURITY ENVIRONMENT containing a ‘cryptographic mechanism reference’ designating the GlobalPlatform Secure Channel Protocol '10', see section F.4.5 for further details;
- PERFORM SECURITY OPERATION [verify certificate] command, see section F.4.7 for further details;
- GET DATA [certificate] command, see section F.4.3 for further details.

On receipt of a MANAGE SECURITY ENVIRONMENT command, any existing Secure Channel Session (on the same logical channel of the same card I/O interface) shall be terminated, regardless of the validity of the command: both the Current Security Level and Session Security Level are reset to NO_SECURITY_LEVEL. The MANAGE SECURITY ENVIRONMENT command may refer to a previously validated public key. If the MANAGE SECURITY ENVIRONMENT command is omitted, Security Domain default values and options shall apply, in particular for the initial default public key to use in subsequent command processing.

On receipt of a PERFORM SECURITY OPERATION [verify certificate] command not preceded by a MANAGE SECURITY ENVIRONMENT or another PERFORM SECURITY OPERATION [verify certificate] command, any existing Secure Channel Session (on the same logical channel of the same card I/O interface) shall be terminated, regardless of the result of the certificate verification: both the Current Security Level and Session Security Level are reset to NO_SECURITY_LEVEL.

Multiple PERFORM SECURITY OPERATION [verify certificate] commands may be received. In this case, a chain of certificates is presented to the Security Domain and the certificate contained in each command is verified using the Current Public Key. The Current Public Key is the public key that the Security Domain validated when verifying the last certificate presented by the Off-Card Entity during the same Secure Channel Session initiation. The Security Domain shall have a default Current Public Key at the start of a Secure Channel Session initiation phase: Trust Point for External Authentication (PK.TP_EX.AUT).

Any failure in verifying an Off-Card Entity's certificate aborts the current Secure Channel Session initiation phase, and any public keys validated during that initiation phase shall be discarded.

Multiple GET DATA [certificate] commands may be received. They may be interleaved with PERFORM SECURITY OPERATION [verify certificate] commands. The Security Domain makes no assumptions about whether the Off-Card Entity has obtained enough certificates in order to validate the Security Domain's public key. The Off-Card Entity may use the GET DATA [certificate] command for EF.OD to retrieve information about the different certificates the Security Domain holds, see F.1.2.3 for further details on EF.OD.

F.1.3.3 Certificate Information

The Security Domain shall support access to EF.OD with a 'data object id' set to '5031'. EF.OD identifies each certificate that can be retrieved by the Off-Card Entity for verification. EF.OD contains a set of Cryptographic Information Objects (CIOs) as defined in [ISO 7816-15], each of which describes a certificate and gives a pointer (in the form of a 'data object id') to the certificate data present within the Security Domain. The contents of EF.OD and its consistency with the certificates actually present within the Security Domain are the responsibility of the Security Domain's Provider and beyond the control of the card.

According to [ISO 7816-15], an individual CIO for a certificate may be coded as follows (xxCertificate being of type CertificateChoice):

```
xxCertificate:
{
 commonObjectAttributes
      {
      label                Label              OPTIONAL
      flags                CommonObjectFlags OPTIONAL,
      authId               Identifier  OPTIONAL,
      userConsent          INTEGER (1..cia-ub-userConsent)       OPTIONAL,
      accessControlRules   SEQUENCE SIZE (1..MAX) OF AccessControlRule OPTIONAL,
      ….},
 classAttributes
      {
      id              Identifier,
      authority       BOOLEAN      DEFAULT FALSE,
      identifier      CredentialIdentifier {{KeyIdentifiers}}       OPTIONAL,
      certHash        [0] CertHash     OPTIONAL,
      trustedUsage    [1] Usage OPTIONAL
      identifiers     [2] SEQUENCE OF CredentialIdentifier {{KeyIdentifiers}} OPTIONAL,
      validity        [4] Validity     OPTIONAL,
      ….},
 typeAttributes
      {
      value           ObjectValue {Certificate},
      ….}
}
```

Where typeAttributes vary per type of CertificateChoice, for instance x509CertificateAttributes are:

```
 typeAttributes
      {
      value           ObjectValue {Certificate},
      subject         Name             OPTIONAL,
      issuer          [0] Name  OPTIONAL,
      serialNumber    CertificateSerialNumber     OPTIONAL,
      ….}
```

F.1.3.4 Certificate Formats

Certificate contents and encoding are outside the scope of this specification. Examples are given for illustration only. Certificates may contain various data objects and elements as defined in [ISO 7816-4], [ISO 7816-6], and ISO 7816-8 (using application tagging), [X.509] (universal and context-specific tagging) and elsewhere. The data objects included are defined by the certificate issuer (CA).

The following table identifies some data items that appear in certificates – 'presence' indicates when the item can be expected to be present:

Table F-2: Example of Data Included in Certificates

Name as used in [X.509]	Name as used in ISO 7816-8	Application Tag assigned by ISO 7816	Presence
Issuer	Issuer Identification Number	'42' (7816-8)	Always
Subject	Certificate holder reference OR Cardholder name	'5F20' (7816-8)	Always
SubjectPublicKey	Cardholder public key	'5F49' or '7F49' (7816-8)	Always
KeyUsage	Certificate holder authorization	'5F4C' (7816-9)	As required
CertificateSerialNumber	Certificate serial number	-	Always
n/a	Certificate contents	'5F4E' (7816-8)	Non-self descriptive certificates
Signature	Static internal authorization OR Digital signature	'9E' (7816-8)	Always
n/a	Public key remainder	'5F38' ([ISO 7816-6])	Signature scheme with recovery

Certificates may either be self descriptive, where the signature is across individual data objects; or non-self descriptive, where the signature is across concatenated value fields, without tags and lengths. Whether the certificate to be verified by the Security Domain is self-descriptive or non-self descriptive must be indicated in the PERFORM SECURITY OPERATION [verify certificate] command.

A Security Domain is only required to handle a single certificate format throughout a chain of certificates to be verified by the Security Domain.

Tag '67' in Card Recognition Data or Security Domain Management Data may contain one or more OIDs identifying the Security Domain's Trust Point's certification policy and/or the format of certificates that can be verified by the Security Domain and/or the format of certificates that can be retrieved from the Security Domain. They may also identify the cryptographic algorithms used for certificates, unless they are indicated in the certificates themselves.

F.1.3.4.1 Certificate without Message Recovery

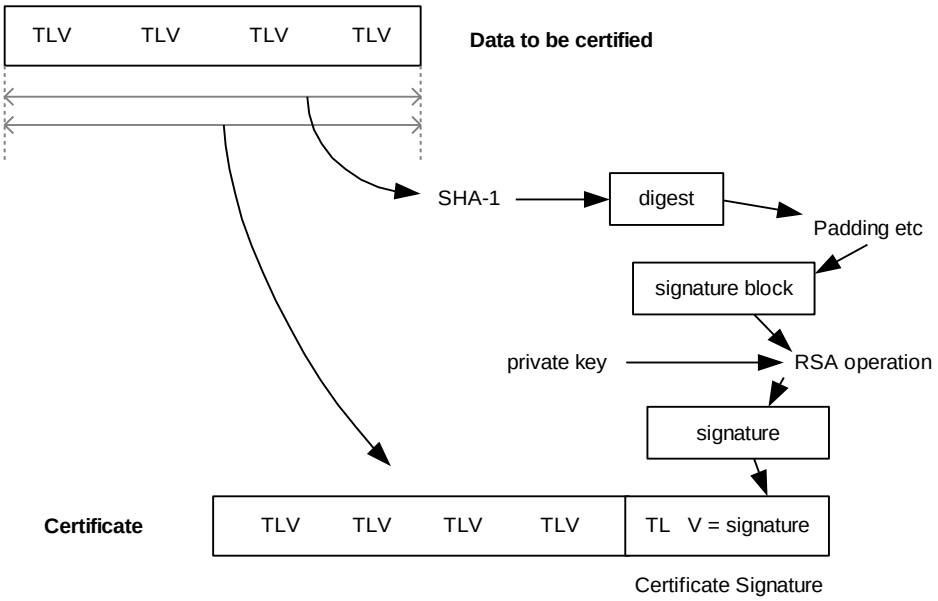
The data and public key to be certified are concatenated and a digest is created. A signature block is then prepared containing the digest and any necessary padding, constant and random data. The format of the signature block is defined by the certificate issuer (CA). The signature block size depends on the asymmetric algorithm and the certifying key size. The signature block is then signed using the certificate issuer's private certifying key, to form the value field of the Certificate Signature data object. The result is a certificate typically containing the data and public key to be certified and the Certificate Signature.

Certificate without Message Recovery, Self Descriptive Certificate

The data objects to be certified, including the public key data object, are TLV encoded and are concatenated TLV-encoded before the digest is created and the signature block prepared. The result of the signing operation is the Certificate Signature. The certificate is TLV encoded as a series of data objects, containing all the data objects to be certified, including the public key data object and the Certificate Signature data object. X.509 certificates fall into this category.

The formation of the certificate is shown in the following example:

Figure F-5: Certificate Formation – Self Descriptive Certificate without Message Recovery

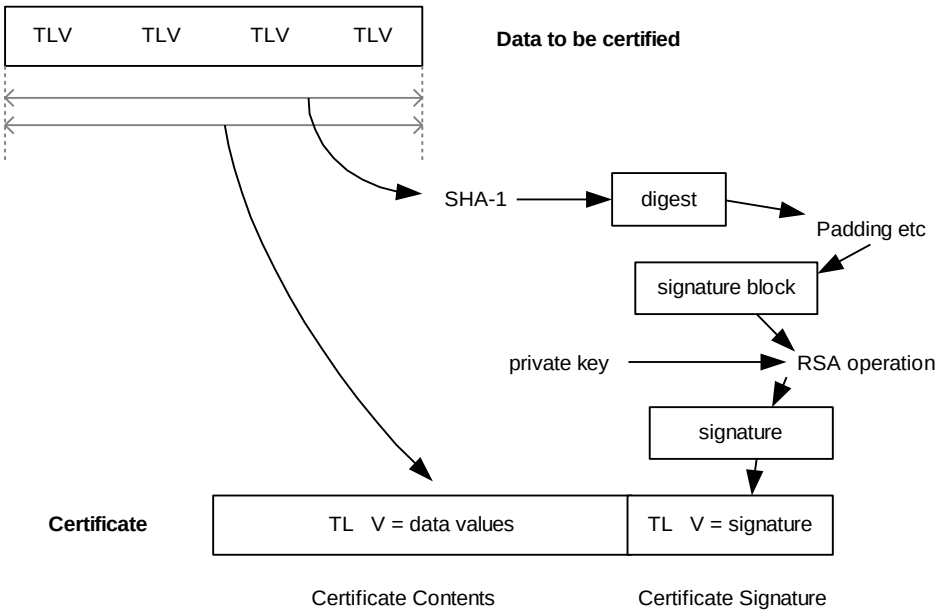


Certificate without Message Recovery, Non-Self Descriptive Certificate

The data and public key to be certified are a concatenated set of value fields. The format of the value fields included is defined by the certificate issuer (CA) and implicitly known to the recipients. A digest of the concatenated value fields is created and the signature block prepared. The result of the signing operation is the Certificate Signature. The certificate contains two data objects: the Certificate Contents, whose value is the concatenated set of value fields being certified, and the Certificate Signature.

The formation of the certificate is shown in the following example:

Figure F-6: Certificate Formation – Non-Self Descriptive Certificate without Message Recovery



F.1.3.4.2 Certificate with Message Recovery

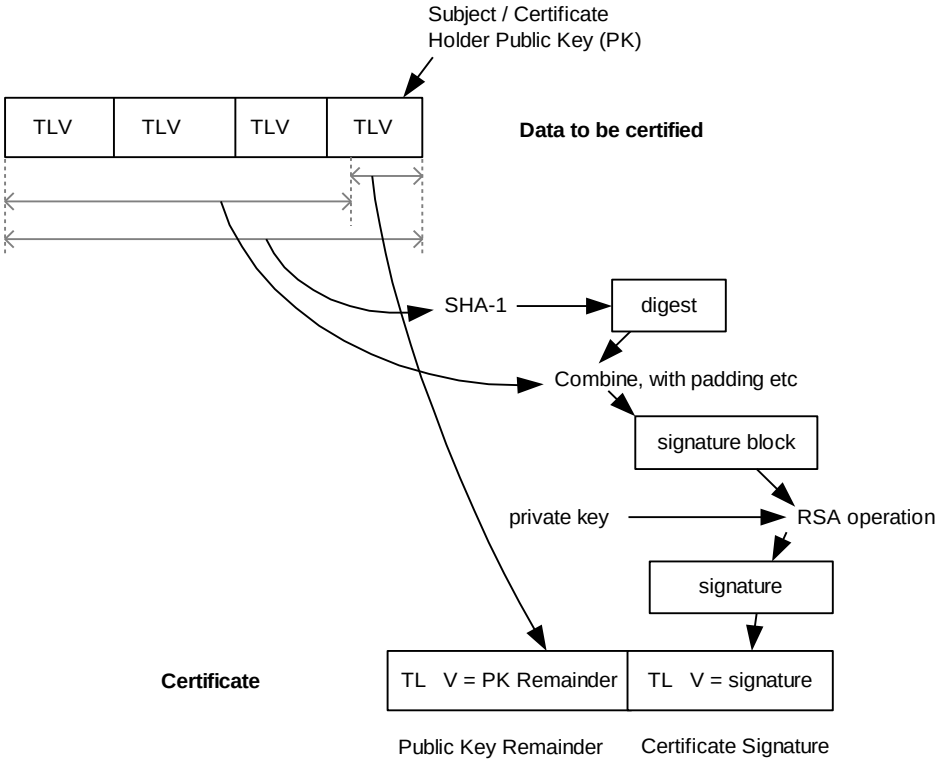
The data and public key to be certified is concatenated and a digest is created. A signature block is then prepared containing the digest, as much of the concatenated set of data and public key to be certified as can be included in the block and any necessary padding, constant and random data. The format of the signature block is defined by the certificate issuer (CA). The signature block size depends on the asymmetric algorithm and the certifying key size. The signature block is then signed using the certificate issuer's private certifying key, to form the value field of the Certificate Signature data object. Any part of the public key that could not be included in the Certificate Signature is then included in the value field of a separate Public Key Remainder data object. The result is a certificate typically containing two data objects: the Public Key Remainder and the Certificate Signature.

Certificate with Message Recovery, Self Descriptive Certificate

The data objects to be certified, including the public key data object, are TLV encoded and are concatenated TLV-encoded before the digest is created and the signature block prepared. The result of the signing operation is the Certificate Signature.

The formation of the certificate is shown in the following diagram:

Figure F-7: Certificate Formation – Self Descriptive Certificate with Message Recovery

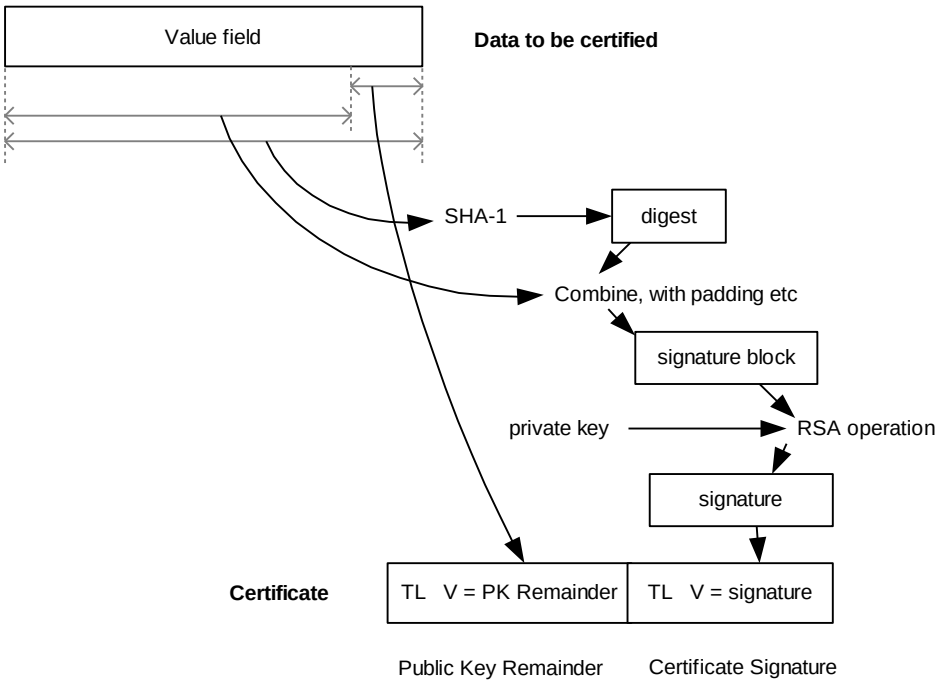


Certificate with Message Recovery, Non-Self Descriptive Certificate

The data and public key to be certified is a concatenated set of value fields. The format of the value fields included is defined by the certificate issuer (CA) and implicitly known to the recipients. A digest of the concatenated value fields is created and the signature block prepared. The result of the signing operation is the Certificate Signature.

The formation of the certificate is shown in the following diagram:

Figure F-8: Certificate Formation – Non-Self Descriptive Certificate with Message Recovery



F.1.4 Entity Authentication

F.1.4.1 Overview of Entity Authentication

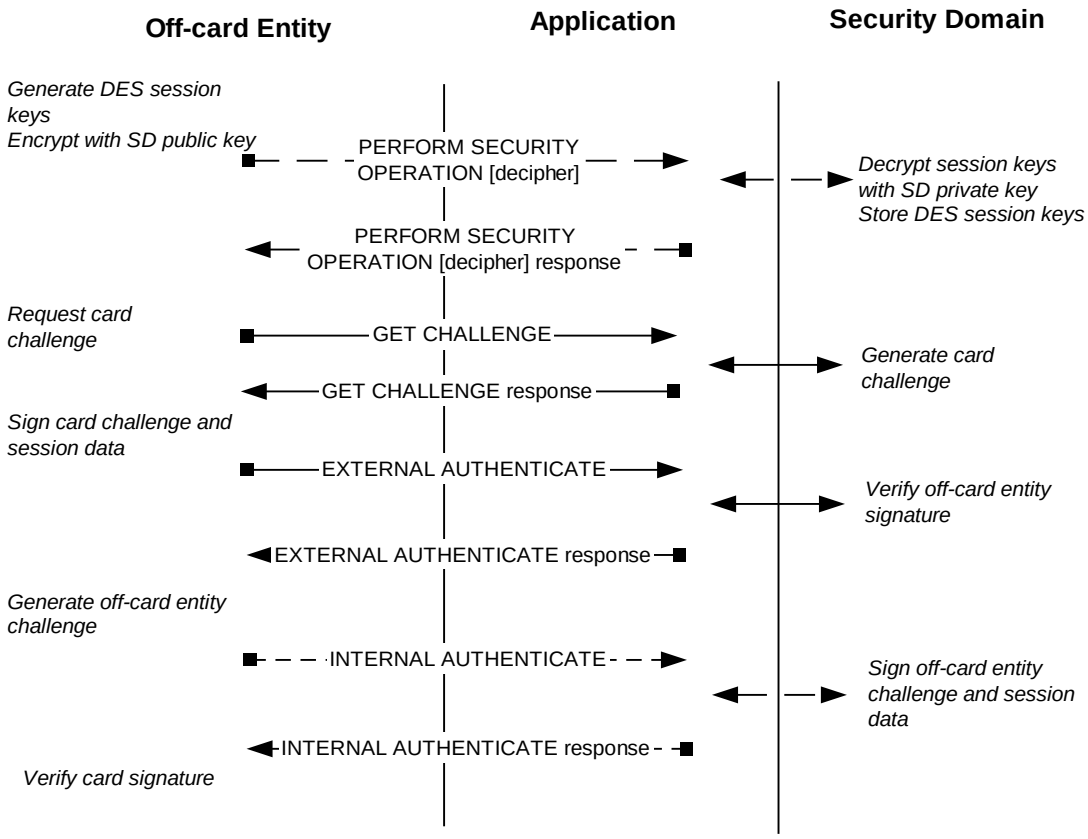
Once the parties have validated each other's public key, the Security Domain shall authenticate the Off-Card Entity using a challenge / response mechanism, and the Off-Card Entity may also authenticate the Security Domain in the same way.

Entity authentication of the Security Domain is optional, at the discretion of the Off-Card Entity. However, if the INTERNAL AUTHENTICATE command is required for use in session key agreement, then authentication of the Security Domain is performed.

F.1.4.2 Entity Authentication Process Flow

The following diagram gives an overview of the flow for Entity Authentication.

Figure F-9: Entity Authentication Flow



The following commands shall be supported:

- PERFORM SECURITY OPERATION [decipher] command, see section F.4.6 for further details;
- GET CHALLENGE command, see section F.4.2 for further details;
- EXTERNAL AUTHENTICATE command, see section F.4.1 for further details;
- INTERNAL AUTHENTICATE command, see section F.4.4 for further details.

The PERFORM SECURITY OPERATION [decipher] command is optional but shall be executed when value '02' of parameter “i” is supported – see section F.1.1.

The successful processing of the EXTERNAL AUTHENTICATE command (i.e. successful verification of the Off-Card Entity’s signature) shall set both the Current Security Level and Session Security Level according to the rules described in section F.1.5.2 – *Security Level Establishment*. The failed processing of the EXTERNAL AUTHENTICATE command (i.e. unsuccessful verification of the Off-Card Entity’s signature) shall reset both the Current Security Level and Session Security Level to NO_SECURITY_LEVEL and all the public keys previously verified within the current initiation phase shall be discarded.

The INTERNAL AUTHENTICATE command is optional for the key transport option and mandatory for key agreement. The INTERNAL AUTHENTICATE command shall only be received and processed once during Secure Channel Session initiation. Any error in the sequence flow or signing operation must abort the current Secure Channel Session: both the Current Security Level and Session Security Level shall be reset to NO_SECURITY_LEVEL.

F.1.4.3 Security Domain Authentication

The format and contents of Security Domain signature vary depending on the options chosen, as follows:

Key transport, signature without message recovery

The Security Domain signature is the result of generating a digest (hash) over a set of data, creating a signature block, and signing the signature block with the Security Domain private key (SK.SD.AUT). The data to be hashed and the contents of the signature block are as shown below.

Table F-3: Data to Hash

Name	Length	Value	Presence
Session Key(s)	n x (16 or 24)	'xxxx...'	Mandatory
Off-Card Entity challenge	16	'xxxx...'	Mandatory

Session Keys shall be in the same order as provided in the PERFORM SECURITY OPERATION [decipher] command; see sections F.3.1.2 and F.4.7.

Table F-4: Security Domain Signature Block

Name	Length	Value	Presence
Padding	2	'0001'	Mandatory
Padding ('FF')	8-n	'FF'...'FF'	Mandatory
Padding ('00')	1	'00'	Mandatory
DER encoded digest algorithm id - encoded as an object identifier	Variable	'xxxx...' (see section F.2.2)	Mandatory
DER encoded Hash (length and contents depend on the digest algorithm)	Variable	'xxxx...'	Mandatory

Key agreement, signature with message recovery

The Security Domain signature is the result of generating a digest (hash) over a set of data, creating a signature block, and signing the signature block with the Security Domain private key (SK.SD.AUT). The data to be hashed and the contents of the signature block are as shown below.

Table F-5: Data to Hash

Name	Length	Value	Presence
Random Padding (RP)	1-n	Same value as RP in Table F-6	Mandatory
Card Secret (CS)	32	Same value as CS in Table F-6	Mandatory
Off-Card Entity challenge	8	'xxxx...'	Mandatory
Off-Card Entity id	8	'xxxx...' (part of CERT.OCE.AUT)	Mandatory

Table F-6: Security Domain Signature Block

Name	Length	Value	Presence
Padding	1	'6A'	Mandatory
Random Padding (RP)	1-n	Same value as RP in Table F-5	Mandatory
Card Secret (CS)	32	Same value as CS in Table F-5	Mandatory
Hash	20	'xxxx...'	Mandatory
Padding	1	'BC'	Mandatory

The Security Domain signature is encrypted to ensure that the Card Secret is not divulged. To do this, the minimum of the values SIG.SD.AUT and (N.PK.SD.AUT – SIG.SD.AUT) is encrypted with the Off-Card Entity public key (PK.OCE.AUT), where N.PK.SD.AUT denotes the modulus of the Security Domain public key. This ensures that the data to be encrypted is always smaller than the modulus of the Off-Card Entity public key. Note that the modulus of the Security Domain public key and modulus of the Off-Card Entity public key must have the same length in bits. See [ISO 9796-2], Digital Signature scheme 1.

F.1.4.4 Off-Card Entity Authentication

The format and contents of Off-Card Entity signature vary depending on the option chosen, as follows:

Key transport, signature without message recovery

The Off-Card Entity signature is the result of generating a digest (hash) over a set of data, creating a signature block, and signing the signature block with the Off-Card Entity private key (SK.OCE.AUT). The data to be hashed and the contents of the signature block are as shown below.

Table F-7: Data to Hash

Name	Length	Value	Presence
Tag of Security Level	1	'D3'	Mandatory
Length of Security Level	1	'01'	Mandatory
Security Level	1	'xx'	Mandatory
Control reference template (CRT) tag	1	'B4' or 'B8'	Mandatory
Length of CRT	1	'00' - '7F'	Mandatory
CRT for Session Key(s)	n	'xxxx...'	Mandatory
…	…	…	…
CRT tag	1	'B4' or 'B8'	Optional
Length of CRT	1	'00' - '7F'	Conditional
CRT for Session Key(s)	n	'xxxx...'	Conditional
Card challenge	16		Mandatory

The control reference templates (CRTs) include session keys, and must be in the same order as provided in the PERFORM SECURITY OPERATION [decipher] command – see Table F-32.

Table F-8: Off-Card Entity Signature Block

Name	Length	Value	Presence
Padding ('0001')	2	'0001'	Mandatory
Padding ('FF')	8-n	'FF'	Mandatory
Padding ('00')	1	'00'	Mandatory
DER encoded digest algorithm id (encoded as an object identifier)	Variable	'xxxx...' (see section F.2.2)	Mandatory
DER encoded Hash (length and contents depend on the digest algorithm)	Variable	'xxxx...'	Mandatory

Control reference templates (CRTs) include session keys, and must be in the same order as provided in the PERFORM SECURITY OPERATION [decipher] command.

The format and contents of Card Signature vary depending on the options chosen, as discussed below.

Key agreement, signature with message recovery

The Off-Card Entity signature is the result of generating a digest (hash) over a set of data, creating a signature block, and signing the signature block with the Off-Card Entity private key (SK.OCE.AUT). The data to be hashed and the contents of the signature block are as shown below.

Table F-9: Data to Hash

Name	Length	Value	Presence
Random Padding (RPD)	1-n	Same value as RPD in Table F-10	Mandatory
Tag of Security Level	1	'D3'	Mandatory
Length of Security Level	1	'01'	Mandatory
Security Level	1	'xx'	Mandatory
CRT tag	1	'B4' or 'B8'	Mandatory
Length of CRT	1	'00' - '7F'	Mandatory
CRT for session key(s)	n	'xxxx...'	Mandatory
…	…	…	…
CRT tag	1	'B4' or 'B8'	Optional
Length of CRT	1	'00' - '7F'	Conditional
CRT for Session Key(s)	n	'xxxx...'	Conditional
Off-Card Entity Secret (OES)	32	Same value as OES in Table F-10	Mandatory
Card challenge	8	'xxxx...'	Mandatory
Card id: TBD (part of CERT.SD.AUT)	8	'xxxx...'	Mandatory

Control reference templates (CRTs) exclude session keys, and must be in the same order as provided in signature block.

Table F-10: Off-Card Entity Signature Block

Name	Length	Value	Presence
Padding	1	'6A'	Mandatory
Random Padding (RPD)	1-n	Same value as RPD in Table F-9	Mandatory
Tag of Security Level	1	'D3'	Mandatory
Length of Security Level	1	'01'	Mandatory
Security Level	1	'xx'	Mandatory
CRT tag	1	'B4' or 'B8'	Mandatory
Length of CRT	1	'00' - '7F'	Mandatory
CRT for session key(s)	1-n	'xxxx...'	Mandatory
…	…	…	…
CRT tag	1	'B4' or 'B8'	Optional
Length of CRT	1	'00' - '7F'	Conditional
CRT for Session Key(s)	n	'xxxx...'	Conditional
Off-Card Entity Secret (OES)	32	Same value as OES in Table F-9	Mandatory
Hash	20	'xxxx...'	Mandatory
Padding	1	'BC'	Mandatory

Control reference templates (CRTs) exclude session keys, and must be in the same order as input to the hash function.

The Off-Card Entity signature is encrypted to ensure that the Off-Card Entity secret is not divulged. To do this, the minimum of the values SIG.OCE.AUT and (N.PK.OCE.AUT – SIG.OCE.AUT) is encrypted using the Security Domain public key (PK.SD.AUT). N.PK.OCE.AUT denotes the modulus of the Off-Card Entity public key. This ensures that the data to be encrypted is always smaller than the modulus of the Security Domain public key. Note that the modulus of the Security Domain public key and modulus of the Off-Card Entity public key must have the same length in bits. See [ISO 9796-2], Digital Signature scheme 1.

F.1.5 Session Key and Security Level Establishment

When using the key transport option, Entity Authentication is preceded by the session keys and requested Security Level being sent to the Security Domain using the PERFORM SECURITY OPERATION [decipher] command; the Security Domain stores them until session initiation is complete.

A Security Domain supporting the key transport option shall decrypt with its private key (SK.SD.AUT or if the Security Domain has more than one key pair, the private key identified in the MANAGE SECURITY ENVIRONMENT command) the command data field of the PERFORM SECURITY OPERATION [decipher] command. Any failure in the decryption operation aborts the current Secure Channel Session initiation phase, and any public keys validated during that initiation phase shall be discarded.

In the key transport option the Secure Channel Session is established after successful processing of the EXTERNAL AUTHENTICATE command. An INTERNAL AUTHENTICATE command can be issued immediately after the EXTERNAL AUTHENTICATE command without secure messaging.

In the key agreement option the Secure Channel Session is established after successful processing of the EXTERNAL AUTHENTICATE and INTERNAL AUTHENTICATE commands.

F.1.5.1 Session Key Establishment

The Off-Card Entity supplies the Security Domain with details of what session keys are to be established. This information is in the form of 'control reference templates' (see section F.3.1.2) which are supplied either in the PERFORM SECURITY OPERATION [decipher] command (with the key transport option) or in the EXTERNAL AUTHENTICATE command (with the key agreement option).

If the key transport option is used, then the session keys are provided by the Off-Card Entity within the 'control reference templates' (see section F.3.1.2).

If the key agreement option is used, then the secrets exchanged between the Off-Card Entity and the Security Domain during the Entity Authentication process are used to establish session keys, as defined in section F.3.1 – *DES Session Keys*.

Once session keys have been established successfully, ICV sequence counter(s), used for secure messaging on subsequent commands and responses, are initialized as described in section F.3.2 – *Secure Messaging*.

F.1.5.2 Security Level Establishment

The requested Security Level is supplied in the PERFORM SECURITY OPERATION [decipher] command (with the key transport option) or in the EXTERNAL AUTHENTICATE command (with the key agreement option).

The successful initiation of a Secure Channel Session shall set the Current Security Level and Session Security Level to the requested Security Level combined with the AUTHENTICATED or ANY_AUTHENTICATED indicator (see section 10.4.2 – *Authentication with Asymmetric Cryptography* for further details). If the requested Security Level is set to zero, the successful initiation of a Secure Channel Session shall set the Current Security Level and Session Security Level to AUTHENTICATED or ANY_AUTHENTICATED only.

F.1.6 Protocol Rules

The Current Security Level of a communication not included in a Secure Channel Session shall be set to NO_SECURITY_LEVEL. In accordance with the general rules described in Chapter 10 – *Secure Communication*, the following rules shall apply:

- The successful initiation of a Secure Channel Session shall set the Current Security Level to the requested Security Level from the selected Application's perspective: it is at least set to AUTHENTICATED or ANY_AUTHENTICATED (see section 10.4.2 – *Authentication with Asymmetric Cryptography* for details;
- The Current Security Level shall apply to the entire Secure Channel Session unless successfully modified at the request of the Application;
- When the Current Security Level is set to NO_SECURITY_LEVEL, then:
  - If the Secure Channel Session was aborted during the same Application Session, the incoming command shall be rejected with a security error;
  - Otherwise no security verification of the incoming command shall be performed. The Application processing the command is responsible for applying its own security rules.
- If a Secure Channel Session is active for incoming commands (i.e. Current Security Level at least set to either AUTHENTICATED or ANY_AUTHENTICATED), the security of the incoming command shall be checked according to the Current Security Level, or if the APDU class byte indicates Secure Messaging and Secure Messaging data objects are present in the command data field:
  - When the security of the command does not match or exceed the Current Security Level, the command shall be rejected with a security error, the Secure Channel Session aborted and the Current Security Level reset to NO_SECURITY_LEVEL;

  - If a security error is found, the command shall be rejected with a security error, the Secure Channel Session aborted and the Current Security Level reset to NO_SECURITY_LEVEL;
  - If (one of) the appropriate session key(s) is not available, the command shall be rejected with a security error, the Secure Channel Session aborted and the Current Security Level reset to NO_SECURITY_LEVEL;
  - In all other cases, the Secure Channel Session shall remain active and the Current Security Level shall reflect the level of security established by the current command (e.g. C-MAC and/or C-ENCRYPTION). The Application is responsible for further processing the command.
- If a Secure Channel Session is active for outgoing responses (i.e. Current Security Level at least set to AUTHENTICATED or ANY_AUTHENTICATED), secure messaging protection shall be applied to the outgoing response according to the Current Security Level (i.e. R-MAC and/or R-ENCRYPTION):
  - If a cryptographic error occurs, a security error shall be returned, the Secure Channel Session aborted and the Current Security Level reset to NO_SECURITY_LEVEL;
  - If (one of) the appropriate session key(s) is not available, a security error shall be returned, the Secure Channel Session aborted and the Current Security Level reset to NO_SECURITY_LEVEL;
  - Otherwise, the Secure Channel Session shall remain active and the Current Security Level unmodified.
- If a Secure Channel Session is aborted, it is still considered not terminated;
- If the Security Domain supports application data encryption and/or decryption, it shall decrypt or encrypt a block of secret data upon request. If the service is not supported or if (one of) the appropriate cryptographic key(s) is not available, the request shall be rejected but the Current Security Level, Session Security Level and Secure Channel Session in operation (if any) shall not be impacted;
- The current Secure Channel Session shall be terminated (if aborted or still open), both the Current Security Level and Session Security Level reset to NO_SECURITY_LEVEL on either:
  - Attempt to initiate a new Secure Channel Session;
  - Termination of the Application Session (e.g. new Application selection);
  - Termination of the associated logical channel;
  - Termination of the Card Session (card reset or power off);
  - Explicit termination by the Application (e.g. invoking GlobalPlatform API).

F.2 Cryptographic Algorithms

The cryptographic and hashing algorithms described in Appendix B – *Algorithms (Cryptographic and Hashing)* apply to SCP10. This section defines the additional requirements for SCP10.

F.2.1 Asymmetric Cryptography

In this appendix, signing means the deciphering of a signature block using the signer's private RSA key.

For certificates to be verified by the card, and message signatures signed and verified by the card, the cryptographic scheme shall be RSA. The signature block and signature are the same length as the key modulus.

In Entity Authentication, Digital Signature Scheme 1 in [ISO 9796-2] shall be used for signature with message recovery, and the signature scheme with appendix RSASSA-PKCS1-V1_5 in [PKCS#1] for signature without message recovery.

The details of the signature scheme for certificates shall be either implicitly known by the Off-Card Entity or specified by the Security Domain Trust Point through the contents of Card Recognition Data or Security Domain Management Data in tag '67'.

In the key transport option, the cryptographic scheme for encrypting the session keys and their CRT templates shall be RSA according to the encryption scheme RSAES-PKCS1-v1_5 as defined in [PKCS#1].

F.2.2 Digest Algorithm

The default digest algorithm for use in conjunction with SCP10 asymmetric cryptography in Entity Authentication shall be SHA-1 for this version of the Specification. An alternative algorithm may be specified in Card Recognition Data or Security Domain Management Data in tag '67'.

The Object Identifier for SHA-1 is:

```
{iso(1) identified-organization(3) oiw(14) secsig(3) algorithms(2) 26}
```

which is DER-TLV encoded as '2B 0E 03 02 1A'.

F.2.3 Message Integrity ICV

The ICV for the each C-MAC and R-MAC calculation is obtained by enciphering an ICV sequence counter. The ICV sequence counter is initialized during Secure Channel initiation to either:

- the value supplied in tag '91' of the control reference template, for the key transport option (separate initial value for each key), or
- the concatenation of the last 4 bytes of the card secret and the last 4 bytes of Off-Card Entity secret, for the key agreement option (same initial value for all keys).

The ICV sequence counter is incremented by one for each C-MAC and R-MAC. For calculating a C-MAC ICV, the ICV sequence counter is single-DES enciphered using the first part of the Secure Channel C-MAC key. For calculating an R-MAC ICV, the ICV sequence counter is single-DES enciphered using the first part of the Secure Channel R-MAC key.

F.2.4 Message Integrity C-MAC and R-MAC

Message integrity is achieved by applying a MAC to message data. The MAC may be:

- C-MAC for APDU command messages (generated by the Off-Card Entity);
- R-MAC for APDU response messages (generated by the card).

The receiving entity, on receipt of the message containing a MAC, using the same session key, performs the same operation and by comparing its generated MAC with the MAC received from the sending entity is assured of the integrity of the full command or response.

The integrity of the sequence of APDU command or response messages being transmitted to the receiving entity is achieved by using an encrypted sequence counter as part of the MAC generation. This ensures the receiving entity that all messages in a sequence have been received.

F.2.5 APDU Encryption and Decryption for Message Confidentiality

Message confidentiality is achieved by encrypting the whole of the command or response data field. This includes any data within the data field that has already been protected for another purpose, such as secret or private keys encrypted with the data encryption key.

F.3 Cryptographic Usage

F.3.1 DES Session Keys

F.3.1.1 Overview

All session keys shall be double or triple length DES keys.

The Off-Card Entity supplies information on cryptographic keys to be established for the session in a set of control reference templates. Each control reference template specifies the usage of the key.

In the case of key transport, the templates are supplied in the PERFORM SECURITY OPERATION [decipher] command and contain the actual key values.

In the case of key agreement, the templates are supplied in the EXTERNAL AUTHENTICATE command, and do not contain the key values.

F.3.1.2 Control Reference Templates

A control reference template (CRT) is structured as follows:

Table F-11: Single CRT

Tag	Length	Name	Presence
'B4' or 'B8'	'00' - '7F'	CRT tag = 'B4' (CCT) or 'B8' (CT)	Mandatory
'95'	1	Key Usage Qualifier = '10' (secure messaging for commands) '20' (secure messaging for responses) '30' (secure messaging for commands and responses) '40' (encipherment of sensitive data in responses) '80' (encipherment of sensitive data in commands) 'C0' (encipherment of sensitive data in commands & responses)	Mandatory
'80'	0 or 1	Optional cryptographic mechanism. Contains Key Type, coded according to Table 11-16	Optional
'D1'	0, 16, or 24	Off-Card Entity Session Key	Conditional
'91'	0 or 8	Initial value of sequence counter, for use in secure messaging	Conditional

Note: Key Usage Qualifier and the CRT tag together define the key usage, which is C-MAC and/or R-MAC for control reference template 'B4', and C-ENC and/or R-ENC or DEK for control reference template 'B8'.

The Key Type identifies the cryptographic algorithm.

The Off-Card Entity Session Key and sequence counter data objects are only present with the key transport option.

The sequence counter data object is used in secure messaging as the initial value for the ICV sequence counter for this key, which is encrypted as defined in section F.2.3 – *Message Integrity ICV* to derive the ICV for the first MAC generated using this key.

F.3.1.3 Session Key Derivation

With the key agreement option, session keys are derived from the secret data exchanged during Secure Channel initiation as follows:

- The two 32-byte secrets Off-Card Entity Secret and Card Secret are exclusive or-ed, giving result (1);
- A 32-byte binary counter is set to a value depending on the key usage and its position in the set of CRTs supplied, as shown below;
- Result (1) is appended with a 32 bit binary counter with the appropriate value for this key, and the result is hashed using SHA-1, giving result (2);
- Bytes 1-16 of result (2) form the double-length DES session key.

Table F-12: Counter Value for Session Key Calculation

Counter Value	Key
1	The MAC key whose CRT is first in the set of CRTs supplied by the Off-Card Entity
2	The ENC key whose CRT is first in the set of CRTs supplied by the Off-Card Entity
3	A subsequent MAC key, if any
4	A subsequent ENC key, if any
5	The data encryption key whose CRT is first in the set of CRTs supplied by the Off-Card Entity
6	A subsequent data encryption key, if any

F.3.2 Secure Messaging

F.3.2.1 APDU Command C-MAC Protection

This section applies where command integrity (C-MAC) is required but not command confidentiality (C-ENC).

A C-MAC is generated by an Off-Card Entity and applied across the full APDU command being transmitted to the card including the header, the command data field (if present) and Le (if present). Input data to the MAC calculation is first prepared as defined in [ISO 7816-4]:

- The following data is concatenated:
  - The command header CLA, INS, P1, P2 from the unprotected APDU, with the logical channel bits in the CLA byte set to zero, and appended with four bytes '80 00 00 00';
  - If a command data field is present in the unprotected APDU, a BER-TLV data object with tag '81' containing the complete original command data field, regardless of its contents and format;
  - If Le is present in the unprotected APDU, a BER-TLV data object with tag '97' containing the original Le value;
- DES padding is applied as defined in section B.1.3.

A C-MAC is generated using the Secure Channel C-MAC session key, the encrypted sequence counter as the ICV as defined in section F.2.3, and the signature method described in section B.1.2.2 – *Single DES Plus Final Triple DES MAC* across the input data.

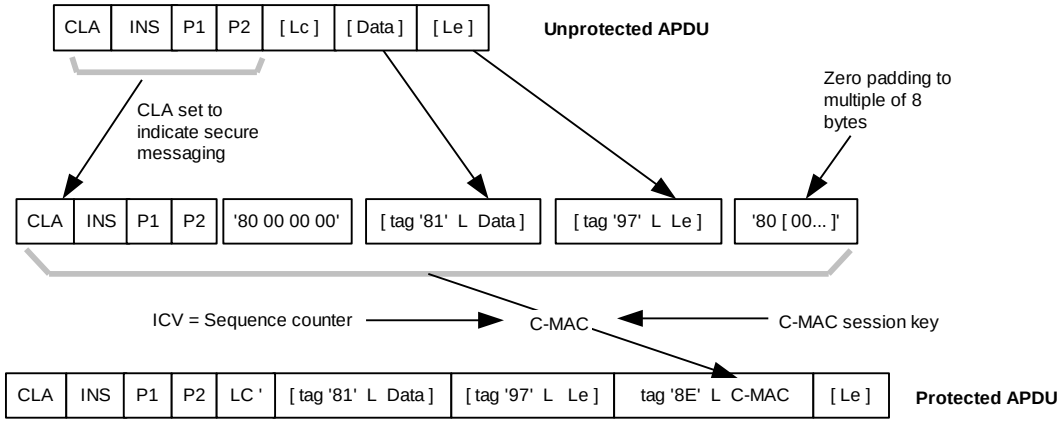
To reflect the presence of a C-MAC in the command message, the unprotected APDU shall be modified as follows:

- The class byte shall be modified to indicate that this APDU command includes secure messaging. This is achieved by setting to ‘11’ bits 4-3 of a class byte indicating a logical channel number 0 to 4 (unprotected CLA set to '00' - '03' or '80' - '83') or by setting to ‘1’ bit b6 of a class byte indicating a logical channel number 4 to 19 (unprotected CLA set to '40' - '4F' or 'C0' - 'CF'); see section 11.1.4. The logical channel bits are unchanged;
- The length of the command message (Lc) shall be incremented by:
  - 10 bytes to allow for the C-MAC data object, plus
  - 2 or more bytes to allow for the tag and length of the command data field data object (if command data is present - note: the length field may be longer than one byte), plus
  - 3 bytes to allow for the Le data object (if Le is present).
- The command data, if present in the unprotected APDU, shall be encapsulated in a BER-TLV data object with tag '81' and a length field coded according to [ISO 8825-1];
- The Le byte, if present in the unprotected APDU, shall be contained in a BER-TLV data object with tag '97';
- The C-MAC shall be encapsulated in a BER-TLV data object with tag '8E' and appended at the end of the command data field.

No padding is present in the transmitted APDU.

The following diagram shows the message reformatting that is performed by the Off-Card Entity when a command is protected for integrity.

Figure F-10: APDU C-MAC Generation



The card, in order to verify the C-MAC, shall perform the same procedure as employed by the Off-Card Entity in order to verify the C-MAC. The ICV sequence counter used in ICV calculation is then incremented. This is true regardless of whether the APDU processing completes successfully or not; i.e. a new sequence counter value shall always be used for the next C-MAC or R-MAC.

F.3.2.2 APDU Command C-ENC Protection

This section applies where command confidentiality (C-ENC) is required but not command integrity (C-MAC).

No encryption shall be applied to a command where there is no command data field: in this case the command message (header and optional Le) is sent without modification.

Otherwise the Off-Card Entity encrypts the command data field of the command message being transmitted to the card. This includes any data within the data field that has already been protected for another purpose; e.g. secret or private keys encrypted with the data encryption session key.

Prior to encrypting the data, DES padding is applied as defined in B.1.3.

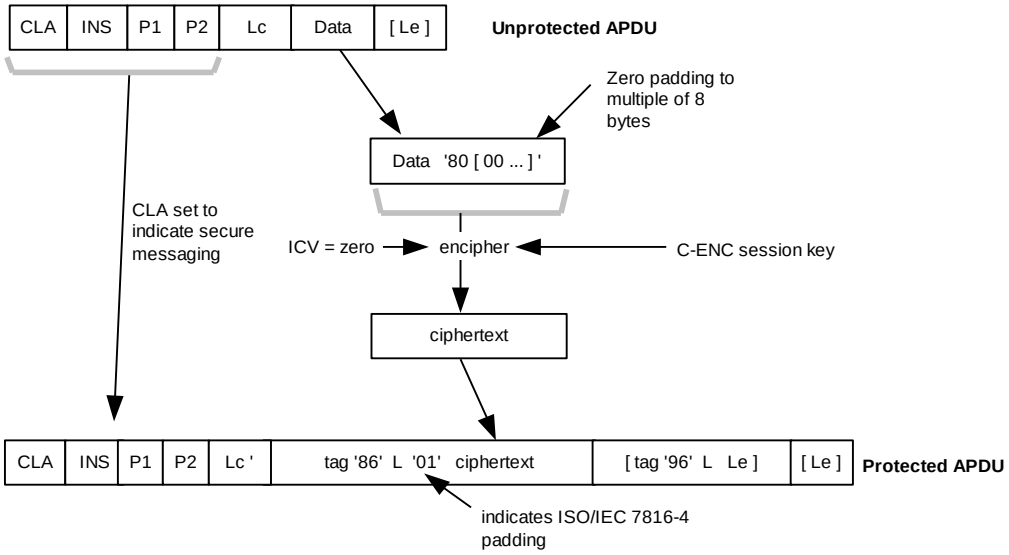
The padded command data field is enciphered using triple DES in CBC mode as defined in section B.1.1.1, the C-ENC session key established during the Secure Channel initiation process and an ICV of zero.

To reflect the C-ENC protection of the command, the unprotected APDU shall be modified as follows:

- The class byte shall be modified to indicate that this APDU command includes secure messaging. This is achieved by setting to ‘10’ bits 4-3 of a class byte indicating a logical channel number 0 to 4 (unprotected CLA set to '00' - '03' or '80' - '83') or by setting to ‘1’ bit b6 of a class byte indicating a logical channel number 4 to 19 (unprotected CLA set to '40' - '4F' or 'C0' - 'CF'); see section 11.1.4. The logical channel bits are unchanged;
- Lc shall be incremented by:
  - 4 or more bytes to allow for the tag and length of the command data field data object, the padding indicator and the variable padding (the length field may be longer than one byte), plus
  - 3 bytes to allow for the Le data object (if Le is present).
- The encrypted command data shall be preceded by the ISO/IEC 7816 padding indicator '01' and encapsulated in a BER-TLV data object with tag '86' and a length field coded according to [ISO 8825-1];
- The Le byte, if present in the unprotected APDU, shall be contained in a BER-TLV data object with tag '96'.

The following diagram shows the message reformatting that is performed by the Off-Card Entity when a command is protected for confidentiality.

Figure F-11: Secure Messaging: Command Message Protected for Confidentiality



F.3.2.3 APDU Command C-MAC and C-ENC Protection

This section applies where both command confidentiality (C-ENC) and integrity (C-MAC) are required.

No encryption shall be applied to a command where there is no command data field: in this case the message shall be protected as defined in section F.3.2.1 – *APDU Command C-MAC Protection*.

Otherwise the Off-Card Entity first encrypts the command data field of the command message being transmitted to the card as defined in section F.3.2.2.

A C-MAC is generated by an Off-Card Entity as defined in section F.3.2.1. Input data to the MAC calculation is first prepared as defined in [ISO 7816-4]:

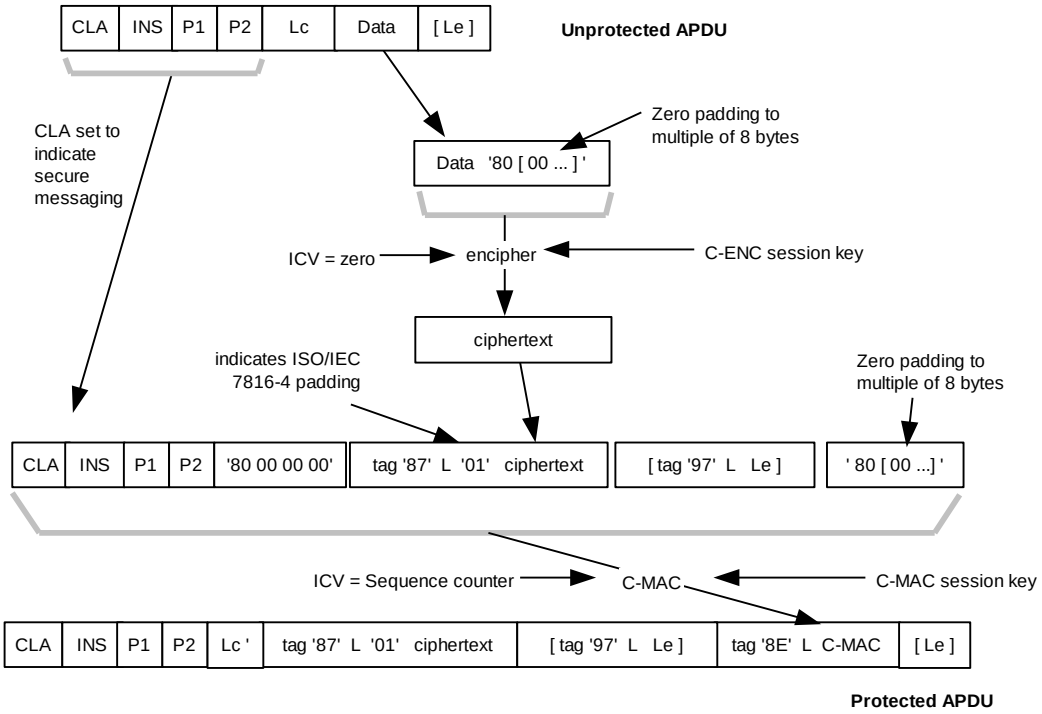
- The following data is concatenated:
  - The command header CLA, INS, P1, P2 from the unprotected APDU, with the logical channel bits in the CLA byte set to zero, appended with four bytes '80 00 00 00';
  - If a command data field is present in the unprotected APDU, a BER-TLV data object with tag '87' containing the ISO/IEC 7816 padding indicator '01' followed by the encrypted command data;
  - If Le is present in the unprotected APDU, a BER-TLV data object with tag '97' containing the original Le value;
- DES padding is applied as defined in section B.1.3.

To reflect the presence of a C-MAC and C-ENC protection of the command, the unprotected APDU shall be modified as follows:

- The class byte shall be modified to indicate that this APDU command includes secure messaging. This is achieved by setting to ‘11’ bits 4-3 of a class byte indicating a logical channel number 0 to 4 (unprotected CLA set to '00' - '03' or '80' - '83') or by setting to ‘1’ bit b6 of a class byte indicating a logical channel number 4 to 19 (unprotected CLA set to '40' - '4F' or 'C0' - 'CF'); see section 11.1.4. The logical channel bits are unchanged;
- Lc shall be incremented by:
  - 10 bytes to allow for the C-MAC data object, plus
  - 4 or more bytes to allow for the tag and length of the command data field data object, the padding indicator and the variable padding (the length field may be longer than one byte), plus
  - 3 bytes to allow for the Le data object (if Le is present).
- The encrypted command data shall be preceded by the ISO/IEC 7816 padding indicator '01' and encapsulated in a BER-TLV data object with tag '87' and a length field coded according to [ISO 8825-1];
- The Le byte, if present in the unprotected APDU, shall be contained in a BER-TLV data object with tag '97';
- The C-MAC shall be encapsulated in a BER-TLV data object with tag '8E' and appended at the end of the command data field.

The following diagram shows the message reformatting that is performed by the Off-Card Entity when a command is protected for integrity and confidentiality.

Figure F-12: Secure Messaging: Command Message Protected for Integrity and Confidentiality



F.3.2.4 APDU Response R-MAC Protection

This section applies where response integrity (R-MAC) is required but not confidentiality (R-ENC).

No R-MAC shall be generated and no protection shall be applied to a response where status bytes SW1 and SW2 indicate an error: in this case only status bytes shall be returned in the response.

When R-MAC protection is required for a case 1 or case 3 command, the card shall process the command as a case 2 or case 4 command respectively and treat Le as if it were present and set to zero.

An R-MAC is generated by the card across the response data field (if present) and status bytes. Input data to the MAC calculation is first prepared as defined in [ISO 7816-4]:

- The following data is concatenated:
  - If a response data field is present in the unprotected APDU, a BER-TLV data object with tag '81' containing the complete original response data field, regardless of its contents and format;
  - A BER-TLV data object with tag '99', containing the original status word SW1 - SW2 value.
- DES padding is applied as defined in section B.1.3.

An R-MAC is generated using the Secure Channel R-MAC session key, the encrypted sequence counter as the ICV as defined in section F.2.3, and the signature method described in section B.1.2.2 – *Single DES Plus Final Triple DES MAC* across the input data.

To reflect the presence of an R-MAC protection of the response, the unprotected APDU shall be modified as follows:

- The response data, if present in the unprotected APDU, shall be encapsulated in a BER-TLV data object with tag '81' and a length field coded according to [ISO 8825-1];

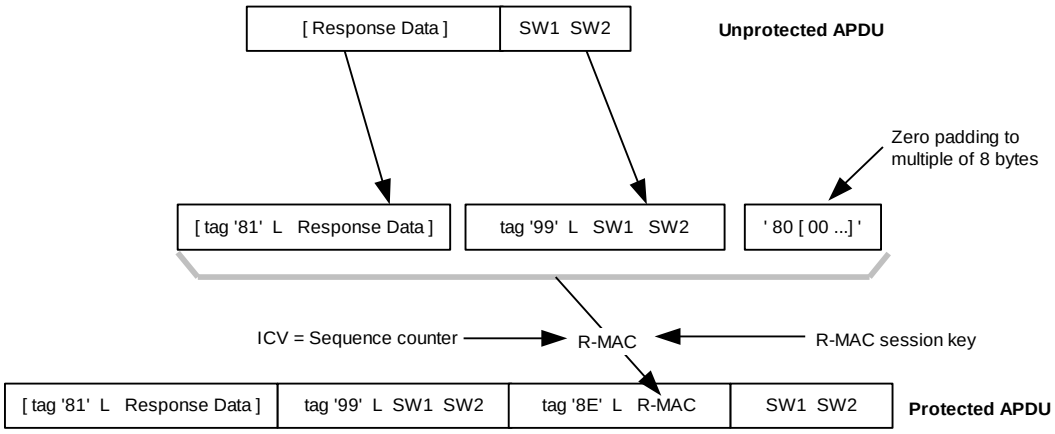
- The status word SW1 - SW2 of the unprotected APDU shall be contained in a BER-TLV data object with tag '99';
- The R-MAC shall be encapsulated in a BER-TLV data object with tag '8E' and appended at the end of the response data field.

No padding is present in the transmitted APDU.

The Off-Card Entity, in order to verify the R-MAC, shall perform the same processing in order to generate an R-MAC and compare it with the transmitted R-MAC.

The following diagram shows the message reformatting that is performed by the card when a response message is protected for integrity.

Figure F-13: Secure Messaging: Response Message Protected for Integrity



F.3.2.5 APDU Response R-ENC Protection

This section applies where response confidentiality (R-ENC) is required but not integrity (R-MAC).

No protection shall be applied to a response where status bytes SW1 and SW2 indicate an error or where there is no response data field: in this case only status bytes shall be returned in the response.

Otherwise, the Security Domain encrypts the response data field. This includes any data within the data field that has already been protected for another purpose, such as secret or private keys encrypted with the data encryption session key.

Prior to encrypting the response data field, DES padding is applied as defined in section B.1.3.

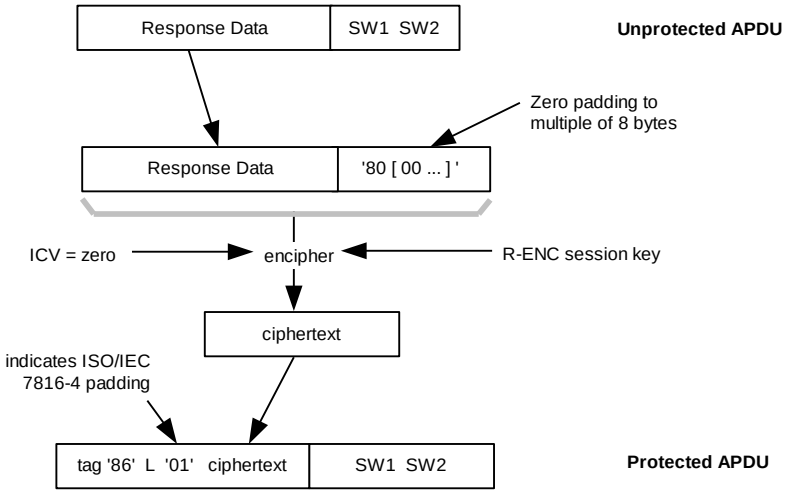
The padded response data field is then enciphered using triple DES in CBC mode as defined in section B.1.1.1 – *CBC Mode*, the R-ENC session key established during the Secure Channel initiation process and an ICV of zero.

To reflect the R-ENC protection of the response, the unprotected APDU shall be modified as follows:

- The encrypted response data shall be preceded by the ISO/IEC 7816 padding indicator '01' and encapsulated in a BER-TLV data object with tag '86' and a length field coded according to [ISO 8825-1].

The following diagram shows the message reformatting that is performed by the card when a response message is protected for confidentiality.

Figure F-14: Secure Messaging: Response Message Protected for Confidentiality



F.3.2.6 APDU Response R-MAC and R-ENC Protection

This section applies where both response confidentiality (R-ENC) and response integrity (R-MAC) are required.

No R-MAC or encryption shall be applied to a response where status bytes SW1 and SW2 indicate an error: in this case only status bytes shall be returned in the response.

No encryption shall be applied to a response where there is no response data field: in this case the message shall be protected as defined in section F.3.2.4 – *APDU Response R-MAC Protection*.

Otherwise, the card first encrypts the response data field of the response message being transmitted to the Off-Card Entity as defined in section F.3.2.5.

An R-MAC is then generated by the card as defined in section F.3.2.4. Input data to the MAC calculation is first prepared as defined in [ISO 7816-4]:

- The following data is concatenated:
  - If a response data field is present in the unprotected APDU, a BER-TLV data object with tag '87' containing the ISO/IEC 7816 padding indicator '01' followed by the encrypted response data;
  - A BER-TLV data object with tag '99' containing the original SW1 - SW2 status word value.
- DES padding is applied as defined in section B.1.3.

To reflect the presence of an R-MAC and R-ENC protection of the response, the unprotected APDU shall be modified as follows:

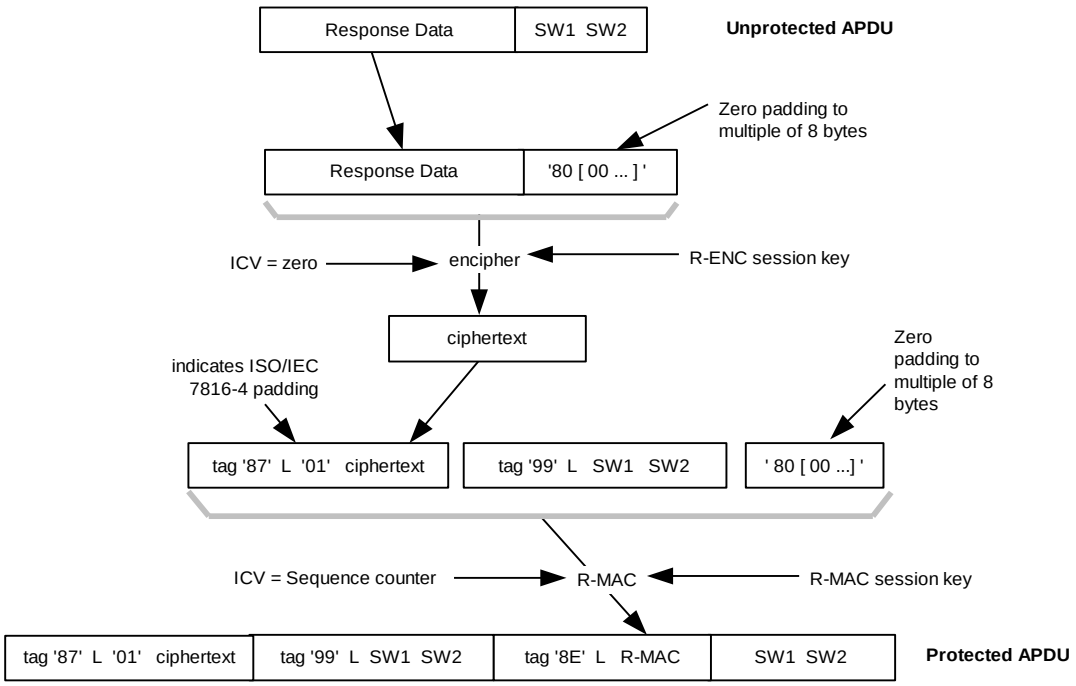
- The encrypted response data shall be preceded by the ISO/IEC 7816 padding indicator '01' and encapsulated in a BER-TLV data object with tag '87' and a length field coded according to [ISO 8825-1];
- The status word SW1 - SW2 of the unprotected APDU shall be contained in a BER-TLV data object with tag '99';
- The R-MAC shall be encapsulated in a BER-TLV data object with tag '8E' and appended at the end of the response data field.

No R-MAC padding is present in the transmitted APDU.

The Off-Card Entity, in order to verify the R-MAC, shall perform the same processing in order to generate an R-MAC and compare it with the transmitted R-MAC.

The following diagram shows the message reformatting that is performed by the card when a response message is protected for integrity and confidentiality.

Figure F-15: Secure Messaging: Response Message Protected for Integrity and Confidentiality



F.3.2.7 Sensitive Data Encryption and Decryption

Data encryption is used when transmitting sensitive data to and from the card. For instance all keys transmitted to a card (e.g. in a PUT KEY command) should be encrypted. Data encryption is over and beyond the Current Security Level required for the Secure Channel Session. The encryption process uses the relevant data encryption session key (DEK) for sensitive data in command messages or for sensitive data in response messages. The encryption method uses DES in ECB or CBC mode depending on the Key Type of the DEK Key in the CRT; see section B.1.1.1 – *CBC Mode* or section B.1.1.2 – *ECB Mode*. If the key type is omitted for the DEK Key it shall be known implicitly. The sensitive data block length shall be constructed as a multiple of 8-byte long block before the encryption operations: the eventual padding method is application specific.

The encryption is performed across the sensitive data and the result of each encryption becomes part of the encrypted data. This encrypted data becomes part of the clear text data field in the command/response message. The decryption is the exact opposite of the above operation: in particular, no padding is removed by the decryption operation.

F.4 Commands

Because certificates, digital signatures and some data fields can be long, command and response chaining as defined in [ISO 7816-4] is used to transfer successive data blocks.

With command chaining, the command data is sent in multiple APDUs, the command data being segmented arbitrarily. All except the final command in the chain shall indicate command chaining by setting to '1' bit 5 of the class byte according to [ISO 7816-4].

With response chaining, the response data is sent in multiple APDUs, the response data being segmented arbitrarily.

Table F-13: SCP10 Command Support

Command	Secure Channel Initiation	
	Signature Without Message Recovery	Signature With Message Recovery
EXTERNAL AUTHENTICATE	✓	✓
GET CHALLENGE	✓	✓
GET DATA [certificate]	✓	✓
INTERNAL AUTHENTICATE	✓	✓
MANAGE SECURITY ENVIRONMENT	✓	✓
PERFORM SECURITY OPERATION [decipher]	✓	
PERFORM SECURITY OPERATION [verify certificate]	✓	✓

The following table summarizes the minimum security requirements for the APDU commands.

Table F-14: Minimum Security Requirements for SCP10 commands

Command	Minimum Security
EXTERNAL AUTHENTICATE	Validated PK.OCE.AUT and card challenge
GET CHALLENGE	None
GET DATA [certificate]	None
INTERNAL AUTHENTICATE	Current Security Level is at least AUTHENTICATED or ANY_AUTHENTICATED
MANAGE SECURITY ENVIRONMENT	None
PERFORM SECURITY OPERATION [decipher]	None
PERFORM SECURITY OPERATION [verify certificate]	None

The following table provides the list of SCP10 command support per card Life Cycle State.

Table F-15: SCP10 Command Support per Card Life Cycle State

Command	OP_READY			INITIALIZED			SECURED			CARD_LOCKED	TERMINATED
	AM SD	DM SD	SD	AM SD	DM SD	SD	AM SD	DM SD	SD	SD	SD
EXTERNAL AUTHENTICATE	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
GET CHALLENGE	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
GET DATA [certificate]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
INTERNAL AUTHENTICATE	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
MANAGE SECURITY ENVIRONMENT	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
PERFORM SECURITY OPERATION [decipher]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
PERFORM SECURITY OPERATION [verify certificate]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	

Legend of Table F-13 and Table F-15

AM SD	Security Domain with Authorized Management privilege.
DM SD	Supplementary Security Domain with Delegated Management privilege.
SD	Other Security Domain.
✓	Support required.
Blank cell	Support optional.
Striped cell	Support prohibited.

F.4.1 EXTERNAL AUTHENTICATE Command

F.4.1.1 Definition and Scope

This command is used to authenticate the Off-Card Entity by the Security Domain. This command is also used with the key agreement option to support session key establishment. It shall be immediately preceded (on the same logical channel of the same card I/O interface) by a GET CHALLENGE command. It may be followed (on the same logical channel of the same card I/O interface) by an INTERNAL AUTHENTICATE command.

F.4.1.2 Command Message

The EXTERNAL AUTHENTICATE command message is coded as follows:

Table F-16: EXTERNAL AUTHENTICATE Command Message

Code	Value	Meaning
CLA	'00' - '03', '40' - '4F', '10' - '13', or '50' - '5F'	See section 11.1.4
INS	'82'	EXTERNAL AUTHENTICATE
P1	'00'	Reference control parameter P1: no information given
P2	'00'	Reference control parameter P2: no information given
Lc	'xx'	Length of Entity Signature (key transport) or encrypted Off-Card Entity Signature (key agreement)
Data	'xx xx…'	Command data field
Le	-	Not present

A Security Domain may support other values of Reference Control Parameters P1 and P2 as defined in [ISO 7816-4].

F.4.1.3 Data Field Sent in the Command Message

The data field of the EXTERNAL AUTHENTICATE command message contains the Off-Card Entity Signature (key transport) or encrypted Off-Card Entity Signature (key agreement) – see section F.1.4.4 – *Off-Card Entity Authentication* for further details.

F.4.1.4 Data Field Returned in the Response Message

No data is returned by this command.

F.4.1.5 Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

This command may return either a general error condition as listed in section 11.1.3 – *General Error Conditions* or one of the following specific errors and warning conditions.

Table F-17: Error Conditions

SW1	SW2	Meaning
'63'	'00'	Verification of certificate failed
'94'	'84'	Algorithm not supported

F.4.2 GET CHALLENGE Command

F.4.2.1 Definition and Scope

This command is used to obtain a random challenge from the Security Domain, to support authentication of the Off-Card Entity to the Security Domain. It precedes the EXTERNAL AUTHENTICATE command. It shall have been preceded (on the same logical channel of the same card I/O interface), immediately or not, by a MANAGE SECURITY ENVIRONMENT or PERFORM SECURITY OPERATION [verify certificate] command.

F.4.2.2 Command Message

The GET CHALLENGE command message is coded according to the following table:

Table F-18: GET CHALLENGE Command Message

Code	Value	Meaning
CLA	'00' - '03' or '40' - '4F'	See section 11.1.4
INS	'84'	GET CHALLENGE
P1	'00'	Reference Control Parameter P1: no information given
P2	'00'	Reference Control Parameter P2: no information given
Lc	Absent	
Data	Absent	
Le	'00'	

A Security Domain may support other values of Reference Control Parameters P1 and P2 as defined in [ISO 7816-4].

F.4.2.3 Data Field Sent in the Command Message

There is no command data.

F.4.2.4 Data Field Returned in the Response Message

Data returned comprises a card challenge, coded on 8 bytes with the key agreement option and 16 bytes with the key transport option.

F.4.2.5 Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

The command may return a general error condition as listed in section 11.1.3 – *General Error Conditions*.

F.4.3 GET DATA [certificate] Command

F.4.3.1 Definition and Scope

The GET DATA [certificate] command is used to retrieve either information about all certificates that can be retrieved from the card, or a single certificate. This command may be issued at any time, in particular it may be interleaved with PERFORM SECURITY OPERATION [verify certificate] commands. If it is issued when a Secure Channel Session is active, it must comply with the Current Security Level of that Secure Channel Session.

F.4.3.2 Command Message

The following command is used to obtain certificate information or a single certificate from the Security Domain.

Table F-19: GET DATA [certificate] Command Message

Code	Value	Meaning
CLA	'00' - '0F', '40' - '4F', '60' - '6F', '80' - '8F', 'C0' - 'CF', or 'E0' - 'EF'	See section 11.1.4
INS	'CA' or 'CB'	If CLA = '00' - '0F', '40' - '4F', or '60' - '6F', even or odd instruction code 'CA' or 'CB' If CLA = '80' - '8F', 'C0' - 'CF', or 'E0' - 'EF', even instruction code 'CA'
P1 P2		Reference Control Parameters P1 and P2: there are three options:
	'7F 21'	Tag of certificate
	'50 31'	Data object identifier of EF.OD
	'xx xx'	(Any other value) data object identifier of a certificate
Lc	'xx' or omitted	Not present if P1 P2 = '7F 21', otherwise length of command data
Data	'xx xx...' or omitted	Not present, or command data
Le	'00'	

F.4.3.3 Reference Control Parameters P1 and P2

If P1 P2 = '7F 21', the command is a request to retrieve the certificate of the Security Domain's default public key, CERT.SD.AUT.

Otherwise, for any value other than those defined in section 11.3 – *GET DATA Command*, the command is a request to access EF.OD or another certificate and the instruction code shall be set to 'CA' if the class byte indicates a GlobalPlatform command (CLA set to '80' - '8F', 'C0' - 'CF', or 'E0' - 'EF') or 'CB' if the class byte indicates an ISO command (CLA set to '00' - '0F', '40' - '4F', or '60' - '6F').

- If P1 P2 = '50 31', the command is a request to retrieve details of all available certificates held by the Security Domain for retrieval and verification by an Off-Card Entity;

- Otherwise, for any other value of P1 P2, the command shall be treated as a request to retrieve a certificate whose pointer is given in P1 P2. This would typically follow a command with P1 P2 = '50 31', where the Cryptographic Information Objects returned in the response have pointers to individual certificates. However, the Off-Card Entity may already know the location of a required certificate, and issue this command directly.

F.4.3.4 Data Field Sent in the Command Message

When P1-P2 is different from '7F21', the command data shall be present and coded as follows:

Table F-20: GET DATA [certificate] Command Data Message

Name	Length	Name	Presence
Tag list tag	1	'5C'	Mandatory
Tag list length	1	'00' (empty, indicating ‘retrieve all data’)	Mandatory

F.4.3.5 Data Field Returned in the Response Message

When the command is issued to retrieve a certificate (P1-P2 different from '5031') and the class byte indicates a GlobalPlatform command (CLA set to '80' - '8F', 'C0' - 'CF', or 'E0' - 'EF'), the certificate shall be returned TLV-coded as follows:

Table F-21: GET DATA [certificate] Response Data Field – Certificate

Name	Length	Name	Presence
Certificate tag	2	'7F21'	Conditional
Certificate length	1, 2, or 3	'00' - '7F', or '81 80' - '81 FF', or '82 01 00' - '82 FF FF'	Conditional
Certificate data	n	'xxxx...' (as defined in section F.1.3.4).	Mandatory

When the command is issued to retrieve a certificate (P1-P2 different from '5031') and the class byte indicates an ISO command (CLA set to '00' - '0F', '40' - '4F', or '60' - '6F'), only the certificate value shall be returned.

If the command was issued to retrieve certificate information details (P1-P2 = '5031'), the contents of EF.OD listing all the Cryptographic Information Objects for the different certificates held in by the Security Domain shall be returned as defined in section F.1.3.3.

F.4.3.6 Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

This command may return either a general error condition as listed in section 11.1.3 – *General Error Conditions* or one of the following errors and warning conditions:

Table F-22: Error Conditions

SW1	SW2	Meaning
'6A'	'80'	Incorrect values in command data
'6A'	'88'	Referenced data not found

F.4.4 INTERNAL AUTHENTICATE Command

F.4.4.1 Definition and Scope

This command is used to authenticate the Security Domain, by the Off-Card Entity. This command is also used with the key agreement option, to support session key establishment. It shall be immediately preceded (on the same logical channel of the same card I/O interface) by an EXTERNAL AUTHENTICATE command.

F.4.4.2 Command Message

The INTERNAL AUTHENTICATE command message is coded according to the following table:

Table F-23: INTERNAL AUTHENTICATE Command Message

Code	Value	Meaning
CLA	'00' - '03' or '40' - '4F'	See section 11.1.4
INS	'88'	INTERNAL AUTHENTICATE
P1	'00'	Reference control parameter P1: no information given
P2	'00'	Reference control parameter P2: no information given
Lc	'xx'	Length of Off-Card Entity challenge
Data	'xx xx…'	Command data field
Le	'00'	

A Security Domain may support other values of Reference Control Parameters P1 and P2 as defined in [ISO 7816-4].

F.4.4.3 Data Field Sent in the Command Message

The data field of the INTERNAL AUTHENTICATE command message contains the following data:

Table F-24: INTERNAL AUTHENTICATE Command Data Field

Name	Length	Name	Presence
Off-Card Entity challenge	8 or 16	'xxxx...'	Mandatory
Off-Card Entity id	8	'xxxx...' (part of CERT.OCE.AUT)	Mandatory

Off-Card Entity challenge is 16 bytes for the key transport option, 8 bytes for the key agreement option.

F.4.4.4 Data Field Returned in the Response Message

The Card Signature (key transport) or encrypted Card Signature (key agreement) is returned – see section F.1.4.3 for details.

F.4.4.5 Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

This command may return either a general error condition as listed in section 11.1.3 – *General Error Conditions* or one of the following specific errors and warning conditions:

Table F-25: Warning Conditions

SW1	SW2	Meaning
'61'	'xx'	Response data incomplete, 'xx' more bytes available

Table F-26: Error Conditions

SW1	SW2	Meaning
'6A'	'80'	Incorrect values in command data

F.4.5 MANAGE SECURITY ENVIRONMENT Command

F.4.5.1 Definition and Scope

This command selects the Secure Channel Protocol '10' and its options as well as defining specific keys to be used by the Security Domain. If a Secure Channel Session is active on this logical channel and card I/O interface, it shall be terminated on receipt of this command, regardless of the validity of the command.

F.4.5.2 Command Message

The MANAGE SECURITY ENVIRONMENT command message is coded according to the following table:

Table F-27: MANAGE SECURITY ENVIRONMENT Command Message

Code	Value	Meaning
CLA	'00' - '03' or '40' - '4F'	See section 11.1.4
INS	'22'	MANAGE SECURITY ENVIRONMENT
P1	'81' or 'C1'	Reference Control Parameter P1 '81': External (Off-Card Entity) Authentication only 'C1': External and Internal (Mutual) Authentication
P2	'A4' or 'B6'	Reference Control Parameter P2: 'A4': Authentication: no certificate verification will be performed by the card 'B6': Digital signature: certificate verification will be performed by the card
Lc	'xx'	Length of command data
Data	'xx xx…'	Off-Card Entity data
Le	-	Not present

F.4.5.3 Reference Control Parameter P1

The value of P1 is based on [ISO 7816-4], as follows:

Table F-28: MANAGE SECURITY ENVIRONMENT Reference Control Parameter P1

b8	b7	b6	b5	b4	b3	b2	b1	Description
1	-	-	-	-	-	-	-	Verification, encipherment, external authentication and key agreement
-	1	-	-	-	-	-	-	Computation, decipherment, internal authentication and key agreement
-	-	-	-	-	-	-	1	SET
-	-	X	X	X	X	X	-	Values defined in [ISO 7816-4]

F.4.5.4 Reference Control Parameter P2

This is set according to the template that is appropriate for the subsequent message flow, as defined in [ISO 7816-4]: 'A4' if certificate verification by the card is omitted, 'B6' if certificate verification is to be performed by the card.

F.4.5.5 Data Field Sent in the Command Message

The command data field is formatted as follows:

Table F-29: MANAGE SECURITY ENVIRONMENT Command Data Field

Tag	Length	Name	Presence
'80'	2	Cryptographic mechanism reference: SCP id + options “i” (= scp || i)	Mandatory
'83'	0 or 1-n	Public key reference	Conditional
'84'	0 or 1-n	Private key reference	Conditional

The ‘cryptographic mechanism reference’ (tag '80', value '10') designates GlobalPlatform asymmetric Secure Channel Protocol '10' (SCP10) and its options, and distinguishes it from any other protocol that might be supported by the selected Issuer Security Domain, Security Domain or Application. Option “i” is described in section F.1.1.

The ‘public key reference’ (tag '83') designates the public key to be used by the Security Domain in subsequent cryptographic operations during Secure Channel Session initiation. The referenced public key shall have already been validated by the Security Domain or shall be the public key of the Security Domain’s Trust Point for External Authentication. If no reference value is provided in the command, the Security Domain shall use by default the public key of its Trust Point for External Authentication, designated PK.TP_EX.AUT, as the first key to use for certificate verification within a session; see section F.1.3.1.

The ‘private key reference’ (tag '84') designates the private key to be used by the Security Domain in subsequent cryptographic operations during Secure Channel Session initiation. The referenced private key shall be present and known to the Security Domain. If no reference value is provided in the command, the Security Domain shall use by default its private key designated SK.SD.AUT

A Security Domain may support other data elements as defined in [ISO 7816-4].

F.4.5.6 Data Field Returned in the Response Message

There is no data returned from this command.

F.4.5.7 Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

This command may return either a general error condition as listed in section 11.1.3 – *General Error Conditions* or one of the following specific errors and warning conditions.

Table F-30: Error Conditions

SW1	SW2	Meaning
'6A'	'88'	Referenced data not found
'94'	'84'	Algorithm not supported

F.4.6 PERFORM SECURITY OPERATION [decipher] Command

F.4.6.1 Definition and Scope

This command is used with the session key transport option, and transmits the DES session keys from the Off-Card Entity to the Security Domain. It shall have been preceded (on the same logical channel of the same card I/O interface), immediately or not, by a MANAGE SECURITY ENVIRONMENT or PERFORM SECURITY OPERATION [verify certificate] command.

F.4.6.2 Command Message

The PERFORM SECURITY OPERATION [decipher] command message is coded according to the following table:

Table F-31: PERFORM SECURITY OPERATION [decipher] Command Message

Code	Value	Meaning
CLA	'00' - '03', '40' - '4F', '10' - '13', or '50' - '5F'	See section 11.1.4
INS	'2A'	PERFORM SECURITY OPERATION [decipher]
P1	'80'	Reference Control Parameter P1: clear text object
P2	'84'	Reference Control Parameter P2: cryptogram (plain value encoded in BER-TLV) present in the command
Lc	'xx'	Length of encrypted data
Data	'xx xx…'	Encrypted data
Le	-	Not present

A Security Domain may support other values of Reference Control Parameters P1 and P2 as defined in [ISO 7816-4].

F.4.6.3 Data Field Sent in the Command Message

The data field of the PERFORM SECURITY OPERATION [decipher] command message contains the Encrypted Off-Card Entity Session Data.

The Off-Card Entity session keys are in control reference templates as defined in section F.3.1.2, one template per session key, concatenated for encryption. Off-Card Entity Session Key Data is formed by padding the session key CRTs to the length of the Security Domain public key modulus as shown in the table below, and encrypting the result with the Security Domain public key (PK.SD.AUT).

Table F-32: Off-Card Entity Session Key Data – Clear Text before Encryption

Meaning	Length	Meaning	Presence
Padding	2	'0002'	
Padding	8-n	'FF'...'FF'	
Padding	1	'00'	
Tag of Security Level ('D3')	1	'D3'	Mandatory
Length of Security Level	1	'01'	Mandatory
Security Level	1	'xx'	Mandatory
CRT tag	1	'B4' (CCT) or 'B8' (CT)	Mandatory
Length of CRT	1	'00' - '7F'	Mandatory
CRT contents (with session key)	n	'xxxx...'	Mandatory
…..	….	…..	…
CRT tag	1	'B4' (CCT) or 'B8' (CT)	Optional
Length of CRT	1	'00' - '7F'	Conditional
CRT contents (with session key)	n	'xxxx...'	Conditional

F.4.6.4 Data Field Returned in the Response Message

No data is returned from this command.

F.4.6.5 Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

This command may return either a general error condition as listed in section 11.1.3 – *General Error Conditions* or one of the following specific errors and warning conditions.

Table F-33: Error Conditions

SW1	SW2	Meaning
'6A'	'80'	Incorrect values in command data

F.4.7 PERFORM SECURITY OPERATION [verify certificate] Command

F.4.7.1 Definition and Scope

This command is used to provide a certificate to the Security Domain for verification. It may be preceded (on the same logical channel of the same card I/O interface) by a MANAGE SECURITY ENVIRONMENT command or a PERFORM SECURITY OPERATION [verify certificate] command and may be interleaved with GET DATA [certificate] commands.

F.4.7.2 Command Message

The PERFORM SECURITY OPERATION [verify certificate] command message is coded according to the following table:

Table F-34: PERFORM SECURITY OPERATION [verify certificate] Command Message

Code	Value	Meaning
CLA	'00' - '03', '40' - '4F', '10' - '13' or '50' - '5F'	See section 11.1.4
INS	'2A'	PERFORM SECURITY OPERATION [verify certificate]
P1	'00'	Reference Control Parameter P1: no object in the response
P2	'AE' or 'BE'	Reference Control Parameter P2: input template for certificate verification present in the command 'AE': non-self descriptive card verifiable certificate (only the concatenated value fields are certified) 'BE': self-descriptive card verifiable certificate (the TLV data elements are certified)
Lc	'xx'	Length of certificate data
Data	'xx xx…'	Certificate data
Le	-	Not present

F.4.7.3 Data Field Sent in the Command Message

The command data field is certificate data as follows.

Table F-35: PERFORM SECURITY OPERATION [verify certificate] Command Data Field

Name	Length	Name	Presence
Certificate tag	2	'7F21'	Mandatory
Certificate length	1, 2, or 3	'00' - '7F', or '81 80' - '81 FF' or '82 01 00' - '82 FF FF'	Mandatory
Certificate data	n	'xxxx...' (as described in section F.1.3.4)	Mandatory

The Security Domain verifies the certificate presented using the Current Public Key as known to the Security Domain. For the first certificate presented in the session, the Current Public Key is either the default Public Key - the Public Key of the Trust Point, or another Public Key as announced in the MANAGE SECURITY ENVIRONMENT command. A series of certificates can be presented to the Security Domain in subsequent PERFORM SECURITY OPERATION [verify certificate] commands. For each subsequent PERFORM SECURITY OPERATION [verify certificate] command, the Current Public Key is the one that was certified in the certificate presented in the previous PERFORM SECURITY OPERATION [verify certificate] command.

F.4.7.4 Data Field Returned in the Response Message

No data is returned from this command.

F.4.7.5 Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'.

This command may return either a general error condition as listed in section 11.1.3 – *General Error Conditions* or one of the following specific errors and warning conditions.

Table F-36: Error Conditions

SW1	SW2	Meaning
'63'	'00'	Verification of the certificate failed
'68'	'83'	The last command of the chain was expected
'6A'	'80'	Incorrect values in command data

G Trusted Framework Inter-Application Communication

GlobalPlatform supports a general Trusted Framework scheme for inter-application communication. This general scheme is compatible with the CAT Runtime Environment and secure communication as defined in [TS 102 225], [TS 102 241], and related specifications. In summary:

- An Application, the Receiving Entity, receives a message from an off-card entity, whose contents is destined to another Application (the Target Application);
- The Receiving Entity interacts with a Trusted Framework on the card to communicate with the Target Application;
- The Trusted Framework handles the security of the inter-application communication by applying its own rules to the interaction, which may include finding the Security Domain associated with the Target Application and having it handle the security of the incoming message;
- The Trusted Framework finds the appropriate interface of the Target Application and passes to it the incoming message contents;
- The Target Application processes the event and eventually provides to the Trusted Framework some response message to be returned;
- The Trusted Framework, being responsible of the inter-application communication security, applies its own security rules to the response message, which may include requesting the Security Domain associated with the Target Application to handle the response message security;
- The Receiving Entity handles the transmission of the response message to the off-card entity.

Passing the incoming message through to the Target Application, and returning the Target Application's response to the off-card entity implies security requirements for not only the Trusted Framework but also the Receiving Entity. Each Application present on the card playing the role of a Receiving Entity shall:

- Enforce the Issuer's security rules for inter-application communication;
- Ensure that incoming messages are properly provided unaltered to the Trusted Framework;
- Ensure that any response messages are properly returned unaltered to the off-card entity.

The 'Trusted Path' privilege qualifies an Application as a Receiving Entity.

To be able to interact with a required Trusted Framework, the Receiving Entity shall obtain a handle to the required Trusted Framework.

To be able to receive incoming messages from a Receiving Entity, the Target Application declares its interface(s) to the corresponding Trusted Framework. The Target Application is registered at installation time for each event it is capable of handling.

When requested to perform inter-application communication, the Trusted Framework shall check that:

- The Receiving Entity has the 'Trusted Path' privilege or is somehow registered as a valid Receiving Entity by the OPEN;
- The Target Application is registered as being willing to accept the event or incoming message.

Trusted Frameworks shall comply with the general requirements stated in this section but are otherwise outside the scope of this version of the specification.

H GlobalPlatform Data Values

H.1 Miscellaneous Data Values

H.1.1 GlobalPlatform OID

The Object Identifier (OID) assigned to GlobalPlatform is as follows:

GlobalPlatform OID ::= `{iso(1) member-body(2) country-USA(840) GlobalPlatform(114283)}`

The hexadecimal representation of the above OID is '2A864886FC6B'. The Object Identifier value for Card Recognition Data `{globalPlatform 1}` or `{1 2 840 114283 1}`, which also identifies GlobalPlatform as the Tag Allocation Authority for Card Recognition Data objects has an hexadecimal representation of '2A864886FC6B01' and so on and so forth for all subsequent GlobalPlatform Object Identifiers.

The encoding of an OID is defined in [ISO 8825-1].

H.1.2 GlobalPlatform RID

The Registered Application Provider Identifier (RID) assigned to GlobalPlatform is as follows:

'A000000151'

The definition of a RID is described in [ISO 7816-4].

H.1.3 Default AID for Issuer Security Domain

Based on the above ISO assigned RID, the default AID of the Issuer Security Domain is:

'A0000001510000'

H.2 Structure of Card Recognition Data

All data is TLV encoded. Application tags not defined in this specification are RFU. GlobalPlatform may assign additional Application tags in the future.

Many of the data objects contain Object Identifiers (OIDs). An OID is made up of a series of numeric values. Each OID is shown here in curly brackets, with its component values separated by spaces. The actual numeric values of some of the symbolic OID values shown, such as ‘globalPlatform’, are defined in this document and may themselves comprise a series of values.

Card Recognition Data shall be formatted according to Table H-1 or Table H-2. Off-card entities interested in such data should be able to interpret both formats correctly.

Table H-1: Structure of Card Recognition Data (Format 1)

Tag	Explanation	Length	Value	Presence
'66'	Card Data tag	Variable	Data objects identified in [ISO 7816-6], including tag '73'	Mandatory
'73'	Card Recognition Data tag	Variable	Data objects listed below	Mandatory
'06'	OID tag	Variable	{globalPlatform 1} OID for Card Recognition Data, also identifies GlobalPlatform as the Tag Allocation Authority	Mandatory
'60'	Application tag 0	Variable	See note 2	-
'06'	OID tag	Variable	{globalPlatform 2 v} OID for Card Management Type and Version	Mandatory
'63'	Application tag 3	Variable	See note 3	-
'06'	OID tag	Variable	{globalPlatform 3} OID for Card Identification Scheme	Mandatory
'64'	Application tag 4	Variable	See note 4	-
'06'	OID tag	Variable	{globalPlatform 4 scp i} OID for Secure Channel Protocol of the Issuer Security Domain and its implementation options	Mandatory
'64'	Application tag 4	Variable	See note 4	-
'06'	OID tag	Variable	{globalPlatform 4 scp i} OID for Secure Channel Protocol of the Issuer Security Domain and its implementation options	Conditional
…	…	…	…	…
'65'	Application tag 5	Variable	Card configuration details – see note 5	Optional
'66'	Application tag 6	Variable	Card / chip details – see note 6	Optional
'67'	Application tag 7	Variable	Issuer Security Domain's Trust Point certificate information – see note 7	Optional
'68'	Application tag 8	Variable	Issuer Security Domain certificate information – see note 8	Conditional

Table H-2: Structure of Card Recognition Data (Format 2)

Tag	Explanation	Length	Value	Presence
'66'	Card Data tag	Variable	Data objects identified in [ISO 7816-6], including tag '73'	Mandatory
'73'	Card Recognition Data tag	Variable	Data objects listed below	Mandatory
'06'	OID tag	Variable	{globalPlatform 1} OID for Card Recognition Data, also identifies GlobalPlatform as the Tag Allocation Authority	Mandatory
'60'	Application tag 0	Variable	See note 2	-
'06'	OID tag	Variable	{globalPlatform 2 v} OID for Card Management Type and Version	Mandatory
'63'	Application tag 3	Variable	See note 3	-
'06'	OID tag	Variable	{globalPlatform 3} OID for Card Identification Scheme	Mandatory
'64'	Application tag 4	Variable	See note 4	-
'06'	OID tag	Variable	{globalPlatform 4 scp i} OID for Secure Channel Protocol of the Issuer Security Domain and its implementation options	Mandatory
'06'	OID tag	Variable	{globalPlatform 4 scp i} OID for Secure Channel Protocol of the Issuer Security Domain and its implementation options	Conditional
…	…	…	…	…
'65'	Application tag 5	Variable	Card configuration details – see note 5	Optional
'66'	Application tag 6	Variable	Card / chip details – see note 6	Optional
'67'	Application tag 7	Variable	Issuer Security Domain's Trust Point certificate information – see note 7	Optional
'68'	Application tag 8	Variable	Issuer Security Domain certificate information – see note 8	Conditional

Note 1: Void.

Note 2: Tag '60': The OID {globalPlatform 2 v} identifies a card that conforms to the GlobalPlatform Card Specification version "**v**". Thus a card conforming to the GlobalPlatform Card Specification 2.3 would use OID {globalPlatform 2 3} and a card conforming to the GlobalPlatform Card Specification 2.2.1 would use OID {globalPlatform 2 2 2 1}.

Note 3: Tag '63': The OID `{globalPlatform 3}` indicates a GlobalPlatform card that is uniquely identified by the Issuer Identification Number (IIN) and Card Image Number (CIN), as defined in sections 7.4.1.1 – *Issuer Identification Number* and 7.4.1.2 – *Card Image Number*. The objective is that an off-card entity is able to construct a globally unique identifier for the card by concatenating this `{globalPlatform 3}` OID, the IIN, and the CIN.

Note 4: Tag '64'. The OID `{globalPlatform 4 scp i}` identifies the Secure Channel Protocol of the Issuer Security Domain. "**scp**" identifies the Secure Channel Protocol identifier as defined in section 10.7 – *Secure Channel Protocol Identifier*. "**i**" identifies the eventual implementation. Using Format 1 (see Table H-1), one occurrence (at least) or multiple occurrences of tag '64' may be present, each one embedding a single OID. Using Format 2 (see Table H-2), a single occurrence of tag '64' shall be present, embedding one (at least) or multiple OID(s).

Note 5: The data object with tag '65' may contain information about the GlobalPlatform implementation details or commonly used Card Issuer options. Such information shall be TLV encoded. The structure of this data object is under definition by GlobalPlatform.

Note 6: Tag '66': This data object may contain information about the card and chip implementation, such as the operating system/runtime environment or a security kernel. Such information shall be TLV encoded and may consist of one (or more) OID(s), each OID being introduced by tag '06' and indicating the organization responsible for specifying the operating system, runtime environment or security kernel, and the identification of the corresponding specification and its version number.

Note 7: Tag '67': This data object is related to the use of Secure Channel Protocol '10' and may contain information on the certification policies, certificate formats and certificate ids associated with the Issuer Security Domain's Trust Point (TP_ISD), primarily relating to the use of a public key Secure Channel Protocol. Such information shall be TLV encoded and may consist of one (or more) OID(s).

Note 8: Tag '68': This data object is related to the use of Secure Channel Protocol '10' and may contain information such as certificate types, formats and ids associated with on-card Security Domains relating to the use of a public key Secure Channel Protocol. Such information shall be TLV encoded and may consist of one (or more) OID(s).

H.3 Structure of Security Domain Management Data

The Security Domain Management Data may be returned in the SELECT response message within template '73' as described in section 11.9.3.1 – *Data Field Returned in the Response Message*. When present, the Security Domain Management Data shall be formatted according to Table H-3 or Table H-4. Off-card entities interested in such data should be able to interpret both formats correctly.

Table H-3: Security Domain Management Data (Format 1)

Tag	Explanation	Length	Value	Presence
'73'	Card Recognition Data tag	Variable	Data objects listed below	Mandatory
'06'	OID tag	Variable	{globalPlatform 1} OID for Card Recognition Data, also identifies GlobalPlatform as the Tag Allocation Authority	Mandatory
'60'	Application tag 0	Variable		-
'06'	OID tag	Variable	{globalPlatform 2 v} OID for Card Management Type and Version	Optional
'63'	Application tag 3	Variable		-
'06'	OID tag	Variable	{globalPlatform 3} OID for Card Identification Scheme	Optional
'64'	Application tag 4	Variable		-
'06'	OID tag	Variable	{globalPlatform 4 scp i} OID for Secure Channel Protocol of the Security Domain and its implementation options	Optional
'64'	Application tag 4	Variable		-
'06'	OID tag	Variable	{globalPlatform 4 scp i} OID for Secure Channel Protocol of the Security Domain and its implementation options	Conditional
…	…	…	…	…
'65'	Application tag 5	Variable	Card configuration details	Optional
'66'	Application tag 6	Variable	Card / chip details	Optional
'67'	Application tag 7	Variable	Security Domain's Trust Point certificate information	Optional
'68'	Application tag 8	Variable	Security Domain certificate information	Conditional

Table H-4: Security Domain Management Data (Format 2)

Tag	Explanation	Length	Value	Presence
'73'	Card Recognition Data tag	Variable	Data objects listed below	Mandatory
'06'	OID tag	Variable	{globalPlatform 1} OID for Card Recognition Data, also identifies GlobalPlatform as the Tag Allocation Authority	Mandatory
'60'	Application tag 0	Variable		-
'06'	OID tag	Variable	{globalPlatform 2 v} OID for Card Management Type and Version	Optional
'63'	Application tag 3	Variable		-
'06'	OID tag	Variable	{globalPlatform 3} OID for Card Identification Scheme	Optional
'64'	Application tag 4	Variable		-
'06'	OID tag	Variable	{globalPlatform 4 scp i} OID for Secure Channel Protocol of the Issuer Security Domain and its implementation options	Optional
'06'	OID tag	Variable	{globalPlatform 4 scp i} OID for Secure Channel Protocol of the Security Domain and its implementation options	Conditional
…	…	…	…	…
'65'	Application tag 5	Variable	Card configuration details	Optional
'66'	Application tag 6	Variable	Card / chip details	Optional
'67'	Application tag 7	Variable	Security Domain's Trust Point certificate information	Optional
'68'	Application tag 8	Variable	Security Domain certificate information	Conditional

The data objects in above tables are equivalent to those in Table H-1 and Table H-2, but are specific to this Security Domain and override data objects in Card Recognition Data. See Table H-1 and Table H-2 and the notes that follow them for further details of each data object.

H.4 Structure of Card Capability Information

Card Capability Information provides information complementary to Card Recognition Data about the cipher suites actually supported by the card. If available, Card Capability Information as follows:

Table H-5: Card Capability Information

Tag	Length	Data / Description	Presence
'67'	Variable	Card Capability Information	Mandatory
'A0'	Variable	SCP information for first or only SCP	Mandatory
'A0'	Variable	SCP information for additional SCP(s)	Conditional
…		…	
'81'	3	Privileges that can be assigned to an SSD	Conditional
'82'	3	Privileges that can be assigned to any application	Mandatory
'83'	Variable	Supported LFDBH algorithms	Mandatory
'84'	Variable	Cipher suites for LFDB encryption	Conditional
'85'	Variable	Cipher suites supported for tokens	Conditional
'86'	Variable	Cipher suites supported for receipts	Conditional
'87'	Variable	Cipher suites supported for DAPs	Conditional
'88'	Variable	Key Parameter Reference List	Conditional

Tag 'A0' indicates support of a Secure Channel Protocol by the card. At least one occurrence of tag 'A0' shall be present. This tag may occur more than once if the card supports several Secure Channel Protocols.

Tag '81' shall be present if the card supports Supplementary Security Domains. It shall be encoded as a bitmap of privileges, as specified in section 11.1.2, and shall indicate the privileges that may actually be assigned to Supplementary Security Domains on this card.

Tag '82' shall be present. It shall be encoded as a bitmap of privileges, as specified in section 11.1.2, and shall indicate the privileges that may actually be assigned to Applications on this card.

Tag '83' shall be present. It shall indicate the algorithms supported by the card to compute the Load File Data Block Hash.

Tags '84' shall be present if the card supports Ciphered Load File Data Block. It shall be encoded as specified in Table H-8 and shall indicate the encryption schemes supported by the card.

Tags '85' and '86' shall be present if the card supports Delegated Management. These tags shall be encoded as specified in Table H-9 and Table H-10 and shall indicate the signature schemes supported by the card, respectively for Token Verification and Receipt Generation.

Tag '87' shall be present if the card supports Supplementary Security Domains. It shall be encoded as specified in Table H-9 and Table H-10 and shall indicate signature schemes supported by the card for DAP Verification.

Tag '88' shall be present if the card supports Delegated Management and/or DAP Verification schemes based on EC cryptography.

The following table describes the structure of the ‘SCP Information’ TLV:

Table H-6: SCP Information

Tag	Length	Data / Description	Presence
'A0'	Variable	SCP information	
'80'	1	SCP type ('02', '03', '80', '81')	Mandatory
'81'	Variable	List of supported options for that protocol (e.g. '15 55' for SCP02)	Mandatory
'82'	Variable	Supported keys for SCP03	Conditional
'83'	Variable	Supported TLS cipher suites for SCP81	Conditional
'84'	1	Maximum length of Pre Shared Key in bytes (unsigned integer) (for SCP81 only)	Conditional

The following table describes the coding of the 'Supported Keys for SCP03' TLV:

Table H-7: Supported Keys for SCP03

b8	b7	b6	b5	b4	b3	b2	b1	Description
-	-	-	-	-	-	-	1	AES-128
-	-	-	-	-	-	1	-	AES-192
-	-	-	-	-	1	-	-	AES-256
X	x	x	x	x	-	-	-	RFU (0)

The 'Supported TLS cipher suites for SCP81' TLV contains a sequence of supported cipher suite numbers as defined in [RFC 4279], [RFC 4785], and [RFC 5487] (limited to cipher suites actually referenced in [Amd B]). Each cipher suite number is itself a 2-byte data.

The 'Supported LFDBH algorithms' TLV contains a sequence of supported LFDBH algorithms. Each LFDBH algorithm is encoded as a single byte as follows:

'01' SHA-1

'02' SHA-256

'03' SHA-384

'04' SHA-512

The following table describes the coding of the 'Cipher Suites for LFDB Encryption' TLV:

Table H-8: Cipher Suites for LFDB Encryption

b8	b7	b6	b5	b4	b3	b2	b1	Description
-	-	-	-	-	-	-	1	Triple DES with 16 byte key length
-	-	-	-	-	-	1	-	AES-128
-	-	-	-	-	1	-	-	AES-192
-	-	-	-	1	-	-	-	AES-256
1	-	-	-	-	-	-	-	ICV supported for LFDB encryption
-	x	x	x	-	-	-	-	RFU (0)

Additional bytes may be appended in the future.

The following tables describe the coding of the 'Cipher Suites for Tokens', 'Cipher Suites for Receipts', and 'Cipher Suites for DAPs' TLVs:

Byte 1:

Table H-9: Cipher Suites for Signatures – Byte 1

b8	b7	b6	b5	b4	b3	b2	b1	Description
-	-	-	-	-	-	-	1	RSA-1024 / RSASSA-PKCS-v1_5 / SHA-1 (see section B.3.1.1)
-	-	-	-	-	-	1	-	RSA >1024 / RSASSA-PSS / SHA-256 (see section B.3.2.1)
-	-	-	-	-	1	-	-	16 byte key / Single DES plus Final Triple DES MAC (see section B.1.2.2)
-	-	-	-	1	-	-	-	CMAC using AES-128 (see section B.2.2)
-	-	-	1	-	-	-	-	CMAC using AES-192 (see section B.2.2)
-	-	1	-	-	-	-	-	CMAC using AES-256 (see section B.2.2)
-	1	-	-	-	-	-	-	ECDSA using ECC-256 and SHA-256 (see section B.4.3)
1	-	-	-	-	-	-	-	ECDSA using ECC-384 and SHA-384 (see section B.4.3)

Byte 2:

Table H-10: Cipher Suites for Signatures – Byte 2

b8	b7	b6	b5	b4	b3	b2	b1	Description
-	-	-	-	-	-	-	1	ECDSA using ECC-512 and SHA-512 (see section B.4.3)
-	-	-	-	-	-	1	-	ECDSA using ECC-521 and SHA-512 (see section B.4.3)
X	x	X	X	x	x	-	-	RFU (0)

Byte 2 may be missing if its content is zero. Additional bytes may be appended in the future.

The ‘Key Parameter Reference List’ TLV contains a sequence of global Key Parameter References corresponding to the sets of ECC curve parameters that are available on the card.