GlobalPlatform Card

Mapping Guidelines of Existing GP v2.1.1 Implementation on v2.2.1

Version 1.0.1

Member Release

January 2011

Document Reference: GPC_GUI_003

Table of contents

1 Overview

This document provides implementation guidelines for mapping a GlobalPlatform card based on the Card Specification version 2.1.1 (GP CS 2.1.1 [0]) onto the Card Specification version 2.2.1 (GP CS 2.2.1 [1]). This guideline defines a subset of features defined in GP CS 2.1.1 [0] thus it defines sample of implementations. These implementations are based on the Java Card™ 2.1.1 or Java Card™ 2.2 specifications and implement the Java Card™ 2.1.1 API (JC 2.1.1 API [2]) or the Java Card™ 2.2 API (JC 2.2 API [5]).

Throughout this document, special clarification will be provided if a particular guideline does not apply to all implementations or only applies to a particular implementation.

1.1 Security Domains

Two types of Security Domains exist for this guideline; the Issuer Security Domain that is present and active for all implementations and a Security Domain application code (an Executable Load File) from which additional Security Domains can be installed for implementations supporting additional Security Domains (i.e. a Supplementary Security Domain).

From this point on in this document the term Security Domain usually refers to the behavior of both the Issuer Security Domain and any installed Security Domain. Wherever the behavior differs, this will be clarified with the terms Issuer Security Domain to identify the issuer's Security Domain and Supplementary Security Domain to identify an Application Provider's or Controlling Authorities' Security Domain.

1.2 Recommended Privileges

This guideline supports the following privileges only:

– Security Domain
– DAP Verification
– Card Lock
– Card Terminate
– Card reset
– CVM Management
– Mandated DAP Verification
– Trusted Path
– Authorized Management
– Global Lock
– Global Registry
– Final Application

1.3 Recommended Application Programming Interfaces

1.3.1 GlobalPlatform 2.2.1

The following are the Application Programming Interfaces (API) that are accessible to applications developed according to the GlobalPlatform 2.1.1 API [0]. (See description in Appendix A of GP CS 2.2.1 [1] and the recommendations in section 7 of this document).

The `GPSystem` methods to be present on all implementations.

– setATRHistBytes
– setCardContentState
– getCardContentState

- getCardState
- getRegistryEntry
- getSecureChannel
- getService
- getCVM
- lockCard
- terminateCard

The `SecureChannel` interface methods to be present for the Issuer Security Domain and for each instance of a Supplementary Security Domain if the implementation supports Supplementary Security Domains.

- decryptData
- encryptData
- getSecurityLevel
- processSecurity
- resetSecurity
- unwrap
- wrap

The `GPRegistryEntry` object interface to be present for the Issuer Security Domain, for each instance of a Supplementary Security Domain if the implementation supports Supplementary Security Domains and for each instance of Application. For each `GPRegistryEntry` object the interface methods to be present.

- deregisterService
- getAID
- getPrivileges
- getState
- isAssociated
- isPrivileged
- registerService
- setState

The `CVM` interface methods to be present on all implementations.

- blockState
- getTriesRemaining
- isActive
- isBlocked
- isSubmitted
- isVerified
- resetState
- resetAndUnblockState
- setTryLimit
- update
- verify

The `SecureChannelx` interface method to be present for the Issuer Security Domain and for each instance of a Supplementary Security Domain if the implementation supports Supplementary Security Domains.

- setLevel

1.3.2 Java Card™

- All implementations support the Java Card™ API and pass the corresponding compliance tests.

More specifically the following recommendations apply:

The package `javacard.security` defines a set of classes and interfaces for the Java Card™ security framework. The full set of classes is present and the presence of the interfaces depends on the implementation.

All implementations support symmetric cryptography (DES). Asymmetric cryptography is also supported on implementations supporting PKI functions.

The following interfaces are supported on all implementations:

– DESKey
– Key
– SecretKey

In addition to the above 3 interfaces, the following interfaces are also supported implementations supporting PKI functions:

– PrivateKey
– PublicKey
– RSAPrivateCrtKey
– RSAPrivateKey
– RSAPublicKey

The `CryptoException` class is supported on all implementations.

The following fields within the `KeyBuilder` class are present on all implementations:

– LENGTH_DES
– LENGTH_DES3_2KEY
– TYPE_DES
– TYPE_DES_TRANSIENT_DESELECT
– TYPE_DES_TRANSIENT_RESET

In addition to the above 5 fields, the following fields within the `KeyBuilder` class are also present for implementations supporting PKI functions:

– LENGTH_RSA_1024
– LENGTH_RSA_512
– LENGTH_RSA_768
– TYPE_RSA_CRT_PRIVATE
– TYPE_RSA_PRIVATE
– TYPE_RSA_PUBLIC

For implementations supporting PKI functions, the `buildKey()` method supports any other `keyLength` parameter for a `keyType`; `TYPE_RSA_CRT_PRIVATE`, `TYPE_RSA_PRIVATE` or `TYPE_RSA_PUBLIC`, as long as the length is between 512 and the maximum length supported by the card and is a multiple of 32 bits.

The following field within the `MessageDigest` class is supported for implementations supporting PKI functions:

– ALG_SHA

The complete `RandomData` class is supported for all implementations.

The following fields within the `Signature` class is present for all implementations:

- ALG_DES_MAC8_ISO9797_M1
- ALG_DES_MAC8_ISO9797_M2
- ALG_DES_MAC8_ISO9797_1_M2_ALG3
- ALG_DES_MAC8_NOPAD
- MODE_SIGN
- MODE_VERIFY

In addition to the above 5 fields, the following fields within the `Signature` class are also present for implementations supporting PKI functions:

- ALG_RSA_SHA_ISO9796
- ALG_RSA_SHA_PKCS1

The following fields within the `Cipher` class are present for all implementations:

- ALG_DES_CBC_ISO9797_M1
- ALG_DES_CBC_ISO9797_M2
- ALG_DES_CBC_NOPAD
- ALG_DES_ECB_ISO9797_M1
- ALG_DES_ECB_ISO9797_M2
- ALG_DES_ECB_NOPAD
- MODE_DECRYPT
- MODE_ENCRYPT

In addition to the above 8 fields, the following fields within the `Cipher` class are also present for implementations supporting PKI functions:

- ALG_RSA_PKCS1
- ALG_RSA_NOPAD

The following field within the `Checksum` class is present for all implementations:

- ALG_ISO3309_CRC16

1.4 Clarifications for Java Card™ and EMV

While this implementation is based upon the Java Card™ specifications, the following areas exist where the information contained in GP CS 2.2.1 [1]overrides the Java Card™ 2.1.1 documentation (the Java Card™ 2.2 specifications have in some cases been updated to reflect these recommendations):

- The RID of an Application instance AID does not have to be the same as the RID of the Package or the same as the RID of the application class from which the Application was instantiated.
- The parameters defined for the install() method contain more than just the Application Specific Parameters i.e. the instance AID and Application Privileges are present in addition to the Application Specific Parameters (see section 6.6.2.3).
- The GP CS 2.2.1 [1] mandates the suitable order as noted in the Java Card™ Virtual Machine [6] specification for the components of the CAP file.
- If the GlobalPlatform implementation is for use on EMV compliant terminals, EMV level 1 requirements, except for item 2 and 3 below, take precedence over ISO/IEC 7816-4 [8] and the protocol rules described in the Java Card™ Runtime Environment (JCRE) Specification [7] and Java Card™ Application Programming Interface [6].
- There are 3 known protocol discrepancies between ISO, EMV and Java Card™:

1. Responding to a case 2 command issued with a length (Le) of binary zero ('00') for a T=0 card.

All implementations respond to such a command with a '6Cxx' response instead of the '61xx' response suggested by the Java Card™ specification and this, along with the subsequent re-issued command, are managed by the JCRE. Refer to section 8.4.1.1 of JC RE 2.1.1 [4], section 9.4.1.1 of JCRE 2.2 [7], and section 5.3.1.2.1 of EMV Book 1 [9].

A simple example of the expected behavior is as such:

When an applet invokes the `setOutGoingAndSend(bOff, len)` method, the JCRE responds with a '6Cxx' (xx = len) response if Le on the incoming command was '00'.

If the subsequent command is not the same as the previous incoming command, the card behavior can be unpredictable. If the subsequent command is the same as the previous incoming command with a Le not equal to '00':

- The JCRE does not invoke the `process()` method of the currently selected application.
- The JCRE does output Le bytes of data from the APDU buffer followed by a '9000' or '61xx' returned code.

2. Responding to a case 4 command with a warning return code for a T=0 card.

While not defined as an option in EMV, a Java Card™ can only respond with the warning return code following the last GET RESPONSE command. Refer to *section 5.3.1.1* of EMV Book 1 [9] and the JC 2.1.1 API [2]or JC 2.2 API [5] (specifically the APDU class).

A simple explanation of the issue relates to the SELECT command when the card is in the Life Cycle State CARD_LOCKED and is as such:

When the Issuer Security Domain invokes the `setOutGoingAndSend(bOff, len)` method, the JCRE does not at this point have any knowledge of the warning return code. The only option available to the JCRE is to use the '61xx' return code and the GET RESPONSE command sequence. Only when control is passed back to the Issuer Security Domain, it can set the value of the return code to '6283'.

3. The class byte of the GET RESPONSE command.

While EMV states that the class (CLA) byte of a GET RESPONSE command will always be zero ('00'), ISO/IEC 7816-4 and other specifications do not mandate this. From a GlobalPlatform point of view, in order to ensure that GlobalPlatform cards are accepted in the widest range of terminals possible and in order to support logical channels, these implementations do not limit the value of the CLA byte for a GET RESPONSE command.

2 Security Principles

The following are security principles as defined by this guideline for the possible entities on a GlobalPlatform card:

2.1 Privileges

The following table lists the Privileges supported by the Issuer Security Domain and by each instance of the Supplementary Security Domain if the implementation supports Supplementary Security Domain and by Applications:

		ISD	SSD	Application
Privileges	Security Domain	✓	✓	
	DAP Verification		✓	
	Delegated Management			
	Card Lock	✓		✓
	Card Terminate	✓		✓
	Card Reset	✓		✓
	CVM management	✓		✓
	Mandated DAP Verification		✓	
	Trusted Path	**O**	**O**	
	Authorized Management	✓		
	Token Verification			
	Global Delete	✓		
	Global Lock	✓		**O**
	Global Registry	✓		**O**
	Final Application	✓		
	Global Service			
	Receipt Generation			

Table 1: Supported Privileges

Ticks (✓) denote recommended support.

Blanks denote that the support of the privilege is beyond scope of this guideline.

O denotes optional support.

2.2 Issuer Security Domain

When the Issuer Security Domain is the selected Application, all commands besides those required to setup the Secure Channel are performed within a Secure Channel Session. The one exception to this rule relates to the GET DATA command that can be issued to the Issuer Security Domain without first setting up a Secure Channel.

The Issuer Security Domain supports Secure Channel Protocol '02' and specifically implementation option '55'.

The card is not required to support more than one Secure Channel Session at a time.

If the implementation supports logical channels in addition to the Basic Logical Channel, and if the card supports only one Secure Channel session at a time, the Issuer Security Domain rejects the initiation of a Secure Channel if a Secure Channel Session is currently available on one of the other logical channels.

The level of security of the commands following the setup of the Secure Channel Session is defined within the Secure Channel setup and may be enforced by the Issuer Security Domain depending on the Life Cycle State of the card: (This is immaterial of whether the Issuer Security Domain is the selected Application or if an associated Application is using the services of the Issuer Security Domain.)

- For commands issued to the Issuer Security Domain when the card is in a Life Cycle State of OP_READY or INITIALIZED, it is only recommended that a Secure Channel Session has been setup with security level AUTHENTICATED.
- For commands issued to the Issuer Security Domain when the card is in a Life Cycle State of SECURED or CARD_LOCKED, a Secure Channel Session must have been setup and the security level must indicate AUTHENTICATED and C-MAC on all commands within the Secure Channel Session.
- Any additional security required by the issuer over and above that enforced by the card Life Cycle State must be indicated in the setup of the Secure Channel Session.

All DES keys are double length keys (16 bytes).

All MAC generations result in an 8-byte field. These 8 bytes constitute the MAC.

In the CARD_LOCKED Life Cycle State the level of security required for commands is the same as in the SECURED state but this only applies to certain commands, as Card Content management commands are not processed at all in the CARD_LOCKED Life Cycle State i.e. in the CARD_LOCKED Life Cycle State, the INSTALL command, and by association the LOAD command, the DELETE command, the PUT KEY command and the STORE DATA command are not processed.

The Issuer Security Domain has always the following privileges:

- Security Domain
- Card Lock
- Card Terminate
- CVM Management
- Authorized Management
- Global Delete
- Global Lock
- Global Registry
- Final Application

In addition to above list the Issuer Security Domain has initially the following privilege:

- Card reset

And if the implementation has support for personalization the Issuer Security Domain also has the following privilege:

- Trusted Path

2.3 Supplementary Security Domains present on an implementation

When a Supplementary Security Domain is the selected Application, all commands besides those required to setup the Secure Channel are performed within a Secure Channel Session. The one exception to this rule relates to the GET DATA command that can be issued to the Security Domain without first setting up a Secure Channel.

Following installation of a Security Domain, no keys exist in the Security Domain and the Security Domain uses the services of the Issuer Security Domain to set up a Secure Channel until one complete Secure Channel DES Key Version Number has been successfully loaded to the Security Domain. From that point on the Security Domain always uses its own Secure Channel keys to set up a subsequent Secure Channel.

The installable Security Domain code supports Secure Channel Protocol '02' and specifically implementation option '55'.

The card is not required to support more than one Secure Channel Session at a time.

If the implementation supports logical channels in addition to the Basic Logical Channel, and if the card supports only one Secure Channel session at a time, the Supplementary Security Domain rejects the initiation of a Secure Channel if a Secure Channel Session is currently available on one of the other logical channels. When the Security Domain is using its own keys, the level of security of the commands following the setup of the Secure Channel Session is defined within the Secure Channel setup and is enforced by the Security Domain depending on the Life Cycle State of the card: (This is immaterial of whether the Security Domain is the selected Application or if an associated Application is using the services of the Security Domain.)

- For commands issued to the Security Domain when the card is in a Life Cycle State of OP_READY or INITIALIZED, it is only recommended that a Secure Channel Session has been setup to security level AUTHENTICATED.
- For commands issued to the Security Domain when the card is in a Life Cycle State of SECURED, a Secure Channel Session must have been setup and the security level must indicate AUTHENTICATED and C-MAC on all commands within the Secure Channel Session.
- Any additional security required by the Application Provider over and above that enforced by the card Life Cycle State must be indicated in the setup of the Secure Channel Session.

All DES keys are double length keys (16 bytes).

All MAC generations result in an 8-byte field. These 8 bytes constitute the MAC.

If the Supplementary Security Domain is intended to perform Load File Data Block verification (DAP Verification) the use of a 1024-bit RSA key is recommended.

Any Supplementary Security Domain has the option to reject or accept Applications being extradited to it. This option is specified in the install parameters of the INSTALL [for install and make selectable] command. (By default, Supplementary Security Domains rejects any extradition request.)

A Supplementary Security Domain supports the following privileges in addition to Security Domain Privilege:

- DAP Verification
- Mandated DAP Verification

And if the implementation has support for personalization the following privilege is implicitly given during installation of an instance when install parameters are coded in 1 byte:

- Trusted Path

2.4 Applications

Application behavior is outside the scope of this guideline and the following is only included for informational purposes.

Applications may choose to follow exactly the same security principles as a Security Domain and to utilize the same APDU command structures to provide similar functionality e.g. GET STATUS, PUT KEY, STORE DATA etc.,

If an application utilizes the services of its associated Security Domain to provide it with Secure Channel functionality, the Application developer must be aware that its associated Security Domain will enforce the same security principles that apply to the Security Domain to all Secure Channel functionality that the Security Domain is requested to process by the Application.

Each instance of an application may have the following privileges:

- Card Lock
- Card Terminate
- Card Reset
- CVM Management
- Global Lock
- Global Registry

3 Data Recommendations

The following are recommendations for support of data within the OPEN and Security Domains and for support of data within the `CVM` interface:

3.1 OPEN

The AID for the Issuer Security Domain Application within the GlobalPlatform Registry is initially 'A0 00 00 01 51 00 00 00'. The Application Privileges of the Issuer Security Domain is initially set to '9E DE 00' if it implements support for personalization (i.e. it has the Trusted Path privilege) otherwise to '9E 5E 00'.

For implementations supporting Supplementary Security Domains, the AID for the installable Security Domain Executable Load File (package) within the GlobalPlatform Registry is 'A0 00 00 01 51 53 50' and the Executable Module AID (applet) within the GlobalPlatform Registry is 'A0 00 00 01 51 53 50 41'. The Issuer Security Domain is this Executable Load File's associated Security Domain.

If any other installable Executable Load Files exist within the GlobalPlatform Registry when the card reaches the OP_READY Life Cycle State, the Issuer Security Domain is the associated Security Domain for each of these Executable Load Files.

The GlobalPlatform Registry is able to store at least 21 entries of a combination of at least one Executable Load File, at least one Executable Module and Applications.

Implementations are not required to store the package AID for any API in the GlobalPlatform Registry. If the AID of any of these packages is stored in the GlobalPlatform Registry, they are not externally visible using the GET STATUS command.

For implementations supporting symmetric cryptography only, the internal APDU buffer has the capability of handling, at a minimum, a command or response data field of 128 bytes.

For implementations supporting symmetric and asymmetric cryptography, the internal APDU buffer has the capability of handling, at a minimum, a response data field of 256 bytes.

On implementations that support at least one Supplementary Logical Channel, the number of available logical channels and the fact that the card assigns logical channel numbers are indicated in the historical bytes of the Answer To Reset string as defined in ISO/IEC 7816-4 (specifically table 87).

This guideline defines the Answer to Reset string for a T=0 protocol card as:
'3B 68 00 00 00 73 C8 40 zz 00 90 00' or '3F 68 00 00 00 73 C8 40 zz 00 90 00'

This guideline defines the Answer to Reset string for a T=1 protocol card as:
'3B E8 00 00 81 31 20 45 00 73 C8 40 zz 00 90 00' + TCK or

'3F E8 00 00 81 31 20 45 00 73 C8 40 zz 00 90 00' + TCK

Further, for the historical bytes, the byte 'zz' is either '00' (indicating support for the Basic Logical Channel only) or '11', '12' or '13' (indicating logical channel assignment by the card only and support for 1, 2 or 3 Supplementary Logical Channels respectively).

Initially and unless an Application with the Card Reset privilege invokes the `setATRHistBytes()` method, the above recommendation applies to both the cold and warm reset (note that it is possible for the cold reset string to reflect alternative interface characters). If an Application with the Card Reset privilege does invoke the `setATRHistBytes()` method, the historical bytes for the warm reset is never altered.

3.2 Issuer Security Domain

3.2.1 Data Store

The Issuer Security Domain reserves or is able to allocate at least 250 bytes of space as a general data store (Data Store) used to maintain GlobalPlatform specified data (tags '42', '45' and '66'), tag 'CF' and issuer proprietary TLV coded data that may be personalized by the issuer. No data element can be larger than 127 bytes.

Due to the fact that the length of data written to the Data Store can vary, an implementation is aware of the maximum length of the data in addition to the actual length. The maximum length of each value is either defined in this section or is the actual length of the value the first time it is populated

Data is populated to this Data Store using the STORE DATA command and retrieved from the Data Store using the GET DATA command except where the following cases define additional or contradictory recommendations.

- Additional recommendations, beyond storing the data in the Data Store, exist for tag 'CF'. (see section 6.5)
- The GlobalPlatform defined tag '4F' has additional recommendations as defined in section 6.13. Although tag '4F' can be modified using the STORE DATA command, it cannot be retrieved using the GET DATA command, as it will not be found in the Data Store.
- The GlobalPlatform defined tags 'C1', 'E0' and 'C0' have additional recommendations as defined in sections 3.2.3 and 3.2.4 below and while they can be retrieved using the GET DATA command they cannot be directly personalized using the STORE DATA command.

A vendor does not place tag '45' in the Data Store unless it has been informed by the intended Card Issuer what the maximum length for this value is. If a vendor places tag '42' in the Data Store the maximum length for this value is 6 bytes.

3.2.1.1 Card Recognition Data (tag '66')

The Issuer Security Domain is able to store Card Recognition Data up to a maximum of 127 bytes. Initially the following information is recommended in the Card Recognition Data depending on the implementation. Subsequently this data can be updated using the STORE DATA command.

Tag	Length	Data/Description
'66'	'4C'	Card Data
'73'	'4A'	Card Recognition Data
'06'	'07'	'2A 8648 86FC6B 01'
'60'	'0C'	Card management type and version
'06'	'09'	'2A 8648 86FC6B 02 02 02'
'63'	'09'	Card identification scheme
'06'	'07'	'2A 8648 86FC6B 03'
'64'	'0B'	Secure Channel Protocol of Issuer Security Domain and its implementation options
'06'	'09'	'2A 8648 86FC6B 04 02 55'
'65'	'0B'	Card configuration details
'06'	'09'	'2A 8648 86FC6B 02 01' concatenated with '01', '02' or '03'
'66'	'0C'	Card/chip details
'06'	'0A'	'2B 06 01 04 01 2A 02 6E 01 01' Java Card™ 2.1.x '2B 06 01 04 01 2A 02 6E 01 02' Java Card™ 2.2

Table 2: Card Recognition Data

Implementations are not responsible for insuring the integrity of Card Recognition Data.

The Card Recognition Data is retrievable using the GET DATA command.

Optionally the Card Recognition Data (tag '73') is returned in response to a SELECT command sent to the Issuer Security Domain.

3.2.1.2 Key Derivation Data (tag 'CF')

It is the responsibility of the issuer rather than the card to ensure that the Key Derivation Data and all keys within the Issuer Security Domain remain synchronized. The Issuer Security Domain is responsible for maintaining a maximum of 10 bytes of derivation data that presumably applies to all Secure Channel keys within the Issuer Security Domain.

Initially this data is set to the last 2 bytes of the Issuer Security Domains AID followed by a unique data assigned by the chip manufacturer.

It is possible to modify this derivation data using the STORE DATA command.

The content of the Key Derivation Data has no effect on the keys stored in the Issuer Security Domain i.e. there is no requirement that this derivation data actually be used by the issuer to derive keys.

The Key Derivation Data is returned as part of the response to the INITIALIZE UPDATE command.

The Key Derivation Data is also retrievable using the GET DATA command.

3.2.2 Secure Channel Keys

The Issuer Security Domain has three identical 16-byte initial static keys. The Key Version Numbers of these keys is set to 'FF' and the Key Identifiers are set to '01', '02' and '03' respectively. Unless otherwise requested by the Card Issuer, the value of any initial key is:

'40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F'

The Issuer Security Domain also reserves, or is able to allocate, space for an additional 3 Key Version Numbers each with 3 Key Identifiers, i.e. 9 keys of 16 bytes. These Key Version Numbers can range from '01' to '6F'. The Key Identifiers are always '01', '02' and '03'.

Each Secure Channel Key Version Number has an associated Sequence Counter that is initially zero on creation of the Key Version Number, reset to zero any time values within the Key Version Number are updated and incremented following each successful issuer authentication performed using this Key Version Number. The Secure Channel Sequence Counter for a particular Key Version Number is returned as part of the response to the INITIALIZE UPDATE command.

Secure Channel keys are populated using the PUT KEY (DES keys) command or the STORE DATA commands.

3.2.3 Default Secure Channel Sequence Counter (tag 'C1')

The Issuer Security Domain identifies the Secure Channel Sequence Counter of its default Key Version Number. As with all Sequence Counters in the Issuer Security Domain the value of this Sequence Counter changes as defined in section 3.2.2 above. This is the only manner in which this data is updated.

In addition to being returned in the response to the INITIALIZE UPDATE command, this default Secure Channel Sequence Counter is also retrievable using the GET DATA command.

3.2.4 Key Information Template and Key Information Data (tags 'E0' and 'C0')

The Issuer Security Domain is able to build a Key Information Template containing multiple Key Information Data depending on the available keys.

The Key Information Template is retrievable using the GET DATA command.

3.3 Supplementary Security Domains present on the implementations

3.3.1 Data Store

Each instance of the installable Security Domain reserves or is able to allocate 128 bytes of space as a general Data Store used to maintain tag 'CF' and proprietary TLV coded data that may be personalized by the Application Provider or Controlling Authority. No data element can be larger than 127 bytes.

Due to the fact that the length of data written to the Data Store can vary, an implementation is aware of the maximum length of the data in addition to the actual length. The maximum length of each value is either defined in this section or is the actual length of the value the first time it is populated

Data is populated to this Data Store using the STORE DATA command and retrieved from the Data Store using the GET DATA command except where the following cases define additional or contradictory recommendations.

- Additional recommendations, beyond storing the data in the Data Store, exist for tag 'CF'. (see section 6.5)
- The GlobalPlatform defined tags 'C1', 'E0' and 'C0' have additional recommendations as defined in sections 3.3.4 and 3.3.5, below, and while they can be retrieved using the GET DATA command they cannot be directly personalized using the STORE DATA command.

Key Derivation Data (tag 'CF')

It is the responsibility of the Application Provider or the Controlling Authority rather than the card to ensure that the Key Derivation Data and all keys within the Security Domain remain synchronized. Each Supplementary Security Domain is responsible for maintaining a maximum of 10 bytes of derivation data that presumably applies to all Secure Channel keys within the Security Domain.

Following installation of the Security Domain this data is set to the last 2 bytes of the Security Domain’s AID followed by a unique data assigned by the chip manufacturer.

It is possible to modify this derivation data using the STORE DATA command.

The content of Key Derivation Data has no effect on the keys stored in the Security Domain i.e. there is no requirement that this derivation data actually be used by the Application Provider or Controlling Authority to derive keys.

This information is returned as part of the response to the INITIALIZE UPDATE command once the Security Domain has been populated with its first Secure Channel Key Version Number.

The Key Derivation Data is also retrievable using the GET DATA command.

3.3.2 Secure Channel Keys

The Supplementary Security Domain code reserves, or is able to allocate, space for at least 3 Key Version Numbers each with 3 Key Identifiers i.e. 9 keys of 16 bytes. These Key Version Numbers can range from '01' to '6F'. The Key Identifiers are always '01', '02' and '03'. Initially (i.e. on installation) this space is not populated with any key data.

Each Secure Channel Key Version Number has an associated Sequence Counter that is initially zero on creation of the Key Version Number, reset to zero any time values within the Key Version Number are updated and incremented following each successful Application Provider or Controlling Authority authentication performed using this Key Version Number. The Secure Channel Sequence Counter for a particular Key Version Number is returned as part of the response to the INITIALIZE UPDATE command.

Secure Channel keys are populated using the PUT KEY (DES keys) command or the STORE DATA command.

3.3.3 DAP Verification Key

A Supplementary Security Domain that has been installed with the privilege of performing DAP Verification reserves, or is able to allocate, space for a DAP Verification public key (Key Version Number '73' / Key Identifier '01'). Initially (i.e. on installation) this space is not populated with any key data.

The DAP Verification key is populated using the PUT KEY (RSA Public Key) command or the STORE DATA command.

3.3.4 Default Secure Channel Sequence Counter (tag 'C1')

Each Supplementary Security Domain identifies the Secure Channel Sequence Counter of its default Key Version Number. As with all Sequence Counters in the Supplementary Security Domain the value of this Sequence Counter changes as defined in section 3.3.2 above. This is the only manner in which this data is updated.

In addition to being returned in the response to the INITIALIZE UPDATE command, this default Secure Channel Sequence Counter is also retrievable using the GET DATA command.

3.3.5 Key Information Template and Key Information Data (tags 'E0' and 'C0')

The Supplementary Security Domain is able to build a Key Information Template containing multiple Key Information Data depending on the available keys.

The Key Information Template is retrievable using the GET DATA command.

3.4 `CVM` Interface

The implementation initially stores the PIN value in BCD format and reserve space for a PIN that could be 12 numerical digits long i.e. stored in 6 bytes.

The following are the initial recommended values:

- The initial length of the PIN is 3 (i.e. number of bytes used to store the PIN).
- The initial value of the PIN is '12 34 5F'.
- The initial Retry Limit is 3.
- The CVM state of the PIN is ACTIVE.

These values can be modified using the `update()` and `setTryLimit()` of the new API.

This guideline assumes that the PIN will always be presented to the card within a PIN data block as defined in EMV Book 1 [9].

4 Key Usage

Each Security Domain requires keys that will facilitate the setting up of a Secure Channel.

For the Issuer Security Domain, this is initially Key Version Number 'FF' which has been deliberately chosen to be outside of the allowable range ('01' to '7F') for a Key Version Number.

It is logical that the initial keys in the Issuer Security Domain be replaced by an initial issuer Key Version Number in the range '01' to '6F'.

Initially and when supported, for a Supplementary Security Domain on an implementation, no keys exist and the Security Domain uses the keys of its associated Security Domain to set up a Secure Channel. This Secure Channel Session is used to populate the Security Domain with its own initial Secure Channel Key Version Number in the range '01' to '6F'.

The 3 DES keys within a Key Version Number ranging from '01' to '6F' and the initial Key Version Number 'FF' have the following functionality:

- The encryption key (Key Identifier '01') is used to generate an encryption session key that is then used both for mutual authentication and, depending on the security level, to decrypt the complete APDU command data field.
- The MAC key (Key Identifier '02') is used to generate a MAC session key that is then utilized to generate the card challenge value and to verify a MAC for APDU commands (command header and command data).
- The data encryption key (DEK) (Key Identifier '03') is used to generate a data encryption session key (DEK session key) that is utilized to decrypt secret or application specific data.

Key Version Numbers '70' to '72' and '74' to '7F' are reserved for future use.

On an implementation supporting Supplementary Security Domains, the RSA public key with a Key Version Number '73' and a Key Identifier of '01' has the following functionality in a Supplementary Security Domain with the DAP Verification privilege:

- The DAP Verification key is used to verify the signature of a Load File Data Block generated by an Application Provider or Controlling Authority.

Management of any of the above keys is possible using either a PUT KEY command or the STORE DATA command.

5 Secure Channel

This section describes the generic recommendations of a Security Domain when processing a command within a Secure Channel Session (SCP '02', implementation option '55').

Within a Secure Channel, the mandatory level of security of each command is dependent on the security level defined in the EXTERNAL AUTHENTICATE command. If the security level mandates secure messaging the class byte for each command received by a Security Domain or passed to a Security Domain for processing should also reflect the presence of secure messaging i.e. bit 3 of the class byte is set.

It is the responsibility of the Security Domain to “unwrap” the command i.e. the Security Domain decrypts the data field (if necessary) and verifies the MAC.

Any command within a Secure Channel Session, immaterial of the security level defined in the EXTERNAL AUTHENTICATE command, may also contain data known by the Security Domain to be encrypted secret data (i.e. the PUT KEY (DES) command or a STORE DATA command containing DES keys). The Security Domain is also responsible for decrypting this data.

Regardless of the number of available logical channels only one Secure Channel Session is active at any one time. No Application, other than the Application context that initiated the Secure Channel, will be able make use of the Secure Channel.

5.1 Data Field Decryption

If so indicated by the security level, data field decryption is performed using the Secure Channel encryption session key as specified in Appendix E.4.6 of the GP CS 2.1.1 [1].

The format of the resulting decrypted data field is verified by checking the padding in the last block of the data field i.e. the block should contain an '80' possibly followed by up to 7 '00's. If the format of the data field is invalid, a response of '6982' is returned and the Secure Channel terminates i.e. the card challenge, session keys and any other data relating to the Secure Channel Session are cleared.

5.2 MAC Verification

If so indicated by the security level, MAC verification is performed using the Secure Channel MAC session key as specified in Appendix E.4.4 of the GP CS 2.2.1 [1] and specifically on the modified APDU as defined in Figure E-4 of [1]. Except for the MAC of the EXTERNAL AUTHENTICATE command, that uses an ICV of binary zeroes, the MAC on all other commands uses an ICV that is the encrypted MAC of the previous command as specified in Appendix E.3.4 of the GP CS 2.2.1 [1]. If the verification of the MAC fails, a response of '6982' is returned and the Secure Channel terminates i.e. the card challenge, session keys and any other data relating to the Secure Channel are cleared.

5.3 Decryption using DEK

The presence of known encrypted secret data within the data field of a command is application dependant. A Security Domain decrypts DES key data when processing the PUT KEY (DES keys) command or when processing a STORE DATA command containing DES keys.

Decryption of the key is performed using the data encryption session key (DEK session key) as defined in Appendix E.4.7 of the GP CS 2.2.1 [1] and specifically the mechanism defined in Appendix B.1.1.2 of [1].

6 APDU Commands

All APDU commands are received first by the JCRE/OPEN for dispatching to Applications. Except for the SELECT and MANAGE CHANNEL commands whose behavior is described later (see sections 6.11 and 6.8, below) and the two T=0 protocol specific commands whose behavior is described below, all other APDU commands are dispatched immediately to the currently selected Application.

If the card is in the Life Cycle State TERMINATED, any Application command received is immediately dispatched to the Issuer Security Domain. It is the responsibility of the Issuer Security Domain to reject all commands except the GET DATA command when the card Life Cycle State is TERMINATED.

Support of logical channels (defined in ISO/IEC 7816-4 [8]) is recommended. Implementations may support only the required Basic Logical Channel or 1, 2 or 3 Supplementary Logical Channels in addition to the required Basic Logical Channel. Implementations are also recommended to support the assignment of logical channels numbers by the card therefore if the logical channel bits of the class byte (bits 1 and 2) indicate a supported Supplementary Logical Channel that is not open, a response of '6881' is returned.

For cards implementing the T=0 protocol, the JCRE/OPEN is responsible for returning a '61xx' return code and managing the subsequent GET RESPONSE command i.e. the GET RESPONSE command, immaterial of its level of security, is not dispatched to the currently selected application but is managed by the JCRE/OPEN itself.

Again for cards implementing the T=0 protocol, the JCRE/OPEN is responsible for returning a '6Cxx' return code and managing the subsequent re-issue of the case 2 command i.e. the re-issued case 2 command is not dispatched to the currently selected application but is managed by the JCRE/OPEN itself (see section 1.4).

The commands described in this guideline are the followings:

- EXTERNAL AUTHENTICATE
- DELETE
- GET DATA
- GET STATUS
- INITIALIZE UPDATE
- INSTALL [for extradition]
- INSTALL [for install and make selectable]
- INSTALL [for load]
- INSTALL [for install]
- INSTALL [for make selectable]
- INSTALL [for personalization]
- INSTALL [for registry update]
- LOAD
- PUT KEY (DES)
- PUT KEY (Public Key)
- SELECT
- SET STATUS
- STORE DATA

6.1 DELETE

See section 11.2 of GP CS 2.2.1 [1] for the structure of the DELETE command. Also refer to section 9.5 of [1] for the description of the OPEN relating to content removal.

6.1.1 Definition

A DELETE command is used by the Issuer Security Domain only to delete an Application or an Executable Load File. It cannot be used to delete packages required for the operation of a GlobalPlatform Java Card™ i.e. it is not possible to delete the GlobalPlatform API or Java Card™ API packages immaterial of whether these are present in the GlobalPlatform Registry.

The DELETE command may only be issued within a Secure Channel Session and the level of security for the command is dependent on the security level defined in the EXTERNAL AUTHENTICATE command.

On an implementation supporting Supplementary Security Domains, the Issuer Security Domain is able to delete any Application or Executable Load File immaterial of the Application’s or Executable Load File’s associated Security Domain. Additional restrictions may exist when deleting a Security Domain (see section 6.2.2 below)

6.1.2 Recommendations

If the card Life Cycle State is TERMINATED, a response of '6D00' is returned.

If the card Life Cycle State is CARD_LOCKED, a response of '6985' is returned.

On an implementation supporting Supplementary Security Domains, the DELETE command is able to delete an instance of a Supplementary Security Domain that has the DAP Verification privilege, however, a Security Domain with the Mandated DAP privilege cannot be deleted.

The DELETE command is able to delete an instance of an Application that is not referenced by any other Application instance on the card.

The DELETE command is able to delete an Executable Load File including all the Executable Modules as long as:

- Any Applications instantiated from the Executable Load File have been previously deleted;
- An Application instance does not reference code in the Executable Load File to be deleted; and
- Another Executable Load File does not reference code in the Executable Load File to be deleted.

The DELETE command is able to delete an Executable Load File and all related Applications as long as:

- An unrelated Application instance does not reference any of the related Applications to be deleted; and
- An unrelated Application instance does not reference code in the Executable Load File to be deleted; and
- Another Executable Load File does not reference code in the Executable Load File to be deleted.

Class Byte

If the class byte excluding the Logical Channel number is not '80' or '84', a response of '6E00' is returned.

Reference Control Parameter 1

If a value other than '00' is present in this parameter, a response of '6A86' is returned.

Reference Control Parameter 2

If a value other than '00' or is '80' present in this parameter, a response of '6A86' is returned.

Data Field Sent in the Command Message

If the data field is not formatted correctly, a response of '6A80' is returned.

If the AID is not present as an Application or Executable Load File in the GlobalPlatform Registry, a response of '6A88' is returned.

If the AID references the Issuer Security Domain, a response of '6A88' is returned.

If the tags 'B6' or '9E' are present, a response of '6A88' is returned.

If the Application or Executable Load File is still being referenced, a response of '6985' is returned.

On an implementation supporting Supplementary Security Domains, if an attempt is being made to delete an instance of a Supplementary Security Domain and the Supplementary Security Domain has the DAP Verification privilege (bit 7 of the privileges byte set), the Security Domain is not deleted and a response of '6985' is returned.

If an Application is being deleted and this Application has the Card Reset privilege, the Issuer Security Domain becomes the application with the Card Reset privilege i.e. the implementation sets the Issuer Security Domain's Application Privileges to '9E DE 00' if the Issuer Security Domain implements support for personalization (i.e. it has the Trusted Path privilege) otherwise to '9E 5E 00'.

If an Executable Load File is being deleted and this Executable Load File resides in Immutable Persistent Memory, it can obviously not be deleted and as such only the entries within the GlobalPlatform Registry are deleted (i.e. the Executable Load File entry and each related Executable Module entry).

If an Executable Load File is being deleted and this Executable Load File resides in Mutable Persistent Memory, all the related application code is deleted as well as the entries within the GlobalPlatform Registry (i.e. the Executable Load File entry and each related Executable Module entry).

If an Application is being deleted, all Application data (of the Application instance being deleted) residing in Mutable Persistent Memory is deleted as well as the Application's entry within the GlobalPlatform Registry.

Each implementation makes its best effort to make space freed up due to deletion re-usable.

Data Field Returned in the Response Message

The data field returned in the response message contains a value of '00'.

6.2 EXTERNAL AUTHENTICATE

See Appendix E.5.2 of the GP CS 2.2.1 [1] for the structure of the EXTERNAL AUTHENTICATE command.

6.2.1 Definition

The EXTERNAL AUTHENTICATE command is used by a Security Domain to authenticate the host and to determine the level of security required for all subsequent commands.

A previous and successful execution of the INITALIZE UPDATE command is necessary prior to processing this command.

6.2.2 Recommendations

For the Issuer Security Domain, if the card Life Cycle State is TERMINATED, a response of '6D00' is returned.

If a Supplementary Security Domain has not yet been populated with its own Secure Channel Key Version Number it uses the services of its associated Security Domain (i.e. the Issuer Security Domain) to process this command. This behavior is further described in section 7.2.1.

If a previous INITALIZE UPDATE command has not been received, a response of '6985' is returned.

Class Byte

If the class byte excluding the Logical Channel number is not '84', a response of '6E00' is returned.

Reference Control Parameter 1 (security level)

This parameter defines the level of security required for all commands following this EXTERNAL AUTHENTICATE command (it does not apply to this command) and within this Secure Channel Session.

This may be an addition to the security required depending on the state of the card i.e. a host (issuer, Application Provider or Controlling Authority) may require a higher level of security than that defined above in section 2.

The following rules apply if a Security Domain is the selected Application:

- '00' – All subsequent commands received by the Security Domain will not include any security i.e. no MAC of the command header and data field nor any encryption of the data field. This option is only applicable if the card is in the Life Cycle States OP_READY or INITIALIZED. If the card is not in either of these states, a response of '6985' is returned.
- '01' – All subsequent commands received by the Security Domain must contain an 8-byte MAC of the command header and data field. If the card is in the Life Cycle State OP_READY or INITIALIZED, this setting indicates an additional level of security. If the card is in the Life Cycle State SECURED or CARD_LOCKED, this level of security is recommended.
- '03' – All subsequent commands received by the Security Domain must contain an 8 byte MAC of the command header and data field and the data field must be encrypted. This setting indicates an additional level of security if the card is in the Life Cycle State SECURED or CARD_LOCKED and 2 additional levels of security if the card is in the Life Cycle State OP_READY or INITIALIZED.

If the value in this field indicates a security level other than those defined above or defined in Appendix E.5.2 of the GP CS 2.2.1 [1] but not supported by the implementation, a response of '6A86' is returned and the Secure Channel Session terminates i.e. the card challenge, session keys and any other data relating to the Secure Channel Session are cleared. A new INITIALIZE UPDATE command would have to be issued to the card prior to attempting another EXTERNAL AUTHENTICATE command.

Reference Control Parameter 2

If a value other than '00' is present in this parameter, a response of '6A86' is returned.

Data Field Sent in the Command Message

If MAC verification (see section 5.2) fails, a response of '6982' is returned.

Once the command MAC has been verified, the host cryptogram is verified. This cryptogram has been generated by the host in order for the card to authenticate that the host is able to generate the same Secure Channel encryption session key as the card. A cryptogram is generated for comparison with the received host cryptogram using the Security Channel encryption session key as described in Appendix E.4.2.2 of the GP CS 2.2.1 [1]. If the comparison fails, a response of '6300' is returned.

In either of the above error cases the Secure Channel Session terminates i.e. the card challenge, session keys and any other data relating to the Secure Channel Session are cleared. A new INITIALIZE UPDATE command would have to be issued to the card prior to attempting another EXTERNAL AUTHENTICATE command.

Once the MAC and host cryptogram have been verified, mutual authentication between the card and the host has occurred. All subsequent commands will be assumed to conform to the security level requirements.

The Secure Channel Sequence Counter of the utilized Key Version Number is incremented by 1.

6.3 GET DATA

See section 11.3 of the GP CS 2.2.1 [1] for the structure of the GET DATA command.

6.3.1 Definition

The GET DATA command is used to retrieve information pertaining to the keys present within a Security Domain and to retrieve tagged data elements from the Data Store of a Security Domain.

If the GET DATA command is issued within a Secure Channel Session, the level of security for the command is dependent on the security level defined in the EXTERNAL AUTHENTICATE command. Note that the GET DATA command is not issued within a Secure Channel Session.

6.3.2 Recommendations

The GET DATA command is the only command that can be processed by the Issuer Security Domain when the card is in the TERMINATED Life Cycle State.

The GET DATA command is the only command besides the INITIALIZE UPDATE command that is fully processed immediately following the selection of the Security Domain.

A Security Domain only returns data that has been populated i.e. tagged data elements that have been previously written to the Security Domain's Data Store or information pertaining to keys that have been populated using the PUT KEY or the STORE DATA commands.

Class and Instruction Bytes

If the class byte excluding the Logical Channel number is not '00', '80' or '84', a response of '6E00' is returned.

If the instruction byte is 'CB', a response of '6D00' is returned.

Reference Control Parameters 1 and 2

If these parameters are not '00E0' or '00C1' a search is conducted through the Data Store for the data identified by the tags contained in these parameters. If these parameters do not reference tags of data currently in the Data Store, a response of '6A88' is returned. (Note that for the Issuer Security Domain, parameters '9F66', '9F67', '00CD' and '004F' cause a response of '6A88' to be returned, as these values are never stored as individual objects in the Data Store).

Data field sent in the command message

No data field is present.

Data Field Returned in the Response Message

The information is output in the correct format dependant on the class byte (ISO or proprietary).

The response message to an ISO type command only contains the value of the data identified in parameters 1 and 2. (If this value is retrieved from the Data Store, the number of bytes returned is the actual length).

The response message to a GlobalPlatform type command contains the tag, length and value (TLV) of the data identified in parameters 1 and 2. (If this value is retrieved from the Data Store, the length field is the actual length).

If the tag is 'C1', the Secure Channel Sequence Counter of the default Key Version Number is returned. If a Supplementary Security Domain has not yet been populated with keys or the Secure Channel keys have not yet been used, the response is either 'C1020000' or '0000' dependant on the class byte.

If the tag has a value of 'E0', the data pertaining to each key (starting with the default Key Version Number) in the Security Domain is returned. On an implementation supporting Supplementary Security Domains, if a Supplementary Security Domain has not yet been populated with keys, the response is either 'E002C000' or 'C000', depending on the class byte.

The following tables contain two examples of a response message for a tag of 'E0':

An Issuer Security Domain that has two Secure Channel keys (Key Version Numbers 1 and 4):

Value in Hex	Description
E0	Key Information Template (Conditional according to proprietary class byte)
24	Length (Conditional according to proprietary class byte)
C0	Key information data
04	Length
01	Key Identifier
01	Key Version Number (default)
80	Key type (DES – mode implicitly known)
10	Key length
C0	Key information data
04	Length
02	Key Identifier
01	Key Version Number (default)
80	Key type (DES – mode implicitly known)
10	Key length
C0	Key information data
04	Length
03	Key Identifier
01	Key Version Number (default)
80	Key type (DES – mode implicitly known)
10	Key length
C0	Key information data
04	Length
01	Key Identifier
04	Key Version Number
80	Key type (DES – mode implicitly known)
10	Key length

C0	Key information data
04	Length
02	Key Identifier
04	Key Version Number
80	Key type (DES – mode implicitly known)
10	Key length
C0	Key information data
04	Length
03	Key Identifier
04	Key Version Number
80	Key type (DES – mode implicitly known)
10	Key length

Table 3: Example Response 1

A Supplementary Security Domain that has 1 Secure Channel key (Key Version Number 2) and a DAP Verification key (Key Version Number '73'):

Value in Hex	Description
E0	Key Information Template (Conditional according to proprietary class byte)
1A	Length (Conditional according to proprietary class byte)
C0	Key information data
04	Length
01	Key Identifier
02	Key Version Number (default)
80	Key type (DES – mode implicitly known)
10	Key length
C0	Key information data
04	Length
02	Key Identifier
02	Key Version Number (default)
80	Key type (DES – mode implicitly known)
10	Key length
C0	Key information data
04	Length
03	Key Identifier
02	Key Version Number (default)
80	Key type (DES – mode implicitly known)
10	Key length
C0	Key information data
06	Length
01	Key Identifier
73	Key Version Number
A1	Key type (RSA Public Key – modulus)
80	Key length
A0	Key type (RSA Public Key – public exponent)
01 or 03	Exponent length

Table 4: Example Response 2

6.4 GET STATUS

See section 11.4 of the GP CS 2.2.1 [1] for the structure of the GET STATUS command.

6.4.1 Definition

The GET STATUS command is used by the Issuer Security Domain only to retrieve Executable Load File, Executable Module, Issuer Security Domain and Application Life Cycle data.

The GET STATUS command may only be issued within a Secure Channel Session and the level of security for the command is dependent on the security level defined in the EXTERNAL AUTHENTICATE command.

6.4.2 Recommendations

If the card Life Cycle State is TERMINATED, a response of '6D00' is returned.

This command only returns Executable Load Files, Executable Modules and Applications as defined in the GlobalPlatform specifications. It is not intended as a means to return information regarding the available API packages. Information relating to the API packages is not returned in the response to this command.

Class Byte

If the class byte excluding the Logical Channel number is not '80' or '84', a response of '6E00' is returned.

Reference Control Parameter 1

If a value other than '80', '40', '20' or '10' is received in this parameter and is not supported by the implementation, a response of '6A86' is returned.

Implementations that do not support the retrieval of data relating to Executable Load Files and their Executable Modules (value of '10' in this parameter) may return a response of '6A81'.

Reference Control Parameter 2

A value '00' for this parameter indicates the deprecated response data structure. A value of '02' indicates the response data TLV structure. If the previously issued GET STATUS command returned a response of '6310' this value is '01' or '03' respectively. If neither of the preceding conditions is met, a response of '6A86' is returned.

Data Field Sent in the Command Message

If the data field is not formatted correctly, a response of '6A80' is returned.

If the value of P1 is '80', the Issuer Security Domain is located.

If the value of P1 is '40', a sequential search through the GlobalPlatform Registry for each Application (includes each instance of the Supplementary Security Domain but always excludes the Issuer Security Domain) that fully or partially matches the AID, is performed. If no entries match the AID, a response of '6A88' is returned. If the data field contains '4F00' and no Applications besides the Issuer Security Domain exist in the GlobalPlatform Registry, a response of '6A88' is returned.

If the value of P1 is '20', a sequential search through the GlobalPlatform Registry for each Executable Load File that fully or partially matches the AID is performed. If no entries match the AID, a response of '6A88' is returned. If the data field contains '4F00' and no Executable Load File exists in the Open Platform Registry, a response of '6A88' is returned.

If the value of P1 is '10', a sequential search through the GlobalPlatform Registry for each Executable Load File that fully or partially matches the AID is performed. If no entries match the AID, a response of '6A88' is returned. If the data field contains '4F00' and no Executable Load File exists in the GlobalPlatform Registry, a response of '6A88' is returned. For each Executable Load File located, its linked Executable Modules are also located.

The implementation may support tag list (tag '5C') according to section 11.4.2.3 of the GP CS 2.2.1 [1]. If the tag list is present and the bit 2 of P2 is not set to 1, a response of '6A86' is returned. If an on-card entity that matches the search criteria is found in GlobalPlatform Registry without the corresponding data object, a response of '6A88' is returned.

Data Field Returned in the Response Message

All matching entries located in the GlobalPlatform Registry is returned.

If the GET STATUS command request requires more Executable Load Files, Executable Modules or Applications than will fit within a single APDU command, following the data that fits within the single APDU, a warning response of '6310' is returned.

The data structure contains the entries as they were loaded or installed on the card i.e. the first Executable Load File returned should be that of the first Load File Data Block loaded to the card or present in Mutable Persistent Memory and the first Application should be the first Application installed on the card. In this manner the structure could be viewed as a history of the loading and installation of Load File Data Blocks and Applications to the card.

6.5 INITIALIZE UPDATE

See Appendix E.5.1 of the GP CS 2.2.1 [1] for the structure of the INITIALIZE UPDATE command.

6.5.1 Definition

The INITIALIZE UPDATE command is used to exchange Secure Channel Session data between the card and the host. This data facilitates the generation of the session keys used for the duration of the Secure Channel Session. The INITIALIZE UPDATE command is only successfully processed if no Secure Channel Session is currently active or if the Secure Channel Session is active on the same logical channel that this command is being issued on.

Information relating to a Secure Channel Session is discarded and the Secure Channel Session terminated for any one of the following reasons:

- Loss of power or card reset.
- If the command immediately following this INITIALIZE UPDATE command on this logical channel is not an EXTERNAL AUTHENTICATE command.
- Application selection. The Security Domain code uses an implementation specific mechanism to close an existing Secure Channel Session when a change of the current applet context occurs on the logical channel on which the Secure Channel Session is active i.e. if the Application invoking a method in the `SecureChannel` class is not the same Application that initiated the setting up of a Secure Channel, the current Secure Channel is either already terminated or is terminated at this point. This is over and above the responsibility of an Application that uses the services of its associated Security Domain to invoke the `closeSecureChannel()` or `resetSecurity()` methods within its own `applet.deselect()` method.
- Secure Channel failure. Any error thrown by the Security Domain on the logical channel on which the Secure Channel Session is active indicates a termination of the current Secure Channel Session. These errors are caused by:
    - The inability of the Security Domain to verify the host cryptogram of the EXTERNAL AUTHENTICATE command.
    - The inability of the Security Domain to verify a MAC of any command received within the Secure Channel Session.
    - The Security Domain determining that padding resulting from command data field decryption is not correct.
    - The Security Domain receiving a message that did not have the required level of security.

  All of these occurrences are further detailed below in sections 7.2.1 and 7.2.5. While no application specific errors causes a Secure Channel Session to terminate, an application using the services of its associated Security Domain may terminate the session at any time by invoking the `closeSecureChannel()` or `resetSecurity()` methods. The Secure Channel does not fail if an Application on another logical channel attempts to initiate a new Secure Channel or attempts to use the Secure Channel even though the attempt will be rejected.
- Receipt of a subsequent INITIALIZE UPDATE command on the logical channel on which the Secure Channel is active. At any point within a current Secure Channel Session, the INITIALIZE UPDATE command can be issued to the card on the same logical channel on which the current Secure Channel was initiated in order to initiate a new Secure Channel Session.

6.5.2 Recommendations

For the Issuer Security Domain, if the card Life Cycle State is TERMINATED, a response of '6D00' is returned.

For an implementation supporting Supplementary Security Domains, if a Supplementary Security Domain has not yet been populated with its own Secure Channel Key Version Number, it uses the services of its associated Security Domain (i.e. the Issuer Security Domain) to process this command. This behavior is further detailed in section 7.2.1.

On an implementation supporting Supplementary Security Domains, the Security Domain code has a mechanism of identifying whether another instance of a Security Domain, selected on another logical channel, has already initiated a Secure Channel. If a Secure Channel is currently active on a logical channel other than the logical channel on which this command was issued, a response of '6985' is returned.

Command Class Byte

If the class byte excluding the Logical Channel number is not '80', a response of '6E00' is returned.

Reference Control Parameter 1

If the Security Domain has more than one Key Version Number, this field may indicate the keys within a particular Key Version Number ('01' to '6F') to be used to create session keys.

A value of '00' in this parameter indicates that the default Key Version Number is used. (For more information on the default key, see sections 6.9 or 6.13.)

If the value in this field has a value greater than '6F', a response of '6A86' is returned.

If the value indicates a Key Version Number that is not present within the Security Domain or indicates a Key Version Number that is incomplete (i.e. the Key Version Number does not contain Key Identifiers 1, 2 and 3), a response of '6A88' is returned.

Reference Control Parameter 2

If a value other than '00' is present in this parameter, a response of '6A86' is returned.

Data Field Sent in the Command Message

The (C-)MAC, encryption and DEK session keys are generated as described in Appendix E.4.1 of the GP CS 2.2.1 [1].

If the received host challenge is identical to the concatenation of the Secure Channel Sequence Counter of the identified Key Version Number and the card challenge (6 bytes generated as described in section 1.1), a response of '6982' is returned.

A card cryptogram is generated for the response message using the Secure Channel encryption session key as described in Appendix E.4.2.1 of [1].

Data Field Returned in the Response Message

The 10-byte value of tag 'CF' from the Security Domain’s Data Store is returned as Key Derivation Data.

The key information data includes the Key Version Number and the Secure Channel Protocol ('02'). If a Key Version Number was specified in the command data, then that same Key Version Number is returned in the response, else the Key Version Number chosen (default) by the Security Domain is returned.

The Sequence Counter is the Secure Channel Sequence Counter of the identified Key Version Number.

The card cryptogram is the cryptogram as described in Appendix E.4.2.1 of the GP CS 2.2.1 [1].

6.6 INSTALL

See section 11.5 of the GP CS 2.2.1 [1] for the structure of the INSTALL command.

6.6.1 Definition

The INSTALL command is used to inform the Issuer Security Domain of the various steps required to load application code and to install and make an Application selectable. On an implementation supporting Supplementary Security Domains, the INSTALL command can also be used to inform the Issuer Security Domain to extradite an Application from the Issuer Security Domain to a Supplementary Security Domain within the card.

The INSTALL command may only be issued within a Secure Channel Session and the level of security for the command is dependent on the security level defined in the EXTERNAL AUTHENTICATE command.

On an implementation with support for Supplementary Security Domains, the Issuer Security Domain installs and makes selectable any Application immaterial of the Application’s associated Security Domain.

6.6.2 Recommendations

6.6.2.1 General Recommendations

If the card Life Cycle State is TERMINATED, a response of '6D00' is returned.

If the card Life Cycle State is CARD_LOCKED, a response of '6985' is returned.

For all implementations, the Issuer Security Domain supports INSTALL [for load] and INSTALL [for install and make selectable] commands and may support INSTALL [for install], INSTALL [for make selectable] and INSTALL [for personalization] commands. INSTALL [for extradition] is supported on an implementation supporting Supplementary Security Domains.

For an implementation supporting Supplementary Security Domains, a Supplementary Security Domain may support INSTALL [for personalization] command.

Class Byte

If the class byte excluding the Logical Channel number is not '80' or '84', a response of '6E00' is returned.

Reference Control Parameter 1

The following are the allowable values of this parameter:

'02' – Indicates that an attempt is being made to load application code to the card.

'0C' – Indicates that an attempt is being made to install and make selectable an instance of previously loaded application code.

'10' – For an implementation supporting Supplementary Security Domains, this indicates that an attempt is being made to extradite an Application associated with the Issuer Security Domain to a Supplementary Security Domain.

If INSTALL [for install] supported then '04' – Indicates that an attempt is being made to install an instance of previously loaded application code.

If INSTALL [for make selectable] supported then '08' – Indicates that an attempt is being made to make selectable an instance of previously installed application.

If INSTALL [for install] is supported then INSTALL [for make selectable] is supported too.

If INSTALL [for make selectable] is supported then INSTALL [for install] is supported too.

If INSTALL [for personalization] supported then '20' – Indicates that an attempt is being made to the currently selected Security Domain to personalize one of its associated Applications and a subsequent STORE DATA command is to be expected. See section 11.11 of the GP CS 2.2.1 [1] for the structure of the subsequent STORE DATA commands.

This mechanism (i.e. the INSTALL [for personalization] command and the subsequent STORE DATA commands) may be supported by implementations supporting Supplementary Security Domains to personalize applications only.

An implementation without support for Supplementary Security Domains supports a value of '02' or '0C' and may support a value of '04', '08' or '20' in this parameter. Any other value causes a response of '6A86' to be returned.

An implementation supporting Supplementary Security Domains:

- the Issuer Security Domain supports a value of '02', '0C' or '10' and may support a value of ‘04’, ‘08’ or '20' in this parameter. Any other value causes a response of '6A86' to be returned.
- a Supplementary Security Domain may support a value of '20' in this parameter. Any other value causes a response of '6A86' to be returned.

Reference Control Parameter 2

If a value other than '00' is present in this parameter, a response of '6A86' is returned.

Data Field Sent in the Command Message

If the data field is not formatted correctly, a response of '6A80' is returned.

6.6.2.2 INSTALL [for load]

Refer to section 9.3.2 of the GP CS 2.2.1 [1] for the description of the OPEN relating to initiating the loading of application code. Refer to Table 11-42 of [1] for the data field structure of this command.

The following recommendations exist for these fields:

Load File AID
The Load File AID is the Java Card™ name of the package present within the header component of the '.cap' file. The Load File AID cannot already be present within the GlobalPlatform Registry as another Load File AID, as a Module AID or as an Application instance AID. If this AID is already present, a response of '6985' is returned.

On completion of a successful load process, this AID will be used in future to reference the Executable Load File.

Security Domain AID
For an implementation without support for Supplementary Security Domains, the presence of the Security Domain AID is optional within the command data. If a value is not present then the Issuer Security Domain AID is the Security Domain associated to the Executable Load File. If a value is present but does not reference the Issuer Security Domain, a response of '6985' is returned.

For an implementation supporting Supplementary Security Domains, the presence of the Security Domain AID is optional within the command data. If a value is not present then the Issuer Security Domain AID is the Security Domain associated to the Executable Load File. If a value is present then it references the Issuer Security Domain or any Supplementary Security Domain within the GlobalPlatform Registry that is in the Life Cycle State PERSONALIZED. If the value does not reference a Security Domain within the GlobalPlatform Registry, a response of '6A88' is returned. If this value references a Supplementary Security Domain that is not in the Life Cycle State PERSONALIZED, a response of '6985' is returned.

Load File Data Block Hash
In an implementation does not support Supplementary Security Domains, the content of this field is ignored.

For an implementation supporting Supplementary Security Domains, the presence of the Load File Data Block Hash is conditional on a DAP Block being required in the first LOAD command. If present, this hash is retained and verified on receipt of the full Load File. See the description in section 6.7 , below, for verifying this hash. The conditions are:

- If a Supplementary Security Domain in the Life cycle State PERSONALIZED exists in the GlobalPlatform Registry with bit 1 of its privilege byte set (mandates DAP Verification), a Load File Data Block Hash is present.
- If the Supplementary Security Domain to be associated to the resultant Executable Load File has bit 7 of it privileges byte set (requires DAP Verification), a Load File Data Block Hash is present.

Load Parameters
An implementation supports Load Parameters Length coded in 1 byte and may support it in 2 bytes.

An implementation may require that TLV coded System Specific Parameters be present.

If an implementation does not require the presence of load parameters, the implementation ignores these load parameters if they are present.

If an implementation does require the presence of load parameters tags 'C6', 'C7' and 'C8', the implementation abides by the following rules.

- If the load parameters are not present, a response of '6985' is returned.
- For each tag that contains a value of 'FFFF', the implementation is not check for the available space.
- For each tag that contains a value other than 'FFFF', the behavior of the implementation is outside the scope of this document i.e. this guideline has no recommendations as to how these values are used.

If the load parameters tag 'D6' is present but is not supported by the implementation, a response of '6A80' is returned.

If an implementation supports the presence of load parameters tag 'D6', the implementation abides by the following rules.

- The command is not rejected if the load parameter is not present.
- If the requested space is not available, a response of '6985' is returned.
- If the requested space is available the requested code memory space is reserved for this Load File.

If the tag 'B6' is present, a response of '6A80' is returned.

Load Token
The content of this field is not present. If the Load Token field contains a value, a response of '6A80' is returned.

The successful processing of all the above fields has no lasting effect on the card at all. This command is only a precursor to a sequence of LOAD commands. The Load File AID, Security Domain AID and Load File Data Block hash is however retained, as they are needed for processing during or on completion of the receipt of the LOAD commands.

6.6.2.3 INSTALL [for install and make selectable]

Refer to section 9.3.3 of the GP CS 2.2.1 [1] for the description of the OPEN relating to installing and making an Application selectable. Refer to Table 11-43 of [1] for the data field structure of this command.

The following recommendations exist for these fields:

Executable Load File AID
The Executable Load File AID indicates within which Executable Load File the applet to be installed resides. If this value does not reference an Executable Load File present within the GlobalPlatform Registry, a response of '6A88' is returned.

Executable Module AID
The Executable Module AID is the Java Card™ name of the applet present within the applet component of an Executable Load File already present on the card. If this value does not reference an Executable Module present within the GlobalPlatform Registry, a response of '6A88' is returned.

Application AID
The Application AID is the AID with which the applet will be instantiated and as such become the AID that will be used to select this Application. It may have the same value as the Executable Module AID but if this AID is already present within the GlobalPlatform Registry as an Application AID or Executable Load File AID, a response of '6985' is returned.

Application Privileges

An implementation supports Application Privileges coded in 1 or 3 bytes. When coded in 1 byte or the leftmost byte when coded in 3 bytes the bitmap value and its implication are defined as follows:

Bit 8 set (Security Domain):

- For an implementation without support for Supplementary Security Domains, if this bit is set, a response of '6A80' is returned.
- For an implementation supporting Supplementary Security Domains, this privilege indicates that the Application being installed is a Security Domain and Applications associated to this Supplementary Security Domain may use its Secure Channel services. If the above Executable Load File AID does not match the AID for the installable Security Domains code defined in section 3, a response of '6985' is returned.

Bit 7 set (DAP Verification):

- For an implementation without support for Supplementary Security Domains, if this bit is set, a response of '6A80' is returned.
- For an implementation supporting Supplementary Security Domains, if this bit is not set in conjunction with bit 8, a response of '6A80' is returned. This privilege indicates that the Application being installed is an Application Provider’s on card Supplementary Security Domain and may, once in the Life Cycle State PERSONALIZED, be requested to perform DAP Verification. As an implementation is only recommended to support the presence of one DAP Block in the Load File, if a Supplementary Security Domain already exists within the GlobalPlatform Registry that has bit 7 of its Application Privileges set, the implementation may reject this INSTALL command with a '6985' response code.

Bit 6 set (Delegated Management):

- If this bit is set, a response of '6A80' is returned.

Bit 5 set (Card Lock):

- This privilege indicates that the Application being installed will have the ability to successfully invoke the `lockCardManager()` or `lockCard()` methods, which will cause the card to be locked.

Bit 4 set (Card Terminate):

- This privilege indicates that the Application being installed will have the ability to successfully invoke the `terminateCardManager()` or `terminateCard()` methods, which will cause the card to reject all commands besides the GET DATA command.

Bit 3 set (Card Reset):

- This privilege indicates that the Application being installed becomes the default selected application and has the privilege to modify historical bytes on one or more card interfaces. If the Issuer Security Domain is not the current application with Card Reset Privilege (i.e. some other Application within the GlobalPlatform Registry has already been installed with this privilege), a response of '6985' is returned.

Bit 2 set (CVM management):

- This privilege indicates that the Application being installed will have the ability to successfully invoke the `setPin()`, `update()`, `blockState()`, `resetAndUnblockState()` and `setTryLimit()` methods which will cause the value of the global PIN or variables pertaining to the management of the Global PIN to change.

Bit 1 set (Mandated DAP Verification):

- For an implementation without support for Supplementary Security Domains, if this bit is set, a response of '6A80' is returned.
- For an implementation supporting Supplementary Security Domains, if this bit is not set in conjunction with bit 8 and bit 7, a response of '6A80' is returned. This privilege indicates that the Application being installed is a Controlling Authorities' on card Supplementary Security Domain and once in the Life Cycle State PERSONALIZED mandates that Load Files contain a DAP Block.

When coded in 3 bytes the middle byte bitmap value and its implication are defined as follows:

Bit 8 set (Trusted Path):

- This privilege indicates that the Application being installed is a Security Domain and supports personalization of its associated applications through Trusted Path using its Secure Channel services. If the Security Domain does not support this function a response of '6A80' is returned.
- If this bit is not set in conjunction with bit 8 of byte 1, a response of '6A80' is returned.

Bit 4 set (Global Lock):

- This privilege indicates that the Application being installed may lock or unlock any other application. If the implementation does not support this option a response of '6A80' is returned.

Bit 3 set (Global Registry):

- This privilege indicates that the Application being installed Application may access any entry in the GlobalPlatform Registry. If the implementation does not support this option a response of '6A80' is returned.

If any other bit is set (bit 7, 6, 5, 2 or 1), a response of '6A80' is returned.

When coded in 3 bytes the rightmost byte is not set to '00', a response of '6A80' is returned.

Install and Make Selectable Parameters

An implementation supports Install Parameters Length coded in 1 byte and may support it in 2 bytes.

The install parameters are parameters that are specific to the system and the Application being installed. Install parameters are TLV structured data and at least contains the tag 'C9'.

Application Specific Parameters (tag 'C9') are outside the scope of this document except in the case of a Security Domain being installed i.e. if it is intended that the Security Domain be capable of accepting extradition requests, this field contains 'C90145'.

An implementation may require that TLV coded System Specific Parameters be present.

If an implementation does not require the presence of install parameters tags 'C7' and 'C8', the implementation ignores these install parameters if they are present.

If an implementation does require the presence of install parameters tags 'C7' and 'C8', the implementation abides by the following rules.

- If the install parameters are not present, a response of '6A80' is returned.
- For each tag that contains a value of 'FFFF', the implementation does not check for space nor it reserves space. The implementation allocates space to the Application on a per use basis.

- For each tag that contains a value other than 'FFFF', the behavior of the implementation is outside the scope of this document this guideline has no recommendations as to how these values are used.

If the tag 'CB' is present, a response of '6A80' is returned.

If the install parameters tag 'D7' or 'D8' is present but is not supported by the implementation, a response of '6A80' is returned.

If an implementation supports the presence of install parameters tags 'D7' and 'D8', the implementation abides by the following rules.

- The command is not rejected if the install parameter is not present.
- If the requested space is not available, a response of '6985' is returned.
- If the requested space is available the requested memory space is reserved for this instance.

The presence of tag 'CA' is ignored.

The presence of tag 'CF' is ignored if not supported by the implementation.

If an implementation supports the presence of install parameters tag 'CF', the implementation abides by the following rules in addition to the rules defined in section 11.1.7 of the GP CS 2.2.1 [1]. For each tag 'CF' the bitmap value and its implication are defined as follows:

Bit 8 set (Contactless I/O):

- The application becomes implicitly selectable on contactless interface and specifically on the logical channel defined by bits 1 and bit 2. If an application other than Issuer Security Domain is already the implicitly selectable application on contactless interface and specifically on the logical channel defined by bit 1 and bit 2, a response of '6985' is returned.
- If the contactless interface or the logical channel defined by bit 1 and bit 2 is not supported then this parameter is ignored.

Bit 7 set (Contact I/O):

- The application becomes implicitly selectable on contact interface and specifically on the logical channel defined by bits 1 and bit 2. If an application other than Issuer Security Domain is already the implicitly selectable application on contact interface and specifically on the logical channel defined by bit 1 and bit 2, a response of '6985' is returned.
- If the contact interface or the logical channel defined by bit 1 and bit 2 is not supported then this parameter is ignored.

Bit 6 set (RFU):

- The content of this bit is ignored.

Bit 5 to bit 3 set (Logical Channel number 4 to 19):

- The contents of these bits are ignored.

Bit 2 – bit 1 set (Logical Channel number 0 to 3):

- The application becomes implicitly selectable on the logical channel number 0, 1, 2 or 3 as defined by these two bits and on contact or contactless interface depending on bit 8 and bit 7.
- If the logical channel defined by bit 1 and bit 2 is not available then this parameter is ignored.

The presence of tag 'EA' is ignored.

If the tag 'B6' is present, a response of '6A80' is returned.

Install and Make Selectable Token

The content of this field is not present. If the Install Token field contains a value, a response of '6A80' is returned.

The successful processing of the above fields results in the following:

The `install()` method of the application code is invoked.

The parameter `bArray` of the `install()` method contains the concatenation of the following data:

- The length of the Application AID
- The Application AID
- The length of the Application Privileges

- The Application Privileges
- The length of the Application Specific Parameters (byte following the tag 'C9')
- Any Application Specific Parameters if present.

The parameter `bLength` is set to the sum of the above data.

On an implementation supporting Supplementary Security Domains, when this command is being used to install a Supplementary Security Domain, the implementation (e.g. the `install()` method of the installable Security Domain code) checks for the presence of install parameters and:

- If bit 8 of the Application Privileges is not set, an exception of '6985' is returned.
- If Application Specific Parameters do not exist (length of the Application Specific Parameters is '00'), the Supplementary Security Domain always, when requested, rejects extraditions to this Security Domain.
- If Application Specific Parameters do exist (length of the Application Specific Parameters is '01') but the byte following the length is not '45', the Supplementary Security Domain always, when requested, rejects extraditions to this Security Domain.
- If the length of the Application Specific Parameters do exist (length of the Application Specific Parameters is not '00') and the byte following the length is '45', the Supplementary Security Domain always, when requested, accepts extraditions to this Security Domain.

On successful completion of the `install()` method, an entry is created for the Application in the GlobalPlatform Registry and the Life Cycle State of the Application is set to SELECTABLE.

The Application Privileges are linked to this Application. If the Application is to have the Card Rest privilege, the Card Reset privilege is linked to this Application and the implementation sets the bit 3 of byte 1 (leftmost byte) of the Issuer Security Domain’s Application Privileges to 0.

The Security Domain linked to the Executable Load File from which this Application was instantiated is associated to this Application.

If the install parameters within the command data field are coded in one byte the implementation sets the value of byte 2 (middle byte) and byte 3 (rightmost byte) as follows:

- When installing a Supplementary Security Domain and the support for personalization is implemented, to '80 00'. If the support for personalization is not implemented, to '00 00'
- When installing an Application, to '00 00'

6.6.2.4 INSTALL [for extradition]

Refer to section 9.4.1 of the GP CS 2.2.1 [1] for the description of the OPEN relating to extraditing an Application. Refer to Table 11-45 of [1] for the data field structure of this command.

The following recommendations exist for these fields:

Security Domain AID
The Security Domain AID can reference any Supplementary Security Domain within the GlobalPlatform Registry that is in the Life Cycle State PERSONALIZED. If this value does not reference a Security Domain in the GlobalPlatform Registry, a response of '6A88' is returned. If this value references a Supplementary Security Domain that is not in the PERSONALIZED Life Cycle State, a response of '6985' is returned.

A request is to be made to this Security Domain, to ensure that the Security Domain is willing to accept extradition. (A Supplementary Security Domain always rejects extradition unless it was installed with an Application Specific Parameter of '45') If the Supplementary Security Domain refuses to accept extradition, a response of '6985' is returned.

Application AID
The Application AID references an Application within the GlobalPlatform Registry that is associated to the Issuer Security Domain. If this value does not reference an Application in the GlobalPlatform Registry, a response of '6A88' is returned. If this value references a Supplementary Security Domain, a response of '6985' is returned. If this value does not reference an Application associated to the Issuer Security Domain, a response of '6985' is returned.

Extradition Parameters
If any install parameters is present, a response of '6A80' is returned.

Extradition Token
The content of this field is not present. If the Extradition Token field contains a value, a response of '6A80' is returned.

The successful processing of the above fields results in the following:

The above Supplementary Security Domain replaces the Issuer Security Domain as the Application's associated Security Domain.

6.6.2.5 INSTALL [for install]

Refer to section 9.3.3 of the GP CS 2.2.1 [1] for the description of the OPEN relating to installing an Application. Refer to Table 11-43 of [1] for the data field structure of this command. If the data field is not formatted correctly (i.e. missing or incorrect length fields), a response of '6A80' is returned.

The following recommendations exist for these fields:

Executable Load File AID
The Executable Load File AID indicates within which Executable Load File the applet to be installed resides. If this value does not reference an Executable Load File present within the GlobalPlatform Registry, a response of '6A88' is returned.

Executable Module AID
The Executable Module AID is the Java Card™ name of the applet present within the applet component of an Executable Load File already present on the card. If this value does not reference an Executable Module present within the GlobalPlatform Registry, a response of '6A88' is returned.

Application AID
The Application AID is the AID with which the applet will be instantiated and as such become the AID that will be used to select this Application. It may have the same value as the Executable Module AID but if this AID is already present within the GlobalPlatform Registry as an Application AID or Executable Load File AID, a response of '6985' is returned.

Application Privileges

An implementation supports Application Privileges coded in 1 or 3 bytes. When coded in 1 byte or the leftmost byte when coded in 3 bytes the bitmap value and its implication are defined as follows:

Bit 8 set (Security Domain):

- For an implementation without support for Supplementary Security Domains, if this bit is set, a response of '6A80' sis returned.
- For an implementation supporting Supplementary Security Domains, this privilege indicates that the Application being installed is a Security Domain and Applications associated to this Supplementary Security Domain may use its Secure Channel services. If the above Executable Load File AID does not match the AID for the installable Security Domains code defined in section 3, a response of '6985' is returned.

Bit 7 set (DAP Verification):

- For an implementation without support for Supplementary Security Domains, if this bit is set, a response of '6A80' is returned.
- For an implementation supporting Supplementary Security Domains, if this bit is not set in conjunction with bit 8, a response of '6A80' is returned. This privilege indicates that the Application being installed is an Application Provider’s on card Supplementary Security Domain and may, once in the Life Cycle State PERSONALIZED, be requested to perform DAP Verification. As an implementation is only required to support the presence of one DAP Block in the Load File, if a Supplementary Security Domain already exists within the GlobalPlatform Registry that has bit 7 of its Application Privileges set, the implementation may reject this INSTALL command with a '6985' response code.

Bit 6 set (Delegated Management):

- If this bit is set, a response of '6A80' is returned.

Bit 5 set (Card Lock):

- This privilege indicates that the Application being installed will have the ability to successfully invoke the `lockCardManager()` or `lockCard()` methods, which will cause the card to be locked.

Bit 4 set (Card Terminate):

- This privilege indicates that the Application being installed will have the ability to successfully invoke the `terminateCardManager()` or `terminateCard()` methods, which will cause the card to reject all commands besides the GET DATA command.

Bit 3 set (Card Reset):

- If this bit is set, a response of '6A80' is returned.

Bit 2 set (CVM management):

- This privilege indicates that the Application being installed will have the ability to successfully invoke the `setPin()`, `update()`, `blockState()`, `resetAndUnblockState()` and `setTryLimit()` methods which will cause the value of the global PIN or variables pertaining to the management of the Global PIN to change.

Bit 1 set (Mandated DAP Verification):

- For an implementation without support for Supplementary Security Domains, if this bit is set, a response of '6A80' is returned.
- For an implementation supporting Supplementary Security Domains, if this bit is not set in conjunction with bit 8 and bit 7, a response of '6A80' is returned. This privilege indicates that the Application being installed is a Controlling Authorities' on card Supplementary Security Domain and once in the Life Cycle State PERSONALIZED mandates that Load Files contain a DAP Block.

When coded in 3 bytes the middle byte bitmap value and its implication are defined as follows:

Bit 8 set (Trusted Path):

- This privilege indicates that the Application being installed is a Security Domain and supports personalization of its associated applications through Trusted Path using its Secure Channel services. If the Security Domain does not support this function a response of '6A80' is returned.
- If this bit is not set in conjunction with bit 0, a response of '6A80' is returned.

Bit 4 set (Global Lock):

- This privilege indicates that the Application being installed may lock or unlock any other application. If the implementation does not support this option a response of '6A80' is returned.

Bit 3 set (Global Registry):

- This privilege indicates that the Application being installed Application may access any entry in the GlobalPlatform Registry. If the implementation does not support this option a response of '6A80' is returned.
- If any other bit is set, a response of '6A80' is returned.

When coded in 3 bytes the rightmost byte is not set to '00', a response of '6A80' is returned.

Install Parameters

An implementation supports Install Parameters Length coded in 1 byte and may support it in 2 bytes.

The install parameters are parameters that are specific to the system and the Application being installed. Install parameters are TLV structured data and at least contains the tag 'C9'.

Application Specific Parameters (tag 'C9') are outside the scope of this document except in the case of a Security Domain being installed i.e. if it is intended that the Security Domain be capable of accepting extradition requests, this field contains 'C90145'.

An implementation may require that TLV coded System Specific Parameters be present.

If an implementation does not require the presence of install parameters tags 'C7' and 'C8', the implementation ignores these install parameters if they are present and are not supported by the implementation.

If an implementation does require the presence of install parameters tags 'C7' and 'C8', the implementation abides by the following rules.

- If the install parameters are not present, a response of '6A80' is returned.
- For each tag that contains a value of 'FFFF', the implementation does not check for space nor does it reserve space. The implementation allocates space to the Application on a per use basis.
- For each tag that contains a value other than 'FFFF', the behavior of the implementation is outside the scope of this document this guideline has no recommendations as to how these values are used.

If the tag 'CB' is present, a response of '6A80' is returned.

If the install parameters tag 'D7' or 'D8' is present but is not supported by the implementation, a response of '6A80' is returned.

If an implementation supports the presence of install parameters tags 'D7' and 'D8', the implementation abides by the following rules.

- The command is not rejected if the install parameter is not present.
- If the requested space is not available, a response of '6985' is returned.
- If the requested space is available the requested memory space is reserved for this instance.

The presence of tag 'CA' is ignored.

If the tag 'CF' is present, a response of '6A80' is returned.

The presence of tag 'EA' is ignored.

If the tag 'B6' is present, a response of '6A80' is returned.

Install Token
The content of this field is not present. If the Install Token field contains a value, a response of '6A80' is returned.

The successful processing of the above fields results in the following:

The `install()` method of the application code is invoked.

The parameter `bArray` of the `install()` method contains the concatenation of the following data:

- The length of the Application AID
- The Application AID
- The length of the Application Privileges
- The Application Privileges
- The length of the Application Specific Parameters (byte following the tag 'C9')
- Any Application Specific Parameters if present.

The parameter `bLength` is set to the sum of the above data.

On an implementation supporting Supplementary Security Domains, when this command is being used to install a Supplementary Security Domain, the `install()` method of the installable Security Domain code checks for the presence of install parameters and:

- If bit 8 of the Application Privileges is not set, an exception of '6985' is returned.
- If Application Specific Parameters do not exist (length of the Application Specific Parameters is '00'), the Supplementary Security Domain always, when requested, rejects extraditions to this Security Domain.
- If Application Specific Parameters do exist (length of the Application Specific Parameters is '01') but the byte following the length is not '45', the Supplementary Security Domain always, when requested, rejects extraditions to this Security Domain.
- If the length of the Application Specific Parameters do exist (length of the Application Specific Parameters is not '00') and the byte following the length is '45', the Supplementary Security Domain always, when requested, accepts extraditions to this Security Domain.

On successful completion of the `install()` method, an entry is created for the Application in the GlobalPlatform Registry and the Life Cycle State of the Application is set to INSTALLED.

The Application Privileges are linked to this Application.

The Security Domain linked to the Executable Load File from which this Application was instantiated is associated to this Application.

If the install parameters within the command data field are coded in one byte the implementation sets the value of byte 2 (middle byte) and byte 3 (rightmost byte) as follows:

- When installing a Supplementary Security Domain and the support for personalization is implemented, to '80 00'. If the support for personalization is not implemented, to '00 00'
- When installing an Application, to '00 00'

6.6.2.6 INSTALL [for make selectable]

Refer to section 9.3.7 of the GP CS 2.2.1 [1] for the description of the OPEN relating to installing an Application. Refer to Table 11-44 of [1] for the data field structure of this command. If the data field is not formatted correctly (i.e. missing or incorrect length fields), a response of '6A80' is returned.

The following recommendations exist for these fields:

Length
If the Length field contains a value other than '00', a response of '6A80' is returned.

Length
If the Length field contains a value other than '00', a response of '6A80' is returned.

Application AID
The Application AID is the AID with which the applet is instantiated and as such becomes the AID that will be used to select this Application.

If this value does not reference an Application AID in the GlobalPlatform Registry, a response of '6A88' is returned.

If this value does reference an Application AID in the GlobalPlatform Registry but the Application is not in the state `INSTALLED`, a response of '6A80' is returned.

Application Privileges

An implementation supports Application Privileges coded in 1 or 3 bytes. When coded in 1 byte or the leftmost byte when coded in 3 bytes the bitmap value and its implication are defined as follows:

Beside the application Card Reset privilege bit all other application privilege bits is ignored.

Bit 3 set (Card Reset):

- This privilege indicates that the Application being installed for make selectable becomes the default selected application and has the privilege to modify historical bytes on one or more card interfaces. If the Issuer Security Domain is not the current application with Card Reset Privilege (i.e. some other Application within the GlobalPlatform Registry has already been installed with this privilege), a response of '6985' is returned.

Install Parameters
The presence of tag 'CF' is ignored if not supported by the implementation.

If an implementation supports the presence of install parameters tag 'CF', the implementation abides by the following rules in addition to the rules defined in section 11.1.7 of the GP CS 2.2.1 [1]. For each tag 'CF' the bitmap value and its implication are defined as follows:

Bit 8 set (Contactless I/O):

- The application becomes implicitly selectable on contactless interface and specifically on the logical channel defined by bits 1 and bit 2. If an application other than Issuer Security Domain is already the implicitly selectable application on contactless interface and specifically on the logical channel defined by bit 1 and bit 2, a response of '6985' is returned.
- If the contacless interface or the logical channel defined by bit 1 and bit 2 is not supported then this parameter is ignored.

Bit 7 set (Contact I/O):

- The application becomes implicitly selectable on contact interface and specifically on the logical channel defined by bits 1 and bit 2. If an application other than Issuer Security Domain is already the implicitly selectable application on contact interface and specifically on the logical channel defined by bit 1 and bit 2, a response of '6985' is returned.
- If the contact interface or the logical channel defined by bit 1 and bit 2 is not supported then this parameter is ignored.

Bit 6 set (RFU):

- The content of this bit is ignored.

Bit 5 to bit 3 set (Logical Channel number 4 to 19):

- The contents of these bits are ignored.

Bit 2 – bit 1 set (Logical Channel number 0 to 3):

- The application becomes implicitly selectable on the logical channel number 0, 1, 2 or 3 as defined by these two bits and on contact or contactless interface depending on bit 8 and bit 7.
- If the logical channel defined by bit 1 and bit 2 is not available then this parameter is ignored.

If the tag 'B6' is present, a response of '6A80' is returned.

Make Selectable Token
The content of this field is not present. If the Install Token field contains a value, a response of '6A80' is returned.

The successful processing of the above fields results in the following:

The Life Cycle State of the Application in the entry created for the Application in the GlobalPlatform Registry is set to SELECTABLE. If the Application is to have the Card Rest privilege, the Card Reset privilege is linked to this Application and the implementation sets the bit 3 of byte 1 (leftmost byte) of the Issuer Security Domain’s Application Privileges to 0.

6.6.2.7 INSTALL [for personalization]

Refer to section 7.3.3 of the GP CS 2.2.1 [1] for the description of the OPEN relating to personalizing an Application. Refer to Table 11-47 of [1] for the data field structure of this command. If the Security Domain does not have the Trusted Path privilege, a response of '6985' is returned. If the data field is not formatted correctly (i.e. missing or incorrect length fields), a response of '6A80' is returned.

The following recommendations exist for this field:

Application AID
The Application AID references an Application within the GlobalPlatform Registry that is associated to the Security Domain to which this command is submitted. If this value does not reference an Application in the GlobalPlatform Registry, a response of '6A88' is returned. If this value does not reference an Application associated to the Security Domain to which this command is submitted, a response of '6985' is returned.

An implementation supporting Supplementary Security Domains the Issuer Security Domain or a Supplementary Security Domain may support this mechanism (i.e. the INSTALL [for personalization] command and the subsequent STORE DATA commands) to personalize its associated applications.

If an attempt is made to personalize a Supplementary Security Domain that is in the PERSONALIZED state, a response of '6985' is returned.

The successful processing of the above fields results in the following:

The subsequent STORE DATA command(s) would be pre-processed by the Security Domain according to the current Secure Channel prior the command being forwarded to the application being personalized. See section 11.11 of the GP CS 2.2.1 [1] for the structure of the subsequent STORE DATA commands.

6.6.2.8 INSTALL [for registry update]

Refer to section 9.4.2 of the GP CS 2.2.1 [1] for the description of the OPEN relating to registry update. Refer to Table 11-46 of [1] for the data field structure of this command. In this guideline this command is used only to restrict a functionality of OPEN.

The following recommendations exist for these fields:

Application Privileges
If the Application Privileges Length is different of '00' i.e. the Application Privileges field is present, a response of '6A80' is returned.

Registry Update Parameters
The install parameter is specific to the system. Install parameters are TLV structured data and contains the tag 'EF'. Within the tag 'EF' the system parameter 'D9' is present in one byte and lists the function to be disabled. Following indicate the bitmap value of tag 'D9' and its implication:

- Bit 1 may be set to 1, indicating the disabling of the delete function. The successful processing of the above fields results in the immediate and irreversible disablement of the following command for any existing security domains or any security domains instantiated from this point on:
  a. DELETE command
- Bit 2 set to 1, indicating the disabling of the load function. The successful processing of the above fields results in the immediate and irreversible disablement of the following commands for any existing security domains or any security domains instantiated from this point on:
  b. INSTALL [for load] command
  c. LOAD command
- Disabling functions associated to other bitmap values are out of scope of this document.

A request to disable load function that is already disabled is rejected and a response of ‘6985’ is returned.

Registry Update Token
The content of this field is not present. If the Extradition Token field contains a value, a response of '6A80' is returned.

The successful processing of the above fields results in disabling the requested functionality of OPEN.

Data Field Returned in the Response Message

The data field returned in the response message contains a value of '00'.

6.7 LOAD

See section 11.6 of the GP CS 2.2.1 [1] for the structure of the LOAD command. Refer to section 9.3.2 of [1] for the description of the OPEN relating to processing a Load File.

6.7.1 Definition

A sequence of LOAD commands follows a successful INSTALL [for load] command and contains a Load File containing application code and possibly authentication data.

6.7.2 Recommendations

If the card Life Cycle State is TERMINATED, a response of '6D00' is returned.

If the card Life Cycle State is CARD_LOCKED, a response of '6985' is returned.

For an implementation supporting Supplementary Security Domains, if more than one DAP Block is present in the Load File, the behavior of such an implementation is outside the scope of this document.

If a LOAD command is not preceded by a successful INSTALL [for load] command, a response of '6985' is returned.

As the size of the LOAD command is constrained by the maximum size of the command data and maybe the size of the APDU buffer within the card, the DAP Block and Load File Data Block is divided into as many numbered LOAD commands as required. An implementation is capable of managing data that is split across multiple LOAD commands.

Any error returned by this command indicates that the full load sequence has failed i.e. the content of the card is not altered in any way (no trace of the load process remains) and the full sequence will have to be restarted including the INSTALL [for load] command.

Class Byte

If the class byte excluding the Logical Channel number is not '80' or '84', a response of '6E00' is returned.

Reference Control Parameter 1

If a value other than '00' or '80' is present in this parameter, a response of '6A86' is returned.

Reference Control Parameter 2 (block number)

The block number is coded sequentially from '00' to 'FF'.

If a block number is received out of sequence, a response of '6A86' is returned.

Data Field Sent in the Command Message

If the data field does not contain a portion of a correctly formatted Load File, a response of '6A80' is returned. The recommendations listed hereunder relate to the Load File and not to a specific data field within the load sequence.

The following are recommendations only for an implementation supporting Supplementary Security Domains:

- If one of the following 2 conditions are met but no DAP Block is present in the Load File, a response of '6985' is returned:
  - o If a Supplementary Security Domain exists in the GlobalPlatform Registry with bit 1 of its Application Privileges set (mandates DAP Verification), a DAP Block (tag 'E2') precedes the Load File Data Block.

  - If the Supplementary Security Domain to be associated to the resultant Executable Load File (Security Domain AID present in the INSTALL [for load] command and not the Issuer Security Domain) has bit 7 of it Application Privileges set, a DAP Block (tag 'E2') precedes the Load File Data Block.
- If, due to one of the above 2 conditions, a DAP Block is present, then:
  - If the Security Domain AID present in the DAP Block does not match the AID of the Security Domain requiring the DAP Block, a response of '6985' is returned.
  - The Supplementary Security Domain requiring the presence of the DAP Block is requested to verify the Load File Data Block signature and:
    - If the Supplementary Security Domain is not in the Life Cycle State PERSONALIZED, the Security Domain indicates that signature verification failed.
    - Load File Data Block signature verification is performed using Key Version Number '73' of the Security Domain as defined in Appendix C.6.1 of the GP CS 2.2.1 [1].
  - If no hash was present in the INSTALL [for load] command or if the Supplementary Security Domain indicated that signature verification failed, a response of '6982' is returned.

For an implementation supporting Supplementary Security Domains, the TLV structure for the Load File Data Block follows a required DAP Block in the Load File. If no DAP Block is required, the Load File only contains the TLV structure for the Load File Data Block.

For an implementation without support for Supplementary Security Domains, the Load File only contains the TLV structure for the Load File Data Block.

Note: unlike most data in commands that is either less than 128 bytes long or in the LV format as opposed to the TLV format, the content of the Load File Data Block is TLV coded and the length field adheres to TLV coding rules:

- The length field of the DAP Block is 2 bytes long and will contain a value between '818A' and '8195' (1 byte tag and 1 byte length for the Security Domain AID with a length of 5 to 16 bytes plus 1 byte tag and 2 byte length for the Load File Data Block signature with a length of 128 bytes).
- The length field of the Load File Data Block Signature is 2 bytes long and will contain a value of '8180' (length of 128 bytes).
- The length field of the Load File Data Block could theoretically range from 1 to 3 bytes i.e. '01' to '7F' if less than 128 bytes long, '8180' to '81FF' if ranging from 128 bytes to 255 bytes and '820100' onwards if greater than 255 bytes.

Besides the DAP Blocks, if the content of the Load File is not structured according to the suggested order of the JC VM 2.1.1 [3] and JC VM 2.2 [6] specifications, a response of '6A80' is returned.

For an implementation supporting Supplementary Security Domains, as part of the processing performed for the Load File Data Block, if a DAP Block is present within the Load File the hashing of the Load File Data Block is performed according to Appendix C.2 of the JC CS 2.2.1 [1].

On receipt of the last LOAD command in the sequence (P1 set to '80'), the following processing occurs:

For an implementation supporting Supplementary Security Domains that required the presence of a DAP Block, if the hash does not match the hash received in the INSTALL [for load] command, a response of '6982' is returned.

An entry is created for the Executable Load File within the GlobalPlatform Registry.

The Security Domain identified in the INSTALL [for load] processing is associated to the Executable Load File.

For each applet present within an applet component of a '.cap' file, an entry for an Executable Module with this AID is created in the GlobalPlatform Registry.

Data Field Returned in the Response Message

The data field returned in the response message contains a value of '00'.

6.8 MANAGE CHANNEL

See section 11.7 of the GP CS 2.2.1 [1] for the structure of the MANAGE CHANNEL command. Also refer to sections 6.4.2.2 and 6.4.3.2 of [1] for the description of the OPEN relating to MANAGE CHANNEL command processing.

6.8.1 Definition

The MANAGE CHANNEL command is the only command described in this document that is handled exclusively by the OPEN and never forwarded to the Application. The intention of the MANAGE CHANNEL command is to open a Supplementary Logical Channel that is not yet open and close a Supplementary Logical Channel that is currently open. A Supplementary Logical Channel can be used to close itself. As this guideline specifies that the card assign logical channel numbers, using the MANAGE CHANNEL command to specify which logical channel is to be opened is outside the scope of this guideline.

6.8.2 OPEN Recommendations

The runtime environment (combination of OPEN and JCRE) is responsible for determining if a Supplementary Logical Channel can be opened or closed.

If the implementation supports the Basic Logical Channel only, a response of '6881' is returned.

If the card Life Cycle State is TERMINATED, a response of '6D00' is returned (This response is only returned following the closing of the Supplementary Logical Channel indicated in P2).

If the card Life Cycle State is CARD_LOCKED, a response of '6A81' is returned (This response is only returned following the closing of the Supplementary Logical Channel indicated in P2).

If a Supplementary Logical Channel is being opened from the Basic Logical Channel but the Application with the Card Reset privilege (not being the Issuer Security Domain) is already selected on another logical channel and this Application does not implement the `multiSelectable` interface, an error of '6985' is returned.

If a Supplementary Logical Channel is being opened from a Supplementary Logical Channel but the Application currently selected on the Supplementary Logical Channel on which this command was issued does not implement the `multiSelectable` interface, an error of '6985' is returned.

Class Byte

If the class byte excluding the Logical Channel number is not '00', a response of '6E00' is returned.

Reference Control Parameter 1

If a value other than '00' or '80' is present in this parameter, a response of '6A81' is returned.

Reference Control Parameter 2

If P1 contains a value of '80' (indicating that a Supplementary Logical Channel is being closed), and this parameter:

- contains a value of '00' or a value of '04' to 'FF', a response of '6A81' is returned, or
- indicates a logical channel that is not currently open, a response of '6200' is returned.

If P1 contains a value of '00' (indicating that a Supplementary Logical Channel is being opened) and this parameter contains a value other than '00', a response of '6A81' is returned.

If P1 contains a value of '00' (indicating that a Supplementary Logical Channel is being opened) and all supported Supplementary Logical Channels are already in use, a response of '6A81' is returned.

Data Field Returned in the Response Message

The data field returned in the response message only contains a value if P1 and P2 both contained a value of '00'. In this case the number of the Supplementary Logical Channel just opened is returned.

6.9 PUT KEY (DES keys)

See section 11.8 of the GP CS 2.2.1 [1] for the structure of the PUT KEY command.

An implementation supporting Supplementary Security Domains is able to differentiate between a PUT KEY command that contains DES keys and a PUT KEY command that contains a Public Key. This can be achieved through the Key Version Number.

This guideline defines an option that allows DES keys to be populated using the STORE DATA command.

6.9.1 Definition

The PUT KEY command is used to:

- Replace a single key
- Replace multiple keys
- Add one or multiple key(s).

For all Security Domains, these keys will be in the Key Version Number range '01' to '6F'.

The PUT KEY command may only be issued within a Secure Channel Session and the level of security for the command is dependent on the security level defined in the EXTERNAL AUTHENTICATE command.

For an implementation supporting Supplementary Security Domains, a Supplementary Security Domain only uses the Secure Channel services of the Issuer Security Domain to populate its first Secure Channel Key Version Number. Once it has been populated with this Key Version Number, the Secure Channel is set up using the Security Domain’s own Secure Channel keys.

6.9.2 Recommendations

For the Issuer Security Domain, if the card Life Cycle State is TERMINATED, a response of '6D00' is returned.

For the Issuer Security Domain, if the card Life Cycle State is CARD_LOCKED, a response of '6985' is returned.

For an implementation supporting Supplementary Security Domains, if a Supplementary Security Domain has not yet been populated with its own Key Version Number, the Secure Channel Session will have been set up using the services of the Security Domain’s associated Security Domain (i.e. the Issuer Security Domain) and therefore the Issuer Security Domain will also manage the level of security of the command and decrypt the keys within the data field. This behavior is further detailed in sections 7.2.5 and 7.2.6.

As all Secure Channel Key Version Numbers in the range '01' to '6F' require 3 keys, even if a single key is being added for a new Key Version Number, the implementation may immediately allocate space for 3 keys. Note that a Secure Channel Key Version Number is only usable once all 3 keys have been populated.

On an implementation supporting Supplementary Security Domains, the first Application Provider’s or Controlling Authorities’ Key Version Number loaded to a Supplementary Security Domain using this command becomes the default Key Version Number for the Supplementary Security Domain.

For the Issuer Security Domain, when loading the issuer’s first Key Version Number, the Key Version Number 'FF' (currently the default) is replaced or disabled and this first Key Version Number becomes the default Key Version Number for the Issuer Security Domain.

For any Security Domain, if the keys within the default Key Version Number are updated at a later stage and the Key Version Number is also updated, the new Key Version Number becomes the default.

Class Byte

If the class byte excluding the Logical Channel number is not '80' or '84', a response of '6E00' is returned.

Reference Control Parameter 1 (Key Version Number)

The high order bit of this parameter indicates to the Security Domain that this command either contains the full content required for updating of the key data or that subsequent PUT KEY commands are expected. This guideline has no recommendations as to the behavior of the implementation with regards to this bit.

The low order 7 bits of this parameter define the current Key Version Number that will be modified based on information in the data field.

If this parameter contains a value greater than '6F', a response of '6A88' is returned.

A value of zero in these 7 bits indicates that no existing Key Version Number will be overwritten i.e. a new Key Version Number (defined in the command data) is being added.

If this parameter contains a value of zero, the new Key Version Number (in the command data) has a value other than zero.

If this parameter contains a value other than zero, this current Key Version Number is present within the Security Domain. If this condition is not met, a response of '6A88' is returned.

It is not possible to explicitly replace or disable the initialization key (i.e. 'FF' would be an illegal value for this parameter and a response of '6A86' is returned). In order to replace or disable the initial key, the value in this parameter is '00'.

Reference Control Parameter 2

The high order bit of this parameter indicates to the Security Domain that the data field contains either a single key or multiple keys. If multiple keys exist in the data field, they all relate to the same Key Version Number.

The low order 7 bits of this parameter contain the Key Identifier of the key or the Key Identifier of the first key in the data field. If this Key Identifier is zero or greater than 3, a response of '6A86' is returned.

If this PUT KEY command is being issued to the Issuer Security Domain that only has initial keys (Key Version Number 'FF') or is the first PUT KEY command issued to a Supplementary Security Domain, this parameter has a value of '81'. If not, a response of '6A86' is returned.

Data Field Sent in the Command Message

The data field is coded according to section 11.8.2.3 Format 1 of the GP CS 2.2.1 [1] as shown in Table 5 when populating a complete Secure Channel key set:

Name	Length
New Key Version Number ('01' to '6F')	1 byte
Key type ('80')	1 byte
DES key length ('10')	1 byte
DES key	16 bytes
Check value length ('03')	1 byte
Check value	3 bytes
Key type ('80')	1 byte
DES key length ('10')	1 byte
DES key	16 bytes
Check value length ('03')	1 byte
Check value	3 bytes
Key type ('80')	1 byte
DES key length ('10')	1 byte
DES key	16 bytes
Check value length ('03')	1 byte
Check value	3 bytes

Table 5: DES Key Data Field Structure

If the data field is not formatted correctly or is formatted according to section 11.8.2.3 Format 2 of the GP CS 2.2.1 [1]but Format 2 is not supported by the implementation, a response of '6A80' is returned.

The new Key Version Number identifies either the Key Version Number that will replace a current Key Version Number or that a key will be added to the Security Domain.

If the new Key Version Number has a value of '00' or a value greater than '6F', a response of '6A80' is returned.

If the key type of any of the keys in the data field is not '80', a specific response of '6A80' is returned.

If the key length of any of the keys in the data field is not '10', a response of '6A80' is returned.

If the combination of the Key Identifier in P2 and the number of keys in the data field would result in a Key Identifier greater than 3 for a Key Version Number, a response of '6A80' is returned.

Each encrypted key within the data field is decrypted and verified.

Decryption of the key is performed as described in section 5.3 and the resulting key is verified against its associated check value. This is achieved by encrypting binary zeroes (8 bytes of '00') with the plain text value of the key and comparing the left most 3 bytes with the associated check value. If this comparison fails, a response of '6982' is returned.

If keys are being updated (current Key Version Number identified in P1), the Secure Channel Sequence Counter associated with a Key Version Number is reset to zero and:

- Each key in the data field replaces the current key data. This starts with the Key Identifier defined in P2 and if there are subsequent keys in the data field, each subsequent Key Identifier is replaced.
- If a new Key Version Number is indicated in the data field, the Key Version Number for all Key Identifiers previously associated with the current Key Version Number, is changed regardless of how many keys were updated.
-

If a new Key Version Number is being added:

- If the Security Domain has not previously encountered this Key Version Number, a new Secure Channel Sequence Counter with an initial value of zero is associated to this Key Version Number.
- Populate the key data. This starts with the Key Identifier defined in P2 and if there are subsequent keys in the data field, each subsequent Key Identifier is populated.

The implementation associates the Key Usage with value '18' (C-ENC), '14' (C-MAC) and '48' (C-DEK) respectively to the first, second and third key.

The implementation associates the Key Access value '00' to these three keys meaning the key can be used by the Security Domain and any associated Application.

On an implementation supporting Supplementary Security Domains, a Supplementary Security Domain is responsible for transitioning its own Life Cycle State to PERSONALIZED on receipt of its first set of Secure Channel keys except if the Supplementary Security Domain has the DAP Verification privilege. In this case, the Supplementary Security Domain will only transition its Life Cycle State to PERSONALIZED on receipt of its DAP Verification key (see sections 6.10 and 6.13).

Data Field Returned in the Response Message

The data field of the response message contains the new Key Version Number followed by the same check values that were passed to the Application in the command data field.

6.10 PUT KEY (RSA Public Key)

This command is based upon the structure of the PUT KEY command as described in section 11.8 of the GP CS 2.2.1 [1].

A Supplementary Security Domain on an implementation supporting Supplementary Security Domains is able to differentiate between a PUT KEY command that contains DES keys and a PUT KEY command that contains a public key. This can be achieved through the Key Version Number.

This guideline defines an option that allows the RSA Public Key to be populated using the STORE DATA command.

6.10.1 Definition

This PUT KEY command is used by a Supplementary Security Domain that supports DAP Verification (i.e. has the DAP Verification privilege) to add a public key modulus and it related exponent.

The PUT KEY command may only be issued within a Secure Channel Session and the level of security for the command is dependent on the security level defined in the EXTERNAL AUTHENTICATE command.

A Supplementary Security Domain may not use the Secure Channel services of its associated Security Domain (i.e. the Issuer Security Domain) to load a public key i.e. the Secure Channel is set up using the Security Domain's own Secure Channel keys.

6.10.2 Recommendations

If the Supplementary Security Domain does not have DAP Verification privileges, a response of '6A81' is returned.

If the Supplementary Security Domain does not yet have its own Secure Channel Key Version Number, a response of '6985' is returned.

Class Byte

If the class byte excluding the Logical Channel number is not '80' or '84', a response of '6E00' is returned.

Reference Control Parameter 1

The high order bit of this parameter indicates that this command contains the full content of the key data.

As the public key components are only loaded once and not updated, the low order 7 bits of this parameter are always zero.

If this parameter contains a value other than '00', a response of '6A86' is returned.

Reference Control Parameter 2

The high order bit of this parameter indicates that the data field contains a single key.

The low order 7 bits of this parameter contain the Key Identifier of the key that is always 1.

If this parameter does not contain a value of '01', a response of '6A86' is returned.

Data Field Sent in the Command Message

The data field is coded according to section 11.8.2.3 Format 1 of the GP CS 2.2.1 [1] as shown in Table 6:

Name	Length
New Key Version Number ('73')	1 byte
Key type ('A1')	1 byte
Modulus length ('80')	1 byte
Public key modulus	128 bytes
Key type ('A0')	1 byte
Exponent length ('01' or '03')	1 byte
Exponent ('03' or '010001')	1 or 3 bytes
Length of key check value ('00')	1

Table 6: Public Key Data Field Structure

If the data field is not formatted correctly or is formatted according to section 11.8.2.3 Format 2 of the GP CS 2.2.1 [1] but Format 2 is not supported by the implementation, a response of '6A80' is returned.

The new Key Version Number ('73') identifies the Key Version Number that will be added to the Supplementary Security Domain. If the Key Version Number does not contain a value of '73', a response of '6A80' is returned.

If the first key type is not 'A1' or the second key type is not 'A0', a specific response of '6A80' is returned.

If the modulus length is not 128, a response of '6A80' is returned.

If the exponent length is not 1 or 3, a response of '6A80' is returned.

Populate the key data i.e. the public key modulus of 128 bytes and the exponent of 1 or 3 bytes.

The implementation associates the Key Usage with value '84' (DAP) to this key.

The implementation associates the Key Access value '01' to this key meaning the key can only be used by the Security Domain.

A Supplementary Security Domain is responsible for transitioning its own Life Cycle State to PERSONALIZED on receipt of its DAP Verification key.

Data Field Returned in the Response Message

The data field of the response message contains the new Key Version Number (i.e. '73').

6.11 SELECT

See section 11.9 of the GP CS 2.2.1 [1] for the structure of the SELECT command. Also refer to sections 6.4.2.1.2 and 6.4.3.1.2 of [1] for the description of the OPEN relating to SELECT command processing.

6.11.1 Definition

This SELECT command is utilized for selecting an application on an open logical channel. As this guideline specifies that the card assign logical channel numbers, using the SELECT command to open a logical channel is outside the scope of this guideline.

6.11.2 OPEN Recommendations

The runtime environment (combination of OPEN and JCRE) is responsible for determining which Application will become the currently selected Application on the indicated logical channel on receipt of a SELECT [by name] command.

If the card Life Cycle State is TERMINATED, a response of '6D00' is returned. (The selected Application is first deselected and if the command is being issued on a Supplementary Logical Channel, that logical channel is closed.)

If an attempt is made to select an Application on a Supplementary Logical Channel when the card Life Cycle is in the state CARD_LOCKED a response of '6A81' is returned (The selected Application is first deselected and the Supplementary Logical Channel closed).

If an attempt is made to select an Application other than the Issuer Security Domain on the Basic Logical Channel when the card Life Cycle is in the state CARD_LOCKED, a response of '6A81' is returned (The Application on the Basic Logical Channel is first deselected and the Issuer Security Domain becomes the selected Application on the Basic Logical Channel).

Class Byte

If the class byte excluding the Logical Channel number is not '00', a response of '6E00' is returned.

Reference Control Parameter 1

If this parameter contains a value other than '04' (SELECT [by name] command), the SELECT command is passed on to the Application currently selected on this logical channel for further processing.

Reference Control Parameter 2

If this parameter contains a value of '02' (SELECT [next occurrence] command) but a previous SELECT [first or only occurrence] command has not been received within this Card Session on the Basic Logical Channel or since the opening of the Supplementary Logical Channel, a response of '6A86' is returned.

Data Field Sent in the Command Message

If the data field is empty the Issuer Security Domain becomes the currently selected application on the specified logical channel and the SELECT command is dispatched to the Issuer Security Domain.

If the value in P2 indicates next occurrence ('02') but the data field does not match the data field of the previous SELECT command, the behavior is at the discretion of the implementer.

If a matching Application is found but the Application is in the Life Cycle State LOCKED, then if no subsequent matching Application exists in the GlobalPlatform Registry, a response of '6A82' is returned.

If a matching Application that does not implement the `multiSelectable` interface is found and this Application or any other Application instantiated from the same Executable Load File is selected on another logical channel, then if no subsequent matching Application exists in the GlobalPlatform Registry, a response of '6985' is returned.

If no matching Application is found and the value in P2 indicates next occurrence ('02'), a response of '6A82' is returned.

6.11.3 Security Domain Recommendations

The following description of the SELECT command relates directly to how the Issuer Security Domain should process this command and how a Supplementary Security Domain should process this command.

Reference Control Parameter 1

All Security Domains are only required to process the SELECT [by name] command. If a value other than '04' is received in this parameter, a response of '6A86' is returned.

Reference Control Parameter 2

The content of this parameter is ignored.

Data Field Sent in the Command Message

The Security Domain ensures that the SELECT command received is due to this Application being selected. If the SELECT command has been received due to the AID being selected not being located by the OPEN in the GlobalPlatform Registry, a response of '6A82' is returned.

Data Field Returned in the Response Message

The following table defines the FCI returned by a successful selection of a Security Domain:

Tag	Description	Presence
'6F'	File Control Information (FCI template)	Mandatory
'84'	Security Domain AID	Mandatory
'A5'	Proprietary data	Mandatory
'73'	Security Domain Management Data	Optional
'9F65'	Maximum block length	Mandatory
'BF0C'	FCI discretionary data	Optional

Table 7: Security Domain SELECT Response Message

The maximum block length defines the size of the APDU buffer internally to the card. This length relates specifically to the length of the data field of an APDU (i.e. it does not include the header). For an implementation supporting symmetric cryptographic only, the Issuer Security Domain responds with a minimum length of 128, immaterial of the protocol. For an implementation supporting symmetric and asymmetric cryptography, the Issuer Security Domain responds with a length of at least 248.

The optional FCI discretionary may contain issuer proprietary data in the case of an Issuer Security Domain.

The optional FCI discretionary may contain card vendor proprietary data in the case of a Supplementary Security Domain.

The optional Security Domain Management Data for the Issuer Security Domain is coded according to *section 3.2.1.1* of this document.

The optional Security Domain Management Data for a Supplementary Security Domain is coded using the same values defined in s*ection 3.2.1.1* of this document except for the Card configuration details that the value to be used is '2A 8648 86FC6B 02 01 03'.

In the case of the Issuer Security Domain, if the card Life Cycle is in the state CARD_LOCKED, a warning response of '6283' is returned. (Contrary to EMV Book 1 [9], it is only necessary to return this warning response following the transmission of the response message.)

6.12 SET STATUS

See section 11.10 of the GP CS 2.2.1 [1] for the structure of the SET STATUS command. This guideline defines an option that allows the Life Cycle State of the card to be transitioned using the STORE DATA command.

6.12.1 Definition

The SET STATUS command is used by the Issuer Security Domain only to change the Life Cycle State of the card and to lock or unlock an Application.

The SET STATUS command may only be issued within a Secure Channel Session and the level of security for the command is dependent on the security level defined in the EXTERNAL AUTHENTICATE command.

6.12.2 Recommendations

If the card Life Cycle State is TERMINATED, a response of '6D00' is returned.

Class Byte

If the class byte excluding the Logical Channel number is not '80' or '84', a response of '6E00' is returned.

Reference Control Parameter 1

If a value other than '80' or '40' or '60' is received in this parameter, a response of '6A86' is returned.

If the implementation does not support the value of '60' i.e. functions relating to Security Domain and its associated Applications, a response of '6A86' is returned.

Reference Control Parameter 2

The content of this parameter can only be verified in conjunction with the AID in the command message.

Data Field Sent in the command message

If P1 indicates the Issuer Security Domain ('80') the following applies:

- The content of the command message is ignored.
- If the value of P2 is not coded according to Table 11-6 of [1], a response of '6A86' is returned.
- If the value of P2 does not abide by the transitioning rules diagrammed in Figure 5-1 of [1], a response of '6985' is returned.
- If the Life Cycle State of the card is currently set to the value of P2, a response of '6985' is returned.
- The Life Cycle State of the card is set to the value of P2.

If P1 indicates an Application ('40') the following applies:

- If the AID cannot be located in the GlobalPlatform Registry, a response of '6A88' is returned. This search through the GlobalPlatform Registry only locates Applications (no Executable Load Files or Executable Modules) that fully match the AID.
- If bit 8 of P2 is set and the Life Cycle State of the Application is LOCKED, a response of '6985' is returned.
- If bit 8 of P2 is not set and the Life Cycle State of the Application is not LOCKED, a response of '6985' is returned.
- If bit 8 of P2 is set, the Life Cycle State of the Application is set to LOCKED i.e. bit 8 of the Life Cycle State byte is set.

- If bit 8 of P2 is not set, the Life Cycle State of the Application is set to its Life Cycle State prior to becoming LOCKED i.e. bit 8 of the Life Cycle State byte is cleared.

If P1 indicates an Application ('60') the following applies:

- If the AID cannot be located in the GlobalPlatform Registry, a response of '6A88' is returned. This search through the GlobalPlatform Registry only locates Security Domains (no Executable Load Files or Executable Modules or Applications) that fully match the AID.
- If bit 8 of P2 is set and the Life Cycle State of the Security Domain is LOCKED, a response of '6985' is returned.
- If bit 8 of P2 is not set and the Life Cycle State of the Security Domain is not LOCKED, a response of '6985' is returned.
- If bit 8 of P2 is set, the Life Cycle State of the Security Domain and all its associated Applications is set to LOCKED i.e. bit 8 of the Life Cycle State byte is set.
- If bit 8 of P2 is not set, the Life Cycle State of the Security Domain is set to its Life Cycle State prior to becoming LOCKED, i.e. bit 8 of the Life Cycle State byte is cleared. The Life Cycle State of the Security Domain's associated Applications are set to their Life Cycle State prior to the instant when the Security Domain's Life Cycle was set to LOCKED.

Data Field Returned in the Response Message

The data field of the response message is not present.

6.13 STORE DATA

See section 11.11 of the GP CS 2.2.1 [1] for the structure of the STORE DATA command.

6.13.1 Definition

The STORE DATA command is specifically used to personalize non-key related data for Security Domains. The key data for a Security Domain is personalized either using the PUT KEY command(s) or this STORE DATA command. The Life Cycle of the card is transitioned either using the SET STATUS command or this STORE DATA command.

The STORE DATA command may only be issued within a Secure Channel Session and the level of security for the command is dependent on the security level defined in the EXTERNAL AUTHENTICATE command.

Initially to populate data or DES keys within a Supplementary Security Domain, the Security Domain may use the Secure Channel services of its associated Security Domain. To populate the DAP Verification key within a Supplementary Security Domain, the Security Domain may only set up the Secure Channel using the Security Domain's own Secure Channel keys.

6.13.2 Recommendations

For the Issuer Security Domain, if the card Life Cycle State is TERMINATED, a response of '6D00' is returned.

For the Issuer Security Domain, if the card Life Cycle State is CARD_LOCKED, a response of '6985' is returned.

For the Issuer Security Domain, data being populated is indiscriminately written to, or updated in, the Issuer Security Domain's Data Store except in the cases listed below.

On an implementation supporting Supplementary Security Domains, data being populated to a Supplementary Security Domain is indiscriminately written to, or updated in, the Security Domain's Data Store except in the cases listed below.

As the STORE DATA command data field may contain multiple DGI blocks, and within each DGI, multiple pieces of TLV coded data, successful completion indicates that all data has been populated correctly while unsuccessful completion indicates that no data has been populated.

Class Byte

If the class byte excluding the Logical Channel number is not '80' or '84', a response of '6E00' is returned.

Reference Control Parameter 1

The content of this parameter is ignored.

Reference Control Parameter 2 (block number)

The block number is coded sequentially from '00' to 'FF'.

If a block number is received out of sequence, a response of '6A86' is returned.

Data field sent in the command message

The Issuer Security Domain supports the following Data Grouping Indexes contained in the command message:

DGI	Length	Description of contents
'0070'	variable	One or more TLV coded objects
'8F01'	48 bytes	Secure Channel DES keys. See Table 10
'7F01'	12 bytes	DES key related data. See Table 10
'9F70'	1 byte	Card Life Cycle State
'00CF'	10 bytes	Key Derivation Data

Table 8: Issuer Security Domain supported Data Grouping Indexes

The Issuer Security Domain expects to receive DGI '8F01' and DGI '7F01' within a single STORE DATA command and expects any other STORE DATA command to contain a single DGI. If a DGI other than those listed in Table 8 is received, a response of '6A88' is returned.

The Supplementary Security Domain supports the following Data Grouping Indexes contained in the command message:

DGI	Length	Description of contents
'0070'	variable	One or more TLV coded objects
'8F01'	48 bytes	Secure Channel DES keys. See Table 10
'7F01'	12 bytes	DES key related data. See Table 10
'9102'	129 or 131 bytes	DAP Verification key data. See Table 11
'0102'	6 bytes	DAP Verification key related data. See Table 11
'00CF'	10 bytes	Key Derivation Data

Table 9: Supplementary Security Domain supported Data Grouping Indexes

The Supplementary Security Domain expects to receive DGI '8F01' and DGI '7F01' within a single STORE DATA command and DGI '9102' and '0102' within a single STORE DATA command and expects any other STORE DATA command to contain a single DGI. If a DGI other than those listed in Table 9 is received, a response of '6A88' is returned.

For the Issuer Security Domain, if the value of the DGI is:

- Card Life Cycle State (DGI '9F70') then
  - If the length of the data is not 1, a response of '6A80' is returned.
  - If the value within this DGI is not coded according to Table 11-6 of the GP CS 2.2.1 [1], a response of '6A86' is returned.
  - If the value within this DGI does not abide by the transitioning rules diagrammed in Figure 5-1 of [1], a response of '6985' is returned.
  - If the Life Cycle State of the card is currently set to the value within this DGI, a response of '6985' is returned.
  - The Life Cycle State of the card is set to the value within this DGI.
- Key Derivation Data (DGI '00CF') then

  - If the length of the data is not 10, a response of '6A80' is returned.
  - The Key Derivation Data that is currently present in the Data Store is overwritten.

For the Issuer Security Domain, if any TLV object within a DGI of '0070' contains:

- The Issuer Security Domain AID (tag '4F') and the length of the AID is not between 5 and 16 bytes, a response of '6A80' is returned.
  The value within this TLV object is not stored by the Issuer Security Domain at all but becomes the AID of the Issuer Security Domain (the interface between the Issuer Security Domain and the GlobalPlatform Registry that achieves this result is outside the scope of this guideline). While it is not logical to change the AID of the Issuer Security Domain more than once, there is no recommendation for an implementation to prohibit an attempt to do so.
- Secure Channel Sequence Counter (tag 'C1'), a response of '6A80' is returned.
- Key information (tags 'E0' or 'C0'), a response of '6A80' is returned.
- Key Derivation Data (tag 'CF'), a response of '6A80' is returned.

On an implementation supporting Supplementary Security Domains, if the value of the DGI being directed to a Supplementary Security Domain is '00CF' then:

- If the length of the data is not 10, a response of '6A80' is returned.
- The Key Derivation Data that is currently present in the Data Store is overwritten.

On an implementation supporting Supplementary Security Domains, if any TLV object within a DGI of '0070' being directed to a Supplementary Security Domain contains:

- The Secure Channel Sequence Counter (tag 'C1'), a response of '6A80' is returned.
- Key information (tags 'E0' or 'C0'), a response of '6A80' is returned.
- Key Derivation Data (tag 'CF'), a response of '6A80' is returned.

For all other TLV coded pieces of data received within a DGI of '0070' by a Security Domain (Issuer Security Domain or Supplementary Security Domain) a search is conducted through the Data Store of that Security Domain. If the tag already exists within the Data Store, but the new data length is greater than the maximum data length, a response of '6A80' is returned. If the remaining TLV coded pieces of data (those not already present in the Data Store) will not fit within the Data Store, a response of '6A84' is returned. Data that is currently present within the Data Store is overwritten in the Data Store i.e. the actual length and the value. Data that is not currently present within the Data Store is written to the end of the Data Store i.e. the tag, the maximum length and actual length and the value. The maximum length and actual length is identical in value.

The data field of the STORE DATA command containing Secure Channel key information is expected to be formatted as such:

Description	Length	Value
Key data DGI	2 bytes	'8F01'
Key data length	1 byte	'30'
Encrypted Key Identifier 1 (encryption)	16 bytes	Variable
Encrypted Key Identifier 2 (MAC)	16 bytes	Variable
Encrypted Key Identifier 3 (DEK)	16 bytes	Variable
Related data DGI	2 bytes	'7F01'
Related data length	1 byte	'0C'
Current Key Version Number	1 byte	'00' or '01' to '6F'
New Key Version Number	1 byte	'01' to '6F'
Key type	1 byte	'80'
Check value for Key Identifier 1	3 bytes	Variable
Check value for Key Identifier 2	3 bytes	Variable
Check value for Key Identifier 3	3 bytes	Variable

Table 10: Secure Channel Key Information

DGI's '8F01' and '7F01' shall be present within a single STORE DATA command.

A value of zero in the current Key Version Number indicates that no existing Key Version Number will be overwritten i.e. a new Key Version Number is being added.

If the current Key Version Number has a value other than zero, this current Key Version Number is present within the Security Domain. If this condition is not met, a response of '6A88' is returned.

For the Issuer Security Domain, it is not possible to explicitly replace or disable the initialization key i.e. 'FF', or any value greater than '6F' for the Issuer Security Domain and Supplementary Security Domains would be an illegal value for the current Key Version Number and a response of '6A80' is returned. In order to replace or disable the initial key, the value of the current Key Version Number is '00'.

The new Key Version Number identifies either the Key Version Number that will replace a current Key Version Number or the Key Version Number of a key set that will be added to the Security Domain. A value greater than '6F' would be an illegal value for the new Key Version Number and a response of '6A80' is returned.

Decryption of each 16-byte key is performed as described above in section 5.3 and the resulting key is verified against the associated check value within DGI '7F01'. This verification is achieved by encrypting binary zeroes (8 bytes of '00') with the plain text value of the key and comparing the left most 3 bytes with the associated check value. If this comparison fails, a response of '6982' is returned.

If keys are being updated (current Key Version Number within DGI '7F01' is not '00'), the Secure Channel Sequence Counter associated with a Key Version Number is reset to zero and each key in the data field replaces the current key data i.e. the first key becomes the key value for Key Identifier 1, the second key becomes the key value for Key Identifier 2 and the third key becomes the key value for Key Identifier 3.

If a new Key Version Number is being added (current Key Version Number within DGI '7F01' is '00'), a new Secure Channel Sequence Counter with an initial value of zero is associated to this Key Version Number and the first key becomes the key value for Key Identifier 1, the second key becomes the key value for Key Identifier 2 and the third key becomes the key value for Key Identifier 3.

The implementation associates the Key Usage with value '18' (C-ENC), '14' (C-MAC) and '48' (C-DEK) respectively to the first, second and third key.

The implementation associates the Key Access value '00' to these three keys meaning the key can be used by the Security Domain and any associated Application.

A Supplementary Security Domain is responsible for transitioning its own Life Cycle State to PERSONALIZED on receipt of its first set of Secure Channel keys except if the Supplementary Security Domain has the DAP Verification privilege. In this case, the Supplementary Security Domain will only transition its Life Cycle State to PERSONALIZED on receipt of its DAP Verification key (see *6.10* of this section or the subsequent description below).

The data field of the STORE DATA command containing the DAP Verification key information is expected to be formatted as such by a Supplementary Security Domain:

Description	Length	Value
Key Data DGI	2 bytes	'9102'
Key Data Length	1 byte	'81' or '83'
Public key modulus	128	Variable
Public key exponent	1 or 3 byte(s)	'03' or '010001'
Related Data DGI	2 bytes	'0102'
Related Data Length	1 byte	'06'
Current Key Version Number	1 byte	'00'
New Key Version Number	1 byte	'73
Key type (public key modulus)	1 byte	'A1'
Length of modulus	1 byte	'80'
Key type (exponent)	1 byte	'A0'
Length of exponent	1 byte	'01' or '03'

Table 11: DAP Verification key information

DGI's '9102' and '0102' shall be present within a single STORE DATA command.

If the Supplementary Security Domain does not have DAP Verification privileges, a response of '6A81' is returned.

If the Supplementary Security Domain does not yet have its own Secure Channel Key Version Number, a response of '6985' is returned.

As the public key components are only loaded once and not updated, if the current Key Version Number does not contain a value of '00', a response of '6A80' is returned.

The new Key Version Number ('73') identifies the Key Version Number that will be added to the Supplementary Security Domain. If the Key Version Number does not contain a value of '73', a response of '6A80' is returned.

If the first key type is not 'A1' or the second key type is not 'A0', a specific response of '6A80' is returned.

If the modulus length is not 128, a response of '6A80' is returned.

If the exponent length is not 1 or 3, a response of '6A80' is returned.

Populate the key data i.e. the public key modulus of 128 bytes and the exponent of 1 or 3 bytes.

The implementation associates the Key Usage with value '84' (DAP) to this key.

The implementation associates the Key Access value '01' to this key meaning the key can only be used by the Security Domain.

A Supplementary Security Domain is responsible for transitioning its own Life Cycle State to PERSONALIZED on receipt of its DAP Verification key.

Data Field Returned in the Response Message

The data field of the response message is not present.

6.14 Response Codes

This section describes the response codes for the OPEN, Issuer Security Domain and Supplementary Security Domains.

SW1	SW2	Meaning
'6E'	'00'	Class not supported
'6D'	'00'	Instruction not supported or invalid
'64'	'00'	No specific diagnosis error
'62'	'00'	No specific diagnosis warning
'62'	'83'	Card locked
'63'	'00'	Authentication failed warning
'63'	'10'	More data available
'67'	'00'	Wrong length
'6F'	'00'	No specific diagnosis
'68'	'81'	Logical channel error
'69'	'82'	Security status not satisfied
'69'	'85'	Conditions of use not satisfied
'6A'	'80'	Incorrect values in data field
'6A'	'81'	Function not supported
'6A'	'82'	Application not found
'6A'	'84'	Insufficient memory space
'6A'	'86'	Incorrect P1/P2
'6A'	'88'	Referenced data not found
'90'	'00'	Successful execution of command

Table 12: Response Codes

- '6E00' – Class not supported. This response indicates that the class byte received is not recognized or supported by the application.
- '6D00' – Instruction not supported or invalid. This response indicates that the instruction byte received is not recognized or supported by the application or contains an invalid value e.g. an odd value.
- '6400' – No specific diagnosis error. This response indicates that processing could not occur. Typically this would be due to a low level error e.g. memory could not be written.
- '6200' – No specific diagnosis warning. This response indicates that processing did not complete fully i.e. the request was fulfilled but all the actions required could not take place.
- '6283' – Card locked warning. This response indicates that while the SELECT command completed successfully, the subsequent functionality of the card is limited i.e. it is not possible to update content or select any other Application.
- '6300' – Authentication failed warning. This response indicates that the authentication of the host cryptogram failed.

- '6310' – More data available. This response warns that all the available data have not yet been returned. The command that caused this response should be re-issued with the [get next] option set to retrieve the remaining data.
- '6700' – Wrong length error. This response indicates that the length byte of the header (Lc or Le) was not a length expected by the application.
- '6F00' – No specific diagnosis error. This response indicates that processing aborted for some reason. Typically this would be due to a low level error e.g. runtime exception.
- '6881' – Logical channel error. This response indicates that a Supplementary Logical Channel is either not supported by the card, currently not opened or cannot currently be opened.
- '6982' – Security status not satisfied error. This response indicates that the required authentication failed i.e. this error could be due to the padding for encryption being inconsistent with that expected (this indicates that decryption of the command data field failed), that the verification of the MAC failed or that some other security condition was not met.
- '6985' – Conditions of use not satisfied error. This response indicates that the correct sequence of events has not been followed or that some control has not been performed prior to issuing the current command.
- '6A80' – Incorrect values in data field. This response indicates that the data in the command data field is not that expected by the application.
- '6A81' – Function not supported. This response indicates that the function or command is not, or is no longer, supported by the card.
- '6A82' – Application not found. This response indicates that the application is not present on the card or that no more applications with the partial AID are present on the card.
- '6A84' – Insufficient memory space. This response indicates that there is not enough memory in order to complete the request.
- '6A86' – Incorrect P1/P2 – This response indicates that the values in P1/P2 are not valid for the command or the required process.
- '6A88' – Referenced data not found. This response indicates that data referenced by the command is not present.
- '9000' – Successful execution of command. This response indicates that the required process completed correctly.

7 Application Programming Interface

This section describes the GlobalPlatform 2.2.1 API.

The recommended functionality of each method is generally defined. The exact implementation and order in which checks are performed is vendor specific.

Any operations performed within this API is independent of, and not interfere with, a transaction that may currently be active (e.g. if the `CVM.verify()` method is invoked from within a transaction and this transaction is aborted, the retry counter is not revert to its original value).

The export and related files for this API can be obtained from the GlobalPlatform website (www.globalplatform.org).

This data is available at:

http://www.globalplatform.org

The AID of the export file of this package is 'A0 00 00 01 51 00'.

7.1 class GPSystem

7.1.1 byte **getCardContentState()**

The entry of the current applet context is located in the GlobalPlatform Registry.

The Life Cycle State of this entry is returned.

7.1.2 byte **getCardState()**

The Life Cycle State of the card is returned.

7.1.3 CVM **getCVM(**byte bCVMIdentifier**)**

The handle to the `CVM` interface object reference is returned.

If `bCVMIdentifier` is not '11' (`CVM_GLOBAL_PIN`) an exception of '6A81' is thrown.

7.1.4 GPRegistryEntry **getRegistryEntry(**AID reqAID**)**

The handle to the GPRegistryEntry interface object reference is returned.

7.1.5 SecureChannel **getSecureChannel()**

The entry of the current applet context is located within the GlobalPlatform Registry.

The handle to the SecureChannel interface object reference of the current applet context's associated Security Domain is returned.

7.1.6 GlobalService **getService(**AID serverAID, short sServiceName**)**

The support for Global Service is beyond the scope of this guideline. If not supported, `null` is returned.

7.1.7 boolean **lockCard()**

The entry of the current applet context is located within the GlobalPlatform Registry.

If the current card Life Cycle State is not SECURED, return `false`.

If the Application does not have card lock privileges (bit 5 of the privileges byte not set), return `false`.

The card Life Cycle State transitions to CARD_LOCKED and return `true`.

7.1.8 boolean **terminateCard()**

The entry of the current applet context is located within the GlobalPlatform Registry.

If the Application does not have card terminate privileges (bit 4 of the Application Privileges not set), return `false`.

The card Life Cycle State transitions to TERMINATED and return `true`.

7.1.9 boolean **setATRHistBytes(**byte[] baBuffer, short sOffset, byte bLength**)**

If the `bLength` parameter is greater than 15, return `false`.

The entry of the current applet context is located within the GlobalPlatform Registry.

If the Application does not have Card Reset privileges (bit 3 of the Application Privileges not set), return `false`.

The historical bytes contained in the `baBuffer` parameter at the `sOffset` parameter is the historical bytes to be returned in the ATR string to a cold reset.

The `bLength` parameter is reflected in the format character (low order nibble of T0) of the ATR string.

Return `true`.

7.1.10 boolean **setCardContentState(**byte bState**)**

The entry of the current applet context is located within the GlobalPlatform Registry.

If the three low order bits of the bState value are not set or if the high order bit of the bState value is set, return false.

If the Life Cycle State of the Application is LOCKED, return `false`.

A logical OR operation of `bState` and the value '07' is performed and this result replaces the Application Life Cycle State in the GlobalPlatform Registry.

Return `true`.

7.2 **Interface** SecureChannel

The following descriptions make use of a flag (SCFlag on a byte) to keep track of the steps within a Secure Channel Session. This flag mirrors the return value of the `getSecurityLevel()` method.

Initially, following power on of the card or reset of the card, the SCFlag would be set to '00' (`NO_SECURITY_LEVEL`).

For any invocation of a `SecureChannel` method, if the Life Cycle State of the Security Domain has transitioned to LOCKED, an exception of '6F00' is thrown.

A Security Domain keeps track of a Secure Channel Session and ensures that any information relating to that Secure Channel is erased whenever the Secure Channel is terminated.

One event that causes a Secure Channel Session to terminate is the selection of an Application on the same logical channel on which the Secure Channel was initiated. An implementation ensures that all calls to `SecureChannel` methods are invoked by the Application that initiated the Secure Channel Session. If the Application invoking a `SecureChannel` method on the logical channel on which this Secure Channel was initiated did not actually initiate the setting up of a Secure Channel, any information relating to a previous Secure Channel Session is cleared and the SCFlag is set to '00' (`NO_SECURITY_LEVEL`). If an Application invokes a `SecureChannel` method on a logical channel that is different from the logical channel that the current Secure Channel was initiated, the Secure Channel Session is not terminated but the invocation of the method fails.

7.2.1 `short` **`processSecurity(`**`apdu`**`)`**

If the `INS` byte of the command header is not '50' or '82', an exception of '6D00' is thrown.

If the `INS` byte of the command header is '50' but the `CLA` byte of the command header (excluding logical channel information) is not '80' an exception of '6E00' is thrown.

If the `INS` byte of the command header is '82' but the `CLA` byte of the command header is not '84' (excluding logical channel information) an exception of '6E00' is thrown.

7.2.2 `Processing for INITIALIZE UPDATE command`

If the `LC` byte of the command header is not '08' an exception of '6700' is thrown.

If a Secure Channel Session is currently active on a logical channel other than the logical channel on which this command was issued, an exception of '6985' is thrown.

Any information relating to a current Secure Channel Session is discarded and the SCFlag would be set to '00' (`NO_SECURITY_LEVEL`).

If the `P1` byte of the command header is '00' this indicates that the default Key Version Number is used. (For more information on the default key, refer to sections 6.9 and 6.13.)

If the value in the `P1` byte of the command header indicates a Key Version Number that is not present within the Security Domain, does not correspond to a Secure Channel keys set, or indicates a Key Version Number that is incomplete (i.e. the Key Version Number does not contain Key Identifiers 1, 2 and 3), an exception of '6A88' is thrown.

The Secure Channel (C-)MAC, encryption and DEK session keys are generated as described in Appendix E.4.1 of the GP CS 2.2.1 [1].

If the received host challenge is identical to the concatenation of the Secure Channel Sequence Counter of the identified Key Version Number and the card challenge, an exception of '6982' is thrown.

The APDU buffer is populated as such:

- The value of tag 'CF' (Key Derivation Data) from the Security Domain’s Data Store is placed at offset `CDATA`.
- The Key Version Number used to generate the session keys is placed at offset `CDATA+10`.
- The Secure Channel Protocol identifier ('02') is placed at offset `CDATA+11`.
- The Secure Channel Sequence Counter of the identified Key Version Number is placed at offset `CDATA+12`.
- The data (card challenge) is placed at offset `CDATA+14`.
- The data (card cryptogram) generated using the Secure Channel encryption session key as described in Appendix E.4.2.1 of the GP CS 2.2.1 [1] is placed at offset `CDATA+20`.

Return a short of 28 i.e. the length of data to be output by the application.

SCFlag is still '00' (`NO_SECURITY_LEVEL`).

7.2.3 Processing for EXTERNAL AUTHENTICATE command

If the `LC` byte of the command header is not '10' an exception of '6700' is thrown.

If the secure messaging bit of the class byte (bit 3 of the `CLA` byte of the command header) is not set, an exception of '6E00' is thrown.

If a previous INITIALIZE UPDATE command has not been received, an exception of '6985' is thrown.

If the `P1` byte of the command header is not '00', '01' or '03', any information relating to a current Secure Channel Session is discarded and an exception of '6A86' is thrown. If the card Life Cycle State is SECURED or LOCKED and the `P1` byte of the command header is '00', a response of '6985' is thrown.

If MAC verification, as described in section 5.2 fails, any information relating to a current Secure Channel Session is discarded and an exception of '6982' is thrown. If the MAC is valid, it is maintained for the life of the Secure Channel Session. (It will be used to precede the command header for the next MAC verification in the `unwrap()` method if SCFlag is set to '81' or '83'.)

A cryptogram is generated for comparison with the received host cryptogram using the Security Channel encryption session key as described in Appendix E.4.2.2 of the GP CS 2.2.1 [1]. If the comparison fails, any information relating to a current Secure Channel Session is discarded and an exception of '6300' is thrown.

If P1 byte of the command header was set to '00', set SCFlag to '80' (`AUTHENTICATED`).

If P1 byte of the command header was set to '01', set SCFlag to '81' (`AUTHENTICATED` and `C_MAC`).

If P1 byte of the command header was set to '03', set SCFlag to '83' (`AUTHENTICATED`, `C_MAC` and `C_DECRYPTION`).

Increment the Secure Channel Sequence Counter of the identified Key Version Number by 1.

A Secure Channel Session now exists.

Return a short of 0.

7.2.4 short **wrap (**byte[] baBuffer, short sOffset, short sLength**)**

If option R-MAC is not supported by the implementation, this method will do no processing and returns a short of `sLength`.

7.2.5 short **unwrap (**byte[] baBuffer, short sOffset, short sLength**)**

If SCFlag is '00' (`NO_SECURITY_LEVEL`),

- If the requested security level is reset then return a short of `sLength`.
- Otherwise an exception of '6985' is thrown.

If SCFlag is '80' (`AUTHENTICATED`), return a short of `sLength`.

If SCFlag is '83' (`AUTHENTICATED`, `C_MAC` and `C_DECRYPTION`):

- Perform data field decryption as described in section 5.1.
- If the format of the data field is deemed to be incorrect by the implementation, any information relating to a current Secure Channel Session is discarded, the SCFlag is set to '00' (`NO_SECURITY_LEVEL`) and an exception of '6982' is thrown.
- The decrypted data is positioned to start at offset `CDATA`.

If SCFlag is '81' (`AUTHENTICATED` and `C_MAC`) or '83' (`AUTHENTICATED`, `C_MAC` and `C_DECRYPTION`):

- Perform MAC verification as described in section 5.2.
- If MAC verification fails, any information relating to a current Secure Channel Session is discarded, the SCFlag is set to '00' (`NO_SECURITY_LEVEL`) and an exception of '6982' is thrown.
- If the MAC is valid, it replaces the previous MAC and will be maintained for the life of the Secure Channel Session (It will be used to precede the command header for the next MAC verification in a subsequent `unwrap()` method).

Bit 3 of the `CLA` byte of the command header is cleared.

The `LC` byte of the command header is set to the new length of the data field i.e. excluding the command header, any padding due to data field encryption and excluding the length of the MAC.

Return a short indicating the unwrapped length of the command data i.e. including the header but excluding any padding due to data field encryption and excluding the length of the MAC.

7.2.6 short **decryptData (**byte[] baBuffer, short sOffset, short sLength**)**

If SCFlag is '00' (`NO_SECURITY_LEVEL`), an exception of '6985' is thrown.

If `sLength` is not a multiple of 8, an exception of '6700' is thrown.

Decrypt `sLength` bytes of data in `baBuffer` at offset `sOffset`, using the data encryption session key (DEK session key), as described in Appendix E.4.7 of the GP CS 2.2.1 [1] and specifically the mechanism defined in Appendix B.1.1.2 of [1].

The decrypted data replaces the data in `baBuffer` at offset `sOffset` for a length of `sLength`.

Return a short with a value of `sLength`.

7.2.7 short **encryptD**ata **(**byte[] baBuffer, short sOffset, short sLength**)**

If this method is not supported and is invoked, an exception of '6982' is thrown.

7.2.8 void **resetSecurity()**

Any information relating to a current Secure Channel Session is discarded.

The SCFlag is set to '00' (`NO_SECURITY_LEVEL`)

7.2.9 byte **getSecurityLevel()**

Return SCFlag.

7.3 **Interface** CVM

The following descriptions make use of a flag (CVMFlag) that describes the various states of the Global PIN.

The following descriptions also make use of the Retry Counter and Retry Limit terms used in the GP CS 2.2.1 [1]. In these descriptions it is assumed that the Retry Counter is initially set to the Retry Limit and it is decremented when the Global PIN is incorrectly presented.

7.3.1 `boolean` **`isActive()`**

Return `true`.

7.3.2 `boolean` **`isSubmitted()`**

If CVMFlag is set to INVALID_SUBMISSION or VALIDATED, return `true`.

If CVMFlag is set to ACTIVE or BLOCKED, return `false`.

7.3.3 `boolean` **`isVerified()`**

If CVMFlag is set to VALIDATED, return `true`.

If CVMFlag is set to ACTIVE, INVALID_SUBMISSION or BLOCKED, return `false`.

7.3.4 `boolean` **`isBlocked()`**

If CVMFlag is set to BLOCKED, return `true`.

If CVMFlag is set to ACTIVE, INVALID_SUBMISSION or VALIDATED, return `false`.

7.3.5 `byte` **`getTriesRemaining()`**

Return the Retry Counter.

7.3.6 `boolean` **`update(`**`byte[] baBuffer, short sOffset, byte bLength, byte bFormat`**`)`**

The entry of the current applet context is located within the GlobalPlatform Registry.

If the Application does not have CVM management privileges (bit 2 of the Application Privileges not set), return `false`.

Replace the value of the Global PIN with `bLength` bytes of data at `sOffset` in `baBuffer`.

Reset the Retry Counter to Retry Limit.

CVMFlag is set to ACTIVE.

Return `true`.

7.3.7 `boolean` **`resetState()`**

If CVMFlag is BLOCKED, return `false`.

If CVMFlag is ACTIVE, return `true`.

If CVMFlag is VALIDATED or INVALID_SUBMISSION, set CVMFlag to ACTIVE.

Return `true`.

7.3.8 `boolean` **`blockState()`**

The entry of the current applet context is located within the GlobalPlatform Registry.

If the Application does not have CVM management privileges (bit 2 of the Application Privileges not set), return `false`.

Set CVMFlag to BLOCKED.

Set the Retry Counter to 0.

Return `true`.

7.3.9 `boolean` **`resetAndUnblockState()`**

The entry of the current applet context islocated within the GlobalPlatform Registry.

If the Application does not have CVM management privileges (bit 2 of the Application Privileges not set), return `false`.

Set CVMFlag to ACTIVE.

Reset the Retry Counter to Retry Limit.

Return `true`.

7.3.10 `boolean` **`setTryLimit (`**`byte bTryLimit`**`)`**

The entry of the current applet context is located within the GlobalPlatform Registry.

If the Application does not have CVM management privileges (bit 2 of the Application Privileges not set), return `false`.

If CVMFlag is BLOCKED, return `false`.

Set Retry Limit to `bTryLimit`.

Reset the Retry Counter to Retry Limit.

Return `true`.

7.3.11 `short` **`verify(`**`byte[] baBuffer, short sOffset, byte bLength, byte bFormat`**`)`**

The entry of the current applet context is located within the GlobalPlatform Registry.

If CVMFlag is BLOCKED, return a short of –1 (`CVM_FAILURE`).

Compare the value of the Global PIN with the data at `sOffset` in `baBuffer` for the length of the Global PIN.

If the comparison is successful:

- Reset the Retry Counter to Retry Limit.
- CVMFlag is set to VALIDATED.
- Return a short of 0 (`CVM_SUCCESS`).

If the comparison is unsuccessful:

- CVMFlag is set to INVALID_SUBMISSION
- Decrement the Retry Counter by 1.
- If Retry Counter equals 0, CVMFlag is set to BLOCKED.
- Return a short of –1 (`CVM_FAILURE`).

7.4 **Interface** GPRegistryEntry

7.4.1 `void` **`deregisterService(`**`short sServiceName`**`)`**

This method is not supported in this guideline, an exception of '6982' is thrown.

7.4.2 `AID` **`getAID()`**

This method returns the Application's AID registered in the current GlobalPlatform Registry’s entry.

Return AID object.

7.4.3 `short` **`getPrivileges(`**`byte baBuffer, short sOffset`**`)`**

This method returns in the baBuffer at sOffset location all the Privileges bytes registered in the current GlobalPlatform registry entry.

Return sOffset + Length of the Privileges.

7.4.4 `byte` **`getState()`**

This method returns the Life Cycle State registered in the current GlobalPlatform Registry entry.

7.4.5 `boolean` **`isAssociated(`**`AID SDAID`**`)`**

The OPEN determines if the SDAID is registered in the current GlobalPlatform Registry’s entry as the associated Security Domain.

Return `True` if the GP Registry references the Security Domain as being associated with this GPRegistryEntry, or `False` otherwise.

7.4.6 `boolean` **`isPrivileged(`**`byte bPrivilege`**`)`**

This method allows an Application (e.g. a CVM Application) to verify if a given Privilege defined by bPrivilege is registered in this GPRegistryEntry (e.g. check the CVM Management privilege of another Application invoking the CVM.update() method). The bPrivilege is the privilege number as defined in Table 6-1 of the GP CS 2.2.1 [1].

Return `True` if at least the referenced Privilege is registered in the GP Registry entry, or

`False` if the referenced Privilege is not registered in the GP Registry entry.

7.4.7 `void` **`registerService(`**`short sServiceName`**`)`**

This method is not supported in this guideline, an exception of '6982' is thrown.

7.4.8 `boolean` **`setState(`**`byte bState`**`)`**

This method allows the Life Cycle state of this GPRegistryEntry to be transitioned to the requested target state.

Return `True` if the transition is successful or `False` otherwise.

7.5 **Interface** SecureChannelx

7.5.1 `void` **`setSecurityLevel(`**`byte bSecurityLevel`**`)`**

This method is not supported in this guideline, an exception of '6982' is thrown.

8 Normative References

Standard / Specification	Description	Ref
GlobalPlatform Card Specification v2.1.1	GlobalPlatform Card Specification v2.1.1	[0]
GlobalPlatform Card Specification v2.2.1	GlobalPlatform Card Specification v2.2.1	[1]
Java Card™ 2.1.1 API	Sun Microsystems Java Card™ 2.1.1 Application Programming Interface, Revision 1.0, May 18, 2000	[2]
Java Card™ 2.1.1 VM Specification	Sun Microsystems Java Card™ 2.1.1 Virtual Machine Specification, Revision 1.0, May 18, 2000	[3]
Java Card™ 2.1.1 JCRE Specification	Sun Microsystems Java Card™ 2.1.1 Runtime Environment (JCRE) Specification, Revision 1.0, May 18, 2000	[4]
Java Card™ 2.2 API	Sun Microsystems Java Card™ 2.2 Application Programming Interface Specification, September 2002	[5]
Java Card™ 2.2 VM Specification	Sun Microsystems Java Card™ 2.2 Virtual Machine Specification, June 2002	[6]
Java Card™ 2.2 JCRE Specification	Sun Microsystems Java Card™ Runtime Environment (JCRE), June 2002	[7]
ISO/IEC 7816-4	ISO/IEC 7816-4	[8]
EMV Book 1 Application Independent ICC to Terminal Interface Requirements v4.2	EMVCo - EMV Integrated Circuit Card Specifications for Payment Systems – Book 1 – Application Independent ICC to Terminal Inteface requirements, Version 4.2, June 2008	[9]

Table 13: Normative References

9 List of Tables

10 Revision History

- November 2006 – Release v1.0.0
  - Initial Release
- January 2011 – Release v1.0.1
  - Applied precisions required in development of the "UICC Configuration v1.0.1"
  - Updated references to GlobalPlatform v2.2.1

END OF DOCUMENT