GLOBALPLATFORM

ADVANCING STANDARDS FOR SMART CARD GROWTH

# Card Specification 2.1 Frequently Asked Questions

**July 2002**

# Table of Contents

Note: the latest Questions and Answers are in blue characters.

# 1. General

## 1.1 GlobalPlatform Data Elements

What is the AID value of the Java Card Export File for GlobalPlatform Card Specification 2.1 API?

The final AID value for the Java Card Export File for GlobalPlatform Card Specification 2.1 API is: '**A00000015100**'. It is based on the RID recently assigned by ISO to GlobalPlatform, i.e. 'A000000151'.

What is the hexadecimal value of the GlobalPlatform Object Identifier (OID)?

The Object Identifier (OID) assigned by ISO to GlobalPlatform (see *appendix F1- Data Values*) is: **{1 2 840 114283}** (see *GlobalPlatform Card Specification 2.1 FAQ, November 2001*). The hexadecimal representation of the GlobalPlatform OID is: '**2A864886FC6B**'.

What is GlobalPlatform Object Identifier (OID) Value?

The Object Identifier (OID) assigned to GlobalPlatform (see *Appendix F1- Data Values*) is as follows:
GlobalPlatform OID ::= {iso(**1**) member-body(**2**) country-USA(**840**) Global-Platform(**114283**)}

What is GlobalPlatform Registered Application Provider Identifier (RID) Value?

The Registered Application Provider Identifier (RID) assigned to GlobalPlatform (see *Appendix F1- Data Values*) is as follows: GlobalPlatform RID = '**A000000151**'. Note that the current Java Card Export File for GlobalPlatform Card Specification 2.1 API available on GlobalPlatform website has a provisional AID, i.e. 'GPPKGEXP01'. It will be updated shortly to reflect the RID recently assigned by ISO to GlobalPlatform.

## 1.2 Card Lifecycle and Application Lifecycle

How to mute a card?

The difference between GlobalPlatform Card Specification 2.0.1' and GlobalPlatform Card Specification 2.1 is that now a card can never be totally mute. The card life cycle state 'Terminated' defined in *section 6.7.3 – Card Termination* disables any communication to and from the card besides the GET DATA command, processed by the Issuer Security Domain.

What entities can initiate the transition from CARD_LOCKED back to SECURED?

The Issuer Security Domain can initiate the transition from CARD_LOCKED back to SECURED. This is explained in *section 5.1.1 – Card Life Cycle States*.

How can an application, via its Security Domain, use a Set Status command to lock itself?

This feature is not available in the GlobalPlatform Card Specification 2.1 version. According to *section 5.3.1.3 – Application Life Cycle State LOCKED*, only the Issuer Security Domain or the OPEN can lock an Application. Allowing a Security Domain to lock one of its Applications or the Application to lock itself may be addressed in a future release of the Specification.

## 1.3 Security Domain

Can an Application share its Security Domain services with another Application which is not associated to that Security Domain?

The current version of GlobalPlatform Card Specification requires that the OPEN controls an Application request to access Security Domain services to the Applications associated with that Security Domain. It does not preclude the Application associated to that Security Domain and authorized to access that Security Domain services to share those with another Application (even not associated to that Security Domain). In other words, if Application A is associated with Security Domain SDa and Application B associated with Security Domain SDb, A can access SDa services and share them with B.

Sharing mechanisms should be used with extra care by applications. For security reasons, it is recommended that Card Issuers and Application Providers verify thoroughly application code before authorizing its load onto a card, especially re: usage of sharing mechanisms.

Can an extradited Application still access its previous Security Domain services?

The OPEN controls an Application request to access Security Domain services to the Applications currently associated with that Security Domain. The control mechanisms depend on the underlying Run-Time Environment implementation. For instance, on a Java Card based smart card, that control is performed when processing the getSecureChannel() method of the org.globalplatform.GPSystem class (see *Appendix A2 – GlobalPlatform on a Java Card*). Once the reference to the Secure Channel object interface is obtained, no extra controls of the Application identity are required by the GlobalPlatform Card Specification for executing a method of the org.globalplatform.SecureChannel interface.

Applications for Java Card based smart cards should not store references to the Secure Channel object interface. For consistency reasons, it is recommended that Card Issuers and Application Providers verify thoroughly application code before authorizing its load onto a card supporting Card Content Extradition, especially re: usage of Security Domain services.

May a Security Domain contain proprietary data?

A Security Domain may contain proprietary data. More information is available in *section 9.3 - Get Data Command* and *section 9.10 - Store Data Command*.

## 1.4 Command/response (APDU)

Is it possible to add search criteria to the GET STATUS command?

*Section 9.4.2.3 – Data Field in the GET STATUS Command Message* defines only one mandatory search criteria: Application AID (tag '4F'). Depending on the card implementation, additional search criteria may be appended to the Application AID and included in the GET STATUS command data field. In such case, the additional search criteria shall be TLV coded and appended after the mandatory search criteria Application AID (tag '4F').
An additional search criteria, that was described in GlobalPlatform Card Specification 2.0.1', is the Application Life Cycle Status (tag '9F70'). An example of use would be to search all Applications and Security Domains (if any) that are in the Life Cycle State SELECTABLE on the card. In such example, the data field of the GET STATUS command would be coded as follows: '4F 00 9F70 01 07'.

Can the search criteria of the GET STATUS command be less than 5 bytes?

Yes. The data field of the GET STATUS command contains the search qualifier that is TLV coded (tag '4F'-length-value) and can be of any length, see *section 9.4.2.3 – Data Field Sent in the GET STATUS Command Message*. Note that a tag of '4F' and a length of zero indicate a search for all the occurrences matching the selection criteria defined in the GET STATUS command parameter P1 (e.g. Applications, Executable Load Files).

Can the key information for all the Secure Channel Keys be retrieved with a single GET DATA command?

Yes. A GET DATA command with the Key Information Template (tag 'E0') retrieves all the key information available for the currently selected Security Domain (including the Issuer Security Domain). When a Security Domain contains multiple Secure Channel Keys, template 'E0' contains as many key information data objects (tag 'C0') as there are Secure Channel Keys, see *section 9.3.3.1 – Data Field Returned in the GET DATA Response Message*.

Are Logical Channels supported?

No. Only Logical Channel 00 is supported in command and response APDUs defined in *section 9 – APDU Command Reference*. Other Logical Channels are not covered in this version of the GlobalPlatform Card Specification and may be included in a future release.

Is there a maximum size of the GlobalPlatform APDU command messages?

Yes. All GlobalPlatform command messages (including the APDU header) are limited to 255 bytes in length. See for instance *section 9.1.5 - APDU command and response data* requiring Lc to be coded on one byte. See also s*ection 9.6.2 - LOAD Command Message, section 9.7.2 – PUT KEY Command Message,* and *section 9.10.2 – STORE DATA Command Message* restricting the overall length of these command messages (read 255 bytes, not 256).

Is the support of the combined INSTALL[for install & make selectable] command optional?

No. The support of the combined INSTALL[for install & for make selectable] command is the minimum requirement for a card to be compliant with GlobalPlatform Card Specification. The support of the individual INSTALL[for install] and INSTALL[for make selectable] commands is an option of the GlobalPlatform Card Specification 2.1. See *section 9.5 – INSTALL command* for further details.

## 1.5 Card Identification and Management

What is the purpose of Card Recognition Data?

GlobalPlatform, in conjunction with MAOSCO and the Java Card Forum, has devised a scheme that allows an off-card system to discover how to work with a smart card of unknown characteristics without having to resort to trial and error (which can be slow and unreliable). The idea is that the card may divulge Card Recognition Data at any time (see *appendix F.2- Structure of Card Recognition Data* for further details), and through it an off-card system can learn how to work with the card.

## How is Card Recognition Data accessed?

The method for accessing Card Recognition Data is as follows:

1. Use the SELECT FILE command ("select by name") with no command data and with the "first or only occurrence" option set. This selects the card management function on the card, and where appropriate returns the identifier of the card management function for subsequent use. Ignore any response status other than '61xx' or '6Cxx'.
2. Issue a GET DATA command as defined in ISO/IEC standard 7816-4 (i.e. with class byte set to '00', see also *section 9.3.2- GET DATA Command Message*) for a data object with tag '66', Card Data. (Note that this command only returns the value field of Card Data, not its tag or length). Card Data, which is a constructed data object, TLV encoded as described in ISO/IEC standard 7816-6, contains another constructed data object, Card Recognition Data, tag '73', which provides details of the card and how to access it, primarily for card management purposes.
3. Check that the Card Recognition Data data object contains the GlobalPlatform Card Recognition Data Object Identifier introduced by tag '06'. The Object Identifier (OID) value for Card Recognition Data is: {globalPlatform 1}**,** that is: {1 2 840 114283 1}, or in hexadecimal: '2A864886FC6B01'.
4. If this fails, then trial-and-error is necessary, as the card does not support Card Recognition Data.

Note that, in a future release of the specification, there may be a further option following step 1, but this relies upon obtaining a "neutral" AID for this purpose. The option would be that if step 1 gave an error status other than '6Cxx', the SELECT FILE command should be issued with the "neutral" AID, and any status other than '61xx' or '6Cxx' should be ignored.

## What is the minimum content of Card Recognition Data?

Card Recognition Data should contain for any card, GlobalPlatform compliant or not, the following minimum set of data objects:

- Tag '73' identifying unambiguously Card Recognition Data by containing the Object Identifier {globalPlatform 1}. This Object Identifier also identifies GlobalPlatform as the Tag Allocation Authority for application tags within Card Recognition Data.
- Application tag 0 (tag '60') identifying the Card Management Type and Version provided by the card. The value field of this data object is an Object Identifier (OID), which is typically based on the OID of the organization that operates the card management scheme; for example, "GlobalPlatform version 2.1", "MULTOS version 5", or "Scheme X Proprietary Card Management version N".
- Application tag 3 (tag '63') identifying the Card Identification Scheme. The value field of this data object is an Object Identifier (OID), which is typically based on the OID of the organization that operates the card identification scheme. This OID, when added as a prefix to the "local" (scheme) card identifier, forms a globally unique card identifier. For instance, if the card uses a data object "Card Number" which is unique within the card identification scheme, then the concatenation "OID | cardNumber" can be used as a globally unique identifier for the card.

Usage of tags defined in *appendix F.2- Structure of Card Recognition Data* shall comply to the GlobalPlatform Card Specification. Application tags are allocated and defined by GlobalPlatform who acts as the Tag Allocation Authority for Card Recognition Data. Proprietary tags may be used to carry additional information.

## How is Card Recognition Data encoded for a GlobalPlatform compliant card?

Card Recognition Data shall be encoded according to *appendix F.2- Structure of Card Recognition Data*. Taking an example of a GlobalPlatform 2.1 compliant card whose Issuer Security Domain supports Secure Channel Protocol '01', this example shows the coding of Card Recognition Data:

a) Universal tag 06: GlobalPlatform Tag Allocation Authority OID: **{GlobalPlatform 1}**
b) Application tag 0: GlobalPlatform Card Specification Version 2.1: **{GlobalPlatform 2 2 1}**
c) Application tag 3: GlobalPlatform Card Identification Scheme: **{GlobalPlatform 3}**
d) Application tag 4: GlobalPlatform Secure Channel Protocol (SCP) '01' with Implementation Option "i" of '05': **{GlobalPlatform 4 1 5}** (1 being the decimal value of SCP Identifier '01'; 5 being the decimal value of SCP Option "i" = '05')

| Coding (hexadecimal) | Meaning |
|---|---|
| **73** | Tag for Card Recognition Data |
| 2E | Length of Card Recognition Data (46 bytes) |
| **06** | Tag for Object Identifier |
| 07 | Length of Object Identifier (7 bytes) |
| 2A864886FC6B01 | Object Identifier for **{GlobalPlatform 1}** or **{1 2 840 114283 1}** which implicitly defines this as Card Recognition Data and GlobalPlatform as the Tag Allocation Authority |
| **60** | Application tag 0: Card Management Type and Version |
| 0B | Length of Card Management Type and Version identifier (11 bytes) |
| 06 | Tag for Object Identifier |
| 09 | Length of Object Identifier (9 bytes) |
| 2A864886FC6B020201 | Object Identifier for **{GlobalPlatform 2 2 1}** or **{1 2 840 114283 2 2 1}** |
| **63** | Application tag 3: Card Identification Scheme |
| 09 | Length of Card Identification Scheme identifier (9 bytes) |
| 06 | Tag for Object Identifier |
| 07 | Length of Object Identifier (7 bytes) |
| 2A864886FC6B03 | Object Identifier for **{GlobalPlatform 3}** or **{1 2 840 114283 3}** |
| **64** | Application tag 4: Secure Channel Protocol and Implementation Option |
| 0B | Length of Secure Channel Protocol and Option identifier (11 bytes) |
| 06 | Tag for Object Identifier |
| 09 | Length of Object Identifier (9 bytes) |
| 2A864886FC6B040105 | Object Identifier for **{GlobalPlatform 4 1 5}** or **{1 2 840 114283 4 1 5}** |

## How can Card Recognition Data be used by non GlobalPlatform cards?

Taking an example of a fictitious card management Scheme X in Romania, the fictitious Object Identifiers of this Scheme X example are:

| Object identifier value | Meaning |
|---|---|
| **{ 1 }** | ISO |
| **{ 1 2 }** | ISO member body |
| **{ 1 2 642 }** | Romania |
| **{ 1 2 642 327 }** | Scheme X |
| **{ 1 2 642 327 1 }** | Scheme X Card Management |
| **{ 1 2 642 327 1 6 }** | Scheme X Card Management Version 6 |
| **{ 1 2 642 327 2 }** | Scheme X Card Numbering Scheme |

Example Card Recognition Data coding of this fictitious Scheme X is:

a) Universal tag 06: GlobalPlatform Tag Allocation Authority OID: **{GlobalPlatform 1}**
b) Application tag 0: Scheme X Card Management Version 6: **{SchemeX 1 6}**
c) Application tag 3: Scheme X Card Identification Scheme: **{SchemeX 2}**

| Coding (hexadecimal) | Meaning |
|---|---|
| **73** | Tag for Card Recognition Data |
| 1E | Length of Card Recognition Data (30 bytes) |
| **06** | Tag for Object Identifier |
| 07 | Length of Object Identifier (7 bytes) |
| 2A864886FC6B01 | Object Identifier for **{GlobalPlatform 1}** or **{1 2 840 114283 1}** which implicitly defines this as Card Recognition Data and GlobalPlatform as the Tag Allocation Authority |
| **60** | Application tag 0: Card Management Type and Version |
| 09 | Length of Card Management Type and Version identifier (9 bytes) |
| 06 | Tag for Object Identifier |
| 07 | Length of Object Identifier (7 bytes) |
| 2A850282470106 | Object Identifier for **{SchemeX 1 6}** or **{1 2 642 327 1 6}** |
| **63** | Application tag 3: Card Identification Scheme |
| 08 | Length of Card Identification Scheme identifier (8 bytes) |
| 06 | Tag for Object Identifier |
| 06 | Length of Object Identifier (6 bytes) |
| 2A8502824702 | Object Identifier for **{SchemeX 2}** or **{1 2 642 327 2}** |

## How are Object Identifiers (OID) encoded?

An Object Identifier (OID) is expressed as a series of integers, inside curly brackets and separated by spaces, for example the GlobalPlatform OID: {1 2 840 114283}. Object Identifiers are used widely to identify organizations, roles, pieces of equipment and other objects. When used in messages, they are coded in binary as follows:

- The first byte is calculated by multiplying the first integer of the OID by 40, and adding the second digit.
- Each subsequent integer in the OID is encoded in binary across a series of bytes. Only the low-order 7 bits of each byte are used. The high order bit of each byte is then set to 1 for all except the last byte, where it is set to zero.
- This is repeated for each integer, and then all the values are concatenated contiguously.

For example, these encoding rules applied to the GlobalPlatform OID give '2A864886FC6B', that is:

| Original Value | hexadecimal equivalent | binary value | binary base 128 (7 bits per byte) | binary base 128 with high order bit set | hexadecimal coding |
|---|---|---|---|---|---|
| 1 2<br>(40x1)+2=42 | 2A | | | | 2A |
| 840 | 348 | 11<br>01001000 | 0000110<br>1001000 | 10000110<br>01001000 | 8648 |
| 114283 | 1BE6B | 1<br>10111110<br>01101011 | 0000110<br>1111100<br>1101011 | 10000110<br>11111100<br>01101011 | 86FC6B |

### 1.6 Certification and licensing

What is the evaluation process of GlobalPlatform 2.1 cards?

> The GlobalPlatform Card Compliance Program for smart cards is now available, see GlobalPlatform website at http://www.globalplatform.org/compliance.asp. The GlobalPlatform Card Compliance Program contains: Card Compliance Packages defining the minimum mandatory and optional functionality, a Card Test Plan specifying the functional tests to perform, a Card Test Suite comprised of test scripts implementing the Card Test Plan, and a Test Tool to execute the Card Test Suite. The GlobalPlatform Card Compliance Packages cover all mandatory functionality and optional features as defined in GlobalPlatform Card Specification version 2.1.

If an issuer wishes to issue GlobalPlatform smart cards, what registration or licensing is required by GlobalPlatform Consortium?

> None. There is no restriction and no license to sign for issuing and using GlobalPlatform compliant smart cards.

# 2. Card Content Management

## 2.1 Loading and Installing

What is the 'condition' for the System Specific Parameters of the Load and Install Parameters?

> In *tables 9-32 – Load Parameter Tags* and *9-33 – Install Parameter Tags*, the System Specific Parameters template shall be present: i.e. tag 'EF' followed by a non-zero length indicator, when one (or more) of the embedded data elements: Non-Volatile Code Space Limit, Volatile Data Space Limit, Non-Volatile Data Space Limit (tags 'C6', 'C7', or 'C8') is present. For further information on these system parameters, see *section 9.5.2.3.6 – INSTALL [for load] and INSTALL [for install] Parameters*.

Is it possible to create several Application Instances and related data from an Executable Module?

> Yes. As described in *section 3.5 – Card Content and figure 3.2 – Card Content Relationships*, more than one Application instance may be created from the same Executable Module. This feature was already present in the GlobalPlatform Card Specification v2.0.1'. The implementation is possible on a current Java Card based smart card. However note that, due to Java Card restrictions no firewall is active between two Application instances of the same Executable Module.

## 2.2 Deletion

How to require a pre-authorization on the Delegated Deletion?

> This version of the GlobalPlatform Card Specification does not provide Tokens for Delegated Deletion. See *section 7.6.4 – Delegated Deletion* for further details. Deletion Tokens may be addressed in a future release of the GlobalPlatform Card Specification.

How is the memory being managed in a GlobalPlatform card after an Application has been deleted?

Physical memory management is beyond the scope of the GlobalPlatform Card Specification. GlobalPlatform smart cards rely on an underlying runtime environment. Memory defragmentation is a runtime environment feature, therefore GlobalPlatform Card Specification *section 6.4.2 – Content Removal* is silent on the subject. For instance, Java Card 2.1.2 doesn't support defragmentation, therefore a scenario where total available space is large enough but fragmented enough that a new load file cannot be loaded is quite possible.

What is the difference between Application Deletion and Executable Load File Deletion?

GlobalPlatform Card Specification *section 6.4.2 – Content Removal* distinguishes 'Application deletion' (which refers to Application instance and Application data, therefore EEPROM) from 'Executable Load File deletion' (which refers to application code that may reside in ROM or EEPROM). Physical deletion only applies to 'Executable Load Files' and 'Application instances and data' residing in EEPROM.

Are related Application instances also removed on deletion of the Executable Load File?

Each Application instance has to be deleted before the corresponding Executable Load File can be deleted. See *section 6.4.2.2 – Executable Load File Removal* for further details.

How upgrading from an older version of an application software?

The GlobalPlatform Card Specification 2.1 does not include on-card application software upgrades without loss of application data. The current version of the Specification only allows deleting an Application and its corresponding Executable Load File followed by loading the latest version of the Executable Load File and re-instantiating the Application. According to *section 6.4.2 – Content Removal*, deleting an Application instance, along with all its data, is required before deleting its corresponding Executable Load File. On-card application software upgrades may be addressed in a future release of the Specification.

Is it possible to delete multiple objects in a single DELETE command?

No. Only one object is deleted per DELETE command, except in case of keys where multiple keys with the same Key Identifier (tag ‘D0’ in the command data field) or the same Key Version Number (tag ‘D2’ in the command data field) may be deleted as described in *section 9.2 – DELETE command*. The following known limitation is acknowledged: deletion of Applications cross-referencing each other is not defined in GlobalPlatform Card Specification 2.1: such mechanism is beyond the scope of the current version of the Specification.

Is deletion allowed when the card is in the Life Cycle State CARD_LOCKED?

No. According to *section 5.1.1.4 - Card Life Cycle State CARD_LOCKED*, all card content functionality is prohibited when the card is in the Life Cycle State CARD_LOCKED, including Application and Executable Load File removal.

## 2.3 Personalization

Where to find Personalization related items?

The GlobalPlatform Card Specification 2.1 defines a set of mandatory Data Elements in *section 9.10.2.3 – Data field in the STORE DATA command message*. Keys for Secure Channel Protocols are detailed in *Appendix C.1 – Keys*, *Appendix D.2 – Cryptographic K*eys for Secure Channel Protocol '01', and *Appendix E.2 - Cryptographic K*eys for Secure Channel Protocol '02'. Personalization of other data elements is Issuer and/or Application Provider specific and is not covered by the GlobalPlatform Card Specification.

Why should the Card Recognition Data not exceed 127 bytes?

The Card Recognition Data is defined in *section 6.8.3 – Card Recognition Data* and further refined in *Appendix F.2 – Structure of Card Recognition Data*. The recommendation of coding the length field on one byte is intended to express the need to keep this data as small as possible

## 2.4 CVM

Is the CVM Handler optional?

Yes. According to *section 6.1.3 – CVM Handler*, all CVM services described in the specification are optional. For instance, when the CVM services are not available on a Java Card based smart card, the getCVM()method of the class org.globalplatform.GPSystem shall return a null reference, see *Appendix A2 – GlobalPlatform on a Java Card*.

How to retrieve the status of the CVM?

The CVM status of the CVM service provided by Card Manager can be retrieve by any on-card Application using the GlobalPlatform API. For instance on a Java Card based smart card, the following methods: isActive(), isSubmitted(), isVerified(), and isBlocked() of the interface org.globalplatform.CVM are provided, see *Appendix A2 – GlobalPlatform on a Java Card*.

How to reset the CVM Retry Counter?

The GlobalPlatform Card Specification 2.1 defines the CVM services in *section 6.1.3 –CVM Handler* and *section 6.9 – CVM Management*. The CVM Handler is responsible for cardholder verification velocity checking. The Applications use the GlobalPlatform API to request services from the CVM Handler and reset the CVM Retry Counter. For instance on a Java Card based smart card, the resetAndUnblockState() method of the interface org.globalplatform.CVM defined in *Appendix A2 – GlobalPlatform on a Java Card* resets the Retry Counter to the Retry Limit value.

How is the CVM length expressed for the BCD format?

The CVM length is always defined in bytes regardless of the CVM format. For instance on a Java Card based smart card, in the update() and verify() methods of the interface org.globalplatform.CVM the bLength parameter reflects the length in bytes of the value given in the parameter baBuffer (not the number of digits of the BCD format), see *Appendix A2 – GlobalPlatform on a Java Card*.

# 3. Security

## 3.1 Secure Channel Protocol

What is the appropriate APDU command to terminate a Secure Channel Session?

The appropriate APDU command terminating a Secure Channel Session may be a SELECT command that is selecting another Application (see 3rd bullet of *section 8.2.3 – Secure Channel Termination*) or any other proprietary command of the currently selected Application triggering the termination of the current Secure Channel Session through the use of the appropriate GlobalPlatform API (e.g. resetSecurity() method of the org.globalplatform.SecureChannel interface, see *Appendix A.2 – GlobalPlatform on a Java Card*).

What are the rules defining the Security Level of a Secure Channel Session applicable to Secure Channel Protocol '01'?

The Security Level of a communication not included in a Secure Channel Session shall be set to NO_SECURITY_LEVEL.
In Secure Channel Protocol '01', the Security Level established in a Secure Channel Session is a combination of the following values: AUTHENTICATION, C_MAC, and C_DECRYPTION.
The Secure Channel Security Level shall be set as follows:

- NO_SECURITY_LEVEL when a Secure Channel Session is closed or not yet fully initiated.
- AUTHENTICATION after a successful processing of an EXTERNAL AUTHENTICATE command: AUTHENTICATION shall be cleared once the Secure Channel Session is closed.
- AUTHENTICATION & C_MAC after a successful processing of an EXTERNAL AUTHENTICATE command with P1 indicating C-MAC (P1='01'): AUTHENTICATION & C-MAC shall be cleared once the Secure Channel Session is closed.
- AUTHENTICATION & C_MAC & C_DECRYPTION after a successful processing of an EXTERNAL AUTHENTICATE command with P1 indicating C-MAC and Command Encryption (P1= '03'): AUTHENTICATION & C_MAC & C_DECRYPTION shall be cleared once the Secure Channel Session is closed.

See *GlobalPlatform Card Specification 2.1 Errata and Precisions List 0.1, November 2001*.

What are the rules defining the Security Level of a Secure Channel Session applicable to Secure Channel Protocol '02'?

The Security Level of a communication not included in a Secure Channel Session shall be set to NO_SECURITY_LEVEL.
In Secure Channel Protocol '02' with Explicit Initiation mode, the Security Level established in a Secure Channel Session is a combination of the following values: AUTHENTICATION, C_MAC, R_MAC, and C_DECRYPTION. The Secure Channel Security Level shall be set as follows:

- NO_SECURITY_LEVEL when a Secure Channel Session is closed or not yet fully initiated.
- AUTHENTICATION after a successful processing of an EXTERNAL AUTHENTICATE command: AUTHENTICATION shall be cleared once the Secure Channel Session is closed.
- C_MAC after a successful processing of an EXTERNAL AUTHENTICATE command with P1 indicating C-MAC (P1='x1' or 'x3'): C-MAC shall be cleared once the Secure Channel Session is closed. Note that C_MAC is always combined with AUTHENTICATION and simultaneously set and cleared.

- C_DECRYPTION after a successful processing of an EXTERNAL AUTHENTICATE command with P1 indicating Command Encryption (P1= 'x3'): C_DECRYPTION shall be cleared once the Secure Channel Session is closed. Note that C_DECRYPTION is always combined with AUTHENTICATION & C_MAC and simultaneously set and cleared.
- R_MAC after a successful processing of an EXTERNAL AUTHENTICATE command with P1 indicating R-MAC (P1='1x'): R-MAC shall be cleared once the Secure Channel Session is closed. Note that in this case R_MAC is always combined with AUTHENTICATION and simultaneously set and cleared. R_MAC may also be combined with C_MAC or C_DECRYPTION (according to the P1 value of the EXTERNAL AUTHENTICATE command) and simultaneously set and cleared.
- R_MAC after a successful processing of a BEGIN R-MAC SESSION command: R-MAC shall be cleared after a successful processing of an END R-MAC SESSION command. Note that in this case R_MAC may be combined with AUTHENTICATION, C_MAC, or C_DECRYPTION depending on the pre-existing Security Level of the Secure Channel Session. R_MAC is set and cleared independently of AUTHENTICATION, C_MAC, or C_DECRYPTION.

In Secure Channel Protocol '02' with Implicit Initiation mode, the Security Level established in a Secure Channel Session is a combination of the following values: AUTHENTICATION, C_MAC, and R_MAC. The Secure Channel Security Level shall be set as follows:

- NO_SECURITY_LEVEL when a Secure Channel Session is closed or not yet initiated.
- AUTHENTICATION & C_MAC after a successful verification of the C- MAC of the first command message with a C-MAC: C_MAC shall be cleared after the reception of the first command message without a C-MAC. AUTHENTICATION & C_MAC shall be cleared once the Secure Channel Session is closed.
- R_MAC after a successful processing of a BEGIN R-MAC SESSION command: R-MAC shall be cleared after a successful processing of an END R-MAC SESSION command.

See *GlobalPlatform Card Specification 2.1 Errata and Precisions List 0.1, November 2001*.

What happens during a Secure Channel Session requiring secure messaging if an unsecured Get Data command is sent to the card?

In Secure Channel Protocol '01' and Secure Channel Protocol '02' with Explicit Initiation mode, one can initiate a Secure Channel Session with the following required Security Level: AUTHENTICATION & C_MAC (and optionally C_DECRYPTION and/or R_MAC). According to *section 8.2.3 – Secure Channel Termination*, on reception of a command message without the required secure messaging, the Secure Channel Session is terminated and the Security Level reset to NO_SECURITY_LEVEL. Note that, on reception of a command message with an incorrect secure messaging, the Secure Channel Session is terminated and the Security Level reset to NO_SECURITY_LEVEL.

In Secure Channel Protocol '02' with Implicit Initiation mode, the required Security Level of a Secure Channel Session is: AUTHENTICATION. Since the Secure Channel Session was opened on reception of a command with a correct C-MAC, the current Security Level of the Secure Channel Session is set to: AUTHENTICATION & C_MAC. On reception of a command message without C-MAC, the Secure Channel Session Security Level is reset to AUTHENTICATION and the Secure Channel Session remains open. Note that, according to *section 8.2.3 – Secure Channel Termination*, on reception of a command message with an incorrect C-MAC, the Secure Channel Session is terminated and the Security Level reset to NO_SECURITY_LEVEL.

## How many Secure Channel Protocols can be implemented on a card?

According to *section 4.3 – Cryptographic support*, the Issuer Security Domain shall implement one Secure Channel Protocol. A Security Domain other than the Issuer Security Domain shall also implement one Secure Channel Protocol, which may differ from the Secure Channel Protocol of the Issuer Security Domain. Furthermore, different Security Domains may implement different Secure Channel Protocols. Then, depending on the card implementation, more than one Secure Channel Protocol may be available on a given card. Note that *Appendix F2 – Structure of Card Recognition Data* and *Appendix F3 – Security Domain Management Data* provide a means with Application Tag 4 (tag '64') for an off-card entity to be aware of the Secure Channel Protocol supported by a Security Domain of a specific card.

## What is the difference between the EXTERNAL AUTHENTICATE and BEGIN R-MAC SESSION commands?

When P1 indicates 'R-MAC' (P1 = '1x') in the EXTERNAL AUTHENTICATE command, the R-MAC shall be computed and returned with all subsequent response messages of the current Secure Channel Session, see *Appendix E.5.2 – EXTERNAL AUTHENTICATE Command-Response APDU*.
The BEGIN R-MAC SESSION command allows R-MACs to be computed for a subset of the commands received in a Secure Channel Session, or even (for Secure Channel Protocol '02' in Implicit Initiation mode) outside any Secure Channel Session. It also gives a choice for returning the R-MAC:

- P1 indicates 'no secure messaging' (P1 = '00'), the R-MAC shall be computed with all subsequent response messages and only returned with the END R-MAC SESSION response message,
- P1 indicates 'R-MAC' (P1 = '10'), the R-MAC shall be computed and returned with all subsequent response messages,

see *Appendix E.5.3 – BEGIN R-MAC SESSION Command-Response APDU*.

## What is the difference between the Secure Channel Protocol S_ENC and DEK keys?

The S-ENC key is applicable to APDU command data field encryption/decryption and, in the Explicit Initiation mode, Authentication Cryptogram. See *Appendix D.3.4 – APDU Data Field Encryption and Decryption* and *Appendix D.3.2 – Authentication Cryptograms* for Secure Channel Protocol '01'. See *Appendix E.4.6 – APDU Data Field Encryption and Decryption* and *Appendix E.4.2 – Authentication Cryptograms in Explicit Secure Channel Initiation* for Secure Channel Protocol '02'. Note that command message data field encryption and S_ENC key do not apply to Secure Channel Protocol '02' in Implicit Initiation mode.
The DEK key is applicable to sensitive data encryption/decryption. See *Appendix D.3.5 – Key Sensitive Data Encryption and Decryption* for Secure Channel Protocol '01' and *Appendix E.4.7 – Key Sensitive Data Encryption and Decryption* for Secure Channel Protocol '02'.

## How is the Sequence Counter managed for Secure Channel Protocol '02' in the Explicit Initiation mode?

According to *Appendix E.4.2.1 – Explicit Secure Channel Initiation,* the Sequence Counter is incremented when both the External Authenticate command MAC and the host cryptogram are valid and successfully verified by the card; otherwise the Sequence Counter is not incremented.

How to include the APDU command message in the R-MAC computation of Secure Channel Protocol ‘02’?

The application should always call the unwrapping service of its associated Security Domain on the incoming APDU command message, regardless of the existence or not of a Secure Channel Session (C_MAC and/or C_DECRYPTION). The Security Domain shall know the level of security involved in the current Secure Channel Session, e.g. having previously processed either External Authenticate (P1 set to '1x') or Begin R-MAC Session commands. When the security level is set to R_MAC, the Security Domain shall begin computing the R-MAC with the command data and Lc received when processing the unwrapping services (for a Java Card based smart card, see the unwrap() method of the interface org.globalplatform. SecureChannel in *Appendix A2 – GlobalPlatform on a Java Card*).
Once the response data is ready for transmission, the wrapping service shall then be called by the application in order for the Security Domain to complete the computation of the R-MAC (for a Java Card based smart card, see the wrap() method of the interface org.globalplatform. SecureChannel in *Appendix A2 – GlobalPlatform on a Java Card*).

How to include the status words in the R-MAC computation of Secure Channel Protocol ‘02’?

Once the response data is ready for transmission, the wrapping service is called by the application (for a Java Card based smart card, see the wrap() method of the interface org.globalplatform.SecureChannel in *Appendix A2 – GlobalPlatform on a Java Card*). According to *section E.4.5 - APDU Response R-MAC Generation and Verification*, the Status Words are included in R-MAC calculation and shall be appended by the application at the end of the response data to be MACed in order for the Security Domain to compute properly the R-MAC (for a Java Card based smart card, at the end of the baBuffer of the wrap() method).

What is the algorithm for computing the R-MAC?

The algorithm for computing the R-MAC in Secure Channel Protocol ‘02’ is identical to the one used for the C-MAC generation: the Retail MAC, see *Appendix B.1.2.2 - Single DES Plus Final Triple DES MAC*.

## 3.2 Key Management

What are the attributes of the Initial Key(s)?

The Initial Key(s) is(are) intended to support the Secure Channel Protocol of the Issuer Security Domain as soon as the card is in the OP_READY state, see *section 5.1.1.1 – Card Life Cycle State OP_READY*. The number of Initial Key(s) is dependent of the Secure Channel Protocol and eventually dependent of additional card scheme and/or card issuer specific requirements, e.g. 1 or 3 for Secure Channel Protocol ‘01’ and ‘02’. The Key Type of the Initial Key(s) is dependent of the Secure Channel Protocol, e.g. Key Type ‘80’ (DES) for Secure Channel Protocol ‘01’ and ‘02’. The Initial Key(s) Key Identifier and Key Version Number are outside the scope of GlobalPlatform Card Specification and ultimately depend on card scheme and/or card issuer specific requirements.

What are the values acceptable for Key Identifiers and Key Version Numbers?

According to *section 6.8.4 – On-card Key Information*, there is no restriction for attributing values to Key Identifiers and Key Version Numbers. Key Identifiers and Key Version Numbers shall be coded on one byte according to *table 9.17 – Key Information Data Structure* of *section 9.3.3 – Get Data Response Message*. Any value between '00' and FF' may be attributed by a key management scheme for a specific key.

Use of the PUT KEY command brings some restrictions on the acceptable value ranges. When using the PUT KEY command to add or replace a key on the card, the Key Version Number shall have a value between '01' and '7F' and the Key Identifier a value between '00' and '7F', see *section 9.7.2 – PUT KEY Command Message*. Note that some keys may be loaded by other means, e.g. the very initial key that is loaded prior to the Card Life Cycle State OP_READY (see *section 5.1.1.1 – Card Life cycle State OP_READY*).

Are application keys deleted when the application instance is deleted?

According to *section 6.4.2.1 – Application Removal*, Application keys are to be handled like any other Application data and be deleted when the Application instance is deleted, i.e. the corresponding memory space is reclaimed.

## 3.3 Cryptographic Algorithms

Is data padding performed by the sensitive data encryption and decryption services of a Security Domain?

In the current version 2.1 of the GlobalPlatform Card Specification, both Secure Channel Protocols '01' and '02' do not define any padding scheme for sensitive data encryption and decryption services. Therefore, a Security Domain, providing the services of Secure Channel Protocol '01' or '02', does not discard any data of the decrypted text data (unaware if any padding is present) when processing the sensitive data decryption service (for a Java Card based smart card, see the decryptData() method of the interface org.globalplatform.Secure Channel in *Appendix A2 – GlobalPlatform on a Java Card*).
Similarly, a Security Domain, providing the services of Secure Channel Protocol '01' or '02', does not apply any padding data to the clear text data prior to encryption when processing the sensitive data encryption service (for a Java Card based smart card, see the encryptData() method of the interface org.globalplatform.SecureChannel in *Appendix A2 – GlobalPlatform on a Java Card*).

## 3.4 Other Security

How to trace and log events on a GlobalPlatform card?

The detailed implementation for tracing and logging events is not covered by this version of the GlobalPlatform Card Specification. S*ection 6.7.5 – Tracing and Event Logging* describes it as an optional feature. It may be completed in a future release of the Specification.

# 4. API

## 4.1 Java Card

What are the parameters of the Application.processData() method when a Security Domain forwards an APDU to the Application?

The APDU command forwarded by the Security Domain shall be unwrapped according to the Security Level of the current Secure Channel Session. The baBuffer parameter is the APDU buffer, the sOffset is set to zero, and the sLength is sent to the length of the unwrapped APDU command message.

Can an Application use the getSecurityLevel() method during its processing of the processData() method?

Yes. The APDU command forwarded by the Security Domain to the Application has been unwrapped according to the Security Level of the current Secure Channel Session. The Application may interrogate the current Security Level with the getSecurityLevel() method of the interface org.globalplatform.SecureChannel. There is no restriction on the usage of the SecureChannel interface while the processData() method of the interface org.globalplatform.Application is executed. For instance, the decryptData() method may also be called by the Application.

Can an Application use the decryptData() or encryptData() methods when a Secure Channel Session is not opened?

No. A Secure Channel Session must be active in order for the Security Domain to correctly process the decryptData() and encryptData() methods of the interface org.globalplatform.SecureChannel, e.g. for deriving the DEK session key of Secure Channel Protocol '02'. If a Secure Channel Session is not opened prior to the methods being called by an Application, an ISO exception 'Security Status Not Satisfied' will be thrown.

What is the full name of the Deprecated GlobalPlatform Java Card API package?

The deprecated API for Java Card defined in *Appendix A1 – Deprecated GlobalPlatform Java Card API* allows backward compatibility with existing applications developed for GlobalPlatform 2.0.1' cards. Its full package name is 'visa.openplatform'.

How long will the deprecated API for Java Card be supported?

The deprecated API for Java Card defined in *Appendix A1 – Deprecated GlobalPlatform Java Card API* allows backward compatibility with existing applications developed for GlobalPlatform 2.0.1' cards. It will remain valid at least until the next release of the GlobalPlatform Card Specification. Note that its presence on-card is optional.

How are the exceptions of the CVM interface handled?

On a Java Card based smart card, when processing any method of the interface org.globalplatform.CVM, all exceptions are caught internally by the CVM interface method which returns a boolean status set to false.

## 4.2 Other API

Is there an API between Security Domains and OPEN e.g. for DAP verification?

The GlobalPlatform Card Specification 2.1 *section 7.5 – DAP Verification* and *section 7.6 – Delegated Management* do not include a definition of an API between OPEN (or OP Registry) and Security Domains. *Section 7.1 – Overview* of the Application Providers' Security Domains explicitly states in its last paragraph that the current version of the specification does not define such an API. However such an API may be considered in a future release of the GlobalPlatform Card Specification.

## 5. Configuration and Security Policies

The GlobalPlatform Card Specification offers many choices to Card Issuers and Application Providers for selecting options and implementing features left open in the specification. Following is a list of such options and features available to Card Issuers and Application Providers choices.

### Policy for Key Identification

According to *Errata & Precisions List 0.3 – July 2002* expanding *section 9.7.1 – PUT KEY Command Definition and Scope*, a key may be replaced with the same key identification attributes: Key Identifier and Key Version Number. Note this possibility was present in version 2.0.1' of GlobalPlatform Card Specification.
A configuration policy for accepting (or rejecting) a key replacement with identical key identification attributes must be defined and enforced on-card for each Security Domain, including the Issuer Security Domain.

### Policy for Load File Data Block Hash Verification

According to *Errata & Precisions List 0.3 – July 2002* expanding *section 9.5.2.3.1 – Data Field for INSTALL [for load]*, a Load File Data Block Hash present in the INSTALL [for load] command may be systematically verified by the OPEN, beyond Delegated Management and DAP Verification cases.
A configuration policy for systematically verifying (or not) a Load File Data Block Hash present in the INSTALL [for load] command must be defined and implemented on-card for the OPEN.

### Policy for Security Domain Deletion

The Card Issuer is allowed, once authenticated by the Issuer Security Domain, to delete any Application present on the card (see *section 6.4.2 – Content Removal*). This Card Issuer's privilege includes deleting any Security Domain, regardless of the Security Domain's own privileges. In order to implement a Controlling Authority or card scheme policy, the Card Issuer's deletion privilege may be limited to exclude specific categories of Security Domain such as those with DAP Verification or Mandated DAP Verification privilege.
A security policy for limiting (or not) Security Domain's deletion must be defined and enforced on-card for the OPEN.

### Policy for Extradition

According to *section 7.1 – Security Domains Overview*, a Security Domain may accept or reject Extradition request, see also *section 6.4.3 – Content Extradition*. Extradition acceptance criteria are not defined in the GlobalPlatform Card Specification.
A configuration policy for accepting (or rejecting) Extradition requests must be defined and enforced on-card for each Security Domain.

### Policy for Velocity Checking

*Section 6.7.4 – Operational Velocity Checking* recommends the use of velocity checks regarding Card Content load and installation, memory allocation, and exceptions. Other velocity checks may also be implemented regarding for instance key usage, see *section 6.8.4 – On-card Key Information*. Actions to be taken in case of violation are not defined in the GlobalPlatform Card Specification: they may encompass locking an Application, locking a key, locking or terminating the card.
A security policy for Velocity Checking operations must be defined and enforced on-card for the OPEN as well as each Security Domain, including the Issuer Security Domain..

### Policy for Receipts generation

According to *section 6.5 – Delegated Management*, the Card Issuer security policy may require the generation of Receipts for Delegated Card Content Management (see also *appendix C.4 – Receipts*).
A configuration policy for generating Receipts (or not) must be defined and implemented on-card for the Issuer Security Domain.

### Policy for DAP Verification

According to *appendix C.5 – DAP Verification*, a Security Domain with DAP Verification privilege may implement either a PKCS scheme or a DES scheme for verifying a Load File Data Block Signature (see also *appendix C.1.2 – Secure Content Management Security Domain Keys*).
A configuration policy defining the DAP Verification algorithm must be set up and implemented on-card for each Security Domain supporting DAP Verification.

### Policy for APDU Minimum Security Requirements

The GlobalPlatform Card Specification defines some minimum security requirements for the APDU commands defined in the specification, see *table 9.2 – Minimum Security Requirements for GP Commands*, *table D.2 – Minimum Security Requirements for SCP '01' Commands*, and *table E.4 – Minimum Security Requirements for SCP '02' Commands*. A Card Issuer or an Application Provider may define its own security policy for some (or all) of the APDU commands defined in the specification. For instance, a Card Issuer (or Application Provider) security policy may be based on the Card Life Cycle State, e.g. higher security requirements when the card is in the Life Cycle State SECURED.
A security policy for APDU commands (at least the one defined in tables 9-2, D-2, and E-4) must be defined and enforced on-card for each Security Domain, including the Issuer Security Domain.

### Policy for loading Public Key values

According to *section 9.7.2.3.3 – PUT KEY Command Processing Rules*, public key values may be presented in clear text or encrypted form. The on-card receiving entity (e.g. a Security Domain) is assumed to implicitly know which encryption policy to apply when processing the PUT KEY command for public keys.
A configuration policy for encrypting (or not) public key values must be defined and enforced on-card for each Security Domain, including the Issuer Security Domain.

### Policy for Key Check Values

According to *section 9.7.2.3.3 – PUT KEY Command Processing Rules*, before adding or replacing keys, the on-card receiving entity (e.g. a Security Domain) is assumed to verify the Key Check Value that is present in the PUT KEY command message. The Key Check Value algorithm is not defined in the GlobalPlatform Card Specification.
A configuration policy defining both the required presence (or not) and the algorithm of the Key Check Value verification must be set up and enforced on-card for each Security Domain, including the Issuer Security Domain.

## Policy for Secure Channel Key(s) identification

According to *appendices D.3.1 – Secure Channel Protocol '01' DES Session Keys* and *E.4.1 – Secure Channel Protocol '02' DES Session Keys*, session keys are created using the static Security Domain Secure Channel Key(s). The (Issuer) Security Domain is assumed to implicitly know how to identify its Secure Channel Key(s). When the set of Secure Channel Keys comprises 3 keys, the (Issuer) Security Domain is also assumed to implicitly know how to identify the S-MAC Key (i.e. appropriate S-MAC Key Identifier and Key Version Number) for deriving the S-MAC Session Key, and similarly identify the S-ENC Key and eventually the DEK Key.
A configuration policy for unambiguously identifying the Secure Channel Key(s) with the appropriate Key Identifier(s) and Key Version Number must be defined and implemented on-card for each Security Domain, including the Issuer Security Domain.

## Policy for Secure Content Management Key(s) identification

According to *appendix C.1 – Secure Content Management Keys*, when Delegated Management is supported, the Issuer Security Domain is assumed to implicitly know how to identify its Token Verification Key and optional Receipt Generation Key. A Security Domain supporting DAP Verification is also assumed to implicitly know how to identify its DAP Verification Key.
A configuration policy for unambiguously identifying the Secure Content Management Keys with the appropriate Key Identifiers and Key Version Numbers must be defined and implemented on-card for the Issuer Security Domain supporting Delegated Management and for each Security Domain supporting DAP Verification.