GLOBALPLATFORM

ADVANCING STANDARDS FOR SMART CARD GROWTH

Card Specification 2.1
Errata & Precisions List 0.3
July 2002

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights or other intellectual property rights of which they may be aware which might be infringed by the implementation of the specification set forth in this document, and to provide supporting documentation.

Table of Contents

Table of Errata and Precisions

The following table classifies all the Errata and Precisions of the different Errata and Precisions Lists into a sequential order that reflects the Card Specification index. The latest Errata and Precisions are in blue characters.

Errata / Precision number	Card Specification section number	Description
P.2.1	section 5.3.1.5	Application Life Cycle Specific State Transitions
E.2.1	section 5.3.2.4	Security Domain Life Cycle State LOCKED
P.3.1	sections 5.3.2.4, 7.5	Security Domain Life Cycle State LOCKED
P.1.3	section 6.4.1	Figure 6-3 Load and Installation Flow Diagram
P.3.2	section 6.4.2.1	Security Domain Deletion
P.3.3	section 6.9.1	Resetting the CVM State
E.2.2	section 9.3.1	Key Information using GET DATA command
P.3.4	sections 9.5.2.3, 6.4.1	Load File AID in the INSTALL [for load] command
P.3.5	section 9.5.2.3	Load File Data Block Hash in the INSTALL [for load] command
E.2.3	sections 9.6.2, 9.7.2, 9.10.2	Maximum size of APDU commands
P.3.6	section 9.7.1	Key Replacement with the same key identification attributes
E.3.1	section 9.7.3.2	PUT KEY command status words coding
P.3.7	section 9.8.2.3	SELECT Command with no data
E.2.4	section 9.9.2.2	SET STATUS command parameter P2
P.2.2	section 9.9.2.2	SET STATUS command: Life Cycle State transitions
E.1.4	section 9.10.2	Le for STORE DATA command
P.1.5	appendix A.2	Atomicity for GlobalPlatform API on Java Card
P.2.3	appendix A.2	GlobalPlatform API invocation from within install() method
E.1.2	appendix A.2	Naming of GlobalPlatform interface CVM for Java Card
E.3.2	appendix A.2	GlobalPlatform API: processSecurity() method exception coding
P.2.5	appendix A.2	GlobalPlatform API: decryptData() & encryptData() method errors
P.1.6	appendix A.2	GlobalPlatform API: encryptData() method exceptions
P.2.6	appendix A.2	GlobalPlatform API: unwrap() method errors
P.1.7	appendix A.2	GlobalPlatform API: getSecurityLevel() method response
P.2.4	appendix A.2	GlobalPlatform API: setATRHistBytes() method errors
P.3.8	appendix A.2	GlobalPlatform API: CVM.update() and CVM.setTryLimit methods
P.1.4	appendix A.2	CVM Format conversion
E.1.1	appendices D.1.1, E.1.1	Coding of Implementation Option “i”
E.2.5	appendices D.3.4, E.4.6	APDU Data Field Encryption and Decryption
E.2.6	appendices D.4.2.2, E.5.2.2	Le for EXTERNAL AUTHENTICATE command
P.1.8	appendix E.1.2.2	Sequence Counter in Secure Channel Protocol ‘02’
E.1.3	appendix E.4.2	Authentication Cryptograms computation for Secure Channel Protocol ‘02’
P.3.9	appendix E.4.5	Length of Response Data in R-MAC
E.2.7	appendix E.4.7	SCP ‘02’ Sensitive Data Encryption and Decryption
P.1.9	appendix E.5.3.2	Coding of P2 for BEGIN R-MAC SESSION command
P.1.10	appendix E.5.4.2	Coding of P1 and P2 for END R-MAC SESSION command
P.1.1	appendix F1	GlobalPlatform Object Identifier (OID) Value

P.1.2	appendix F1	GlobalPlatform Registration Identifier (RID) Value
P.2.7	appendix F1	Card Recognition Data Object Identifier value
P.2.8	appendix F1	AID value for Java Card Export File for GlobalPlatform API
P.3.10	appendix F1	Default AID value for Issuer Security Domain

E. ERRATA LISTS

E.1 Errata List 0.1

E.1.1 Coding of Implementation Option "i"

Values of Implementation Option data element "i" are incorrectly coded.

In *Appendix D - Secure Channel Protocol '01' section D.1.1 – SCP01 Secure Channel*, the value of Implementation Option data element "i" shall be '**05**' (not '50'). The last paragraph shall read:
"In SCP01 the card shall support the following implementation option defined by "i" (see *Appendix F - GlobalPlatform Data Values and Card Recognition Data*):

- "i" = '**05**': Initiation Mode explicit, C-MAC on modified APDU, ICV set to zero, 3 Secure Channel Keys"

In *Appendix E - Secure Channel Protocol '02' section E.1.1 – SCP02 Secure Channel*, the values of Implementation Option data element "i" shall be '**04**' (not '40'), '**05**' (not '50'), '**0A**' (not 'A0'), and '**0B**' (not 'B0'). The last paragraph shall read:
"In SCP02 the card shall support at least one of the following implementation options as defined by "i" (see *Appendix F - GlobalPlatform Data Values and Card Recognition Data*):

- "i" = '**04**': Initiation Mode explicit, C-MAC on modified APDU, ICV set to zero, 1 Secure Channel Base Key"
- "i" = '**05**': Initiation Mode explicit, C-MAC on modified APDU, ICV set to zero, 3 Secure Channel Keys"
- "i" = '**0A**': Initiation Mode implicit, C-MAC on unmodified APDU, ICV set to MAC over AID, 1 Secure Channel Base Key"
- "i" = '**0B**': Initiation Mode implicit, C-MAC on unmodified APDU, ICV set to MAC over AID, 3 Secure Channel Keys"

E.1.2 Naming of GlobalPlatform interface CVM for Java Card

The GlobalPlatform interface CVM is incorrectly named.

In *Appendix A.2 – GlobalPlatform on a Java Card*,

- The correct name of the CVM interface shall be 'org.**globalplatform**.CVM' (not 'org.globalplatformx.CVM')
- The method getCVM() of the org.globalplatform.GPSystem class shall return the CVM object reference of type 'org.**globalplatform**.CVM' (not 'org.globalplatformx.CVM'): "public static org.**globalplatform**.CVM getCVM(byte bCVMIdentifier)"

E.1.3 Authentication Cryptograms computation for Secure Channel Protocol '02'

In *Appendix E - Secure Channel Protocol '02'*, the text in *section E.4.2 –Authentication Cryptograms in Explicit Secure Channel Initiation* is misleading.

The 16 byte-block for computing the Card Authentication Cryptogram shall be the concatenation of the 8-byte Host Challenge with the **Sequence Counter and Card Challenge** (not '8-byte Card Challenge'). Similarly, the 16 byte-block for computing the Host Authentication Cryptogram shall be the concatenation of the **Sequence Counter and Card Challenge**' (not '8-byte Card Challenge') with the 8-byte Host Challenge. The INITIALIZE UPDATE response described in

section E.5.1.6 – INITIALIZE UPDATE Response Message contains the Sequence Counter (2 bytes) followed by the Card Challenge (6 bytes).

The first paragraph of *section E.4.2.1 – Card Authentication Cryptogram* shall read as follows:
"The generation and verification of the Card Cryptogram is performed by concatenating the 8-byte Host Challenge, **2-byte Sequence Counter, and 6-byte Card Challenge** resulting in a 16-byte block."

The first paragraph of *section E.4.2.2 – Host Authentication Cryptogram* shall read as follows:
"The generation and verification of the Host Cryptogram is performed by concatenating the **2-byte Sequence Counter, 6-byte Card Challenge**, and 8-byte Host Challenge resulting in a 16-byte block."

E.1.4 Le for STORE DATA command

In *section* 9*.10.2 – STORE DATA Command Message*, the parameter Le shall be **absent** (not set to '00') since no response data is expected per *section* 9*.10.3 – STORE DATA Response Message*.

Table 9-56 - STORE DATA Command Message shall read as follows:

Code	Value	Meaning
CLA	'80' or '84'	
INS	'E2'	STORE DATA
P1	'xx'	Reference control parameter P1
P2	'xx'	Block Number
Lc	'xx'	Length of data field
Data	'xxxxx…'	Application data and MAC (if present)
Le		**Not present**

E.2 Errata List 0.2

E.2.1 Security Domain Life Cycle State LOCKED

The text in *section 5.3.2.4 – Security Domain Life Cycle State LOCKED* describing the Application access to the Security Domain services is in contradiction with the requirements set forth in *section 6.6.2.5 – Associated Security Domain AID*.

The 4th paragraph of *section 5.3.2.4 – Security Domain Life Cycle State LOCKED* shall read as follows:
"Locking a Security Domain prevents this Security Domain from being associated with new Executable Load Files or Applications and **prohibits access** (not: "does not have any required effect") to that Security Domain's services by Applications through the OPEN."

E.2.2 Key Information using GET DATA command

The text in *section 9.3.1 – GET DATA Definition and Scope* is misleading since the command does not retrieve a key value but key information (as described in *section 9.3.1 – GET DATA Response Message*).

The 2nd sentence of *section 9.3.1 – GET DATA Definition and Scope* shall read as follows:
" The data object may contain **information pertaining to** a key."

E.2.3 Maximum size of APDU commands

The maximum length of the LOAD, PUT KEY, and STORE DATA command messages is incorrectly set and shall be **255 bytes** (not 256 bytes).

In s*ection 9.6.2 - LOAD Command Message*, the sentence following the table describing the LOAD command message structure shall read as follows:
"The overall length of the command message shall not exceed **255 bytes**".

In s*ection 9.7.2 – PUT KEY Command Message*, the sentence following the table describing the PUT KEY command message structure shall read as follows:
"The overall length of the command message shall not exceed **255 bytes**".

In s*ection 9.10.2 – STORE DATA Command Message*, the sentence following the table describing the STORE DATA command message structure shall read as follows:
"The overall length of the command message shall not exceed **255 bytes**".

E.2.4 SET STATUS command parameter P2

The text in *section 9.9.2.2 – SET STATUS Reference Control Parameter P2* shows some contradiction between its 2nd paragraph and the following paragraphs.

State transition requests for the Issuer Security Domain shall be coded according to *table 9-6 – Issuer Security Domain Life Cycle Coding*. State transition requests for a Security Domain shall be coded according to *table 9-5 – Security Domain Life Cycle Coding*. State transition requests for an Application shall be coded according *to table 9-4 – Application Life Cycle Coding* and follow the rule described in the last paragraph of *section 9.9.2.2 – SET STATUS Reference Control Parameter P2* for Application Life Cycle State transitions to and from the BLOCKED state.

The 2nd paragraph of *section 9.9.2.2 – SET STATUS Reference Control Parameter P2* shall **be deleted.**

E.2.5 APDU Data Field Encryption and Decryption

In *appendices D.3.4 – SCP '01' APDU Data Field Encryption and Decryption* and *E.4.6 – SCP '02' APDU Data Field Encryption and Decryption*, the encryption method is incorrectly referenced and shall be ***appendix B.1.1.1 - Encryption/Decryption CBC Mode*** (not *B.1.2 - MACing*).

The last sentence of the 2nd paragraph of *appendix D.3.4 – SCP '01' APDU Data Field Encryption and Decryption* shall read as follows:
"Command message encryption and decryption uses the Secure Channel Encryption (S-ENC) session key and the full triple DES encryption method described in **Appendix B.1.1.1 *Encryption/Decryption CBC Mode***".

The 4th paragraph of *appendix E.4.6 – SCP '02' APDU Data Field Encryption and Decryption* shall read as follows:
"Command message encryption and decryption uses the Secure Channel Encryption Session Key and the encryption method described in **Appendix B.1.1.1 *Encryption/Decryption CBC Mode***".

E.2.6 Le for EXTERNAL AUTHENTICATE command

In *appendices D.4.2.2 – EXTERNAL AUTHENTICATE Command Message* and *E.5.2.2 – EXTERNAL AUTHENTICATE Command Message*, the parameter Le shall be **absent** (not set to '00') since no response data is expected per *appendices D.4.2.6 – Data Field Returned in the EXTERNAL AUTHENTICATE Response Message* and *E.5.2.6 – Data Field Returned in the EXTERNAL AUTHENTICATE Response Message*.

Tables D-7 - EXTERNAL AUTHENTICATE Command Message and *E-9 – SCP02 EXTERNAL AUTHENTICATE Command Message* shall read as follows:

Code	Value	Meaning
CLA	'84'	GlobalPlatform command with secure messaging
INS	'82'	EXTERNAL AUTHENTICATE
P1	'xx'	Security Level
P2	'00'	Reference control parameter P2
Lc	'10'	Length of data
Data	'xxxxx…'	Host cryptogram and MAC
Le		**Not present**

E.2.7 SCP '02' Sensitive Data Encryption and Decryption

In *appendix E.4.7 – SCP '02' Sensitive Data Encryption and Decryption*, the encryption method is incorrectly referenced and shall be ***appendix B.1.1.1 - Encryption/Decryption CBC Mode* for Implicit Initiation of the Secure Channel and *appendix B.1.1.2 - Encryption/Decryption ECB Mode* for Explicit Initiation of the Secure Channel** (not *B.1.2 - MACing*).

The 2nd paragraph of *appendix E.4.7 – SCP '02' Sensitive Data Encryption and Decryption* shall read as follows:
"The data encryption process uses the Data Encryption Session Key and the encryption method described in **Appendix B.1.1.2 *Encryption/Decryption ECB Mode* when using Explicit Initiation of the Secure Channel or Appendix B.1.1.1 *Encryption/Decryption CBC Mode* when using Implicit Initiation of the Secure Channel**".

E.3 Errata List 0.3

E.3.1 PUT KEY command status words coding

The value of the error code 'Memory failure' described in *table 9-46 – PUT KEY Error Conditions* of *section 9.7.3.2 – Processing State Returned in the PUT KEY Response Message* is incorrect and shall be **'6581'** (not '6A81').

The 1st row of *table 9-46 – PUT KEY Error Conditions* shall read as follows:

SW1	SW2	Meaning
'65'	'81'	Memory failure

E.3.2 GlobalPlatform API: processSecurity() method exception coding

The reason code 'ISO7816.SW_WRONG_INS' described in the processSecurity() method of the org.globalplatform.SecureChannel interface in *Appendix A.2 – GlobalPlatform on a Java Card* is incorrectly labeled and shall read **'ins byte'** (not 'class byte').

The 2nd reason code of the javacard.framework.ISOException of the processSecurity() method (see org.globalplatform.SecureChannel interface in *Appendix A.2 – GlobalPlatform on a Java Card*) shall read as follows:

- "ISO7816.SW_WRONG_INS **ins** byte is not recognized by the method."

P. PRECISIONS LISTS

P.1 Precisions List 0.1

P.1.1 GlobalPlatform Object Identifier (OID) Value

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

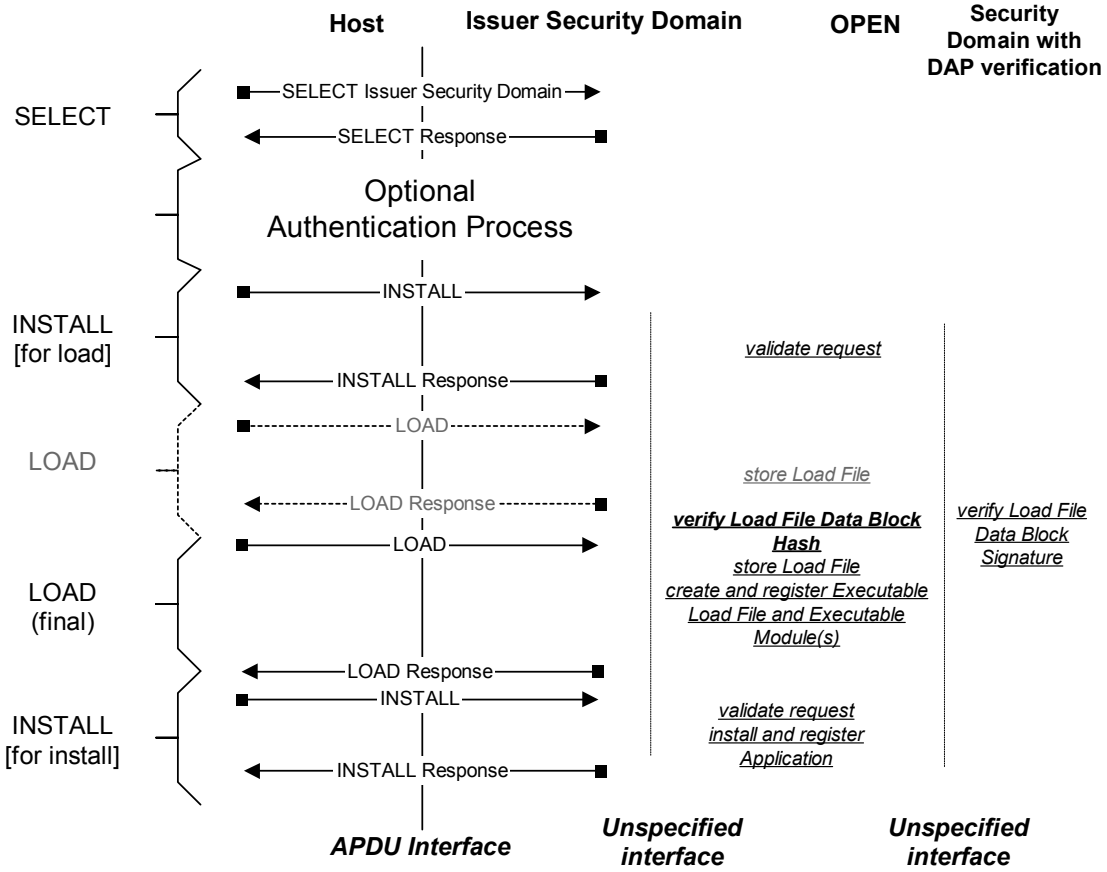
P.1.2 GlobalPlatform Registration Identifier (RID) Value

In *Appendix F1- Data Values*, the Registration Identifier (RID) assigned to GlobalPlatform is as follows:
GlobalPlatform RID = 'A000000151'

P.1.3 Figure 6-3 Load and Installation Flow Diagram

In *section 6.4.1 - Card Content Loading and Installation*, the list of actions performed by the Issuer Security Domain shall include the **verification of the Load File Data Block Hash**.

Figure 6-3 - Load and Installation Flow Diagram shall read as follows:



P.1.4 CVM Format conversion

In *Appendix A.2 - GlobalPlatform on a Java Card*, 3 formats for the CVM are defined: ASCII, BCD, and HEX.

A new *section 6.9.x – CVM Format* needs to be added describing the **rules** for implementing CVM Format conversion as follows:
"The 3 following formats are defined for the CVM value:

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

The following rules apply for the CVM Format conversion:

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

The internal format for storing the CVM value by the CVM Handler is implementation dependent and shall be transparent to Applications. It shall not preclude any format used by Applications requesting CVM services."

In *Appendix A.2 - GlobalPlatform on a Java Card*, the processing rules of the update() method of the org.globalplatform.CVM interface shall be completed with the following rule:

- "Data presented always replaces the previous data regardless of its format or length. The CVM Handler shall remember the format, length, and value of the CVM data. The CVM Handler may (or may not) do editing checks on the data and reject the CVM update if the data fails the editing checks (e.g. reject data that is presented as BCD that is not numerical)."

In *Appendix A.2 - GlobalPlatform on a Java Card*, the processing rules of the verify() method of the org.globalplatform.CVM interface shall be completed with the following rules:

- "If HEX format is presented for CVM verification and ASCII or BCD format was used for updating the CVM value, the comparison fails.
- If HEX format is presented for CVM verification and HEX format was used for updating the CVM value, the comparison succeeds when the length and the data value match exactly.
- If BCD or ASCII format is presented for CVM verification and HEX format was used for updating the CVM value, the comparison fails.
- If ASCII format is presented for CVM verification and BCD format was used for updating the CVM value, the comparison fails if the ASCII characters presented for verification are not all numerical (zero to nine). If all the ASCII characters are numerical, format conversion occurs and the comparison succeeds when the length and the data value match exactly.
- If BCD format is presented for CVM verification and ASCII format was used for updating the CVM value, the comparison fails if the CVM value contains non-numerical ASCII characters. If the CVM value contains only numerical ASCII characters, format conversion occurs and the comparison succeeds when the length and the data value match exactly."

P.1.5 Atomicity for GlobalPlatform API on Java Card

In the *sub-section Atomicity* of *Appendix A.2 - GlobalPlatform on a Java Card*, atomicity shall **not** apply to velocity checking operations performed by methods of the GlobalPlatform API, such as the verify() method of the org.globalplatform.CVM interface.

The last paragraph of this *sub-section Atomicity* shall be completed as follows:
"Objects used to enforce the implementation of velocity checking shall **not** conform to a transaction in progress."

The **verify()** method of the org.globalplatform.CVM interface shall be completed with the following rule:
"The Retry Counter object, the CVM states VALIDATED and INVALID_SUBMISSION shall **not** conform to a transaction in progress, i.e. it shall not revert to a previous value if a transaction in progress is aborted."

P.1.6 GlobalPlatform API: encryptData() method exceptions

In *Appendix A2 – GlobalPlatform on a Java Card*, the encryptData() method of the interface org.globalplatform.SecureChannel shall throw an **ISO exception** in case of invalid length of the clear text data to encrypt.

"public short encryptData(byte[] baBuffer,
short sOffset,
short sLength)
throws java.lang.ArrayIndexOutOfBoundsException
javacard.framework.ISOException"

P.1.7 GlobalPlatform API: getSecurityLevel() method response

According to *Appendix A2 – GlobalPlatform on a Java Card*, the getSecurityLevel() method of the interface org.globalplatform.SecureChannel returns either the value NO_SECURITY_LEVEL or a bit-map combination of the following values: AUTHENTICATION, C_MAC, R_MAC, C_DECRYPTION, and R_ENCRYPTION (see *section 8.2.3 – Secure Channel Termination* for further information on actions and events closing a Secure Channel Session and resetSecurity() method).

In *Appendix D - Secure Channel Protocol '01'*, a new *section D.1.x – Security Level* needs to be added describing the Security Level as follows:
"The Security Level of a communication not included in a Secure Channel Session shall be set to NO_SECURITY_LEVEL.
For Secure Channel Protocol '01', the Security Level established in a Secure Channel Session is a bitmap 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 and C_MAC after a successful processing of an EXTERNAL AUTHENTICATE command with P1 indicating C-MAC (P1='01'): AUTHENTICATION and C-MAC shall be cleared once the Secure Channel Session is closed.

- AUTHENTICATION, C_MAC, and C_DECRYPTION after a successful processing of an EXTERNAL AUTHENTICATE command with P1 indicating C-MAC and Command Encryption (P1= '03'): AUTHENTICATION, C_MAC, and C_DECRYPTION shall be cleared once the Secure Channel Session is closed.

In *Appendix E - Secure Channel Protocol '02'*, a new *section E.1.x – Security Level* needs to be added describing the Security Level as follows:
"The Security Level of a communication not included in a Secure Channel Session shall be set to NO_SECURITY_LEVEL.
For Secure Channel Protocol '02' with Explicit Initiation mode, the Security Level established in a Secure Channel Session is a bitmap 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 and C_MAC and simultaneously set and cleared.
- R_MAC after a successful processing of an EXTERNAL AUTHENTICATE command with P1 indicating R-MAC (P1='1x'): R-MAC shall be cleared once the Secure Channel Session is 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.

For Secure Channel Protocol '02' with Implicit Initiation mode, the Security Level established in a Secure Channel Session is a bitmap 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 and 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 and 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."

P.1.8 Sequence Counter in Secure Channel Protocol '02'

In Secure Channel Protocol '02' with Implicit Initiation mode, the value of the Sequence Counter used for deriving the Data Encryption Session Key and the R-MAC Session Key is the new incremented value resulting of the Secure Channel Session initiation.

In *Appendix E - Secure Channel Protocol '02' section E.1.2.2 - Implicit Secure Channel Initiation*, the 4th paragraph shall read as follows:
"The card, on receipt of this first C-MAC, creates **the C-MAC** Session Key using its internal card static key(s) and Secure Channel Sequence Counter."
And add a new and last paragraph as follows:
"For sensitive data decryption, the card creates the Data Encryption Session Key using its internal card static key(s) and the newly incremented value of the Secure Channel Sequence Counter. For R-MAC generation, the card creates the R-MAC Session Key using its internal card static key(s) and the newly incremented value of the Secure Channel Sequence Counter."

P.1.9 Coding of P2 for BEGIN R-MAC SESSION command

In *Appendix E - Secure Channel Protocol '02' section E.5.3.2 – BEGIN R-MAC SESSION Command Message,* the value of P2 shall be set to '**01**' (not 'xx').

Table E-12 - BEGIN R-MAC SESSION Command Message shall read as follows:

Code	Value	Meaning
CLA	'80' or '84'	Open Platform command
INS	'7A'	BEGIN R-MAC SESSION
P1	'xx'	Reference Control Parameter P1
P2	'**01**'	Reference Control Parameter P2
Lc	'xx'	Length of data
Data	'xx xx…'	BEGIN R-MAC SESSION data and C-MAC, if needed
Le	'00'	

A new *sub-section E.5.3.x - Reference Control Parameter P2* needs to be added describing P2 as follows:
"This parameter defines the beginning of the session for APDU response message integrity.

b8	b7	b6	b5	b4	b3	b2	b1	Description
0	0	0	0	0	0	0	1	Begin R-MAC session.

Table E-xx: BEGIN R-MAC SESSION Parameter P2"

P.1.10 Coding of P1 and P2 for END R-MAC SESSION command

In *Appendix E - Secure Channel Protocol '02' section E.5.4.2 – END R-MAC SESSION Command Message,* the value of P1 shall be set to '**00**' (not 'xx') and the value of P2 shall be set to '**03**'(not 'xx').

Table E-16 - END R-MAC SESSION Command Message shall read as follows:

Code	Value	Meaning
CLA	'80' or '84'	Open Platform command
INS	'78'	END R-MAC SESSION
P1	'**00**'	Reference Control Parameter P1
P2	'**03**'	Reference Control Parameter P2
Lc	'xx'	Length of data, if any
Data	'xx xx…'	C-MAC, if needed
Le	'00'	Always return the R-MAC

A new *sub-section E.5.4.x - Reference Control Parameter P1* needs to be added describing P1 as follows:
"This parameter defines the level of security for all subsequent APDU response messages following this END R-MAC SESSION command (it does not apply to this command).

b8	b7	b6	b5	b4	b3	b2	b1	Description
0	0	0	0	0	0	0	0	No secure messaging expected

Table E-xx: END R-MAC SESSION Parameter P1"

A new *sub-section E.5.4.x - Reference Control Parameter P2* needs to be added describing P2 as follows:
"This parameter defines the end of the session for APDU response message integrity.

b8	b7	b6	b5	b4	b3	b2	b1	Description
0	0	0	0	0	0	1	1	End R-MAC session

Table E-xx: END R-MAC SESSION Parameter P2"

P.2 Precisions List 0.2

P.2.1 Application Life Cycle Specific State Transitions

In *section 5.3.1.5 – Application Life Cycle Specific States*, add the following sentence:
"**The OPEN does not enforce any control on Application Life Cycle Specific State transitions**".

The requirements for the deprecated setCardContentState() method of the openplatform.OPSystem class (see *appendix A.1 – Deprecated GlobalPlatform Java Card API*) shall be in concordance with the requirements defined for the setCardContentState() method of the org.globalplatform.GPSystem class (see *appendix A.2 – GlobalPlatform on a Java Card*). In particular, the Card Manager (i.e. OPEN) is **not responsible** for enforcing any Application Life Cycle Specific State transition rules.

P.2.2 SET STATUS command: Life Cycle State transitions

According to *section 9.9.2.2 – SET STATUS Reference Control Parameter P2*, the Life Cycle State transition requests shall abide by the transitioning rules defined in *section 5 – Life Cycle Models*. A Life Cycle State transition request for the Issuer Security Domain, a Security Domain, or an Application to an identical state shall **be rejected** by the card with an error code such as 'Incorrect P1 P2' (status words set to '6A86').

P.2.3 GlobalPlatform API invocation from within install() method

The required behavior of the card in case an Application invokes (incorrectly) any method of the org.globalplatform.GPSystem class other than getCardState() and getCVM() methods (see *Appendix A.2 – GlobalPlatform on a Java Card*) is undefined. For example, an exception may be thrown and the processing of the install() method may be aborted.

P.2.4 GlobalPlatform API: setATRHistBytes() method errors

In case of an Application without the Default Selected privilege, the (incorrect) invocation of the setATRHistBytes()method of the org.globalplatform.GPSystem class (see *Appendix A.2 – GlobalPlatform on a Java Card*) shall return: '**False**', and not throw an exception.

P.2.5 GlobalPlatform API: decryptData() & encryptData() method errors

In case a Secure Channel Session has not been opened prior to being called by an Application, the decryptData() and encryptData() methods of the interface org.globalplatform.SecureChannel (see *Appendix A.2 – GlobalPlatform on a Java Card*) shall throw an ISO exception with the reason code: '**Security Status Not Satisfied**', which sets the status words to '**6982**'.

P.2.6 GlobalPlatform API: unwrap() method errors

In case of an invalid secure messaging verification (e.g. erroneous C-MAC), the unwrap() method of the interface org.globalplatform.SecureChannel (see *Appendix A.2 – GlobalPlatform on a Java Card*) shall throw an ISO exception with the reason code: '**Security Status Not Satisfied**', which sets the status words to '**6982**'.

P.2.7 Card Recognition Data Object Identifier value

In *Appendix F1- Data Values*, the Object Identifier (OID) value assigned by ISO to GlobalPlatform being: **{1 2 840 114283}** (see *Precision P.1.1*), its hexadecimal representation is: '**2A864886FC6B**'.

In *Appendix F1- Data Values*, the Object Identifier (OID) value for Card Recognition Data, i.e. {globalplatform 1}, which also identifies GlobalPlatform as the Tag Allocation Authority for Card Recognition Data data objects, is the following: **{1 2 840 114283 1}**, that is in hexadecimal: '**2A864886FC6B01**'.

P.2.8 AID value for Java Card Export File for GlobalPlatform API

In *Appendix F1- Data Values*, the AID value for the Java Card Export File for GlobalPlatform Card Specification 2.1 API (based on the RID assigned by ISO to GlobalPlatform, i.e. 'A000000151', see *Precision P.1.2*) is the following: **'A00000015100'**.

P.3 Precisions List 0.3

P.3.1 Security Domain Life Cycle State LOCKED

If a Security Domain in the Life Cycle State LOCKED is requested by the OPEN to perform DAP Verification, the Security Domain is unavailable to process the request and the verification of the Load File Data Block Signature fails.

The 4th paragraph of *section 5.3.2.4 – Security Domain Life Cycle State LOCKED* shall be completed as follows:
"In this state, when requested by the OPEN to perform DAP Verification, the Security Domain shall indicate that **verification of the Load File Data Block Signature failed**."

And the 6th paragraph of *section 7.5 – DAP Verification* shall be completed as follows:
"In the Life Cycle State LOCKED, the Security Domain shall always inform the OPEN that the signature is invalid."

For additional information on this Security Domain Life Cycle State, see also *Errata E.2.1*.

P.3.2 Security Domain Deletion

The runtime behavior described in *section 6.4.2.1 – Application Removal* requires in its 2nd bullet that an Application be not deleted if any other Application present on the card makes a reference to it. Since Security Domains are considered as being Applications of a special type (see *section 7.1 – Overview of Security Domains*), a Security Domain to which at least one Application still makes reference to (is associated with) **may not be deleted**. In such case, the deletion request must be rejected by the OPEN.

Add a new bullet to the Runtime Behavior description in s*ection 6.4.2.1 – Application Removal*:

- "In case of a Security Domain deletion, determine if any Applications present in the card are associated with this Security Domain,"

For additional information on Security Domain deletion policy, see *FAQ List 0.3 – July 2002*.

P.3.3 Resetting the CVM State

According to *section 6.9.1.1 – CVM State ACTIVE*, an Application with the CVM Management privilege may reset the CVM State to ACTIVE from the State BLOCKED (see the resetAndUnblockState() method of the org.globalplatform.CVM interface in *Appendix A.2 – GlobalPlatform on a Java Card*) or when changing the CVM value or CVM Try Limit (see also *Precision P.3.8)*. Any Application with or without the CVM Management privilege may reset the CVM State to ACTIVE from either the State VALIDATED or INVALID_SUBMISSION (see the resetState() method of the org.globalplatform.CVM interface in *Appendix A.2 – GlobalPlatform on a Java Card*)..

The 3rd bullet of s*ection 6.9.1.2 – CVM State INVALID_SUBMISSION* shall read as follows:

- "**An Application resets the CVM State**,
- A privileged Application either blocks the CVM or changes the CVM value **or CVM Try Limit**"

The 3rd bullet of s*ection 6.9.1.3 – CVM State VALIDATED* shall be completed as follows:

- "**An Application resets the CVM State**,
- A privileged Application either blocks the CVM or changes the CVM value **or CVM Try Limit**"

The 4th sentence of s*ection 6.9.1.4 – CVM State BLOCKED* shall read as follows:
"The CVM state may only transition from the BLOCKED state back to the ACTIVE state on instruction from a privileged Application, **which either resets (unblocks) the CVM State or changes the CVM value or CVM Try Limit**."

P.3.4 Load File AID in the INSTALL [for load] command

The Load File Data Block may contain information on the Load File attributes, e.g. its name, version number, size, etc. For instance, for a Java Card based smart card, since the CAP File contains its own AID, the Load File AID given in the INSTALL [for load] command must be consistent with the CAP File AID present in the Load File Data Block.

Add to the 6th paragraph of s*ection 6.4.1 – Card Content Loading and Installation* the following sentence:
"The Load File Data Block may also contain information on the Load File Data Block attributes such as its name, version number, and size."

And add to *section 9.5.2.3.1 – Data Field for INSTALL [for load]* the following sentence:
"The Load File AID and Load Parameters given in the INSTALL [for load] command **shall be consistent with the information contained in the Load File Data Block (if any)**."

P.3.5 Load File Data Block Hash in the INSTALL [for load] command

In cases other than Delegated Management and DAP Verification, the card may also request the OPEN to verify a Load File Data Block Hash present in the INSTALL [for load] command. S*ection 9.5.2.3.1 – Data Field for INSTALL [for load]* shall be completed as follows:
"In all other cases, the Load File Data Block Hash is **optional and may be verified** by the card."

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 (see *FAQ List 0.3 – July 2002*).

P.3.6 Key Replacement with the same key identification attributes

According to *section 6.8.4 – On-Card Key Information*, "Key Identifiers and Key Version Numbers may have arbitrary values for the card and these values may vary from one key management scheme to the next". A specific key management scheme may require to update/replace a key value without modifying its key identification attributes.

To reflect such possibility, the two first bullets of *section 9.7.1 – PUT KEY Command Definition and Scope* shall be completed as follows:

- "Replace an existing key with a new key: The new key has **the same or** a different Key Version Number and the same Key Identifier as the key being replaced;
- Replace multiple existing keys with new keys: The new keys have **the same or** a different Key Version Number (identical for all new keys) and the same Key Identifiers as the keys being replaced."

Note this possibility was present in version 2.0.1' of GlobalPlatform Card Specification. A configuration policy for accepting (or rejecting) such a possibility must be defined and enforced on-card for each Security Domain, including the Issuer Security Domain (see *FAQ List 0.3 – July 2002*).

P.3.7 SELECT Command with no data

When selecting the Issuer Security Domain using the SELECT command with no data, the SELECT command is processed as a case 2 command. Last sentence of *section 9.8.2.3 – Data Field in the SELECT Command Message* shall be completed as follows:
"The **Lc and** data field of the SELECT command may be omitted if the Issuer Security Domain is being selected. In this case, Le shall be set to '00' and the command is a case 2 command according to ISO 7816-4."

P.3.8 GlobalPlatform API: CVM.update() and CVM.setTryLimit methods

According to *Appendix A.2 – GlobalPlatform on a Java Card*, the update() method of the org.globalplatform.CVM interface resets the CVM Try Counter when it successfully changes/initializes the CVM value. Similarly the setTryLimit() method resets the CVM Try Counter when it successfully changes/initializes the CVM Try Limit. If the current CVM State was BLOCKED, a successful processing of these methods also unblocks the CVM.

Add a new bullet to the processing notes of the update()method of the org.globalplatform.CVM interface:

- "The **CVM State is reset to ACTIVE** when changing the CVM value".

Add a new bullet to the processing notes of the setTryLimit()method of the org.globalplatform.CVM interface:

- "The **CVM State is reset to ACTIVE** when changing the CVM Try Limit".

P.3.9 Length of Response Data in R-MAC

In *Figure E-5: SCP02 – R-MAC Generation*, an arrow is missing from the 'Li: length of R-Data' box to the 'R-MAC generation' box (see *Appendix E.4.5 – APDU Response R-MAC Generation and Verification*).

When the length of the response data is 256 bytes, the value of the Li byte is '00'. The 2nd bullet of the 3rd paragraph of *Appendix E.4.5 – APDU Response R-MAC Generation and Verification* shall be completed as follows:
"The R-MAC is computed on the following data block:

- The stripped APDU command message, i.e. without any C-MAC and modified command header,
- The response data preceded with a byte that codes its length **modulo 256**,
- The status words."

P.3.10 Default AID value for Issuer Security Domain

In *Appendix F1- Data Values*, the default AID value for the Issuer Security Domain (based on the RID assigned by ISO to GlobalPlatform, i.e. 'A000000151', see *Precision P.1.2*) is the following: **'A000001510000'**.
Note it complies with the Java Card rules for on-card instantiation of packages and is consistent with the AID assigned to the Java Card Export File for GlobalPlatform Card Specification 2.1 API (i.e. 'A00000015100', see *Precision P.2.8*).