Advanced Card Systems Ltd.
Card & Reader Technologies

ACOSJ-G

(Contact/Contactless/Combi)

Functional Specifications V2.09

info@acs.com.hk
www.acs.com.hk

Table of Contents

List of Figures

List of Tables

1.0. Overview

The ACOSJ is a smart card operating system developed by Advanced Card Systems Ltd. It works based on the JAVA Card Virtual Machine and complies with GlobalPlatform Card Specification Version 2.2.1, JAVA Card Specification Version 3.0.4 and Mapping Guidelines 1.0.1 on its functions and configurations.

The ACOSJ is available in three interface options:

- *Combi: Please refer to sessions where both Contact and Contactless interfaces apply (**2.3**/ **2.7**/ **2.8**)*
- *Contact Only: Please refer to sessions where the Contact interface applies (**2.3.1**/ **2.7**)*
- *Contactless Only: Please refer to sessions where the Contactless interface applies (**2.3.2**/ **2.8)***

The purpose of this document is to describe in detail the features and functions of the ACOSJ smart card operating system.

1.1. ACOSJ-G Versions

Version	Date Released	Modifications
ACOSJ v2.04	April 2019	• 95KB EEPROM • Operating voltage: 2.1 V to 5.5 V • Changes in the configuration memory address • Features enhancement
ACOSJ v2.05	June 2021	• 95KB EEPROM • Operating voltage: 2.1 V to 5.5 V • Added ALG_RSA_SHA_1/224/256/512_PKCS1 • Features enhancement
ACOSJ v2.07	March 2023	• 95KB EEPROM • Operating voltage: 2.1 V to 5.5 V • Improvement RSA • Features enhancement
ACOSJ v2.08	June 2023	• 95KB EEPROM • Operating voltage: 2.1 V to 5.5 V • Expand APDU Buffer 1480 • Features enhancement

Table 1: History of Modifications

1.2. Symbols and Abbreviations

Abbreviation	Description
AES	Advanced Encryption Standard
AID	Application Identifier
APDU	Application Protocol Data Unit
API	Application Programming Interface

Abbreviation	Description
ASCII	American Standard Code for Information Interchange
ATR	Answer-to-Reset
ATQ	Answer-to-Request (for contactless cards)
BCD	Binary Coded Decimal
BER	Basic Encoding Rules
CAT	Card Application Toolkit; or Cryptographic Authorization Template
CBC	Cipher Block Chaining
CCT	Control Reference Template for Cryptographic Checksum
CIN	Card Image Number/Card Identification Number
CLA	Class byte of the command message
CRT	Control Reference Template
CT	Control Reference Template for Confidentiality
CVM	Cardholder Verification Method
DAP	Data Authentication Pattern
DEK	Data Encryption Key
DER	Distinguished Encoding Rules
DES	Data Encryption Standard
DST	Control Reference Template for Digital Signature
ECB	Electronic Code Book
EMV	Europay, Mastercard, and VISA; used to refer to the ICC Specifications for Payment Systems
ENC	Encryption
FCI	File Control Information
HEX	Hexadecimal
HMAC	Keyed-Hash Message Authentication Code
ICC	Integrated Circuit Card
ICV	Initial Chaining Vector
IIN	Issuer Identification Number
INS	Instruction byte of the command message
ISO	International Organization for Standardization
Lc	Exact length of data in a case 3 or case 4 command
Le	Maximum length of data expected in response to a case 2 or case 4 command
LV	Length Value
MAC	Message Authentication Code
MEL	MULTOS Executable Language. The instruction set of the MULTOS™ runtime environment
OID	Object Identifier

Abbreviation	Description
P1	Reference control parameter 1
P2	Reference control parameter 2
PIN	Personal Identification Number
PKI	Public Key Infrastructure
RAM	Random Access Memory
RFU	Reserved for Future Use
RID	Registered Application Provider Identifier
ROM	Read-only Memory
RSA	Rivest/Shamir/Adleman asymmetric algorithm
SCP	Secure Channel Protocol; or (ETSI) Smart Card Platform
SW	Status Word
SW1	Status Word One
SW2	Status Word Two
TLV	Tag Length Value
TP	Trust Point
'xx'	Hexadecimal values are expressed as hexadecimal digits between single quotation marks
'X'	A value in a cell of a table whose purpose is described in the 'Meaning' column of the table
'-'	A value (0 or 1) in a cell of a table that does not affect the 'Meaning' given for that row of the table

Table 2: Symbols and Abbreviations

2.0. Card Specifications

This section summarizes the features and functionalities of the ACOSJ-G.

2.1. Electrical Specifications

- Operating Voltage (ACOSJ-G v2.04, v2.05, v2.07, v2.08):
  - 2.1 V to 5.5 V supply voltages
- Maximum External Clock Frequency: 10 MHz
- Maximum CPU Clock Frequency: 28 MHz
- ESD protection greater than 5 kV (HBM)

2.2. Environmental Specifications

- Operating Temperature: -25°C to +85°C

2.3. Communication Protocols

2.3.1. Contact interface

- T=0, T=1 with baud up to 625 kbps (external clock frequency 5MHz)

2.3.2. Contactless Interface

- T=CL protocol with baud up to 848 kbps

2.4. Memory

- Capacity: 95 KB (ACOSJ-G v2.04, v2.05, v2.07, v2.08)
- EEPROM Endurance: 500,000 erase/write cycles (25°C)
- Data Retention: 30 years (25°C)

2.5. Cryptographic Functionalities

- DES,2K3DES,3K3DES (ECB and CBC)
- AES: 128/192/256 bits (ECB and CBC)
- RSA: 768 to 2048 bits
- ECC: Modulus 112/128/160/192/224/256/384 bits
- Hash: SHA1, SHA224, SHA256, SHA384, SHA512
- SM2/SM3/SM4
- SEED: 128 bits

2.6. Compliance to Standards

- Compliance with ISO 7816 Parts 1, 2, 3, 4
- Compliance with ISO 14443 (Type A and B)
- Compliance with JAVA Card Specification Version 3.0.4
- Compliance with Global Platform Specification Version 2.2.1
- Compliance with Mapping Guidelines 1.0.1

2.7. Answer-to-Reset (ATR, Contact Card)

After a card reset (e.g., power up) is performed, the card transmits an Answer-to-Reset (ATR) in compliance with ISO 7816 Part 3. ACOSJ supports the contact protocol type T=0 and T=1 with direct or inverse convention.

The following is the default ATR:

Parameter	ATR	Description
TS	3Bh	Direct convention, the least significant bit is sent first
T0	69h	TB1, TC1 and TD1 followed with 9 historical characters
TB1	00h	No programming voltage required
TC1	02h	Extra guard time
9 historical characters (ACOSJvXXX)		

Table 3: Configuration of the Answer-to-Reset

The 9 historical bytes are composed of the following:

Historical Bytes	ATR	Description
T1	41h	Indicates 'A'
T2	43h	Indicates 'C'
T3	4Fh	Indicates 'O'
T4	53h	Indicates 'S'
T5	4Ah	Indicates 'J'
T6	76h	Indicates 'v'
T7~T9	31h 30h 31h or 32h 30h 34h	Indicates '101' Or Indicates '204'

Table 4: Answer-to-Reset Historical Bytes

2.8. Answer to Select (ATS, Contactless Card)

After receiving a Request for Answer to Select (RATS) command from the card reading device, the card transmits an Answer to Select (ATS) in compliance with ISO 14443 Part 4.

The following table shows the default ATS:

Parameter	ATS	Description
TL	0Eh	Length
T0	78h	Format byte …codes Y(1) and FSCI
TA1	00h	Interface byte…codes DS and DR
TB1	71h	Codes FWI and SFGI
TC1	02h	Codes protocol options
T1	41h	Indicates 'A'
T2	43h	Indicates 'C'
T3	4Fh	Indicates 'O'
T4	53h	Indicates 'S'
T5	4Ah	Indicates 'J'
T6	76h	Indicates 'v'
T7~T9	31h 30h 31h or 32h 30h 34h	Indicates '101' Or Indicates '204'

Table 5: Configuration of Answer-to-Select

Note: *For full description of the ATS, kindly refer to ISO 14443 Part 4.*

3.0. Card Life Cycle States

The ACOSJ has five card states: OP_READY, INITIALIZED, SECURED, CARD_LOCKED and TERMINATED. The figure below shows the card life cycle state transition:

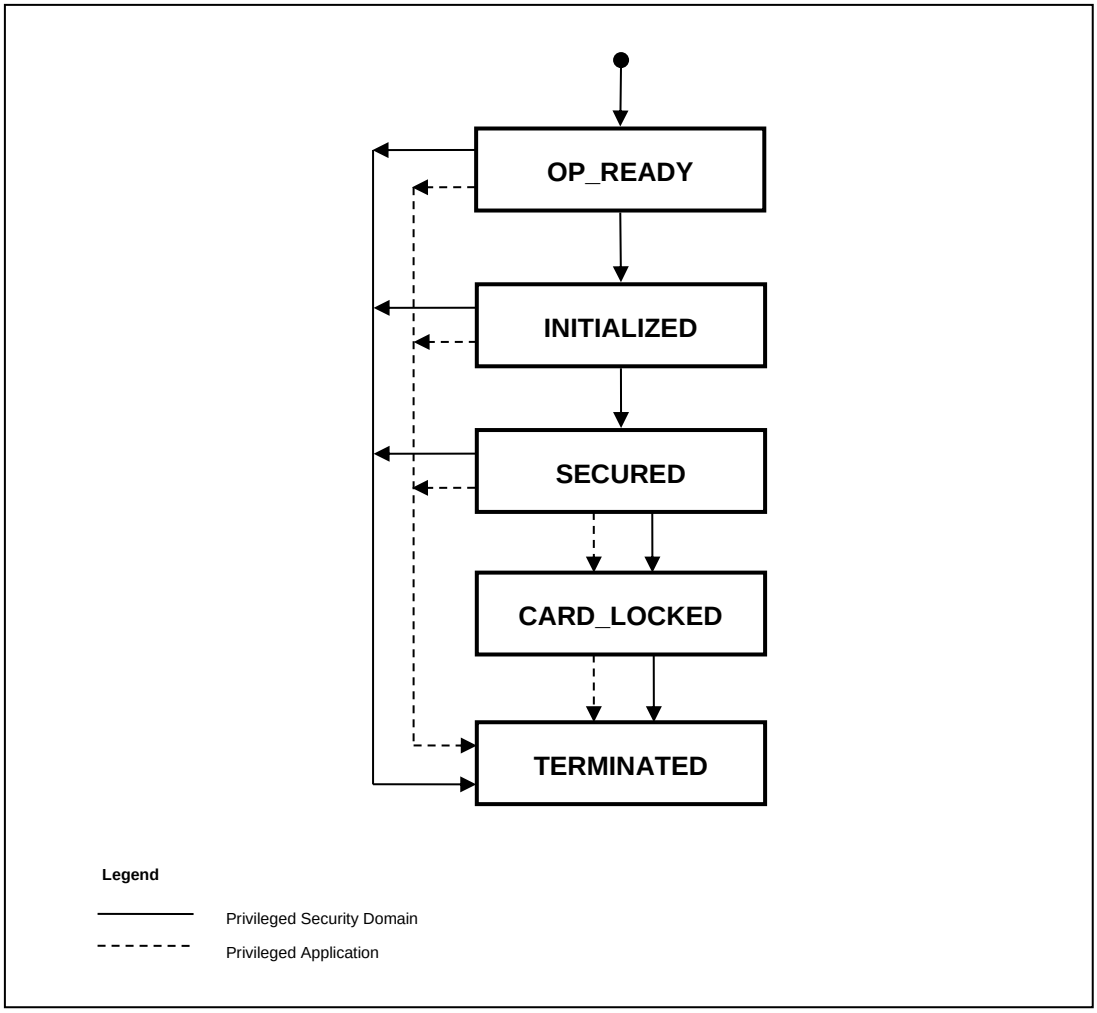

Figure 1: Card Life Cycle

3.1. OP_READY

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

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

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

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

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

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

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

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

3.2. INITIALIZED

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

3.3. SECURED

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

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

3.4. CARD_LOCKED

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

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

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

Either the OPEN or a Security Domain with Card Lock privilege, or an Application with Card Lock privilege, may initiate the transition from the state SECURED to the state CARD_LOCKED.

3.5. TERMINATED

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

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

The OPEN itself, or a Security Domain with Card Terminate privilege, or an Application with Card Terminate privilege, may initiate the transition from any of the previous states to the state TERMINATED.

4.0. Card Architecture

To meet the GlobalPlatform specification for Java card, the ACOSJ card has the architecture for applications as shown in the figure below:

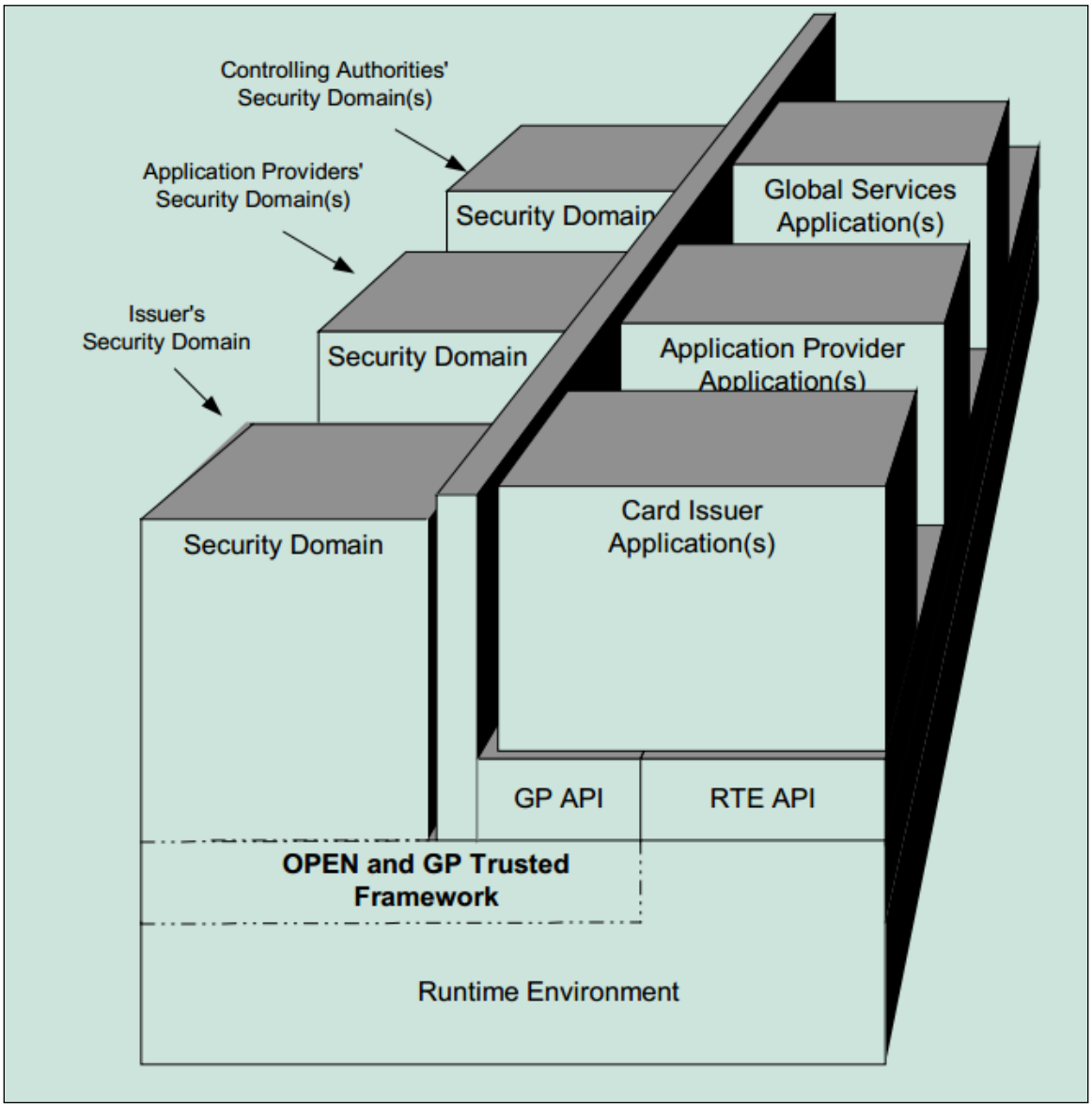

Figure 2: ACOSJ System Architecture

5.0. GP APDU Command Reference

This section details the GlobalPlatform APDU commands that may be implemented. The commands are listed alphabetically.

Table 6 summarizes the APDU commands that are supported by the Issuer Security Domains, and the requirements for support of these APDU commands by other Security Domains. When logical channels are supported, the MANAGE CHANNEL command is only processed by the OPEN and no Security Domain support is required for this command.

Command	OP_READY			INITIALIZED			SECURED			CARD_LOCKED		TERMINATED	
	AM SD	DM SD	SD	AM SD	DM SD	SD	AM SD	DM SD	SD	FASD	SD	FASD	SD
DELETE Executable Load File													
DELETE Executable Load File and related Application(s)													
DELETE Application	✓			✓			✓						
DELETE Key													
GET DATA	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	
GET STATUS	✓			✓			✓			✓			
INSTALL[for load]													
INSTALL[for install]													
INSTALL[for load, install and make selectable]													
INSTALL[for install and make selectable]	✓	✓		✓	✓		✓	✓					
INSTALL[for make selectable]													
INSTALL[for extradition]													
INSTALL[for registry update]													
INSTALL[for personalization]													
LOAD													
PUT KEY	✓			✓			✓						
SELECT	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			
SET STATUS	✓			✓			✓			✓			
STORE DATA	✓			✓			✓						

Table 6: Authorized GlobalPlatform Commands per Card Life Cycle State

AM SD Security Domain with Authorized Management privilege.

DM SD Security Domain with Delegated Management privilege.

FA SD Security Domain with Final Application privilege.

Note: *Command support for an Application with Final Application Privilege which is not a Security Domain is subject to Issuer policy, within the constraints of what is permitted according to the Card Life Cycle State.*

SD Other Security Domain.

✓ Support required.

Blank Cell Support optional.

Striped Cell Support prohibited.

Table 7 summarizes the minimum security requirements for the APDU commands.

Command	Minimum Security
DELETE	Secure Channel initiation or digital signature verification
GET DATA	None
GET STATUS	Secure Channel initiation
INSTALL	Secure Channel initiation or digital signature verification
LOAD	Secure Channel initiation or digital signature verification
MANAGE CHANNEL	Not applicable
PUT KEY	Secure Channel initiation
SELECT	Not applicable
SET STATUS	Secure Channel initiation
STORE DATA	Secure Channel initiation

Table 7: Minimum Security Requirements for Global Platform Commands

5.1. General Coding Rules

Please refer to the ACOSJ Reference Manual.

5.2. DELETE Command

The DELETE command is used to delete a uniquely identifiable object such as an Executable Load File, an Application, an Executable Load File and its related Applications or a key. In order to delete an object, the object should be uniquely identifiable by the selected Application.

5.3. GET DATA Command

The GET DATA command is used to retrieve either a single data object, which may be constructed, or a set of data objects. Reference control parameters P1 and P2 coding are used to define the specific data object tag. The data object may contain information pertaining to a key.

5.4. GET STATUS Command

The GET STATUS command is used to retrieve Issuer Security Domain, Executable Load File, Executable Module, Application or Security Domain Life Cycle status information according to a given match/search criteria.

5.5. INSTALL Command

The INSTALL command is issued to a Security Domain to initiate or perform the various steps required for Card Content management.

5.6. LOAD Command

This section defines the structure of the Load File transmitted in the LOAD command data field for loading a Load File. The ICC internal handling or storage of the Load File is beyond the scope of this Specification.

Multiple LOAD commands may be used to transfer a Load File to the card. The Load File is divided into smaller components for transmission. Each LOAD command shall be numbered starting at 00h. The LOAD command numbering shall be strictly sequential and increments by one. The card shall be informed of the last block of the Load File.

After receiving the last block of the Load File, the card shall execute the internal processes necessary for the Load File and any additional processes identified in the INSTALL [for load] command that preceded the LOAD commands.

5.7. MANAGE CHANNEL Command

The MANAGE CHANNEL command is processed by the OPEN on cards that are aware of logical channels. It is used to open and close Supplementary Logical Channels. The Basic Logical Channel (channel number zero) can never be closed.

5.8. PUT KEY Command

The PUT KEY command is used to either:

- Replace an existing key with a new key: The new key has the same or a different Key Version Number but the same Key Identifier as the key being replaced;
- Replace multiple existing keys with new keys: The new keys have the same or a different Key Version Number (identical for all new keys) but the same Key Identifiers as the keys being replaced;
- Add a single new key: The new key has a different combination Key Identifier/Key Version Number than that of the existing keys;

- Add multiple new keys: The new keys have different combinations of Key Identifiers/Key Version Number (identical to all new keys) than that of the existing keys;

When the key management operation requires multiple PUT KEY commands, chaining of the multiple PUT KEY commands is recommended to ensure integrity of the operation.

In this version of the Specification the public values of asymmetric keys are presented (transmitted) in clear text.

5.9. SELECT Command

The SELECT command is used for selecting an Application. The OPEN only processes SELECT commands indicating the SELECT [by name] option. All options other than SELECT [by name] shall be passed to the currently selected Security Domain or Application on the indicated logical channel.

5.10. SET STATUS Command

The SET STATUS command shall be used to modify the card Life Cycle State or the Application Life Cycle State.

5.11. STORE DATA Command

The STORE DATA command is used to transfer data to an Application or the Security Domain processing the command.

The Security Domain determines if the command is intended for itself or an Application depending on a previously received command. If a preceding command was an INSTALL [for personalize] command, the STORE DATA command is destined for an Application.

Multiple STORE DATA commands are used to send data to the Application or Security Domain by breaking the data into smaller components for transmission. The Security Domain shall be informed of the last block.

A personalization session starts when a Security Domain receives a valid INSTALL [for personalize] command designating an Application (implementing either the Application or the Personalization interface) to which the Security Domain shall forward subsequently received STORE DATA commands.

A personalization session ends when:

- The card is reset;
- The Security Domain is deselected (i.e. another, or the same, applet is selected on the same logical channel);
- The Security Domain is selected on the same or another logical channel;
- The Secure Channel session (if any) established by the Security Domain is reset, possibly by the targeted application;
- The Security Domain receives an INSTALL [for personalize] command (starting a new personalization session for another application);
- The Security Domain receives a STORE DATA command indicating P1.b8=1 (last block);

Any STORE DATA command received out of such a personalization session shall be processed by the Security Domain itself.

6.0. GlobalPlatform API

6.1. GlobalPlatform on a JAVA Card

This section contains only the API required for GlobalPlatform 2.2.x Java Cards. Use of the Open Platform 2.0.1' API is still allowed for supporting older versions of Applications but is deprecated. It is defined in version 2.1.1 of this Specification.

The deprecated API and new API both access the same objects where applicable. While this may seem obvious for methods that have the same name across both classes (e.g. setATRHistBytes(), setCardContentState() and getCardContentState()) it shall also be noted as for example that an application that uses the update() method in the new API to change the value of the global PIN will affect the same global PIN of an application that uses the setPIN() method in the deprecated API to verify the global PIN.

6.1.1. GlobalPlatform Specific Requirements

In order to ensure the highest level of interoperability of GlobalPlatform implementations, GlobalPlatform also adopts the order defined in section 6.2 - *Java Card™ 2.2.x Virtual Machine Specification*.

The following minor modifications to the standard functionality defined in the *Java Card™ 2.1.1 Runtime Environment (JCRE) Specifications* and the *Java Card™ 2.1.1 Application Programming Interface* are present in the implementation of GlobalPlatform. Some of the modifications are already consistent with the *Java Card™ 2.2.x Specification* and therefore not modifications any more.

GPRegistryEntry objects shall be implemented as Shareable Interface Objects as defined in section 6.2.4 *'Shareable Interfaces' of the Java Card™ 2.2.x Runtime Environment (JCRE) Specification* in order to ensure shared access.

6.1.2. GlobalPlatform package AID

Each GlobalPlatform package AID will be a concatenation of a RID and a PIX. The AID value of the Java Card Export File for the new GlobalPlatform API (identical for both GlobalPlatform 2.1 and 2.1.1) based on the RID specified in *Appendix H - GlobalPlatform Data Values and Card Recognition Data* is 'A00000015100'.

6.1.3. Installation

In section 3.1 - *The Method install of the Java Card™ 2.2.x Runtime Environment (JCRE) Specifications*, the parameters passed to the method are defined to be initialization parameters from the contents of the incoming byte array parameter.

This specification expands on this requirement and further defines the content of the Install Parameters. This expansion affects both the implementation of an OPEN and the behavior of a Java Card applet developed for a GlobalPlatform card. It does not affect the definition of the install () method of the Class Applet of the *Java Card™ 2.2.x Application Programming Interface* specification.

The Install Parameters shall identify the following data, present in the INSTALL [for install] command:

- The instance AID;
- The Privileges;
- The Application Specific Parameters[1].

[1] While the APDU command contains Install Parameters representing TLV coded system and Application Specific Parameters, the application only requires knowledge of the Application Specific Parameters, i.e. only LV of the TLV coded structure 'C9' are present as application specific parameters.

The OPEN is responsible for ensuring that the parameters (bArray, bOffset and bLength) contain the following information:

The array, bArray, shall contain the following consecutive LV coded data:

- Length of the instance AID;
- The instance AID;
- Length of the Privileges;
- The Privileges;
- Length of the Application Specific Parameters;
- The Application Specific Parameters.

The byte, bOffset, shall contain an offset within the array pointing to the length of the instance AID.

The byte, bLength, shall contain a length indicating the total length of the above-defined data in the array.

The applet is required to utilize the instance AID as a parameter when invoking the register (byte [] bArray, short bOffset, byte bLength) method of the Class Applet of the *Java Card™ 2.2.x Application Programming Interface* specification.

6.1.4. T=0 Transmission Protocol

GlobalPlatform cards are intended to be functional in the widest range of environments (i.e. Card Acceptance Devices). Currently the *Java Card™ 2.1.1 Runtime Environment (JCRE) Specifications* and *Java Card™ 2.2.x Runtime Environment (JCRE) Specifications* describe the behavior for case 2 commands (when using the T=0 protocol) in contradiction to EMV 2000. GlobalPlatform mandates that the JCRE shall handle this case of command in accordance with ISO/IEC 7816: An applet receiving a case 2 command builds the response and invokes the appropriate API to output the data. If the data is less than the data expected by the terminal, the OPEN will store the data and output a '6Cxx' response code and wait for the CAD to re-issue the command with the correct length. When the re-issued command is received the JCRE will manage the outputting of the stored data.

6.1.5. Atomicity

Unless otherwise specified all internal persistent objects of the GlobalPlatform API must conform to a transaction in progress.

All operations performed by this API, except the Application.processData() method shall be executed atomically. Objects used to enforce the implementation of velocity checking shall not conform to a transaction in progress.

6.1.6. Logical channels

The following logical channel restrictions apply to Java Card™ 2.2.x (see the *Java Card™ 2.2.x Runtime Environment (JCRE) Specifications* for more details):

Selection of an Application on a logical channel as defined in section 6.3 - *Command Dispatch* will be unsuccessful if this same Application, or any other Application instantiated from code in the same package from which the Application being selected was instantiated, is currently selected on another logical channel but the application code does not implement the MultiSelectable interface. Security Domains shall implement the MultiSelectable interface.

Changing context from a Security Domain to an Application as defined in section 7.3.3 - *Personalization Support* will be unsuccessful if this same Application, or any other Application instantiated from code in the same package from which the Application being personalized was instantiated, is currently selected on another logical channel but the application code does not implement the MultiSelectable interface.

An Application that has the privilege of being selected by default and is intended for a card that supports Supplementary Logical Channels should implement the MultiSelectable interface.

GlobalPlatform only defines the assignment of logical channel numbers by the card. Optionally and as defined in Java Card 2.2, a card may also support assignment of logical channel numbers by the terminal.

6.1.7. Cryptographic Algorithms

GlobalPlatform cards supporting RSA cryptography should support key sizes not defined as constants in the Key Builder class. More specifically, support for key sizes being a multiple of 4 bytes (32 bits), and being within the allowed key lengths defined by the implementation should be available.

6.1.8. Level of trust

The Java Card 2.2.x specifications assume that the RID of the AID of packages, applets and instances will be utilized to ensure a level of trust between these entities. In section 4.2.2 - *AID Usage of the Java Card™ 2.2.1 Application Programming Interface* it is defined that the RID of an AID of a component must match the RID of the AID of the package, and in the definition of the register (byte [] bArray, short bOffset, byte bLength) method of the *Java Card™ 2.2.x Application Programming Interface* specification it is defined that an exception must be thrown if the RID portion of the AID bytes in the bArray parameter does not match the RID portion of the Java Card name of the applet.

From a real world implementation point of view, mandating that the RID of the instance AID must be the same as the RID of the component from which it was instantiated is not practical. GlobalPlatform implementations shall not mandate that there be any link through the AID of an instance to its original package (requiring no connections between the AID of an instance and the AID of its original package). It does however assume that all applications in the same package share the same level of trust.

6.1.9. Invocation of GlobalPlatform methods

The Application Programming Interface defined herein is accessible to any Java Card applet developed with the intention of being present on a GlobalPlatform card. One limitation does exist relating to the constructor of the applet and to the install() method of the Class Applet of the *Java Card™ 2.2.x Application Programming Interface*. As this specification does not define exactly when the instance of an applet becomes an entry in the card's GlobalPlatform Registry, an applet developer can only assume that this has occurred following the successful completion of the install () method. To ensure interoperability, GlobalPlatform API methods that require access to the GlobalPlatform Registry entry of the applet invoking the method, shall not be invoked from within the constructor or the install() method.

The following is a list of methods that may be invoked from within the constructor or the install() method:

- getCardState;
- getCVM;
- getService.

The required behavior of the card in the event that an Application incorrectly invokes a method of the org.globalplatform.GPSystem class other than those listed above is undefined. For example, an exception may be thrown and the processing of the install() method may be aborted.

6.1.10. Selection

On GlobalPlatform cards, if an error occurs during the processing of the select() method or if the select() method returns False or if the Application cannot be selected because it does not implement the Multiselectable interface, the OPEN shall continue searching through the GlobalPlatform Registry for a subsequent full or partial match as defined in section 6.4.2.1.2. If no Application is selected, the corresponding logical channel remains open with no currently selected Application. Since no Application is currently selected, any subsequent command, other than a MANAGE CHANNEL or SELECT command, will be rejected. It is expected that the off-card entity will take appropriate action on such an error, e.g. select another Application, close the corresponding logical channel, reset or power off the card.

The Method Summary and Details for the GlobalPlatform Java Card™ API specification is now available in a separate document, which may be found on the GlobalPlatform website.

7.0. Pre-personalization

When the IC is powered for the first time it is in the pre-personalization state.

The INIT_CARD Command (see **8.2.2 INIT_CARD Command**) has to be the first command at the pre-personalization state, the command initializes the card memory area with default values, and then a so-called ACOSJ ROOT Application (see **8.0 ACOSJ ROOT Application**) is available to write critical system data(for example, "ATR", "ATS" and so on) into the EEPROM. Once the data has been loaded and confirmed no further changes, a command that disables the ACOSJ ROOT Application has to be executed.

When the IC exits the pre-personalization state with the successful execution of the Active Card command (see **8.2.5 ACTIVE Command**), it is in the default GlobalPlatform card life cycle OP_READY state if it is not set to a different life cycle state.

It is not possible to return to the pre-personalization state once the active card command has been processed successfully.

***Note**: The Transport Key is the ACOSJ Root Application pre-personalization key, and the Transport Key must be requested from ACS.*

8.0. ACOSJ ROOT Application

8.1. ACOSJ ROOT Application Description

The commands in the ROOT application can be used only after the ROOT application is correctly selected (using the select command and Transport Key). After entering into the ROOT Application, the user can read or configure card parameters through commands supported by the ROOT Application. To exit ROOT Application and select another application, the card must be reset.

Under the ROOT Application, parameters of the card can be read or configured.

The ROOT Application will become invalid once the card is activated.

8.2. ACOSJ ROOT Application Command Reference

8.2.1. SELECT Command

The SELECT command is used to select the ROOT application.

8.2.2. INIT_CARD Command

This command is used to reset the card. This command will initialize all the EEPROM.

8.2.3. READ Command

This command is used to read from the configuration area. Configuration parameters of the card can be read through this command.

8.2.4. WRITE Command

This command is used to write data to the configuration area. Configuration parameters of the card can be set through this command.

8.2.5. ACTIVE Command

This command is used to activate the card. Once this command is implemented successfully, the ROOT Application will become invalid, and the configuration data of the card cannot be read or set directly any more.

9.0. ACOSJ IDENTIFY Application

9.1. ACOSJ IDENTIFY Application Description

After the IDENTIFY Application (AID: 6163732E636F732E61636F736A766572) is selected with a SELECT Command, ACOSJ will return the version number of ACOSJ and indicates whether the card has been activated.

9.2. ACOSJ IDENTIFY Application Command Reference

9.2.1. SELECT Command

The SELECT command is used for selecting the IDENTIFY Application.

Status Bytes

SW1 SW2	Description
90 00h	Correct command
61 XX	SW2 more response bytes available
62 83h	Application permanently blocked
63 00h	Verification failed
63 CX	X retries left
67 00h	Wrong length of data
68 81h	Logical channel not supported
68 82h	Secure messaging not supported
69 84h	Invalid data
69 82h	Security status not satisfied
69 85h	Conditions of use not satisfied
69 88h	Wrong MAC
6A 86h	Incorrect P1 P2
6A 80h	Incorrect values in command data
6A 82h	File not found
6A 81h	Application locked
6A 83h	Record not found
6A 84h	Not enough memory space
6A 88h	Referenced data not found
6D 00h	Invalid INS, Command not existing
6E 00h	Invalid Class

Table 8: Status Codes

References

The documents below served as references for the ACOSJ Functional Specification:

- GlobalPlatform Card Specification Version 2.2.1
- GlobalPlatform Card API Version 1.6
- Java Card 3 API, Classic Edition Version 3.0.4
- Java Card 3 Platform Runtime Environment Specification, Classic Edition Version 3.0.4 September 2011
- Java Card 3 Platform Virtual Machine Specification, Classic Edition Version 3.0.4 September 2011
- GlobalPlatform Card Mapping Guidelines of Existing GP v2.1.1 Implementation on v2.2.1 Version 1.0.1