Advanced Card Systems Limited

Card and Reader Technologies

REFERENCE MANUAL

ACOS5

CONTENTS

1 INTRODUCTION

1.1 Scope

The purpose of this document is to describe in detail
the features and functions of the ACS Smart Card Operating System Version 5.0 (ACOS5)
developed by Advanced Card Systems Ltd.

1.2 Features

- Full 32Kbytes of EEPROM memory for application data
- Compliance with ISO 7816 Parts 1,2,3,4,8,9
- ISO7816-2 compliant 8-contact module
- High baud rate switchable from 9.6 Kbps to 115.2 Kbps
- Supports ISO7816 Part 4 file structures: Transparent, Linear fixed, Linear Variable, Cyclic
- DES / Triple DES / AES / RSA capability
- Mutual Authentication with Session Key generation
- Multilevel secured access hierarchy
- Anti-tearing done on file headers and system information

1.3 Technical Specifications

Electrical

- Operating at 5V DC+/-10 %(Class A), 3V DC +/-10% (Class B) and 1.8V DC +/-10% (Class C)
- Maximum supply current: <20 mA
- ESD protection: ≤ 5 KV

EEPROM

- Capacity: 32Kbytes
- EEPROM endurance: 500K Erase/Write cycles
- Data retention: 10 years

Environmental

- Operating temperature: -25 °C to 85 °C
- Storage temperature: -40°C to 100°C

1.4 Abbreviations

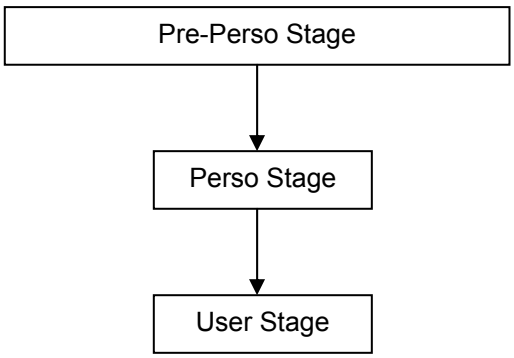
3DES	Triple DES
AMB	Access Mode Byte
AMDO	Access Mode Data Object
APDU	Application Protocol Data Unit
ATR	Answer To Reset
DES	Data Encryption Standard
DF	Dedicated File
EEPROM	Electrically Erasable Programmable Read-Only Memory
EF	Elementary File
EF1	PIN File
EF2	KEY File
ISO	International Organization for Standardization
FCP	File Control Parameters
FDB	File Descriptor Byte
LCSI	Life Cycle Status Integer
LSb	Least Significant Bit
LSB	Least Significant Byte
MAC	Message Authentication Code
MF	Master File
MSb	Most Significant Bit
MSB	Most Significant Byte
FU	Reserved for Future Use
SAC	Security Attribute – Compact
SAE	Security Attribute – Expanded
SCB	Security Condition Byte

SCDO	Security Condition Data Object
SE	Security Environment
SFI	Short File Identifier
SM-MAC	MAC for Secure Messaging
TLV	Tag-Length-Value
UQB	Usage Qualifier Byte

1.5 Anti Tearing

ACOS5 uses an Anti Tearing mechanism in order to protect card from data corruption due to card tearing (i.e., card suddenly pulled out of reader during data update, or reader suffer mechanical failure during card data update). On card reset, ACOS5 looks at the Anti-Tearing fields and does the necessary data recovery. In such case, the COS will return the saved data to its original address in the EEPROM.

1.6 Card Life Stages



ACOS5 has the following card stages:

Pre-Perso Stage – is the initial stage without the Master File. This is the stage where the card is shipped from ACS. In this stage, the ATR TA1 (communication speed) and historical bytes can be personalized by writing directly to the card's physical memory. Users can create a MF with attributes according to their specifications.

Perso Stage – card goes into this stage once the MF is successfully created from the previous stage. User can no longer directly access the card's physical memory. User can create and test files under the MF as if in Operational Mode.

Card will go to User Stage if any file is activated, causing the MF (or any of its descendants) to be undeletable due to DELETE FILE security settings.

User Stage – is equivalent to the card's Operational Mode, this is where all of the card's settings (security, file organizations, etc.) take effect.

2 ANSWER TO RESET

After a hardware reset (e.g. power up), the card transmits an Answer-to-Reset (ATR) in compliance with ISO7816 part 3, and it follows the same format as that of ACOS2. ACOS5 supports the protocol type T=0 in direct convention. The following is the default ATR. For full descriptions of ATR options see ISO 7816 part 3.

Parameter	ATR	Description
TS	3B	Direct Convention.
T0	BE	TA1, TB1, TD1 follows with 14 historical characters.
TA1	18	Capable of high-speed data transfer.
TB1	00	No programming voltage required.
TD1	00	No further interface bytes follow.
T1	41	Historical Data: Indicates ACOS Card
T2	05	Historical Data: Major version
T3	01	Historical Data: Minor version
T4	00	
T5	00	Not used. Compatible with ACOS2
T6	00	
T7	00	
T8	00	
T9	00	
T10	00	
T11	00	
T12	00	
T13	90	Not used. Compatible with ACOS2
T14	00	

2.1 Customizing the ATR

ACOS5's ATR can be customized to change card speed or have specific identification information in the historical string. The new ATR must be compliant to ISO-7816 Part 3, otherwise the card may become unresponsive and non-recoverable at the next power-up or card reset.

The ATR can be customized in Pre-Perso Stage of the card. Use UPDATE BINARY to directly change the contents of the following memory locations:

EEPROM Memory Location	Valid Values	Description
30B4	11h – 9600 bps 18h – 115200 bps	Indicates the card's baud rate in TA1
30B5	00 – 0F	Indicates the length of the customized historical bytes. If set to 0, ACOS5 will use the default historical string
30B6 – 30C4	Any value	Customized historical data

For Example:

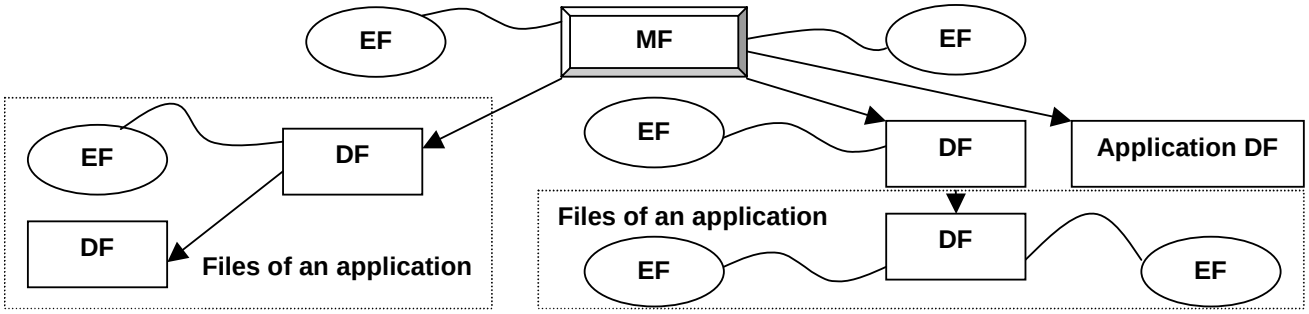
To change the card's ATR to 3B B5 11 00 00 AA BB CC DD EE 90 00:
; Set the speed to 9600, historical length to 5, and historical bytes: AA BB CC DD EE
< 00 D6 03 B4 07 11 05 AA BB CC DD EE
> 9000
; Power up the card, new ATR will take effect

3 FILE SYSTEM

3.1 Hierarchical File System

ACOS5 is fully compliant to ISO 7816 Part 4 file system and structure. The file system is very similar to that of the modern computer operating system. The root of the file is the Master File (of MF). Each Application or group of data files in the card can be contained in a directory called a Dedicated File (DF). Each DF or MF can store data in Elementary Files (EF).

The ACOS5 allows arbitrary depth DF tree structure. That is, the DFs can be nested. Please see the figure below.



3.2 File Header Data

ACOS5 organizes the user EEPROM area by files. Every file has a File Header, which is a block of data that describes the file's properties. Knowledge of the file header block will help the application developer for file creation and accurately plan for the usage of the EEPROM space. The File Header Block consists of the following fields:

File Header	Number of bytes	Applies to
File Descriptor Byte	1	MF / DF / EF
Data Coded Byte	1	MF / DF / EF
File ID	2	MF / DF / EF
File Size	2	EF
Short File Identifier (SFI)	1	MF / DF / EF
Life Cycle Status Integer (LCSI)	1	MF / DF / EF
Security Attribute Compact Length	1	MF / DF / EF
Security Attribute Expanded Length	1	MF / DF / EF
DF Name Length / First Cyclic Record	1	MF / DF / EF
Parent Address	2	MF / DF / EF
Checksum	1	MF / DF / EF
Security Attribute Compact (SAC)	0-8	MF / DF / EF
Security Attribute Expanded (SAE)	0-32	MF / DF
Security Environment File ID	2	MF / DF

FCI File ID	2	MF / DF
DF Name	16 max	MF / DF

Table 1: File Header Block

The subsequent sections will describe each item in the header.

3.2.1 File Descriptor Byte (FDB)

This field indicates the file's type. It can have the following values:

b7	b6	b5	b4	b3	b2	b1	b0	Hex	File type
0	0	1	1	1	1	1	1	3F	**MF**
0	0	1	1	1	0	0	0	38	**DF**
0	0	0	0	0	-	-	-	-	**Working EF**
0	0	0	0	0	0	0	1	01	Transparent (Binary) EF
0	0	0	0	0	0	1	0	02	Linear Fixed EF
0	0	0	0	0	1	0	0	04	Linear Variable EF
0	0	0	0	0	1	1	0	06	Cyclic EF
0	0	0	0	1	-	-	-	-	**Internal EF**
0	0	0	0	1	1	0	0	0C	Internal Linear Variable EF (or KEY EF)
0	0	0	0	1	0	0	1	09	Internal Transparent File (or Asymmetric KEY EF)

The size of the File Header block varies depending on the file type. Other values of FDB are considered invalid.

3.2.2 Data Coded Byte (DCB)

ACOS5 does not use this field. It is part of the header to comply with ISO-7816 part 4.

3.2.3 File ID

This is a 16-bit field that uniquely identifies a file in the MF or a DF. Each file under a DF (or MF) must be unique. There are a few pre-defined File ID's. They are:

3F00 - Master File

3FFF - Current DF

FFFF - RFU

A file cannot have an ID of 3FFF, FFFF and 0000.

3.2.4 File Size

This is a 16-bit field that specifies the size of the file. It does not include the size of the file header. For record-based EF's, the 1st byte indicates the maximum record length (MRL), while the 2nd indicates the number of records (NOR). For non record-based EF (Transparent EF), the 1st byte represents the high byte of the file size and the 2nd is the low-order byte. For MF and DF's, this field is not used.

3.2.5 Short File Identifier (SFI)

This is a 5-bit value that represents the file's Short ID. ACOS5 allows file referencing through SFI. The last 5 bits of the File ID does not necessarily have to match this SFI. 2 files may have the same SFI under a DF. In such case, ACOS5 will select the one created first.

3.2.6 Life Cycle Status Integer (LCSI)

This byte indicates the life status of the file, as defined in ISO7816 part 4. Each file, MF/DF/EF, would have a LCSI bye. In the MF's LCSI, the byte will determine the card's overall life cycle status. It can have the following values:

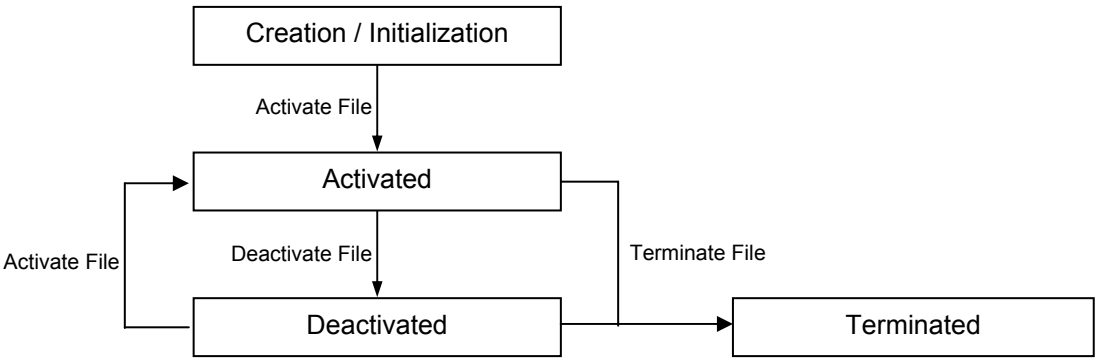
b8	b7	b6	b5	b4	b3	b2	b1	Hex	Meaning
0	0	0	0	0	0	-	1	01 or 03	Creation / Initialization state (default)
0	0	0	0	0	1	-	1	05 or 07	Operational state – activated
0	0	0	0	0	1	-	0	04 or 06	Operational state – deactivated
0	0	0	0	1	1	-	-	0C to 0F	Termination state

In Creation / Initialization states, all commands to the file will be allowed. **IMPORTANT:** After personalization, it is important to ACTIVATE the files to bring the card to operational state. This will enforce each file's security conditions.

In Activated state, commands to the file are allowed only if the file's security conditions are met.

In Deactivated state, only most commands to the file are not allowed except SELECT FILE, ACTIVATE FILE, DELETE FILE, and TERMINATE DF/EF.

In Terminated State, all commands to the file will not be allowed.



3.2.7 Security Attribute Compact Length (SAC Len)

This byte indicates the length of the SAC structure that is included in the file header below.

3.2.8 Security Attribute Expanded Length (SAE Len)

This byte indicates the length of the SAE structure that is included in the file header below.

3.2.9 DF Name Length / First Cyclic Record

If the file is a DF, this field indicates the length of the DF's Name. If the file is a Cyclic EF, this field holds the index of the last-altered record. Otherwise, this field is not used.

3.2.10 Parent Address

2 bytes indicating the physical EEPROM address of the file's parent DF.

3.2.11 Checksum

To maintain data integrity in the file header, a checksum is used by the COS. It is computed by XOR-ing all the preceding bytes in the header. Commands to a file will not be allowed if the file is found to have a wrong checksum.

3.2.12 Security Attribute Compact (SAC)

This is a data structure that represents security conditions for certain file actions. The data is coded in an Access Mode / Security Condition (AM/SC) structure as defined in ISO-7816. The maximum size of this field is 8 bytes. See Section 4.1.1 for more information.

3.2.13 Security Attribute Expanded (SAE)

This is a data structure that represents security conditions for certain card or application (directory) actions and is present in the either MF/DF. The data is coded differently from SAC, and is also defined in ISO-7816. The maximum size of this field is 32 bytes. See Section 4.1.2 for more information.

3.2.14 SE File ID (for DF only)

For DF, this field is made up of 2 bytes containing the File ID of one of its children. That file is known as the Security Environment File, which is processed internally by the COS.

3.2.15 File Control Information File ID (FCI File ID)

ACOS5 does not use this field. It is part of the header to comply with ISO-7816 part 4.

3.2.16 DF Name (for DF only)

For DF, this field is the file's Long Name. Files can be selected through its long name - which can be up to 16 bytes.

3.3 Internal Security Files

The behavior of the COS will depend on the contents of the security-related internal files. Typically, a DF should have: (1) a Key File to hold PIN codes (referred to as EF1) for verification, (2) a Key File to hold KEY codes (referred to as EF2) for authentication, (3) an SE file to hold security conditions and templates, and (4) an Asymmetric KEY EF to store RSA Keys.

A Key file is an Internal Linear Variable file. It may contain (1) PIN data structure or (2) KEY data structure.

3.3.1 PIN Data Structure

The PIN is used for VERIFY command. The file that contains PIN records must have an SFI of 1. This file will be referred to as EF1. Each PIN record has the following structure:

	PIN Identifier Byte	Error Counter	PIN
Byte	1	1	16 (max)

PIN Identifier Byte: PIN Identifier Byte identifies the PIN number and other options:

b7	b6	b5	b4	b3	b2	b1	b0	Description
1	-	-	-	-	-	-	-	The PIN is valid and activated
-	-	-	-	-	-	-	-	RFU
-	-	-	-	-	-	-	-	RFU
-	-	-	x	x	x	x	x	PIN Identifier

Error Counter: The error counter limits the number of consecutive unsuccessful attempts of PIN submission. This byte is split into two parts. The low nibble indicates the allowed number of retries ($CNT_{Allowed}$) and the high nibble indicates the number of retries left ($CNT_{Remaining}$).

b7	b6	b5	b4	b3	b2	b1	b0
$CNT_{Remaining}$				$CNT_{Allowed}$			

Each unsuccessful attempt will decrement $CNT_{Remaining}$. A successful submission of the PIN number will reset the $CNT_{Remaining}$ to the $CNT_{Allowed}$. If the lower nibble reaches zero, then the PIN is locked and further PIN submission is not possible.

If the error counter is 0xFF, then the error counter is not used and the PIN allows for unlimited number of retries. Hence, the maximum number of retries short of unlimited is 14 or 0x0E.

PIN: The PIN code whose length is between 1 and 16 bytes.

3.3.2 Key data structure

Keys are used for authentication commands. The file that contains the key records must have an SFI of 2. This file will be referred to as EF2. Each KEY record has the following structure:

Key ID	Key Type	Key Info	Algorithm	Key

				Reference	
Byte	1	1	1 or 2	1	8 or 16

Key ID: The five LSb's uniquely identifies a key record. If the MSb = 1, the current key record is valid, else it is invalid and cannot be used.

Key Type: This byte indicates the key's capabilities; and also tells the OS how to interpret the Key Info field.

Bit	Description
b7 – b2	RFU
b1	Key is capable of internal authentication, Key Info contains *usage counter.*
b0	Key is capable of external authentication, Key Info contains *error counter.*

Key Info: This field depends on the Key Type field. It contains Retry or Usage Counter(s) of the key. Depending on the Key Type field's bits, this field will hold the following: *Internal Authentication Usage Counter* (2 bytes) and External Authentication Error Counter (1 byte).

The Internal Authentication Usage Counter can limit the number of times a key can be used for Internal Authentication. Each execution attempt of the mutual authentication procedure will decrement the usage counter. When the counter reaches zero, the key will become invalid. The counter is two bytes allowing a counter up to 65,534 (0xFFFE). If the counter is 0xFFFF, then the usage of the key is unlimited.

The External Authentication Error Counter works the same as PIN Error Counter.

Key Algorithm Reference: this byte indicates what cryptograph algorithm the key is capable of:

B7 = RFU	B6 = RFU	B5 = RFU	B4 = RFU	B3 = RFU	B2 = ~AES	B1 = ~DES	B0 = 1DES/3DES	Value	Description
0	0	0	0	0	0	0	0	00	Key can be applied on 3DES and AES
0	0	0	0	0	0	0	1	01	Key can be applied on 1DES and AES (for 1DES, only 1st 8 bytes of key is used)
0	0	0	0	0	0	1	X	02-03	Key can be applied on AES only
0	0	0	0	0	1	0	0	04	Key can be applied on 3DES only
0	0	0	0	0	1	0	1	05	Key can be applied on 1DES only
								06-FF	Undefined

Key: The key whose length must be 8 (1DES) or 16 (3DES or AES) bytes.

3.3.3 Security Environment File

A Security Environment (SE) File is an Internal Linear Variable EF that stores SE templates. Every DF has a designated SE FILE, whose file ID is indicated in the DF's header block. An SE file can have up to 15 records; each record can have any of the following DO templates:

Tag	Description

80	SE identifier – this ID is referenced to by the SC byte(s) of a file's SAC or SAE
A4	Authentication Template – this determines if an action is to be allowed by COS
B4	Cryptographic Checksum Template
B8	Confidentiality Template
B6	Digital Signature Template
AA	Hash Template

3.3.4 Asymmetric Key EF

An asymmetric key file is an internal transparent file with FDB = 0x09. It stores public and private keys of RSA information. Get Data/Key is used to retrieve data from this type of file while Put Data/Key is used to store data in this file. Each file will contain a key and follows this format:

<table>
<tr><th>Byte</th><th>Size</th><th colspan="3">Description</th></tr>
<tr><td>1</td><td>1</td><td colspan="3">Key type:
00=Public;
01=Private non CRT
03=Private CRT
05=Private non CRT, capable of decipher (private exponentiation)
07=Private CRT, capable of decipher (private exponentiation)</td></tr>
<tr><td>2</td><td>1</td><td colspan="3">Key Length:
04 for 512-bit key (64 bytes)
08 for 1024-bit key (128 bytes)
10h for 2048-bit key (256 bytes)</td></tr>
<tr><td>3</td><td>2</td><td colspan="3">File ID of Key Pair
If the Key is public, then this field points to the EFID of its private key partner.
If the Key is private, then this field points to the EFID of its public key partner.</td></tr>
<tr><td>5</td><td>1</td><td colspan="3">Flags – There are two flags in the asymmetric key file occupying bit 0 and bit 1 of this byte:
b0 – Key file complete flag. Specifies if this key has been completely loaded.
b1 – Key file validated. Specifies a trial encryption and decryption has been done using this completed key.</td></tr>
<tr><td rowspan="2">6</td><td rowspan="2">Variable</td><td>Public key type</td><td>Private key type</td><td>Private CRT key type</td></tr>
<tr><td>e [8]
N [64, 128, or 256]</td><td>d [64, 128]</td><td>p [32, 64, or 128]
q [32, 64, or 128]
dp [32, 64, or 128]
dq [32, 64, or 128]
qInv [32, 64, or 128]</td></tr>
</table>

The flag Key file complete is used to specify if the key has been completely loaded. In commands like PUT KEY, each part of the key (RSA Private CRT key for example) cannot be loaded all at once from the limitation of the ISO-in command of 255 bytes. Therefore, we need to load it part by part. Only after all 5 parts of the Private CRT key is loaded would the *Key file complete* flag be set.

Once the *complete* flag is set and if the Key Pair field is specified with the counterpart's key pair *complete* flag set. The card will perform a trial encryption on the keys. It will pick a random number $R_{TRIAL} < N$, encrypt it using the public key file, then decrypt it using the private key file to ensure the matching key set works.

4 SECURITY

File commands are restricted by the COS depending on the target file's (or current DF's) security Access Conditions. These conditions are based on PINs and KEYs being maintained by the system. Card Commands are allowed if certain PINs or KEYs are submitted or authenticated.

Global PINs are PINs that reside in a PIN EF (EF1) directly under the MF. Likewise, local Keys are KEYs that reside in a KEY EF (EF2) under the currently selected DF. There can be a maximum of: 31 Global PINs, 31 Local PINs, 31 Global Keys, and 31 Local Keys at a given time.

4.1 File Security Attributes

Each file (MF, DF, or EF) has a set of security attributes set in its headers. There are two types of security attributes Security Attribute Compact (SAC) and Security Attribute Expanded (SAE).

4.1.1 Compact (SAC)

The SAC is a data structure that resides in each file. It indicates what file actions are allowed on the file, and what conditions need to be satisfied for each action. It starts with the AM byte, followed by SC byte(s). The AM byte is bit- coded as follows:

	b7	b6	b5	b4	b3	b2	b1	b0
For DF	Not used	Delete Self	Terminate	Activate	Deactivate	Create DF	Create EF	Delete child
For EF	Not used	Delete Self	Terminate	Activate	Deactivate	-	Update	Read
For Key	Not used	Delete Self	Terminate	Activate	Deactivate	MSE / PSO	Put data	Get data
For SE	Not used	Delete Self	Terminate	Activate	Deactivate	MSE restore	MSE store / delete	Read

The number of "1" bits in the AM byte determines the number of SC byte(s) that follow. Bits that are "0" imply that the associated action to the file has no condition (free). Bits with "1" means that a corresponding SC byte exists in the SAC, and that the associated action is allowed only if the SC condition is satisfied.

The SC byte is interpreted as follows:

b7	b6	b5	b4	b3	b2	b1	b0	Hex	Meaning
0	0	0	0	0	0	0	0	00	No condition (allow always)
1	1	1	1	1	1	1	1	FF	Never
-	-	-	-	0	0	0	0		RFU
-	-	-	-	Not all equal					Security environment identifier (SEID byte) from one to fourteen
-	-	-	-	1	1	1	1		RFU
0	-	-	-	-	-	-	-		At least one condition
1	-	-	-	-	-	-	-		All conditions
-	1	-	-	-	-	-	-		Secure messaging

The SE record is found in the SE file - whose ID is specified in the current DF's header. If the SE file is not found, or has incompatible file attributes (internal LV, MRL, NOR, etc.), then the command is denied.

Example: a DF with SAC of **7D 02 03 04 FF FF 02** means:
AM: 7D -> has 6 "1" bits (01111101), 6 SC bytes follow; all file actions except "create child EF" is present, "create child EF" is therefore free;

SC: 02 03 04 FF FF 02

- allow Delete Self if SE#2 is satisfied
- allow Terminate if SE#3 is satisfied
- allow Activate if SE#4 is satisfied
- do not allow Deactivate
- do not allow Create child DF
- allow Delete Child DF if SE#2 is satisfied

4.1.2 Expanded (SAE)

The SAE is a data structure that resides in each DF file header. It tells the COS whether or not to allow file commands to proceed. SAE is more general compared to SAC. The format of SAE is an access mode data object (AMDO) followed by one or more security condition data objects (SCDO).

<$AMDO_1$><$SCDO_1$> <$AMDO_2$><$SCDO_2$> … <$AMDO_n$><$SCDO_n$>

The COS will check if the command (APDU) falls under the SAE's <AMDO$_i$> if it does it will check the corresponding <SCDO$_i$>.

AMDO: The value field of this data object specifies the command description: CLA INS P1 P2. The low nibble of the TAG indicates which command byte(s) follow:

b7	b6	b5	b4	b3	b2	b1	b0	Meaning
1	0	0	0	x	x	x	x	The command description includes
1	0	0	0	1	-	-	-	CLA
1	0	0	0	-	1	-	-	INS
1	0	0	0	-	-	1	-	P1
1	0	0	0	-	-	-	1	P2

SCDO: May have the following Tags (and Length):

Tag	Length	Value	Meaning
90	00	-	Allow
97	00	-	Never
9E	1	SC Byte	SC Byte the same as SAC
A4	Var.	Authentication Template	Allow if AT condition is satisfied
A0	Var.	SC DO	One or more SC DO follows where at least 1 condition is met
AF	Var.	SC DO	One or more SC DO follows where all conditions are met

Example 1: An SAE of: **86 02 22 F2 97 00** means:
<AMDO>: 86 02 22 F2 -> command with INS=22 and P1 =F2
<SCDO>: 97 00 -> never
Hence, the SAE tells the COS not to allow APDU commands with INS=22 and P1=F2 in the current DF.

Example 2: An SAE of: **84 01 22 A4 06 83 01 81 95 01 08** means:
<AMDO>: 84 01 22 -> command with INS=22
<SCDO>: A4 06 83 01 81 95 01 08 -> allow command if local PIN #81 is submitted
Hence, the SAE tells the COS to allow commands with INS=22 only if the local PIN#1 is verified.

* In ACOS5, if one <AMDO> matches the command APDU, the COS will not check the succeeding <AMDO>'s.

4.2 Security Environment

Security conditions are coded in an SE File. Every DF has a designated SE FILE, whose file ID is indicated in the DF's header block. Each SE record has the following format:

<SE ID Template> <SE DO Template>

4.2.1 SE ID Template

The SE ID Template is a mandatory data object whose value states the identifier that is referenced by the SC byte of the security file attributes. The Tag is 0x80 with the length of 0x01.

4.2.2 SE DO Template

An SE DO template can be any of the following:

Template	Tag	Description
Authentication Template (AT)	A4	Stores condition(s) for a file action
Cryptographic Checksum Template (CCT)	B4	Specifies parameters for PSO or Secure Messaging Authenticity
Confidentiality Template (CT)	B8	Specifies parameters for PSO or Secure Messaging Confidentiality
Digital Signature Template (DST)	B6	Specifies parameters for RSA-related PSO commands
Hash Template (HT)	AA	Specifies parameters for hash-related PSO commands

In the templates below there will be the following fields:

Algorithm Reference with respect to DO template:

Value (hex)	Mode of operation	AT	HT	CCT	DST	CT-sym	CT-asym
00	ECB cipher with 3DES					X	
01	ECB cipher with 1DES					X	
02	CBC cipher with 3DES			X		X	
03	CBC cipher with 1DES			X		X	
04	ECB cipher with AES					X	
05	ECB cipher with AES					X	
06	CBC cipher with AES					X	
07	CBC cipher with AES					X	
10	RSA with PKCS#1 Signature mechanism				X		
11	RSA with no padding (raw)				X		
12 or 13	RSA Private decrypt/Public encrypt						X
20	SHA-1 hash		X				

Usage Qualifier Byte with respect to DO template:

bit	Description	AT	HT	CCT	DST	CT-sym	CT-asym
b7	Verification, Encipherment, External	X		X	X	X	X

	authentication						
b6	Computation, Decipherment, Internal authentication			X	X	X	X
b5	Secure messaging in response data.			X		X	
b4	Secure messaging in command data.			X		X	
b3	User authentication using PIN key.	X					
b2	0 – RFU						
b1	0 – RFU						
b0	0 – RFU						

4.2.3 Authentication Template:

The Authentication Template defines the security condition that must be met for this SE to be satisfied. The security conditions are either PIN or Key authentications.

The tag of AT is 0xA4 and its value contains one more data objects.

Tags recognized within AT:

Tag	Length	Meaning
83	01	It indicates the identifier of which Key or PIN to use. If its MSb is 1, use the Pin or Key file under the currently selected DF. Otherwise, use the Pin or Key file under the MF.
95	01	Indicates what action to perform: Authenticate or Verify. The value of this tag has the following meaning: bit7 = AUTHENTICATE the referenced key in tag 83 bit3 = VERIFY the referenced PIN in tag 83

Example #1: If the following record is specified in the SE File:

80 01 03 A4 09 83 01 84 83 01 81 95 01 80

It has the following meaning:

- The SE ID is 03 (SE#3).
- AT is: A4 09 83 01 84 83 01 81 95 01 80; this contains two 0x83 tags inside. This means:
  - 1st condition in AT is: 83 01 84 95 01 80 -> allow command if local KEY 84 is authenticated;
  - 2nd condition in AT is: 83 01 81 95 01 80 -> allow command if local KEY 81 is authenticated;
- SE conditions are referenced by an SC byte (in the SAC field of the target file). If the SC byte's MSb is set, then allow command if both conditions are satisfied. If the SC byte's MSB is not set, allow command if at least one condition is satisfied.

Example #2: If the following record is specified in the SE File:

80 01 05 A4 09 83 01 84 83 01 01 95 01 88

It has the following meaning:

- The SE ID is 05 (SE#5).

- AT is: A4 09 83 01 84 83 01 01 95 01 88; contains 2 conditions inside it.
  - 1st condition in AT is: 83 01 84 95 01 88 -> allow command if local KEY 84 is authenticated AND if local PIN 84 is verified;
  - 2nd condition in AT is: 83 01 01 95 01 88 -> allow command if global KEY 81 is authenticated AND if global PIN 01 is verified;
- SE conditions are referenced by an SC byte (in the SAC field of the target file). If the SC byte's MSB is set, then allow command if both conditions are satisfied. If the SC byte's MSB is not set, allow command if at least one condition is satisfied.

4.2.4 Cryptographic Checksum Template

CCT defines which parameters to use in computing for the MAC, which is used in Secure Messaging and/or PSO. It has the following sub-tags:

Tag	Length	Meaning
83	01	It indicates the identifier of which Key to use. If its MSb is 1, use EF2 under the currently selected DF. Otherwise, use the EF2 under the MF.
95	01	Qualifying Byte: bit4 and bit5 indicates that the template can be used in Secure messaging. Bit6 indicates that the template can be used for PSO.
80	01	Algorithm Reference (excluding AES), please refer to section 4.2.1
84	00	If present, this tells ACOS5 to use the Session Key as key and ignore tag 83.
87	08	If present, this tells ACOS5 to use the initial vector; Algorithm Reference must be CBC

4.2.5 Confidentiality Template

CT defines which parameters to use in encrypting / decrypting data in Secure Messaging and/or PSO. It has the following sub-tags:

Tag	Length	Meaning
83	01	It indicates the identifier of which Key to use. If its MSb is 1, use EF2 under the currently selected DF. Otherwise, use the EF2 under the MF.
95	01	Qualifying Byte: bit4 and bit5 indicates that the template can be used in Secure messaging. Bit6 indicates that the template can be used for PSO.
80	01	Algorithm Reference, please refer to section 4.2.1
84	00	If present, this tells ACOS5 to use the Session Key as key and ignore tag 83.
87	08/10	If present, this tells ACOS5 to use the initial vector; Algorithm Reference must be CBC
81	02	References an Asymmetric key EF to be used in PSO encipher or decipher.

4.2.6 Digital Signature Template

DST defines which parameters to use in asymmetric key-related operations. It has the following sub-tags:

Tag	Length	Meaning
80	01	Algorithm Reference, please refer to 4.2.1

95	01	Qualifying Byte: bit6 indicates that the template can be used in signing / decrypting; and the file referenced is a PRIVATE key EF. Bit7 indicates that the template can be used in verifying signature / encrypting and the referenced file is a PUBLIC key.
81	02	References an Asymmetric key EF to be used in PSO encipher or decipher.

4.2.7 Hash Template

HT defines which parameters to use in PSO-HASH. It has just one sub-tag:

Tag	Length	Meaning
80	01	Algorithm Reference: ACOS5 only supports SHA1 (20h)

4.3 Mutual Authentication

Mutual Authentication is a process in which both the card and the card-accepting device verify that the respective entity is genuine. A *Session Key* is the result of a successful execution of mutual authentication. The session key is only valid during a *session*. A session is defined as the time after a successful execution of the mutual authentication procedure and a reset of the card or the execution of another mutual authentication procedure.

4.3.1 Mutual Authentication Procedure

To provide maximum flexibility, ACOS5 uses two different pairs of DES keys: A *terminal key* K_T and a *card key* K_C. These pair of keys should both be onboard, and they both should either be 8 bytes (for single DES) or both 16 bytes (for triple DES)

A random number generator is onboard ACOS5 to generate a *card random number,* RND_C, in response of the GET CHALLENGE command.

Please follow the below diagram for the detailed steps of mutual authenticate procedure.

Terminal		ACOS5
GET CHALLENGE 00 84 00 00 08	➔ ➔	Generate and return card challenge RND_C
1. Compute Re Re = ENC (RNDc, K_T) 2. EXTERNAL AUTHENTICATE 00 82 00 K_T 08 R_e	➔	1. Check if K_T index is valid and is capable of EXT AUTH. 2. Verify if R_e = ENC (RND_C, K_T)
	➔	Return 9000 if correct
1. Generate Terminal challenge data		

RND_T 2. INTERNAL AUTHENTICATE 00 88 00 K_C 08 RND_T	➔ ←	1. Compute Ri Ri = ENC (RND_T, K_C) 2. Return 6108
GET RESPONSE 00 C0 00 00 08 Verify if Ri = ENC (RND_T, K_C)	➔ ←	Return Ri

In the procedure, the encryption, ENC, is either DES or 3DES depending on the Keys' algorithm Reference used. Note that AES is not supported for mutual authentication.

4.3.2 Session Key Computation

The Session Key is formed through Mutual Authentication between card and terminal. The Session Key's formula depends on the Algorithm Reference field of the KEY. If both Internal key (card key) and external key (terminal key) have Algorithm Reference field of 3DES, then a 16-byte session key will be formed, otherwise, the session key formed is 8 bytes long (Single DES).

For 16-byte session key K_S, triple DES is used and K_S is generated as follows:

$$K_S = K_{sL} \,||\, K_{sR}$$

Where K_{sL} is the first 8-byte (or the left half) of the session key:

$$K_{sL} = K_{cLeft} \otimes R_E \otimes K_{TLeft} \otimes R_I$$

And K_{sR} is the second 8-byte (or right half) of the session key:

$$K_{sR} = K_{cRight} \otimes \sim R_E \otimes K_{TRight} \otimes \sim R_I$$

For 8-byte session key K_S, single DES is used and K_S is generated as follows:

$$K_S = K_c \otimes R_E \otimes K_T \otimes R_I$$

4.4 Secure Messaging

ACOS5 supports Secure Messaging (SM) for Authentication and Confidentiality. It applies to the following commands:

ISO-IN	ISO-OUT
Create File Update Binary	Read Binary Read Record

Erase Binary Update Record Append Record Activate File Deactivate File Terminate DF Terminate EF Delete File Manage Security Environment Perform Security Operation Generate RSA Key Pair Put Key	Get Key

There are 2 modes of SM that can be applied to two different situations. The first mode is SM for authenticity (SM-sign) the other is SM for confidentiality (SM-enc). The SM modes will be applied to both the command and response data. Whether to use the SM mode will depend on the conditions set in the current security environment and the command to perform. If the target file has an SAC security condition byte with bit 6 set using an SE ID number. The SE in that SE ID number must have CCT and optionally CT-sym templates with usage qualifier byte bit 4 and 5 set.

Notations:

To describe the two SM modes adequately, there are some notations to be introduced in this section. The following are the possible ISO7816 part 3 communications protocol command and response pairs.

Commands:

a. Without command data:

	CLA	INS	P1	P2	P3
Bytes	1	1	1	1	1

b. With command data

	CLA	INS	P1	P2	P3 = n	Data in
Bytes	1	1	1	1	1	n

Responses:

a. Without response data

	Sw1 Sw2
Bytes	2

b. With response data

	Response	Sw1 Sw2
Bytes	n	2

The class (CLA) byte stated above does not have the SM bits set.
The modified CLA* below has the SM bits b3 and b2 set to 1. That is, the original CLA XOR 0x0C.
P3 is the normal length of the command data.
P3* is the length of the command data under secure messaging.

Sequence number:

The sequence number (seq#) *n* is used as the Initial vector in the CBC calculation. The start of the sequence number is the increment of the random number of the LAST TWO bytes of the GET CHALLENGE command. The response data is the increment of the command sequence number. The sequence number is to prevent man-in-the-middle attack.
If a command is sent from the terminal to the card SM'ed with a sequence number *n*, and secure messaging is successful, the card will reply with a MAC computed with an initialization vector of *n* + 1. The next secure messaging command to send will use the sequence number *n* + 2.

Authentication and integrity

The COS uses the SM key and DES / 3DES in CBC mode to compute the MAC. The notation is as follows:

SIGN_CBC$_4$ (data to sign, sequence number)

Only the first 4 bytes of the resultant MAC are used for the command and response data.

Confidentiality

The COS uses the SM key and DES / 3DES in CBC mode to compute the encryption. The notation is as follows:

ENC_CBC (plaintext to encrypt, sequence number)

Padding:

DES and 3DES algorithms take input block sizes of 8. Therefore appropriate padding is necessary. If the data to sign or encrypt is not in a multiple of 8, a 0x80 byte is appended followed by 0 to 6 0x00 to make the data to sign to be a multiple of 8 bytes.

Secure Messaging Data Object:

Secure Messaging Data Objects (SMDO) are necessary to describe the data elements fully when transferring in SM mode. The following SMDO are supported in ACOS5 Secure Messaging:

Tag	Length	Description
81	Var.	Plain Value
87	Var.	Padding-content indicator byte followed by cryptogram
89	4	Command header (CLA* INS P1 P2)
8E	4	Cryptographic checksum
97	1	Original P3 in an ISO-out command
99	2	Processing status word (Sw1 Sw2) of the command.

The exact usage of these SMDO is stated in the next section.
The following subsections list the possible command and response data pair. These will depend on if the current SE requires SM on the operation to be performed. This is stated by the current EF's SAC conditions. If the SC byte of an operation requires secure messaging with an SE ID number. In the SE file, that SE ID number must contain the CT-sym and/or CCT templates set with usage qualifier byte set with bit b4 and b5 set.

4.4.1 SM for Authenticity

4.4.1.1. Case 1: No command and no response.

	CLA*	INS	P1	P2	P3*	8E	04	MAC_{cmd}
Bytes	1	1	1	1	1	1	1	4

CLA* = CLA OR 0x0C
P3* = 06
MAC_{cmd} = SIGN_CBC (<89 04 CLA* INS P1 P2> padding, ++Seq#)

If the command accepts secure messaging, the key has been established and the MAC_{cmd} is correct, the response will be 0x610C. Note that 0x610C may not actually mean that the command is successful. It merely means that the Secure Messaging is successful. A subsequent call to GET RESPONSE will yield the actual Status Words stating success or error. The GET RESPONSE must have P3 = 0C exactly. Otherwise 0x6C0C will be replied without SM.

	99	02	Sw1 Sw2	8E	04	MAC_{rsp}	90 00
Bytes	1	1	2	1	1	4	2

MAC_{rsp} = SIGN_CBC (<89 04 CLA* INS P1 P2> <99 02 Sw1 Sw2> padding, ++Seq #)

The field Sw1 Sw2 is the actual status word returned for the command. The last status word bytes 90 00 states that the SM is successful.
If the key hasn't been established or the MAC_{cmd} is incorrect, the response will only have Sw1 Sw2 stating 0x6985 or 0x6884 respectively (instead of 0x610C).

4.4.1.2. Case 2: ISO-in

	CLA*	INS	P1	P2	P3*	81	P3	Command Data	8E	04	MAC_{cmd}
Bytes	1	1	1	1	1	1	1	N	1	1	4

CLA* = CLA OR 0x0C
P3* = 2 + n + 2 + 4
MAC_{cmd} = SIGN_CBC (<89 04 CLA* INS P1 P2> <81 P3 Command-Data> padding, ++Seq#)

Since the maximum of P3* must be less than 255, with the MAC, P3 is set to be less than or equal to 240.

The response, if the command is successful, as in the first case, is as follows:

	99	02	Sw1 Sw2	8E	04	MAC_{rsp}	90 00
Bytes	1	1	2	1	1	4	2

MAC_{rsp} = SIGN_CBC (<89 04 CLA* INS P1 P2> <99 02 Sw1 Sw2> padding, ++Seq#)

4.4.1.3. Case 3: ISO-out

ISO-out effectively becomes an ISO-in command when the SM field is set.

	CLA*	INS	P1	P2	P3*	97	01	P3	8E	04	MAC_{cmd}
Bytes	1	1	1	1	1	1	1	1	1	1	4

CLA* = CLA OR 0x0C
P3* = 09
MAC_{cmd} = SIGN_CBC (<89 04 CLA* INS P1 P2> <97 01 P3> padding, ++Seq#)

If the command accepts secure messaging, the key has been established and the MAC_{cmd} is correct, the response will be 0x61xx. Same as ISO-in, the status word of 0x61xx only means that SM on the command is successful. The actual success of the overall command will depend on the SW1 SW2 data object when a subsequent GET RESPONSE is called with P3 = xx, where xx can be 00-FF. If xx is 00, 256 bytes of data can be returned.

Note the Get Response must be called with P3 = xx exactly, else 0x6Cxx will be returned (without SM). This is to prevent man in the middle attack. The original response data *n* must be less than or equal to 240 bytes.

The response is as follows:

	81	P3	Response Data	99	02	Sw1 Sw2	8E	04	MAC_{rsp}	90 00
Bytes	1	1	n	1	1	2	1	1	4	2

MAC_{rsp} = SIGN_CBC (<89 04 CLA* INS P1 P2> <99 02 Sw1 Sw2> <81 P3 Response Data> padding, ++Seq #)

4.4.1.4. Case 4: ISO-in-out

SM also applies to ISO-in commands that returns data (SW1 SW2 = 61XX). Instead of the command returning SW1 SW2 = 61XX, it will return the response data together with SW1 SW2 = 9000.

	CLA*	INS	P1	P2	P3*	81	P3	Command Data	8E	04	MAC_{cmd}
Bytes	1	1	1	1	1	1	1	n	1	1	4

CLA* = CLA OR 0x0C
P3* = 2 + n + 2 + 4
MAC_{cmd} = SIGN_CBC (<89 04 CLA* INS P1 P2> <81 P3 Command-Data> padding, ++Seq#)

Since the maximum of P3* must be less than 255, with the MAC, P3 is set to be less than or equal to 240.

If the command accepts secure messaging, the key has been established and the MAC_{cmd} is correct, the response will be 0x61xx. Same as ISO-in, the status word of 0x61xx only means that SM on the command is successful. The actual success of the overall command will depend on the SW1 SW2 data object when a subsequent GET RESPONSE is called with P3 = xx, where xx can be 0x00-0xFF. If xx is 00, 256 bytes of data can be returned.

Note the Get Response must be called with P3 = xx exactly, else 0x6Cxx will be returned (Without SM). This is to prevent man in the middle attack. The original response data *n* must be less than or equal to 240 bytes.

The response is as follows:

	81	P3	Response Data	99	02	Sw1 Sw2	8E	04	MAC_{rsp}	90 00
Bytes	1	1	n	1	1	2	1	1	4	2

MAC_{rsp} = SIGN_CBC (<89 04 CLA* INS P1 P2> <99 02 Sw1 Sw2> <81 P3 Response Data> padding, ++Seq #)

4.4.2. SM for Authenticity and Confidentiality

4.4.2.1. Case 1: ISO no command and no response.

	CLA*	INS	P1	P2	P3*	8E	04	MAC_{cmd}
Bytes	1	1	1	1	1	1	1	4

CLA* = CLA OR 0x0C
P3* = 06
MAC_{cmd} = SIGN_CBC (<89 04 CLA* INS P1 P2> padding, ++Seq#)

If the command accepts secure messaging, the key has been established and the MAC_{cmd} is correct, the response will be 0x610C. Note that 0x610C may not actually mean that the command is successful. It merely means that the Secure Messaging is successful. A subsequent call to GET RESPONSE will yield the actual Status Words stating success or error. The GET RESPONSE must have P3 = 0C exactly. Otherwise 0x6C0C will be replied without SM.

	99	02	Sw1 Sw2	8E	04	MAC_{rsp}	90 00
Bytes	1	1	2	1	1	4	2

MAC_{rsp} = SIGN_CBC (<89 04 CLA* INS P1 P2> <99 02 Sw1 Sw2> padding, ++Seq#)

The field Sw1 Sw2 is the actual status word returned for the command. The last status word bytes 90 00 states that the SM is successful.

If the key hasn't been established or the MAC_{cmd} is incorrect, the response will only have Sw1 Sw2 stating 0x6985 or 0x6884 respectively.

4.4.2.2. ISO-in

	CLA*	INS	P1	P2	P3*	87	L_{87}	Pi	Encrypted Data	8E	04	MAC
Bytes	1	1	1	1	1	1	1	1	n*	1	1	4

CLA* = CLA OR 0x0C
P3* = 3 + n* + 2 + 4
L_{87} = n* + 1 (Length of Tag 87)
Pi = Padding indicator: 00 – No padding is used in encrypted data
01 – Padding is used in encrypted data
n* = P3 + length (padding)
Encrypted Data = ENC_CBC (<Command Data> padding, ++Seq#))
MAC = SIGN_CBC (<89 04 CLA* INS P1 P2> <87 L_{87} Pi Encrypted Data> padding, Seq#)

Since the maximum of P3* must be less than 255, with the MAC, padding and the SMDO, the original P3 must be less than or equal to 240 bytes. Note that in the MAC calculation, the sequence number is not pre-incremented. This is because the encryption and the MAC will use the same sequence number with the encryption to be performed first.

The response, as in the first case, is as follows:

	99	02	Sw1 Sw2	8E	04	MAC_{rsp}	90 00
Bytes	1	1	2	1	1	4	2

MAC_{rsp} = SIGN_CBC (<89 04 CLA* INS P1 P2> <99 02 Sw1 Sw2> padding, ++Seq#)

4.4.2.3. ISO-out

ISO-out effectively becomes an ISO-in command when the SM field is set.

	CLA*	INS	P1	P2	P3*	97	01	P3	8E	04	MAC_{cmd}
Bytes	1	1	1	1	1	1	1	1	1	1	4

CLA* = CLA OR 0x0C
P3* = 9
MAC_{cmd} = SIGN_CBC (<89 04 CLA* INS P1 P2> <97 01 P3> padding, ++Seq#)

If the command accepts secure messaging, the key has been established and the MAC_{cmd} is correct, the response will be 0x61xx. Same as ISO-in, the status word of 0x61xx only means that SM on the command is successful. The actual success of the overall command will depend on the SW1 SW2 data object when a subsequent GET RESPONSE is called with P3 = xx, where xx can be 0x00-0xFF. If xx is 00, 256 bytes of data can be returned.

Note the get response must be called with P3 = xx exactly, else 0x6Cxx will be returned. This is to prevent man in the middle attack. The original data out *n* must be less than or equal to 240 bytes. Else, it will be segmented appropriately before it is encrypted, MAC'ed and sent.

The response is as follows:

	87	L_{87}	Pi	Encrypted data	99	02	Sw1 Sw2	8E	04	MAC_{rsp}	90 00
Bytes	1	1	1	n*	1	1	2	1	1	4	2

L_{87} = n* + 1 (Length of Tag 87)
Pi = Padding indicator length : 00 – No padding is used in encrypted data
01 – Padding is used in encrypted data
Encrypted Data = ENC_CBC (<Response Data> padding, ++Seq#))
MAC_{rsp} = SIGN_CBC (<89 04 CLA* INS P1 P2> <99 02 Sw1 Sw2> <87 L_1 Pi Encrypted Data> padding, Seq #)

Note that in the MAC calculation, the sequence number is not pre-incremented. This is because the encryption and the MAC will use the same sequence number with the encryption to be performed first.

4.4.2.4. ISO in and out

For ISO-IN commands that returns data (SW1 SW2 = 61XX):

	CLA*	INS	P1	P2	P3*	87	L_{87}	Pi	Encrypted Data	8E	04	MAC
Bytes	1	1	1	1	1	1	1	1	n*	1	1	4

CLA* = CLA OR 0x0C
P3* = 3 + n* + 2 + 4
L_{87} = n* + 1 (Length of Tag 87)

Pi = Padding indicator: 00 – No padding is used in encrypted data
01 – Padding is used in encrypted data
n* = P3 + length (padding)
Encrypted Data = ENC_CBC (<Command Data> padding, ++Seq#))
MAC = SIGN_CBC (<89 04 CLA* INS P1 P2> <87 L_{87} Pi Encrypted Data> padding, Seq#)

Since the maximum of P3* must be less than 255, with the MAC, padding and the SMDO, the original P3 must be less than or equal to 240 bytes. Note that in the MAC calculation, the sequence number is not pre-incremented. This is because the encryption and the MAC will use the same sequence number with the encryption to be performed first.

The response is as follows:

	87	L_{87}	Pi	Encrypted data	99	02	Sw1 Sw2	8E	04	MAC_{rsp}	90 00
Bytes	1	1	1	n*	1	1	2	1	1	4	2

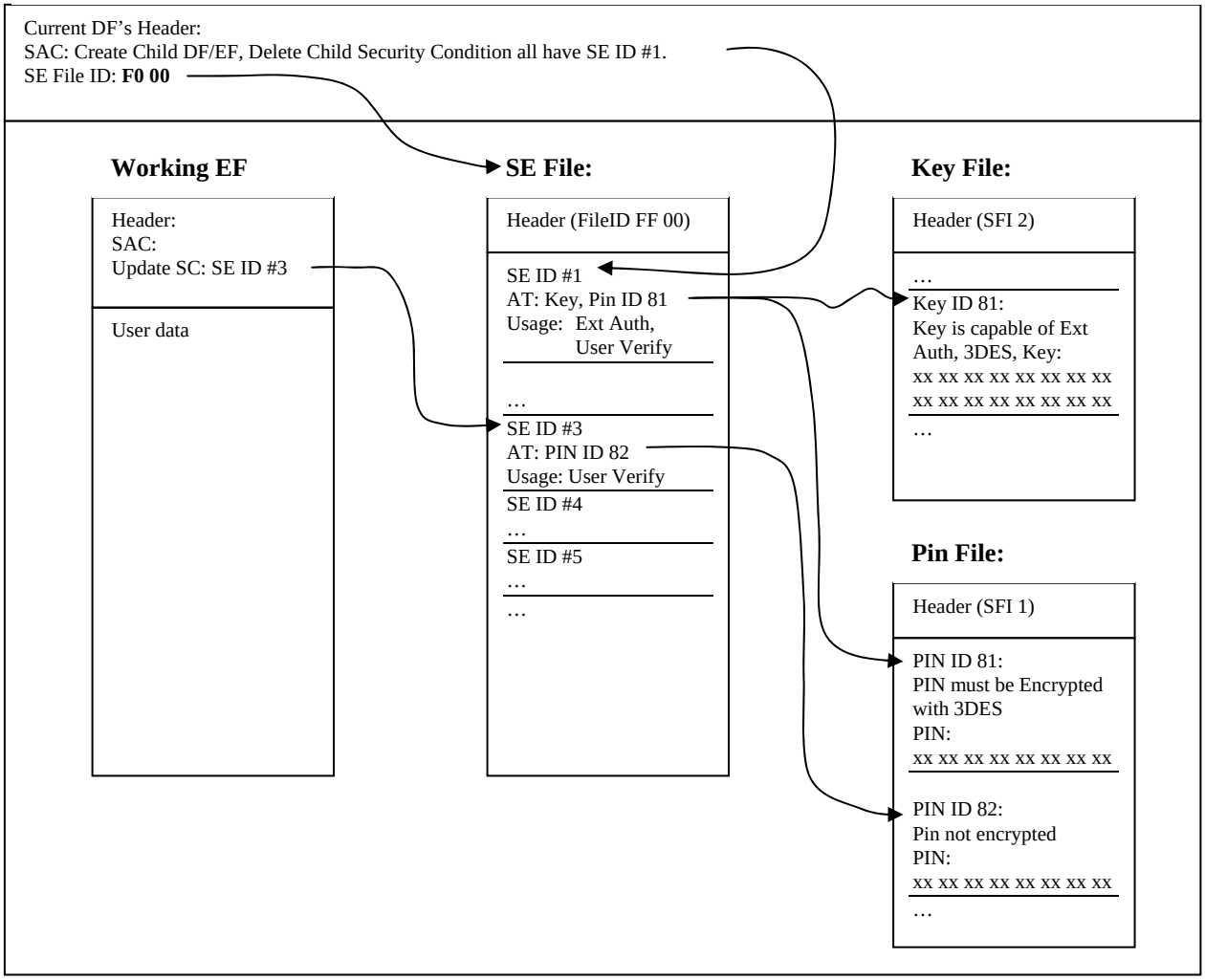
L_{87} = n* + 1 (Length of Tag 87)
Pi = Padding indicator length : 00 – No padding is used in encrypted data
01 – Padding is used in encrypted data
Encrypted Data = ENC_CBC (<Response Data> padding, ++Seq#))
MAC_{rsp} = SIGN_CBC (<89 04 CLA* INS P1 P2> <99 02 Sw1 Sw2> <87 L_1 Pi Encrypted Data> padding, Seq #)

Note that in the MAC calculation, the sequence number is not pre-incremented. This is because the encryption and the MAC will use the same sequence number with the encryption to be performed first.

4.5 Interaction between Security Conditions and Internal Security EFs

At first glance, the previous sections may seem unclear as to how the DF or working EFs interacts with different internal EFs (SE File, Key File, PIN File) to provide security for the files in question. The figure below provides an example of how all these files work together.

Current DF/MF:



In this example, the current DF has an SE file of F0 00. Inside this SE file, all the security environments are defined for the current DF. SE ID #1 defines the security conditions that must be satisfied before the current DF can execute a create child EF/DF or delete child. SE ID #1 contains an AT template for both external authentication and user verification with key reference of 81 (The most significant bit states the use of local key). In the DF's key file and PIN file the record of 81 (the most significant bit states the key is valid) will define the key and PIN that must be satisfied before any create or delete file command can be called in this DF.

In the Working EF, the SAC in the file header has update security condition set with SE ID #3. When update record/binary command is called on this file, the card will check the SE ID number 3 and find the Authentication Template. It will then see that User Verification is needed using PIN ID of 82. Then, it will check if the PIN ID of 82 in the PIN file has been verified beforehand.

4.5.1. Command security conditions

Before ACOS5 checks the security conditions of a file command, it will allow it based on the following considerations:

- Target file's LCSI is in creation / initial stage
- Target file's SAC field is not present
- The corresponding SC byte from the target file's SAC is 0.

And deny the action if:

- The corresponding SC byte from the target file's SAC is 0xff
- The SE file is not present or is invalid
- The SE record referenced by SC is not found or is invalid
- The AT template is not found in the SE record referenced by SC
- If both 97 and 90 tags are found in the AT, COS will choose 97
- If tag 95 is not present in AT
- If any sub-tag of the AT has invalid tag(s) or length(s)

4.5.2. Secure Messaging conditions

Possible causes of Secure Messaging error (SW1 SW2 = 6884) are:

- Card Challenge Data not available (user did not issue GET CHALLENGE)
- P3 is less than 6
- MF does not exist
- Current DF is blocked, or target file is blocked
- The target file's SAC is not present
- The SC of the target file's SAC is 0xFF or 0x00
- SE file is not present, or is invalid
- SE record referenced by SC is not found
- Plain or encrypted data length is 0x00 or > 240
- Encrypted data length is not a multiple of 8
- For ISO-OUT command, tag 97 (original P3) field is not present
- Contains encrypted data but no qualifying CT is found in the corresponding SE record
- No qualifying CCT SM template is found in the corresponding SE record
- CCT SM template's Key and Algorithm Reference are not compatible (must be CBC DES)
- MAC presented is wrong
- CT SM template' Algorithm Reference can only be DES

5 COMMANDS

5.1 Create File

CLA	00 - Clear Mode 0C - SM Mode
INS	E0
P1	00
P2	00
P3	Length of Data
Data	FCP TLV of File to be created

The FCP TLV has tag of 0x62. Its Length is the size of the template, and must be equal to (P3 – 2). The template has one or more of the following encapsulated TLV's:

Tag	Length	Value	MF	DF	Transparent	Linear Fixed	Linear Variable	Cyclic	Key EF	Purse EF	Remarks
80	02	Size (in bytes) of the Transparent EF			O						If not specified, default is 0x0000
82	01	File Descriptor Byte (FDB)	M	M	M	M	M	M	M	M	Please refer to Section 3.2.1 for valid values
	02	FDB + DCB	O	O	O	O	O	O	O	O	If DCB is not specified, default is 0x00
	05	FDB + DCB + 00 + MRL + NOR				O	O	O	O	O	If MRL / NOR is not specified, default is 0x00
	06	FDB + DCB + 00 + MRL + 00 + NOR				O	O	O	O	O	If MRL / NOR is not specified, default is 0x00
83	02	File ID	M	M	M	M	M	M	M	M	Please refer to Section 3.2.3 for valid values
84	<= 10h	DF Long Name	O	O							
88	01	Short File ID	O	O	O	O	O	O	O	O	If not specified, the default is the 5 LSb of the File ID
8A	01	Life Cycle Stage Integer	O	O	O	O	O	O	O	O	If not specified, the default is 0x01. Refer to section 3.2 for valid values.
8C	<= 08	Security Attributes Compact	O	O	O	O	O	O	O	O	Please refer to Section 4.1.1 for more details
AB	<= 20h	Security Attributes Extended	O	O							Please refer to Section 4.1.2 for more details
8D	02	SE File ID	O	O							Please refer to Section 4.2 for more details

M – Mandatory
O – Optional
Blank – encapsulated TLV will be ignored

The mandatory fields are: (1) File ID and (2) FDB. The newly created file will then be the Currently Selected EF/DF.

If duplicate tags are sent to the command, the latter one will be used.

Valid FDB values are: 3F (MF), 01 (Transparent EF), 02 (Linear Fixed EF), 04 (Linear Variable EF), 06 (Cyclic EF), 0x0C (Internal LV EF, or KEY EF), and 0x0E (Internal Cyclic EF, or Purse EF).

File ID's cannot be: 0xFFFF, 0x0000 and 0x3FFF.

Valid LCSI are: 01 (Creation); 03 (Initialization); 04 or 06 (Deactivated); 05 or 07 (Activated). LCSI having value >= 08 are considered Terminated.

The MF's file ID should always be 0x3F00, and should always be the 1st created file. You can create as many DF / EF files, and as many DF levels, as long as the card memory space holds. There cannot be duplicate File ID's / DF Names under a DF.

Please refer to section 5 of this document for examples on using this command.

For Secure Messaging, ACOS5 will use bit1 of the current DF's SAC AM byte as SE identifier (hence, "Create EF Child" is always assumed).

Return Status

6A86	Incorrect P1 / P2
6700	Wrong P3, must be consistent with the Length of the FCP TLV Must be: 9 <= P3 <= 90, and FCP length must be equal to (P3 – 2)
6984	FCP tag must be 0x62 FCP has invalid sub-tag(s) FCP sub-tag(s) has invalid length Invalid FDB Invalid File ID Invalid LCSI Trying to create a non-MF file on an empty card Trying to create an MF, but an MF already exists
6283	Current DF is terminated/ deactivated
6982	Security condition not satisfied
6A89	Target file ID matches current DF's ID Target file ID already exists in current DF Target DF Name matches current DF's name Target DF Name already exists in current DF
6A84	Not enough free space in card to create file MF Offset is not set in OTP area

5.2 Select File

CLA	00
INS	A4
P1	00 - if Data contains File ID 04 - if Data contains DF name
P2	00
P3	00 - select MF, 02 - Data contains File ID 1 <= P3 <= 16 – select DF by name
Data	File ID or DF Name

Search Sequence for Target File ID is: current DF -> current DF's children -> current DF's parent -> current DF's siblings -> MF -> MF's children.

Search Sequence for Target DF Name is: current DF -> current DF's children -> current DF's parent

On success, SELECT FILE will return SW1 SW2 = 61XX. You can retrieve XX bytes (via GET RESPONSE command with P3 = XX) to get the FCP template of the selected file. The template complies with the TLV table defined in section 5.1.

Return Status

6283	Target file is blocked, but is selected
6982	Target file has wrong checksum in header, is corrupted
6986	No MF found in card
6A82	File not found
6A86	Invalid P1 / P2: P2 must be 0x00, P1 must be 0 or 4
6A87	Wrong P3, P3 not compatible to P1
61nn	File selected successfully. Issue GET RESPONSE with P3 =NN in order to retrieve the file's FCI template

5.3 Read Binary

CLA	00 - Clear mode 04 - SM mode
INS	B0
P1	If MSb = 1, P1 holds SFI in 5 LSb, else P1 holds high offset
P2	Low offset
P3	Bytes to Read

If MF does not exist, READ BINARY allows you to directly read the EEPROM's contents. P1-P2 holds the physical address while P3 holds the number of bytes to read (please refer to section 8 for valid EEPROM physical addresses).
If MF exists, this command follows READ BINARY command specified in ISO7816 part 4.

For SM, SFI referencing is not allowed.

Return Status

6282	Invalid offset
6283	Current DF is blocked or target EF is blocked
6700	Incorrect P3, length exceeds file size limit
6981	Wrong file type; target file must be TRANSPARENT EF
6982	Target file's header block has wrong checksum, or security condition not satisfied
6986	No EF selected
6A82	Referenced SFI not found
6B00	Invalid P1/P2, invalid SFI Invalid P1/P2 offset
6Cnn	Wrong P3; NN = maximum bytes available in file to read
6F00	I/O error

5.4 Update Binary

CLA	00 – Clear mode 04 – SM mode
INS	D6
P1	If MSb = 1, P1's 5 LSb holds SFI, else P1 holds high start offset
P2	Low start offset
P3	Number of Bytes to Update
Data	Bytes to Update

If MF does not exist, UPDATE BINARY allows you to directly write the EEPROM contents. P1-P2 holds the starting physical address of the card while P3 specifies the number of byte to update (please refer to section 8 for valid EEPROM physical addresses).

If MF exists, this command follows UPDATE BINARY command specified in ISO7816 part 4.

For SM, SFI referencing is not allowed.

Return Status

6282	Invalid offset
6283	Current DF is blocked or target EF is blocked
6700	Incorrect P3, length exceeds file size limit
6981	Wrong file type; target file must be TRANSPARENT EF
6982	Target file's header block has wrong checksum, or security condition not satisfied
6986	No EF selected
6A82	Referenced SFI not found
6B00	Invalid P1/P2, invalid SFI Invalid P1/P2 offset
6Cnn	Wrong P3; NN = maximum bytes available in file to update
6F00	I/O error

5.5 Read Record

CLA	00 - Clear mode 04 - SM mode
INS	B2
P1	Depends on P2, see below
P2	If 5 MSb <> 00000, it is SFI; 3 LSb: see below
P3	Bytes to Read

The last 3 bits of P2:
0: reference 1st record
1: reference last record
2: reference next record
3: reference previous record
4: reference record indexed by P1
others: RFU

If P2 = 4 then record indexing starts at 1.

If the file's MRL or NOR field is zero, the COS will return 6A83 status code (record not found).

For Linear EF's, it is possible to have P3 < the EF's MRL.

For Cyclic EF's, it is possible to read the Current Record (see 3.2.9) by having P2 = 4 and P1 = 0.

Return Status

6283	Current DF is blocked, or Target EF is blocked
6700	Incorrect P3
6981	Wrong file type; target file must be RECORD-BASED EF
6982	Target file's header block has wrong checksum, may be corrupted Security condition not satisfied
6986	No DF selected, or no EF selected
6A82	Referenced SFI not found
6A83	Record not found, invalid record reference
6B00	Invalid P1/P2 Invalid SFI SFI referencing is not allowed in SM mode
6Cnn	nn is the maximum valid value for P3; Wrong P3, P3 is greater than the file's MRL, use nn
6F00	I/O error

5.6 Update Record

CLA	00 - Clear mode 04 - SM mode
INS	DC
P1	Depends on P2, see below
P2	If 5 MSb <> 00000, it is SFI; 3 LSb: see below
P3	Number of Bytes to Write
Data	Bytes to Write

Last 3 bits of P2:
0: reference 1st record
1: reference last record
2: reference next record
3: reference previous record
4: reference record indexed by P1

If P2 = 4 then record indexing starts at 1.

If the file's MRL or NOR field is zero, the COS will return 6A83 status code (record not found).

For Linear EF, P3 can be < the file's MRL.

For Cyclic EF's, it is possible to update the Current Record (see 3.2.9) by having P2 = 4 and P1 = 0.

Return Status

6283	Current DF is blocked, or Target EF is blocked
6700	Incorrect P3
6981	Wrong file type; target file must be RECORD-BASED EF
6982	Target file's header block has wrong checksum, may be corrupted Security condition not satisfied
6986	No DF selected, or no EF selected
6A82	Referenced SFI not found
6A83	Record not found, invalid record reference
6B00	Invalid P1/P2 Invalid SFI SFI referencing is not allowed in SM mode
6Cnn	nn is the maximum valid value for P3; Wrong P3, P3 is greater than the file's MRL, use nn
6F00	I/O error

5.7 Append Record

CLA	00 - Clear Mode 04 - SM Mode
INS	E2
P1	00
P2	00
P3	Length of Data
Data	Data to be appended

This command applies only to Linear Variable EF. The new record will be written to the 1st record whose 1st byte is 0x00 (a record whose 1st byte is 0x00 is considered 'empty', and therefore is at the 'end' of the file). P3 can be less than the file's MRL, in such case; 0x00 will be padded to the new record.

Return Status

6283	Current DF is blocked, or Target EF is blocked
6700	Incorrect P3
6981	Wrong file type; target file must be RECORD-BASED EF
6982	Target file's header block has wrong checksum, may be corrupted Security condition not satisfied
6986	No DF selected, or no EF selected
6A82	Referenced SFI not found
6A83	Record not found, invalid record reference
6B00	Invalid P1/P2 Invalid SFI SFI referencing is not allowed in SM mode
6Cnn	nn is the maximum valid value for P3; Wrong P3, P3 is greater than the file's MRL, use nn
6A84	No empty record found
6F00	I/O error

5.8 Erase Binary

CLA	00 – Clear mode 04 – SM mode
INS	0E
P1	If MSb = 1, P1's 5 LSb holds SFI, else P1 holds high start offset
P2	Low start offset
P3	00 – erase up to end of file 02 – erase up to offset specified in Data
Data	End offset

For SM, SFI referencing is not allowed.

NOTE: To better control reader timing, only erase up to 1280 bytes of data.

Return Status

6282	Invalid offset
6283	Current DF is blocked or target EF is blocked
6700	Incorrect P3, can be 0 or 2 only
6981	Wrong file type; target file must be TRANSPARENT EF
6982	Target file's header block has wrong checksum, or security condition not satisfied
6986	No EF selected
6A82	Referenced SFI not found
6B00	Invalid P1/P2, invalid SFI Invalid P1/P2 offset SFI referencing not allowed in SM mode
6A80	End Offset is < Start Offset
6F00	I/O error

5.9 Activate File

CLA	00 - Clear Mode 04 - SM Mode
INS	44
P1	00
P2	00
P3	02 – Activate the File whose ID is referenced by the DATA 00 – Activate the currently selected EF or DF
Data	File ID

This command will activate the target file. Once activated, the file's security settings will take effect. In SM mode, P3 must be 0.

Return Status

6283	Current DF is blocked
6400	Target file is already terminated
6982	Target file has wrong checksum, may be corrupted Security condition not satisfied
6A82	File referenced not found
6986	MF does not exist, or no file is selected
6A86	Invalid P1 / P2, must be 0
6700	Invalid P3, can only be 0x00 or 0x02 P3 must be 0x00 in SM mode
6F00	EEPROM I/O failure

5.10 Deactivate File

CLA	00 - Clear Mode 04 - SM Mode
INS	04
P1	00
P2	00
P3	02 – Deactivate the File whose ID is referenced by the DATA 00 – Deactivate the currently selected EF or DF
Data	File ID

This command will invalidate or deactivate the target file. Once a file is deactivated, all commands (except ACTIVATE FILE) to the file will be rejected. In SM mode, P3 must be 0.

Return Status

6283	Current DF is blocked
6400	Target file is already terminated
6982	Target file has wrong checksum, may be corrupted Security condition not satisfied
6A82	File referenced not found
6986	MF does not exist, or no file is selected
6A86	Invalid P1 / P2, must be 0
6700	Invalid P3, can only be 0x00 or 0x02 P3 must be 0x00 in SM mode
6F00	EEPROM I/O failure

5.11 Terminate DF

CLA	00 - Clear Mode 04 - SM Mode
INS	E6
P1	00
P2	00
P3	02 – Terminate the DF File whose ID is referenced by the DATA 00 – Terminate the currently selected DF
Data	File ID

This command will irreversibly send the target DF to TERMINATED state. In such case, all commands to the file will be rejected.

Return Status

6283	Current DF is blocked
6400	Target file is already terminated
6982	Target file has wrong checksum, may be corrupted Security condition not satisfied
6A82	File referenced not found
6986	MF does not exist, or no file is selected
6A86	Invalid P1 / P2, must be 0
6700	Invalid P3, can only be 0x00 or 0x02 P3 must be 0x00 in SM mode
6F00	EEPROM I/O failure
6981	Target file is not DF

5.12 Terminate EF

CLA	00 - Clear Mode 04 - SM Mode
INS	E8
P1	00
P2	00
P3	02 – Terminate the EF File whose ID is referenced by the DATA 00 – Terminate the currently selected EF
Data	File ID

This command will irreversibly send the target DF to TERMINATED state. In such case, all commands to the file will be rejected.

Return Status

6283	Current DF is blocked
6400	Target file is already terminated
6982	Target file has wrong checksum, may be corrupted Security condition not satisfied
6A82	File referenced not found
6986	MF does not exist, or no file is selected
6A86	Invalid P1 / P2, must be 0
6700	Invalid P3, can only be 0x00 or 0x02 P3 must be 0x00 in SM mode
6F00	EEPROM I/O failure
6981	Target file is not EF

5.13 Delete File

CLA	00 – Clear Mode 04 - SM Mode
INS	E4
P1	00
P2	00
P3	00 – if file is already selected and data is absent, P3 = 02 if data is present
Data	File ID

Delete File will work if the target file is the last file created in EEPROM memory area. ACOS5 will not allow deletion of a DF that has children.

Return Status

6283	Current DF is blocked (deactivated or terminated)
6982	Target file has wrong checksum, may be corrupted Security condition not satisfied
6986	MF does not exist, or no file is selected
6A86	Invalid P1 / P2, must be 0
6700	Invalid P3, can only be 0x00
6F00	EEPROM I/O failure
6a80	Delete file: the DF to be deleted has children Delete file: the file to be deleted is not the last-positioned file in EEPROM

5.14 Get Card Info

CLA	80
INS	14
P1	See options below
P2	
P3	

This command returns card or file information of the ACOS5 card. The following card information is available depending on the parameters of the command.

P1	P2	P3	Description
00	00	06	Returns card's unique serial number
01	00	00	Returns the number of files under the currently selected DF in SW1 SW2 = 90XX, where XX is the number of files.
02	xx	08	Returns file information of the P2th file in the DF. The 8 bytes are: {FDB, DCB, FILE ID, FILE ID, SIZE or MRL, SIZE or NOR, SFI, LCSI};

If P1 = 02 then P2 starts at 00 to reference the 1st file, 01 to reference the 2nd file and so on.

Return Status

6700	Incorrect P3
6A80	Wrong P1/P2, data not available

5.15 Get Challenge

CLA	00
INS	84
P1	0
P2	0
P3	8

This command generates an 8-byte Random Challenge Data, to be used for authentication purposes.

Return Status

6700	Incorrect P3
6A86	Wrong P1 / P2, must be 0

5.16 Get Response

CLA	00
INS	C0
P1	0
P2	0
P3	Bytes to receive

This command returns information available in the card OS, with regards to the previous command.

Return Status

6b00	Wrong P1 / P2, must be 0
6Cnn	Incorrect P3, P3 must be nn
6a88	No data available

5.17 Verify

CLA	00
INS	20
P1	00
P2	PIN ID: if MSb = 1, use Local PIN, else use Global PIN
P3	Length of PIN
Data	PIN

This command is used to submit a PIN code to gain access rights. Access rights achieved will be invalidated when a new DF is selected. Command submission with P3=0 will return the remaining number of retries left for the PIN.

Return Status

6283	Current DF is blocked; EF1 is blocked
63Cn	If P3 == 0: N tries remaining If P3 <> 0: operation FAIL, n tries remaining
6700	Incorrect P3, does not match PIN length, must be <= 32
6981	EF1 has wrong FDB EF1 is not a valid PIN file
6983	Referenced PIN is locked
6986	No DF selected
6A83	PIN ID referenced in P2 is not found in EF1
6A86	Invalid P1/P2 Invalid P2, PIN index cannot be > 31 or == 0 P1 must be 00
6A88	EF1 not found
6984	VERIFY: PIN record is disabled, verify SUCCESS

5.18 Change Code

CLA	00
INS	24
P1	00 – data holds current and new PIN 01 – data holds new PIN, current PIN must be verified
P2	PIN ID: if MSb = 1, use Local PIN, else use Global PIN
P3	<= 20h
Data	Current PIN + New PIN or New PIN

This command allows you to change a PIN code in EF1.

Return Status

6283	Current DF is blocked; EF1 is blocked
6700	Incorrect P3, does not match PIN length, new PIN's length must be <= 32 and <= EF1's MRL
6981	EF1 has wrong FDB EF1 is not a valid PIN file
6983	Referenced PIN is locked
6986	No DF selected
6A83	PIN ID referenced in P2 is not found in EF1
6A86	Invalid P1/P2 Invalid P2, PIN index cannot be > 31 or == 0
6A88	EF1 not found
6984	VERIFY: PIN record is disabled, cannot proceed
6982	PIN not verified previously

5.19 Enable PIN Verification

CLA	00
INS	28
P1	00 – data holds current PIN 01 – data is empty, current PIN must be verified
P2	PIN ID: if MSb = 1, use Local PIN, else use Global PIN
P3	<= 20h
Data	PIN

Return Status

6283	Current DF is blocked; EF1 is blocked
6700	Incorrect P3, does not match PIN length, not consistent with P1
6981	EF1 has wrong FDB EF1 is not a valid PIN file
6983	Referenced PIN is locked
6986	No DF selected
6A83	PIN ID referenced in P2 is not found in EF1
6A86	Invalid P1/P2 Invalid P2, PIN index cannot be > 31 or == 0
6A88	EF1 not found
6984	PIN is already enabled
6982	PIN not verified previously

5.20 Disable PIN Verification

CLA	00
INS	26
P1	00 – data holds current PIN 01 – data is empty, current PIN must be verified
P2	PIN ID: if MSb = 1, use Local PIN, else use Global PIN
P3	<= 20h
Data	PIN

This command disables a PIN record. When a PIN is disabled, the user need not submit the correct PIN to achieve access rights related to the PIN.

Return Status

6283	Current DF is blocked; EF1 is blocked
6700	Incorrect P3, does not match PIN length, not consistent with P1
6981	EF1 has wrong FDB EF1 is not a valid PIN file
6983	Referenced PIN is locked
6986	No DF selected
6A83	PIN ID referenced in P2 is not found in EF1
6A86	Invalid P1/P2 Invalid P2, PIN index cannot be > 31 or == 0
6A88	EF1 not found
6984	PIN is already disabled
6982	PIN not verified previously

5.21 Reset PIN Verification

CLA	00
INS	28
P1	01
P2	PIN ID: if MSb = 1, use Local PIN, else use Global PIN
P3	01
Data	New Error Counter

This command modifies the error counter of a PIN record (please refer to section 3.3.1 for valid format). User is required to previously submit the PIN in question before changing its counter.

Return Status

6283	Current DF is blocked; EF1 is blocked
6700	Incorrect P3, must be 0
6981	EF1 has wrong FDB EF1 is not a valid PIN file
6983	Referenced PIN is locked
6986	No DF selected
6A83	PIN ID referenced in P2 is not found in EF1
6A86	Invalid P1/P2 Invalid P2, PIN index cannot be > 31 or == 0
6A88	EF1 not found
6984	PIN is disabled, cannot proceed
6982	PIN not verified previously

5.22 Internal Authentication

CLA	00
INS	88
P1	00
P2	KEY ID index: if MSb = 1, use Local KEY, else use Global KEY
P3	08
Data	Terminal Challenge Data

Return Status

6F00	Crypto Library not ready
6283	Current DF is blocked; EF2 is blocked
6700	Incorrect P3, must be 8
6981	EF2 has wrong FDB EF2 is not a valid KEY file
6983	Referenced Key's Usage Counter is 0
6986	No DF selected
6A83	KEY ID referenced in P2 is not found in EF2
6A86	Invalid P1/P2 Invalid P2, KEY index cannot be > 31 or == 0
6A88	EF2 not found
6984	KEY is disabled, cannot proceed
6A87	Referenced KEY is not capable on Internal authentication
6108	Authentication complete, issue get response to get Ri

5.23 External Authentication

CLA	00
INS	82
P1	00
P2	KEY ID index: if MSb = 1, use Local KEY, else use Global KEY
P3	08
Data	ENC (RNDc, Kt)

Return Status

6F00	Crypto Library not ready
6985	Card Challenge Data not ready, issue Get Challenge first
6283	Current DF is blocked; EF2 is blocked
6700	Incorrect P3, must be 8
6981	EF2 has wrong FDB EF2 is not a valid KEY file
6983	Referenced Key's Error Counter is locked
6986	No DF selected
6A83	KEY ID referenced in P2 is not found in EF2
6A86	Invalid P1/P2 Invalid P2, KEY index cannot be > 31 or == 0
6A88	EF2 not found
6984	KEY is disabled, cannot proceed
6A87	Referenced KEY is not capable on External authentication
63Cn	Wrong Crypto Data, Operation Fail, n tries left for KEY

5.24 Mutual Authentication

This command authenticates the referenced key of the currently selected DF. Both internal and external authentication processes are performed. Access rights are gained if successful.

CLA	00
INS	82
P1	00
P2	Key index; if MSb=1, use local EF2 else use global EF2
P3	10h
Data	Terminal Challenge Data + ENC (RNDc, Kt)

On success, COS will return 0x6108. The result is: ENC (RNDt. Kc). The key referenced in P2 must be capable of both Internal and External Authentication.

Return Status

6F00	Crypto library not ready
6A87	KEY referenced in P2 not capable of Internal and External authentication
63Cn	Wrong Crypto Data, Operation Fail, n tries left for KEY
6700	Incorrect P3, must be 16
6983	Referenced KEY's error counter is locked or Usage counter is 0
6985	GET CHALLENGE command not previously called
6A86	Invalid P1 / P2
6986	No DF selected
6283	Current DF is blocked EF2 is blocked
6A88	EF2 not found
6A83	Key not found in EF2, key has invalid length
6981	Invalid EF2 (FDB, MRL, etc not consistent)
6984	KEY is disabled, cannot proceed
6108	Authentication OK, issue GET RESPONSE to authenticate card key

5.25 Manage Security Environment

CLA	00 – Plain mode 0C – SM Mode
INS	22
P1	MSE sub command: X1 – MSE_SET F2 – MSE_STORE F3 – MSE_RESTORE F4 – MSE_ERASE
P2	Depends on P1 SET: CRT tag STORE / RESTORE / ERASE: target SE ID
P3	SET: Length of CRT STORE / RESTORE / ERASE: 0
Data	SET: CRT

MSE SET:

This command sets up the key locations and initialization vectors for different cryptographic computations in Perform Security Operation. The computations are set up using control reference templates (CRT) which can be Authentication Template (AT), Cryptographic Checksum Template (CCT), Hash Template (HT), Digital Signature Template (DST), and Confidentiality Template (CT-sym and CT-asym). These CRT's are accumulated in system memory and can be used in PSO command, or stored into the current SE file. To clear the accumulated CRT's, issue a SELECT FILE command.

Please refer to section 4.2 on what valid sub tags to set in this command.

MSE STORE:

The accumulated CRT's created by MSE SET will be written to the current SE file. ACOS5 will create a new SE record with ID specified in P2. If there is an existing SE ID, this command will fail. Make sure the "MSE Store" security access condition of the SE file is satisfied when you use this command (section 4.1.1).

MSE RESTORE:

Use this command to load an existing SE record to ACOS5 system memory. The SE record whose ID is specified in P2 will be placed in the ACOS5 system memory, just like in MSE SET. Make sure the "MSE Restore" security access condition of the SE file is satisfied when you use this command (section 4.1.1).

MSE ERASE:

Use this command to erase an existing SE record in the current SE file. The SE record whose ID is specified in P2 will be completely erased. Make sure the "MSE Erase" security access condition of the SE file is satisfied when you use this command (section 4.1.1).

Return Status

6283	Current DF is blocked; SE file is blocked
6700	Invalid P3
6A83	MSE_RESTORE: SE record referenced by P2 is not found MSE_ERASE: SE record referenced by P2 is not found
6982	Security condition not satisfied
6986	No DF currently selected
6A80	Invalid sub tags or length in input Duplicate sub-tags in input

	Asymmetric key EF referenced by tag 81 is invalid* Symmetric key EF referenced by tag 83 is invalid* Cannot have any 2 tags: 83 / 84 / 81 present Required tags for CRT is not present in input Cannot STORE empty CRT to SE MSE_STORE: SE record referenced by P2 already exists
6A84	Not enough system memory to store the templates Input size is > SE's MRL
6B00	Invalid P1 / P2: P1 can only be F2, F3, F4, X1 P2 can only be B8, b6, AA, B4, A4
6A82	SE file is not present, or is invalid

* The referenced Key File will be checked if:

- File is blocked
- File has wrong FDB
- File's size is not enough for key's fields (PUB, PRI, PRI CRT – 4, 8, 16)
- File's key pair is not present or invalid
- File's key pair have the same type
- File's key pair do not have matching length

5.26 Perform Security Operation

CLA	00 – Plain mode 10 – Plain Chaining Mode 0C – SM Mode 1C – SM Chaining Mode
INS	2A
P1	Please see below
P2	Please see below
P3	Please see below
Data	Please see below

PEFORM SECURITY OPERATION (PSO) command performs the following security operations:

- Computation of a cryptographic checksum
- Computation of a digital signature;
- Calculation of a hash-code;
- Verification of a cryptographic checksum;
- Verification of a digital signature;
- Encipherment using symmetric and asymmetric key / Public Key Exponentiation
- Decipherment using symmetric and asymmetric key / Private Key Exponentiation

The security operations are compliant to ISO7816 Part 8 and they are grouped into the instruction code 0x2A. The parameters P1 and P2 specify the output and input data objects respectively. The PSO command must be preceded by a MANAGE SECURITY ENVIRONMENT (MSE) command to correctly set-up the initialization values before execution.

			Output Tag	Input Tag	L	V
Description:	**CLA**	**INS**	**P1**	**P2**	**P3**	**Command Data**
Compute Cryptographic Checksum	10	2A	8E	80	08-F8	Plaintext data to MAC
	00	2A	8E	80	08-F8	Plaintext data to MAC
Verify Cryptographic Checksum	10	2A	00	A2	0A-FA	T: 80 L: 08-F8 V: Plaintext to be verified
	00	2A	00	A2		T: 80 L: 00-F0 V: Plaintext to be verified T: 8E L: 08 V: Candidate checksum
Compute Hash	10	2A	90	80	40, 80, C0	Plaintext data to hash
	00	2A	90	80	01-FF	Plaintext data to hash
Compute RSA Signature with PKCS#1	00	2A	9E	9A	14	Hashed data (SHA-1, 20 bytes)
	00	2A	9E	9A	0	Hashed data is already in the card by previous calls to Compute Hash
Recover RSA Signed DATA	10	2A	90	9E	80	PKCS#1 Signature (if using 2048 bit key)
	00	2A	90	9E	40, 80	PKCS#1 Signature
Symmetric Encipher	10	2A	84	80	08-F8	Plaintext data to encipher
	00	2A	84	80	08-F8	Plaintext data to encipher
Symmetric Decipher	10	2A	80	84	08-F8	Cryptogram to decipher
	00	2A	80	84	08-F8	Cryptogram to decipher
Asymmetric Public Exponentiation	10	2A	84	80	80	Data to encrypt / verify (Command chaining required if RSA key is 2048 bits)
	00	2A	84	80	40, 80	Data to encrypt / verify
Asymmetric Private Exponentiation	10	2A	80	84	80	Data to decrypt / sign (Command chaining required if RSA key is 2048 bits)
	00	2A	80	84	40, 80	Data to decrypt / sign

Compute Cryptographic Checksum

This operation initiates computation of DES / 3DES cryptographic checksum in CBC mode. It takes input messages with block size in multiples of 8 bytes and stores the resulting checksum until a GET RESPONSE command is called.
This operation requires MSE Command to initialize the key set and initialization vector to use in the Cryptographic Checksum Template (CCT). If the template is not set, the card will respond with the corresponding APDU error.
The command allows command-chaining control to compute cryptographic checksum of messages greater than 248 bytes. The CLA byte of 10 indicates that the command being sent is not the last. The CLA byte of 00 indicates that the command being sent is the only or the last of the message.
The length of the data field, P3, must be in multiples of 8 bytes. This is the block length of the input to the DES encryption in CBC mode. The response message of this command depends on the command chaining mode. If CLA is 0x10, a successful execution yields status word 90 00 to indicate the command expects more data. If CLA is 0x00, cryptographic checksum is complete and a subsequent call to GET RESPONSE will output the checksum.

Result Status

6F00	Crypto Library not ready
6986	No Currently Selected DF
6283	Current DF is blocked SE file is blocked
6a82	SE is not present, or invalid
6982	Security status does not satisfy “PSO” condition of the Key EF
6700	P3 cannot be 0 P3 must be a multiple of 8 and < 0xF8
6985	No template found in system memory; MSE not called previously
6a86	Invalid P1-P2
6a80	Qualifying CCT template not found in system memory Qualifying CCT template has invalid key reference (sub tag 83 or 84) Qualifying CCT template Algorithm Reference is not CBC Qualifying CCT template Algorithm Reference is not compatible with Referenced Key’s Algorithm
61xx	Operation completed, Result is ready

Verify Cryptographic Checksum

The VERIFY CRYPTOGRAPHIC CHECKSUM computes a cryptographic checksum then verifies it internally. The input is a data object consisting of an input data to compute cryptographic checksum, and the cryptographic checksum to verify. The function will return 9000 if success.
This operation requires MSE Command to initialize the key set and initialization vector to use in the Cryptographic Checksum Template (CCT). If the template is not set, the card will respond with the corresponding APDU error.
The data object of checksum verification contains the input plaintext data to compute cryptographic checksum and the cryptographic checksum to verify the input data against.
Command chaining must be used for computing cryptographic checksum verification for a plaintext data with more than 240 bytes. The CLA byte of 0x10 indicates that the command being sent is not the last. The command computes partial checksum which each chaining call. Therefore, they must be segmented at multiples of 8 bytes. The checksum verification data object in this case is as follows.

Tag	Length	Value
80	08 – F8 (in multiples of 8)	Input data for checksum computation

The CLA byte of 0x00 indicates that this is the last (or only) VERIFY CRYPTGRAPHIC CHECKSUM command. The data field contains the data object with the remaining (or all) input data to send for verification and checksum. The remaining input data can have a length of 0. In this case, the value field does not exist and the verification will be based on previously computed partial checksum.

Tag	Length	Value
80	00 – F0 (in multiples of 8)	Input data for checksum computation
8E	08	Candidate checksum

Result Status

6F00	Crypto Library not ready
6986	No Currently Selected DF
6283	Current DF is blocked SE file is blocked
6a82	SE is not present, or invalid
6982	Security status does not satisfy “PSO” condition of the Key EF
6700	P3 must be >= 0x0A and <= 0xFA
6985	No template found in system memory, MSE not called previously
6a86	Invalid P1-P2
6a80	Qualifying CCT template not found in system memory Qualifying CCT template has invalid key reference (sub tag 83, 84) Qualifying CCT template Algorithm Reference is not CBC Qualifying CCT template Algorithm Reference is not compatible with that of referenced Key Input must start with tag 80 or 8E, and length must be consistent with P3 Input tag 80 / 8E’s length must be multiple of 8
6988	Verification of cryptographic checksum fail
9000	Verification of cryptographic checksum success

Compute Hash

This operation computes a SHA-1 hash with input data. The operation assumes input data as little endian format in multiples of 8 bytes and outputs a 20-byte (160 bits) hash value also in little endian format. This operation must be preceded with a MSE Command to initialize the key reference in the Hash Template (HT). If the template is not set, the card will respond with the corresponding APDU error.

The command uses command chaining to compute a hash for data longer than 255 bytes. The CLA byte of 0x10 indicates that the command being sent is not the last and no data is to be returned. The CLA byte of 0x00 indicates that the command being sent is the only or the last set of data. A GET RESPONSE will output the 20-byte hash.

This command can be linked with COMPUTE RSA SIGNATURE by executing that command with length equals to 0 immediately after a proper completion of COMPUTE HASH. The GET RESPONSE command after the COMPUTE HASH command is optional if application does not require the hash code.

The successful execution will yield a response of 61 14 where 0x14 is the length of the 20-byte hash of the

SHA-1 algorithm.

Result Status

6F00	Crypto Library not ready
6986	No Currently Selected DF
6283	Current DF is blocked SE file is blocked
6a82	SE is not present, or invalid
6982	Security status does not satisfy “PSO” condition of the Key EF
6700	P3 must be > 0; for chaining, must be multiple of 0x40
6985	No template found in system memory; MSE not called previously
6a86	Invalid P1-P2
6a80	Qualifying HT template not found in system memory
6114	Hash result is ready via GET RESPONSE

Compute RSA Signature

The COMPUTE RSA SIGNATURE operation computes a digital signature from a RSA private key or a RSA CRT private key. The function will compute RSA signature with hash data computed on card or inserted with previously executed COMPUTE HASH command. The operation assumes that the input hash is in little endian format. The hash value will be padded according to PKCS #1 type 1 padding before it is encrypted.

This operation requires MSE Command to initialize the key set to use in the Digital Signature Template (DST). If the template is not set, the card will respond with the corresponding APDU error.

This command does not support command chaining, as there is no hash value currently that exceeds 255 bytes. The input hashed data can be hashed by the terminal or submitted from a previously called hash operation. In the case of hashing by the terminal, the Len must be non-zero and the data field contains the hash. In the case that the hash is submitted in a previous command, the COMPUTE RSA SIGNATURE is called immediately after a successful execution of COMPUTE HASH command.

The successful execution will yield a response of 61 nn where nn is the output of the RSA signature equal to the key length in bytes. In the case of 2048 bit keys, the status word of 61 00 indicating 256 bytes (2048 bits) output. An execution of GET RESPONSE with the proper P3 will output the result.

Result Status

6F00	Crypto Library not ready
6986	No Currently Selected DF
6283	Current DF is blocked SE file is blocked
6a82	SE is not present, or invalid
6982	Security status does not satisfy “PSO” condition of the Key EF
6700	P3 must be 20 or 0x00
6985	No template found in system memory, MSE not called previously
6a86	Invalid P1-P2
6a80	P3 is 0, but no HASH data is available

	Qualifying DST template not found in system memory Asymmetric keys referenced by DST is invalid Private key referenced by DST does not exist or is invalid Public key referenced by DST does not exist or is invalid Data to be signed is not compatible with Public Key's N
61xx	RSA signature complete, data ready

Recover RSA Signed Data

The RECOVER RSA SIGNED DATA operation computes the inverse of the COMPUTE RSA SIGNATURE command. The command takes as input an RSA signature in PKCS#1 format and outputs the hash with the PKCS #1 padding removed.

This operation does partial verification of the signature by the removal of PKCS #1 padding. If the command detects that the signature has invalid padding, it will output 6A80 – invalid data. This either indicates it is encrypted with the wrong key or it has been tempered. However, it does not authenticate the signature appendix to its message. To do this, the calling terminal must compute a hash of the original data and compare the output of this operation. This can be done on board with the COMPUTE HASH command if SHA-1 is used.

This operation requires MSE Command to initialize the key set to use in the Digital Signature Template (DST). If the template is not set, the card will respond with the corresponding APDU error.
The command utilizes command chaining in the case that the digital signature is signed with 2048 bit keys. In this case, the first call with CLA of 0x10 must have length 128 bytes. The second call must have CLA of 0x00 and length equal to the remaining 128 bytes. For all other key lengths, a single call to this command is sufficient.

The output length is based on the hashing algorithm. The successful execution will yield a response of 61 nn where nn is the hash. In the case of SHA-1 hash that is supported on board, the status word is 61 14 indicating 20 bytes output. An execution of GET RESPONSE with the proper P3 will output the result.

Result Status

6F00	Crypto Library not ready
6986	No Currently Selected DF
6283	Current DF is blocked SE file is blocked
6a82	SE is not present, or invalid
6982	Security status does not satisfy "PSO" condition of the Key EF
6700	P3 must be 0x40 or 0x80, 0x80 if chaining
6985	No template is found in system memory, MSE not called previously
6a86	Invalid P1-P2
6a80	If chaining, there should be no previous call to PSO_RECOVER Qualifying DST template not found in system memory Qualifying DST template Algorithm must be 0x10 Public key referenced by DST must exist and be valid
6114	RSA verify signature complete, data ready; use GET RESPONSE to retrieve the hashed data
6988	RSA signature verification fail

Symmetric Encipher

The SYMMETRIC ENCIPHER operation computes DES / 3DES / AES in ECB or CBC mode. The command takes blocks of data in multiples of 8 (for DES) or 16 (AES) and encrypts it.

This operation requires MSE Command to initialize the algorithm, its mode of operation, initialization vector and key set to use in the Symmetric Confidentiality Template (CT). If the template is not set, the card will respond with the corresponding APDU error.

The command utilizes command chaining to encrypt data in CBC mode longer than 248 bytes. Using command chaining, the command will retain the last 8-byte block to be used as the initialization vector for the next block. After execution of the last block with CLA byte of 0x00, the next call to SYMMETRIC ENCIPHER will begin with initialization vector defined in MSE. If encryption in ECB mode is used, it is not required to use command chaining, as each block computation does not depend on the last.

The successful execution will yield a response of 61 nn where nn is the encrypted data. Following each SYMMETRIC ENCIPHER command, a call to GET RESPONSE with the proper P3 will yield the enciphered text regardless of command chaining.

Result Status

6F00	Crypto Library not ready
6986	No Currently Selected DF
6283	Current DF is blocked SE file is blocked
6a82	SE is not present, or invalid
6982	Security status does not satisfy “PSO” condition of the Key EF
6700	P3 must be multiple of the Algorithm Reference length, and <= 0xF8
6985	No template found in system memory, MSE not called previously
6a86	Invalid P1-P2
6a80	Key referenced by qualifying CT template is invalid or not present
61xx	Operation complete, result available via GET RESPONSE

Symmetric Decipher

The SYMMETRIC DECIPHER operation computes the inverse of DES / 3DES / AES in ECB or CBC mode. The command takes blocks of data in multiples of 8 (for DES) or 16 (for AES) and decrypts it.

This operation requires MSE Command to initialize the algorithm, its mode of operation, initialization vector and key set to use in the Symmetric Confidentiality Template (CT). If the template is not set, the card will respond with the corresponding APDU error.

The command utilizes command chaining to decrypt data in CBC mode longer than 248 bytes. Using command chaining, the command will retain the last 8-byte input block to be used as the initialization vector for the next block. After execution of the last block with CLA byte of 0x00, the next call to SYMMETRIC DECIPHER will begin with initialization vector defined in MSE. If decryption in ECB mode is used, it is not required to use command chaining, as each block computation does not depend on the last.

The successful execution will yield a response of 61 nn where nn is the decrypted data. Following each SYMMETRIC DECIPHER command, a call to GET RESPONSE with the proper P3 will yield the deciphered text regardless of command chaining.

Result Status

6F00	Crypto Library not ready
6986	No Currently Selected DF
6283	Current DF is blocked SE file is blocked
6a82	SE is not present, or invalid
6982	Security status does not satisfy “PSO” condition of key EF
6700	P3 must be multiple of the Algorithm Reference length, and <= 0xF8
6985	No template found in system memory, MSE not called previously
6a86	Invalid P1-P2
6a80	Key referenced by CT template is invalid or not present
61xx	Operation complete, result available via GET RESPONSE

RSA Public Exponentiation / Asymmetric Encipher

This operation computes raw exponentiation of data using the public key. This command can be used as part of RSA encryption or verification where the system designer prefers to use padding method of their choice. The command requires an input data length that must be the same as the RSA key length being used. This operation requires MSE Command to set the algorithm type and location of the public key file using the Asymmetric Confidentiality Template (CT-asym). If the template is not set, the card will respond with the corresponding APDU error.

The command chaining is only used in the case of a 2048-bit key where the data must be in 256 bytes. In this case, the first call with CLA byte equal to 0x10 must have data length of 128 bytes. The second call must have CLA byte to be 0x00 and data length of 128 bytes.

The successful execution will yield a response of 61 nn where nn is the resulting data. A call to GET RESPONSE with the appropriate P3 will output the exponentiation.

Result Status

6F00	Crypto Library not ready
6986	No Currently Selected DF
6283	Current DF is blocked SE file is blocked
6a82	SE is not present, or invalid
6982	Security status does not satisfy “PSO” condition of key EF
6700	P3 must be 0x40 or 0x80, 0x80 if chaining
6985	No template found in system memory, MSE is not called previously
6a86	Invalid P1-P2
6a80	Key referenced by CT template is invalid or not present Qualifying CT template’s Algorithm Reference must be 0x12 or 0x13 Qualifying CT template must refer to a valid PUBLIC key If chaining, there should be no previous call to PSO ENCIPHER
61xx	Operation complete, result available via GET RESPONSE

RSA Private Exponentiation / Asymmetric Decipher

This operation computes raw exponentiation of data using the private key. This command can be used as part of RSA decryption or signature generation if the system designer prefers to use padding method of their choice. The command requires an input data length that must be the same as the RSA key length being used. This operation requires MSE Command to set the algorithm type and location of the private key file using the Asymmetric Confidentiality Template (CT asym). If the template is not set, the card will respond with the corresponding APDU error.

The command chaining is only used in the case of a 2048-bit key where the data must be in 256 bytes. In this case, the first call with CLA byte equal to 0x10 must have data length of 128 bytes. The second call must have CLA byte equals to 0x00 and data length of 128 bytes.

The successful execution will yield a response of 61 nn where nn is the resulting plain data. A call to GET RESPONSE with the appropriate P3 will output the result.

Result Status

6F00	Crypto Library not ready
6986	No Currently Selected DF

6283	Current DF is blocked SE file is blocked
6982	Security status does not satisfy “PSO” condition of key EF
6985	No template found in system memory, MSE is not called previously
6a82	SE is not present, or invalid
6700	P3 must be 0x40 or 0x80, 0x80 for chaining
6a86	Invalid P1-P2
6a80	Key referenced by CT template is invalid or not present Qualifying CT template’s Algorithm reference must be 0x12 or 0x13 Qualifying CT template must refer to a valid PRIVATE key If chaining, there should be no previous call to PSO_DECIPHER
61xx	Operation complete, result available via GET RESPONSE

5.27 Generate Key Pair

CLA	00 – Clear Mode 0C – SM Mode
INS	46
P1	80
P2	00
P3	09 – Length of Key + Prime Exponent 01 – Length of Key
Data	Length of Key – 1 byte: 4, 8, 16 Prime Exponent – 8 byte prime number in little endian

This command will generate a CRT Key Pair. Update access conditions of the Private and Public EF’s must be satisfied.

Return Status

6F00	Crypto Library not ready
6283	Current DF is blocked
6700	Wrong P3
6986	MF does not exists, no DF is selected
6A80	Wrong data input: Key length can only be: 04, 08, 0x10 Cannot find qualifying DST or CT-ASYM in system memory Missing or Invalid public KEY EF referenced by DST / CT-ASYM Missing or Invalid private KEY EF referenced by DST / CT-ASYM Public KEY EF’s PUTDATA access condition not satisfied Private KEY EF’s PUTDATA access condition not satisfied Public or private key EF is blocked
6B00	Wrong P1/P2

5.28 Put Data / Put Key

CLA	80 - last or non-chaining 90 - chaining
INS	DA
P1	High Offset – asymmetric key 01 – symmetric key
P2	Low Offset – asymmetric key Record number index – symmetric key
P3	Length of data to write
Data	Data to write

This command allows you to change the contents of an internal key EF. You will have to select a key EF first before issuing this command.

Chaining is necessary when setting up an asymmetric key EF whose length is > 255.

Return Status

6F00	Crypto Library not ready
6283	Current DF is blocked / currently selected key EF is blocked
6700	P3 must be > 0 P3 > the body size of the asymmetric EF P3 > the MRL of the symmetric EF
6981	Currently selected EF is not a key EF
6982	Key EF's PUT DATA condition not satisfied
6986	MF does not exist / no currently selected key EF
6A83	For symmetric key EF, P2 refers to an invalid record #
6A80	If EF is symmetric key, P3 is > than the EF's MRL If EF is symmetric key, its MRL is < than the bytes required to form a valid key structure
6282	For asymmetric key EF, P1-P2 forms an invalid offset, or P1-P2 + P3 exceeds file body size
6B00	Wrong P1 / P2: If current EF is a symmetric key, P1 must be 1 and P2 must be a record #
9000	Write to key EF success; for non-chaining, COS will test if the new data written is a valid ASYM key, the key EF's valid and complete flags will be set accordingly

5.29 Get Data / Get Key

CLA	80
INS	CA
P1	High Offset – asymmetric key 01 – symmetric key
P2	Low Offset – asymmetric key Record number index – symmetric key
P3	Length of data to read
Data	-

This command allows you to retrieve the contents of an internal key EF. You will have to select a key EF first before issuing this command.

Return Status

6F00	Crypto Library not ready
6283	Current DF is blocked / currently selected key EF is blocked
6700	P3 must be > 0 P3 > the body size of the asymmetric EF P3 > the MRL of the symmetric EF
6981	Currently selected EF is not a key EF
6982	Key EF's GET DATA condition not satisfied
6986	MF does not exist / no currently selected key EF
6A83	For symmetric key EF, P2 refers to an invalid record #
6A80	If EF is symmetric key, P3 is > than the EF's MRL If EF is symmetric key, its MRL is < than the bytes required to form a valid key structure
6282	For asymmetric key EF, P1-P2 forms an invalid offset, or P1-P2 + P3 exceeds file body size
6B00	Wrong P1 / P2: If current EF is a symmetric key, P1 must be 1 and P2 must be a record #

5.30 Set Baud Rate

CLA	80
INS	32
P1	00 if first time setting baud rate or Current TA1 value (11h or 18h)
P2	New TA1 value (11h or 18h)
P3	00
Data	-

This command will modify the card's ATR TA1 value to either 00h or 115200 bps or 9600 bps. Upon next reset, the ATR will be modified and communication baud rate will be negotiated between ACOS5 and the smart card reader.

Return Status

6B00	MF is not the current DF Incorrect P1 / P2
9000	New Baud Rate set, will take effect in next reset

6 GETTING STARTED: FILE CREATION

Below will demonstrate how to create a simple file system. Assuming the card still has no MF, and in Pre-Perso Stage.

Syntax for ISO-IN: CLA INS P1 P2 P3 Data (SW1 SW2)
Syntax for ISO-OUT: CLA INS P1 P2 P3 [Expected Data Result] (SW1 SW2)

; Create MF with DCB = 0xFF
00 E0 00 00 0A 62 08 82 02 3F FF 83 02 3F 00 (9000)

; Create DF under MF: ID=4000, DCB=0x00, with SAC and SAE, SE file is 4003
00 E0 00 00 34 62 32 82 01 38 83 02 40 00 84 10 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 8A 01 01 8C 08 7F 03 03 03 03 FF FF 03 AB 06 86 02 22 2A 97 00 8D 02 40 03 (9000)

; Under DF 4000, Create KEY FILE EF2: ID=4002, MRL=0x15, NOR=0x04 with SAC (read access = NEVER)
00 E0 00 00 1B 62 19 82 05 0C 01 00 15 04 83 02 40 02 88 01 02 8A 01 01 8C 06 6B 03 FF FF FF FF (9000)

; Initialize KEY Records in EF2, follow the KEY data structure
; All KEYS are initially set to 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00
; Keys 1, 2, 3 are external authenticate capable with counter = 0x55 (5 retries)
; while Key 4 is for internal authenticate with usage counter = 0xFFFF (unlimited)
00 DC 00 00 14 81 01 55 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00 (9000)
00 DC 00 02 14 82 01 55 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00 (9000)
00 DC 00 02 14 83 01 55 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00 (9000)
00 DC 00 02 15 84 02 FF FF 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00 (9000)

; CREATE SE File: ID=4003, MRL=0x11, NOR=0x04 with SAC (read access = NEVER)
00 E0 00 00 18 62 16 82 05 0C 01 00 11 04 83 02 40 03 8A 01 01 8C 06 6B 03 FF FF FF FF (9000)

; Initialize SE file, follow SE TEMPLATE STRUCTURE
; SE#1: external authentication of local key 1
00 DC 00 00 0B 80 01 01 A4 06 83 01 81 95 01 80 (9000)
;SE#2: external authentication of local key 2
00 DC 00 02 0B 80 01 02 A4 06 83 01 82 95 01 80 (9000)
;SE#3: external authentication of local key 3
00 DC 00 02 0B 80 01 03 A4 06 83 01 83 95 01 80 (9000)
;SE#4, external authentication of local keys 1 or 2 or 3
00 DC 00 02 11 80 01 04 A4 0C 83 01 81 83 01 82 83 01 83 95 01 80 (9000)

; Create Binary File: ID=4004, SIZE=0096, with SAC
00 E0 00 00 18 62 16 80 02 00 96 82 01 01 83 02 40 04 8A 01 01 8C 06 6E 03 FF FF FF FF (9000)

; Create Binary File: ID=4005, SIZE=0190, with SAC
00 E0 00 00 18 62 16 80 02 01 90 82 01 01 83 02 40 05 8A 01 01 8C 06 6E 03 FF FF FF 03 (9000)

; Create LF File: ID=4006, MRL=005C, NOR=0A, with SAC
00 E0 00 00 19 62 17 82 05 02 41 00 5C 0A 83 02 40 06 8A 01 01 8C 07 6F 03 FF FF 02 03 04 (9000)

; Create LF FILE: ID=4007, MRL=0024, NOR=0A, with SAC
00 E0 00 00 19 62 17 82 05 02 41 00 24 0A 83 02 40 07 8A 01 01 8C 07 6F 03 FF FF 01 03 04 (9000)

; DF 4000 is complete
; Go back to MF, expect SW1SW2 to be 61nn
00 A4 00 00 00 (61XX)

; Create next DF: 4100, SE file=4103, with SAC and SAE
00 E0 00 00 43 62 41 82 01 38 83 02 41 00 84 10 41 00 00 00 00 00 00 00 00 00 00 00 00 00 00 8A 01 01 8C 07 7D 02 02 02 FF FF 02 AB 16 86 02 22 F2 97 00 84 01 22 A0 0B 9E 01 01 A4 06 83 01 81 95 01 08 8D 02 41 03 (9000)

; Create PIN FILE EF1 4101: MRL=12h NOR=1, SFI=1
00 E0 00 00 1B 62 19 82 05 0C 01 00 12 01 83 02 41 01 88 01 01 8A 01 01 8C 06 6B FF FF FF FF FF (9000)

; Initialize PINs to EF1
; PIN 1: 4 retries left, 4 max retries, PIN = 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00
00 E2 00 00 12 81 44 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00 (9000)

; Create KEY FILE EF2 4102, MRL=16, NOR=6, SFI=2
00 E0 00 00 1C 62 1A 82 05 0C 41 00 16 06 83 02 41 02 88 01 02 8A 01 01 8C 07 6F FF FF FF 02 FF FF (9000)

; Initialize KEY RECORDS TO EF2
; All keys are initially set to: 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00
; KEY 1, internal auth., usage counter=0x1234
00 DC 00 00 15 81 02 12 34 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00 (9000)
; KEY 2, internal auth., usage counter=0x0000
00 DC 00 02 15 82 02 00 00 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00 (9000)
; KEY 3, internal and external auth., usage counter=0x1234, retry counter=0x33 (3 tries)
00 DC 00 02 16 83 03 12 34 33 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00 (9000)
; KEY 4, external auth., retry counter=0xFF (unlimited)
00 DC 00 02 14 84 01 FF 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00 (9000)
; KEY 5, internal auth., usage counter=0xFF00
00 DC 00 02 15 85 02 FF 00 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00 (9000)
; KEY 6, external auth., retry counter=0x88
00 DC 00 02 14 86 01 88 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00 (9000)

; Create SE FILE 4103, MRL=27, NOR=05, SFI=3
00 E0 00 00 1B 62 19 82 05 0C 01 00 27 05 83 02 41 03 88 01 03 8A 01 01 8C 06 6B FF FF FF FF FF (9000)

; Initialize RECORD to SE
; SE#1: authenticate local key 3
00 DC 00 00 0B 80 01 01 A4 06 83 01 83 95 01 80 (9000)
; SE#2: authenticate local key 4
00 DC 00 02 0B 80 01 02 A4 06 83 01 84 95 01 80 (9000)
; SE#5; authenticate local key 86
00 DC 00 02 0B 80 01 05 A4 06 83 01 86 95 01 80 (9000)

; Create Transparent File 4104: size=0030, SFI=4
00 E0 00 00 1C 62 1A 80 02 00 30 82 01 01 83 02 41 04 88 01 04 8A 01 01 8C 07 6F FF FF FF FF FF 01 (9000)

; Create Transparent File 4105, size=0064, SFI=5
00 E0 00 00 1B 62 19 80 02 00 64 82 01 01 83 02 41 05 88 01 05 8A 01 01 8C 06 6E FF FF FF FF 02 (9000)

; Go back to MF

00 A4 00 00 00 (61XX)

; Create DATA FILE 4305, SFI=5
00 E0 00 00 1B 62 19 80 02 08 00 82 01 01 83 02 43 05 88 01 05 8A 01 01 8C 06 6E FF FF FF 01 01 (9000)

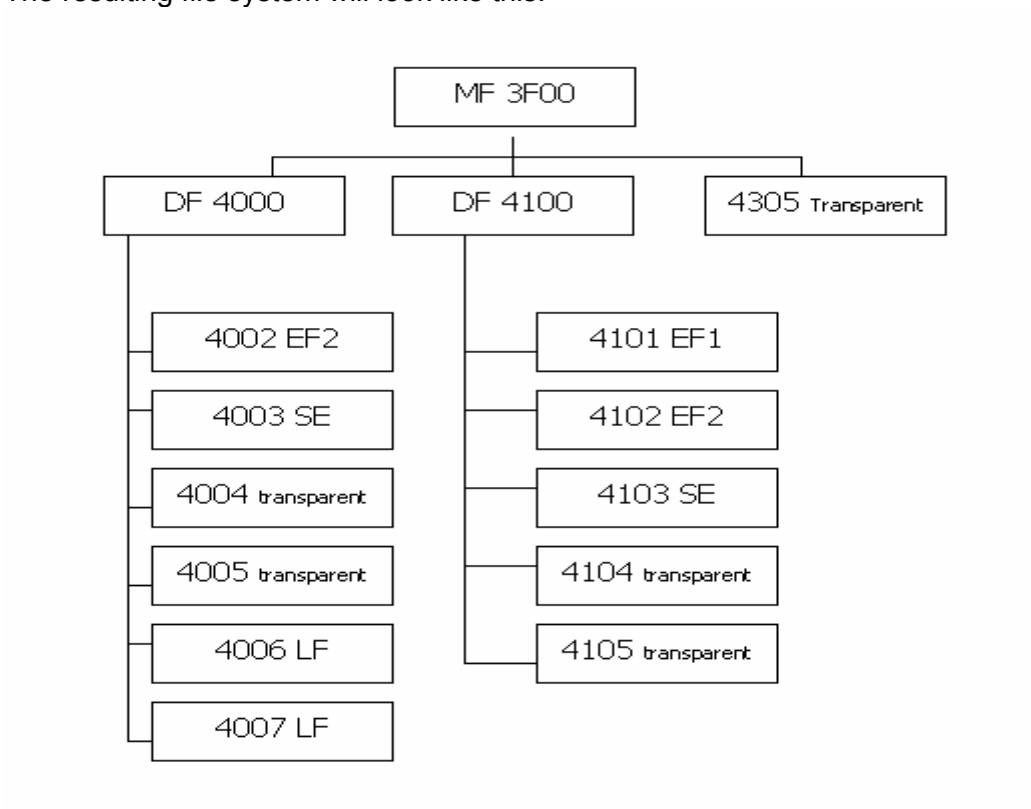
; Read 0x10 bytes, starting from offset 0000
00 B0 00 00 10 [00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00] (9000)

; Update binary at offset 0x00, 0x20 bytes
00 D6 00 00 20 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00 (9000)

; Update binary at offset 0x0100, 9 bytes
00 D6 01 00 09 11 22 33 44 55 66 77 88 99 (9000)
00 D6 07 80 08 11 22 33 44 55 66 77 88 (9000)
00 D6 07 F8 08 88 77 66 55 44 33 22 11 (9000)

; Read binary, check if the data written are correct
00 B0 00 00 20 [11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00] (9000)
00 B0 01 00 09 [11 22 33 44 55 66 77 88 99] (9000)
00 B0 07 F8 08 [88 77 66 55 44 33 22 11] (9000)

The resulting file system will look like this:



More Examples:

Cyclic File behavior

```
; Creating a Cyclic File with file ID=EF09, MRL=0A, NOR=03, R/W access allowed if SE#1 is satisfied
00 E0 00 00 12 62 10 83 02 EF 09 82 05 06 00 00 0A 03 8C 03 03 81 81 (9000)
; If we write 3 records
00 DC 00 00 0A 11 11 11 11 11 11 11 11 11 11 (9000)
00 DC 00 02 0A 22 22 22 22 22 22 22 22 22 22 (9000)
00 DC 00 02 0A 33 33 33 33 33 33 33 33 33 33 (9000)
; the 1st record is the last record written
00 B2 00 00 0A [33 33 33 33 33 33 33 33 33 33] (9000)
; reading the next record will wrap to file forward
00 B2 00 02 0A [11 11 11 11 11 11 11 11 11 11] (9000)
; reading the previous record will wrap the file back
00 B2 00 03 0A [33 33 33 33 33 33 33 33 33 33] (9000)
00 B2 00 03 0A [22 22 22 22 22 22 22 22 22 22] (9000)
```

Linear Variable behavior

```
; Create LV EF0A, MRL=10, NOR=01, R/W access allowed if SE#1 is satisfied
00 E0 00 00 12 62 10 83 02 EF 0A 82 05 04 00 00 0A 01 8c 03 03 81 81 (9000)
; Write and read the record
00 DC 00 00 0A AA AA AA AA AA AA AA AA AA AA (9000)
00 B2 00 00 0A [AA AA AA AA AA AA AA AA AA AA] (9000)
; Write again
00 DC 00 00 03 11 22 33 (9000)
; Read back the whole record. Unlike LF file, the whole record is erased first before writing 11 22 33
00 B2 00 00 0A [11 22 33 00 00 00 00 00 00 00] (9000)
```

More Examples:

Perform Security Operations

```
; First we need to create a 128-byte private and public key pair, they must be big enough to store the fields defined in
section 3.3.4
; Create PUB EF, ID=BBBB, size=0200
00 E0 00 00 17 62 15 83 02 BB BB 82 01 09 80 02 02 00 8C 08 7F FF FF FF FF FF FF FF (9000)

; Create a PRIVATE EF, ID=CCCC, size=0290
00 E0 00 00 17 62 15 83 02 CC CC 82 01 09 80 02 02 90 8C 08 7F FF FF FF FF FF FF FF (9000)
; To generate RSA key pair, build MSE with 2 DST's
00 22 01 B6 0A 80 01 10 81 02 BB BB 95 01 80 (9000)
00 22 01 B6 0A 80 01 10 81 02 CC CC 95 01 40 (9000)
; Generate RSA key pair with len=8, and e = 0000000000000003
00 46 80 00 09 08 03 00 00 00 00 00 00 00 (9000)
; Now RSA sign and verify sign
; Clear the MSE string
00 a4 00 00 02 3F FF (61xx)
; Set up a DST that can sign
00 22 01 B6 0A 95 01 FF 81 02 BB BB 80 01 10 (9000)
; Now sign 20 bytes, expect a 128 byte result
00 2A 9E 9A 14 45 D3 0D 01 73 E9 96 F2 0F 63 C1 C4 F4 CF 80 A5 62 67 C1 53 (6180)
00 C0 00 00 80 [YY YY YY YY … YY YY] (9000)
; Now verify the signature, check if the result is the same as our original data
00 2A 90 9E 80 YY YY YY….YY YY (6114)
00 C0 00 00 14 [45 D3 0D 01 73 E9 96 F2 0F 63 C1 C4 F4 CF 80 A5 62 67 C1 53] (9000)
```

7 GENERAL STATUS CODE

SW1 SW2 Listing

9000	Command OK
61nn	Command OK, issue GET RESPONSE with P3=nn to retrieve data
6E00	Invalid CLA
6D00	Invalid INS, or unsupported INS
6700	Wrong P3
6982	Security condition not satisfied
6884	Wrong MAC is submitted In Secure Messaging
6985	MAC Computation error in Secure Messaging (Random number or Session Key not ready)