SOFTWARE SPECIFICATION	Coupler Software Interface

ASK CSC - COUPLER SOFTWARE INTERFACE

DATE D'APPLICATION / APPLICATION DATE
NB : Restricted Access Document

DOCUMENT WRITTEN BY

Département Department	Nom Name	Fonction Function	Date	Visa
R&D	G. Brandin S. Manigault	Software engineer Software engineer	2009-06	

DOCUMENT APPROUVE PAR / APPROVED BY

Département Department	Nom Name	Fonction Function	Date	Visa
R&D	M. Reichert	Manager	2009-06	

MODIFICATION SHEET

DATE OF UPDATE	MODIFIED PART (section & page)	NEW REVISION INDEX	MODIFICATIONS Type (adding, canceling) and Description
2006/01/03	-	1.0	Initial Project from RD-ST-03317-53 CSC GEN 3XX
2006/03/28		1.1	*Adding new commands F10_0E_ SimpleWriteBlock, F10_0F_ReadSectorData() and F10_10_WriteSectorData() *Adding new commands for CD97 F05_24_Increase_Lg() and F05_23_Decrease_Lg() *Adding new command F06_29_UpdateFieldON() *Miscellaneous enhancements.
2007/01/11	p11 p13 p23,24 p24 p38	1.2	* CMD and STA byte : extended mode added * Frame length : extended mode added * Eeprom configuration : fields added * Read eeprom configuration : status and value order switched * Switch signals : led 4, SIGN_ANT_1 & SIGN_ANT_2 added
2007/03/30	p23,24	1.3	* Eeprom configuration : fields added.
2007/05/25		1.4	* Adding status information in Enter Hunt Phase command * Adding data returned by F05_0E_GetElectronicPurseStatus() * Adding information in F05_02_ChangePIN() et F03_02_ChangePIN() * Adding information in F05_0D_Purchase(), F03_0D_Purchase() and F05_0F_ReloadElectronicPurse(),F03_0F_ReloadElectronicPurse() commands * Adding ST SR/SRI/SRT/SRIX class.
2007/07/04	25-27 37 75 80	1.5 For CSC GEN4XX V1.12	* EHP (01_03) : note on bi-protocol cards. * EHP (01_03) : Calypso ISO A type = 0x0C * EHP (01_03) : Calypso application name optionaly returned * EHP parameters (01_17): options added * Select ISO Application command added (03_15) * Change PIN : CD Light key number added.
2007/12	23-24 27,29 40	1.6 For CSC GEN4XX V1.13	* Read/Write EEPROM configuration : - Slot 4 handling (CAM or SAM) - Strict ISO14443-3B timeout - Strict ISO14443-4B timeout - Delay after REQ/Select * EHP (01_03): SRI detection. * RDR37/447 buzzer management (01_23 command)
2008/07	22-23 38	1.7 For CSC GEN4XX V1.13c	* Read/Write EEPROM configuration : - MUX482 J1/J3 selection are the same as MUX382 - Unconditional Mifare selection before authentication * Switch Signals
2008/09	22-23	1.8 GEN4XX V1.14	* Read/Write EEPROM configuration : - Custom Frame Waiting Time - ISO14443-4 retries on PICC timeout

2009/05	-	1.9 GEN4XX V1.15	* Universal Transit class added ($08).
2009/06	-	2.0 GEN4XX V1.16	* Universal Transit class ($08): updated
2012/04	-	2.1 GEN4XX V1.16	* Universal Transit class ($08): Some corrections

INDEX

Introduction

The GEN4XX reader is a coupler compliant with the ISO 14443 A/B and ISO 15693 norms. It is capable of handling all the existing cards in the ASK range (CD97, GTML, CT2000, CMC 15693, contactless tickets CTS, CTM and CLAB 15693, ePassport and eID) and all ISO 14443-4 A/B and ISO 15693 compliant cards and tickets (see below for full list of supported cards) .

It supports both types of modulation specified he norm (A and B) and is able to support the ISO14443/3 and 4 communication protocols.

This coupler is made up of a software module (also called CSC) which handles the low layers of the card communication (contact and contactless) and SAM, as well as an application layer dealing with successions of card/SAM commands enabling rapid transactions without contact.

The user of the GEN4XX coupler may also implement the different supported cards/ticket, through high-level commands, without bothering about the detail of the set of card/SAM commands.

The GEN4XX coupler has been designed to keep maximum compatibility with the GEN3XX coupler.

The main differences therefore concern:

- RF treatment
- Baud rate

The aim of this document is to present the software interface of the coupler:

- General description of operation
- Communication Protocol
- Description of communication scenarios
- Details of commands

Applications

- transport ticketing
- payment
- access control
- ePaper

Functions supported

- System functions, for example configuration of the coupler
- Software download functions
- Card functions, specific to each type of card handled

Cards supported

- CD97 RJJ and RJL masks
- GTML, GTML2, CT2000
- CTS 256 and 512B Ticket
- CTM 521B Ticket
- CLAB/CMC 15693
- Mifare® Standard 1K, Mifare® 4K, Mifare® UltraLight
- Mifare® ProX
- Philips SmartMX and DESFire
- MV4000 and MV5000
- ePassport and eID
- ST ST19WR66
- ST ST19XR34
- Sharp
- Atmel AT90SC
- Any ISO 14443-4 A/B compliant card and ticket
- Any ISO 15693 / iCode SLI compliant card and ticket
- ST SR Family

For any further information or remarks please contact us:
By mail at:
ASK SA
2260, route des Crêtes - BP 337
06906 Sophia-Antipolis Cedex
Tel. 04 97 21 40 18 - Fax. 04 92 38 93 21 Web : www.ask-rfid.com

Or by E-mail at:
support@ask.fr

1. GENERAL OPERATION

1.1. Introduction

The GEN4XX reader communicates through a serial or a parallel link and behaves as slave during all exchanges it may have with the host (the latter being an application card, computer, etc.).

1.2. Description of interface

The communication interface with the host computer is USB, RS232C, TTL serial or parallel.

The physical characteristics of this interface, as well as the mode of selection are described in the hardware specification documents for GEN4XX couplers.

The rate of serial interface is programmable. The default value retained is 115 200 bauds.

This default value may be changed by the user using a system command.

Rate is programmable between 9600 baud and 691200 baud's.

- The link format in transmission and reception is: 8 bits, no parity, 1 stop bit

Remark on the CSC Parallel interface:
The concept of this Parallel interface is to use only with a motherboard in TTL compatibility, if connected to a PC parallel port it should communicate or not depending on the electrical characteristics of the chip used on the PC (electrical level).

2. COMMUNICATION PROTOCOL

2.1. Description

The reader behaves as slave, thus waits for a command coming from the host (master), carries out this command and sends the host the response to this command.

As soon as the reader is switched on, the host must before anything else, carry out a zero reset of the reader.

After this RESET of the reader, the host may send all the commands necessary to run the application.

2.2. Reader RESET Procedure

The reader is re-initialized at each power-up.

The first command after initialization must be the SoftwareVersion Command which allows the use of other commands.

2.3. Frame format

There are two types of message: 'pure command' messages (reduced to CMD byte) and the messages containing an application command (system and card commands).

2.3.1.Command frames

CMD (1 byte)	See 'Detail of CMD and STA registers'
LNG (1 or 2 byte(s))	See 'Frame length'
DATA (x byte)	See 'Details of reader functions' See 'Software interface'
End of frame = $00	Fixed value 0x00
CRC (2 bytes) CRCL CRCH	CRC CCITT of all the previous bytes See 'Calculation of CRC'

The host sends this frame, and the bytes may be sent one after another without interruption. The maximum time between each byte sent by the host is checked by timer (around 1500 ms).

The 'LNG' value is the length of the 'DATA' block in the frame.

The 'End of frame' byte is a byte which means to the reader that the 'DATA' block is finished and that the two following blocks are the CRC.

The protocol does not include a mechanism of resumption on communication error.

2.3.2.Response frame

STA (1 byte)	See 'Detail of CMD and STA registers'
LNG (1 or 2 byte(s))	See 'Frame length'
DATA (x byte)	See 'Details of reader functions' See 'Software interface'
End of frame = $00	Fixed value 0x00
CRC (2 bytes) CRCL CRCH	CRC CCITT of all the previous bytes See 'Calculation of CRC'

This frame is sent by the reader, in response to a command sent by the host. All the bytes are sent one after the other without interruption.

The 'LNG' value is the length of the 'DATA' block in the frame.

The 'End of frame' byte is a byte which means to the reader that the 'DATA' block is finished and that the two following blocks are the CRC.

2.3.3.Description of pure commands

CMD (1 byte)	See 'Detail of CMD and STS bytes'

It concerns commands reduced to CMD and whose bit 7 (EXEC) is at zero.

CMD = STOP that is to say $02

CMD = RST that is to say $01

Only this byte is sent by the host.

(Remark: the only command that may be interrupted by stop is Enter_HuntPhase)

2.3.4.Description of responses to pure commands

STA (1 byte)	See 'Detail of CMD and STA bytes'

It concerns the response to a pure command; this response is reduced to a single 'STA' byte

After a pure command CMD = STOP

STA = ABORT that is to say $04

After a pure command CMD = RST

STA = RES that is to say $10

2.4. Details of CMD and STA bytes

	7	6						0
CMD	**EXEC**	**EXT**					**STOP**	**RES**

Note: The bits are active at 1 and the unused bits must be fixed at 0. The bits EXEC, STOP and RES are Exclusive Operation and may not be used simultaneously.

RES... **Reset**
Complete Initialization of CSC.

STOP... **Stop**
Cancels current command.
This command only acts on interrupting a current Enter Hunt Phase command and in this case the reader sends back "STA = ABORT"; otherwise the reader is mute.

EXEC... **Execute**
Executes the command transmitted by the host into the following bytes.

EXT... **Extended**
CMD byte is followed by LNG LOW and LNG HIGH.

	7	6						0
STA	**ERR**	**EXT**		**RES**		**ABORT**		**DATA**

Note: The bits are active at 1.

DATA... **Data available**
Data is transmitted following a command received.

ABORT... **Abort execution**
The interruptible current command was stopped.

RES... **Reset**
The CSC was re-initialized by a physical or logical RESET (CMD.RES).
Indicates in addition that the FPGA has been correctly loaded.

ERR... **Error**
The command syntax is erroneous (class, function unknown).

EXT... **Extended**
CMD byte is followed by LNG LOW and LNG HIGH.

2.5. CRC Calculation

The CRC concerns all characters in the frame, including CMD and STA and 0. It is the CRC on 16 bits defined by the norms ISO 3309, CCITT V42 and CCITT X25. A portable implementation in C, usable without modification on PC for calculation and verification of CRC, is available (ref. CSC_ORD.C).

Examples: This function enables calculation of the CRC of bytes in the frame which are in the buffer of non-signed characters 'FRAME'. The length of bytes in this frame are found in 'LNG'

```
int LNG;nsigned char TRAME[256]oid SetCRC(void)
{
unsigned short CRCVal=0; int i ;
for( i = 0 ; i < LNG ; i++ )
  CRCVal = TABLE[( CRCVal ^ = TRAME[ i ] ) & 0xFF ] ^ ( CRCVal >> 8 );
TRAME [ LNG  ] = CRCVal % 256;
TRAME [ LNG + 1 ] = CRCVal / 256;
LNG = LNG + 2;
}
```

In 'TABLE' may be found the calculation constants of the CRC

```
const unsigned short TABLE[ 256 ]={
   0xF078,0xE1F1,0xD36A,0xC2E3,0xB65C,0xA7D5,0x954E,0x84C7,0x7C30,0x6DB9,0x5F22,0x4EAB,
   0x3A14,0x2B9D,0x1906,0x088F,0xE0F9,0xF170,0xC3EB,0xD262,0xA6DD,0xB754,0x85CF,0x9446,
   0x6CB1,0x7D38,0x4FA3,0x5E2A,0x2A95,0x3B1C,0x0987,0x180E,0xD17A,0xC0F3,0xF268,0xE3E1,
   0x975E,0x86D7,0xB44C,0xA5C5,0x5D32,0x4CBB,0x7E20,0x6FA9,0x1B16,0x0A9F,0x3804,0x298D,
   0xC1FB,0xD072,0xE2E9,0xF360,0x87DF,0x9656,0xA4CD,0xB544,0x4DB3,0x5C3A,0x6EA1,0x7F28,
   0x0B97,0x1A1E,0x2885,0x390C,0xB27C,0xA3F5,0x916E,0x80E7,0xF458,0xE5D1,0xD74A,0xC6C3,
   0x3E34,0x2FBD,0x1D26,0x0CAF,0x7810,0x6999,0x5B02,0x4A8B,0xA2FD,0xB374,0x81EF,0x9066,
   0xE4D9,0xF550,0xC7CB,0xD642,0x2EB5,0x3F3C,0x0DA7,0x1C2E,0x6891,0x7918,0x4B83,0x5A0A,
   0x937E,0x82F7,0xB06C,0xA1E5,0xD55A,0xC4D3,0xF648,0xE7C1,0x1F36,0x0EBF,0x3C24,0x2DAD,
   0x5912,0x489B,0x7A00,0x6B89,0x83FF,0x9276,0xA0ED,0xB164,0xC5DB,0xD452,0xE6C9,0xF740,
   0x0FB7,0x1E3E,0x2CA5,0x3D2C,0x4993,0x581A,0x6A81,0x7B08,0x7470,0x65F9,0x5762,0x46EB,
   0x3254,0x23DD,0x1146,0x00CF,0xF838,0xE9B1,0xDB2A,0xCAA3,0xBE1C,0xAF95,0x9D0E,0x8C87,
   0x64F1,0x7578,0x47E3,0x566A,0x22D5,0x335C,0x01C7,0x104E,0xE8B9,0xF930,0xCBAB,0xDA22,
   0xAE9D,0xBF14,0x8D8F,0x9C06,0x5572,0x44FB,0x7660,0x67E9,0x1356,0x02DF,0x3044,0x21CD,
   0xD93A,0xC8B3,0xFA28,0xEBA1,0x9F1E,0x8E97,0xBC0C,0xAD85,0x45F3,0x547A,0x66E1,0x7768,
   0x03D7,0x125E,0x20C5,0x314C,0xC9BB,0xD832,0xEAA9,0xFB20,0x8F9F,0x9E16,0xAC8D,0xBD04,
   0x3674,0x27FD,0x1566,0x04EF,0x7050,0x61D9,0x5342,0x42CB,0xBA3C,0xABB5,0x992E,0x88A7,
   0xFC18,0xED91,0xDF0A,0xCE83,0x26F5,0x377C,0x05E7,0x146E,0x60D1,0x7158,0x43C3,0x524A,
   0xAABD,0xBB34,0x89AF,0x9826,0xEC99,0xFD10,0xCF8B,0xDE02,0x1776,0x06FF,0x3464,0x25ED,
   0x5152,0x40DB,0x7240,0x63C9,0x9B3E,0x8AB7,0xB82C,0xA9A5,0xDD1A,0xCC93,0xFE08,0xEF81,
   0x07F7,0x167E,0x24E5,0x356C,0x41D3,0x505A,0x62C1,0x7348,0x8BBF,0x9A36,0xA8AD,0xB924,
   0xCD9B,0xDC12,0xEE89,0xFF00};
```

2.5.1. CRC format

The frames only have a CRC if they are not reduced to a single byte (pure command).he first byte after the 'End of frame' byte is the lower part of CRC (LSB)The last byte in the frame is the upper part of the CRC (MSB).

2.6. Frame length

Normal mode (EXT bit =0 in CMD or STA byte)

The maximum number of bytes transmitted with the card is a bit more than 256. That's why the length indicator 'LEN' may exceed the 0xFF hexadecimal value. In the case of a length greater than or equal to 255, the length will be encoded on two bytes : the first one LEN_1 will be set to 0xFF, and the second one LEN_2 will represent the rest until the total length, so as LEN will be equal to LEN_1 + LEN_2. Hence, until 0xFE, LEN will remain on one byte, and beyond this value will be on two bytes: 254 → 0xFE

255 → 0xFF 0x00

256 → 0xFF 0x01, and so on…

Extended mode (EXT bit =1 in CMD or STA byte)

The frame length is 2 byte long : LNG LOW and LNG HIGH.

The effective frame length is LNG LOW + (256 * LNG HIGH)

The maximum frame length is 800 bytes.

3. DESCRIPTION OF COMMUNICATION SCENARIOS

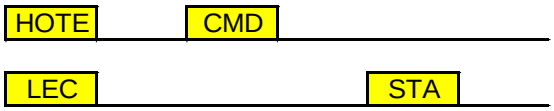
3.1. Restriction

The computer should specify a maximum time for commands that require it (3 seconds for example), and see to only interrupting polling through the STOP command, the bytes of any other command being effectively lost.

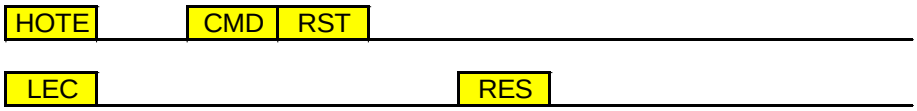
3.2. Frames

3.2.1. Pure command

This exchange is carried out on a RESET or on a STOP command, the host sends a single byte.
(Remark: the only command that may be interrupted by stop is Enter_HuntPhase)
The reader sends back a single byte of 'STA' status in the event of command acknowledgement.

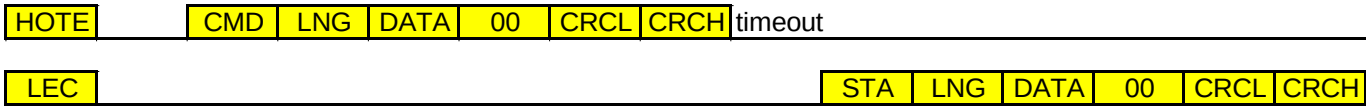


In the event that the reader does not recognize the command before xx ms (xx depends on estimated time of command execution) , the host can carry out a RESET of the reader.



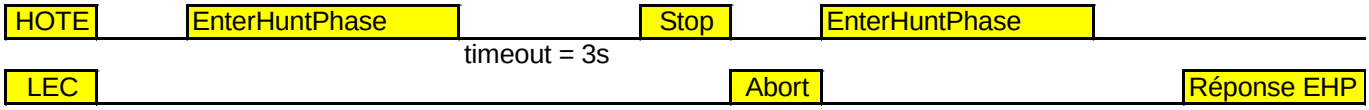
3.2.2. Standard command

The timeout between issuing the command and the reader's response depends on the command sent.



The old style **EnterHuntPhase** commands do not call for responses (search for badge/card) if no card is detected.

Anyway, it is possible to use the parameter "timeout" of the command and use it periodically. However, for compatibility reasons the STOP command is kept.



Stop : a pure command CMD = STOP ($02)

Abort : the response to this pure command STA = ABORT ($04)

EnterHuntPhase : issue of CMD+LNG+DATA+ $00 + CRC

That is to say $80 $07 $01 $03 $00 $00 $00 $00 $01 $00 $65 $18 (See 'Details of reader functions')

EHP answer : the response to EnterHuntPhase if a card is found by the antenna.

4. DESCRIPTION OF THE CONTACT CARD INTERFACE

4.1. Description

Remark on the Contact Interface:
The Contact interface is a set of 4 slots which can be configured in the Innovatron SAM Speed protocol (up to 423 750 bauds) or the ISO 7816 9600 Bauds.

When the Contact interface is selected in the EnterHuntPhase parameters, if a device is found in the fourth Slot it is considered as a contact card and it is used in the ISO7816 protocol.

4.2. Restriction

The Limitation on the implementation of the 7816 protocol is as follow:
- No implementation of the UART repeat procedure
- Only 5 Volt interface restriction : Type A or A&B (no support of Type B card)
- VPP is not connected (thus if such a card is inserted in the slot, a warning will occurs in the status of the connection)
- No clock suspends procedure implemented.
- PPS is not supported.
- Limitation to 4 levels of interface character in the ATR response interpretation
- Only T=0 protocol implemented
- Specific Fixed Guard Time implementation != TC(1) (1 second)
- No management of procedure character 0x60, 0x62, 0x63, INS XOR 0x01, INS XOR 0xFF, INS XOR 0xFE.
- Transparent mode with automatic GetResponse implementation.

4.3. Status Description

Returned Contact Status :
0x00: Correct, No Warning.
0x82: Correct, Warning TA1 value will not be taken into account because no PPS procedure is available
0x81: Correct, Warning TB1 indicates that the Writing condition could not be met.
0xF0 : Bad Ack Answer
0xF1 : Bad SWISW2 Answer
0xF3 : Character Sending Error
0xF4 : Character Receiving Error
0xF5 : PPS Failed
0xF6 : Bad TA1 parameter
0xF7 : Bad TP parameter
0xF8 : Bad TCK control
0xF9 : Bad PPS Mode
0xFA : SAM Signal Locked
0xFB : SAM Timeout
0xFC : SAM Lenght Error
0xFD : SAM Not defined
0xFE : incorrect Parameter
0xFF : Timout occurs during dialogue or SAM not detected.

5. DESCRIPTION OF THE NEW ENTER HUNT PHASE FEATURES

Default configuration for Card Search is

- Max Number of cards to look up in ISO 14443-B mode is **1**
- Type Of ReqB (ReqB or WakeUp) is 00 (REQB)
- Number of slots in case of collision is 00 (No slot marker : only probabilistic method has been tested)
- AFI value to seek is 00 (All)
- Automatic Select Diversifier implementation is Yes for the Normal mode and no for the RS485 mode (acts on both Innovatron and ISO14443B card)
- Deselection of the cards by just switching the field off (value = 0).
- SelectApplication() is sent at the end of the EHP procedure, for the ISOB cards (value = 1).

All theses values can be changed by the Class 01 command 17 : F01_17_REQ_PARAM
Parameters in :

- Byte 0: Max Number of cards to look up at (1 up to 5)
- Byte 1: Type Of ReqB (ReqB =0 or WakeUp = 1)
- Byte 2: Number of slots in case of collision (0 = No Slot marker, 1 to 14 otherwise)
- Byte 3: AFI value to look for: (see the ISO 14443 norm)
- Byte 4: Automatic Selection of Diversifier: (0 to disable automatic)
- Byte 5: Real deselection of the cards (if 1) or only Field On/Off (if 0).
- Byte 6: The SelectApplication() command will be sent at the end of the EHP for ISOB cards(if 1).

If the following command is sent: 01 17 00, the complete status of all the parameters in their current value will be given back.

The choice « No automatic selection of the Select Diversifier » is useful for:

- For the multiSam use to avoid sending this command to a SAM before that the application knows which card is detected and so can choose the good SAM slot.
- For the multiword detection to avoid sending this command to the SAM before knowing which of the card will be chosen first.
- For the RS485 to avoid lacks of communication affected by the critical section of the SAM communication routine.

To allow this choice a new command to Select Diversifier is implemented
Function: F01_16_Select_DIV
Parameters in :

- Byte 0: Sam Slot
- Byte 1: protocol
- Byte 2-5: Card Serial Number

Returned value:

- Status of operation 1 = Ok, 0 = Nok

If the “Timeout” mode is chosen (mode = 1), the search will be performed for the all selected types until the timeout reaches 00 (the timeout value to be written in the EnterHuntPhase command is in 10ms unit). The order of the search is as follows:

- GTML/CD97
- Ticket
- Mifare
- ISO A
- ISO B
- MV4000
- MV5000
- Contact

When the timeout is reached, thanks to a raised flag, the search is given up. Otherwise, the following type of card will be looked for. At the end of the loop, if the timeout is still positive, we go on at the beginning of the searching loop.

6. SOFTWARE INTERFACE

This chapter describes the structures of data sent and received on the DATA zones defined in chapter 2.4. Frame.

6.1. Classes of commands

The software interface of the ASK reader is organized into classes of commands defining major types of functions:

- System commands: configuration, transparent commands "TAG" and "SAM"
- Macro-commands cards (CD97, GTML, CT2000 etc....)

6.2. Structure of commands

The commands are defined by:

- Class (system, badge, etc.)
- Instruction (function number in the class)
- Associated data

CLASS 1 byte	INS 1 byte	Associated data xx bytes (270 bytes MAX)

This information is sent by the host to the 'DATA' zone of command for frames.

Details on available commands are provided in chapter 5 of this document.

6.3. Responses to commands

The responses to commands are identical to the 'DATA' zone of commands seen above.

6.4. Error reports

There are 3 levels of error reports:

- Error detected by the reader at the level of access interface to functions: class, unknown function or syntax error in system class command. These errors are reported by the reader to the host via the 'ERR' bit in STA register. There is no other report.
- Error detected by a reader function called by the host (class other than system class): the error is reported by the reader to the host by an execution report on 3 bytes: 1 byte reporting the origin of the error, 2 bytes of detail on the error.
- Error detected by the badge or a SAM: the error is reported by the reader to the host through an execution report on 3 bytes: 1 byte representing the error family, 2 bytes coming from the badge or the error details SAM.

The error report codes are proper to each function class.

6.5. Specific SAM Errors

The transparent command 01_14_SendToSAM communicates directly with the SAM and can therefore give back very specific error codes about the exchange with the SAM:

SAM_EXEC_OK	$00
SAM_BAD_ACKANSWER	$E1
SAM_BAD_SW12ANSWER	$E2
SAM_NR2S_ERROR	$E3
SAM_NR2R_ERROR	$E4
SAM_BAD_PPSEXEC	$E5
SAM_BAD_TA1PARAM	$E6
SAM_BAD_TPARAM	$E7
SAM_BAD_TCK	$E8
SAM_BAD_PPSMODE	$E9
SAM_SIGNAL_LOCKED	$EA
SAM_TIME_OUT	$EB
SAM_LENGTH_ERROR	$EC
SAM_NOT_DEFINED	$ED
SAM_INCORRECT_PARAM	$EE
SAM_NOT_DETECTED	$FF

7. DETAILS OF ASK READER FUNCTIONS

7.1. Definitions and notations

- **Presentation conventions**

All functions of the reader will be presented as follows:

Function name

Description : actions carried out by the function

CLASS	INS	DATA IN		
[Class]	*[Ins]*	*[P1] (a)*	*[P2]*	*…*

[Px] **:** list of input data

a : Length in bytes of data (x = variable)

CLASS	INS	DATA OUT		
[Class]	*[Ins]*	*[D1] (a)*	*[D2]*	*…*

[Dx] **:** list of output data

a : Length in bytes of data (x = variable)

All values following the character '$' are in hexadecimals.

7.2. DOWNLOAD Class (N°= $00)

Class number : $00

Download start

Description : Starts software download.

CLASS	INS	DATA IN
$00	$01	-

No associated input data

CLASS	INS	DATA OUT
$00	$01	Status(1)

STATUS : 1 byte

0x00 No error
0x01 Error while erasing sector 1
0x02 Error while erasing sector 2
0x03 Error while erasing sector 3
0x0B & 0x0E Error in programming the Flag in EEPROM before loading
0x0C Error in programming the Flag in EEPROM after loading

Change of default rate

Description : Changes the rate of a series link in RS232/RS485/TTL series (default rate configured in factory at 115200 baud). The new value will be taken into account at next reset of the coupler.

CLASS	INS	DATA IN	DATA IN	DATA IN
$00	$04	RS232 (1)	RS485 (1)	TTL (1)

RS232 : baud rate divider on RS232 serial link. (baud rate divider is 1382400 / BAUDRATE)
RS485 : baud rate divider on RS485 serial link. (baud rate divider is 1382400 / BAUDRATE)
TTL : baud rate divider on TTL serial link. (baud rate divider is 1382400 / BAUDRATE)

CLASS	INS	DATA OUT
$00	$04	Status (1)

STATUS : 1 byte : 0x00 Failure
0x01 Success

Write SAM Number

Description : Writes in the EEPROM the SAM Number to use by default. This value will be useful for the initialization of the CSC, when selecting the SAM to use at the beginning, before the first transactions.

CLASS	INS	DATA IN
$00	$06	Number

Number : number of the SAM to use by default.

Note: if a value below 1 or above 4 is wrtten, the default SAM will be the SAM number 1.

CLASS	INS	DATA OUT
$00	$06	Status (1)

STATUS : 1 byte : 0x00 Failure
0x01 Success

Write Config Eeprom

Description : Writes in the EEPROM configuration.

CLASS	INS	DATA IN	
$00	$07	Index	Value

Index: $01 : **Value** = RS232 baud rate divider =1382400 / BAUDRATE

$02 : **Value** = RS485 baud rate divider =1382400 / BAUDRATE

$03 : **Value** = TTL baud rate divider =1382400 / BAUDRATE

$04 : **Value** = SAM Number

$05 : Field off CTx : turn on the field before CTx command and turn off the field during (**Value** * 1 ms) after CTx command. 0x00 and 0xFF disable field managment on CTx

$06 : Auto Led management enabled if **Value** = 1. Leds are manged by firmware (red = power on, orange = field on, green = reader/card communication.

$07 : ISO 15693 modulation, 10% if **Value** = 1, otherwise 100%.

$08 : Host communication frame padding : module 62 byte padding if **Value** = 62.

$09 : ISO14443-4 number of retries.

$0A : Delay between retries (ms).

$0B : default RX RF speed at reset (00=106, 01=212, 02=424, 03=847 kb/s).

$0C : default RX RF speed at reset (00=106, 01=212, 02=424, 03=847 kb/s).

$0D : SAM reset at coupler reset (0=no reset)

$0E : AUX Pin signal

$0F : High baud rate ISO14443-A gain (00=20, 01=24, 02=31, 03=35 dB)

$10 : Slot 4 switch test (1=yes (CAM), other = no (SAM))

$11 : Strict ISO14443-3B timeout (1=strict check, other = no strict check, same as GEN3XX)

$12 : Strict ISO14443-4B timeout (1=strict check, other = no strict check, same as GEN3XX)

$13 : Delay after REQ/Select (0 or FF : no delay, same as GEN3XX, other = delay in ms)

$14 : Unconditional Mifare selection before authentication (if value=1)

$15 : MUX482 J1/J3 selection are the same as MUX382 (if value=1)

$16 : Custom Frame Waiting Time (Value * 10 ms, 00 or FF = no custom FWT)

$17 : ISO14443-4 retries on PICC timeout (if value=1)

CLASS	INS	DATA OUT
$00	$07	Status (1)

STATUS : 1 byte : 0x00 Failure

0x01 Success

Read Config Eeprom

Description : Read the value at the Index EEPROM.

CLASS	INS	DATA IN
$00	$08	Index (1)

Index: EEPROM configuration index

CLASS	INS	DATA OUT	
$00	$08	Status (1)	Value (1)

STATUS : 1 byte : 0x00 Failure
0x01 Success

Value regarding index : **see Write Config Eeprom**

Default values:

CSC Configuration

Index	Label	Hex Value	Signification
1	RS232 Speed	FF	115200 bauds
2	RS485 Speed	FF	115200 bauds
3	TTL Speed	FF	115200 bauds
4	Default SAM	01	SAM n°1
5	Field off CTX	1E	During 30 ms
6	Auto LED	FF	Disabled
7	15693 modulation	FF	100%
8	Frame padding	FF	Disabled
9	ISO14443-4 retries	FF	10
10	ISO14443-4 retries delay	FF	10 ms
11	Default RX RF speed	FF	424 kb/s
12	Default TX RF speed	FF	424 kb/s
13	SAM reset at startup	FF	Yes
14	AUX Pin	FF	VEvalL
15	High baud rate ISO14443A gain	FF	35 dB
16	Contact Slot 4 switch test	FF	No (SAM)
17	ISO14443-3B timeout strict check	FF	No
18	ISO14443-4B timeout strict check	FF	No
19	ISO14443 delay after REQ/SELECT	FF	0 ms
20	Mifare selection before authentication	FF	No
21	MUX482 selection same as MUX382	FF	No
22	ISO14443-AB custom Frame Waiting Time	FF	No
23	ISO14443-4 retries on PICC timeout	FF	Yes

SYSTEM Class (N°= $01)

Class Number : $01

This class is dedicated to module resource administration which is not specific to a card mask (communications, configurations, inputs/outputs, update, etc...).

Software version

Description : Sends back reader software version and allows to enable or disable the CRC computation for communications exchanges between the HOST and the CSC.

CLASS	INS	DATA IN
$01	$01	CRC ON/OFF

CRC ON/OFF: 0xFF: CRC not needed. After reset, Version command with CRC must be issued before using CRC disabling.

All other values: CRC needed for each exchange.

By default, the CRC is needed.

This parameter is not mandatory. If not present in the command, the default value will remain unchanged.

CAUTION: this parameter has been implemented to increase the communication speed!

CLASS	INS	DATA OUT
$01	$01	VERSION (x)

VERSION : character chain of software version (terminated by $00).

GEN4XX CSC xx.yy<LDBXXiii> Mmm jj yyy HH:MM:SS (C) ASK SAMs

where

xx = version number

yy = revision number

iii interface USB, // or serial baud rate from 9600 to 691 200 baud

Mmm = Month (3 first letters in ASCII), jj = Day, yyyy = Year

HH = Hour, MM = Minutes, SS = Seconds

s = SAM used by default (value written in EEPROM), equal to “?” if none selected.

ASK SA - 2260, route des Crêtes - BP 337 - 06906 Sophia-Antipolis Cedex - Tel. 04 97 21 40 18 - Fax. 04 92 38 93 21 Web : www.ask-rfid.com

Enter Hunt Phase

Description : This function enables badges to be searched for in different modes and different ways.
The command may be cancelled by a STOP command if no timout is specified.

CLASS	INS	DATA IN												
$01	$03	MONO	ANT	OTH	CONT	MV	ISO B	ISO A	Mifare	Ticket	INNO	MODE	FORGET	TIMOUT

MONO : 1 Nibble : $4 for a single-shot search, $0 otherwise.

ANT : 1 Nibble : kept for compatibility but unused.

OTH : 1 Nibble : Other chip detection : at this time only SRI is supported.

CONT : 1 Nibble : ISO7816 contact Card search, available if extension board is present

MV(MV5K/MV4K):1st half nibble(Most Significant Bits) = Number of MV5K search, 2nd half nibble(Least Significant Bits) = MV4K.

ISO B : 1 Nibble : Number of successive card search to perform in ISO B norm protocol.

ISO A : 1 Nibble : Number of successive card search to perform in ISO A norm protocol and mandatory for the MIFARE® UltraLight.

MIFARE® : 1 Nibble : Number of successive card search to perform in MIFARE® protocol.

TICKET : 1 Nibble : Number of successive card search to perform in CTS / CTM protocol.

INNO : 1 Nibble : Number of successive card search to perform in Innovatron RF protocol.

MODE : Type of Enter Hunt Phase
- $00 : short (compatibility with the old one)
- $01 : long (take into account the 2 following bytes)
- $02 : response (before tag research used for RS485 communication)

FORGET : 1 byte
- $00 takes into account the serial number of the last badge presented.
- $01 Forgets the serial number of the last badge presented.

TIMOUT: 1 byte (N x 10 ms)
Polling duration of the enter hunt phase. Beyond this time the reader replies a status.
This avoids the reader staying indefinitely awaiting a badge.
If the value is nil, waiting time is infinite.

NB: If the MONO mode of the byte 0 is selected with the LONG MODE (byte 5 = 1), either the Time Out lasts less than the sequence of all the searches selected and the hunt stops before the whole sequence is achieved, or if greater the hunt stops after the end of the sequence, before the Time Out reaches the end.

NB: on multi-protocol cards (INNOVATRON and ISOB), ISOB nibble should be more than 1.

CLASS	INS	DATA OUT				
$01	$03	CNT (1)	COM (1)	LNG (1)	ATR (1)	… ATR (N)

CNT : 1 byte, kept for compatibility,

8x indicates that antenna x has broken down,

00 otherwise

COM : 1 byte, communication mode

$02 : Card recognized with ISO 14443 type A protocol (ISO level 4 compliant but not Calypso)

$03 : Card recognized with INNOVATRON protocol

$04 : Card recognized with ISO 14443 type B protocol (Calypso Card)

$14 : ISO 14443 type B protocol asked but an unwanted collision occurred.

$05 : Card recognized with ISO 14443 type MIFARE® protocol

$15 : ISO 14443 type MIFARE® protocol asked but an unwanted collision occurred.

$06 : CTS or CTM Ticket recognized

$07: Card recognized in contact mode

$08 : Card recognized with ISO 14443 type A part 3 but not compliant with 14443 type A part 4

$18 : ISO 14443 type A protocol asked but an unwanted collision occurred.

$09 : Card recognized with ISO 14443 type B protocol (Non-Calypso card)

$0A : Card recognized with MV4000 protocol (no data format control)

$1A : MV4000 protocol asked but an unwanted collision occurred.

$0B : Card recognized with MV5000 protocol (no data format control)

$1B : MV5000 protocol asked but an unwanted collision occurred.

$0C : Card recognized with ISO 14443 type A protocol ((Calypso Card)

$0D : SRI detected

$6F : Timeout expired.

$7F: Response (before tag research)

NB: no collision code has been implemented for the INNOVATRON or CTx cards.

LNG : 1 byte, length of cards responses

ATR : N bytes :

- INNOVATRON protocol :

  Long REPGEN data in the case of Innovatron protocol cards composed of
  - The serial number (4 bytes) followed by 2 bytes
  - The Answer to reset (17 bytes for CD97 and GTML)
  - The Status words (2 bytes)
- ISO 14443 Type B protocol :
  - The communication channel number CID (1 Byte)
  - The Serial Number if the card is Calypso-Compliant, otherwise the first 4 bytes are filled at 00 and the 4 others are the PUPI (8 bytes).
  - 1 byte optional length (1 to 16) of application name (if bit 3 of byte 6 of EnterHuntPhaseParameter = 1)
  - n byte optional application name (if bit 3 of byte 6 of EnterHuntPhaseParameter = 1)
  - The Historical Bytes if the card is Calypso-Compliant, otherwise the last 3 are filled at 00(7bytes for GTML2) same order as in the SelectApplication command
  - The Status words is returned if the card is Calypso-Compliant, filled at 00 otherwise (2 bytes)

- MIFARE® protocol:

  The answer in this mode depends on two parameters: for the request to be really sent in MIFARE® mode, the presence of the MIFARE® chip in the reader is mandatory AND only one card must be looked for. Otherwise the answer is formatted just as an ISO 14443 Type A (14443-4 compliant or not). But if the search is a true MIFARE® search, the format is as follows:

  - Communication Status (1 Byte): set to 0x00: OK .
  - Type of the card (e.g.: 0x08 for the Mifare® Classic, 0x18 for the Mifare 4k, 0x28 for the Mifare Classic implementation in ProX) (1 Byte)
  - The 4 Serial Number bytes

- ISO 14443 Type A protocol :
  - ISO 14443-4 compliant:
    - The communication channel number CID (1 Byte).
    - Length of the Serial Number (1 Byte).
    - The Serial Number if the card is Calypso-Compliant, otherwise the UID (4, 7 or 10 bytes).
    - 1 byte optional length (1 to 16) of application name (if bit 3 of byte 6 of EnterHuntPhaseParameter = 1)
    - n byte optional application name (if bit 3 of byte 6 of EnterHuntPhaseParameter = 1)
    - Length of the Information (1 Byte).
    - Information given back in the ATS, including:
      - Maximum size of a frame accepted by the card. (1 Byte)

      Warning: 0xFF means 256 bytes!
      - The bit rate from the coupler to the card. (1 Byte)
      - The bit rate from the card to the coupler. (1 Byte)
      - Only the same baud rate for both direction is supported? -> 1 if TRUE. (1 Byte)
      - The Frame Waiting Time Integer which defines the Frame Waiting Time between the end of the frame sent by the coupler and the beginning of the answer of the card. (1 Byte)
      - The Specific Guard Time Integer which defines the time needed by the card before answering after sending the ATS. (1 Byte)
      - NAD supported? -> 1 if TRUE. (1 Byte)
      - CID supported? -> 1 if TRUE. (1 Byte)
      - Information of the Application (historical bytes).
  - ISO 14443-4 non-compliant: (for instance the MIFARE®UltraLight)
    - Set to 00 (1 Byte) (means non ISO–4 compliant).
    - Length of the following Serial Number (1 Byte).
    - Serial Number (Length Bytes)

- CTS or CTM Ticket recognized, 9 bytes (resp. 10) if the EnterHuntPhase has been processed in mode 1 (resp. mode 0) corresponding to:

  In mode 1 (long EnterHuntPhase):

  - Ticket status (1 byte) (0x0F if detection successful)
  - Product code / Manufacturer code (2 bytes)
  - Application code / Embedder code (2 bytes)
  - Serial number in (4 bytes)

In mode 0 (short EnterHuntPhase):

- Manufacturer code / Product code (2 bytes)
- Embedder code / Application code (2 bytes)
- Serial number in reverse order(4 bytes)
- System and Invalidation bits (2 bytes)

- SRI recognized:

- Status (1 byte):	$00:	communication interrupted
	$01:	bad CRC
	$0F:	success
	$80:	collision
- Chip type (1 byte):	$00:	SR176
	$01:	SR512
	$02:	SR4K
- UID:	64-bit (8-byte) UID from LSB to MSB (UID0, UID1… UID7)	

- ISO7816 Contact mode: answer to reset, whose length depends on the different kinds of cards.
- Motorola cards :
  - MV4000 cards:
    - The DAD byte (1 Byte).
    - The PUPI (4 Bytes).
    - Length of the Historical bytes (1 Byte = 0x03).
    - Historical bytes
    - Length of the Serial Number (1 Byte = 0x08).
    - The Serial Number
  - MV5000 cards:
    - The DAD byte (1 Byte).
    - The PUPI (4 Bytes).
    - Length of the Historical bytes (1 Byte = 0x04).
    - Historical bytes
    - Length of the Serial Number (1 Byte = 0x18).
    - The Serial Number

The status words -> 0x90 XX where XX is the MF status

0xX0: card is valid

0xX1: card is invalided

0x0X: no PIN presentation error

0x1X: one PIN presentation error

0x3X: two PIN presentation error

0x7X: PIN blocked

Remark 1 : For detection of several ISO14443 cards, (if multidetection is activated for the ISO card), the card response are appended several times in the response data field.

Remark 2 : For each new detection of ISO A or B by the EnterHuntPhase commands, the data for cards previously found are lost, so it is not possible to communicate with both type A and type B cards in the same time.

Remark 3: By default, the EnterHuntPhase command sends WAKE UP requests. So if multidetection is wished, the selection of the REQ request must be previously done thanks to the 01_17_SetEHPparameters.

Remark 4:In case of short EnterHuntPhase (mode = 0), if a ticket or a card in protocol ISOB, INNOVATRON or Contact Mode has been found once it could not be seen a second time just after having been found, because the reader memorizes the serial number, as if the byte “forget” in the long EnterhuntPhase had been set to 0. If the user want to see the card again, the forgetting of the serial number must be forced thanks to the 01_0E command.

Note : Recognition of the type of card or ticket (e.g. GTML or CD97) is the responsibility of the application which then sends the commands to the card according to the appropriate class.

The GEN4XX accepts GEN3XX format commands to maintain compatibility.

End Tag Communication

Description : Ends communication with the badge or the ticket (in ISO and Innovatron protocols) and allows to configure the end of the transaction with the card.

CLASS	INS	DATA IN
$01	$04	DNX (1)

DNX : This byte is bit-mapped:

Bit0: $xxxx xxx0 does not send disconnection order, but deselects the CIDs in ISO protocol.

Bit0: $xxxx xxx1 sends disconnection order.

And for the Innovatron protocol, more options are available:

Bit1: $xxxx xx1x keeps the field ON during 60 ms after having sent, or not, the disconnection order.

Bit2: $xxxx x1xx switches OFF the field during 10ms and then switches it ON properly.

Bit3: $xxxx 1xxx forgets the last card found.

Bits7..4 are Reserved for Future Use.

In Innovatron protocol, if all the options are selected, their associated functionalities will be performed in the same order as the parametered bits, from bit 0 to bit 7.

Thus, if the user sends 01 04 0F, the coupler will:

- send a disconnexion order.
- Wait for 60 ms after having received the acknowledgement of the disconnection order.
- Switches off the field
- Switches on the field.
- Erases the Serial Number of the last card treated.

CLASS	INS	DATA OUT
$01	$04	ACK (1)

ACK : $00 disconnection acquitted

$01 disconnection not acquitted or input byte = $00

Get Communication Status

Description : The status is that of the last exchange of radio frame ;
The duration of radio communication is measured between the beginning of contact (before APGEN frame) and end of dialogue
(after response to DISC or at the end of timeout of the last repetition).
The number of resumptions corresponds to the difference between the number of messages sent to the card and the number of messages received from the card in the course of communication (not available for all card types).

CLASS	INS	DATA IN
$01	**$05**	**-**

No associated input data

CLASS	INS	DATA OUT			
$01	**$05**	**STATUS (1)**	**TIME (2)**	**REP (1)**	**TIME_END**

STATUS: communication status of last radio exchange
$01 : data received
$00 : no data received in timeout delay
$FF : data coding error
$FE : error detected by communication controller
$FD : reception buffer overflow
$FC : timeout delay expired before end of reception
$FB : CRC error

TIME : duration in milliseconds (whole not signed, MSB at the head) of communication.

REP : number of resumptions

TIME_END : duration in milliseconds (whole not signed, MSB at the head) of last communication.

Switch off antenna

Description : switches ON or OFF the electromagnetic field of the antenna (stop carrier).

CLASS	INS	DATA IN				
$01	$0E	ANT (1)	FIELD (1)	FORGET (1)	MODE(1)	TIMEOUT(1)

ANT : RUF (selection of antenna) $00 : Antenna 1
Others : RUF

FIELD : status of the field: $00 : OFF
$01 : ON

FORGET : $00 Memorizes the serial number of the last badge presented
Parameter kept for compatibility with previous generation couplers
$01 Forgets the serial number of the last badge presented.

MODE : Type of the present command:
$00: short (compatibility with the former one)
$01: long (takes into account the next byte for the timeout)

TIMEOUT : Duration of the present command (in ms).

(Remark : a compatibility is ensured with the old Switch_Off_Antenna format (only the first parameter for the FIELD parameter)

CLASS	INS	DATA OUT
$01	$0E	STATUS (1)

STATUS : $00 : OK
$01 : problem of communication

Send to antenna

Description **:** Sends the data to the chosen transparent mode (either ISOA, ISOB, cards understanding class 5 commands, and MV4000 or MV5000) .

CLASS	INS	DATA IN	
$01	$12	LNG (1)	DATA (x)

LNG : length of DATA

DATA : data made up of :

* The length of the command to transmit to the card + 1(actually must be equal to LNG parameter).

* The real frame to transmit to the card.

CLASS	INS	DATA OUT		
$01	$12	STATUS (1)	LNG (1)	DATA (x)

STATUS : communication status

$01 : data received

$00 (resp $03): no data received in timeout delay in Innovatron or ISOA protocol (resp. ISOB protocol)

$06 : invalid CID

$08 : ICC fails to answer correctly

$FF : data coding error

$FE : error detected by communication controller

$FD : reception buffer overflow

$FC : timeout delay expired before end of reception

$FB : CRC error

LNG : length of DATA

DATA : data made up of :

Length of the response from the card + 1.

The response from the card.

ASK SA - 2260, route des Crêtes - BP 337 - 06906 Sophia-Antipolis Cedex - Tel. 04 97 21 40 18 - Fax. 04 92 38 93 21 Web : www.ask-rfid.com

Send to antenna extended

Description **:** Sends the data to the chosen transparent mode (either ISOA, ISOB, cards understanding class 5 commands, and MV4000 or MV5000) .

CLASS	INS	DATA IN		
$01	$22	LNG LOW (1)	LNG HIGH (1)	DATA (x)

LNG LOW and LNG HIGH : length of DATA = LNG LOW + (256 * LNG HIGH)

DATA : The frame to transmit to the card.

CLASS	INS	DATA OUT			
$01	$12	STATUS (1)	LNG LOW (1)	LNG HIGH (1)	DATA (x)

STATUS : communication status

$01 : data received

$00 (resp $03): no data received in timeout delay in Innovatron or ISOA protocol (resp. ISOB protocol)

$06 : invalid CID

$08 : ICC fails to answer correctly

$FF : data coding error

$FE : error detected by communication controller

$FD : reception buffer overflow

$FC : timeout delay expired before end of reception

$FB : CRC error

LNG LOW and LNG HIGH : length of DATA = LNG LOW + (256 * LNG HIGH)

DATA : the response from the card.

Reset SAM

Description : Executes hardware initialization of a security module.
Returns the module ATR.

CLASS	INS	DATA IN	DATA IN	DATA IN
$01	$13	SAM (1)	INN (1)	ISO (1)

SAM : selection of SAM :
$00 : SAM usually selected
$01 : SAM 1
$02 : SAM 2
$03 : SAM 3
$04 : SAM 4
Others RUF

INN : selection of SAM in Innovatron rapid protocol
If = $01 , selection of SAM in Innovatron protocol
If = $00, no SAM selection in this protocol

ISO : selection of protocol ISO 7816
If = $01 , selection of SAM in ISO 7816protocol
If = $00 , no SAM selection in this protocol

NB1 : if these 2 parameters are positioned, there is first scrutinization in Innovatron protocol, then in ISO7816 protocol.

NB2 : If this command is 01 13 00 (INN= $01 and ISO= $00 by default)

CLASS	INS	DATA OUT		
$01	$13	STATUS (1)	LNG (1)	ATR (x)

STATUS : communication status
$00 : data received in Innovatron protocol
$80 : data received in ISO 7816 protocol.
$81 : data received in ISO 7816 protocol but writing prohibited.
$82 : data received in ISO 7816 protocol but speed is fixed at default speed.
$FF : no data received

LNG : length of data

ATR : Response to SAM RESET selected.

Send to SAM

Description : Sends the data to the chosen transparent mode.
The communication parameters are those chosen using the SAM communications parameter entry function

CLASS	INS	DATA IN			
$01	$14	SAM (1)	LNG (1)	DATA (x)	DIRECTION(1)

SAM : SAM number= $00, $01, $02, $03 or $04 as defined in "Reset Sam" command
LNG : length of data (LNG included)
DATA : data made up of the ISO command to transmit to the SAM
DIRECTION : (requisite for ISO7816 contact mode only) : IN ($01), OUT($02), IN_OUT ($03)

CLASS	INS	DATA OUT		
$01	$14	STATUS (1)	LNG (1)	DATA (x)

STATUS : communication status
$00 : data received
$FF : no data received
Others: see the chapter 8.5

LNG : length of data (LNG included)
DATA : data which makes up the SAM response.

SELECT_CID

Description : Select a communication channel to communicate with an ISO 14443 card.
parameter entry function

CLASS	INS	DATA IN
$01	$15	CID (1)

CID : CID value from 1 to Maxvalue

CLASS	INS	DATA OUT
$01	$15	STATUS (1)

STATUS : Status of operation 1 = Ok, 0 = Nok (Bad CID value)

SELECT_DIV

Description : Select a Diversifier for the transaction. Usable after a Security module change, a CID change or if the option ("no Diversifier") is chosen in the Select option command or in RS485 default option mode.

parameter entry function

CLASS	INS	DATA IN		
$01	$16	SLOT (1)	PROT(1)	CardSerialNumber (4)

SLOT : SAM SLOT

PROT : SAM PROTOCOL (0 = Innovatron, 1 = ISO)

Card Serial Number : Last 4 bytes of the card Serial Number

CLASS	INS	DATA OUT
$01	$16	STATUS (1)

STATUS : Status of operation 1 = Ok, 0 = Nok

EnterHuntPhase Parameters

Description : Initialize parameters for ISOA/B and Innovatron Protocol. If the first parameter is set to 00, the command can send the present configuration back.

CLASS	INS	DATA IN								
$01	$17	MAXNBCARD (1)	REQ (1)	NBSLOT (1)	AFI (1)	AutoSelDiv (1)	Deselect (1)	SelectAppli (1)	LNG (1)	DATA (x)

MAXNBCARD : Max Number of cards to look up at (default value is 1 for single card detection)

Note: if $00 and nothing for others parameters, the present configuration will be given back.

REQ : Type Of Request (00=Req or 01=WakeUp) (default value is WakeUp)

Note : the ISO 14443 A anticollision required type of Request = Req.

NBSLOT : Number of slots in case of collision (default value is 0 for probabilistic method)

AFI : AFI value to seek (default value is 0 for All AFI)

AutoSelDiv : Automatic Select Diversifier method (default value is 0 (FALSE) for CSC RS485 or 1 (TRUE) for other CSC.)

Deselect : When a later deselection will be called, if set to $01-> real deselection of the found cards and if $00-> switches the field off.

SelectAppli **:** Select Application mangement for ISOA / ISOB

%000x xxx1 : send Select Application to card after detection (ISOA / ISOB)

%000x xx1x : Force to $00 (instead of$94) the Select Application « CLA » field.

%000x 1xxx : add selected application name in the F01_03_EnterHuntPhase» answer.

LNG : optional data length

DATA: : optional name of the Application to select (complete or partial, in hexadecimal). Default is “1TIC.” (0x31 0x54 0x49 0x43 0x2E)

CLASS	INS	DATA IN								
$01	$17	MAXNBCARD (1)	REQ (1)	NBSLOT (1)	AFI (1)	AutoSelDiv (1)	Deselect (1)	SelectAppli (1)	LNG (1)	DATA (x)

The Data Out represent the present configuration for the EHP.

Switch Signals

Description : Activates or deactivates the interface signals (Led or Buzzer)

CLASS	INS	DATA IN	
$01	$18	SIGCPU	SIGANT

SIGCPU : 1 byte, CPU description
Bit 0 = Led 1 (red or green), activation if bit = 1, deactivation if bit = 0.
Bit 1 = Led 2 (orange or green), ditto
Bit 2 = Led 3 (green), ditto
Bit 3 = Led 4, (NE or green), ditto

NB: this byte inactive if the AUTO LEDs configuration = 1

SIGANT : 1 byte, antenna switching

Behavior depends of configuration index 21 value (MUX482 J1/J3 selection are the same as MUX382):

Configuration value = all value except 01 (default behavior):

SIGANT Bit 0 = 1 → antenna connected to J1 of MUX 482 is active
SIGANT Bit 1 = 1 → antenna connected to J3 of MUX 482 is active

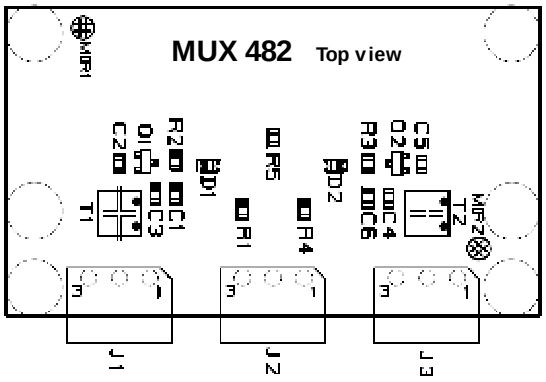
Configuration value = 01 (the CPL427/MUX482 will act as the CPL325/MUX382):

SIGANT Bit 1 = 1 → antenna connected to J1 of MUX 482 is active
SIGANT Bit 1 = 0 → antenna connected to J3 of MUX 482 is active

Note: for SIGANT, one and only one of the values listed above should be used at one time.

CLASS	INS	DATA OUT
$01	$18	STATUS (1)

STATUS : status = $00



Select SAM

Description **:** Selects the SAM chosen for the application

CLASS	INS	DATA IN	
$01	$19	SAM (1)	Protocol (1)

SAM : SAM slot number from 1 to 4

Protocol **:** 0 : Innovatron protocol

1 : ISO protocol

CLASS	INS	DATA OUT
$01	$19	STATUS (1)

STATUS : communication status :

$00 : command possible

$other : SAM absent

Buzzer management

Description **:** sounds the buzzer

CLASS	INS	DATA IN		
$01	$23	TYPE(1)	Frequency (2)	Duration (2)

TYPE : 0 = system beep (500 Hz, 250 ms)

1 = PayPass sequence (Buzzer + 4 leds)

2 = user beep (see parameters below)

Frequency : MSB LSB = frequency in Hz

Duration : MSB LSB = duration in ms

CLASS	INS	DATA OUT
$01	$23	STATUS (1)

STATUS : communication status: always 00

7.3. GTML Class (N°= $02)

Class number : $02

Presentation of GTML badge (Generic Transport Mask Light), This Card is a transport-dedicated card.

For specifications detail on GTML card refer to "GTML User Manual" Ref. 990305-ASK- GTML-UserManual.DOC.

7.3.1. Main functions

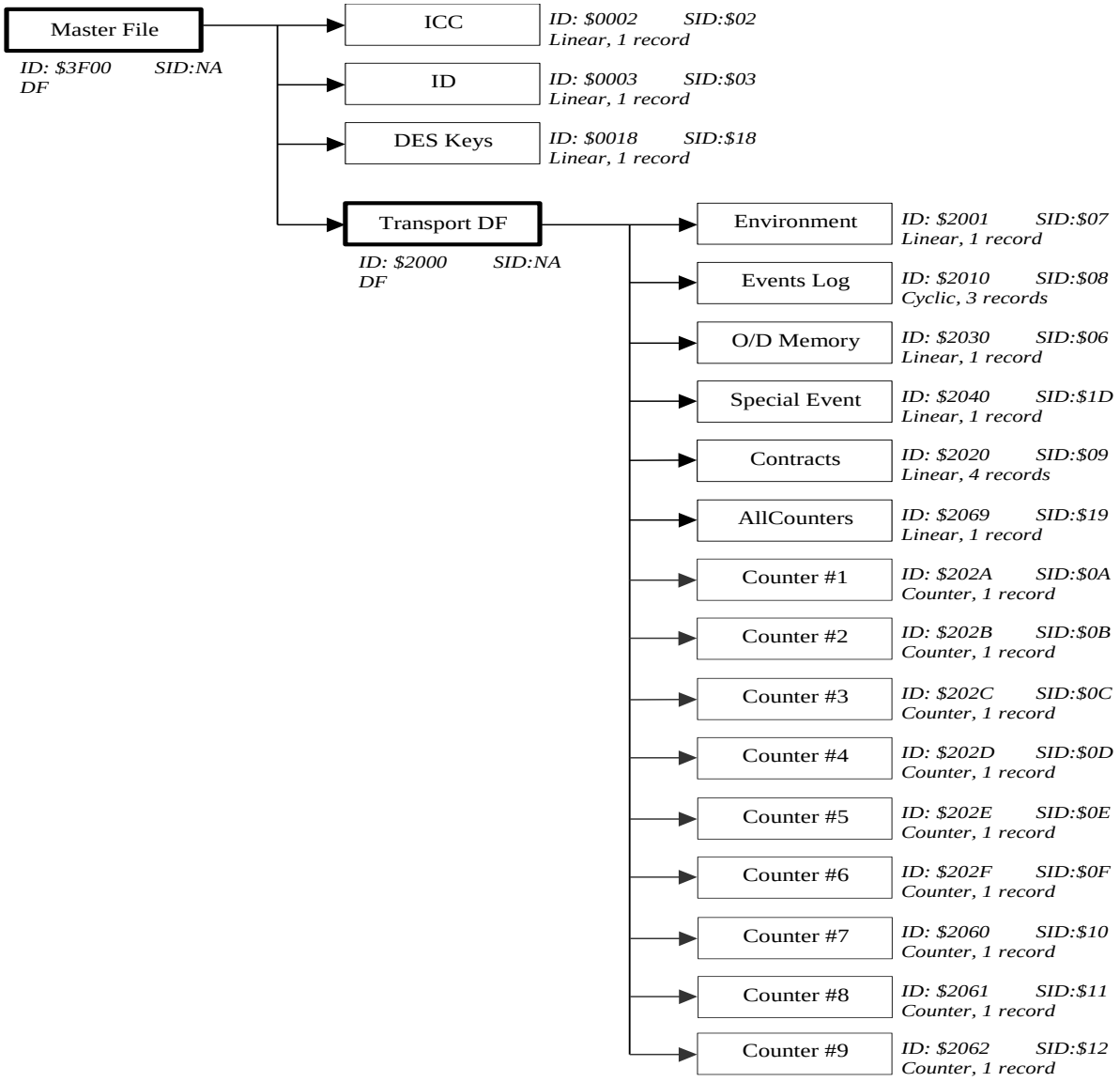
Ticketing : ensures checking of access to public transport networks. This is its main function. This includes the different existing pricing modes, and should enable the introduction of new ones.

7.3.2. Structure of data

The card is organized into files according to ISO/IEC 7816-4 and takes its inspiration from the EN726-3 norm.

The GTML card files have several attributes :

- type : DF (dedicated file) or EF (elementary file),
- long identifier (LID),
- short identifier(SID),
- sub type (only for the EFs) : fixed structure, circular, counter.



7.3.3.Functionalities:

Definitions common to class functions :

7.3.3.1. Access mode:

Value on 1 byte defining the authentication principles used to access data.

DEFAULT	$00	No local cryptogram for the function. If a session is open, it is the session security that is used. Otherwise only data of which the attribute is "ALWAYS" will be accessible.

Concerning GTML the only authentication mode possible for recording is the session mode, and for reading it is “always” mode or the session mode.

7.3.3.2. Execution report :

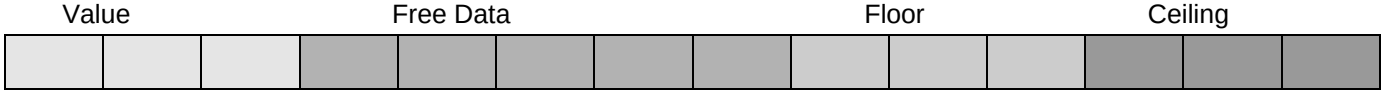
Value on 3 bytes (1 general byte, 2 bytes of detail).

Title	byte 0	Bytes 1-2 (detail)	
Execution carried out(no error)	$00	$90 $00	Command accepted and executed.
File error	$01	$69 $81 $6A $82 $6A $83	Incompatibility with file structure. File not found Recording not found.
Badge side security error	$02	$69 $82 $63 $CX $69 $83	Security conditions not respected (no alea, cryptogram absent, no current session, access condition unknown) Incorrect signature. X represents the number of attempts still authorized (0, 1 or 2) Command rejected since maximum number of errors (3) was reached (PIN access only).
Session Error	$03	$64 $00 $69 $85 $69 $86	EEPROM capacity insufficient for the session. Command forbidden(the mode NEVER is indicated for this command). Command prohibited on a DF.
Badge physical Error	$04	$62 $81 $65 $81	The data item returned is corrupted (Fatal Error) EEPROM Failure (Fatal Error).
Application Error	$05	$6A $80 $6A $81 $6A $87 $67 $00 $6B $00 $6D $00 $6E $00	Value to deduct or add incorrect. Wrong key level specified (1 to 3). Lc incompatible with P1-P2 P3 invalid (returned if P3 = 0). P1 – P2 not supported. INS not supported. CLA not supported.
Invalid card	$06	$62 $83	DF parent or MF invalidated.
Badge error code unknown	$09	Badge return code unknown	

Abnormal execution reported by security module	codes$20-$29	cf spec CD97MiniMS	
SAM side security alert	$20	$69 $00 $69 $85 $69 $88	Command not authorized (A counter of key uses reached its maximum value) Command not authorized (conditions of use nor satisfied) Cryptogram incorrect
SAM physical error	$21	$65 $81	Eeprom Problem
SAM instruction error	$22	$94 $10 $94 $20 $6A $83 $6A $86 $6A $87 $64 $00	Value incorrect in incoming data. Exceeding of PME badge capacity. Recording not found. The key requested is not present in the key file . P1-P2 incorrect. Key reference too large. Lc incompatibly with P1-P2. Execution error.
SAM security alert	$29	SAM return code unknown	
SAM Answer TimeOut	$2F	$00 $EF	No SAM answer
Incorrect card communication (dialogue lost)	$40	byte 0 : byte 1 : if	$00 (RUF) INS code (ISO) of the frame which has failed ($00) multiple command.
Incorrect SAM communication	$41	byte 0 : byte 1 :	$00 (RUF) INS code (ISO) of the frame which has failed.
Abnormal execution reported by the CSC controller	$80	$00$00 $00$02 $00$03 $00$04 $00$05 $00$06 $00$07 $00$08 $00$09 $00$0A $00$0B $00$0C $00$0D 80xx	command unknown$00$01 concatenation not authorized for this function maximum number of concatenable functions reached access mode incompatible with the concatenation access mode prohibited for this function no session open session already open badge response incorrect command prohibited in session overflow of transmission buffer to the badge SAM key reference version "x" not found ChangeKey prohibited if not preceded by SelectFile ChangeKey prohibited when the new version of key is 0 the attempt to ChangeKey with a key version >1 failed since the key in the card is version 0 (personalization to do) error in index input parameter $xx

Note : These report values are proper to the class.

7.3.3.3. Structure of elementary counter files :



Value : 3 bytes, binary unsigned, representing current counter value.
Free data : 5 bytes, Value fixed at $0000000000.
Floor : 3 bytes, representing counter floor (its value is fixed at $000000).
Ceiling : 3 bytes, representing counter ceiling(its value is fixed at $FFFFFF).

7.3.3.4. FCI (file description data) :

Tag 1 byte, Fixed value = $85
Length 1 byte, Fixed value = $17
SID 1 byte, short identifier, = $00 for the DF and for the MF.
Type 1 byte : = $01 for MF
= $02 for DF
= $04 for EF
EFType 1 byte, type of elementary file= $00 for the DF and for the MF.
= $02 Linear fixed structure.
= $04 Circular.
= $08 Counter.
RecSize 1 byte, number of bytes per recording.
= $00 for the DF and MF.
= $1D for the EF.
NumRec 1 byte, number of recordings in the file,
= $00 for the DF and MF.
AC 4 bytes : the 1st for the command index number 0, the 2nd for the command index number 1, the 3rd for the command index number 2 and 4th for the command index number 3.
The possible access modes are:
= $00 NEVER
= $01 PIN
= $10 SESSION
= $1F ALWAYS
Other values RUF.

Correspondence chart between command index number and type of file.

Command index number	MF and DF	EF
0	REHABILITATE	Read Record
1	INVALIDATE	Update Record
2	RUF	Write Record or Decrease
3	RUF	Append Record or Increase

Nkey 4 bytes, key index number to be used.
Status 1 byte :
$X0 Card Valid.
$X1 Card invalidated.
$0X No errors on PIN presentation.
$1X An error of PIN presentation.
$3X Two errors of PIN presentation.
$7X Three errors of PIN presentation. Any new presentation of PIN is rejected.
Other values RUF.
KVC1 1 byte, =$00 for an EF.
KVC2 1 byte, =$00 for an EF.
KVC3 1 byte, =$00 for an EF.
Floor 3 bytes, Value fixed at $000000.
Ceiling 3 bytes, Value fixed at $FFFFFF for the counters and at $000000 for the other types of file.

7.3.4.Set of instructions

APPEND RECORD (GTML)

Description **:** Adds a record to an elementary circular file.
This new record becomes the first in the file, the older one is suppressed.

CLASS	INS	DATA IN			
$02	$01	ACCES = $00	SID (1)	LNG (1)	DATA (x)

ACCES : access mode = $00 (other RFU values).
SID : SID (SID = $00 for usually selected EF)
LNG : length of data to record (n bytes ≤ size of a recording)
DATA : data to record

CLASS	INS	DATA OUT
$02	$01	REND (3)

REND : execution report

READ RECORD (GTML)

Description **:** Reading of a record given in a circular EF, a counter or an EF linear fixed structure.

CLASS	INS	DATA IN			
$02	$06	ACCES = $00	SID (1)	NREC (1)	LNG (1)

ACCES : access mode =$00 (other RFU values).
SID : SID (SID = $00 for usually selected EF)
NREC **:** recording number
LNG **:** length of data to read

CLASS	INS	DATA OUT	
$02	$06	REND (3)	DATA (x)

REND : execution report
DATA : data read (n bytes)

CHANGE PIN (GTML)

Description **:** Records PIN value contained in the MF.
A SelectFile of the MF should be carried out before being able to change the PIN.
This command can't be used during a session.

CLASS	INS	DATA IN			
$02	$02	$04	RFU (1)	OLDPIN (4)	NEWPIN (4)

RFU : For future use
OLDPIN : old PIN
NEWPIN : new PIN

CLASS	INS	DATA OUT
$02	$02	REND (3)

REND : execution report

VERIFY PIN (GTML)

Description **:** Presentation of PIN with counting of number of incorrect presentations.
A SelectFile of the MF should be carried out before being able to check PIN.

CLASS	INS	DATA IN	
$02	$0B	MODE (1)	PIN (4)

MODE : $00, consultation of counter of incorrect presentations
$01, presentation of PIN
$02, presentation of PIN in transparent mode for contact communication

PIN : PIN (4 bytes)

CLASS	INS	DATA OUT
$02	$0B	REND (3)

REND : execution report

DECREASE (GTML)

Description : Decreases the value contained in a counter file.

Remark : This function should be carried out in session and it immediately returns the new value of the counter.

CLASS	INS	DATA IN			
$02	$03	ACCES = $00	SID (1)	VALUE (3)	$00 $00 $00 $00 $00

ACCES : access mode = $00 (other RFU values).

SID : SID (SID = $00 for usually selected EF)

VALUE: value to deduct (3 bytes, binary number positive or nil)

CLASS	INS	DATA OUT	
$02	$03	REND (3)	VALUE (3)

REND : execution report

VALUE: new value (3 bytes, binary number signed)

INCREASE (GTML)

Description : Increases the value contained in a counter file.

The associated data is not written in the GTML.

Remark : This function should be carried out in session and it immediately returns the new value of the counter.

CLASS	INS	DATA IN			
$02	$04	ACCES = $00	SID (1)	VALUE (3)	$00 $00 $00 $00 $00

ACCES : access mode = $00 (other RFU values).

SID : SID (SID = $00 for usually selected EF)

VALUE: value to add (3 bytes, binary number positive or nil)

CLASS	INS	DATA OUT	
$02	$04	REND (3)	VALUE (3)

REND : execution report

VALUE: new value (3 bytes, binary number signed)

INVALIDATE (GTML)

Description **:** Invalidates current DF(and consequently all the related files).

CLASS	INS	DATA IN
$02	$05	ACCES = $00

ACCES : access mode = $00 (other RFU values).

CLASS	INS	DATA OUT
$02	$05	REND (3)

REND : execution report

REHABILITATE (GTML)

Description **:** Cancels file invalidation.

CLASS	INS	DATA IN
$02	$07	ACCES = $00

ACCES : access mode = $00 (other RFU values).

CLASS	INS	DATA OUT
$02	$07	REND (3)

REND : execution report

SELECT FILE (GTML)

Description : Explicit selection of current EF or DF.
This command sends back file description data.

CLASS	INS	DATA IN		
$02	$08	CNTR (1)	LNG (1)	PATH (x)

CNTR : Control of selection
$00, MF
$02, EF in current DF (identifier bytes 2 to n)
$08, bytes 1 to n : path from the MF (excluded)
LNG : length of identifier or path
PATH : identifier or path

NB: The command $02 $08 $00 $00 is not supported by CD97 card.

CLASS	INS	DATA OUT	
$02	$08	REND (3)	FCI (x)

REND : execution report
FCI : FCI

UPDATE RECORD (GTML)

Description : Deletion then writing of a record given in a linear fixed structure EF or the most recent recording of a circular file.

CLASS	INS	DATA IN				
$02	$0A	ACCES = $00	SID (1)	NREC (1)	LNG (1)	DATA (x)

ACCES : access mode =$00 (other RFU values).
SID : SID (SID = $00 for usually selected EF)
NREC : recording number
LNG : number of bytes to write (n bytes ≤ size of a recording)
DATA : data to record

CLASS	INS	DATA OUT
$02	$0A	REND (3)

REND : execution report

WRITE RECORD (GTML)

Description : Writing without deletion of a record given in a linear fixed structure EF or the most recent recording of a circular file.

CLASS	INS	DATA IN				
$02	$0C	ACCES =$00	SID (1)	NREC (1)	LNG (1)	DATA (x)

ACCES : access mode =$00 (other RFU values).
SID : SID (SID = $00 for usually selected EF)
NREC : recording number ($01 for a circular file)
LNG : number of bytes to write (n bytes ≤ size of a recording)
DATA : data to record

CLASS	INS	DATA OUT
$02	$0C	REND (3)

REND : execution report

OPEN SECURED SESSION (GTML)

Description : Opening of a certification session.
Returns the paths of the DF (from the MF excluded) of non-ratified applications and the data read in the indicated recording and file.

CLASS	INS	DATA IN		
$02	$10	TYPE (1)	SID (1)	NREC (1)

TYPE : Type of operation
$00 : personalization
$01 : reloading
$02 : validation
SID : SID (SID = $00 for usually selected EF)
NREC : recording number to read
$00 : no recording to read
$01 : reading of a number 1 or unique recording

CLASS	INS	DATA OUT					
$02	$10	REND (3)	NBAPP (1)	PATH1 (2)	…	PATHn (2)	DATA (29)

REND : execution report
NBAPP: 2*n where n is the number of non-ratified applications
PATH1 : path of the first non-ratified application
....
PATHn: path of the n^{th} non-ratified application
DATA : if recording to read, 29 bytes of data

CLOSE SECURED SESSION (GTML)

Description : Closure of the certification session

CLASS	INS	DATA IN
$02	$11	-

No associated input data

CLASS	INS	DATA OUT
$02	$11	REND (3)

REND : execution report

ABORT SECURED SESSION (GTML)

Description : Stop the current certification session. This still allow to continue to dialogue with the badge and, in particular, open a new session.
Note : this function is emulated in the case of the GTML.

CLASS	INS	DATA IN
$02	$12	-

No associated input data

CLASS	INS	DATA OUT
$02	$12	REND (3)

REND : execution report

7.4. CD 97 class (N°= $03)

Class number : $03

Presentation of CD97 Card : The Travel Card 1997 is a remote multi-application ticket card destined for payment of transport and other services.

For the detail of specifications on the CD97 card, refer to document "Travel card(Carte de déplacement) 1997 - Phase 1" Ref. 970516/SE/SDI/CD97_1.DOC.

7.4.1. Main functions

1. Ticketing : handle control of access to public transport networks. This is its main function. This includes the different existing modes of pricing, and should enable the introduction of new ones.

2. Payment : could be used as a means of payment for operators not linked to transport (France Télécom, Ville de Paris(Paris City Hall), Relais H(chain of newspaper stands), etc.).

3. Private : could be used as private badge by any operator, for example, for control of access.

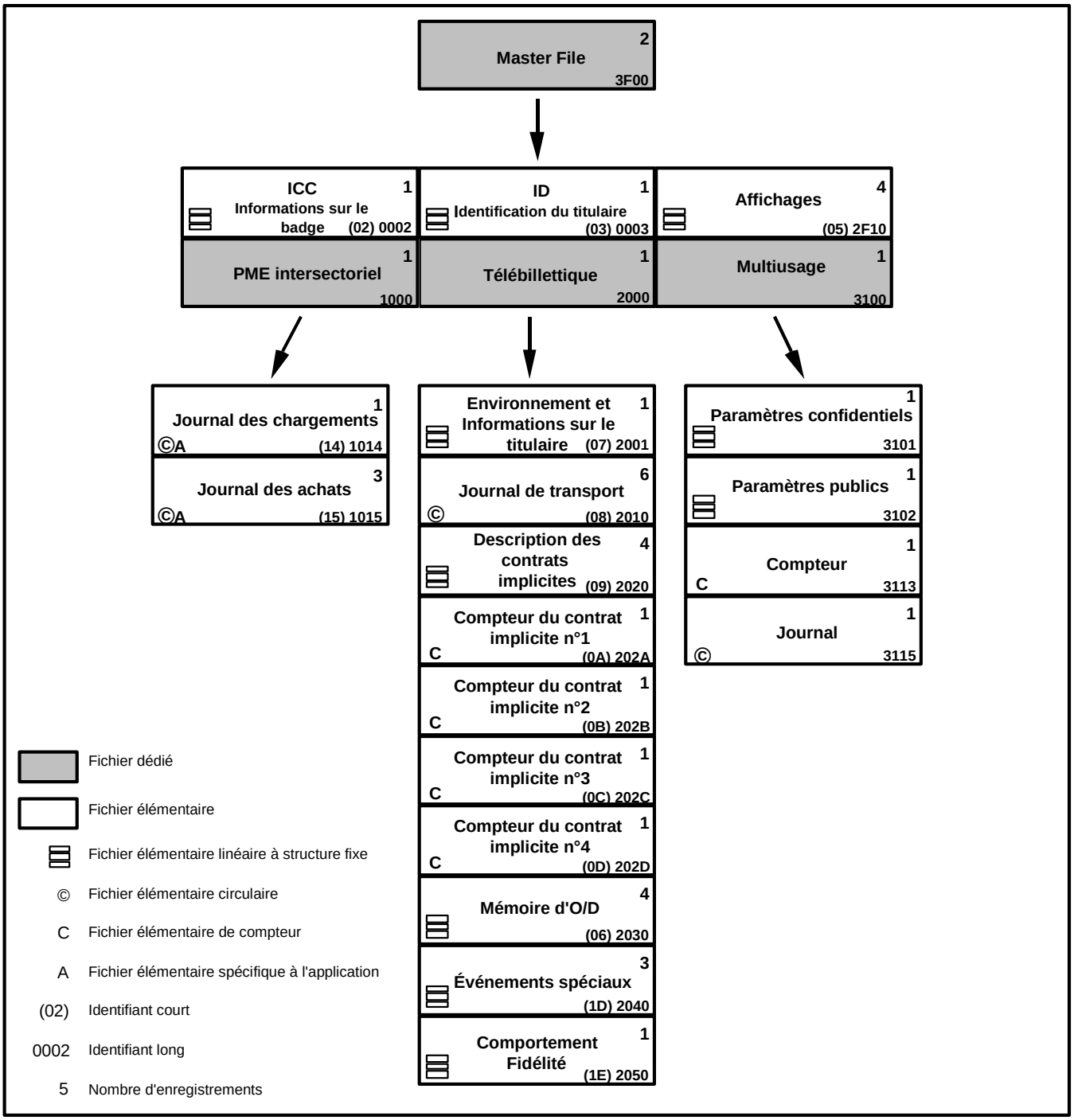
7.4.2. Data structure

The card is organized into files according to the ISO/IEC 7816-4 and takes its inspiration from the EN726-3 norm.

The CD97 card files have various attributes :

- type : DF (dedicated file) or EF (elementary file),
- long identifier (LID),
- short identifier (SID),
- sub type (solely for the EFs) : fixed structure, circular, counter.

The DFs are proper to an application (payment, transport, etc...), they are like folders which contain the EFs.



Definitions common to class functions :

7.4.2.1. Access mode :

Value on 1 byte defining the authentication principles used to access data.

DEFAULT	$00	No local cryptogram for the function. If a session is open, it is the session security that is used. Otherwise only data of which the attribute is "ALWAYS" will be accessible.
PROTECTED	$01	Ground authentication, of incoming data for the badge. Immediate execution by the badge.
STAMPED	$02	Authentication of the badge, of outgoing data for the badge.

Remark : it is impossible to combine the mode « protected » or the mode « stamped » with the session mode.

7.4.2.2. Execution report :

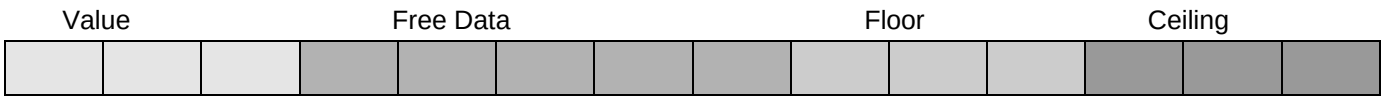
Value on 3 bytes (1 general byte, 2 bytes of details).

Title	byte 0	Bytes 1-2 (detail)	
Execution carried out(no error)	$00	$62 $00 $90 $00	Command accepted execution conditional on good session outcome. Command accepted and executed.
Abnormal execution reported by badge	codes$01 to $09	cf. spec. CD97	
File error	$01	$62 $82 $69 $81 $6A $82 $6A $83 $6A $84 $98 $10	End of recording attained. Incompatibility with file structure. File not found Recording not found. Memory insufficient in the file. Application DF or MF invalidated.
Badge side security error	$02	$69 $82 $63 $00 $63 $CX $69 $83 $98 $00	Security conditions non respected (no alea, cryptogram absent, no current session, access condition unknown) Certificate incorrect in protected mode. Signature incorrect. X represents the number of attempts still authorized (0, 1 or 2) Command rejected since maximum number of errors (3) was reached (PIN access only) PME Certificate incorrect.
Session Error	$03	$69 $86 $64 $00 $69 $85 (mode NEVER).	Command not authorized (no current EF). EEPROM capacity insufficient for the session. Access prohibited with this command Session Problem.
Badge physical error	$04	$65 $81 $96 $10/20/30	Memory failure EEPROM Failure.

Error application	$05	$6A $80	Value to deduct or add incorrect. PIN incorrect
		$6A $81	function not handled.
		$6A $87	Lc incompatible with P1-P2
		$6B$00	P1 or P2 incorrect
		$98 $30	P2 incorrect
		$98 $40	P2 of previous GetEP incorrect (or GetEP omitted)
		$98 $60	Operation impossible, PME already at minimum or PME capacity exceeded.
		$67 $00	P3 incorrect
		$68 $00	Unsupported on CD97 and GTML (KVC request on Open Secured Session)
		$6D $00	INS incorrect
Card invalid	$06	$62 $83	DF parent or MF invalidated.
Badge error code unknown	$09	Badge return unknown code	
Abnormal execution reported by security module	codes$20-$29	cf. spec CD97MiniMS	
SAM side security alert	$20	$69 $00	Command not authorized (A counter of key uses or the n° of PME transactions attained its maximum
		$69 $85	Command not authorized (conditions of use nor satisfied)
		$69 $88	Cryptogram incorrect
SAM physical error	$21	$65 $81	Eeprom Problem
SAM instruction error	$22	$94 $10	Value incorrect in incoming data.
		$94 $20	Exceeding of PME badge capacity.
		$6A $83	Recording not found. The key requested is not present in the key file .
		$6A $86	P1-P2 incorrect. Key reference too large.
		$6A $87	Lc incompatibly with P1-P2.
		$64 $00	Execution error.
SAM security alert	$29	SAM return unknown code	
SAM Answer TimeOut	$2F	$00 $EF	No SAM answer
Communication incorrecte badge (abandon dialogue)	$40	byte 0 :	$00 (RUF)
		byte 1 : if	INS code (ISO) of the frame which has failed ($00) multiple command.
SAM communication incorrect	$41	byte 0 :	$00 (RUF)
		byte 1 :	INS code (ISO) of the frame which has failed.
Abnormal execution reported by the CSC controller	$80	$00$00	command unknown$00$01 concatenation not authorized for this function
		$00$02	maximum number of concatenable functions attained
		$00$03	access mode incompatible with the concatenation
		$00$04	access mode prohibited for this function
		$00$05	no session open
		$00$06	session already open
		$00$07	badge response incorrect
		$00$08	command prohibited in session
		$00$09	overflow of transmission buffer to the badge
		$00$0A	SAM key reference version "x" not found
		$00$0B	Change Key prohibited if not preceded by SelectFile
		$00$0C	Change Key prohibited when the new version of key is 0
		$00$0D	the attempt to ChangeKey with a key version >1 failed since the key in the card is version 0 (personalization to do)
		80xx	error in index input parameter $xx

Note : These report values are proper to the class.

7.4.2.3. Structure of counter elementary files:



Value : 3 bytes, binary unsigned, representing the usual counter value.
Free data : 5 bytes, free to use , but forced at $00 by default.
Floor : 3 bytes, binary unsigned, representing the floor of the counter.
Ceiling : 3 bytes, binary unsigned, representing the ceiling of the counter.

7.4.2.4. FCI (file description data) :

Tag $85
Length $17

Value :

SID 1 byte, short identifier, =$00 for the DF and for the MF.
Type 1 byte :
- $01 MF
- $02 DF
- $04 EF

EFType 1 byte, type of elementary file, =$00 for the DF and for the MF.
- $02 Linear fixed structure.
- $04 Circular.
- $08 Counter.
- $10 Specific to the application.

RecSize 1 byte, number of bytes per recording.
=$00 for the DF and MF.

DataSize 1 byte, number of recordings, =$00 for the DF and MF.

AC 4 bytes : the 1st for the command index number 0, the 2nd for the command index number 1, the 3rd for the command index number 2 and 4th for the command index number 3.

The possible access modes are:
- $00 NEVER
- $01 PIN
- $02 PROTECTED
- $0F APPLICATIVE
- $12 PROTECTED ^ SESSION
- $1F ALWAYS
- Other RUF values.

Correspondence chart between command index number and type of file.

.

Command index number	MF and DF	Linear fixed structure EF	EF circular	Counter EF	circular EF specific to the application
0	REHABILITATE	READ RECORD	READ RECORD	READ RECORD	GetEP and READ RECORD
1	INVALIDATE	UPDATE RECORD	UPDATE RECORD	UPDATE RECORD	ReloadEP
2	-	WRITE RECORD	-	DECREASE	DebitEP and UnDebitEP
3	-	-	APPEND RECORD	INCREASE	-

Nkey 4 bytes, key index number to use.

Status 1 byte :

$X0 File valid(still readable if invalidated).
$X1 File invalidated, readable.
$X2 File valid(not readable if invalidated).
$X3 File invalidated, not readable.
$0X No errors of PIN presentation.
$1X One error of PIN presentation.
$3X Two errors of PIN presentation.
$7X Three errors of PIN presentation. Any new presentation of the PIN is rejected.
Other RUF values.

KVC1 1 byte, =$00 for an EF.

KVC2 1 byte, =$00 for an EF.

KVC3 1 byte, =$00 for an EF.

Floor 3 bytes, =$000000 for files other than counter EFs.

Ceiling 3 bytes, =$000000 for files other than counter EFs.

7.4.3.Set of instructions

APPEND RECORD (CD97)

Description : Adds a record to an elementary circular file.
This new recording becomes the first in the file, the old one is suppressed.

CLASS	INS	DATA IN			
$03	$01	ACCES (1)	SID (1)	LNG (1)	DATA (x)

ACCES : access mode
SID : SID (SID = $00 for usually selected EF)
LNG : length of data to record (n bytes ≤ size of a recording)
DATA : data to record

CLASS	INS	DATA OUT
$03	$01	REND (3)

REND : execution report

CHANGE KEY (CD97)

Description : Writes the values of a key in current DF, contained in the MF.
A SelectFile of the MF or DF should be carried out before being able to change a key.
This command can't be used during a session.

Data In:

CLASS	INS	DATA IN						
$03	$02	IKEY (1)	NKEY (1)	TYPE_CMD (1)	CKEY (1)	ALG_TAG (1)	ALG_SAM (1)	IKEY_TAG (1)

IKEY : index number of the key to modify ($01-$03) (for CD97, GTML)
Or index number (in the SAM) of the new key to be loaded in the card

NKEY : new version of the key to modify (different from 0)
For the Personalization, the new key version is $01

TYPE_CMD : $00, short command (compatibility with the former one)
$01, long command

CKEY : Index Number (in the SAM) of the key to encipher the transfer

ALG_TAG: algorithm key card to recopy

ALG_SAM: algorithm of the SAM used

IKEY_TAG: index number of the new key in the card in the DF

Data Out:

CLASS	INS	DATA OUT
$03	$02	REND (3)

REND : execution report

CHANGE PIN (CD97)

Description : Changes PIN value contained in the MF.
A SelectFile of the MF should be carried out before being able to check PIN.
This command can't be used during a session.

Data In:

CLASS	INS	DATA IN								
$03	$02	IKEY $04	KEY_NUM (1)	OLDPIN (4)	NEWPIN (4)	TYPE_CMD (1)	NKEY/ KIF (1)	00/ KVC (1)	ALG (1)	NSAM (1)

IKEY : $04

KEY_NUM: key number ($00 for CD97, GTML and CT2000,
$04 for GTML2 and CD21,
$09 for POPEYE)

OLDPIN : old PIN (used only for CD97)

NEWPIN : new PIN

TYPE_CMD : $00, short command (compatibility with the former one)
$01, long command

NKEY/ KIF: SAM key number to use
or KIF of the key.

00/ KVC: $00 (if NKEY passed in the previous parameter)
or KVC of the Key.

ALG: algorithm of the SAM used (not used for CD97, GTML1 and GTML2).

NSAM : SAM number
$00 : default SAM,
$01, $02, $03 or $04 : logical number of the wanted SAM number

Data Out:

CLASS	INS	DATA OUT
$03	$02	REND (3)

REND : execution report

DECREASE (CD97)

Description **:** Decreases the value contained in a counter file.
Records the associated data.

Remark **:** Executed in session, this function does not return the new value of the counter ; it will be returned by the CloseSecuredSession function

CLASS	INS	DATA IN			
$03	$03	ACCES (1)	SID (1)	VALUE (3)	$00 $00 $00 $00 $00

ACCES : access mode
SID : SID (SID = $00 for usually selected EF)
VALUE: value to deduct (3 bytes, binary number positive or nil)

CLASS	INS	DATA OUT	
$03	$03	REND (3)	VALUE (3)

REND : execution report
outside session:
VALUE: new value (3 bytes, binary number signed)

INCREASE (CD97)

Description : Increases the value contained in a counter file.
Records the associated data.

Remark : Executed in session, this function does not return the new value of the counter ; it will be returned by the CloseSecuredSession function

CLASS	INS	DATA IN			
$03	$04	ACCES (1)	SID (1)	VALUE (3)	$00 $00 $00 $00 $00

ACCES : access mode
SID : SID (SID = $00 for usually selected EF)
VALUE: value to add (3 bytes, binary number positive or nil)

CLASS	INS	DATA OUT	
$03	$04	REND (3)	VALUE (3)

REND : execution report
outside session:
VALUE: new value (3 bytes, binary number signed)

INVALIDATE (CD97)

Description : Invalidates the current DF (and consequently all related files).

CLASS	INS	DATA IN
$03	$05	ACCES (1)

ACCES : access mode

CLASS	INS	DATA OUT
$03	$05	REND (3)

REND : execution report

REHABILITATE (CD97)

Description : Cancels file invalidation.

CLASS	INS	DATA IN
$03	$07	ACCES (1)

ACCES : access mode

CLASS	INS	DATA OUT
$03	$07	REND (3)

REND : execution report

READ RECORD (CD97)

Description : Reading of a record given in a circular EF, a counter or a linear fixed structure EF

CLASS	INS	DATA IN			
$03	$06	ACCES (1)	SID (1)	NREC (1)	LNG (1)

ACCES : access mode
SID : SID (SID = $00 for usually selected EF)
NREC : recording number
LNG : length of data to read

CLASS	INS	DATA OUT	
$03	$06	REND (3)	DATA (x)

REND : execution report
DATA : data read (n bytes)

SELECT FILE (CD97)

Description : Explicit selection of current EF or DF.
This command sends back the file description data.

CLASS	INS	DATA IN		
$03	$08	CNTR (1)	LNG (1)	PATH (x)

CNTR : Selection check
$00, MF
$02, EF in the current DF (identifier bytes 2 to n)
$08, bytes 1 to n : path from MF (excluded)
LNG : length of identifier or of path
PATH : identifier or path

NB: The command $03 $08 $00 $00 is not supported by CD97 card.

CLASS	INS	DATA OUT	
$03	$08	REND (3)	FCI (x)

REND : execution report
FCI : FCI

STATUS (CD97)

Description : ditto SELECT FILE, but without selecting a file.

CLASS	INS	DATA IN		
$03	$09	CNTR (1)	LNG (1)	PATH (x)

CNTR : Selection check
$00, MF
$02, EF in the current DF (identifier bytes 2 to n)
$08, bytes 1 to n : path from MF (excluded)
LNG : length of identifier or of path
PATH : identifier or path

CLASS	INS	DATA OUT	
$03	$09	REND (3)	FCI (x)

REND : execution report
FCI : FCI

UPDATE RECORD (CD97)

Description : Deletion then writing of a record given in a linear fixed structure EF or the most recent recording of a circular file.

CLASS	INS	DATA IN				
$03	$0A	ACCES (1)	SID (1)	NREC (1)	LNG (1)	DATA (x)

ACCES : access mode
SID : SID (SID = $00 for usually selected EF)
NREC : recording number
LNG : number of bytes to write (n bytes ≤ size of a recording)
DATA : data to record

CLASS	INS	DATA OUT
$03	$0A	REND (3)

REND : execution report

VERIFY PIN (CD97)

Description **:** Presentation of PIN with counting of number of incorrect presentations.
A SelectFile of the MF should be carried out before being able to check PIN.
This command can't be used during a session.

Data In:

CLASS	INS	DATA IN					
$03	$0B	MODE (1)	PIN (4)	TYPE_CMD (1)	NKEY/ KIF (1)	00/ KVC (1)	NSAM (1)

MODE : $00, consultation of counter of number of incorrect presentations
$01, presentation of PIN
$02, presentation of PIN in transparent mode for contact communication

PIN : PIN (4 bytes)

TYPE_CMD: $00, short command (compatibility with the former one)
$01, long command

NKEY/ KIF: SAM key number to use
Or KIF of the key.

00/ KVC: $00 if NKEY passed in the previous parameter
or KVC of the Key.

NSAM : SAM number
$00 : default SAM,
$01, $02, $03 or $04 : logical number of the wanted SAM number

Data Out:

CLASS	INS	DATA OUT
$03	$0B	REND (3)

REND : execution report

WRITE RECORD (CD97)

Description : Writing without deletion of a record given in a linear fixed structure EF or the most recent recording of a circular file.

CLASS	INS	DATA IN				
$03	$0C	ACCES (1)	SID (1)	NREC (1)	LNG (1)	DATA (x)

ACCES : access mode
SID : SID (SID = $00 for usually selected EF)
NREC : recording number ($01 for a circular file)
LNG : number of bytes to write (n bytes ≤ size of a recording)
DATA : data to record

CLASS	INS	DATA OUT
$03	$0C	REND (3)

REND : execution report

GET ELECTRONIC PURSE STATUS (CD97)

Description : Informs on the status of PME and prepares purchase or loading, or purchase cancellation.

CLASS	INS	DATA IN
$03	$0E	TYPE (1)

TYPE : type of transaction to carry out
$00 : loading transaction
$01 : purchase transaction
$02 : purchase cancellation

- If loading transaction :

CLASS	INS	DATA OUT		
$03	$0E	REND (3)	PME (3)	JCHAR (22)

REND : execution report

PME : PME balance (Most significant Byte first)

JCHAR : 22 first bytes of most recent recording in loading journal

Date (2)
Money Batch (2)
Equipment Type (1)
Balance After Reloading (3)
Amount (3)
Time (2)
Security Device ID (4)
Security Device Transaction Number (3)
Tag Transaction Number (2)

- If purchase transaction or purchase cancellation :

CLASS	INS	DATA OUT		
$03	$0E	REND (3)	PME (3)	JPAID (19)

REND : execution report

PME : PME balance (Most significant Byte first)

JPAID : 19 first bytes of most recent recording in payments journal.

Amount (2)
Date (2)
Time (2)
Equipment Type (1)
Security Device ID (4)
Security Device Transaction Number (3)
Balance After Purchase (3)
Tag Transaction Number (2)

PURCHASE (CD97)

Description : Purchase with PME

CLASS	INS	DATA IN		
$03	$0D	ACHAT (1)	JPAID (7)	AFF (6)

ACHAT : type of purchase:
$00 : purchase transaction.
$01 : purchase transaction with display according to type of purchase.

JPAID : 7 first bytes of new recording in payments journal

Amount (2 bytes) : hexadecimal value of amount to debit from the card (Remark : this value should be negative).

Date (2 bytes) : Hexadecimal value of debit date. The debit date is usually equal to the number of days since 1/1/97.

Time (2 bytes) : Hexadecimal value of debit time. This time is usually equal to the number of minutes since 00h00m.

Equipment Type (1 bytes) : Hexadecimal value of type of equipment used to make the debit.

Example for a debit of 3 units on 09/12/98 at 15h29 :

Amount :	$FF	$FD
Date :	$02	$C3
Time :	$03	$A1
Equipment Type :	$03	

- If purchase transaction extended with display

AFF : 6 bytes of display (cf. display file).

CLASS	INS	DATA OUT
$03	$0D	REND (3)

REND : execution report

Remark: in an uninitialized card note that you must do First a Null Purchase
Ex: 03 0E 01 0B 00
03 0D 00 00 00 00 00 00 00 02 (See CD97 specification)

CANCEL PURCHASE (CD97)

Description : Cancellation of previous payment done with the PME.

- Cancellation of a transaction without* display

CLASS	INS	DATA IN	
$03	$13	$00	JPAID (7)

JPAID: 7 first bytes of new recording in payments journal

Amount (2)
Date (2)
Time (2)
Equipment Type (1)

Example for the cancellation of a debit of 3 units on 09/12/98 at 15h29 (example PURCHASE) :

Amount :	$00	$03
Date :	$02	$C3
Time :	$03	$A1
Equipment Type :		$03

- Cancellation of a transaction with display

CLASS	INS	DATA IN		
$03	$13	$01	JPAID (7)	AFF (6)

JPAID: 7 first bytes of the new recording in payments journal

Amount (2)
Date (2)
Time (2)
Equipment Type (1)

AFF : 6 bytes of display (cf. display file)

CLASS	INS	DATA OUT
$03	$13	REND (3)

REND : execution report

RELOAD ELECTRONIC PURSE (CD97)

Description : Charge of PME

CLASS	INS	DATA IN	
$03	$0F	CHARG1 (5)	CHARG2 (5)

CHARG1 : 5 first bytes of new record in loading journal
(Date, Money Batch, Equipment Type)

Date (2 bytes) : Hexadecimal value of loading date. The loading date is usually equal to the number of days since 1/1/97.

Money Batch (2 bytes) : Hexadecimal value of type of changes loaded on the card (i.e. : Francs, Euros, Dollars, etc.…)

Equipment Type (1 bytes) : Hexadecimal value of type of equipment used to do the loading.

CHARG2 : 5 bytes, offset [$08..$13], of new record in loading journal
(Amount, Time)

Amount (3 bytes) : hexadecimal value of amount to credit on the card (Remark : this value should be positive).

Time (2 bytes) : Hexadecimal value of loading time. This time is usually equal to the number of minutes since 00h00m.

CLASS	INS	DATA OUT
$03	$0F	REND (3)

REND : execution report

Remark: in an uninitialized card note that you must do First a Null Reload
Ex: 03 0E 00 0A 00
03 0F 00 00 00 02 00 00 00 00 00 00 (See CD97 specification)

OPEN SECURED SESSION (CD97)

Description : Opening of a certification session.

Description : Opening of a certification session.
Returns DF paths (from MF excluded) from non-ratified applications and the data read in the indicated record and file.

Remark : this command should be preceded by a select file command in the directory concerned.

CLASS	INS	DATA IN						
$03	$10	TYPE (1)	SID (1)	NREC (1)	TYPE_CMD (1)	NKEY/ KIF (1)	00/ KVC (1)	MODE (1)

TYPE : Type of operation : $00 : Personalization, $01 : Reloading, $02 : Validation

SID : SID (SID = $00 for usually selected EF)

NREC : recording number to read
$00 : no recording to read
$01: reading of a number 1 or unique recording

TYPE_CMD: $00, short command (compatibility with the former one for CD97 and GTML)
$01, long command

NKEY/ KIF: SAM key number to use
Or KIF of the key.

00/ KVC: $00 if NKEY passed in the previous parameter
or KVC of the Key.

MODE : Mode of operation ($00 : simple mode, $01 : extended mode)

CLASS	INS	DATA OUT						
$03	$10	REND (3)	NBAPP (1)	PATH1 (2)	…	PATHn (2)	DATA (29)	KVC (1)

REND : execution report

NBAPP: 2*n where n is the number of non-ratified applications

PATH1 : path of the first non-ratified application

....

PATHn: path of the nth non-ratified application

DATA : if recording to read, 29 bytes of data

KVC : KVC in extended mode.

CLOSE SECURED SESSION (CD97)

Description **:** Closure of certification session

Example :

For an Increase function carried out in session we obtain:

$00 $90$00 $04 $nnnnnn where $nnnnnn is the new value of the counter.

Data In:

CLASS	INS	DATA IN	
$03	$11	TYPE_CMD (1)	TIMEOUT (1)

TYPE_CMD: $00, session will be ratified at the reception of the following command

$80, session is ratified immediately (except for CD97 and GTML)

$4A, switches OFF the field if the card doesn't answer.

TIMEOUT: if TYPE=$4A

Data Out:

CLASS	INS	DATA OUT	
$03	$11	REND (3)	RESULT (x)

REND : execution report

RESULT: result of incoming/outgoing orders during the session

The format is as follows :

1st byte : function number

following bytes: same format as for a function execution outside session apart from report (3 first bytes).

ABORT SECURED SESSION (CD97)

Description **:**

Stop the current certification session. This still allow to continue a dialogue with the badge and, in particular, open a new session.

CLASS	INS	DATA IN
$03	$12	-

No associated input data

CLASS	INS	DATA OUT
$03	$12	REND (3)

REND : execution report

SELECT ISO APPLICATION

Description : select application using Select File ISO command.

CLASS	INS	DATA IN		
$03	$15	Select Option (1)	LNG (1)	Data (x)

Select Option : 00 : first application or select by name if LNG <> 0.
01 : select last application (LNG should be 0)
02 : select next application (LNG should be 0)
03 : select previoust application (LNG should be 0)

LNG : length of data. 0 if Select Option <> 0, otherwise <= 16.

DATA : application name (LNG bytes)

CLASS	INS	DATA OUT	
$03	$15	REND (3)	FCI (x)

REND : execution report

FCI : FCI

7.5. Certificate Class (N°= $04)

Class number : $04

7.5.1. Set of instructions

CheckCertificate (CTx)

Description: Certificate check

CLASS	INS	DATA IN					
$04	$04	KeyType	Param	LNG	BUFFER(x)	NB	Certificate(x)

KeyType: key type or number
Param: type of algorithm
LNG: buffer length (diversifier + data)
BUFFER: data to check (minimum 12 bytes = 8 bytes diversifier + 4 bytes data)
NB Certificate length (2 or 4 bytes)
Certificate(x): Certificate read in the CTX

CLASS	INS	DATA OUT		
$04	$04	Status	DATA1	DATA2

STATUS: 0x00: bad certificate
0x02: good certificate
0x03: SAM error
0x04: No SAM answer

DATA1 and DATA2 correspond to a SAM error code and are present only if the status is set to 0x03.

giveCertificate (CTX)

Description: certificate generation

CLASS	INS	DATA IN				
$04	$05	KeyType	Param	LNG	BUFFER(x)	NB

KeyType: Key type or number
Param: type of algorithm
LNG: buffer length (diversifier + data)
BUFFER: data (minimum 12 bytes = 8 bytes diversifier + 4 bytes data)
NB Certificate length (2 or 4 bytes)

CLASS	INS	DATA OUT	
$04	$05	Status	Certificate(x)

STATUS: 0x00: bad certificate
0x02: good certificate
Certificate(x): Certificate calculated

7.6. Variable Class Mapping (N°= $05)

Class number : $05

Created to enable use with identical commands for

- **cards with fixed mapping: CD97 RJJ , GTML, GTML2**
- **cards with variable mapping : C97 RJL, CT2000**

7.6.1. Main functions

The CSC Gen2xx coupler handled cards while taking into account a fixed structure of files (case of function classes CD97 (03) and GTML (02)) using a mapping table of the position of keys in the SAM according to their use.
This had the drawback of not being adaptable to new file structures and not being capable of handling several key versions.

So as to be able to handle the new variable file structure cards of the CD97 RJL and CT2000 types, handling by mapping table has been abandoned in this function class. To afford more scope for development of the applications (without changing coupler software), a "NKEY" parameter has been added to commands enabling the application to attribute the different SAM keys to the different card functions, in compliance with the choices retained in Card/SAM personalization.

The user can either communicate with the SAM thanks to a Key Reference or thanks to the KIF (indication of key function) and the KVC (indication of key version).

The error codes restituted by the card and the SAM are not filtered by the CSC and are consequently restituted wholly to the host.

For more information please refer to documents detailing default card and SAM mapping retained for the personalization of cards and SAMs.

7.6.2. Data structure

Not described.

7.6.3. Rules

7.6.3.1. Access mode :

value on 1 byte defining the principles of authentication used to access data.

DEFAULT	$00	No local cryptogram for the function. If a session is open, it is the session security that is used. Otherwise only the data of which the attribute is "ALWAYS" will be accessible.
PROTECTED	$01	Ground authentication, of incoming data for the badge. Immediate execution by the badge.
STAMPED	$02	Authentication of the badge, of outgoing data for the badge.

Remark : it is impossible to combine the mode "protected" or the mode "stamped" with the session mode.

7.6.3.2. Execution report :

Not described as depending on each card. For more information please refer to each card documentation

.

7.6.4.Set of instructions

APPEND RECORD

Description : Adds a record to an elementary circular file. This new record becomes the first in the file, the old one is suppressed.

CLASS	INS	DATA IN						
$05	$01	ACCES (1)	SID (1)	LNG (1)	DATA (LNG)	LID (2)	NKEY / KIF(1)	00 / KVC(1)

ACCES : access mode
SID : SID (SID = $00 for usually selected EF)
LNG : length of data to record (n bytes ≤ size of a recording)
DATA : data to record
LID : LID (used in PROTECTED mode)
NKEY / KIF: SAM key number to use or KIF of the Key (used in PROTECTED mode)
00 / KVC: 00 if NKEY passed in the previous parameter or KVC of the Key (used in PROTECTED mode).

CLASS	INS	DATA OUT
$05	$01	REND (3)

REND : execution report

CHANGE KEY

WARNING! This is a very specific command for personalization. To know how to use it, please refer to the Application Note referenced by *AN-03389-10-ChangeKey.doc .*

This command can't be used during a session.

CLASS	INS	DATA IN	
$05	$02	$05	…

CHANGE PIN

Description : Changes PIN value contained in the MF.
A SelectFile of the MF should be carried out before being able to change the PIN.
This command can't be used during a session.

Data In:

CLASS	INS	DATA IN						
$05	$02	IKEY $04	KEY_NUM (1)	OLDPIN (4)	NEWPIN (4)	NKEY / KIF (1)	00 / KVC (1)	ALG (1)

IKEY : $04

KEY_NUM: key number ($00 for CD97, GTML and CT2000,
$04 for GTML2, CD21, CD Light
$09 for POPEYE)

OLDPIN : old PIN (used only for CD97)

NEWPIN : new PIN

NKEY / KIF: SAM key number to use or KIF of the Key

00 / KVC: 00 if NKEY passed in the previous parameter or KVC of the Key.

ALG: algorithm of the SAM used (for POPEYE card).

Data Out:

CLASS	INS	DATA OUT
$05	$02	REND (3)

REND : execution report

DECREASE

Description : Decreases the value contained in a counter file.

Records the associated data.

Remark : Executed in session, this function can return or not (depending on the card type)

the new value of the counter; for the CD97 card the new value will be returned by the CloseSecuredSession

CLASS	INS	DATA IN						
$05	$03	ACCES (1)	SID (1)	VALUE (3)	LID (2)	Index counter	NKEY / KIF (1)	00 / KVC(1)

ACCES : access mode (default, protected)

SID : SID (SID = $00 for usually selected EF)

VALUE : value to deduct (3 bytes, binary number positive or nil)

LID : LID (used in PROTECTED mode)

Index counter : index of the counter (used in PROTECTED mode)

NKEY / KIF : SAM key number to use

or KIF of the Key (used in PROTECTED mode)

00 / KVC : $00 if NKEY passed in the previous parameter

or KVC of the Key (used in PROTECTED mode).

CLASS	INS	DATA OUT	
$05	$03	REND (3)	VALUE (3)

REND : execution report

VALUE: **outside session, for all cards,** new value (3 bytes, binary number signed)

in session, for all cards except CD97, new value (3 bytes, binary number signed)

INCREASE

Description : Increases the value contained in a counter file. Records the associated data.

Remark : Executed in session, this function can return or not (depending on the card type) the new value of the counter; for the CD97 card the new value will be returned by the CloseSecuredSession

CLASS	INS	DATA IN						
$05	$04	ACCES (1)	SID (1)	VALUE (3)	LID (2)	Index counter	NKEY / KIF (1)	00 / KVC(1)

ACCES : access mode (default, protected)

SID : SID (SID = $00 for usually selected EF)

VALUE : value to add (3 bytes, binary number positive or nil)

LID : LID (used in PROTECTED mode)

Index counter : index of the counter (used in PROTECTED mode)

NKEY / KIF : SAM key number to use
or KIF of the Key (used in PROTECTED mode)

00 / KVC : $00 if NKEY passed in the previous parameter
or KVC of the Key (used in PROTECTED mode).

CLASS	INS	DATA OUT	
$05	$04	REND (3)	VALUE (3)

REND : execution report

VALUE: **outside session, for all cards,** new value (3 bytes, binary number signed)
in session, for all cards except CD97, new value (3 bytes, binary number signed)

INVALIDATE

Description : Invalidates the current DF (and consequently all offspring files).

CLASS	INS	DATA IN			
$05	$05	ACCES (1)	LID (2)	NKEY / KIF (1)	00 / KVC(1)

ACCES : access mode

LID : LID (used in PROTECTED mode)

NKEY / KIF : SAM key number to use
or KIF of the Key (used in PROTECTED mode)

00 / KVC : $00 if NKEY passed in the previous parameter
or KVC of the Key (used in PROTECTED mode).

CLASS	INS	DATA OUT
$05	$05	REND (3)

REND : execution report

REHABILITATE

Description **:** Cancels file invalidation.

CLASS	INS	DATA IN			
$05	$07	ACCES (1)	LID (2)	NKEY / KIF (1)	00 / KVC (1)

ACCES : access mode
LID : LID (used in PROTECTED mode)
NKEY / KIF : SAM key number to use
or KIF of the Key (used in PROTECTED mode)
00 / KVC : $00 if NKEY passed in the previous parameter
or KVC of the Key (used in PROTECTED mode).

CLASS	INS	DATA OUT
$05	$07	REND (3)

REND : execution report

READ RECORD

Description **:** Reading of a record given in a circular EF, a counter or a linear fixed structure EF.

CLASS	INS	DATA IN						
$05	$06	ACCES (1)	SID (1)	NREC (1)	LNG (1)	LID (2)	NKEY / KIF (1)	00 / KVC (1)

ACCES : access mode
SID : SID (SID = $00 for usually selected EF)
NREC **:** recording number
LNG **:** length of data to read
LID : LID (used in STAMPED mode)
NKEY / KIF: SAM key number to use
or KIF of the Key (used in STAMPED mode)
00 / KVC: $00 if NKEY passed in the previous parameter
or KVC of the Key (used in STAMPED mode).

CLASS	INS	DATA OUT	
$05	$06	REND (3)	DATA (x)

REND : execution report
DATA : data read (n bytes)

SELECT FILE

Description : Explicit selection of current EF or DF.

This command sends back the file description data.

CLASS	INS	DATA IN		
$05	$08	CNTR (1)	LNG (1)	PATH (x)

CNTR : Selection check

$00, MF

$02, EF in the current DF (identifier bytes 2 to n)

$08, bytes 1 to n : path from MF (excluded)

LNG : length of identifier or of path

PATH : identifier or path

NB: The command $05 $08 $00 $00 is not supported by CD97 card.

CLASS	INS	DATA OUT	
$05	$08	REND (3)	FCI (x)

REND : execution report

FCI : FCI

STATUS

Description : ditto SELECT FILE, but without selecting a file. (not available on any card)

CLASS	INS	DATA IN		
$05	$09	CNTR (1)	LNG (1)	PATH (x)

CNTR : Selection check

$00, MF

$02, EF in the current DF (identifier bytes 2 to n)

$08, bytes 1 to n : path from MF (excluded)

LNG : length of identifier or of path

PATH : identifier or path

CLASS	INS	DATA OUT	
$05	$09	REND (3)	FCI (x)

REND : execution report

FCI : FCI

UPDATE RECORD

Description : Deletion then writing of a record given in a linear fixed structure EF or the most recent recording of a circular file.

CLASS	INS	DATA IN							
$05	$0A	ACCES (1)	SID (1)	NREC (1)	LNG (1)	DATA (x)	LID (2)	NKEY / KIF (1)	00 / KVC (1)

ACCES : access mode
SID : SID (SID = $00 for usually selected EF)
NREC : recording number
LNG : number of bytes to write (n bytes ≤ size of a recording)
DATA : data to record
LID : LID (used in PROTECTED mode)
NKEY / KIF: SAM key number to use or KIF of the Key (used in PROTECTED mode)
00 / KVC: 00 if NKEY passed in the previous parameter or KVC of the Key (used in PROTECTED mode).

CLASS	INS	DATA OUT
$05	$0A	REND (3)

REND : execution report

VERIFY PIN

Description : Presentation of PIN with counting of number of incorrect presentations.
A SelectFile of the MF should be carried out before being able to check PIN.
This command can't be used during a session.

CLASS	INS	DATA IN			
$05	$0B	MODE (1)	PIN (4)	NKEY / KIF (1)	00 / KVC (1)

MODE : $00, consultation of the counter of incorrect presentations
$01, presentation of PIN in encrypted mode for RF communication
$02, presentation of PIN in transparent mode for contact communication
PIN : PIN (4 bytes)
NKEY / KIF : SAM key number to use or KIF of the Key
00 / KVC : 00 if NKEY passed in the previous parameter or KVC of the Key.

CLASS	INS	DATA OUT
$05	$0B	REND (3)

REND : execution report

WRITE RECORD

Description : Writing without deletion of a record given in a linear fixed structure EF or the most recent recording of a circular file.

CLASS	INS	DATA IN							
$05	$0C	ACCES (1)	SID (1)	NREC (1)	LNG (1)	DATA (x)	LID (2)	NKEY / KIF (1)	00 / KVC (1)

ACCES : access mode
SID : SID (SID = $00 for usually selected EF)
NREC : recording number ($01 for a circular file)
LNG : number of bytes to write (n bytes ≤ size of a recording)
DATA : data to record
LID : LID (used in PROTECTED mode)
NKEY / KIF: SAM key number to use
or KIF of the Key (used in PROTECTED mode)
00 / KVC: $00 if NKEY passed in the previous parameter
or KVC of the Key (used in PROTECTED mode).

CLASS	INS	DATA OUT
$05	$0C	REND (3)

REND : execution report

OPEN SECURED SESSION

Description : Opening of a certification session.
Returns DF paths (from MF excluded) of non-ratified applications
and data read in the indicated record and file.

Remark: this command should be preceded by a select file command in the directory concerned.

CLASS	INS	DATA IN					
$05	$10	TYPE (1)	SID (1)	NREC (1)	NKEY / KIF (1)	00 / KVC(1) or FF	MODE (1)

TYPE : Type of operation : $00 : Personalization, $01 : Reloading, $02 : Validation
SID : SID (SID = $00 for usually selected EF)
NREC : recording number to read
$00 : no recording to read
$01: reading of a number 1 or unique recording
NKEY / KIF : SAM key number to use or KIF of the Key
00 / KVC or FF : * 00 if NKEY passed in the previous parameter
* or KVC of the Key.
* or 0xFF if the KVC is unknown.
MODE : Mode of operation ($00 : simple mode, $01 : extended mode:the KVC used is sent back)
The extended mode is not supported by CD97 and GTML .

CLASS	INS	DATA OUT						
$05	$10	REND (3)	NBAPP (1)	PATH1 (2)	…	PATHn (2)	DATA (29)	KVC (1)

REND : execution report
NBAPP: 2*n where n is the number of non-ratified applications
PATH1 : path of the first non-ratified application
....
PATHn: path of the nth non-ratified application
DATA : if recording to read, 29 bytes of data
KVC : KVC in extended mode.

CLOSE SECURED SESSION

Description : Closure of certification session

Example: For an Increase function executed in session we obtain:

$00 $90$00 $04 $nnnnnn where $nnnnnn is the new value of the counter.

CLASS	INS	DATA IN	
$05	$11	TYPE_CMD (1)	TIMEOUT (1)

TYPE_CMD: $00, session will be ratified at the reception of the following command

$80, session is ratified immediately (except for CD97 and GTML1)

$4A, switches OFF the field if the card doesn't answer.

TIMEOUT: if TYPE=$4A

CLASS	INS	DATA OUT	
$05	$11	REND (3)	RESULT (x)

REND : execution report

RESULT: result of incoming/outgoing orders during the session

The format is as follows :

1er byte : function number

following bytes: same format as for a function execution outside session apart from the report (3 first bytes).

ABORT SECURED SESSION

Description : Stop the current certification session. This still allow to continue to dialogue with the badge and , in particular, open a new session.

CLASS	INS	DATA IN
$05	$12	-

No associated input data

CLASS	INS	DATA OUT
$05	$12	REND (3)

REND : execution report

DECREASE MULTIPLE

Description : Decrease the value contained in several counters.

Remark : This function should be carried out in session and it immediately returns the new value of the counters.

This command is not supported by CD97 and GTML .

CLASS	INS	DATA IN						[	NBC fois		]
$05	$14	ACCES(1)	SID (1)	LID (2)	NKEY / KIF(1)	00 / KVC (1)	NBC(1)	[	CI (1)	VALUE (3)	]

ACCES : access mode.

SID : Short ID

NKEY / KIF : SAM key number to use or KIF of the Key

00 / KVC : 00 if NKEY passed in the previous parameter or KVC of the Key.

NBC : Number of counters concerned (maximum 7)

CI : Counter index

VALUE : value to deduct (3 bytes, binary number positive or nil)

CLASS	INS	DATA OUT	[	NBC fois		]
$05	$14	REND (3)	[	CI (1)	VALUE (3)	]

REND : execution report

CI : Counter index

VALUE: new value (3 bytes, binary number signed)

INCREASE MULTIPLE

Description **:** Increases the value contained in a counter file.

The associated data is not written in the GTML.

Remark : This function should be carried out in session and it immediately returns the new value of the counter.

This command is not supported by CD97 and GTML .

CLASS	INS	DATA IN						[	NBC fois		]
$05	$15	ACCES(1)	SID (1)	LID (2)	NKEY / KIF(1)	00 / KVC (1)	NBC (1)	[	CI (1)	VALUE (3)	]

ACCES : access mode = See $5.8.2.1.

SID : SID (SID = $00 for usually selected EF)

NKEY / KIF : SAM key number to use or KIF of the Key

00 / KVC : 00 if NKEY passed in the previous parameter or KVC of the Key.

NBC : Number of counters concerned (maximum 7)

CI : Counter index

VALUE: value to add (3 bytes, binary number positive or nil)

CLASS	INS	DATA OUT	[	NBC fois		]
$05	$15	REND (3)	[	CI (1)	VALUE (3)	]

REND : execution report

CI : Counter index

VALUE: new value (3 bytes, binary number signed)

LOCK / UNLOCK

Description **:** Locks / Unlocks the card.

REMARQUE : Before this command, a Verify Pin must be carried out,
After this command, the current file is unchanged.

CLASS	INS	DATA IN
$05	$16	CMD (1)

CMD : Type of Command : LOCK = $00 / UNLOCK = $01

CLASS	INS	DATA OUT
$05	$16	REND (3)

REND : execution report

PURCHASE

Description : Purchase with PME

Remark : maintained for compatibility with the CD97

CLASS	INS	DATA IN		
$05	$0D	ACHAT (1)	JPAID (7)	AFF (6)

ACHAT : Type of purchase : $00 : purchase transaction.

$01 : purchase transaction with display according to type of purchase.

JPAID : 7 1st bytes of new recording in payments journal (Amount, Date, Time, Equipment Type)

NB: the Amount field (2 bytes) has to be specified in negative value.

- If extended purchase transaction with display

AFF : 6 bytes of display (cf. display file).

CLASS	INS	DATA OUT
$05	$0D	REND (3)

REND : execution report

Remark: in an uninitialized card note that you must do First a Null Purchase
Ex: 05 0E 01 0B 00
05 0D 00 00 00 00 00 00 00 02 (See CD97 specification)

GET ELECTRONIC PURSE STATUS

Description : Informs on PME status and prepares purchase or loading or a purchase cancellation.

Remark : maintained for compatibility with the CD97

CLASS	INS	DATA IN		
$05	$0E	TYPE (1)	NKEY (1)	RUF (1)

TYPE : type of transaction to do
$00 : loading transaction
$01 : purchase transaction
$02 : purchase cancellation

NKEY: Key Number to use

- If loading transaction :

CLASS	INS	DATA OUT			
$05	$0E	REND (3)	PME (3)	JCHAR (22)	JINFO(6)

REND : execution report

PME : PME balance (Most significant Byte first)

JCHAR : 22 first bytes of most recent recording in loading journal (Date, Money Batch, Equipment Type, Balance After Reloading, Amount, Time, Security Device ID, Security Device Transaction, Number, Tag Transaction Number)

JINFO: 6 bytes of information (Current KVC, Current Card Transaction Number, Previous CryptoLo)

- If purchase transaction or purchase cancellation :

CLASS	INS	DATA OUT			
$05	$0E	REND (3)	PME (3)	JPAID (19)	JINFO(6)

REND : Execution report

PME : PME balance (Most significant Byte first)

JPAID : 19 first bytes of most recent recording in payments journal . (Amount, Date, Time, Equipment Type, Security Device ID, Security Device Transaction Number, Balance After Purchase, Tag Transaction Number)

JINFO: 6 bytes of information (Current KVC, Current Card Transaction Number, Previous CryptoLo)

RELOAD ELECTRONIC PURSE

Description : PME loading

Remark : maintained for compatibility with the CD97

CLASS	INS	DATA IN	
$05	$0F	CHARG1 (5)	CHARG2 (5)

CHARG1 : 5 first bytes of new recording in loading journal (Date, Money Batch, Equipment Type) **See CD97.**

CHARG2 : 5 bytes, offset [$08..$13], of new recording in loading journal (Amount, Time) **See CD97.**

CLASS	INS	DATA OUT
$05	$0F	REND (3)

REND : execution report

Remark: in an un-initialized card note that you must do First a Null Reload
Ex: 05 0E 00 0A 00
05 0F 00 00 00 02 00 00 00 00 00 00 (See CD97 specification)

CANCEL PURCHASE

Description : Cancellation of previous payment carried out with the PME.

Remark : maintained for compatibility with the CD97

- Cancellation of a transaction without display

CLASS	INS	DATA IN	
$05	$13	$00	JPAID (7)

JPAID: 7 1st bytes of new recording in payments journal (Amount, Date, Time, Equipment Type)

- Cancellation of a transaction with display

CLASS	INS	DATA IN		
$05	$13	$01	JPAID (7)	AFF (6)

JPAID: 7 1st bytes of new recording in payments journal (Amount, Date, Time, Equipment Type)

AFF : 6 bytes of display (cf. display file)

NKEY : SAM key number to use

CLASS	INS	DATA OUT
$05	$13	REND (3)

REND : execution report

DECREASE_LG

Description : It is a command for CD97 card only,
Decreases the value contained in a counter file and writes the 5 free data.
Records the associated data.

Remark : Executed in session, the new value of the counter will be returned by the CloseSecuredSession

Data In:

CLASS	INS	DATA IN						
$05	$23	ACCES (1)	SID (1)	VALUE (8)	LID (2)	Index counter (1)	NKEY / KIF (1)	00 / KVC (1)

ACCES : access mode (default, protected)
SID : SID (SID = $00 for usually selected EF)
VALUE : value to deduct (3 bytes, binary number positive or nil)
+ 5 free bytes.
LID : LID (used in PROTECTED mode)
Index counter : index of the counter (used in PROTECTED mode)
NKEY / KIF : SAM key number to use
or KIF of the Key (used in PROTECTED mode)
00 / KVC : $00 if NKEY passed in the previous parameter
or KVC of the Key (used in PROTECTED mode).

Data Out:

CLASS	INS	DATA OUT	
$05	$23	REND (3)	VALUE (3)

REND : execution report
VALUE: **outside session, for all cards,** new value (3 bytes, binary number signed)
in session, nothing

INCREASE_LG

Description : It is a command for CD97 card only,
Increases the value contained in a counter file and writes the 5 free data.
Records the associated data

Remark : Executed in session, the new value of the counter will be returned by the CloseSecuredSession

Data In:

CLASS	INS	DATA IN						
$05	$24	ACCES (1)	SID (1)	VALUE (8)	LID (2)	Index counter (1)	NKEY / KIF (1)	00 / KVC (1)

ACCES : access mode (default, protected)
SID : SID (SID = $00 for usually selected EF)
VALUE : value to add (3 bytes, binary number positive or nil)
+ 5 free bytes.
LID : LID (used in PROTECTED mode)
Index counter : index of the counter (used in PROTECTED mode)
NKEY / KIF : SAM key number to use
or KIF of the Key (used in PROTECTED mode)
00 / KVC : $00 if NKEY passed in the previous parameter
or KVC of the Key (used in PROTECTED mode).

Data Out:

CLASS	INS	DATA OUT	
$05	$24	REND (3)	VALUE (3)

REND : execution report

VALUE: **outside session, for all cards,** new value (3 bytes, binary number signed)
in session, nothing

7.7. CTS256B Class (N°= $06)

Class number : $06

The CTS256B is one of the members of the range of ASK contactless tickets. It has a memory of 256 bits and is intended for applications where active authentication of the ticket is not necessary, for example ticket for immediate use etc ...

7.7.1. Memory organization

The CTS256B is addressed by words of 2 Bytes. The CSC handles Byte level.

EEPROM 16 words

Address	Area	Msb word	lsb
0	1	Byte 1	Byte 0
1	2	Byte 3	Byte 2
2	3	Byte 5	Byte 4
3		Byte 7	Byte 6
4		Byte 9	Byte 8
5	4	Byte 11	Byte 10
6		Byte 13	Byte 12
7		Byte 15	Byte 14
8		Byte 17	Byte 16
9		Byte 19	Byte 18
10		Byte 21	Byte 20
11	5	Byte 23	Byte 22
12		Byte 25	Byte 24
13	6	Byte 27	Byte 26
14		Byte 29	Byte 28
15		Byte 31	Byte 30

For more details refer to CTS256B user manual: “contactless ticket single ride, CSC interface specification V1.0”.

7.7.2.Set of instructions

ACTIVE (CTS265B)

Description: Activates only the CTS ticket and sends back the first 5 blocks (Equivalent to EnterHuntPhase card).

CLASS	INS
$06	$01

CLASS	INS	DATA OUT		
$06	$01	Length	Status	DATA (x)

Length: number of response bytes
Status:
$00: communication interrupted
$01: bad CRC
$0F: success
$40: detection of a card which is not a CTS 256
$80: collision
DATA: data read (Length -1 byte): maximum 8 bytes of series number

READ (CTS256B)

Description: Reading of a number of bytes at a given address.

CLASS	INS	DATA IN	
$06	$02	ADD	NB

ADD: Address of the first reading(0…31), in bytes.
NB: Number of bytes to read, from 1 to 32

CLASS	INS	DATA OUT		
$06	$02	Length	Status	DATA(x)

Length: response length
Status:
$00: communication interrupted
$01: bad CRC
$02: success
$03: invalid parameters
DATA: data read (Length -1 byte): maximum 8 bytes of series number

UPDATE (CTS256B)

Description: Deletion if necessary, recording, then checks by reading bytes written at ADD address.

CLASS	INS	DATA IN			
$06	**$03**	**ADD**	**NB**	**DATA (x)**	**DATAinCTS (x)**

ADD: Address of 1st byte to record (0…31).
NB: Number of bytes to read, from 1 to 32
DATA: Data to record, NB bytes
DATAinCTS Data already read and recorded in the ticket (NB bytes) enables quicker recording.
If the data is not known or is required to be deleted, every byte of DATAinCTS must be set to 0xEE .

CLASS	INS	DATA OUT		
$06	**$03**	**Length**	**Status**	**DATA(x)**

Length: length of written and read data in bytes.
Status:
$00: No response
$01: Bad CRC
$02: Success
$03: invalid parameters
$8x: Security activated (i.e. data written =/= data read)
$82: Security activated + good CRC

DATA: data read (Length -1 byte)

RELEASE (CTS256B)

Description: Deactivation of the CTS.

CLASS	INS	Parameters
$06	**$04**	**Param**

Param: 00 : deactivation of the ticket using the instruction “deactivate”
Others : RFU

CLASS	INS	DATA OUT
$06	**$04**	**Status**

Status:
$00: ticket always active
$02: ticket deactivated

7.8. CTx512x Class (N°= $06)

Class number : $06

This class contains the functions for the CTx512B and for the Mifare® UltraLight (sometimes called CTS 512A) as well.

7.8.1.CTx512B:

The CTx512B is one of the members of the range of ASK contactless tickets. It has a memory of 512 bits and is intended for applications where anticollision between tickets is necessary...
CTM512B contains very high security features

The CTx512B is addressed by words of 2 bytes. The CSC handles byte level.

CTx512B internal EEPROM 64 bytes organisation and physical address:

Address	Area	Msb word lsb		Address	Area	Msb word lsb	
0	1	Byte 1	Byte 0	16	5	Byte 33	Byte 32
1	2	Byte 3	Byte 2	17		Byte 35	Byte 34
2	3	Byte 5	Byte 4	18	6	Byte 37	Byte 36
3		Byte 7	Byte 6	19		Byte 39	Byte 38
4		Byte 9	Byte 8	20	7	Byte 41	Byte 40
5	4	Byte 11	Byte 10	21		Byte 43	Byte 42
6		Byte 13	Byte 12	22	8	Byte 45	Byte 44
7		Byte 15	Byte 14	23		Byte 47	Byte 46
8		Byte 17	Byte 16	24	9	Byte 49	Byte 48
9		Byte 19	Byte 18	25		Byte 51	Byte 50
10	5	Byte 21	Byte 20	26	10	Byte 53	Byte 52
11		Byte 23	Byte 22	27		Byte 55	Byte 54
12		Byte 25	Byte 24	28	11	Byte 57	Byte 56
13		Byte 27	Byte 26	29		Byte 59	Byte 58
14		Byte 29	Byte 28	30		Byte 61	Byte 60
15		Byte 31	Byte 30	31		Byte 63	Byte 62

For more details refer to CTS512B or CTM512B user manual.

7.8.2.Mifare® UltraLight:

The Mifare® UltraLight is a member of the Mifare® family but is considered as a ticket whose characteristics are very close to the CTx521B's ones. That's why the functions of the Mifare® UltraLight belong to the same class as the CTx521B class. But this card has to be searched with an ISO A search and not a Mifare® search.

The communication layer of the UltraLight complies to parts 2 and 3 (but not 4!) of the ISO 14443 A standard. Anticollision and Security features are also implemented.

The 512 bit EEPROM memory is addressed by 4-bytes wide pages, but the CSC handles byte level. It is organised in 16 pages:

BYTE NUMBER	0	1	2	3	Page
Serial Number	SN 0	SN 1	SN 2	BCC0	**0**
Serial Number	SN 3	SN 4	SN 5	SN 6	**1**
Internal / Lock	BCC1	Internal	Lock 0	Lock 1	**2**
OTP	OTP 0	OTP 1	OTP 2	OTP 3	**3**
DATA (read/write)	Data 0	Data 1	Data 2	Data 3	**4**
DATA (read/write)	Data 4	Data 5	Data 6	Data 7	**5**
DATA (read/write)	Data 8	Data 9	Data 10	Data 11	**6**
DATA (read/write)	Data 12	Data 13	Data 14	Data 15	**7**
DATA (read/write)	Data 16	Data 17	Data 18	Data 19	**8**
DATA (read/write)	Data 20	Data 21	Data 22	Data 23	**9**
DATA (read/write)	Data 24	Data 25	Data 26	Data 27	**10**
DATA (read/write)	Data 28	Data 29	Data 30	Data 31	**11**
DATA (read/write)	Data 32	Data 33	Data 34	Data 35	**12**
DATA (read/write)	Data 36	Data 37	Data 38	Data 39	**13**
DATA (read/write)	Data 40	Data 41	Data 42	Data 43	**14**
DATA (read/write)	Data 44	Data 45	Data 46	Data 47	**15**

For more details refer to the Mifare® UltraLight user manual.

ASK SA - 2260, route des Crêtes - BP 337 - 06906 Sophia-Antipolis Cedex - Tel. 04 97 21 40 18 - Fax. 04 92 38 93 21 Web : www.ask-rfid.com

7.8.3.Functions list:

LIST ***(CTx512B ONLY)***

Description: List performs anticollision and answers the serial numbers of all the CTx512B present in the reader field

After *LIST* instruction, all the tickets are in HALT state. Each ticket has to be selected by its serial number before any other command.

After 3 REQT without answer, *LIST* instruction answers Length = $02, status = $00 and NB = $00

If the Field off CTx configuration is active, the field is turned on and a select is done before the command, and the field is turned off after the command.

CLASS	INS	RFU
$06	$20	$00

CLASS	INS	DATA OUT			
$06	$20	Length	status	NB	SERIAL (x)

Length: number of response bytes (Length = $02 + 2 x NB)

status

$00	no ticket
$x1	CTx512B in antenna field
$x2	CTS256B in antenna field
$x3	CTx512B and CTS256B in antenna field
$x4	identification error (chip version or manufacturer)
$x5	mysterious answer
$8x	Time out reached before end of anti-collision: problem occurs

NB:

$00	no ticket
$xx:	number of tickets which are responding in the field

SERIAL: list of serial data: composed by the 2 LSB serial number (address $03).

SELECT ***(CTx512B ONLY)***

Description: Selects a specific ticket by this serial number

If the Field off CTx configuration is active, the field is turned on and a select is done before the command, and the field is turned off after the command.

CLASS	INS	DATA IN	
$06	$21	serial	serial

SERIAL: Serial number (two LSBytes)

CLASS	INS	DATA OUT		
$06	$21	Length	Status	DATA(x)

Length: answer length
Status: $00: No answer
$01: Bad CRC
$02: Success
DATA: Serial number read (should be same as *serial*)

READ ***(CTx512B and MIFARE® ULTRALIGHT)***

Description: Reading of a number of bytes at a given address.
Internally, the reader chooses read or multi-read instruction depending on NB parameter.

If the Field off CTx configuration is active, the field is turned on and a select is done before the command, and the field is turned off after the command (only in CTx512b mode).

CLASS	INS	DATA IN	
$06	$22	Byte Number	NB

Byte Number: Address of the first reading (0…63), in bytes.
NB: Number of bytes to read, from 1 to 64

CLASS	INS	DATA OUT		
$06	$22	Length	Status	DATA(x)

Length: response length
Status:
$00: No answer
$01: Bad CRC
$02: Success
$03: Bad Parameters
Note: For the Mifare® UltraLight if Status is different from $02 or $03 the card will come into the HALT state, so you have to wake it up to perform other transactions.

DATA: data read (Length -1 byte).

UPDATE ***(CTx512B and MIFARE® ULTRALIGHT)***

Description: Deletes, records, then checks by reading the bytes written at ADD address.

If the Field off CTx configuration is active, the field is turned on and a select is done before the command, and the field is turned off after the command (only in CTx512b mode).

CLASS	INS	DATA IN		
$06	$23	Byte Number	NB	DATA (x)

Byte Number: Address of 1st byte to record (0…63)
NB: Number of bytes to update, from 1 to 64
DATA: Data to be updated, NB bytes

CLASS	INS	DATA OUT		
$06	$23	Length	Status	DATA(x)

Length: length of written and read data in bytes.
Status:
$00: No answer
$01: Bad CRC
$02: Success
$03: Bad parameters
$82: Security activated

Note: For the Mifare® UltraLight if Status is different from $02 or $03 the card will come into the HALT state, so you have to wake it up to perform other transactions.

DATA: data read after a successful write (Length -1 byte)

NB for the Mifare® UltraLight exclusively: due to the checking of the written data, if the writing phase has not completed successfully, the Mifare®UltraLight will goes into the HALT state and so, a TimeOut will be sent back.

HALT ***(CTx512B and MIFARE® ULTRALIGHT)***

Description: Halt CTx512x ticket.

If the Field off CTx configuration is active, the field is turned on and a select is done before the command, and the field is turned off after the command (only in CTx512b mode).

CLASS	INS	Parameters
$06	**$24**	**Param**

Param: 00 : deactivation of the ticket using the instruction “deactivate”
Others : RFU

CLASS	INS	DATA OUT
$06	**$24**	**Status**

Status: $00: Ticket still active
$02: Ticket deactivated

WRITE ***(CTx512B and MIFARE® ULTRALIGHT)***

Description: Performs a true "Write" operation: puts bits to "1" if not but cannot write a 0 instead of a 1. It is useful for writing in OTP area or eventually to use the whole Data area as an OTP zone.

If the Field off CTx configuration is active, the field is turned on and a select is done before the command, and the field is turned off after the command (only in CTx512b mode).

CLASS	INS	DATA IN		
$06	$26	Byte Number	NB	DATA (x)

Byte Number: Address of 1st byte to record (0…63)
NB: Number of bytes to write, from 1 to 64
DATA: Data to be written, NB bytes

CLASS	INS	DATA OUT		
$06	$26	Length	Status	DATA(x)

Length: length of written and read data in bytes.
Status:
$00: No answer
$01: Bad CRC
$02: Success
$03: Bad parameters
$82: Security activated
Note: For the Mifare® UltraLight if Status is different from $02 or $03 the card will come into the HALT state, so you have to wake it up to perform other transactions.

DATA: data obtained after the Write operation: the former existing data plus the written data computed with the "OR" operation (Length -1 byte)

AUTHENTICATE ***(CTM512B ONLY)***

Description: authenticates an area of the ticket (8 consecutive bytes), thanks to the Anticlone function of the SAM.

If the Field off CTx configuration is active, the field is turned on and a select is done before the command, and the field is turned off after the command (only in CTx512b mode).

CLASS	INS	DATA IN		
$06	**$27**	**Address**	**KIF / KeyRef**	**KVC / 0x00**

Address: Address of the area to authenticate
KIF / KeyRef: Specifies the KIF or the KeyRef, but if KeyRef, the following byte must be set to 0x00.
KVC / 0x00: Specifies the KVC if KIF has been specified before, if not must be 0x00.

CLASS	INS	DATA OUT		
$06	**$27**	**Length**	**Status**	**DATA(x)**

Length: Status length + Data length.
Status:
$00: No answer (TimeOut)
$01: Unexpected failure
$02: Success
$03: Bad parameters
$04: No current SAM
$05: SAM not initialized
$06: Bas Status SAM
DATA: present only to get the SAM status code back(only when STATUS = $06).

WRITE KEY ***(CTM512B ONLY)***

Description: After the personalization phase, writes a diversified key in the ticket.

If the Field off CTx configuration is active, the field is turned on and a select is done before the command, and the field is turned off after the command (only in CTx512b mode).

CLASS	INS	DATA IN	
$06	$28	KIF / KeyRef	KVC / 0x00

KIF / KeyRef: Specifies the KIF or the KeyRef, but if KeyRef, the following byte must be set to 0x00.

KVC / 0x00: Specifies the KVC if KIF has been specified before, if not must be 0x00.

CLASS	INS	DATA OUT		
$06	$28	Length	Status	DATA(x)

Length: Status length + Data length.

Status:
- $00: No answer (TimeOut)
- $01: Unexpected failure
- $02: Success
- $03: Bad parameters
- $04: No current SAM
- $05: SAM not initialized
- $06: Bas Status SAM
- $07: Writing prohibited

DATA: present only to get the SAM status code back(only when STATUS = $06).

UPDATE_FIELD_ON ***(CTx512B and MIFARE® ULTRALIGHT)***

Description: Deletes, records, then checks by reading the bytes written at ADD address.

If the Field off CTx configuration is active, the field is turned on and a select is done before the command, but is not turned off after the command (only in CTx512b mode).

CLASS	INS	DATA IN		
$06	$29	Byte Number	NB	DATA (x)

Byte Number: Address of 1st byte to record (0…63)
NB: Number of bytes to write, from 1 to 64
DATA: Data to be updates, NB bytes

CLASS	INS	DATA OUT		
$06	$29	Length	Status	DATA(x)

Length: length of written and read data in bytes.
Status:
$00: No answer
$01: Bad CRC
$02: Success
$03: Bad parameters
$82: Security activated

Note: For the Mifare® UltraLight if Status is different from $02 or $03 the card will come into the HALT state, so you have to wake it up to perform other transactions.

DATA: data read after a successful write (Length -1 byte)

7.9. SR / SRI / SRT / SRIX Class

Class number : $06

The SR / SRI / SRT / SRIX are members of the range of ASK contactless tickets. They come with different memory organizations. The SRIX anti-clone functionality is not handled by the firmware as it is France Telecom proprietary algorithm.
These tickets are intended for applications where active authentication of the ticket is not necessary, for example ticket for immediate use etc ...

7.9.1. Memory organization

	SR176	SR512 family (SRI512 / SRT512 / SRIX512)	SR4K family (SRI4K/SRIX4K)
User EEPROM (bits)	176	512	4096
Mapping (blocks * bits)	11 x 16	16 x 32	128 x 32
UID (bits)	64	64	64

For more details, refer to SR / SRI / SRT / SRIX user manuals.

This chips family is addressed by blocks of 2 bytes (SR176) or 4 bytes (SR512 and SR4K families).

The CSC firmware handles byte level as well as the native blocks organization.

7.9.2.Set of instructions

ACTIVE (SR Family)

Description: Activate and select a SR, SRI, SRT or SRIX ticket and send back the chip type and the 64-bit UID.

CLASS	INS
$06	$31

CLASS	INS	DATA OUT			
$06	$31	Length	Status	Chip type	UID

Length: number of response bytes (here $0A)

Status:
$00: communication interrupted
$01: bad CRC
$0F: success
$80: collision

Chip type:
$00: SR176
$01: SR512
$02: SR4K

UID: 64-bit (8-byte) UID from LSB to MSB (UID0, UID1… UID7)

READ BLOCKS (SR Family)

Description: Read blocks.

CLASS	INS	DATA IN	
$06	$32	BLOCK	NB

BLOCK : Fist block number to read
SR176 : 0 to 15 ($00 to $0F)
SR512 family : 0 to 15 ($00 to $0F) or 255 ($FF)
SR4K family : 0 to 127 ($00 to $7F) or 255 ($FF)

NB: Number of blocks to read
SR176 : 1 to 16 ($01 to $10)
SR512 family : 1 to 16 ($01 to $10)
SR4K family : 1 to 60 ($01 to $3C)

Note : SR176 has 2-byte blocks, SR512 and SR4K families have 4-byte blocks.

CLASS	INS	DATA OUT		
$06	$32	Length	Status	DATA(x)

Length: response length
Status: $00: communication interrupted
$01: bad CRC
$02: success
$03: bad parameters
DATA: data read (Length -1 byte)

WRITE BLOCKS (SR Family)

Description: Write and verify blocks.

CLASS	INS	DATA IN		
$06	$33	BLOCK	NB	DATA(x)

BLOCK: First block number to write
SR176 : 0 to 15 ($00 to $0F)
SR512 family : 0 to 15 ($00 to $0F) or 255 ($FF)
SR4K family : 0 to 127 ($00 to $7F) or 255 ($FF)

NB: Number of blocks to write
SR176 : 1 to 16 ($01 to $10)
SR512 family : 1 to 16 ($01 to $10)
SR4K family : 1 to 60 ($01 to $3C)

DATA: Data to write, NB bytes, multiple of block size.

Note : SR176 has 2-byte blocks, SR512 and SR4K families have 4-byte blocks.

CLASS	INS	DATA OUT		
$06	$33	Length	Status	DATA(x)

Length: response length
Status:
$00: communication interrupted
$01: bad CRC
$02: success
$03: bad parameters
$8x: Security activated (i.e. data written =/= data read)
$82: Security activated + good CRC

DATA: data read back (Length -1 byte)

RELEASE (SR Family)

Description: Deactivation.

CLASS	INS	Parameters
$06	**$34**	**Param**

Param: 00 : deactivation of the ticket using the “completion” instruction.
Others : RFU

CLASS	INS	DATA OUT
$06	**$34**	**Status**

Status: $00: ticket always active
$02: ticket deactivated

READ (SR Family)

Description: Read bytes at a given address. This function handles the chip regardless the blocks organization.

CLASS	INS	DATA IN	
$06	**$35**	**ADD(2)**	**NB**

ADD: Address of the first reading : 2 bytes LSB, MSB
SR176 : 0 to 31 ($00 00 to $1F 00)
SR512 family : 0 to 63 ($00 00 to $3F 00) or 1020 to 1023 ($FC 03 to $FF 03)
SR4K family : 0 to 511 ($00 00 to $FF 01) or 1020 to 1023 ($FC 03 to $FF 03)

Note : ADD must be multiple of block size.

NB: Number of bytes to read
SR176 : 1 to 32 ($01 to $20)
SR512 family : 1 to 64 ($01 to $40)
SR4K family: 1 to 240 ($01 to $F0)

CLASS	INS	DATA OUT		
$06	**$35**	**Length**	**Status**	**DATA(x)**

Length: response length
Status: $00: communication interrupted
$01: bad CRC
$02: success
$03: bad parameters
DATA: data read (Length -1 byte)

WRITE (SR Family)

Description: Write and verify bytes at a given address. This function handles the chip regardless the blocks organization.

CLASS	INS	DATA IN		
$06	$36	ADD(2)	NB	DATA(x)

ADD: Address of the first writing : 2 bytes LSB, MSB
SR176 : 0 to 31 ($00 00 to $1F 00)
SR512 family : 0 to 63 ($00 00 to $3F 00) or 1020 to 1023 ($FC 03 to $FF 03)
SR4K family : 0 to 511 ($00 00 to $FF 01) or 1020 to 1023 ($FC 03 to $FF 03)

Note : ADD must be multiple of block size.

NB: Number of bytes to write
SR176 : 1 to 32 ($01 to $20)
SR512 family : 1 to 64 ($01 to $40)
SR4K family : 1 to 240 ($01 to $F0)

Note : NB must be multiple of block size.

DATA: Data to write, NB bytes

CLASS	INS	DATA OUT		
$06	$36	Length	Status	DATA(x)

Length: response length
Status:
$00: communication interrupted
$01: bad CRC
$02: success
$03: bad parameters
$8x: Security activated (i.e. data written =/= data read)
$82: Security activated + good CRC

DATA: data read back (Length -1 byte)

7.10. Universal Transit Mapping Management (N°= $08)

Class number : $08

This card has a variable mapping capability

7.10.1. Main functions

This class allows dealing with the Universal Transit cards (using a Universal Transit SAM for security management).

The CSC firmware manages all security aspect of the communication.

The error codes restituted by the card and the SAM are not filtered by the CSC and are consequently restituted wholly to the host.

For more information please refer to documents detailing UT card and SAM mapping retained for the personalization of cards and SAMs.

All information regarding data structure, rules, access mode and execution report can be found in “STE030006 – Morpheus External Specification”.

7.10.2. Set of instructions

CREATE FILE

Description : Create a record to an elementary circular or fixed EF record.

CLASS	INS	DATA IN				
$08	$01	SAM (1)	NKEY / KIF(1)	00 / KVC(1)	LG (1)	DATA (LG)

SAM : SAM Number
NKEY / KIF : SAM key number to use or KIF of the Key
00 / KVC : 00 if NKEY passed in the previous parameter or KVC of the Key.
LG : Size of the data for File Control Parameters (FCP).
DATA : TLVs for FCP

Tag	Length	Information
---	------	----------
$80	$02	Number of Data bytes in the file
$82	$01/02	File Descriptor File Type Record Length
$83	$02	File ID
$84	$05-10	AID
$85	$04	Proprietary Information (Secret File) Key Bitmap (1 byte) Mifare Password Bitmap (2 bytes) PIN Presence (1 byte)
$85	$02	Proprietary Information (DF with AID) Authorized Protocol AFI
$86	$08	Access Conditions

CLASS	INS	DATA OUT
$08	$01	REND (3)

REND : execution report (good execution report is $00 $90 $00)

N.B.: Some fields are not mandatory.

LOAD SECRET

Description : Load keys and passwords.

CLASS	INS	DATA IN							
$08	$02	SAM (1)	NKEY / KIF(1)	00 / KVC(1)	NKEY / KIF(1)	00 / KVC(1)	SEC TYPE(1)	SEC NUM(1)	[PIN] (5)

SAM : SAM Number
NKEY / KIF : SAM key number or KIF of the Key to use for transfer ciphering.
00 / KVC : 00 if NKEY passed in the previous parameter or KVC of the Key, to use for transfer ciphering.

NKEY / KIF : SAM key number or KIF of the secret reference
00 / KVC : 00 if NKEY passed in the previous parameter or KVC of the secret reference.

SEC TYPE : Secret Type (0x01 for PIN code, 0x02 for TDES keys, 0x04 for MiFare pwd)
SEC NUM : Secret number [conditional]
PIN : PIN code and preset [conditional]

CLASS	INS	DATA OUT
$08	$02	REND (3)

REND : execution report (good execution report is $00 $90 $00)

DECREASE

Description : Decrease a counter.

CLASS	INS	DATA IN							
$08	$03	SAM (1)	NKEY / KIF(1)	00 / KVC(1)	CKEY (1)	ACCESS (1)	SID (1)	NBCPT (1)	CPT (n)

SAM : SAM Number
NKEY / KIF : SAM key number to use or KIF of the Key
00 / KVC : 00 if NKEY passed in the previous parameter or KVC of the Key.
CKEY : Card key number
ACCESS : Access condition
SID : SID
NBCPT : Number of counter to decrease (1 to 8)
CPT : number of the counter and the value to decrease

CLASS	INS	DATA OUT
$08	$03	REND (3)

REND : execution report (good execution report is $00 $90 $00)

INCREASE

Description : Increase a counter.

CLASS	INS	DATA IN							
$08	$04	SAM (1)	NKEY / KIF(1)	00 / KVC(1)	CKEY (1)	ACCESS (1)	SID (1)	NBCPT (1)	CPT (n)

SAM : SAM Number
NKEY / KIF : SAM key number to use or KIF of the Key
00 / KVC : 00 if NKEY passed in the previous parameter or KVC of the Key.
CKEY : Card key number
ACCESS : Access condition
SID : SID
NBCPT : Number of counter to increase (1 to 8)
CPT : number of the counter and the value to increase

CLASS	INS	DATA OUT
$08	$04	REND (3)

REND : execution report (good execution report is $00 $90 $00)

APPLICATION BLOCK

Description : Block an application (invalidate)???.

CLASS	INS	DATA IN				
$08	$05	SAM (1)	NKEY / KIF(1)	00 / KVC(1)	CKEY (1)	ACCESS (1)

SAM : SAM Number
NKEY / KIF : SAM key number to use or KIF of the Key
00 / KVC : 00 if NKEY passed in the previous parameter or KVC of the Key.
CKEY : Card key number
ACCESS : Access condition (Always, Transaction, Protected)

CLASS	INS	DATA OUT
$08	$05	REND (3)

REND : execution report (good execution report is $00 $90 $00)

APPLICATION UNBLOCK

Description : Unblock an application (Revalidate)???.

CLASS	INS	DATA IN				
$08	$06	SAM (1)	NKEY / KIF(1)	00 / KVC(1)	CKEY (1)	ACCESS (1)

SAM : SAM Number
NKEY / KIF : SAM key number to use or KIF of the Key
00 / KVC : 00 if NKEY passed in the previous parameter or KVC of the Key.
CKEY : Card key number
ACCESS : Access condition (Always, Transaction, Protected)

CLASS	INS	DATA OUT
$08	$06	REND (3)

REND : execution report (good execution report is $00 $90 $00)

READ DATA

Description : Read Data record in the card.

CLASS	INS	DATA IN							
$08	$07	SAM (1)	NKEY / KIF(1)	00 / KVC(1)	CKEY (1)	ACCESS (1)	SID (1)	NB (1)	EF (n)

SAM : SAM Number
NKEY / KIF : SAM key number to use or KIF of the Key
00 / KVC : 00 if NKEY passed in the previous parameter or KVC of the Key.
CKEY : Card key number
ACCESS : Access condition (Always, Transaction, Protected)
SID : SID
NB : Nb byte to read
EF : EF Number or offset TLV Coded with

- Tag $70 if EF number or
- Tag $77 if offset

CLASS	INS	DATA OUT	
$08	$07	REND (3)	DATA (n)

REND : execution report (good execution report is $00 $90 $00)
DATA : Data read

READ KEY PARAMETERS

Description : Read Key Parameters in the card.

CLASS	INS	DATA IN
$08	$08	SEC NUM (1)

SEC NUM : Secret Number

CLASS	INS	DATA OUT	
$08	$08	REND (3)	DATA (n)

REND : execution report (good execution report is $00 $90 $00)
DATA : Key Parameters read

GET TRACABILITY

Description : Retreive information from the card or the chip.

CLASS	INS	DATA IN
$08	$09	INFO (1)

INFO : CHIP or Card info ($00 for chip)

CLASS	INS	DATA OUT
$08	$09	REND (3)

REND : execution report (good execution report is $00 $90 $00)

SELECT FILE

Description **:** Select an EF.

CLASS	INS	DATA IN		
$08	$0A	SAM (1)	TYPE (1)	AID / LID (n)

SAM : SAM Number
TYPE : Selection Type (P1)
AID / LID : AID or LID in TLV (Tag = $80 for AID and $88 for LID)

CLASS	INS	DATA OUT
$08	$0A	REND (3)

REND : execution report (good execution report is $00 $90 $00)

AUTHENTICATION

Description **:** Authenticate the Application.

CLASS	INS	DATA IN			
$08	$0B	SAM (1)	NKEY / KIF(1)	00 / KVC(1)	CKEY (1)

SAM : SAM Number
TYPE : Selection Type (P1)
NKEY / KIF : SAM key number to use or KIF of the Key
00 / KVC : 00 if NKEY passed in the previous parameter or KVC of the Key.
CKEY : Card key number

CLASS	INS	DATA OUT
$08	$0B	REND (3)

REND : execution report (good execution report is $00 $90 $00)

VERIFY PIN

Description **:** Verify Pin of the card.

CLASS	INS	DATA IN
$08	$0C	PIN(4)

PIN : PIN Code

CLASS	INS	DATA OUT
$08	$0C	REND (3)

REND : execution report (good execution report is $00 $90 $00)

WRITE DATA

Description : Read Data record in the card.

CLASS	INS	DATA IN							
$08	$0D	SAM (1)	NKEY / KIF(1)	00 / KVC(1)	CKEY (1)	ACCESS (1)	SID (1)	NB (3 or 4)	DATA (n)

SAM : SAM Number
NKEY / KIF : SAM key number to use or KIF of the Key
00 / KVC : 00 if NKEY passed in the previous parameter or KVC of the Key.
CKEY : Card key number
ACCESS : Access condition (Always, Transaction, Protected)
SID : SID (or $00 for current EF)
NB : Nb byte to Write in TLV form : Numrec (1 byte) or offset (2 bytes)
DATA : DATA to write TLV Coded with Tag $70 if numrec or $77 if offset

CLASS	INS	DATA OUT	
$08	$0D	REND (3)	DATA (n)

REND : execution report (good execution report is $00 $90 $00)
DATA : Data read

OPEN TRANSACTION

Description : Open a transaction session.

CLASS	INS	DATA IN				
$08	$0E	SAM (1)	NKEY / KIF(1)	00 / KVC(1)	CKEY (1)	TYPE (1)

SAM : SAM Number
NKEY / KIF : SAM key number to use or KIF of the Key
00 / KVC : 00 if NKEY passed in the previous parameter or KVC of the Key.
CKEY : Card key number
TYPE : Type of transaction ($00 if no session key, $01 otherwise)

CLASS	INS	DATA OUT
$08	$0E	REND (3)

REND : execution report (good execution report is $00 $90 $00)

CLOSE TRANSACTION

Description **:** Close a transaction session.

CLASS	INS	DATA IN
$08	$0F	TYPE (1)

TYPE : Type of End of transaction with or without aborting ($00 if transaction aborted)

CLASS	INS	DATA OUT
$08	$0F	REND (3)

REND : execution report (good execution report is $00 $90 $00)

LOCK

Description **:** Lock card.

CLASS	INS
$08	$10

CLASS	INS	DATA OUT
$08	$10	REND (3)

REND : execution report (good execution report is $00 $90 $00)

UNLOCK

Description **:** Unlock card.

CLASS	INS
$08	$11

CLASS	INS	DATA OUT
$08	$11	REND (3)

REND : execution report (good execution report is $00 $90 $00)

UPDATE DATA

Description : Update Data record in the card.

CLASS	INS	DATA IN							
$08	$12	SAM (1)	NKEY / KIF(1)	00 / KVC(1)	CKEY (1)	ACCESS (1)	SID (1)	NB (3 or 4)	DATA (n)

SAM : SAM Number
NKEY / KIF : SAM key number to use or KIF of the Key
00 / KVC : 00 if NKEY passed in the previous parameter or KVC of the Key.
CKEY : Card key number
ACCESS : Access condition (Always, Transaction, Protected)
SID : SID (or $00 for current EF)
NB : Nb byte to Write in TLV form : Numrec (1 byte) or offset (2 bytes)
DATA : DATA to write TLV Coded with Tag $70 if numrec or $77 if offset

- Tag $70 if NumRec (for example $70 $01 $02)
- Tag $77 if Offset (for example $77 $02 $01 $36)

CLASS	INS	DATA OUT
$08	$12	REND (3)

REND : execution report (good execution report is $00 $90 $00)

APPEND RECORD

Description : Append record in the card.

CLASS	INS	DATA IN							
$08	$13	SAM (1)	NKEY / KIF(1)	00 / KVC(1)	CKEY (1)	ACCESS (1)	SID (1)	NB (1)	DATA (nb)

SAM : SAM Number
NKEY / KIF : SAM key number to use or KIF of the Key
00 / KVC : 00 if NKEY passed in the previous parameter or KVC of the Key.
CKEY : Card key number
ACCESS : Access condition (Always, Transaction, Protected)
SID : SID (or $00 for current EF)
NB : Nb byte to Write
DATA : DATA to write

CLASS	INS	DATA OUT
$08	$13	REND (3)

REND : execution report (good execution report is $00 $90 $00)

VERIFY SECRET

Description : Verify the Secret.

CLASS	INS		
$08	$14	NB (1)	DATA (nb)

NB : Secret Nb data

DATA : Secret

CLASS	INS	DATA OUT
$08	$14	REND (3)

REND : execution report (good execution report is $00 $90 $00)

SELECT DIVERSIFIER

Description : Select the Diversifier.

CLASS	INS
$08	$15

The used diversifier is obtained by the Select Application performed during Enter Hunt Phase sequence (01 03 command)

For a correct application selection, use the Enter Hunt Phase Parameters function (01 17 command)

These parameters are kept until next reset or power down.

Ex:

01 17 01 01 00 00 01 00 03 05 11 22 33 44 55 → select application, name starting from “11 22 33 44 55”, unsing “00” for APDU class.

CLASS	INS	DATA OUT
$08	$15	REND (3)

REND : execution report (good execution report is $00 $90 $00)

7.11. MIFARE® Class (N°= $10)

Class number : $10

The MIFARE® Class is based on the 14443 type A contactless protocol, it allows communicating and managing the MIFARE® Classic and MIFARE® 4K cards. Once in communication with a MIFARE® card it is not possible to communicate in the same time with a Type B card.
In addition, this class allow the management of the RCXXX chip and RF baurates.

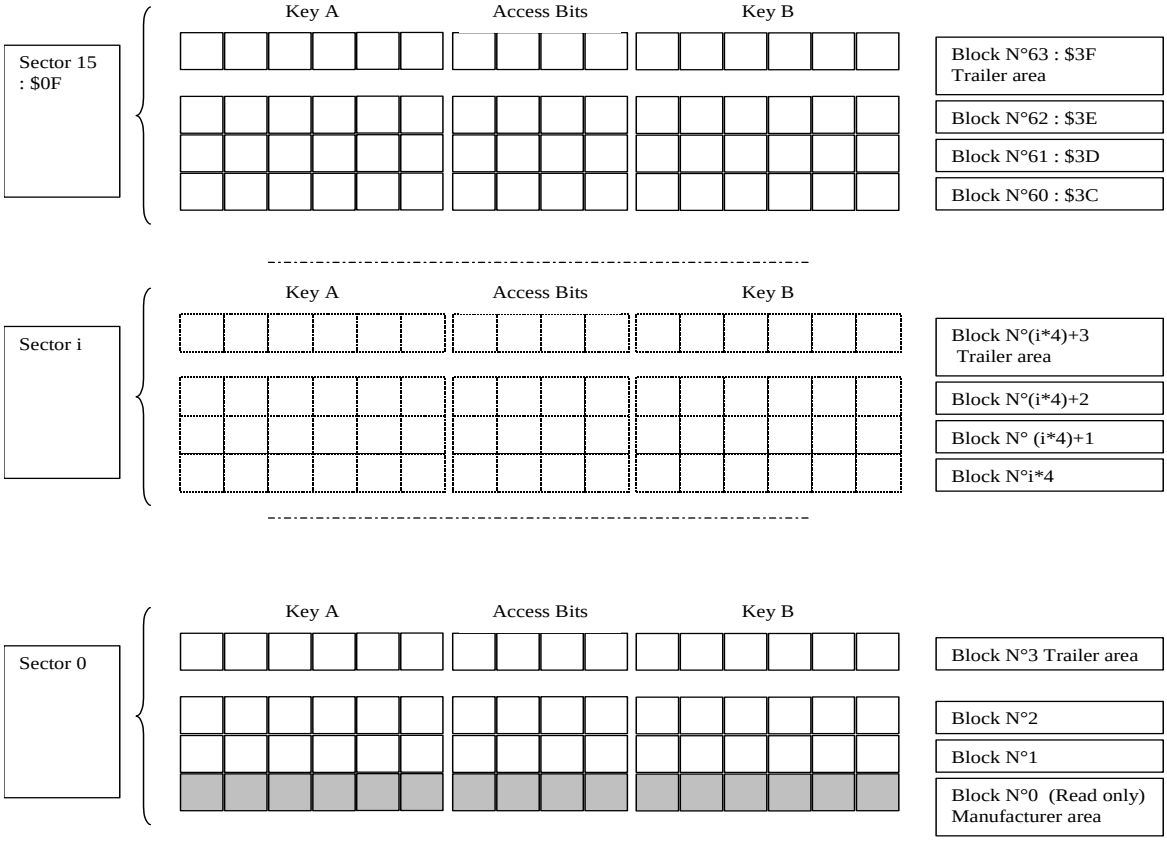
7.11.1. Memory organization

9.9.4.1. Mifare® Classic Cards

The MIFARE® Classic Card is composed of 16 Sectors of 4 blocks each. Each Block is composed of 16 Bytes.

The First Block of the First Sector is a read only block that contains the manufacturer information.

The Fourth Block of each sector contains a Trailer area, which contains the key and access bits for this sector.



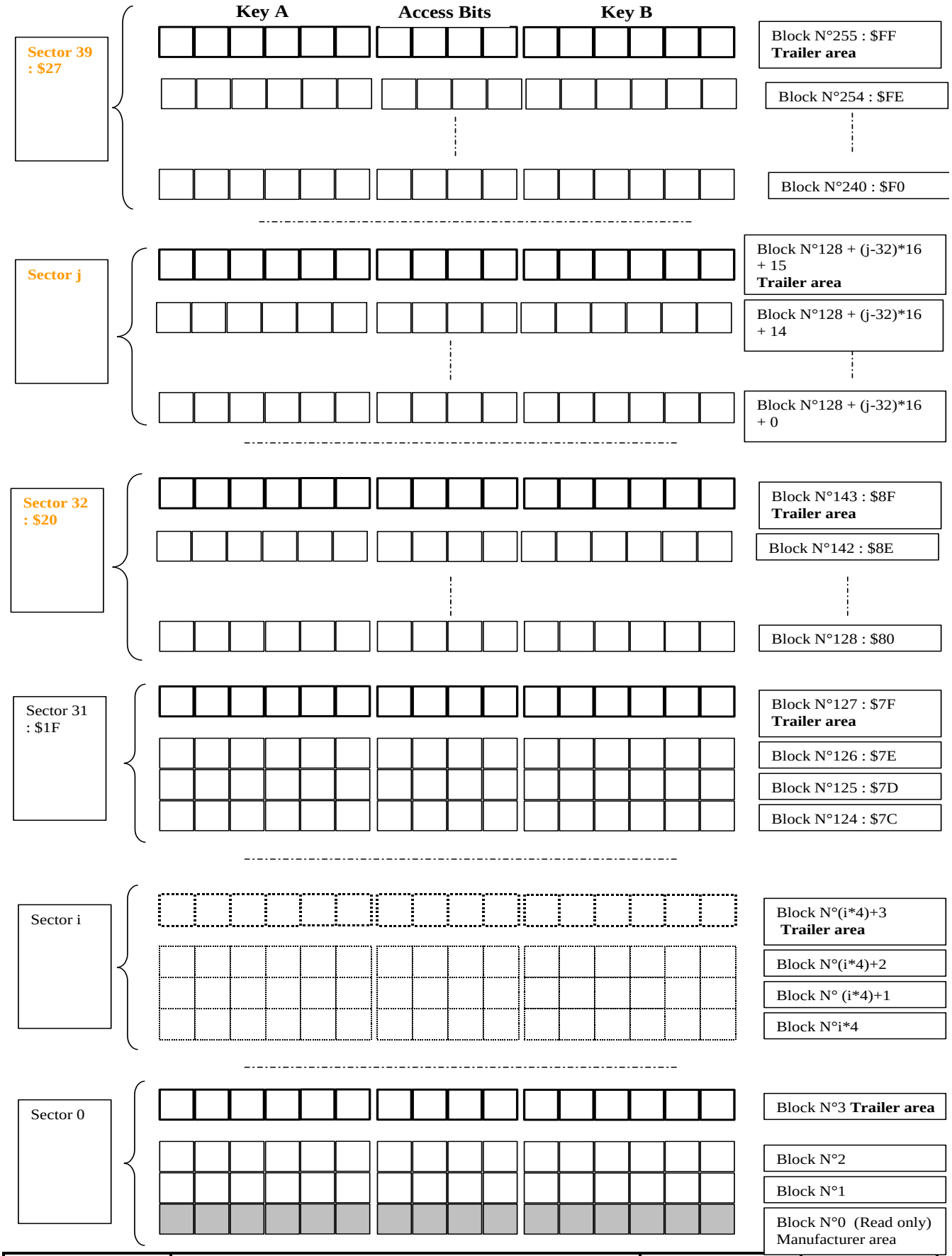
For more details refer to the **MIFARE® STANDARD** user manual.

9.9.4.2. Mifare® 4K Cards

The MIFARE® 4K Card is organised in 32 sectors with 4 blocks and 8 sectors with 16 blocks. Each Block is composed of 16 Bytes.

The First Block of the First Sector is a read only block that contains the manufacturer information.

The Fourth Block of the first 32 sectors and the sixteenth of the last 8 sectors contains a Trailer area, which contains the key and access bits for this sector.



For more details refer to the **MIFARE**® **4K** user manual.

9.9.4.3. Remarks

- The commands **Authenticate**, **ReadSector**, **ChangeKey** authenticates a sector and load the PCD key specified in the internal memory of the MFRC500 chip familly.
- The commands **ReadBlock**, **ReadMultipleBlock**, **WriteBlock**, **IncrementValue**, **DecrementValue** and **BackupRestoreValue** need to be preceded by one of the previous command in order to authenticate the card sector.
- The cards can be detected in the field either by the command EnterHuntPhase or by the command DetectMF.
- For anticollision detection, the command DetectMF indicates the presence of more than one card by its status (0x18, just as the EnterHuntPhase in case of mono-card search) . In multicard mode, to select one specific card among all the cards present, the command selectMF must be used otherwise only the last detected card will be selected.

ASK SA - 2260, route des Crêtes - BP 337 - 06906 Sophia-Antipolis Cedex - Tel. 04 97 21 40 18 - Fax. 04 92 38 93 21 Web : www.ask-rfid.com

7.11.2. Access bit management

The Access Bits are coded redundantly on the bytes 6, 7 and 8 of the trailer block one of the 2 times in complement:

C1	C2	C3	Valid commands	Description
C1/3	C2/3	C3/3	Read/Write	Sector Trailer
C1/2	C2/2	C3/2	Read/Write, Increment/Decrement, Transfert/Restore	Data Block
C1/1	C2/1	C3/1	Read/Write, Increment/Decrement, Transfert/Restore	Data Block
C1/0	C2/0	C3/0	Read/Write, Increment/Decrement, Transfert/Restore	Data Block

Access Bits

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 6	C2/3	C2/2	C2/1	C2/0	C1/3	C1/2	C1/1	C1/0
Byte 7	C1/3	C1/2	C1/1	C1/0	C3/3	C3/2	C3/1	C3/0
Byte 8	C3/3	C3/2	C3/1	C3/0	C2/3	C2/2	C2/1	C2/0

A bad coding of the access block "kills" the block access definitively.

The coding has a different meaning depending if the block is a trailer block or not.

Operation on Data Blocks

C1	C2	C3	Read	Write	Increment	Decrement Transfert Restore	Application
0	0	0	Key A|B	Key A|B	Key A|B	Key A|B	Transport Configuration
0	1	0	Key A|B	Never	Never	Never	Read/Write Block
1	0	0	Key A|B	Key B	Never	Never	Read/Write Block
1	1	0	Key A|B	Key B	Key B	Key A|B	ValueBlock
0	0	1	Key A|B	Never	Never	Key A|B	ValueBlock
0	1	1	Key B	Key B	Never	Never	Read/Write Block
1	0	1	Key B	Never	Never	Never	Read/Write Block
1	1	1	Never	Never	Never	Never	Read/Write Block

Key and access Operation on trailer block

C1	C2	C3	Key A		Access Bits		Key B		Remark
			Read	Write	Read	Write	Read	Write	
0	0	0	never	Key A	Key A	never	Key A	Key A	KeyB can't authenticate
0	1	0	never	never	Key A	never	Key A	never	KeyB can't authenticate
1	0	0	never	Key B	Key A|B	never	never	Key B	
1	1	0	never	never	Key A|B	never	never	never	
0	0	1	never	Key A	Key A	Key A	Key A	Key A	KeyB can't authenticate
0	1	1	never	Key B	Key A|B	Key B	never	Key B	
1	0	1	never	never	Key A|B	Key B	never	never	
1	1	1	never	never	Key A|B	never	never	never	

7.11.3. List of Error Codes

Decimal Value	Hexadecimal Value	Code	Description
0	$00	MF_OK	Normal Execution
1	$01	MF_NOTAGERR	No card response (Timout or card removed)
2	*$02*	*MF_CRCERR*	*Reserve for future use*
3	*$03*	*MF_EMPTY*	*Reserve for future use*
4	$04	MF_AUTHERR	Authentication Error (GetKey or Auth command)
5	*$05*	*MF_PARITYERR*	*Reserve for future use*
6	$06	MF_CODEERR	Coding Error detected during RF reception
7	$07	MF_NORC500CHIP	No RC500 Chip family detected
8	*$08*	*MF_SERNRERR*	*Reserve for future use*
9	*$09*	*MF_EEPROMERR*	*Reserve for future use*
10	$0A	MF_NOTAUTHERR	Command was passed on unauthenticated sector
11	$0B	MF_BITCOUNTERR	Error occurs on number of bits of card response
12	$0C	MF_BYTECOUNTERR	Error occurs on number of bytes of response
13	*$0D*	*MF_IDLE*	*Reserve for future use*
14	*$0E*	*MF_TRANSERR*	*Reserve for future use*
15	$0F	MF_WRITEERR	Write Error
16	*$10*	*MF_VALERR*	*Reserve for future use*
17	$11	MF_KEYERR	Key Error detected (GetKey command)
18	*$12*	*MF_READERR*	*Reserve for future use*
19	$13	MF_OVFLERR	Overflow Error detected (any command)
20	*$14*	*MF_POLLING*	*Reserve for future use*
21	$15	MF_FRAMINGERR	Framing Error detected (any command)
22	*$16*	*MF_ACCESSERR*	*Reserve for future use*
23	$17	MF_UNKNOWN_COMMAND	Unknown Command detected (any command)
24	$18	MF_COLLERR	Collision Error detected (any command)
25	*$19*	*MF_RESETERR*	*Reserve for future use*
26	*$1A*	*MF_INTERFACEERR*	*Reserve for future use*
27	$1B	MF_ACCESSTIMEOUT	Access Timeout on any commands.
28	$1C	MF_BCCERR	BCC Error (Select command) bad Check code
29	$1D	MF_SAKERR	SAK Error (Select command) not acknowledged
30	$1E	MF_PICCKEYIDEER	Picc Key Index Error (Authenticate command)
31	$1F	MF_PICCACCESSBITERR	Incoherent ACCESS BIT condition
32	$20	MF_ERRBV	Error on check block value after value operation
60	$3C	MF_WRONGPARAM	Wrong parameters in the command.
100	$64	MF_NYIMPLEMENTED	No Key Implemented or other error.
120	*$78*	*MF_COMM_ABORT*	*Reserve for future use*
121	*$79*	*MF_CALLOPEN*	*Reserve for future use*

In Italic are the Error codes not implemented yet.

7.11.4. Set of instructions

7.11.4.1. Reader functions

SENDRC500 (MIFARE®)

Description: Manage the Mifare® Chip MFRC500 chip familly.

Theses functions are available but shouldn't be used without a very good knowledge of the RC500 chip familly.

A) Maintenance Functions:

Details: Indicate the command performed on the RC500 chip at Initialization level

CLASS	INS	DATA IN	
$10	**$01**	**Length**	**TYPE**

Length: 1 byte: $01 for the following commands

TYPE: 1 byte:
- $**00** Initialize the RC500 chip (performed automatically at CSC reset)
- $**01** Disable RC500 chips (Needed only on hardware that doesn't support RC500 chip or to turn power down on RC500 chip).
- $**02** MF RC 500 Load of the default configuration in EEPROM Should be used on newly soldered MFRC500 chip to get a usable Configuration for manufacturing
- $**03** Reset the MFRC500 Chip (The default ASK configuration is set)

Returned response

CLASS	INS	DATA OUT	
$10	**$01**	**Length**	**StatusRC500**

Length: 1 byte: $01 for this command

StatusRC500: 1 byte:
- $00: RC500 Chip detected and correct operation
- $07: RC500 Chip not detected
- $09: RC500 Chip EEPROM Error

-=oOo=-

B) Get Key from EEPROM to Internal RC500 RAM Buffer:

CLASS	INS	DATA IN		
$10	**$01**	**Length**	**TYPE**	**KEY Index**

Length: 1 byte: $02 for this command

TYPE: 1 byte: $**04** type for the command that get Key from EEPROM: Load Key **KEYIndex** from EEPROM by its index in the internal buffer for the next cryptographic operation.

KEYIndex: 1 byte: value from $00 to $1F index of one of the 32 keys

Returned response

CLASS	INS	DATA OUT	
$10	**$01**	**Length**	**StatusRC500**

Length: 1 byte: $01 for this response

StatusRC500: 1 byte: idem previous StatusRC500 Status

-=oOo=-

C) Dump Current RC500 Configuration:

CLASS	INS	DATA IN	
$10	$01	Length	Type

Length: 1 byte: $01 for this command
Type: 1 byte: $**05** type of the command that Dump Current configuration of all the internal register of the RC500 chip. This function is only useful for expertise of problem relating to the Chip.

CLASS	INS	DATA OUT		
$10	$01	Length	Status	32REGContent (32 bytes)

Length: 1 byte: $21 in case of success for this Response
Status: 1 byte: idem previous Status:
32REGContent: 32 bytes: The content of the 32 first registers is returned

-=oOo=-

D) Get RC500 Serial Number:

CLASS	INS	DATA IN	
$10	$01	Length	Type

Length: 1 byte: $01 for this command
Type: 1 byte: $**09** type of the command that Gives the Serial Number of the RC500 chip.

Returned response

CLASS	INS	DATA OUT		
$10	$01	Length	Status	MF500SerialNumber (4 bytes)

Length: 1 byte: $05 in case of success for this Response
Status: 1 byte: idem previous Status:
MF500SerialNumber: 4 bytes: The Serial Number of serial number of the RC500 chip is returned

-=oOo=-

E) Load RC500 Internal Key Set:

This function can be used before an operation on a new sector.

Be careful: The Byte order for the KeyValue is different than the one used in the ChangeKey Command.

CLASS	INS	DATA IN			
$10	$01	Length	Type	PCDIndex (1 byte)	Key value (6 bytes)

Length: 1 byte: $08 for this command
Type: 1 byte: $**06** type of the command to Load an internal Key in the MFRC500 chip.
PCDIndex : 1 byte: the PCDIndex parameter contains the Key Index from 0x00 to 0x1F
Key value: 6 bytes: the Key is transmitted unencrypted on 6 bytes MSB First

Returned response

CLASS	INS	DATA OUT	
$10	$01	Length	Status

Length: 1 byte: $01 for this Response
Status: 1 byte: idem previous Status

-=oOo=-

F) Load Key into the RC500 Internal Buffer:

This function can be used before any operation which needs cryptographic work to specify the key to use.

CLASS	INS	DATA IN		
$10	$01	Length	Type	Key value (6 bytes)

Length: 1 byte: $07 for this command
Type: 1 byte: $**0B** type of the command to Load a Key into the MFRC500 internal Buffer.
Key value: 6 bytes: the Key is transmitted unencrypted on 6 bytes MSB First

Returned response

CLASS	INS	DATA OUT	
$10	$01	Length	Status

Length: 1 byte: $01 for this Response
Status: 1 byte: idem previous Status

-=oOo=-

G) SET MFRC5XX Proper Configuration:

CLASS	INS	DATA IN		
$10	$01	Length	Type	Mode

Length: 1 byte: $02 for this command
Type: 1 byte: $**0C** type of the command to set a proper configuration into the MFRC5XX.
Mode: 1 byte: 00 => ISO14443 B
01 => ISO14443 A
02 => ISO 15693

Returned response

CLASS	INS	DATA OUT	
$10	$01	Length	Status

Length: 1 byte: $01 for this Response
Status: 1 byte: idem previous Status

-=oOo=-

H) Get RC5XXX identification

CLASS	INS	DATA IN	
$10	$01	Length	Type

Length: 1 byte: $01 for this command
Type: 1 byte: $**0D** type of the command to know information about the chip MFRC5XX.

Returned response

CLASS	INS	DATA OUT			
$10	$01	Length	Status	ID chip (4 bytes)	Chip Version (1 byte)

Length: 1 byte: $06 for this Response
Status: 1 byte: idem previous Status
ID chip: 4 bytes
Chip version: 1 byte

-=oOo=-

I) WRITE RC500 Internal Register:

CLASS	INS	DATA IN			
$10	$01	Length	Type	Register Address (1 byte)	Value (1 byte)

Length: 1 byte: $03 for this command
Type: 1 byte: $**0E** type of the command to write new value at the specified MFRC5XX register address.
Register Address: 1 byte: specified register address
Value: 1 byte: new value to write

Returned response

CLASS	INS	DATA OUT	
$10	$01	Length	Status

Length: 1 byte: $01 for this Response
StatusRC500: 1 byte: idem previous Status +
$0F: RC500 Chip WRITE Error

-=oOo=-

J) READ RC500 Internal Register:

CLASS	INS	DATA IN		
$10	$01	Length	Type	Register Address (1 byte)

Length: 1 byte: $02 for this command
Type: 1 byte: $**0F** type of the command to read value at the specified MFRC5XX register address.
Register Address: 1 byte: specified register address

Returned response

CLASS	INS	DATA OUT		
$10	$01	Length	Status	Value

Length: 1 byte: $02 for this Response
StatusRC500: 1 byte: idem previous Status +
$12: RC500 Chip READ Error
Value: 1 byte: data read in the register

-=oOo=-

K) Load Speed RF limitation:

CLASS	INS	DATA IN				
$10	$01	Length	Type	Rx	Tx	Don't negociate

Length: 1 byte: $04 for this command
Type: 1 byte: $**10** type of the command to set RF speed limitation.
Rx: 1 byte: $00 => 106 kb/ s
$01 => 212 kb/ s
$02 => 424 kb/ s
$03 => 824 kb/ s
Tx: 1 byte: $00 => 106 kb/ s
$01 => 212 kb/ s
$02 => 424 kb/ s
$03 => 824 kb/ s
Don't negociate: 1 byte: $00 False
$01 True

Returned response

CLASS	INS	DATA OUT	
$10	$01	Length	Status

Length: 1 byte: $01 for this Response
Status: 1 byte: idem previous Status

-=oOo=-

L) Write RCXXX EEPROM:

CLASS	INS	DATA IN			
$10	$01	Length	Type	Address	Data

Length: 1 byte: $03 for this command
Type: 1 byte: $**12** type of the command to write RCXXX eeprom.
Adress: 1 byte: address of data in RCXXX eprom
Data: 1 byte: data value

Returned response

CLASS	INS	DATA OUT	
$10	$01	Length	Status

Length: 1 byte: $01 for this Response
Status: 1 byte: idem previous Status

-=oOo=-

M) Get RXXXX Status:

CLASS	INS	DATA IN	
$10	$01	Length	Type

Length: 1 byte: $01 for this command
Type: 1 byte: $**13** type of the command to get RCXXX status.

Returned response

CLASS	INS	DATA OUT	
$10	$01	Length	Status

Length: 1 byte: $06 for this Response
Status: 6 bytes: RegPrimaryStatus, RegSecondaryStatus, RegErrorFlag, Status, ISO4 Rx Retries Level, ISO4 Rx Retries

-=oOo=-

N) Set custom Frame Waiting Time

CLASS	INS	DATA IN			
$10	$01	Length	Type	Hi	Low

Length: 1 byte: $03 for this command
Type: 1 byte: $**14** type of the command to set custom FWT
Hi: 1 byte: high byte of custom FWT.
Low: 1 byte: low byte of custom FWT.

FWT = Hi*256 + Low. If FWT = 0, FWT is given by the card during detection.

Returned response

CLASS	INS	DATA OUT	
$10	$01	Length	Status

Length: 1 byte: $01 for this Response
Status: 1 byte: idem previous Status

-=oOo=-

O) Get current RF speed (DRI and DSI)

CLASS	INS	DATA IN	
$10	$01	Length	Type

Length: 1 byte: $01 for this command
Type: 1 byte: $**15** type of the command to get current DRI and DSI

Returned response

CLASS	INS	DATA OUT			
$10	$01	Length	Status	DSI	DRI

Length: 1 byte: $02 for this Response
Status: 1 byte: idem previous Status
DSI: 1 byte: PICC to PCD RF speed
DRI: 1 byte: PCD to PICC RF speed

DSI and DRI coding :
00 → 106 kb/s
01 → 212 kb/s
02 → 424 kb/s
03 → 847 kb/s

-=oOo=-

P) Get current Frame Waing Time

CLASS	INS	DATA IN	
$10	$01	Length	Type

Length: 1 byte: $01 for this command
Type: 1 byte: $**16** type of the command to get current FWT

Returned response

CLASS	INS	DATA OUT		
$10	$01	Length	Hi	Low

Length: 1 byte: $02 for this Response
Hi: 1 byte: high byte of current FWT.
Low: 1 byte: low byte of current FWT.

FWT = Hi*256 + Low.

7.11.4.2. Card functions

Select MF (MIFARE®)

Description: This command selects one specific MIFARE® card present in the field by its “Unique ID”. It allows in case of collision to select one card. This command requires the MF RC500 chip. This function realizes the ISO14443 connection such as REQA and SELECT.

CLASS	INS	DATA IN
$10	$02	Picc serial Number (4 bytes)

Picc Serial Number: 4 bytes: Serial number of the card

CLASS	INS	DATA OUT			
$10	$02	Length	Status	Code	Picc Serial Number (4 bytes)

Length: 1 byte: Response length: $06 in case of success for this Response
Status: 1 byte: See List of response code
Code: 1 byte: $08 means MIFARE® STANDARD 1K Card, $18 MIFARE® 4K, 0x28 for ProX.
Picc Serial Number: 4 bytes: Serial number of the card

Change Key (MIFARE®)

Description: This function allows writing in the trailer block of a PICC sector to change the keys and dedicated access rights. In a first step the sector is authenticated with the old key. In a second step the trailer block is modified with this same key and then re-authenticated with this same key or the new parameters.

Remark: The keys are in the inverse order than the LoadKey Command

Caution: The use of this function supposes to know the key present in the PCD and in the PICC and a perfect knowledge of the access bits condition on the MIFARE® Picc. The change of key operation is not always possible depending on the previous choice of the access bit

CLASS	INS	DATA IN								
$10	$03	Length	KeyPICC	SectorNum	KeyPCD	NewKeyA	AccessBits	FreeByte	NewKeyB	FinalAutKey

Length: 1 byte: $14 for this command

KeyPICC: 1 byte: value $0A for KEYA or $0B for KEYB used for initial authentication (default is A)

SectorNum: 1 byte: Sector Number of the sector (value from $00 to $0F for a MIFARE® STANDARD 1K Card, and from $00 to $27 for a MIFARE® 4K) on which the keys have to be changed

KeyPCD: 1 byte: Index of the PCD key (value $00 to $1F for EEPROM keys, $FF:internal buffer Key) used for initial authentication

NewKeyA: 6 bytes: New value of the A Key (unencrypted) with LSB First

AccessBits: 3 bytes: All access bits including trailer block as described in the MIFARE® specification.

FreeByte : 1 byte: Last byte of the access bit of the trailer block, its value can be used for any purpose.

NewKeyB: 6 bytes: New value of the B Key (unencrypted) with LSB First

FinalAutKey: 1 byte: Index of the key value for final Authentication $0A for KEYA or $0B for KEYB

CLASS	INS	DATA OUT			
$10	$03	Length	Status	Code	Picc Serial Number (4 bytes)

Length: 1 byte: Response length: $06 in case of success for this Response

Status: 1 byte: See List of response code

Code: 1 byte: $08 means MIFARE® STANDARD 1K Card, and $18 MIFARE® 4K.

Picc Serial Number: 4 bytes: Serial number of the card

Detect MF (MIFARE®)

Description: This function allows to detect the MIFARE® card present in the antenna field and select it (In case of collision a specific error code is returned). No key is required at this stage in the MF RC500 chip. This function realizes the ISO14443 connection Norm operations such as REQA and SELECT. There is no parameter needed.

CLASS	INS	DATA IN
$10	$04	-

No input data

CLASS	INS	DATA OUT			
$10	$04	Length	Status	Code	Picc Serial Number (4 bytes)

Length: 1 byte: Response length: $06 in case of success for this Response
Status: 1 byte: See List of response code (0x18 for collision between two cards of the same kind (1K or 4K) and 0x1B for collision between one MIFARE® STANDARD 1K and a MIFARE® 4K).
Code: 1 byte: $08 means MIFARE® STANDARD 1K Card, $18 MIFARE® 4K and 0x28 for ProX
Picc Serial Number: 4 bytes: Serial number of the card

Authenticate MF (MIFARE®)

Description: This function authenticates a sector. The right key (according to the block to authenticate: operation can be done only for one sector at a time) is required in the internal buffer of the MF RC500 chip. The Sector number to authenticate and indexes of key in the PCD and the PICC are transmitted as parameters in DataIn.

CLASS	INS	DATA IN			
$10	$05	Length $03	KeyPICC	SectorNum	KeyPCD

Length: 1 byte: fixed value = $03
KeyPICC: 1 byte: value $0A for KEYA or $0B for KEYB
SectorNum: 1 byte: Sector Number of the sector to authenticate value from $00 to $0F for a MIFARE® STANDARD 1K Card, and from $00 to $27 for a MIFARE® 4K.
KeyPCD : 1 byte: Index of the PCD key, value $00 to $1F for EEPROM keys, $FF:internal buffer Key.

CLASS	INS	DATA OUT			
$10	$05	Length	Status	Code	Picc Serial Number (4 bytes)

Length: 1 byte: Response length
Status: 1 byte: See List of response code
Code: 1 byte: $08 means MIFARE® STANDARD 1K Card, and $18 MIFARE® 4K.
Picc Serial Number: 4 bytes: Serial number of the card

ReadBlock (MIFARE®)

Description: The aim of this function is to read an authenticated block, the right key according with the one of the block to authenticate must be present in the internal buffer of the MFRC500.

CLASS	INS	DATA IN	
$10	$06	Length	BlockNum

Length: 1 byte: fixed value = $01

BlockNum : 1 byte: authenticated Block Number to read value from $00 to $3F for a MIFARE® STANDARD 1K Card, and from $00 to $FF for a MIFARE® 4K.

CLASS	INS	DATA OUT		
$10	$06	Length	Status	BlockContent (16 bytes)

Length: 1 byte: Response length
Status: 1 byte: See List of response code

BlockContent: 16 bytes: Block content read in the card

Remark: This command sends back the Type and the Serial Number of the card

In case of bad Transmission Error (0x0C,../..)

ReadSector (MIFARE®)

Description: The aim of this function is to read a Sector of the PICC.

CLASS	INS	DATA IN			
$10	$07	Length	KeyPICC	SectorNum	KeyPCD

Length: 1 byte: fixed value = $03

KeyPICC: 1 byte: value $0A for KEYA or $0B for KEYB

SectorNum: 1 byte: Sector Number of the sector to authenticate value $00 to $0F

KeyPCD : 1 byte : Index of the PCD key value $00 to $1F for EEPROM keys, $FF:internal buffer Key.

CLASS	INS	DATA OUT				
$10	$07	Length	Status	Code	PiccSerialNumber (4 bytes)	SectorContent (N)

Length: 1 byte : Response length $46 in case of successful operation

Status: 1 byte : See List of response code

Code: 1 byte: $08 means MIFARE® STANDARD 1K Card, and $18 MIFARE® 4K.

Picc Serial Number: 4 bytes: Serial number of the card

SectorContent: 64 bytes for a MIFARE® STANDARD 1K Card and 256 for a MIFARE® 4K: Sector read in the card in the following order : [Block0] [Block1] [Block2] … The [Block3] (or [Block16] for a MIFARE® 4K)is the Trailer Block which includes the Keys and access bit, the value read may not be the right one depending on the access bits.

NB: For a MIFARE® 4K Card, if SectorNum >= 0x28, the read function will loop back at the beginning of the sectors of the card. It is identical for the MIFARE® STANDARD 1K except that the Sectors from 0x10 to 0x27 are prohibited.

NB: For a MIFARE® 4K Card, if 256 bytes have to be returned, the returned Length will appear on 2 bytes, with the same rule as the CSC frames. (see chapter 2.6 Frame length)

WriteBlock (MIFARE®)

Description: This function writes an authenticated block. The right key according with the block to write needs to be present in the internal buffer of the MFRC500.

Caution : A block will always be written entirely (16 bytes)

This command can be used for data block as well as for block trailer but in this case access bit would not be verified (the ChangeKey command is safer for block trailer operation).

Remark : Due to card problem, the Write Block on block Trailer 3 sector 0 fails 1 over 2 times on the re-read command.

CLASS	INS	DATA IN		
$10	$08	Length	BlockNum	DataToWrite

Length : 1 byte : length ($11)

BlockNum : 1 byte : authenticated Block Number to be written.

DataToWrite : 16 bytes : Data to write in the selected authenticated block

CLASS	INS	DATA OUT		
$10	$08	Length	Status	DataVerification

Length: 1 byte : Response length $11 for successful operation

Status: 1 byte : See List of response code

DataVerification: 16 bytes: content of the memory read after operation

Halt MF (MIFARE®)

Description: The aim of this function is to send the HALT order on active PICC.

CLASS	INS	DATA IN
$10	$09	-

No parameters required

CLASS	INS	DATA OUT	
$10	$09	Length	Status

Length: 1 byte: Response length

Status: 1 byte: See List of response code

ReadMultipleBlock (MIFARE®)

Description: The aim of this function is to read several blocks in an authenticated sector, the right key according with the one of the block to authenticate must be present in the internal buffer of the MFRC500.

CLASS	INS	DATA IN		
$10	$0D	Length	BlockNum	Number

Length: 1 byte: fixed value = $02

BlockNum : 1 byte: authenticated Block Number to read value from $00 to $3F for a MIFARE® STANDARD 1K Card, and from $00 to $FF for a MIFARE® 4K.

Number: 1 byte: Number of blocks to read, in a single sector. Otherwise an authentication error will be sent back.

CLASS	INS	DATA OUT		
$10	$0D	Length	Status	BlockContent (16 bytes)

Length: 1 byte: Response length
Status: 1 byte: See List of response code

BlockContent: Blocks content read in the card

Remark: This command sends back the Type and the Serial Number of the card in case of bad Transmission Error (0x0C,../..)

SimpleWriteBlock (MIFARE®)

Description:

This function writes an authenticated block. The right key according with the block to write needs to be present in the internal buffer of the MFRC500. SELECT must have been realised, otherwise write will fail.

Caution : A block will always be written entirely (16 bytes)

This command can be used for data block as well as for block trailer but in this case access bit would not be verified (the ChangeKey command is safer for block trailer operation).

Remark : Due to card problem, the Write Block on block Trailer 3 sector 0 fails 1 over 2 times on the re-read command.

CLASS	INS	DATA IN		
$10	$0E	Length	BlockNum	DataToWrite

Length : 1 byte : length ($11)

BlockNum : 1 byte : authenticated Block Number to be written.

DataToWrite : 16 bytes : Data to write in the selected authenticated block

CLASS	INS	DATA OUT	
$10	$0E	Length	Status

Length: 1 byte : Response length $01 for successful operation

Status: 1 byte : See List of response code

ReadSectorData (MIFARE®)

Description: The aim of this function is to read the data blocks of a Sector of the PICC.

CLASS	INS	DATA IN			
$10	$0F	Length	KeyPICC	SectorNum	KeyPCD

Length: 1 byte: fixed value = $03

KeyPICC: 1 byte: value $0A for KEYA or $0B for KEYB

SectorNum: 1 byte: Sector Number of the sector to authenticate value $00 to $0F

KeyPCD : 1 byte : Index of the PCD key value $00 to $1F for EEPROM keys, $FF:internal buffer Key.

CLASS	INS	DATA OUT				
$10	$0F	Length	Status	Code	PiccSerialNumber (4 bytes)	SectorContent (N)

Length: 1 byte : Response length $36 in case of successful operation

Status: 1 byte : See List of response code

Code: 1 byte: $08 means MIFARE® STANDARD 1K Card, and $18 MIFARE® 4K.

Picc Serial Number: 4 bytes: Serial number of the card

SectorContent: 48 bytes for a MIFARE® STANDARD 1K Card or 240 bytes for a MIFARE® 4K: Sector read in the card in the following order : [Block0] [Block1] …[Block2] (or [Block15] for a MIFARE® 4K).

NB: For a MIFARE® 4K Card, if SectorNum >= 0x28, the read function will loop back at the beginning of the sectors of the card. It is identical for the MIFARE® STANDARD 1K except that the Sectors from 0x10 to 0x27 are prohibited.

WriteSectorData (MIFARE®)

Description: The aim of this function is to write the data blocks of a Sector of the PICC.

CLASS	INS	DATA IN				
$10	$10	Length	KeyPICC	SectorNum	KeyPCD	DataToWrite

Length: 1 byte: fixed value = $33 for Mifare1K or $F3 for Mifare4K

KeyPICC: 1 byte: value $0A for KEYA or $0B for KEYB

SectorNum: 1 byte: Sector Number of the sector to authenticate value $00 to $0F

KeyPCD : 1 byte : Index of the PCD key value $00 to $1F for EEPROM keys, $FF:internal buffer Key.

DataToWrite : 48 (or 240) bytes * : Data to write in the selected sector

CLASS	INS	DATA OUT	
$10	$10	Length	Status

Length: 1 byte : Response length $01 for successful operation

Status: 1 byte : See List of response code

7.11.4.3. Value block functions

N.B. : To initialize a block as a "value block" by the access bits, first the data block must be initialized by a WriteBlock command as follow :

Byte Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Description	Value				Value				Value				Ad	$\overline{\text{Ad}}$	Ad	$\overline{\text{Ad}}$

On an increment or a decrement command, the PICC does a hardware update of the 3 fields containing value and complement of this value. The Address of the block (Ad bytes) must be present but they are not updated in the operation.

Increment Value (MIFARE®)

Description: This function is used to increment a PICC block initialized as a "**value block**" which is a kind of counter (see chapter access control bits). The block need to be previously authenticated with the right key dedicated to the increment operation. The max increment value can stand on 4 bytes. (Overflow must be managed by the application).

N.B. The use of this function requires the knowledge of the Key and access bits present in the Picc and the Pcd.

CLASS	INS	DATA IN		
$10	**$0A**	**Length**	**BlockPicc-**	**Increment (B0,B1,B2,B3)**

Length: 1 byte : length fixed at $05 for this command
BlockPicc: 1 byte : block number from $00 to $3F
Increment: 4 bytes : value to increment on the counter (LSB first)

Returned response format

CLASS	INS	DATA OUT		
$10	**$0A**	**Length**	**Status**	**ControlValue (B0,B1,B2,B3)**

Length: 1 byte : Response length $05 for successful operation
Status: 1 byte : See List of response code
ControlValue : 4 bytes : Value after increment operation for control purpose (MSB first).

i.e. : 10 0A 05 04 01 00 00 00 (command for an increment of 1 on the value block 4)
gives for example the response 10 0A 05 00 00 00 01 02 if the previous value was 0x101 the new value is 0x102

Decrement Value (MIFARE®)

Description: This function is used to decrement a PICC block initialized as a "**value block**" which is a kind of counter (see chapter access control bits). The block need to be previously authenticated with the right key dedicated to the decrement operation. The max decrement value can stand on 4 bytes. (Underflow must be managed by the application).

N.B. The use of this function requires the knowledge of the Key and access bits present in the Picc and the Pcd.

CLASS	INS	DATA IN		
$10	$0B	Length	BlockPicc-	Decrement (B0,B1,B2,B3)

Length: 1 byte : length fixed at $05 for this command
BlockPicc: 1 byte : block number from $00 to $3F
Decrement: 4 bytes : value to decrement on the counter (LSB first)

Returned response format

CLASS	INS	DATA OUT		
$10	$0B	Length	Status	ControlValue (B0,B1,B2,B3)

Length: 1 byte : Response length $05 for successful operation
Status: 1 byte : See List of response code
ControlValue : 4 bytes : Value after decrement operation for control purpose (MSB first).

i.e. : 10 0B 05 04 01 00 00 00 (command for an decrement of 1 on the value block 4)
Gives for example the response 10 0B 05 00 00 00 01 01 if the previous value was 0x102 the new value is 0x101

BackUp or Restore a Value Block (MIFARE®)

Description: This function can be use as well as a "Backup Command" or a "Restore Command". The aim of this command set is to restore the previous value of a block as anti-tearing protection (when the card is getting out of the field during a write operation). The previous value can be restored from another "value block" location.

To do so the two blocks must belong to the same sector of the PICC.

The block that need restoration (the one which will be written) need to be previously authenticated with the key dedicated to the "Restore" operation (see access bit condition in the Mifare® documentation)

N.B. The use of this function requires the knowledge of the Key and access bits present in the Picc and the Pcd.

CLASS	INS	DATA IN		
$10	**$0C**	**Length**	**BackupBlock**	**RestoreBlock**

Length: 1 byte : length fixed at $02 for this command
BackupBlock: 1 byte : backup block number from $00 to $3F
(where the value was copied before the failed operation)
RestoreBlock: 1 byte : Restore block number from $00 to $3F
(where the value must be copied to restore its previous value)

Returned response format

CLASS	INS	DATA OUT		
$10	**$0C**	**Length**	**Status**	**ControlValue (B0,B1,B2,B3)**

Length: 1 byte : Response length $05 for successful operation
Status: 1 byte : See List of response code
ControlValue : 4 bytes : Value after restore operation for control purpose (MSB first).

N.B. : On a restore command, the PICC makes a hardware update of the 3 fields containing value and complement of value. The Address (Ad) bytes have nothing to do with the address managed by the restore command (considered as value during the restoration).

8. USE EXEMPLES

8.1. Class CTS

class 6 functions in order to do an UPDATE_CTS command

.... CSC found : reset ASK CSC

Message : 80 02 06 01 00 55 b3

Description: Type command Sens Host -> CSC
.... Command: Active CTS
.... Timing= 0 s 20 ms

Response : 01 0c 06 01 0a 0f 50 02 00 93 96 67 00 00 00 2d

Description: Type command Sens CSC -> Host
.... Function class = class CTS
.... Response : Active CTS
.... Ticket CTS active

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

Message : 80 04 06 02 02 02 00 27 9e

Description: Type command Sens Host -> CSC
.... Command : Read CTS
.... ADD: 02
.... NB : 02
.... Timing= 0 s 15 ms

Response : 01 06 06 02 03 02 00 10 00 c7 30

Description: Type command Sens CSC -> Host
.... Function class = class CTS
.... Response : Read CTS
.... Read command success
.... Data read : 00 10
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

Message : 80 04 06 02 0a 04 00 35 0c

Description: Type command Sens Host -> CSC
.... Command : Read CTS
.... ADD: 0a
.... NB : 04
.... Timing= 0 s 20 ms

Response : 01 08 06 02 05 02 ff 77 77 77 00 a8 85

Description: Type command Sens CSC -> Host
.... Function class = class CTS
.... Response : Read CTS
.... Read command success
.... Data read : ff 77 77 77

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Message : 80 08 06 03 0a 02 55 44 ee ee 00 76 66

Description: Type command Sens Host -> CSC
.... Command : Update CTS
.... NB : 0a
.... ADD: 02
.... Data to write : 55 44 ee ee 00 76 66 cc cc cc
.... Data read   : cc cc cc cc cc cc cc cc cc cc .... Timing= 0 s 26 ms

Response : 01 06 06 03 03 02 55 44 00 25 3a

Description: Type command Sens CSC -> Host
.... Function class = class CTS
.... Response : Update CTS
.... Update command success
.... Data read : 55 44
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

Message : 80 04 06 02 0a 04 00 35 0c

Description: Type command Sens Host -> CSC
.... Command : Read CTS
.... ADD: 0a
.... NB : 04
.... Timing= 0 s 20 ms

Response : 01 08 06 02 05 02 55 44 77 77 00 fb 4a

Description: Type command Sens CSC -> Host
.... Function class = class CTS
.... Response : Read CTS
.... Read command success
.... Data read : 55 44 77 77
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

Message : 80 08 06 03 0a 02 aa bb 55 44 00 95 a7

Description: Type command Sens Host -> CSC
.... Command : Update CTS
.... NB : 0a
.... ADD: 02
.... Data to write : aa bb 55 44 00 95 a7 cc cc cc
.... Data read   : cc cc cc cc cc cc cc cc cc cc .... Timing= 0 s 25 ms

Response : 01 06 06 03 03 02 aa bb 00 16 03

Description: Type command Sens CSC -> Host
.... Function class = class CTS
.... Response : Update CTS
.... Update command success
.... Data read : aa bb

ASK SA - 2260, route des Crêtes - BP 337 - 06906 Sophia-Antipolis Cedex - Tel. 04 97 21 40 18 - Fax. 04 92 38 93 21 Web : www.ask-rfid.com

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Message : 80 04 06 02 0a 04 00 35 0c

Description: Type command Sens Host -> CSC
.... Command : Read CTS
.... ADD: 0a
.... NB : 04
.... Timing= 0 s 20 ms

Response : 01 08 06 02 05 02 aa bb 77 77 00 73 7a

Description: Type command Sens CSC -> Host
.... Function class = class CTS
.... Response : Read CTS
.... Read command success
.... Data read : aa bb 77 77
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

8.2. Variable class mapping for GTML card

class 5 function : APPEND_RECORD command (in session mode)

.... CSC found : reset ASK CSC

Message : 80 02 01 01 00 50 3f

Description: Type command Sens Host -> CSC
.... Function class = system class
.... Command : CSC version
.... Timing= 0 s 15 ms

Response : 01 41 01 01 43 53 43 20 30 32 2e 30 38 20 3c 70
68 2d 39 36 30 30 2d 31 31 35 32 3e 20 4d 61 72
20 31 33 20 32 30 30 31 20 31 38 3a 30 34 3a 31
34 20 28 43 29 20 41 53 4b 20 47 45 4e 5f 33 58
30 20 00 00 91 cd

Description: Type command Sens CSC -> Host
.... Function class = system class
.... Response : CSC version
.... In Ascii :
.... CSC 02.08 <ph-9600-1152> Mar 13 2001 18:04:14 (C) ASK GEN_3X0

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

Message : 80 08 01 03 00 00 00 00 01 00 00 17 69

Description: Type command Sens Host -> CSC
.... Function class = system class
.... Command : Short Enter Hunt Phase
.... Antenna number parameter = 00
.... Contact parameter =00 ISOB=00 ISOA=00
.... TICKET parameter =00 RMT=01
.... Timing= 0 s 25 ms

Response : 01 1e 01 03 00 03 19 00 22 17 6c ff 40 3b 6f 00
00 80 5a 08 03 03 00 00 00 00 22 17 6c 82 90 00
00 39 4f

Description: Type command Direction CSC -> Host
.... Function class = system class
.... Response : TAG search
.... Antenna number parameter = 00 (= 0x80 if failed)
.... Mode RMT Choix Ok
.... ATR length = 19
.... Convention type 3b must be to 3B
.... T0 parameter 6f must be to 6F (if there are 2 parameters after)
.... TB1 parameter 00 must be to 00
.... TC1 parameter 00 must be to 00
.... Indicator category 80 must be to 80
.... Parameter nbconstr 5a must be to 5A
.... Type de composant 08
.... Application type 03
.... ROM version 03 00
.... EEPROM Version 00 00
.... Serial number 00 22 17 6c
.... nbstatus parameter 82 must be to 82
.... Status ATR 90 00 must be to90 00
.... ATR parameter OK

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

Message : 80 08 05 10 02 08 01 0d 00 00 00 ae 4b

Description: Type command Sens Host -> CSC
.... Function class = class 05
.... Command : Open Secured Session
.... 02 : validation session
.... 08: Fichier Journal de transport sélectionné
.... 01 : Record number
.... Timing= 0 s 55 ms

Response : 01 23 05 10 00 90 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 36 58

Description: Type command Sens CSC -> Host
.... Function class = class 5
.... Correct return

.... Response: Open Secured Session
.... Number of non-ratified applications 00

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

Message : 80 0d 05 01 00 08 05 00 00 00 00 00 20 10 0d 00
0c a8

Description: Type command Direction Host -> CSC
.... Function class = class 05
.... Command: Append Record
.... 00: Access in Default mode
.... 08: Fichier Journal de transport sélectionné
.... Timing= 0 s 41 ms

Response: 01 05 05 01 00 90 00 00 d5 64

Description: Type command Sens CSC -> Host
.... Function class = class 5
.... Correct return
.... Response: Append Record

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

Message : 80 08 05 08 08 04 20 00 20 10 00 39 93

Description: Type command Sens Host -> CSC
.... Function class = class 05
.... Command: Select File
.... Timing= 0 s 35 ms

Response: 01 1e 05 08 00 90 00 85 17 08 04 04 1d 03 1f 10
10 10 00 03 03 03 00 00 00 00 00 00 00 00 00 00
00 06 22

Description: Type command Sens CSC -> Host
.... Function class = class 5
.... Correct return
.... Response: Select File
.... Détail du FCI :
.... 85 indique un Tag :
.... SID = 08
.... File type 04 =EF
.... Structure type 04 =Circular
.... Byte number for each record = 1d
.... Record number = 03
.... AC : Access condition for command index 0 1f =ALWAYS Command READ RECORD
.... AC : Access condition for command index 1 10 =INCONNU Command UPDATE RECORD
.... AC : Access condition for command index 2 10 =INCONNU Pas de Commande
.... AC : Access condition for command index 3 10 =INCONNU Command APPEND RECORD
.... Key index to use 00 03 03 03
.... File status 00 Fichier Valide Fichier lisible si Invalide
.... Key index KVC123 pour les DF 00 00 00
.... Plancher des compteurs 00 00 00 Plafond 00 00 00

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

Message : 80 0d 05 01 00 00 05 00 00 00 00 00 20 10 0d 00
a6 14

Description: Type command Direction Host -> CSC
.... Function class = class 05
.... Command : Append Record
.... 00: Access in Default mode
.... 00: Selected file
.... Timing= 0 s 41 ms

Response : 01 05 05 01 00 90 00 00 d5 64

Description: Type command Sens CSC -> Host
.... Function class = class 5
.... Correct return
.... Response : Append Record

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

Message : 80 02 05 11 00 a0 c9

Description: Type command Sens Host -> CSC
.... Function class = class 05
.... Command : Close Secured Session
.... Timing= 0 s 50 ms

Response : 01 05 05 11 00 90 00 00 95 d0

Description: Type command Sens CSC -> Host
.... Function class = class 5
.... Correct return
.... Response : Close Secured Session

8.3. Class CD97

class 3 function : APPEND_RECORD command

..... CSC found : reset ASK CSC

Message : 80 02 01 01 00 50 3f

Description: Type command Sens Host -> CSC
..... Function class = system class
..... Command : CSC version
..... Timing= 0 s 15 ms

Response : 01 41 01 01 43 53 43 20 30 32 2e 30 38 20 3c 70
68 2d 39 36 30 30 2d 31 31 35 32 3e 20 4d 61 72
20 31 33 20 32 30 30 31 20 31 38 3a 30 34 3a 31
34 20 28 43 29 20 41 53 4b 20 47 45 4e 5f 33 58
30 20 00 00 91 cd

Description: Type command Sens CSC -> Host
..... Function class = system class
..... Response : CSC version
..... In Ascii :
..... CSC 02.08 <ph-9600-1152> Mar 13 2001 18:04:14 (C) ASK GEN_3X0

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

Message : 80 08 01 03 00 00 00 00 01 00 00 17 69

Description: Type command Sens Host -> CSC
..... Function class = system class
..... Command : Short Enter Hunt Phase
..... Antenna number parameter = 00
..... Contact parameter=00 ISOB=00 ISOA=00
..... TICKET parameter=00 RMT=01
..... Timing= 0 s 24 ms

Response : 01 1e 01 03 00 03 19 00 10 5e f3 ff c0 3b 6f 00
00 80 5a 03 01 01 11 02 04 00 10 5e f3 82 90 00
00 03 62

Description: Type command Direction CSC -> Host
..... Function class = system class
..... Response : TAG search
..... Antenna number parameter = 00 (a 80 si en panne)
..... Mode RMT Choix Ok
..... ATR length = 19
..... Convention type 3b must be to 3B
..... T0 parameter 6f must be to 6F indique si 2 param suivants
..... TB1 parameter 00 must be to 00
..... TC1 parameter 00 must be to 00
..... Indicator category 80 must be to 80
..... Parameter nbconstr 5a must be to 5A
..... Type de composant 03

.... Application type 01
.... ROM version   01 11
.... EEPROM Version 02 04
.... Serial number 00 10 5e f3
.... Nbstatus parameter   82 must be to 82
.... Status ATR 90 00 must be to 90 00
.... ATR parameter   OK

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

Message : 80 0a 03 01 01 08 05 00 00 00 00 00 00 68 43

Description: Type command Sens Host -> CSC
.... Function class = CD97 class
.... Command : Append Record
.... 01: Access in Protected mode
.... 08: Fichier Journal de transport sélectionné
.... Timing= 0 s 100 ms

Response : 01 05 03 01 00 90 00 00 2f 7c

Description: Type command Sens CSC -> Host
.... Function class = CD97 class
.... Correct return
.... Response : Append Record

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

Message : 80 08 03 08 08 04 31 00 31 15 00 d3 29

Description: Type command Sens Host -> CSC
.... Function class = CD97 class
.... Command : Select File
.... Timing= 0 s 31 ms

Response : 01 1e 03 08 00 90 00 85 17 00 04 04 1d 01 1f 12
00 12 01 03 01 03 00 00 00 00 00 00 00 00 00 00
00 5c 76

Description: Type command Sens CSC -> Host
.... Function class = CD97 class
.... Correct return
.... Response : Select File
.... Détail du FCI :
.... 85 indique un Tag :
.... SID = 00
.... File type 04 =EF
.... Structure type 04 =Circular
.... Byte number for each record = 1d
.... Record number = 01
.... AC : Access condition for command index   0 1f =ALWAYS (READ RECORD Command)
.... AC : Access condition for command index   1 12 =PROTECTED^SESSION (UPDATE RECORD Command)
.... AC : Access condition for command index   2 00 =NEVER (No Command)
.... AC : Access condition for command index   3 12 =PROTECTED^SESSION (APPEND RECORD Command)
.... Key index to use 01 03 01 03

..... File status 00 Fichier Valide Fichier lisible si Invalide
..... Key index KVC123 for DF 00 00 00
..... Plancher des compteurs 00 00 00 Plafond 00 00 00

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

Message : 80 0a 03 01 01 00 05 00 00 00 00 00 00 d4 6e

Description: Type command Sens Host -> CSC
..... Function class = CD97 class
..... Command : Append Record
..... 01: Access in Protected mode
..... 00: Selected file
..... Timing= 0 s 95 ms

Response : 01 05 03 01 00 90 00 00 2f 7c

Description: Type command Sens CSC -> Host
..... Function class = CD97 class
..... Correct return
..... Response : Append Record

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>