IDTECH®

Value through Innovation

Augusta Interface Developer's Guide

80145504-001 Rev. U
9 September 2021

ID TECH

10721 Walker Street, Cypress, CA90630 Voice: (714) 761-6368 Fax: (714) 761-8880

Visit us at http://www.idtechproducts.com

The information contained herein is provided to the user as a convenience. While every effort has been made to ensure accuracy, ID TECH is not responsible for damages that might occur because of errors or omissions, including any loss of profit or other commercial damage, nor for any infringements or patents or other rights of third parties that may result from its use. The specifications described herein were current at the time of publication, but are subject to change at any time without prior notice.

LIMITED WARRANTY

ID TECH warrants to the original purchaser for a period of 12 months from the date of invoice that this product is in good working order and free from defects in material and workmanship under normal use and service. ID TECH's obligation under this warranty is limited to, at its option, replacing, repairing, or giving credit for any product that returned to the factory of origin with the warranty period and with transportation charges and insurance prepaid, and which is, after examination, disclosed to ID TECH's satisfaction to be defective. The expense of removal and reinstallation of any item or items of equipment is not included in this warranty. No person, firm, or corporation is authorized to assume for ID TECH any other liabilities in connection with the sales of any product. In no event shall ID TECH be liable for any special, incidental or consequential damages to purchaser or any third party caused by any defective item of equipment, whether that defect is warranted against or not. Purchaser's sole and exclusive remedy for defective equipment, which does not conform to the requirements of sales, is to have such equipment replaced or repaired by ID TECH. For limited warranty service during the warranty period, please contact ID TECH to obtain a Return Material Authorization (RMA) number & instructions for returning the product.

THIS WARRANTY IS IN LIEU OF ALL OTHER WARRANTIES OF MERCHANTABILITY OR FITNESS FOR PARTICULAR PURPOSE. THERE ARE NO OTHER WARRANTIES OR GUARANTEES, EXPRESS OR IMPLIED, OTHER THAN THOSE HEREIN STATED. THIS PRODUCT IS SOLD AS IS. IN NO EVENT SHALL ID TECH BE LIABLE FOR CLAIMS BASED UPON BREACH OF EXPRESS OR IMPLIED WARRANTY OF NEGLIGENCE OF ANY OTHER DAMAGES WHETHER DIRECT, IMMEDIATE, FORESEEABLE, CONSEQUENTIAL OR SPECIAL OR FOR ANY EXPENSE INCURRED BY REASON OF THE USE OR MISUSE, SALE OR FABRICATIONS OF PRODUCTS WHICH DO NOT CONFORM TO THE TERMS AND CONDITIONS OF THE CONTRACT.

ID TECH and Value through Innovation are trademarks of International Technologies & Systems Corporation. USB (Universal Serial Bus) specification is copyright by Compaq Computer Corporation, Intel Corporation, Microsoft Corporation, and NEC Corporation. Windows is registered trademarks of Microsoft Corporation. Quick Chip is a Visa specification. M/Chip Fast is a Master Card specification.

Augusta KB with Quick Chip and M/Chip Fast is patent-pending.

Revision History

Revision	Date	Description of Changes	By
50	12/20/2015	Initial draft.	KT
50a	12/30/2015	Misc. corrections as suggested by Engineering.	KT
51	01/12/2016	Misc. updates regarding data formats and serial commands (now called USB communication commands).	KT
52	03/10/2016	Incorporated LM's command tables. Added Appendix on Function IDs. USB descriptors added.	KT
53	03/24/2016	Table format applied to remaining commands. New language added to clarify the difference between ITP and NGA commands. Table added, to clarify the categories of NGA commands.	KT
B	04/08/2016	Add Set Encryption and Get Encryption Control (78 53 01 07 and 78 52 01 07).	KT
C	06/22/2016	Added LCD Foreign Language Mapping Table (for EMV-required LCD messages). Added info for 49 72 01 responses (EMV-required LCD messaging).	KT
D	10/25/2016 11/02/2016	Added Terminal Configurations appendix. Added appendix for TransArmor support.	KT
E	11/07/2016	Added Quick Chip appendix.	KT
F	11/29/2016 12/22/2016	Added extra discussion, under Features, for SRED (Augusta S) and Quick Chip. Added MAC Calculation appendix. Added Get/ Set Verify Encrypt Data Output Option commands. Added EMV L2 Output Format chapter. Added VID/PID (USB) for SRED product. Added Get Extended Firmware Version command (78 31).	KT KT
G	02/16/2017 02/22/2017	Revised discussion of Quick Chip to include "Faster EMV" terminology. Corrected DFEF12 to DFEE12 in Appendix J. Corrected DFEF62 data values. Add commands for getting/setting postamble characters in Quick Chip & M/Chip Fast mode. Modify Quick Chip discussion to use the "Quick Chip and M/Chip Fast" terminology.	KT
H	05/18/2017 05/31/2017	Split User Manual off into separate doc. Enhanced discussion of response codes (App. D)	KT
I	06/13/2017	Add new EMV result codes to App. C	KT
J	07/25/2017	Remove Default All command (destructive reset)	KT
K	02/26/2018	Add additional information about Quick Chip Mode	KT
M	05/28/2019	Renamed to Augusta Interface Developers Guide. Font facelift, repagination, format standardization. 6.15 Set Encryption Control: added note that turning on encryption is a one-time setting that cannot be disabled. Appendix C: Error codes: Added 0xD00 - Unable to turn off encryption Minor style and highlighting corrections throughout	CB
N	07/02/2019	Added Get CTL2 Transaction Mode	CB
	7/9/2019 7/19/2019 8/5/2019 8/13/2019	Added Get Quick Chip Mode command Added Remove White List command for both EMV and MSR Added Appendix L: White List Format Added Beeper Control command Minor update to Beeper Control command description Corrections to Set White List commands, both EMV and MSR.	CB CB CB CB
P	3/27/2020	Added Appendix M: US Common AID Support Corrected error in Get Encryption Control command response.	CB
R	6/10/2020	Added Appendix N: Setting Parameters (Function ID) and Values	CB

		Add the following commands: Get Model Status Set TransArmor RSA TID Get TransArmor RSA TID Load Certificate for TransArmor RSA Algorithm Read Certificate for TransArmor RSA Algorithm Set ICC Reading Characteristics Get ICC Reading Characteristics Set EMV CT L2 Transaction Interval Get EMV CT L2 Transaction Interval MSR Setting Command Set MSR Reading Characteristics Load Certificate for TransArmor RSA Algorithm Read Certificate for TransArmor RSA Algorithm Set Intercharacter Delay for USB-KB interface Review Intercharacter Delay for USB-KB interface	
S	04/12/2021	• Default General Group (78 53 00) o Updated the list of settings reset to default. • Appendix N: Setting Parameters (Function ID) and Values o D6: updated default value to 00.	CB
T	07/26/2021	• Corrected Get KSN command body to 78 46 3E. • Corrected command bodies for Set Whitelist, Retrieve Whitelist, and Remove Whitelist commands.	CB
U	09/09/2021	• Added Set QuickChip Mode USB-KB Output Data Postfix, Get QuickChip Mode USB-KB Output Data Postfix, Set QuickChip Mode USB-KB Output Data Prefix, and Get QuickChip Mode USB-KB Output Data Prefix commands.	CB

Table of Contents

1.0 Introduction

ID TECH's Augusta series reader is a two-in-one MagStripe and EMV ICC card reader, built for maximum durability, security, and cost-effectiveness. It is available in both standard and PCI SRED (Augusta S) versions.

This manual (a companion to the *Augusta User Manual*, P/N 80145503-001) describes the low-level firmware commands that can be used to control the Augusta and Augusta S series readers.

1.1 Audience

This guide is intended to be used as a reference by integrators and developers interested in controlling Augusta through firmware commands issued over the Universal Serial Bus. users of this guide will already be familiar with the sending and receiving of raw data payloads over a serial interface.

The sending and receiving of raw firmware commands affords a level of control over the device that is difficult to achieve any other way. Most of Augusta's functionality can be accessed via high level language (C#) using the Universal SDK (described briefly below), but not all of the more than 100 firmware commands available in Augusta have corresponding SDK methods. For the ultimate in low-level control, it's necessary to know how to issue commands directly. This guide explains what all the commands are, and how they are used.

1.2 High-Level Language Development with the Universal SDK

Developers using C# can take advantage of ID TECH's Universal SDK for Augusta, which provides extensive libraries and convenience methods that make communication, error handling, data parsing, and other routine chores much easier than they would be otherwise.

The Universal SDK comes with sample code and documentation designed to make development relatively quick and straightforward. ID TECH also makes available a Universal Demo program (or "UDemo" app) that has a rich GUI for making use of Universal SDK functionality. The UDemo app is a useful configuration utility (as well as a learning tool) in its own right. We strongly recommend you become thoroughly familiar with it.

The SDK, the UDemo app, and related documentation can be downloaded from the ID TECH website. Please check the ID TECH Knowledge Base for the latest downloads. Downloads do not require registration. Also note that the UDemo app does not require you to download the entire SDK; it's available as a separate download.

The Universal SDK hides many of the low-level details of dealing with firmware commands. There are times, however, when you may want to issue low-level commands directly. The document you are reading now is designed to help with that.

Sending low-level commands is not difficult. First, you need to establish USB connectivity with the Augusta, as discussed in the next chapter. You also need to know how to wrap commands in the so-called NGA communication protocol wrapper (a simple matter of putting 0x02 at the start of every command string, and 0x03 at the end, with an LRC and checksum coming before the 0x03). Finally, you need to know what the actual commands are, and which data arguments they take (if any). All these topics (plus error codes, data structures, etc.) are explained in subsequent chapters of this guide.

1.3 For Further Information

If you have additional questions about Augusta or this guide, visit ID TECH's public Knowledge Base.

Also, for the most recent SDK, sample code, utilities, firmware updates, articles, manuals, etc., visit the Augusta downloads page.

To obtain product support for Augusta, contact support@idtechproducts.com.

2.0 USB Communication

Augusta can accomplish two-way communication using USB-KB (for SecureMag compatibility mode; that is, magstripe-only operation) or via USB-HID (magstripe plus EMV).

In MSR-only mode (USB-KB), Augusta sends and receives commands using the ID TECH ITP Protocol, a simple command/response scheme that wraps data with start and end codes plus an LRC value (longitudinal redundancy check; see below). In combined EMV+MSR mode (that is, USB-HID mode; EMV enabled), Augusta sends commands via ID TECH NGA Protocol, a very simple command-and-response scheme; see further below.

Before going further, let's take a moment to discuss the ITP and NGA protocols.

2.1 ITP Protocol Format

- <STX> is defined as 0x02
- <ETX> is defined as 0x03
- <ACK> is defined as 0x06
- <NAK> is defined as 0x15

Configuration Setting Command: <STX><S><FuncSETBLOCK1>…<FuncBLOCKn><ETX><LRC>
Response: <ACK> or <NAK> for wrong command (invalid funcID, length and value)

Configuration Review Command: <STX><R><ReviewID><ETX><LRC>
Response: <ACK> <STX> <FuncID> <Len> <FuncData> <ETX> <LRC 2>

<FuncID>, <Len> and <FuncData> definition are same as described above.
Note: ReviewID (value 0x1F) will return all funcID-s.

Where:

- <Length> = is a two-byte counter (high then low) from <Command ID> to the byte before LRC
- <Command ID> = is a one-byte value identifying a specific command ID.
- <FuncID> = is a one-byte Function ID, which identifies the particular function or settings affected.
- <Len> = is a one-byte length count for the data block "<FuncData>"
- <FuncData> = is the data block for the function
- <Response Data> = is the data block associated with the Response.
- <Status> is a two-byte value indicating the success or failure of a command. There are a few responses without status (e.g., Version command).

Note that the LRC comes at the end and represents the XOR of all preceding byte values including STX and ETX.

3.0 NGA Protocol Format

<STX><CLenL><CLenH><Command_Body/Response_Body/Notification/Output Body/Input Body>…<LRC><CheckSum><ETX>

3.1 Function Command

Command Format

<TaskID><F><Command>…

Response

If the command is valid and executes correctly:
<ACK>[<Response>]…
Otherwise:
<NAK><ErrorCode1>][<ErrorCode2>

3.2 Setting Command

Command Format

<TaskID> <S> <00>
or <TaskID><S><NoFunc><FuncBlock>

Where:

- <TaskID> <S> <00> means Default the Task
- <All (0x7F)> <S> <00> means Default All
- <NoFunc> is number of function blocks to set in a task. Now only support <NoFunc> = 1
- <FuncBlock> has the following format of <FuncID><Len><FuncData>
- <FuncID> is a one byte Property ID
- <Len> is a one byte length count for a setting <FuncData>
- <FuncData> is a setting for a property.

Response

If the command is valid and executes correctly:
<ACK>
Or (fail)
<NAK><ErrorCode1><ErrorCode2>

For setting a command, the reader will first scan the command and send back codes for any errors it finds. The reader will complete the setting if it finds no errors.
<Unknown ID in Setting/Review list> (1600) and <Setting value out of range> (1400) will be treated as warning. i.e.: Do setting for other properties but skip setting for those properties which have warning. Warning will be sent to host.

3.3 Host Review Command

Command Format

<TaskID><R><00> or
<TaskID><R><NoFunc><FuncID>

Where:
<NoFunc> is number of properties to review in a task. Now only support <NoFunc> = 1
<TaskID><R><00> allows host to review all properties in a task.
<TaskID><R><01><FuncID> allows host to review setting for a property.

Response:
<ACK><TaskID><NoFunc><FuncBLOCK> or (fail)
<NAK><ErrorCode1><ErrorCode2>

<TaskID> is needed for each <FuncID> in review command.

Note that in contrast to the ITP format, in NGA protocol the LRC value does *not* come at the end, and does *not* include STX, length bytes, nor ETX in its calculated XOR value.

4.0 USB Descriptors

Augusta uses the following USB descriptors.

4.1.1 USB-HID

Device Descriptor:

Field	Value (Hex)	Description (ACR38)
Length	12	Length = 18
Des type	01	
BCD USB	00 02	USB 2.0
Device Class	00	Not specific
Sub Class	00	Unused
Device Protocol	00	Unused
Max Packet Size	40	Max Packet Size is 64 bytes
VID	0A CD	
PID	38 20 (39 20 for SRED)	
BCD Device Release	00 01	
i-Manufacture	01	
i-Product	02	
i-Serial-Number	03	
# Configuration	01	

Configuration Descriptor:

Field	Value (Hex)	Description
Length	09	
Des type	02	
Total Length	29 00	

No. Interface	01	
Configuration Value	01	
iConfiguration	00	
Attributes	A0	Bus power; With remote wakeup
Power	32	100 mA

Interface Descriptor (USB-HID):

Field	Value (Hex)	Description
Length	09	
Des type	04	
Interface No.	00	
Alternator Setting	00	
# EP	02	
Interface Class	03	HID
Sub Class	00	
Interface Protocol	00	
iInterface	00	

HID Descriptor:

Field	Value (Hex)	Description
Length	09	
bDescriptorType	21	HID
bcdHID(L/H)	11 01	Rev 1.11
bCountryCode	00	
bNumDescriptors	01	
bDescriptorType	22	Report
wDescriptorLength(L/H)	1C 00	L H

End Point Descriptor (EP2):

Field	Value	Description
Length	07	
Des Type	05	End Point
EP Addr	02	EP2 Out, Command Down
Attributes	03	Interrupt
wMaxPacketSize	40 00	64 Bytes
bInterval	04	4 milliseconds

End Point Descriptor (EP1):

Field	Value	Description
Length	07	
Des Type	05	End Point
EP Addr	81	EP1 In, Response Up
Attributes	03	Interrupt
Size	40 00	64 Bytes
Interval	04	4 milliseconds

4.1.2 USB-KB

Augusta can use USB-KB communication when in SecureMag-compatibility mode (magstripe only). For EMV/ICC-enabled operation, use USB-HID instead.

Device Descriptor:

Field	Value (Hex)	Description (ACR38)
Length	12	Length = 18
Des type	01	
bcd USB	00 02	USB 2.0
Device Class	00	Not specific
Sub Class	00	Unused
Device Protocol	00	Unused
Max Packet Size	08	Max Packet Size is 8 bytes
VID	0A CD	
PID	38 10 (39 10 for SRED)	
BCD Device Release	00 01	
i-Manufacture	01	
i-Product	02	
i-Serial-Number	03	
# Configuration	01	

Configuration Descriptor:

Field	Value (Hex)	Description
Length	09	
Des type	02	
Total Length	22 00	
No. Interface	01	
Configuration Value	01	
iConfiguration	00	
Attributes	80	Bus power; Without remote wakeup
Power	32	100 mA

Interface Descriptor (USB-HID):

Field	Value (Hex)	Description
Length	09	
Des type	04	
Interface No.	00	
Alternator Setting	00	
# EP	01	
Interface Class	03	HID
Sub Class	01	01:boot
Interface Protocol	02	02:keyboard

iInterface	00	

HID Descriptor:

Field	Value (Hex)	Description
Length	09	
bDescriptorType	21	HID
bcdHID(L/H)	11 01	Rev 1.11
bCountryCode	00	
bNumDescriptors	01	
bDescriptorType	22	Report
wDescriptorLength(L/H)	52 00	L H

End Point Descriptor (EP1):

Field	Value	Description
Length	07	
Des Type	05	End Point
EP Addr	81	EP1 In
Attributes	03	Interrupt
Size	08 00	8 Bytes
Interval	01	1 milliseconds

HID report descriptor for IDTech Key Board Protocol:

Field	Value (Hex)	Description
Usage Page	05 01	Generic Desktop
Usage	09 06	Keyboard
Collection	A1 01	Application
Usage Page	05 07	Key codes
Usage Minimum	19 E0	224
Usage Maximum	29 E7	231
Logical Minimum	15 00	0
Logical Maximum	25 01	1
Report Size	75 01	1
Report Count	95 08	8
Input	81 02	Data, Variable, Absolute
Report Count	95 01	1
Report Size	75 08	
Input	85 01	Constant
Report Count	95 05	5
Report Size	75 01	
Usage Page	05 08	LED
Usage Minimum	19 01	1
Usage Maximum	29 05	5
Output	91 02	Data Var Absolute
Report Count	95 01	1
Report Size	75 03	

Output	91 01	Constant
Report Count	95 06	6
Report Size	75 08	
Logical Minimum	15 00	0
Logical Maximum	25 66	102
Usage Page	05 07	Key codes
Usage Minimum	19 00	
Usage Maximum	29 66	102
Input	81 00	Data,Array
Usage Page	06 2D FF	ID TECH
Report Count	95 01	1
Logical maximum	26 FF 00	255
Logical Minimum	15 01	
Report Size	75 08	
Usage	09 20	Setup data byte
Report Count	95 08	
Feature	B2 02 01	Data Var, Abs
End Collection	C0	

4.2 USB KB Communication Command and Response Format

Augusta supports USB-KB communication only while in "Legacy Mode."
Legacy Mode is compatible with ID TECH's SecureMag magstripe-only card reader.

See ID TECH manual P/N 80096504-001, the *SecureMag Encrypted MagStripe Reader User Manual*, for information regarding the SecureMag (Legacy Mode) protocol and command set.

Note that USB-KB commands use ID TECH's ITP protocol, in which commands are wrapped with STX (0x02) and ETX (0x03), followed by an LRC value that represents the XOR of all preceding byte values (no checksum is used). For example:

Command: 52 22 (Get Firmware Version)

This command is sent as (hex) 02 52 22 03 71, where 0x71 is the LRC value.

4.3 USB HID Communication Command and Response Format

In EMV Hybrid Mode, Augusta uses ID TECH's proprietary NGA protocol format commands and responses in USB communications. The NGA protocol exchanges data using the following format:

<0x02> <Len_Low><Len_High> <Command Body / Response Body / Notification Body> <LRC> <CheckSUM> <0x03>

Where:

- <0x02> is STX (Start of Text)
- <Len_Low><Len_High> is the length of the <Command Body / Response Body / Notification Body>, in hexadecimal byte values.
- <LRC> is LRC (exclusive OR) of <Command Body / Response Body / Notification Body> byte values. (XOR all bytes together. He result is the LRC.)
- <CheckSUM> is SUM of <Command Body / Response Body / Notification Body> values, disregarding overflow. (Add all bytes; the result, mod-256, is the checksum.)
- <0x03> is ETX (End of Text)
- Response Body: [<Response Status> + <Response Data>]
  <Response Status>: 1 byte.
  <Response Data>: n bytes.
  If <Response Status> is ACK, Several bytes needed.
  If <Response Status> is NAK, Response data is Error code (2 bytes), or Error code (2 bytes) + Tag (1 or 2 bytes; this is for ICC L2 response).
- Command Body: [<Command> + <Command Parameters>]
- Notification Body: [xxxxxxxxxxxxxxxxxxxx]

Note that NGA commands tend to begin with a high nibble value 7:

Command Prefix	Category
72	ICC (smart card commands)
73	MSR (magstripe commands)
75	PCI Miscellaneous
78	Configuration commands
7F	Reserved

In a multi-byte NGA command, the first byte indicates the category (as shown above); the second byte is typically one of 'F', 'R', or 'S' (Function, Read, Set; hex 0x46, 0x52, or 0x53 respectively); and the third byte is the specific function ID.

4.3.1 Data Formats

Note that when a chip-card (ICC) transaction is performed, EMV data is returned using a tag-based (TLV) representation as described in detail in ID TECH P/N 80000502-001 *ID TECH Encrypted Data Output.*

When a conventional magstripe (MSR) transaction is performed, MSR data is returned following the ID TECH Enhanced Encrypted MSR Data Format described in ID TECH P/N 80000502-001 *ID TECH Encrypted Data Output.*

4.3.2 MSR Modes of Operation

MSR operation occurs in two modes: Auto or Buffer.

4.3.3 Auto Mode (default)

In Auto Mode, Augusta automatically sends out MSR data after a magnetic card is swiped. Auto Mode only supported in the USB-KB interface. Auto Mode can be disabled.

After MSR data is auto-sent out, the Buffer Data is erased directly.

4.3.4 Buffer Mode

In Buffer Mode, Augusta sends out MSR data in response to the Read Buffer Data command. Buffer Mode is supported in USB-HID interface or USB-KB interface. Buffer Mode can be disabled.

Implementation:

	USB-KB with ITP Protocol	**USB-HID with NGA Protocol**
Step 0	If the device is in Auto Mode or Disable Buffer Mode, set to Enable Buffer Mode.	If the device is Disable Buffer Mode, set to Enable Buffer Mode.
Step 1	Send “Arm to Read” Command. If ACK is received, enter the next step.	Send “Arm to Read” Command. If ACK is received, enter the next step.
Step 2	While in this state, a user can swipe their card. Timeout is 30 seconds. Send “Read Buffer Data” to read MSR data. The response should be: • NAK for No Swipe Card and No Timeout, and Buffer is erased. Or: • NAK for No Swipe Card and Timeout. Or: • ACK + “Successful MSR Data” for swipe OK, and Buffer is erased. Or:	While in this state, a user can swipe their card. Timeout is 30 seconds. Send “Read Buffer Data” to read MSR data. The response should be: • NAK + “No Swipe Card Error” for No Swipe Card and No Timeout, and Buffer is erased. Or: • NAK + “Timeout Error” for No Swipe Card and Timeout. Or: • “Successful MSR Data” for swipe OK, and Buffer is erased. Or:

	• NAK for swipe failed, and Buffer is erased. If Augusta sends "Read Buffer Data" before sending "Arm to Read," the response should be 0x18. After Swipe Card, the MSR data should exist at most 15 Minutes. If there is a timeout, the Buffer is erased.	• NAK + "Failed Get MSR Data" for swipe failed, and Buffer is erased. If Augusta sends "Read Buffer Data" before sends "Arm to Read," the response should be NAK + "Operate Error." After Swipe Card, the MSR data should exist at most 15 Minutes. If there is a timeout, the Buffer is erased.

5.0 EMV L2 Data Output Formats

5.1 Standard Transaction Mode

5.1.1 No Data encryption Key & No TransArmor Certificate:

ErrorResponse Body: 15 04 00

The Response Data – 02 <Len_L><Len_H>15 04 00<LRC><SUM> 03

5.1.2 TDES/AES mode & Data encryption Key exist:

Without MAC Data Output:
EMV L2 Transaction Result – 06 <2 bytes Response Code><Attribution><DF EF 12 0A (10 bytes KSN)><5A A1 08 (8 bytes Tag5A Mask Data)><5A C1 10 (16 bytes Tag5A Encryption Data)><57 A1 13 (19 bytes Tag57 Mask Data)><57 C1 18/20 (24 bytes/32bytes Tag57 Encryption Data)><9F 1F C1 N (N bytes Tag9F1F Encryption Data)><9F 20 n (n bytes Raw Value)><TLV1><TLV2> … <TLVn><DF EF 4C 06 00 26 00 10 00 00><DF EF 4D 38/40 (56/64 bytes Encrypted Trk1_Data Trk2_Data Trk3_Data PAN_Data)>

The Response Data – 02 <Len_L><Len_H> 06 <2 bytes Response Code><Attribution><DF EF 12 0A (10 bytes KSN)><5A A1 08 (8 bytes Tag5A Mask Data)><5A C1 10 (16 bytes Tag5A Encryption Data)><57 A1 13 (19 bytes Tag57 Mask Data)><57 C1 18/20 (24 bytes/32bytes Tag57 Encryption Data)><9F 1F C1 N (N bytes Tag9F1F Encryption Data)><9F 20 n (n bytes Raw Value)><TLV1><TLV2> … <TLVn><DF EF 4C 06 00 26 00 10 00 00><DF EF 4D 38/40 (56/64 bytes Encrypted Trk1_Data Trk2_Data Trk3_Data PAN_Data)><LRC><SUM> 03

With MAC Data Output:
EMV L2 Transaction Result – 06 <2 bytes Response Code><Attribution><DF EF 12 0A (10 bytes KSN)><5A A1 08 (8 bytes Tag5A Mask Data)><5A C1 10 (16 bytes Tag5A Encryption Data)><57 A1 13 (19 bytes Tag57 Mask Data)><57 C1 18/20 (24 bytes/32bytes Tag57 Encryption Data)><9F 1F C1 N (N bytes Tag9F1F Encryption Data)><9F 20 n (n bytes Raw Value)><TLV1><TLV2> … <TLVn><DF EF 4C 06 00 26 00 10 00 00><DF EF 4D 38/40 (56/64 bytes Encrypted Trk1_Data Trk2_Data Trk3_Data PAN_Data)><DF EF 41 10 (16 bytes MAC Value)><DFEF420A (10 bytes MAC Key KSN)>

The Response Data – 02 <Len_L><Len_H> 06 <2 bytes Response Code><Attribution><DF EF 12 0A (10 bytes KSN)><5A A1 08 (8 bytes Tag5A Mask Data)><5A C1 10 (16 bytes Tag5A Encryption Data)><57 A1 13 (19 bytes Tag57 Mask Data)><57 C1 18/20 (24 bytes/32bytes Tag57 Encryption Data)><9F 1F C1 N (N bytes Tag9F1F Encryption Data)><9F 20 n (n bytes Raw Value)><TLV1><TLV2> … <TLVn><DF EF 4C 06 00 26 00 10 00 00><DF EF 4D 38/40 (56/64 bytes Encrypted Trk1_Data Trk2_Data Trk3_Data PAN_Data)><DF EF 41 10 (16 bytes MAC Value)><DF EF 42 0A (10 bytes MAC Key KSN)><LRC><SUM> 03

5.1.3 TransArmor mode & TransArmor Certificate exist:

Without MAC Data Output:

Step1:
EMV L2 Transaction Result – 06 <2 bytes Response Code><Attribution><DF EF 12 0A (10 bytes KSN)><DF EE 26 02 (2 bytes data)><5A A1 08 (8 bytes Tag5A Mask Data)><5A C201 58 (344 bytes Tag5A TransArmor Data)><57 A1 13 (19 bytes Tag57 Mask Data)><57 C201 58 (344 bytes Tag57 TransArmor Data)><TLV1><TLV2> … <TLVn><DF EF 48 0A DF EF 4C DF EF 4D 9F 1F 9F 20>

- Response Data – 02 <Len_L><Len_H> 06 <2 bytes Response Code><Attribution><DF EF 12 0A (10 bytes KSN)><DF EE 26 02 (2 bytes data)><5A A1 08 (8 bytes Tag5A Mask Data)><5A C201 58 (344 bytes Tag5A TransArmor Data)><57 A1 13 (19 bytes Tag57 Mask Data)><57 C201 58 (344 bytes Tag57 TransArmor Data)><TLV1><TLV2> … <TLVn><DF EF 48 0A DF EF 4C DF EF 4D 9F 1F 9F 20><LRC><SUM> 03

Step2:
The terminal can send the "Retrieve Transaction Result" command with four tags (DFEF4C, DFEF4D, 9F1F and 9F20) to get the transaction results.

- Response Data – 06 <DF EF 4C 06 00 26 00 10 00 00><DF EF 4D 82 02 B0 (344 bytes Tag57 TransArmor Data) (344 bytes Tag5A TransArmor Data)><9F 1F C2 01 58 (344 bytes Tag9F1F TransArmor Data) ><DF EF 48 02 9F 20>
- Response Data – 02 <Len_L><Len_H> 06 <DF EF 4C 06 00 26 00 10 00 00><DF EF 4D 82 02 B0 (344 bytes Tag57 TransArmor Data) (344 bytes Tag5A TransArmor Data)><9F 1F C2 01 58 (344 bytes Tag9F1F TransArmor Data) ><DF EF 48 02 9F 20><LRC><SUM> 03

Step3:
The terminal can send the "Retrieve Transaction Result" command with four tags (9F20) to get the transaction results.

- Response Data – 06 <9F 20 C2 01 58 (344 bytes Tag9F20 TransArmor Data) >
- Response Data – 02 <Len_L><Len_H> 06 <9F 20 C2 01 58 (344 bytes Tag9F20 TransArmor Data) ><LRC><SUM> 03

With MAC Data Output:

Step1:
EMV L2 Transaction Result – 06 <2 bytes Response Code><Attribution><DF EF 12 0A (10 bytes KSN)><DF EE 26 02 (2 bytes data)><5A A1 08 (8 bytes Tag5A Mask Data)><5A C201 58 (344 bytes Tag5A TransArmor Data)><57 A1 13 (19 bytes Tag57 Mask Data)><57 C201 58 (344 bytes Tag57 TransArmor Data)><TLV1><TLV2> … <TLVn><DF EF 48 0A DF EF 4C DF EF 4D 9F 1F 9F 20><DF EF 41 10 (16 bytes MAC Value)><DF EF 42 0A (10 bytes MAC Key KSN)>

- Response Data – 02 <Len_L><Len_H> 06 <2 bytes Response Code><Attribution><DF EF 12 0A (10 bytes KSN)><DF EE 26 02 (2 bytes data)><5A A1 08 (8 bytes Tag5A Mask Data)><5A C201 58 (344 bytes Tag5A TransArmor Data)><57 A1 13 (19 bytes Tag57 Mask Data)><57 C201 58 (344 bytes Tag57 TransArmor Data)><TLV1><TLV2> … <TLVn><DF EF 48 0A DF EF 4C DF EF 4D 9F 1F 9F 20><DF EF 41 10 (16 bytes MAC Value)><DF EF 42 0A (10 bytes MAC Key KSN)><LRC><SUM> 03

Step2:
The terminal can send the “Retrieve Transaction Result” command with four tags (DFEF4C, DFEF4D, 9F1F and 9F20) to get the transaction results.

- Response Data – 06 <DF EF 4C 06 00 26 00 10 00 00><DF EF 4D 82 02 B0 (344 bytes Tag57 TransArmor Data) (344 bytes Tag5A TransArmor Data)><9F 1F C2 01 58 (344 bytes Tag9F1F TransArmor Data) ><DF EF 48 02 9F 20><DF EF 41 10 (16 bytes MAC Value)><DF EF 42 0A (10 bytes MAC Key KSN)>
- Response Data – 02 <Len_L><Len_H> 06 <DF EF 4C 06 00 26 00 10 00 00><DF EF 4D 82 02 B0 (344 bytes Tag57 TransArmor Data) (344 bytes Tag5A TransArmor Data)><9F 1F C201 58 (344 bytes Tag9F1F TransArmor Data) ><DF EF 48 02 9F 20><DF EF 41 10 (16 bytes MAC Value)><DF EF 42 0A (10 bytes MAC Key KSN)><LRC><SUM> 03

Step3:
The terminal can send the “Retrieve Transaction Result” command with four tags (9F20) to get the transaction results.

- Response Data – 06 <9F 20 C2 01 58 (344 bytes Tag9F20 TransArmor Data) ><DF EF 41 10 (16 bytes MAC Value)><DF EF 42 0A (10 bytes MAC Key KSN)>
- Response Data – <Len_L><Len_H> 06 <9F 20 C2 01 58 (344 bytes Tag9F20 TransArmor Data) ><DF EF 41 10 (16 bytes MAC Value)><DF EF 42 0A (10 bytes MAC Key KSN)><LRC><SUM> 03

5.2 Quick Chip and M/Chip Fast Transaction Mode

This mode of operation is used in conjunction with USB "Keyboard" mode (KB), a mode in which the card reader acts as a keyboard device, outputting data spontaneously upon card presentation.

5.2.1 No Data encryption Key & No TransArmor Certificate:

Error Response Body: 15 04 00

- The Response Data – 02 <Len_L><Len_H>15 04 00<LRC><SUM> 03

5.2.2 TDES/AESmode & Data Encryption Key exist:

Transaction OK:
EMV L2 Transaction Result – 06 <2 bytes Response Code><Attribution><DF EF 12 0A (10 bytes KSN)><DF EF 5B 08 (8 bytes Tag5A Mask Data)><5A 10 (16 bytes Tag5A Encryption Data)><DF EF 5D 13 (19 bytes Tag57 Mask Data)><57 18/20 (24 bytes/32bytes Tag57 Encryption Data)><9F 1F N (N bytes Tag9F1F Encryption Data)><9F 20 N (N bytes Raw Value)><TLV1><TLV2> … <TLVn><DF EF 4C 06 00 26 00 10 00 00><DF EF 4D 38/40 (56/64 bytes Encrypted Trk1_Data Trk2_Data Trk3_Data PAN_Data)>

Without MAC Data Output:
KB final output (ASCII Code) – <DF EE 25 02 (2 bytes Response Code)><DF EE 26 02 (2 bytes Attribution)><DF EF 12 0A (10 bytes KSN)><DF EF 5B 08 (8 bytes Tag5A Mask Data)><5A 10 (16 bytes Tag5A Encryption Data)><DF EF 5D 13 (19 bytes Tag57 Mask Data)><57 18/20 (24 bytes/32bytes Tag57 Encryption Data)><9F 1F N (N bytes Tag9F1F Encryption Data)><9F 20 N (N bytes Raw

Value)><TLV1><TLV2> … <TLVn><DF EF 4C 06 00 26 00 10 00 00><DF EF 4D 38/40 (56/64 bytes Encrypted Trk1_Data Trk2_Data Trk3_Data PAN_Data)>

With MAC Data Output:

KB final output (ASCII Code) – <DF EE 25 02 (2 bytes Response Code)><DF EE 26 02 (2 bytes Attribution)><DF EF 12 0A (10 bytes KSN)><DF EF 5B 08 (8 bytes Tag5A Mask Data)><5A 10 (16 bytes Tag5A Encryption Data)><DF EF 5D 13 (19 bytes Tag57 Mask Data)><57 18/20 (24 bytes/32bytes Tag57 Encryption Data)><9F 1F N (N bytes Tag9F1F Encryption Data)><9F 20 N (N bytes Raw Value)><TLV1><TLV2> … <TLVn><DF EF 4C 06 00 26 00 10 00 00><DF EF 4D 38/40 (56/64 bytes Encrypted Trk1_Data Trk2_Data Trk3_Data PAN_Data)><DF EF 41 10 (16 bytes MAC Value)><DF EF 42 0A (10 bytes MAC Key KSN)>

Note: msgX is <DF EE 25 02 (2 bytes Response Code)> … …<DF EF 4C 06 00 26 00 10 00 00><DF EF 4D 38/40 (56/64 bytes Encrypted Trk1_Data Trk2_Data Trk3_Data PAN_Data)><DF EF 41 10>

Transaction Error:

EMV L2 Transaction Result – 15 + <2 bytes Error Code>
KB final output (ASCII Code) - <DF EF 61 02 (2 bytes Error Code)>

5.2.3 TransArmormode &TransArmor Certificate exist:

Step1:

Transaction OK:

EMV L2 Transaction Result – 06 <2 bytes Response Code><Attribution><DF EF 12 0B (11 bytes KID)><DF EE 26 02 (2 bytes data)><DF EF 5B 08 (8 bytes Tag5A Mask Data)><5A 8201 58 (344 bytes Tag5A TransArmor Data)><DF EF 5D 13 (19 bytes Tag57 Mask Data)><57 8201 58 (344 bytes Tag57 TransArmor Data)><TLV1><TLV2> … <TLVn><DF EF 48 0A DF EF 4C DF EF 4D 9F 1F 9F 20>

Without MAC Data Output:

KB final output (ASCII Code) – <DF EE 25 02 (2 bytes Response Code)><DF EE 26 02 (2 bytes Attribution)><DF EF 12 0B (11 bytes KID)><DF EE 26 02 (2 bytes data)><DF EF 5B 08 (8 bytes Tag5A Mask Data)><5A 8201 58 (344 bytes Tag5A TransArmor Data)><DF EF 5D 13 (19 bytes Tag57 Mask Data)><57 8201 58 (344 bytes Tag57 TransArmor Data)><TLV1><TLV2> … <TLVn><DF EF 48 0A DF EF 4C DF EF 4D 9F 1F 9F 20>

With MAC Data Output:

KB final output (ASCII Code) – <DF EE 25 02 (2 bytes Response Code)><DF EE 26 02 (2 bytes Attribution)><DF EF 12 0B (11 bytes KID)><DF EE 26 02 (2 bytes data)><DF EF 5B 08 (8 bytes Tag5A Mask Data)><5A 8201 58 (344 bytes Tag5A TransArmor Data)><DF EF 5D 13 (19 bytes Tag57 Mask Data)><57 8201 58 (344 bytes Tag57 TransArmor Data)><TLV1><TLV2> … <TLVn><DF EF 48 0A DF EF 4C DF EF 4D 9F 1F 9F 20><DF EF 41 10 (16 bytes MAC Value)><DF EF 42 0A (10 bytes MAC Key KSN)>

Note: msgX is <DF EE 25 02 (2 bytes Response Code)> … … <DF EF 41 10>

Transaction Error:
EMV L2 Transaction Result – 15 + <2 bytes Error Code>

KB final output (ASCII Code) - <DF EF 61 02 (2 bytes Error Code)>

Step2:
Without MAC Data Output:
KB final output (ASCII Code) – <DF EF 4C 06 00 26 00 10 00 00><DF EF 4D 82 02 B0 (344 bytes Tag57 TransArmor Data) (344 bytes Tag5A TransArmor Data)><9F 1F 82 01 58 (344 bytes Tag9F1F TransArmor Data) ><DF EF 48 02 9F 20>

With MAC Data Output:
KB final output (ASCII Code) – <DF EF 4C 06 00 26 00 10 00 00><DF EF 4D 82 02 B0 (344 bytes Tag57 TransArmor Data) (344 bytes Tag5A TransArmor Data)><9F 1F 82 01 58 (344 bytes Tag9F1F TransArmor Data) ><DF EF 48 02 9F 20><DF EF 41 10 (16 bytes MAC Value)><DF EF 42 0A (10 bytes MAC Key KSN)>
Note: msgX is <DF EF 4C 06 00 26 00 10 00 00> … … <DF EF 41 10>

Step3:
Without MAC Data Output:
KB final output (ASCII Code) – <9F 20 82 01 58 (344 bytes Tag9F20 TransArmor Data) >

With MAC Data Output:
KB final output (ASCII Code) – <9F 20 82 01 58 (344 bytes Tag9F20 TransArmor Data) ><DF EF 41 10 (16 bytes MAC Value)><DF EF 42 0A (10 bytes MAC Key KSN)>
Note: msgX is <9F 20 82 01 58 (344 bytes Tag9F20 TransArmor Data) ><DF EF 41 10>

6.0 Device Commands

6.1 Get DUKPT Key KSN

This command gets the DUKPT Key KSN from the reader.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	09	00	78 46 3E 04 00 02 01 00 00	07	03	03

Command Body: 78 46 3E<Length of Data><KeyNameIndex><Length of Key Slot><Key Slot>

Where:

- <Length of Data> is 2 bytes, format is Len_L Len_H, is length of <KeyNameIndex><Length of Key Slot><Key Slot>
- <KeyNameIndex> is 1 byte, please refer to below table
- <Length of Key Slot> is 2 bytes, format is Len_L Len_H
- <Key Slot> is any byte (1 byte or 2 bytes), value always 0

KeyNameIndex	Key Name
0x14	LCL-KEK
0x02	Data encryption Key
0x0C	RKI-KEK

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	06	00	15 90 46 78 46 3E	C3	E7	03

Return Hex String: 020600159046784 63ec3e703

Response Body: 15 <Error Code>78 46 3E - Invalid related Key (90 42, or 9046, or 9052), or
Key STOP (73 00), or
Do Not Support the Key (90 47), or
06 78 46 3E <Length of KSN><KSN>
Where:

- <Length of KSN> is 2 bytes, format is Len_L Len_H, value always 0A 00
- <KSN> is 10 bytes KSN

6.2 Get Firmware Version

This command requests the Firmware Version from the reader.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	03	00	78 46 01	3F	BF	03
Output Hex String: 0203007846013FBF03						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	1E	00	06494420544543482041756775737461205553422D4849442056312E3030	34	12	03
Return Hex String: 021E00064944205445434820417567757374612055534 22D4849442056312E3030341203.						
Response Body ASCII: ID TECH Augusta USB-HID V1.00.						

6.3 Get Extended Firmware Version

This command requests the Extended Firmware Version from the reader. Use this command when you need to see the detailed version number of the firmware.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	02	00	78 31	49	A9	03
Output Hex String: 020200783149A903						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	26	00	022600064944205445434820417567757374612 053205553422d4849442056312e30312e303036 2e5323cb03	23	CB	03
Return Hex String: 022600064944205445434820417567757374612053205553422d4849442056312e30312e3030362e5 323cb03						
Response Body ASCII: ID TECH Augusta S USB-HID V1.01.006						

6.4 Enter into Bootloader

This command enters the reader into the bootloader.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	0F	00	78 46 7A 49 52 46 57 00 00 00 00 00 00 00 00	4E	70	03
Output Hex String: 020F0078467A4952465700000000000000004E7003						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	01	00	06	06	06	03
Return Hex String: 02010006060603						
Response Body Interpretation: 06 = ACK						

6.5 Get Serial Number

This command requests the Device Serial Number from the reader.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	03	00	78 46 02	3C	C0	03
Output Hex String: 0203007846023CC003						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	0B	00	06363032543332373833356	E2	E0	03
Return Hex String: 020B000636303254333237383335 6E2E03						
Response Body ASCII: 602T327835						

6.6 Get Model Number

This command requests Model Number from the reader.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	03	00	78 46 20	ED	E0	03
Output Hex String: 0203007846201EDE03						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	09	00	06 49 44 45 4D 2D 32 35 58	71	11	03
Return Hex String: 020900064944454D2D323558711103						
Response Body ASCII: IDEM-25X (Where IDEM-25X = USB HID, IDEM-24X = USB-KB)						

6.7 Get Model Status

This command retrieves the reader's model status.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	78 46 20			03
Command Body: 78 46 20						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00	06 + Model Status			03
Response Body: 06 + Model Status **Where:** Model Status is IDEM-25XX						

6.8 Reset

This command initiates a soft reset of the device.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	03	00	78 46 49	77	07	03
Output Hex String: 020300784649770703						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	01	00	06	06	06	03
Return Hex String: 02010006060603						

Response Body Interpretation: 06 = ACK

- The device will Reset (Re-Start) after it receives an ACK Response Body.
- This command is the Highest Priority Command in the device except for the Key Loading State.

6.9 Get Key Status

This command returns the overall key inventory and status for all keys loaded in the reader.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	03	00	78 46 30	0E	EE	03
Output Hex String: 0203007846300EEE03						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	07	00	06 < PIN DUKPT Status> + <PIN Master Key Status> + <PIN Session Key Status> + <Account/MSR DUKPT Key Status> + <Account/ICC DUKPT Key Status> + <Admin DUKPT Key (RKI-KEK key)>	07	09	03
Return Hex String: 02070006000000010101070903 (Example)						

Response Body: 06 + < PIN DUKPT Status> + <PIN Master Key Status> + <PIN Session Key Status> + <Account/MSR DUKPT Key Status> + <Account/ICC DUKPT Key Status> + <Admin DUKPT Key (RKI-KEK key)>

Where:

Key	Status	Note
PIN DUKPT Key	00: None. 01: Exist 0xFF: STOP	Always 00
PIN Master Key	00: None 01: At least Exist a Master Key	Always 00
PIN Session Key	00: None. 01: Exist	Always 00
Account/MSR DUKPT Key	00: None. 01: Exist FF: STOP	
Account/ICC DUKPT Key	00: None. 01: Exist FF: STOP	
Admin DUKPT Key (RKI-KEK key)	00: None. 01: Exist FF: STOP	

6.10 Get Status for Key

This command retrieves the status of the currently-set key.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	03	00	78 46 25	1B	E3	03
Output Hex String: 0203007846251BE303						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	0F	00	060300140000000200000000C000000	1F	2B	03
Return Hex String: 020F00060300140000000200000000C0000001F2B03						

Response Body:
06 <Block Length> <KeyStatusBlock1> <[KeyStatusBlock2]> …<[KeyStatusBlockN]>, or
15 <Error Code>

Where:

- <Block Length> = 2 bytes, (Low, High).
- <KeyStatusBlockX> = 4 bytes, format is <Key Index and Key Name> <Key Slot> <Key Status>.
    - **Where:** Key Index = 1 byte, Key Slot = 2 bytes (range 0000-9999), Key Status = 1 byte.
    - **Where:** Key Status = 00 Key does not exist, 01 Key exists, FF = Stop (DUKPT only).

KeyNameIndex	Key Name	Definition	Key Slot
0x14	HSM KEK DUKPT Key	KEK	0
0x02	Data (Account) DUKPT Key	Encrypt ICC Data	0
0x0C	Admin DUKPT Key (RKI-KEK key)	Remote Key Injection	0

6.11 Retrieve White List

This command retrieves a white list for the device.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02			78 46 4D 01	72	0C	03
Output Hex String: 02040078464D01720C03						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02			06 <Length Whitelist ASN.1 BLK><Whitelist ASN.1 BLK> (See below)			03

Response Body: 06 <Length Whitelist ASN.1 BLK><Whitelist ASN.1 BLK>

Example: 021800061500301302010131 0e300c130002030b019402030b01f7419d03

Where:

- <Length Whitelist ASN.1 BLK> is 2 bytes, format is LenL LenH, it is length of <Whitelist ASN.1 BLK>.
- <Whitelist ASN.1 BLK> is N bytes, it is ASN.1 Block data for White List for Gift Card.

6.12 Remove White List

This command removes a white list for the device.

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	04		78 46 4D 02	71	0D	03
Output hex string: 02040078464D02710D03						

Response Body: 06

6.13 Set White List

This command sets the white list for the device. See Appendix L: White List Format for information about white list structure.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02			78 46 4D 03			03

Command body: 46 4D 03 <Length Whitelist ASN.1 BLK><Whitelist ASN.1 BLK><2 bytes MAC Length><MAC Data>

Where:

- <Length Whitelist ASN.1 BLK> is 2 bytes, format is LenL LenH; it is the length of <Whitelist ASN.1 BLK>.
- <Whitelist ASN.1 BLK> is N bytes; it is ASN.1 Block data for White List for Gift Card.
- <MAC Length> is 2 byte data – the length of <MAC Data>.
- Is fixed to 0x00 0x00 for non-PCI devices.
- Is fixed to 0x1E 0x00 for PCI devices.
- <MAC Data>:
    - Does not exist for non-PCI devices.
    - For PCI devices, MAC is Fixed 30 bytes data:
        - <MAC Value Length> is 2 byte and fixed to 0x10 0x00
        - <MAC Value> is 16 bytes – MAC value is MAC-HOST. The msgX is "46 4D 03 <Length Whitelist ASN.1 BLK><Whitelist ASN.1 BLK><2 bytes MAC Length><MAC Value Length>"
        - <MAC Key KSN Length> is 2 byte and fixed to 0x0A 0x00
    - <MAC Key KSN> is 10 bytes – MAC DUKPT Key KSN

Note:

1. The new White List overwrites the old one.
2. The White List is saved using ASN.1 Block format.
3. The Total Length of <Length Whitelist ASN.1 BLK><Whitelist ASN.1 BLK> is not more than 512.

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	01	00	06	06	06	03
Response Body: 06 = ACK						

6.14 Set Data Encryption Key Variant

This command sets the Data Encryption Key variant.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	03	00	78 53 01 02 01 <Option>	72	0C	03
Command Body: 78 53 01 02 01 <Option>						

Key Variant	Option
Data Key	**0x00**
PIN Key	0x01

Response Body: 06 (Data encryption Key did not exist), or
15 6A 00 (Data encryption Key existed).

6.15 Get Data Encryption Key Variant

This command requests the currently-set Data Encryption Key variant.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	03	00	78 52 01 02	29	CD	03
Output Hex String: 0204007852010229cd03						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	06	00	06 78 01 02 01 <Option>	7C	82	03
Output Hex String: 02060006780102010 07c8203 In this example, Option is 0x00 (Data Key). 0x01 is the PIN key.						

6.16 Get Data Encryption Key Encryption / Decryption Mode

This command requests the currently-set Data Encryption Key encryption / decryption mode.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	04	00	78 52 01 03	28	CE	03
Output Hex String: 0204007852010328ce03						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	06	00	06 78 01 03 01 <Option>	7D	83	03
Output Hex String: 0206000678010301007d8303						

Encryption/Decryption Mode	Option
TDES	**0x00**
AES	0x01

6.17 Set Encryption Control

Option is a one-byte value that contains MSR encryption status (ON or OFF) in the zero bit (1 or 0), and ICC encryption status (ON or OFF) in the No. 1 bit (1 or 0).

Note: Turning on encryption when a key is loaded is a **one-time setting**. Encryption **cannot be disabled after it has been enabled**.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	07	00	78 53 01 07 02 <Option> 00	2C	D8	03
Output Hex String: 02070078530107020300 2cd803						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	01	00	06	06	06	03
Response Body: 06 = ACK						

6.18 Get Encryption Control

This command retrieves the reader's current encryption control settings.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	04	00	78 52 01 07	2C	D2	03
Output Hex String: 020400785201072cd203						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	01	00	06 78 01 07 02 <Option> <00>	7A	88	03
Response Body: 02070006780107020000 7a8803. (The Option byte here is 0x00, meaning encryption has not yet been turned on.) <Option> is a one-byte value that contains MSR encryption status (ON or OFF) in the zero bit (1 or 0), and ICC encryption status (ON or OFF) in the No. 1 bit (1 or 0).						

6.19 Set Verify Encrypt Data Output Option

This command sets the Verify Encrypt Data Output option. Use this command to control whether or not MAC data (an authenticated hash of track data) appears in data output.

Command Body: 78 53 01 08 01 <Option>

Where:

Option Value	Option
0x00	**Encrypted Data Output without MAC Data**
0x01	Encrypted Data Output with MAC Data

Note: Option Value can be set to On (MAC Data included in output), or disabled if set to Off (no MAC Data in output).

Response Body: 06

6.20 Get Verify Encrypt Data Output Option

This command requests the currently-set Verify Encrypt Data Output option.

Command Body: 78 52 01 08

Response Body: 06 78 01 08 01 <Option>, where Option can be 0x00 (no MAC data) or 0x01 (MAC data enabled).

6.21 Set Date & Time

This command sets the Date and Time programmed in to the reader.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	0D	00	78 53 01 50 08 06 17 03 27 02 14 00 00	51	81	03

Output Hex String: 020D0078530150080617032702140000518103.

The example above sets date and time as follows:
Year = 2017, Month =03, Date = 27, Hour = 02 (2:00 am), Minutes = 14, Seconds = 00.

Command Body Format:
78 53 01 50 <FunLen> <Date/Time Length> <Date/Time> <MAC Length> <MAC Data>.
Where:

- <FunLen> = 1 byte (Represents the length of:
  - <Date/Time><Date/Time><MAC Length><MAC Data>
  - (Augusta = 0x08, Augusta S = 0x26)
- <Date/Time Length> = 1 byte (Fixed value of 0x06).
- <Date/Time> = 6 bytes of data: Year, Month, Date, Hour, Minute, Second.
  - (see table below for values)

Item	Value Area (BCD Code)
Year	15~99, 00
Month	01~12
Date	01~31
Hour	00~23
Minute	00~59
Second	00~59

- <MAC Length> = 1 byte (Length of MAC data)
  - (Augusta = 0x00, Augusta S = 0x1E)
  - **Note: MAC only required on PCI SRED version: Augusta-S.**
- <MAC Value Length> = 2 bytes (Fixed value of: 0x10 0x00)
  <MAC Value> = 16 bytes

MAC Value is MAC-HOST. The msgX is:
78 53 01 50 <FunLen><Date/TimeLen><Date/Time><MAC Length><MAC Value Length>.

- <MAC Key KSN Length> = 2 bytes (Fixed value of: 0x0A 0x00
- <MAC Key KSN> = 10 bytes (MAC DUKPT Key KSN)

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	01	00	02010006060603	06	06	03
Return Hex String: 02010006060603.						
Response Body Interpretation: 06 = ACK.						

6.22 Set TransArmor RSA TID

This command sets the reader's TransArmor RSA TID.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	78 53 01 40 08 <Ascii code TID>			03
Command Body: 78 53 01 40 08 <Ascii code TID> Note: This command is only Valid in firmware support TransArmor RSA algorithm.						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00	06			03
Response Body: 06 = ACK						

6.23 Get TransArmor RSA TID

This command retrieves the reader's current TransArmor RSA TID.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	78 52 01 40			03
Command Body: 78 52 01 40 Note: This command is only Valid in firmware support TransArmor RSA algorithm.						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00	See below			03
Response Body: 06 78 01 40 08 <Ascii code TID> or 15 62 02 (No TransArmor RSA TID).						

6.24 Load Certificate for TransArmor RSA Algorithm

This command loads the reader's TransArmor RSA Algorithm certificate.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	78 46 12 <Length of Certification> <Certification, X.509, PEM format>			03

Command Body: 78 46 12 <Length of Certification> <Certification, X.509, PEM format>

Where:

- <Length of Certification> is 2 bytes, format is LenL LenH, it is length of <Certification, X.509, PEM format>
- The Certification Chain includes the following:
  - Root CA Certification
  - Intermediate CA Certification
  - TA Key Certification
- The detailed:
  - Root CA Certification is hardcoded in firmware. The **Certificate** needs to be loaded is **Intermediate CA Certificate** or **TA Key Certificate**.
  - POS software sends Key Update message requesting for Intermediate CA Certificate and it receives the Intermediate CA Certificate in the Key Update response from First Data Server. POS software passes the Intermediate CA Certificate to Card Reader. The Card Reader uses the Root CA Key to validate the Intermediate CA Certificate. After the Intermediate CA Certificate is validated, the Root CA Key extracts the Intermediate CA Key from the Intermediate CA Certificate.
  - POS software sends Key Update message requesting for TA Key Certificate and it receives the TA Key Certificate in the Key Update response from First Data Server. POS software passes the TA Key Certificate to Card Reader. The Card Reader uses the Intermediate CA Key to validate the TA Key Certificate. After the TA Key Certificate is validated, the Intermediate CA Key extracts the TA Key. The TA Key is used to encrypt the sensitive cardholder data in the TA authorization request. TA Key Certificate uses TA Key ID as Common Name.
- The more detailed:
  - If Intermediate CA Certification is loaded successfully, TA Key Certification should be erased immediately.

Note: This command is only Valid in firmware support TransArmor RSA algorithm.

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00	06			03
Response Body: 06 = ACK						

6.25 Read Certificate for TransArmor RSA Algorithm

This command reads the reader's TransArmor RSA Algorithm certificate.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	78 46 13 <Certificate Type>			03

Command Body: 78 46 13 <Certificate Type>

Where:

- <Certificate Type> - 1 byte

Certificate Type Value	Certificate Type
0x01	Intermediate CA Certificate
0x02	TA Key Certificate

Note: This command is only Valid in firmware support TransArmor RSA algorithm.

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00	06 <Certificate Type> <Length of Certification> <Certification, X.509, PEM format>			03

Response Body: 06 <Certificate Type> <Length of Certification> <Certification, X.509, PEM format>

Where:

- <Length of Certification> is 2 bytes, format is LenL LenH, it is length of <Certification, X.509, PEM format>
- The Certification Chain includes the following:
  - Intermediate CA Certification
  - TA Key Certification

6.26 Get Date & Time

This command requests the Date and Time programmed in the reader.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	04	00	78 52 01 50	7B	1B	03

Output Hex String: 020400785201507B1B03.

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	0B	00	0678015006160130140018	02	48	03

Return Hex String: 020B000678015006160130140018024803.

Response Body Interpretation: 067801500**6160130140018**.

Where: 6 highlighted bytes represent: Year, Month, Date, Hour, Minute, Second

16 = 2016, 01 = January, 30 = Date, 14 = 1400 hours military time, 00 = Minute, 18 = Seconds

6.27 Set Interface Type

This command sets the Interface Type that the reader is using.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	04	00	78 53 01 10 02 02 00	3B	DB	03
Command Format: 78 53 01 10 02 <Interface ID> <NC>. **Where:** • <Interface ID> = 1 byte, (only 0x02 USB-KB is valid) • <NC> = 1 byte, reserved **Warning:** If the device interface is changed from USB-HID to USB-KB: 1. The device will not support ICC function. 2. All Terminal Data / Application Data / CA Public Key will be erased.						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	07	00	020100**06**060603	C9	92	03
Return Hex String: 02070006780110020100 6C9203.						
Response Body Interpretation: 06780110020100 **Where:** highlighted byte represents: 06 = ACK						

6.28 Get Interface Type

This command requests the Interface Type that the reader is using.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	04	00	78 52 01 10	3B	DB	03
Output Hex String: 020400785201103BDB03						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	07	00	06780110020100	C9	92	03
Return Hex String: 02070006780110020100 6C9203						
Response Body: 06 78 01 10 02 <Interface ID> 00. **Where:** <Interface ID> is 1 byte, e.g. 0x01 (USB-HID).						

6.29 Set LED Control

This command sets the LED control.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	0D	00	78 53 01 11 08 <MSR LED Option> <ICC LED Option> <6 bytes 00>	33	E7	03

Output Hex String: 020d007853011108010100000000000033e703.

Where:

- <MSR LED Option> is a byte

MSR LED Status	Option
FW Control	**0x00**
SW Control	0x01

- <ICC LED Option> is a byte

ICC LEDStatus	Option
SW Control Only	**0x01**

Response Body: 0x06, complete response is 02010006060603

6.30 Get LED Control

This command requests the currently-set LED control.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	04	00	78 52 01 11	3A	DC	03

Output Hex String: 020400785201113adc03

Response Body: 06 78 01 11 08 <MSR LED Option> <ICC LED Option> <6 bytes 00>.

See Set LED Control (above) for Options.

6.31 Set Beeper Control

This command sets the beeper control. To set the beeper control, send **78 53 01 12 04 01 00 00 00** to set it to the SW control. Then, use SW to send the 78 46 04 command to control beeper frequency and duration.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	09	00	78 53 01 12 04 01 00 00 00	3D	E3	03

Output Hex String: 020900785301120401000000 3de303

Command Body: 78 53 01 12 04 <Beeper Option> <3 bytes 00>.

Where:

Beeper Status	Option
FW Control	**0x00**
SW Control	0x01

Response Body: 0x06.

6.32 Beeper Control

This command controls beeper frequency and duration.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
			78 46 04			

Command Body: 78 46 04 <Beeper Index><Beeper (Index) Control>

Where:

- <Beeper Index> - 1 byte. The only accepted value is 0x01.
- <Beeper (1) Control>data format is<Frequency_H><Frequency_L><Duration_H><Duration _L>
    - <Frequency_H><Frequency_L> is Frequency, size is 1K~20K (Suggest 3K~20K: More than 55 dBSPL)
    - <Duration_H><Duration _L> is Duration, size is 1~65535millseconds.

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
			06 78 46 04			
Response Body: 06 78 46 04 or 15 6A 02 (Beeper SW Control Disable – FW Control) or 15 6D 00 (Beeper Control Error)						

6.33 Get Beeper Control

This command requests the currently-set beeper control.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	09	00	78 52 01 12	39	DD	03
Output Hex String: 0209007852011239DD03						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	09	00	06 78 01 12 04 00 00 00 00	69	95	03
Return Hex String: 02090006780112040000000069950 3 **Response Body:** 06 78 01 12 04 <Beeper Option> <3 bytes 00> See Set Beeper Control (above) for Options.						

6.34 Default General Group

This command resets the Remote Key Injection Timeout and Data Encryption Key Variant settings to their default values.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	03	00	78 53 00	2B	CB	03

Output Hex String: 0203007853002bcb03

Response Body: 06

The settings below will be reset to their default value:

Function Name	Default Value
Remote Key Injection Timeout	120 Seconds
LED Control	All FW Control
Beeper Control	FW Control
Encryption Control	ICC On and MSR On
TransArmor TID	None

6.35 Review General Group

This setting reviews the Remote Key Injection Timeout and Data Encryption Key Variant settings.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	03	00	78 52 00	2A	CA	03

Output Hex String: 0203007852002aca03

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	2C	00	06 78 08 01 02 00 78 02 01 00 03 01 00 07 02 00 00 10 01 01 11 08 01 01 00 00 00 00 00 00 12 04 01 00 00 00 50 06 16 05 13 13 02 30	60	1E	03

Return Hex String:
022c000678080102007802010003010007020000100101110801010000000000001204010000005006160513130230601e03

Response Body: 06 78 08 01 02 <Timeout_H><Timeout_L>02 01 <Data encryption KeyVariant Option> 03 01 <Data encryption Key Encryption / Decryption Mode Option>10 02<Interface ID> 00 50 06<Date Time>

7.0 Smart Card Group (ICC EMV Level One Task) – Function Command

7.1 Low Level Commands

7.1.1 A Note on ICCL1 Operation Timing

External Command Exchanging has an Operation Timer for ICC full-speed work. The timer is 8~90 Seconds (the default is 8 seconds; the timeout can be changed).

1. After the card powers on successfully, the timeout starts.
2. Per APDU exchange, the timeout re-starts.
3. After timing out, the device powers down the card automatically and quits ICC work status.

7.2 Get ICC Reader Status

This command requests the ICC (chip card) reader status and attempts to determine if an ICC card is inserted in a slot by checking the "card seated" switch status. If a card is present, the reader attempts a power up.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	03	00	72 46 24	0D	DC	03
Output Hex String: 02030072462410DC03						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	07	00	06 00	60	06	03
Return Hex String: 02020006**00**060603						

Response Body Interpretation:

Where: highlighted byte represents:

00 = Card not Seated, ICC not powered	(No card present in ICC slot)
02 = Card Seated, ICC Not powered	(Card inserted, improper orientation)
03 = Card Seated, ICC powered	(Card inserted properly, ICC powered normally)

Note: the lowest bit is a flag for power on/off. The next bit is a flag for card seated. The possible states are 0, 2, or 3; there is no possibility of 1 because the device will not turn ICC power on without a card seated.

7.3 ICC Power On (Get ATR)

This command attempts to Power On the ICC and return ATR from the card.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	03	00	72 46 6E	5A	26	03
Output Hex String: 02030072466E5A2603						

Response Example (Normal – Card Inserted, ICC Powered, ATR Response)

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	10	00	063B6b00000031C0643F68010307900	01	43	03
Return Hex String: 02100006**3B6B00000031C0643F680103079000**014303						
Response Body Interpretation: **Where:** highlighted byte represents: ATR = 3B6B00000031C0643F680103079000						

Response Example (No Card Inserted in ICC Slot)

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	03	00	152C06	3F	47	03
Return Hex String: 020300152C063F4703						
Response Body Interpretation: **Where:** highlighted byte represents: 2C06 = No card seated. (See Error Code in Appendix.)						

Response Example (Card Inserted incorrect orientation in ICC Slot)

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	03	00	158B10	8E	B0	03
Return Hex String: 020300158B108EB003						
Response Body Interpretation: **Where:** highlighted byte represents: 8B10 = ICC error on power-up. (See Error Code in Appendix.)						

7.4 Power Off

This command turns off power to the ICC card reader.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	03	00	72 46 4D	79	05	03
Output Hex String: 02030072464D790503						

Response Example (Normal – Card Inserted, ICC Powered, ATR Response)

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	01	00	06	60	06	03
Return Hex String: 020100**06**060603						
Response Body Interpretation: **Where:** highlighted byte represents: 06 = ACK (ICC power turned off)						

7.5 Exchange APDU Plaintext

This command turns allows exchange of APDUs in plaintext (non-encrypted) format. The command only works when no DUKPT Data Protection Key is loaded; if a DUKPT Data Protection Key is loaded, the command is not supported.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	72 46 41 <C-APDU>			03
Output Hex String: 02170072464100a404000e315041592e5359532e444446303100bc5403 Here, the Command APDU (which, in this case, is: Select File 1PAY.SYS.DDF01 to get the PSE directory) is represented by 00 A4 04 00 0E 31 50 41 59 2E 53 59 53 2E 44 44 46 30 31 00.						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00	06 00 <R-APDU>			03
Return Hex String: 022c0006006f26840e315041592e5359532e4444463031a5148801015f2d02656e9f110101bf0c059f4d020b0a9000998b03						
Response Body Interpretation: STX + Length bytes + ACK + R-APDU + LRC + SUM + ETX						

NOTES:

1. This command will only work when no DUKPT Data Protection Key is loaded.
2. After a reader receives this command, the unit should stop Operation Timing.

3. After a reader responds to the command successfully, the reader should start Operation Timing. If a timeout occurs, the reader should stop the Timer and power off ICC automatically.

7.6 Exchange APDU Encryption for special case

This command exchanges APDU Encryption for special cases.

Command Body: 72 46 61 <Plaintext C-APDU>

Response Body: 06 + 00 + <Plaintext R-APDU> or
06 + 01 + <KSN Length><KSN><n bytes Encrypted R-APDU without Status Bytes> <2 bytes Status Bytes> or
06 + 01 + 00 + <n bytes Encrypted R-APDU without Status Bytes> <2 bytes Status Bytes>

Note:
1. If a Data Encryption Key was not loaded, the unit should respond with Error Code (04 00) for this command.
2. If a Data Encryption Key is at the end of its useful life, the unit should respond with Error Code (73 00) for this command.
3. After a unit receives this command, the unit should stop Operation Timing.
4. After a unit responds to the command successfully, the unit should start Operation Timing. If a timeout occurs, the unit should stop the Timer and power off ICC automatically.

7.7 Get KSN

Command Body: 78 46 3E

Response Body: 06 + <10 bytes KSN>

Note:
1. If a Data Encryption Key was not loaded, the unit responds with Error Code (04 00) for this command.
2. If a Data Encryption Key is at the end of its useful life, the unit responds with Error Code (73 00) for this command.
3. The KSN in the response should be the KSN exchanging in the loop until ICC is powered off.
4. After a unit receives this command, the unit will stop Operation Timing.
5. After a unit responds to the command successfully, Unit will start Operation Timing. If timeout occurred, Unit will stop the Timer and power off ICC automatically.

7.8 Exchange APDU Encryption

Note:
1. If a Data Encryption Key was not loaded, the unit responds with Error Code (04 00) for this command.
2. If a Data Encryption Key is at the end of its useful life, the unit responds with Error Code (73 00) for this command.
4. After a unit receives this command, the unit should stop Operation Timing.
4. After a unit responds to the command successfully, Unit will start Operation Timing. If timeout occurred, Unit will stop the Timer and power off ICC automatically.

7.9 Get EMV Level One Version Number

This command retrieves the version number of the EMV L1 interface (IFM).

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	04	00	72 46 23 01	16	DC	03
Output Hex String: 0204007246230116DC03						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	0A	00	06 49 46 4D 20 56 31 2E 31 30	2C	18	03
Return Hex String: 020A000649464D2056312E31302C1803						
ASCII String: IFM V1.10						

8.0 Smart Card Command Group (ICC EMV Level 2)

8.1 Setting transaction parameters

Generally, before attempting to process an EMV transaction, you will complete various one-time-only configuration and setup commands. These are commands are typically invoked only during setup; they do not need to be used on a per-transaction basis:

1. Send Set Application Data Command
2. Send Set Terminal Data Command
3. Send Set CA Public Key Command
4. Send Set Certification Revocation List command

Terminal configuration

Augusta Supports the 2C configuration; the Terminal Data Values for 2C configuration are listed below and are set by manufacturers. For more detailed information, refer to Appendix H.

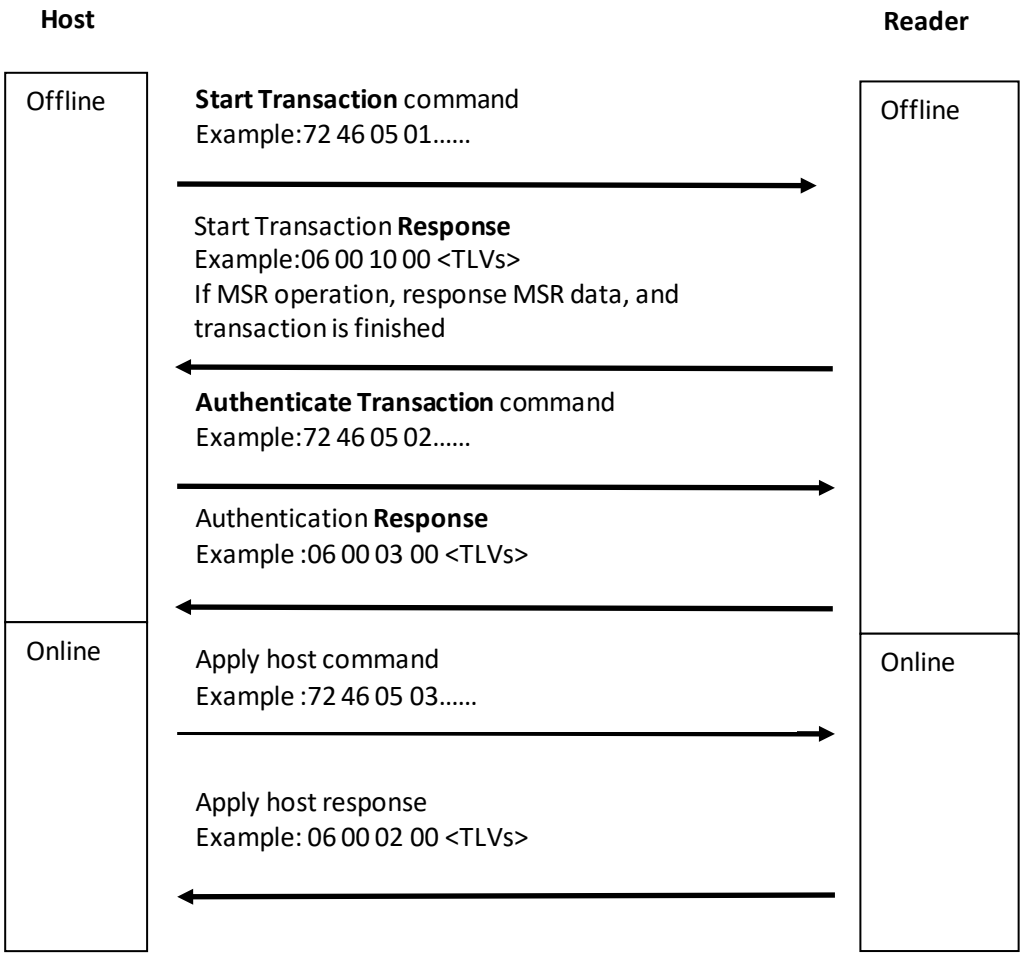
Item	Terminal Data (TLV format)	Definition
1	9F 35 01 21	Terminal Type
2	9F 33 03 60 28 C8	Terminal Capability
3	9F 40 05 F0 00 F0 A0 01	Additional Terminal Capability
4	DF 26 01 01	Terminal Supports CRL
5	DF 10 02 65 6E	Language
6	DF 11 01 00	Not Support Transaction Log
7	DF 27 01 00	Not support Exception File
8	DF EE 15 01 01	Terminal Support ASI
9	DF EE 16 01 00	Terminal Encrypt Mode: 00=DUKPT 01=MK/SK
10	DF EE 17 01 07	Terminal Entry Mode for ICC
11	DF EE 18 01 80	Terminal Encrypt Mode for MSR
12	DF EE 1E 08 D0 DC 20 D0 C4 1E 16 00	Contact Terminal Configuration
13	DF EE 1F 01 80	Issuer Script Limit
14	DF EE 1B 08 30 30 30 31 35 31 30 30	ARC
15	DF EE 20 01 3C	ICC Power on detect waiting time
16	DF EE 21 01 0A	ICC L1 waiting time
17	DF EE 22 03 32 3C 3C	Driver time. byte 1 -> Menu. byte 2 -> Get PIN. byte 3 -> MSR

EMV Transaction Overview

Set Transaction Parameters
Prior to an EMV transaction, set the following transaction parameters:

1. Send Set Application Data Command;
2. Send Set Terminal Data Command;
3. Send Set CA Public Key Command;
4. Send Set Certification Revocation List command

Transaction Flow



1. The host sends a **Start Transaction** command.
2. The host sends an **Authenticate Transaction** command.
3. The host sends a Receive Online Request, then sends a **Complete Transaction** command.

4. When a transaction is completed, the interface outputs ACK+TLV format data list, and a transaction result. To review the transaction result, send the **Retrieve Transaction Result** command.

NOTE:

1. For each step above, upon receipt of ACK, execute next the step. Otherwise, repeat until ACK is returned.
2. The **Retrieve EMV Level Two Version Number** command can be sent at any time.
3. The **Cancel Transaction** command can be sent at any time to terminate a transaction.

8.2 Retrieve Application Data (Retrieve AID)

This command retrieves EMV L2 Application Data (AID) from a reader.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	72 46 01 01…			03
Command Body: 72 46 01 01 <LenL><LenH> <5~16 bytes AID> **Where:** <LenL><LenH> is the length of AID						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00				03
Response Body: 06 <TagCounterL> <TagCounterH> <TLV1> <TLV2>…<TLVn> or 15<ErrorCode>. **Where:** <TagCounterL> <TagCounterH> is the number of <TLV> **Note:** If AID List / Application Data does not exist, response 15 F2						

8.3 Remove Application Data

This command removes EMV L2 Application Data (AID) from a reader.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	72 46 01 02…			03
Command Body: 72 46 01 02 <LenL><LenH> <5~16 bytes AID> **Where:** <LenL><LenH> is the length of AID						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00				03
Response Body: 06 = ACK or 15<ErrorCode> **Note:** If AID List / Application Data does not exist, the response is 15 F2 00						

8.4 Set Application Data (Set AID)

This command sets/sends the EMV L2 Application Data (AID) to the reader.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	72 46 01 03…			03
Command Body: 72 46 01 03 <LenL><LenH> <5~16 bytes AID> <TagCounterL> <TagCounterH> <TLV1> <TLV2>…<TLVn> **Where:** <TagCounterL> <TagCounterH> is the Number of <TLV> <LenL><LenH> is the length of the AID						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00				03
Response Body: 06 = ACK or 15 <ErrorCode> **Note:** If a <TLV> format has error, response: 15 F2 02 If AID List is full (MAX 16), response: 15 F2 03						

8.5 Remove All Application Data (Remove AIDs)

This command erases all EMV L2 Application Data (AIDs) from the reader.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	72 46 01 04			03

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00				03
Response Body: 06 = ACK						

8.6 Retrieve Terminal Data

This command retrieves all EMV L2 Terminal Data from the reader.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	04	00	72 46 02 01	37	BB	03
Output Hex String: 0204007246020137BB03						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	C9	00	06 … TLV Counter, TLV Tags	25	8B	03

Return Hex String:

02C9000618005F3601029F1A0208409F3501229F3303E028C89F4005F000F0A0019F1E085465726D696E616C9F150212349F160F3030303030303030303030303030309F1C0838373635343332319F4E223130373231205761 6C6B65722053742E20437970726573732C204341202C5553412EDF260101DF1008656E667265737A68DF110101DF270100DFEE150101DFEE160100DFEE170107DFEE180180DFEE1E08F0DC3CF0C29E9400DFEE1F0180DFEE1B0830303031353130 30DFEE20013CDFEE21010ADFEE2203323C3C258B03

Response Body:

06 <TagCounterL> <TagCounterH> <TLV1> <TLV2>…<TLVn> or
15<ErrorCode>

Where: <TagCounterL> <TagCounterH> is the number of <TLV>
If Terminal Data does not exist, the response is 15 F2 01

8.7 Remove Terminal Data

This command erases all EMV L2 Terminal Data from the reader.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	72 46 02 02			03

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00				03
Response Body: 06 = ACK						

8.8 Set Terminal Data

This command sets/sends the EMV L2 Terminal Data to the reader.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	72 46 02 03…			03
Command Body: 72 46 02 03 <TagCounterL> <TagCounterH> <TLV1> <TLV2>…<TLVn> **Where:** <TagCounterL> <TagCounterH>: the Number of <TLV>						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00				03
Response Body: 06 or 15 <ErrorCode> **Note:** If a <TLV> format error, response: 15 F2 02						

8.9 Retrieve AID List

This command retrieves the EMV L2 AID (Application Identifier List) from the reader.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	04	00	72 46 03 01	36	BC	03
Output Hex String: 0204007246030136BC03						
Command Body: 72 46 01 01 <LenL><LenH> <5~16 bytes AID>						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	8E	00	See below	12	42	03
Return Hex String: 028E00060E000700A0000000031010060 0A0000099990108 00A0000000031010030800A00000000 31010040800A0000000031010050800A0000000031010060800A0000000031010070700A0000000 0410100700A0000000651010080 0A0000000250105010700A0000001523010080 0A000000333010 1020500A1223344551000A0000000031010010203040506070809124203						
Response Body: 06 <NumberL><NumberH> <AID Block 1> <AID Block 2> … <AID Block N> **Where:** <NumberL><NumberH> is the number of AID Blocks. <AID Block> format is <LenL> <LenH> <Several bytes AID> **Where:** <LenL> <LenH> is the length of AID **Note:** If AID List does not exist, response 15 F2 00						

8.10 Retrieve CA Public Key

This command retrieves the Certificate Authority public key.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	04	00	Command Body: 72 46 04 01…	36	BC	03
Command Body: 72 46 04 01<5 bytes RID> <1 byte Index>						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	8E	00	See below			03
Response Body: 06 <5 bytes RID> <1 byte Index> <1 byte Hash Algorithm> <1 byte Encryption Algorithm> <20 bytes HashValue> <4 bytes Public Key Exponent> <2 bytes Modulus Length> <Variable bytes Modulus> **Where:** • <Hash Algorithm> is the only algorithm supported is SHA-1. The value is set to 0x01						

- <Encryption Algorithm>: the encryption algorithm in which this key is used. Currently supports only RSA. The value is set to 0x01.
- <HashValue>: calculated using SHA-1 over the following fields: RID, Index, Modulus, and Exponent.
- <Public Key Exponent>: the actual length of the exponent is either one byte or 3 bytes. It can have two values: 3 (Format is 0x00 00 00 03) or 65537 (Format is 0x00 01 00 01).
- <Modulus Length>: <LenL> <LenH> Indicates the length of the next field.
- <Modulus>: the modulus field of the public key. Its length is specified in the field above.

Note:
If the CA Key RID does not exist, the response is 15 F2 05
If the CA Key Index does not exist, the response is 15 F2 06

8.11 Remove CA Public Key

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	72 46 04 02…			03
Command Body: 72 46 04 02 <5 bytes RID> <1 byte Index>						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00				03
Response Body: 06 = ACK **Note:** If the CA Key RID does not exist, the response is 15 F2 05 If the CA Key Index does not exist, the response is 15 F2 06						

8.12 Set CA Public Key

This command erases/sets/installs an EMV L2 CA Public Key into the reader.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	72 46 04 03…			03

Command Body:

72 46 04 03 <5 bytes RID> <1 byte Index> <1 byte Hash Algorithm> <1 byte Encryption Algorithm> <20 bytes HashValue> <4 bytes Public Key Exponent> <2 bytes Modulus Length> <Variable bytes Modulus> <2 bytes MAC Length> <MAC Data>

Where:

- <Hash Algorithm>: The only algorithm supported is SHA-1. The value is set to 0x01.
- <Encryption Algorithm>: The encryption algorithm in which this key is used. Currently supports only one type: RSA. The value is set to 0x01.
- <HashValue>: calculated using SHA-1 over the following fields RID, Index, Modulus, and Exponent.
- <Public Key Exponent>: the actual length of the exponent is either one byte or 3 bytes. It can have two values: 3 (Format is 0x00 00 00 03) or 65537 (Format is 0x00 01 00 01),
- <Modulus Length>: <LenL> < LenH> Indicates the length of the next field.
- <Modulus>: the modulus field of the public key. Its length is specified in the field above.
- <MAC Length> is 2 byte data – the length of <MAC Data>:
    - Is fixed to 0x00 0x00 for non-PCI devices.
    - Is fixed to is 0x1E 0x00 for PCI devices.
- <MAC Data>:
    - Does not exist for non-PCI devices.
    - For PCI devices, it is Fix 30 bytes data:
        - <MAC Value Length> is 2 byte and fixed to 0x10 0x00
        - <MAC Value> is 16 bytes. MAC value is MAC-HOST. The msgX is "72 46 04 03 <5 bytes RID> <1 byte Index> <1 byte Hash Algorithm> <1 byte Encryption Algorithm> <20 bytes HashValue> <4 bytes Public Key Exponent> <2 bytes Modulus Length> <Variable bytes Modulus> <2 bytes MAC Length>"
        - <MAC Key KSN Length> is 2 byte and fixed to 0x0A 0x00
        - <MAC Key KSN> is 10 bytes – MAC DUKPT Key KSN

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00				03

Response Body: 06 = ACK

Note:

- Per <RID> has 6 CA Public Key.
- The maximum number of CA public keys the reader can store is 16.
- If all key slots are used (full), then response 15 F2 07
- If CA Key Hash Data has error, then response 15 F2 08
- If <Hash Algorithm> and <Encryption Algorithm> are not 0x01, then response is 15 F2 0E.

8.13 Remove All CA Public Key List

This command erases all EMV L2 CA Public Key List from the reader.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	72 46 04 04			03

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00				03
Response Body: 06 = ACK						

8.14 Retrieve All CA Public Key List

This command retrieves all EMV L2 CA Public Key List from the reader.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	04	00	72 46 04 05	35	C1	03
Output Hex String: 0204007246040535C103						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	A5	00	See Below	9E	BC	03

Return Hex String:

02A50006A200A000009999E1A000009999E2A000009999E3A000009999E4A000009999E5A000009999E6A000000004FAA000000004FBA000000004FCA000000004FDA000000004FEA000000004FFA00000000650 2A00000006503A00000000350A00000000351A00000000353A00000000396A00000000357A00000000358A00000000354A00000002560A00000002561A000000152D0A000000152D1A00000000333C0A000000333C19EBC03

Response Body:

06 <LenL> <LenH> <5Bytes RID1> <1 byte RID1 Index><5Bytes RID2> <1 byte RID2 Index>…..
<5Bytes RIDN> <1 byte RIDN Index>.

Note: If any CA Key does not exist, response 15 F2 04

8.15 Start Transaction

This command is used to start / initiate an EMV L2 transaction.

The command will:

- Initiate a Power On sequence with the ICC reader.
- Establish parameters for support of "Fallback to MSR" operation.
- Establish timeouts parameters for "card insertion", and receipt of "Authenticate Transaction" command.
- Transmit "Application Data" parameters to the card.

In response, the reader will return a minimum of two response messages.
Optionally, additional "call back" messages may be transmitted in-between the first and second response messages (see note below).

Upon completion of transmitting the final response message, the reader will enter a timeout mode and await receipt of the "Authenticate Transaction" command from the host.

Once the Start EMV L2 Transaction sequence has been initiated, the only commands the reader will respond to are: "Authenticate EMV L2 Transaction," "Cancel EMV L2 Transaction," or "Retrieve EMV L2 Transaction Data." Because of this, if the host desires to end the session, it must transmit a "Cancel EMV L2 Transaction" command.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	1E	00	72 46 05 01…	AC	E8	03

Output Hex String:
021E007246050101001E001E9F02060000000001009F03060000000000009C0100ACE803

Command Body:
72 46 05 01 <FallBack><TimeOut1> <TimeOut2> <App Data>

Where:

- <FallBack> (1byte).
  - 0x01 indicates support for FallBack to MSR.
  - 0x00 indicates no support for FallBack to MSR.
- <TimeOut1> (2 bytes, units in Seconds). (Low byte, High byte)
  - Timeout for Card to be seated (inserted) into ICC slot (e.g., 00 1E = 0x1E00 = 30 decimal = 30 seconds).
- <TimeOut2> (2 bytes, units in Seconds).
  - Time out for "**Authenticate Transaction**" command to be received from Host (e.g., 00 1E = 0x1E00 = 30 decimal = 30 seconds).

- <App Data> format is <TLV1> <TLV2> … <TLVn> (refer to Transaction Data & Option Data List).

Additional Notes:

1. If there is an error in the process, the reader will return an error code and terminate transaction.
2. If there is a command format error, the reader response is: 15 F2 09.
3. If there is no application data in the reader, the response is: 15 F2 00.
4. If there is no terminal data in the reader, the response is: 15 F2 01.
5. After a transaction starts, the terminal only can receive: "Authenticate Transaction," "Cancel Transaction," and "Retrieve Transaction Data" commands. If any other command is sent, the response is: 15 F2 0A.
6. If a timeout occurs, the response is: 15 81 00.
7. If a tag is not present in the ICC, the tag will not be present in the output TLV list.
8. Amount, other amount, and transaction type must exist. If not, the terminal responds with error code 15 F2 0D.
9. For MSR fallback operations, the <Response Code> is 00 07, <Attribution> is none.
10. For MSRs with a service code that is not 2 or 6, the <Response Code> is 00 11.

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00				03

Return Hex String 1: 02010006060603 (if needed, insert "call back" messages here; see notes below)

Return Hex String 2:

026F00060010005712517805936510172 5D20072011530357090 1F5A0851780593651017255F3401015F20104D45594552532F4C415752454E4345205F24032007319F20005F25031510015F2D02656E500A4D415354455243415244 4F07A0000000041010 8407A0000000041010 DFEE230008C003

Response Body:

This command results in a minimum of two responses from the reader.

Note: between the first and second mandatory responses from the reader (depending on terminal settings and card type), there may be an additional series of "call back" messages transmitted between the reader and host. These are typically "display messages" and negotiations for App Selection or Language parameters with a card. Such examples are not provided in this section.

The first response is a 06 (ACK) to acknowledge that the "Start Transaction" command was successfully received.

The second response is:

- 06 <Response Code> <Attribution> <Output Data List> (All response except that MSR Fall Back). Or,
- 06 <Response Code> <Output Data List> (MSR Fall Back response)

Where:

- <Response Code> length is 2 bytes. Please refer to "Response Code" Section
- <Attribution>: 1 Byte:
  - BIT0 – Card Type: 0 – Contact Card
  - BIT2,1 – Encryption Mode: 00 – TDES Mode, 01 – AES Mode
  - BIT7~3 - Reserve
- <Output Data List> format is <TLV1> <TLV2> … <TLVn>.

8.16 Authenticate Transaction

Prior to executing this command, a "Start EMV L2 Transaction" command must be completed and the reader must still be in the "timeout" wait mode, awaiting the "Authenticate EMV L2 Transaction" command.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	07	00	72 46 05 02 01 00 1E	2C	DE	03

Output Hex String: 02070072460502010 01E2CDE03

Command Body: 72 46 05 02<ForceOnline><TimeOut> <App Data>

Where:

- <ForceOnline> (1byte).
  - 0x01 indicates it supports Force Online mode.
  - 0x00 indicates it does not support Force Online mode.
- <TimeOut> (2 byte, unit is Seconds).
  - Terminal wait time for the host response when online.
- <App Data> format is <TLV> (V is output tag list).

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	A5	00	See Below	9E	BC	03

Return Hex String:

02A50006A200A000009999E1A000009999E2A000009999E3A000009999E4A000009999E5A000009999E6A000000004FAA000000004FBA000000004FCA000000004FDA000000004FEA000000004FFA00000000650 2A00000006503A00000000350A00000000351A00000000353A00000000396A00000000357A00000000358A00000000354A00000002560A00000002561A000000152D0A000000152D1A000000333C0A000000333C19EBC03

Response Body:
06 <LenL> <LenH> <5Bytes RID1> <1 byte RID1 Index><5Bytes RID2> <1 byte RID2 Index>…..
<5Bytes RIDN> <1 byte RIDN Index>.

Note:

- If any CA Key does not exist, the response is: 15 F2 04.
- If a tag is not supplied, it will not appear in the output.
- If the tags 9F10 and 9F26 are in the terminal they also occur in the command response TLV list.

8.17 Complete Transaction

Prior to executing this command, an "Authenticate EMV L2 Transaction" command needs to be completed, and the reader must still be in the "timeout" wait mode, awaiting the "Complete EMV L2 Transaction" command.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	15	00	72 46 05 03…	A8	0C	03

Output Hex String: 02150072460503018A023030910A1122334455667788303 0A80C03

Command Body:
72 46 05 03 <1Byte ComFlag> [<Authorization Response Code (TLV,Tag 8A)>< Issuer Authentication Data (TLV, Tag 91)>< <Scripts (TLV, Tag 71/72)>] <App Data>

Where:

- <1Byte ComFlag>: 0x01 indicates online with host, 0x00 indicate unable to go online.
- The data in [ ] brackets indicates that the data is optional:
  - If the ComFlag is 0x01, the data exists.
  - If the ComFlag is 0x00, the data does not exist.
- <App Data> format is <TLV> (V is output tag list).

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00	See Below			03

Return Hex String 1: 02010006060603

Return Hex String 2:
026500060203009F10120110201001220400000000000000000000FF9F2608853911F73E359C1B9F2701009F360200399F3704AC8635539F02060000000001009F4D009F4F009F130095050400000800 9B02E8009F03060000000000009F34031E030099009F5B0044EC03

Response Body:

First response: 06

Second response: 06 <Response Code> <Attribution> <Output TLV Data>.

Where:

- <Response Code>length is 2bytes.
  Please refer to "Response Code" Section
- <Attribution>: 1 Byte:
  - BIT0 – Card Type: 0 – Contact Card
  - BIT2,1 – Encryption Mode: 00 – TDES Mode, 01 – AES Mode
  - BIT7~3 - Reserve

Note:
If any parameter was error, response: 15 F2 02.

8.18 Cancel Transaction

This command ends/ cancels an EMV L2 transaction.
It may be sent any time after a "Start EMV L2 Transacation" has been previously sent.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	03	00	72 46 06	32	BE	03
Output Hex String: 02030072460632BE03						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	01	00	02010006060603	06	06	03
Return Hex String: 02010006060603						
Response Body Interpretation: 06 (ACK)						

8.19 Retrieve Transaction Result

This command retrieves requested TLV data of a completed EMV L2 transaction.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	72 46 07 01…			03

Command Body: 72 46 07 01<2 Byte Length><Tags>.

Where:

- <2 Byte Length>format is <LenL><LenH>, is the length of <Tags>. See example below.
- <Tags> these tags are returned as TLV format.

EXAMPLE:
020a00724607010400**959b9f34**93c703
(Requested tags are 95, 9B, and 9F34; 2-byte length is 04 00.)

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00	02010006060603			03

Response Body: 06 <TLV1> <TLV2> … <TLVn>.
For details please refer to EMV4.3 Book 3.

Where:

- <1Byte Message type>, where value 0x00 means transaction message, value 0x01 means advice message.

EXAMPLE: 0212000**695**0504000000000**9b**02e800**9f34**031e0300562003

(The tags requested were 95, 9B, and 9F34.)

8.20 Get EMV Level Two Version Number

This command retrieves the version number of the EMV L2 kernel.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	04	00	72 46 08 01	13	DC	03
Output Hex String: 020400724608013DC103						
Command Body: 72 46 08 01						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	14	00	06454D5620436F6D6D6F6E204C322056322E3030	61	4B	03
Return Hex String: 02140006454D5620436F6D6D6F6E204C322056322E3030614B03						
Response Body: 06<" EMVL2 X.YY" >. X.YY is the version number. The first version number is 1.00. **Response Body Interpretation:** ASCII String: EMV Common L2 V2.00						

8.21 Retrieve Interface Device Serial Number

This command retrieves the Interface Device Serial Number.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	04	00	72 46 86 01	B3	3F	03
Output Hex String: 02040072468601B33F03						
Command Body: 72468601						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	09	00	065465726D696E616C	22	42	03
Return Hex String: 020900065465726D696E616C224203						
Response Body: 06 <Serial Number>						

8.22 Set Interface Device Serial Number

This command sets the Interface Device Serial Number.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	0C	00	72 46 86 03 <Serial Number>	B1	DD	03
Output Hex String: 020C00724686033031323334353637B1DD03						
Command Body: 724686033031323334353637						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00				03
Response Body: 06 = ACK						

8.23 Retrieve Terminal Identification

This command retrieves the Terminal Identification string.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	04	00	72 46 87 01	B2	40	03
Output Hex String: 02040072468701B24003						
Command Body: 72468701						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	09	00	063837363534333231	0E	AA	03
Return Hex String: 0209000638373635343332310EAA03						
Response Body: 06 <Identification> **Response Body Interpretation:** ASCII String: '87654321'						

8.24 Set Terminal Identification

This command sets the Terminal Identification string.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	0C	00	72 46 87 03 <Identification>	B0	DE	03
Output Hex String: 020C00724687033031323334353637B0DE03						
Command Body: 72468703303132333435363 7 **Example:** 3031323334353637 = ASCII '01234567'						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	01	00	06	06	06	03
Return Hex String: 02010006060603						
Response Body: 06 (ACK)						

8.25 Retrieve Certification Revocation List

This command retrieves the CRL (Certificate Revocation List).

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	04	00	72 46 85 01	B0	3E	03
Output Hex String: 02040072468501B03E03						
Command Body:72468501						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	23	01	See below	84	50	03
Return Hex String: 022301062001A00000000350014455A000000004FE092355A000009999E1014394A000000003500 00001A00000000350000002A00000000350000003A00000000350000004A00000000350000005A0 0000000350000006A00000000350000007A00000000350000008A00000000350000009A00000000 350000010A00000000350000011A00000000350000012A00000000350000013A000000003500000 14A00000000350000015A00000000350000016A00000000350000017A00000000350000018A0000 0000350000019A00000000350000020A00000000350000021A00000000350000022A00000000350 000023A00000000350000024A00000000350000025A00000000350000026A00000000350000027A 00000000350000028A00000000350000029845003						
Response Body: 06 <2Byte Length> <CRL1><CRL2>…<CRLn>						

Where:

- <2Byte Length> = <Low byte of length><High byte of length > (Length of following CRLs)
- <CRL>format is:
  - <5Bytes RID><1Byte CA public key Index><3Bytes Certificate Serial Number><3Bytes Date>
  - <3Bytes Date> is the date in YYMMDD format that the certificate was added to the revocation list.

Note: If no CRL exists, the response is 15 F2 0B.

8.26 Remove Certification Revocation List

This command removes the specified CRL(s) from the CRL (Certificate Revocation List).

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	72 46 85 02			03

Command Body: 72 46 85 02 <2 Bytes Length> <CRL1><CRL2>…<CRLn>

Where:

- <2Byte Length> = <Low byte of length><High byte of length >
- <CRL>format is:
  - <5Bytes RID><1Byte CA public key Index><3Bytes Certificate Serial Number>

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00	See below			03

Return Hex String: 020100**06**060603

Response Body: 06 (ACK)

8.27 Set Certification Revocation List

This command adds CRL(s) to the CRL (Certificate Revocation List).

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	04	00	72 46 85 03	B0	3E	03

Command Body: 72 46 85 03 <2 Bytes CRL Length> <CRL1> <CRL2>…<CRLn>.

Where:

- <2Byte CRL Length> = <Low byte of length><High byte of length >
- <CRL>format is:
  - <5Bytes RID><1Byte CA public key Index><3Bytes Certificate Serial Number>
- <MAC Length> is: 2 byte data – the length of <MAC Data>
  - Is fixed to 0x00 0x00 for non-PCI devices
  - Is fixed to 0x1E 0x00 for PCI devices
- <MAC Data>:
  - Does not exist for non-PCI devices.
  - For PCI devices, MAC is Fixed 30 bytes data:
    - <MAC Value Length> is 2 byte and fixed to 0x10 0x00.
    - <MAC Value> is 16 bytes – Please refer to "Verification Algorithm" section. MAC value is MAC-HOST. The msgX is "72 46 85 03 <2 Bytes CRL Length> <CRL1> <CRL2>…<CRLn> <2 bytes MAC Length>".
    - <MAC Key KSN Length> is 2 byte and is fixed to 0x0A 0x00.
    - <MAC Key KSN> is 10 bytes – MAC DUKPT Key KSN.

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	23	01	See below	84	50	03

Response Body: 06 (ACK)

Note: The maximum number of CRLs is 30. If maximum is exceeded, the response is: 15 F2 0C.

8.28 Remove All Certification Revocation List

This command removes/erases the CRL (Certificate Revocation List).

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	72 46 85 04			03

Command Body: 72 46 85 04

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00				03
Return Hex String: 020100**06**060603						
Response Body: 06 (ACK)						

8.29 Get EMV L2 Kernel Check Value

This command retrieves the EMV L2 Kernel checksum value.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	04	00	72 46 09 01	3C	C2	03
Output Hex String: 020400724609013CC203						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	15	00	068D9395C588F2DEC9AE9DC8BB7B9841A3525711C6	6E	60	03
Return Hex String: 021500068D9395C588F2DEC9AE9DC8BB7B9841A3525711C6B6E603						
Response Body Interpretation: L2 Kernel Checksum = 8D9395C588F2DEC9AE9DC8BB7B9841A3525711C6						

8.30 Get ICC L2 Kernel Check Value

This command sets the L2 Kernel Check Value.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	72 46 09 02			03

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00				03
Response Body: 06 <20 bytes L2 Configuration Check Value>						

8.31 Remove Transaction Amount Log

This command removes the transaction amount log.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	72 46 0A 02			03

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00				03
Response Body: 06 (ACK)						

8.32 ICC Bezel ON

This command illuminates the ICC bezel.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	04	00	72 46 20 01	15	D9	03
Output Hex String: 0204007246200115D903						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	01	00	06	06	06	03
Return Hex String: 02010006060603						
Response Body Interpretation: 06 = ACK						

8.33 ICC Bezel BLINK

This command illuminates and blinks the ICC bezel at a determined on/off duty cycle.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	06	00	72 46 20 02 01 F4	E3	CF	03
Output Hex String: 0206007246200201F4E3CF03 **Where:** 0x01F4 = 500 decimal = 500 ms on/off duty cycle (blink rate)						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	01	00	06	06	06	03
Return Hex String: 02010006060603						

Response Body Interpretation: 06 = ACK

8.34 ICC Bezel - OFF

This command turns off the ICC bezel.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	06	00	72 46 20 00 00 00	14	D8	03
Output Hex String: 02060072462000000014D803						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	01	00	06	06	06	03
Return Hex String: 02010006060603						
Response Body Interpretation: 06 = ACK						

9.0 Smart Card Command Group – Input/Output Commands

9.1 Output Body Format

49 72 <DriverID><DriverData>

9.2 Input Body Format

72 49 <DriverID><ACK> or
72 49 <DriverID><ACK><ACK Data> or
72 49 <DriverID><NAK><Error Codes>

9.3 LCD display control

Augusta does not have an LCD display (obviously) but must include LCD display info per EMV requirements. After a transaction has begun (via Start Transaction), the reader responds with various messages to the host application. These messages may include LCD-display control information.

Note that the output body, in such cases, contains 49 72 01.

Output Body is:
49 72 01 <Len_L of Control Data> <Len_H of Control Data> <m bytes Control Data>

Note:
When mode is 01, 02, or 08, the transaction pauses and awaits a selection based on what the EMV kernel is telling you to display. The kernel is awaiting the LINE NUMBER of the selection.

When mode is 0x10, clear screen.
When mode is 0x03, display message only.

When there is a display message, there will be one or more lines to display with field separator 0x1C. The display data starts with line number. If the most significant bit is set (0x81 vs 0x01), then the message ID is provided; otherwise, it is an ASCII message.

Where: <m bytes Control Data> can be in one of the following configurations:

- Display mode: 1 byte
  - 1: Menu Display
  - 2: Normal Display get function key
  - 3: Display without key input (Do Not Receive Input Data)
  - 8: Language Menu Display
  - 16 (0x10): Clear Screen (Do Not Receive Input Data)

If the mode byte is "Clear Screen," it is unnecessary to send the field below.

- If the mode is Normal Display or Menu Display, the Length of Total timeout for keypad entry: 2 bytes (Little-endian).
- If the mode is Display without key input, the value is 00 00.
- If the mode is Normal Display or Menu Display, the value is for Total timeout for keypad entry, in seconds, (Little-endian); default is 30 seconds.

Note: Timeout cancels keypad entry and returns an error.

• Length of Display Message Language: 2 bytes (Little-endian)
• Display Message Language – 2 bytes ASCII:
  - EN (0x45 0x4E)L English (default)
  - ES: Spanish
  - ZH: Chinese
  - FR: French
  - …

• Length Display Message Control: 2 bytes (Little-endian)
• Display Message Control: repeatable combination of <Line> <Message> <0x1C>
<Line>: Display line number (1-First Line, n-nth Line), Maximum 16 lines.
  - The lower 7 bits is for the line number.
  - The MSB is to indicate that the following message is a Message String or Message ID.
  - MSB – 0: Message String (valid for "Menu Display" and "Language Menu Display").
  - MSB – 1: Message ID (valid for "Menu Display").

<Message>: Message String or Message ID.

Message String:
- "Menu Display": character in the range of 0x20 – 0x7f, maximum 16 characters
- "Language Menu Display": 2 bytes Language ID
  - EN – English (default)
  - ES – Spanish
  - ZH – Chinese
  - FR – French
  - …
  - …

Message ID: 1 byte, check LCD Foreign Language Mapping Table
<0x1C>: separator

• Length Back Light On TimerValue: 2 bytes (Little-endian)
• Back Light On TimerValue in seconds: (Little-endian) (all 0-Back Light Off, all 0xff-Back Light always On)

Note:
Length is always 02 00
Value always 00 00 (Back Light Off)

ACK messages have the following format:

Input Body is 72 49 01 <ACK> <Len_L of ACK Data> <Len_H of ACK Data> <n bytes ACK Data>

Where:

<n bytes ACK Data>

• Display mode– 1 byte 01: Cancel (user presses cancel key on the key pad for mode 1) 1: Menu Display 2: Normal Display get function key 8: Language Menu Display If the mode byte is “Cancel” or “Display without key input,” it is unnecessary to send the field below. • If the mode is Menu Display, the Length of Menu value is: (If Normal Display, Length of Key (Get Function)) • If the mode is Menu Display, the Menu value, sequence number of selected line, and hex format are: (If Normal Display, ASCII format (‘E’ is Enter, ‘C’ is Cancel))

10.0 Smart Card Group – Set/Get Commands

10.1 Set Quick Chip Mode

This command sets the Quick Chip mode (On or Off).

Note: The following command should be issued in USB HID mode. It tells the firmware to enable a hard-coded event sequence (corresponding to the sequence for Quick Chip and M/Chip Fast) in which an ARC of 'Z3' (meaning "Could not go online") is automatically sent to the card for the second Gen AC request, resulting in an automatic AAC.

This command must be issued in HID mode. After Quick Chip Mode is enabled, you would normally put the device into KB (keyboard) mode; to switch the device into keyboard mode, run the 78 53 01 10 02 02 00 command.

See Appendix J for more information about using Quick Chip Mode in Augusta and/or Augusta S.

For detailed information about how Quick Chip and M/Chip Fast transactions are implemented, be sure to consult the U.S. Payments Forum white paper.

Also, be sure to consult the ID TECH technical white paper on this subject.

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	72 53 01 29 01 <ON or OFF: 31 or 30>			03
Command Body: 72 53 01 29 01 31 <Quick Chip Mode ON> 72 53 01 29 01 30 <Quick Chip Mode OFF>						

See Appendix J for more information.

10.2 Get Quick Chip Mode

This command requests the currently-set Quick Chip mode.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	72 52 01 29			03
Command Body: 72 52 01 29						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00	06 72 01 29 01 <Option>			03

Response Body: 06 72 01 29 01 <Option>

Where: <Option> is 31 <Quick Chip Mode ON> or 30 <Quick Chip Mode OFF>

10.3 Set Card Type Option

This command sets the card type option.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	72 53 01 04 01			03

Command Body: 72 53 01 04 01 <Option>

Where:

Card Type	Option
EMV	**0xFF**
ISO	0x00

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00				03

Response Body: 06 (ACK)

10.4 Get Card Type Option

This command requests the currently-set card type option.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	72 52 01 04			03

Command Body: 72 52 01 04

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00				03

Response Body: 06 72 01 04 01 <Option>

Where:

Card Type	Option
EMV	**0xFF**
ISO	0x00

10.5 Set ICC L1 Transaction Timeout

This command sets the ICC L1 transaction timeout.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	06	00	72 53 01 05 01	2C	D4	03
Output Hex String: 020600725301050108 2CD403						
Command Body: 72 53 01 05 01 <Timeout> **Where:** <Timeout> is 1 byte. Value is **8 seconds** ~ 90 Seconds						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	01	00	06	06	06	03
Return Hex String: 02010006060603						
Response Body: 06 (ACK)						

10.6 Get ICC L1 Transaction Timeout

This command requests the currently-set ICC L1 transaction timeout.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	04	00	72 52 01 05	24	CA	03
Output Hex String: 0204007252010524CA03						
Command Body: 72 53 01 05 01 <Timeout> **Where:** <Timeout> is 1 byte. Value is **8 seconds** ~ 90 Seconds						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	06	00	7201050108	79	87	03
Return Hex String: 020600067201050108798703						
Response Body: 06 72 01 05 01 <Timeout> Where Timeout is value 08 – 90 seconds						

10.7 Set Pre/Post PAN Data Len

This command sets the number of PAN digits that will be "masked."
Pre = First 4 – 6 digits of PAN.
Post = Last 0 – 4 digits of PAN.
The default mask character is "asterisk" (*).

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	07	00	72 53 01 20 02…	00	F2	03
Output Hex String: 02070072530120020604 00F203						
Command Body: 72 53 01 20 02 <PrePANCtlDataLen> <PostPANCtlDataLen> **Where:** • <PrePANCtlDataLen> need be 0~6, Default is 4 • <PostPANCtlDataLen> need be 0~4, Default is 4						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	01	00	06	06	06	03
Return Hex String: 02010006060603						
Response Body: 06 (ACK)						

10.8 Get Pre/Post Data Len

This command requests the currently-set pre/post PAN mask settings.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	04	00	72 52 01 20	01	E5	03
Output Hex String: 0204007252012001E503						
Command Body: 72 52 01 20						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	07	00	06720120020604	55	A5	03
Return Hex String: 02070006720120020604 55A503						
Response Body: 06 72 01 20 02 <PrePANCtlDataLen> <PostPANCtlDataLen>						

10.9 Set ASCII Mask Data

This command sets the ASCII "mask" character that will mask PAN digits.
The default mask character is "asterisk" (*).

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	06	00	72 53 01 21 01 …	2A	12	03
Output Hex String: 02060072530121012A2A1203						
Command Body: 72 53 01 21 01 <AsciiMaskData> **Where:** <AsciiMaskData> can be 0x20~0x7E Default is 0x2A (*)						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	01	00	06	06	06	03
Return Hex String: 02010006060603						
Response Body: 06 (ACK)						

10.10 Get ASCII Mask Data

This command requests the currenty-set mask character.
The default mask character is "asterisk" (*), 0x2A.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	04	00	72 52 01 21			03
Output Hex String: 0204007252012100E603						
Command Body: 72 52 01 21						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	07	00	06720121012A	7F	C5	03
Return Hex String: 02060006720121012A7FC503						
Response Body: 06 72 01 21 01 <ASCIIMaskCharacter>						

10.11 Set BCD Mask Data

This command sets the BCD "mask" character that will mask PAN digits.
The default mask character is "asterisk" (*).

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	06	00	72 53 01 22 01 0C	0F	F5	03
Output Hex String: 020600725301220 10C0FF503						
Command Body: 72 53 01 22 01 <BCDMaskCharacter> **Where:** <BCDMaskCharacter> can be 0x0A~0x0F Default is 0x0C (*) **Note:** • If 0x23 is masked High BCD, result should be 0xC3 • If 0x23 is masked Low BCD, result should be 0x2C • If 0x23 is masked All BCD, result should be 0xCC						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	01	00	06	06	06	03
Return Hex String: 02010006060603						
Response Body: 06 (ACK)						

10.12 Get BCD Mask Data

This command requests the currently set mask character.
The default BCD mask character is "asterisk" (*), 0x0C.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	04	00	72 52 01 22	03	E7	03
Output Hex String: 0204007252012203E703						
Command Body: 72 52 01 22						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	06	00	06720122010C	5A	A8	03
Return Hex String: 02060006720122010C5AA803						
Response Body: 06 72 01 22 01 <BCDMaskCharacter>						

10.13 Restore Default ICC Group Settings

This command restores the ICC Group Settings to their default values (as shown below).

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	03	00	725300	21	C5	03
Output Hex String: 02030072530021C503						
Command Body: 72 53 00						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	06	00	06	06	06	03
Return Hex String: 02010006060603						

Response Body: 06 (ACK)

Default Values:

Function Name	Default Value
Card Type	EMV
ICC L1 Transaction Timeout	8 (seconds)
PrePANCtlDataLen	4 (characters)
PostPANCtlDataLen	4 (characters)
AsciiMaskData	0x2A (*)
BCDMaskData	0x0C (*)

10.14 Set Quick Chip & M/Chip Fast Mode USB-KB Output Data Postfix

This command appends arbitrary character data (up to 8 characters max) onto the end of the EMV data stream in Quick Chip & M/Chip Fast mode (keyboard mode).

Command Body: 72 53 01 2A <Len> [<Postfix Data>]

Where:

- <Len> is 1 byte, is the length of <Postfix Data>. If <Len> is 0, no <Postfix Data>. **Default is 0**. Size is 0~8.
- <Postfix Data> is 1~8 bytes data.

Response Body: 06

10.15 Get Quick Chip & M/Chip Fast Mode USB-KB Output Data Postfix

This command retrieves the output data postfix in Quick Chip & M/Chip Fast mode (keyboard mode).

Command Body: 72 52 01 2A

Response Body: 06 72 01 2A <Len> [<Postfix Data>]

10.16 Set Quick Chip & M/Chip Fast Mode USB-KB Output Data Prefix

This command inserts arbitrary character data (up to 8 characters max) into the beginning of the EMV data stream in Quick Chip & M/Chip Fast mode (keyboard mode).

Command Body: 72 53 01 2B <Len> [<Prefix Data>]

Where: <Len> is 1 byte, the length of <Prefix Data>. If <Len> is 0, no <Prefix Data> will occur. Default is 0. Size should be 0~8.

<Prefix Data> is 0~8 bytes data.

Example of specifying a carriage return (0x0D) in the prefix:

OUT: 0206007253012b01**0d**07ff03
IN: 02010006060603 (normal response)

10.17 Review ICC Group All Setting

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	03	00	72 52 00	20	C4	03
Output Hex String: 02030072520020C403						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	13	00	0672050401FF05010820020404210 12A22010C	80	34	03
Return Hex String: 0213000672050401FF05010820020404210 12A22010C803403						
Response Body: 06 72 05 04 01 <Card Type Option> 05 01 <L1 Transaction Timeout> 20 02 <PrePANCtlDataLen> <PostPANCtlDataLen> 21 01 <AsciiMaskData> 22 01 <BCDMaskData> **Response Body Interpretation:** Where: 6 highlighted bytes represent: <Card Type Option> = FF (0xFF = EMV, 0x00 = ISO) <L1 Transaction Timeout>= 08 (8 second timeout for EMV L1 Transaction) <PrePANCtlDataLen> = 04 (Mask first 4 digits of PAN) <PostPANCtlDataLen> = 04 (Mask last 4 digits of PAN)						

<AsciiMaskData> = 2A (ASCII Mask Character = *)
<BCDMaskData> = 0C ( )

Note: The byte following Task ID (72) is Block Number Data.

10.18 Get CTL2 Transaction Mode

Command Body: 72 52 01 29

Response Body: 06 72 01 29 01 <Option>

10.19 Set QuickChip Mode USB-KB Output Data Postfix

Command Body: 72 53 01 2A<Len>[<Postfix Data>]

Where:

- <Len> is 1 byte, is the length of <Postfix Data>. If <Len> is 0, no <Postfix Data>. **Default is 0**. Size is 0~8.
- <Postfix Data> is 1~8 bytes data.

Response Body: 06

10.20 Get QuickChip Mode USB-KB Output Data Postfix

Command Body: 72 52 01 2A

Response Body: 06 72 012A<Len>[<Postfix Data>]

10.21 Set QuickChip Mode USB-KB Output Data Prefix

Command Body: 72 53 01 2B <Len>[<Prefix Data>]

Where:

- <Len> is 1 byte, is the length of <Prefix Data>. If <Len> is 0, no <Prefix Data>. **Default is 0**. Size is 0~8.
- <Prefix Data> : 1~8 bytes data.

Response Body: 06

10.22 Get QuickChip Mode USB-KB Output Data Prefix

Command Body: 72 52 01 2B

Response Body: 06 72 012B<Len>[<Prefix Data>]

10.23 Set ICC Reading Characteristics

This command sets the reader's ICC reading characteristics.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	72 53 01 11 01 <ICC Reading Characteristics>			03

Command Body: 72 53 01 11 01 <ICC Reading Characteristics>

Where:

ICC Reading Characteristics	Option
ICC Reading Function Off	0x30
ICC Reading Function Enable & Notification Off	**0x31**
ICC Reading Function Enable & Notification On	0x32

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00	06			03

Response Body: 06 = ACK

Note: If ICC Reading Function Off, the response will be 15 6D 10 for below ICC Group command:

- Power On (Get ATR)
- Exchange APDU Plaintext (If it is Valid)
- Exchange APDU Encryption for special case (If it is Valid)
- Exchange APDU Encryption
- Start Transaction
- Authenticate Transaction
- Complete Transaction

10.24 Get ICC Reading Characteristics

This command retrieves the reader's current ICC reading characteristics.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	72 52 01 11			03

Command Body: 72 52 01 11

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00	06 72 01 11 01 <ICC Reading Characteristics>			03

Response Body: 06 72 01 11 01 <ICC Reading Characteristics>

10.25 Set EMV CT L2 Transaction Interval

This command sets the reader's EMV CT L2 transaction interval.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	72 53 01 2F 01 <CTL2Interval>			03

Command Body: 72 53 01 2F 01 <CTL2Interval>

Where:

CTL2Interval Value	Definition
0	**No Delay**
30 ~ 100	300ms ~ 1000ms

Note: Add interval, Software in some system can process ACK data & Output Message data in time.

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00	06			03

Response Body: 06 = ACK

10.26 Get EMV CT L2 Transaction Interval

This command retrieves the reader's current EMV CT L2 transaction interval.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	72 52 01 2F			03

Command Body: 72 52 01 2F

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00	06 72 01 2F 01 <CTL2Interval>			03

Response Body: 06 72 01 2F 01 <CTL2Interval>

11.0 MSR Commands Group

11.1 Set to Default Setting

This command will reset all configurable reader settings to their default values, with the exception of security settings.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	73 53 00			03
Command Body: 73 53 00						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00	02010006060603			03
Response Body: 06 (ACK) Or, 15 + <Error Code>						

11.2 Arm MSR to Read

This command enables ("arms") the Magnetic Stripe Reader (MSR) to receive a card swipe. Use this command with MSR function ID 0x1A when its value is set to buffer mode (0x32).

This command must be preceded by a "Get/Set One Byte Function ID" command sequence that enables "MSR Buffer Mode" (See example in Get/Set Function ID – Single Byte).

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	05	00	73 46 50 01 <Arm Setting>	54	3A	03
Output Hex String: 02050073465001305 43A03						

Command Body:
73 46 50 01 <Arm Setting >

Where:

- < Arm Setting > is defined as:
  - 0x30: Arm to read magnetic card data
  - 0x32: Enable read magnetic card data when function ID 0x1A is set to 0x30 or disable magnetic card data when function ID 0x1A is set to 0x32

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	01	00	02010006060603	06	06	03
Return Hex String: 02010006060603						
Response Body: 15 + <ERROR CODE> or 06 = ACK						

11.3 Read MSR Buffer Data

This command requests and reads the MSR Data Buffer after a card swipe.

This command needs to be preceded by a "Get/Set One Byte Function ID" and "Arm to Read" command sequence.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	05	00	73 46 51 01 30	54	3A	03

Command Body:

73 46 51 01 <Track Selection Option>

Where:

- <Track Selection Option> =
  - 0x30 = All Tracks
  - 0x31 = Track 1 only
  - 0x32 = Track 2 only
  - 0x33 = Track 3 only

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	01	00		06	06	03

Return Hex String:

00000044220000006825423531373830353933363531303137323 55e4d45594552532f4c415752454e4345205e31383033313031303030303030303030303135313733333239303030303030303f0d3b353137383035393333635313031373235 3d31383033313031313531373333323 93f0d

Response Body:

15 + <ERROR CODE> or

<Magnetic Card Data>

11.4 Get All Settings

This command returns all configurable settings values from the reader.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	03	00	73 52 00	21	C5	03
Output Hex String: 02030073520021C503						
Command Body: 73 52 00						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	AA	00	See below	68	F6	03

Return Hex String:

02AA00067339230130 4E0B0A3630325433323738333558013 17E013185013188013011013012013
013013014010117010D1901311A01321D013121010D2401303001043400350036003700380039004901044A01044B012A4D013050013055013 05C013760013061012563013B64012565013B66012568013B69013F6A013F6B013F6C01256D013B6E012B6F013172017F73017F7B0130840108860107890134AD0102AE0100AF0100D200D30068F603

Response Body:

15 <ERROR CODE> or

06 <Function Block Number><FuncSETBLOCK1>…<FuncSETBLOCKn>

Each function-setting block <FuncSETBLOCK> has the following format:

<FuncID><Len><FuncData>

Where:

- <Function Block Number> is the total number of function blocks.
- <FuncID> is one byte and identifies the setting(s) for the function.
- <Len> is one byte and shows data length for <FuncData>.
- <FuncData> is the current setting for this function. It has the same format as the sending command for this function.
- <FuncSETBLOCK> are in the order of their Function ID<FuncID>

11.5 Set/Get Function ID for One Byte ID

This command retrieves or sets a Function ID based on a single parameter.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	06	00	73 53 01 <Function ID> 01 <Setting Value>	08	14	03
Output Hex String: 0206007353011A0132081403						
Command Body: 73 53 01 <Function ID> 01 <Setting Value> **Example:** In the above example, this command is being used to Enable the MSR Buffer Mode. • 1A represents Function ID for MSR Read ID • 01 represents single byte Function ID • 32 represents "MSR Buffer Mode – Enabled"						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	01	00	06	06	06	03
Return Hex String: 02010006060603						
Response Body: 15 <ERROR CODE> or 06 (ACK)						

11.6 Set/Get Function ID for Multi-Bytes ID

This command retrieves or sets a Function ID that is based on multi-byte parameters.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	06	00	73 53 01…..	08	14	03
Command Body: 73 53 01 <Function ID> <Data Length> <Array Element Length><Array Element Value>						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02	01	00	06	06	06	03
Return Hex String: 02010006060603						
Response Body: 15 + <ERROR CODE> or 06 (ACK)						

11.7 Remote Key Injection

Most customers do not need to use these commands. Instead, they will take advantage of ID TECH's ability to do key injection as a service.

Note: LRC and Checksum are calculated against the Command Body bytes only. For LRC and Checksum, include all bytes after the length bytes (do *not* include STX, length bytes, nor ETX). For example, calculate the LRC of the Initiate RKL command (below) as

LRC = 0x55 ^ 0x52 ^ 0x4B ^ 0x49 ^ 0x30;

Calculate the Checksum as

Checksum = 255 & (0x55 + 0x52 + 0x4B + 0x49 + 0x30);

11.8 Initiate RKL

This command initiates remote key loading.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	05	00	55 52 4B 49 30	35	6B	03
Command Body: 55 52 4B 49 30 **Where:** 52 4B 49 is 'RKI' (Remote Key Injection)						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00	See below			03
Response Body: <ACK><20 bytes ASCII code KSN of RKI-KEK><10 bytes Serial Number><16 bytes ASCII code Authentication Code 1> or <NAK><Error Code>						

11.9 RKL Get Status

This command gets the current remote key loading status.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	25	00	55 52 4B 49 31 <32 bytes ASCII code Encrypted Data (EK1)>			03

Where:

- 52 4B 49 is 'RKI'
- <32 bytes ASCII code Encrypted Data (EK1)> is 16 bytes Encrypted Data (EK1):
  - The Plaintext of 16 bytes Encrypted Data (EK1) is
- <Authentication Code 2><Authentication Code 1>
- Authentication Code 2 is 8 bytes Random generated by RKS
- Authentication Code 1 is 8 bytes data (it should be validated)
  - 16 bytes Encrypted Data (EK1) is encrypted by RKI-KEK according to KSN (CBC TDES algorithm, the IV are 0)

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00	See below			03

Response Body:

<ACK><20 bytes ASCII code KSN of RKI-KEK><32 bytes ASCII code Encrypted Data (EK2)> or <NAK><Error Code>

Where:

- <20 bytes ASCII code KSN of RKI-KEK> is 10 bytes KSN; it is advanced to next KSN (referred to as next-KSN hereafter)
- <32 bytes ASCII code Encrypted Data (EK2)> is 16 bytes Encrypted Data (EK2):
  - The plaintext of 16 bytes Encrypted Data (EK2) is
    - <Authentication Code 1><Authentication Code 2>
    - Authentication Code 1 is 8 bytes data
    - Authentication Code 2 is 8 bytes data
    - 16 bytes Encrypted Data (EK2) is encrypted by RKI-KEK according to next-KSN (CBC TDES algorithm, the IV are 0)

11.10 New KSN/Key Pair

This command gets a new KSN/Key pair.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02			55 52 4B 49 32 <M bytes TR-31 format Data (EK3)>			03

Command Body: 55 52 4B 49 32 < M bytes TR-31 format Data (EK3)>

Where:

- 52 4B 49 is 'RKI'
- <M bytes TR-31 format Data (EK3)> is ASCII code "TR-31 Edition B" format data.
    - The N byte Key data is encrypted by KBEK (generated from KBPK, KBPK is generated from Data Key of RKI-KEK), and 4 bytes MAC is generated from data (detailed in below) encrypted with KBMK (generated from KBPK, KBPK is generated from Data Key of RKI-KEK).
    - The KSN of RKI-KEK is advanced to the next KSN (referred as next-KSN hereafter.) Please see the below detailed algorithm.
- TR-31 Block data format is: KBH + KBH_OB (Optional Block) + ASCII code Result + ASCII code MAC data.
- 2N bytes ASCII code Encrypted Key – N bytes Encrypted Key
- 16 bytes ASCII code MAC data – 8 bytes MAC data

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00	<ACK><6 bytes ASCII code MAC Result> <NAK><Error Code>			03

Where: <6 bytes ASCII code MAC Result> is 3 bytes MAC Result (MAC Result of Key) using 8/16 bytes Key as Key, ECB DES/TDES Encrypt 8/16 bytes 0s.

11.11 Change to Default Setting

This command does not have any function ID and is only two bytes long. It resets all settings to their default values for all groups, except settings of security and encryption type and device serial number.

Note: the final byte of this command is the LRC, which is calculated using all bytes upstream including STX and ETX.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02			53 18			03
Full Command: <STX><0x53><0x18><ETX><LRC>						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00	See below			03
Response Body: <NAK> or <ACK>						

11.12 Get Firmware Version

This command retrieves the current firmware version of the device.

Note: the final byte of this command is the LRC, which is calculated using all bytes upstream including STX and ETX.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02			52 22	71		03
Full Command: <STX><0x52><0x22><ETX><LRC> **Note:** The LRC of 0x71 is the XOR of all previous bytes (STX through ETX inclusive).						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00	See below			03
Response Body: <ACK><STX><Firmware Version String><ETX><LRC> or <NAK>						

11.13 Reset

Device resets and reboots after it responds ACK.

This is the highest priority command in the device except that of Key Loading State.

Note: the final byte of this command is the LRC, which is calculated using all bytes upstream including STX and ETX.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02			46 49	0E		03
Full Command: <STX><0x46><0x49><ETX><0x0F> LRC (value 0x0E) is the result of XOR of all previous bytes.						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00	See below			03
Response Body: <ACK> or <NAK>						

11.14 Get Serial Number

This command retrieves the device serial number: 10 bytes (range 0x20~0x7e).

Note: the final byte of this command is the LRC, which is calculated using all bytes upstream including STX and ETX.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02			52 4E	1D		03
Full Command: <STX><0x52><0x4E><ETX><LRC>						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00	See below			03
Response Body: <NAK> or <ACK><STX><0x4E><Data Length><String Length><Serial Number String><ETX><LRC> Where: • <Data Length> is 1 byte for length of the <String Length> and <Serial Number String> • <String Length>: the length must be 10 (0x0A).						

- <Serial Number String>: The value of ASCII character is between 20h and 7eh

11.15 Get Data Encryption Key KSN

This command gets the DUKPT key serial number and counter.

Note: the final byte of this command is the LRC, which is calculated using all bytes upstream including STX and ETX.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02			52 51	02		03
Full Command: <STX><0x52><0x51><ETX><0x02> LRC is the result of XOR of all previous bytes.						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00	See below			03
Response Body: <NAK> or <ACK><STX><0x51><0x0A><10 Bytes KSN><ETX><LRC>						

11.16 Get Security Level

The Security Level can be raised by loading a data encryption key, but it can never be lowered.

Note: the final byte of this command is the LRC, which is calculated using all bytes upstream including STX and ETX.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02	02	00	52 7E			03
Command body: <STX><'R'><0x7E><ETX><LRC> LRC (value 0x0E) is the result of XOR of all previous bytes.						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00	See below			03
Response Body: <NAK> or						

<ACK><STX><0x51><0x0A><10 Bytes KSN><ETX><LRC>

Response Data:
<NAK> or
<ACK><STX><0x7E><Security level><ETX><LRC>

Where:
<Security level>: '1'- Security level 1 or '3'- Security level 3

11.17 Get Key Status

This command reviews key status for the Data Encryption Key, Manufacture Key, RKI-KEK and firmware key.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02			52 A1	F2		03

Full Command: <STX><0x52><0xA1><ETX><0xF2>

LRC (value 0xF2) is the result of XOR of all previous bytes.

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00	See below			03

Response Body:
<NAK> or
<ACK><STX><0xA1><0x01><Key Status><ETX><LRC>

Where:
<Key Status> is one of the following:

bit 0 is defined for RKI-KEK	0: no key; 1: key exists.
bit 1 is defined for the data encryption key	0: no key; 1: key exists.
bit 2 is defined for the pairing key	0: no key; 1: key exists.
bit 3 is defined for the check value	0: not loaded; 1: has loaded.
bit 4 is defined for the RSA manufacture key	0: not loaded; 1: has loaded.
bit 5 is defined for RSA firmware key	0: not loaded; 1: has loaded.
bit 6, bit 7	Reserved

11.18 Get Model Number

The command reads the part (or model) number of the unit. The part number is read-only.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02			52 A2	F1		03
Full Command: <STX><0x52><0xA2><ETX><0xF1> LRC (value 0xF1) is the result of XOR of all previous bytes.						

Response Exampl1

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00	See below			03
Response Body: <NAK> or <ACK><STX><Model Number><ETX><LRC> **Where:** Model Number is typically IDEM-24X (USB-KB)						

11.19 Arm To Read

This command is used with function ID 0x1A when its value is set to buffer mode (0x32).

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02			50 01	50		03
Full Command: <STX><0x50><0x01><ETX><0x50> LRC (value 0xF2) is the result of XOR of all previous bytes.						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00	See below			03
Response Body: <NAK> or <ACK> **Where:** <Arm Setting> is defined as: 0x30: Arm to read magnetic card data 0x32: Enable read magnetic card data when function ID 0x1A is set to 0x30 or disable magnetic card data when function ID 0x1A is set to 0x32						

11.20 Read Buffer Data

This command reads magnetic card data. Send it after sending the Arm To Read command and swiping the card.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02			51 01	51		03
Full Command: <STX><0x51><0x01><ETX><0x51> LRC (value 0x51) is the result of XOR of all previous bytes.						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00	See below			03
Response Body: <NAK> or <ACK><Magnetic Card Data> **Where:** <Track Selection Option> is defined as: 0x30: any tracks 0x3x: x from 1 to 7, 7 for all three tracks; each bit represents one required track						

11.21 Retrieve White List

The command is used for Gift Card Definition.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02			46 4D 01	0B		03
Full Command: <STX><0x46><0x4D><0x01><0x03><0x0B>						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00	See below			03

Response Body:
<NAK> or
<ACK><Length Whitelist ASN.1 BLK><Whitelist ASN.1 BLK>

Where:

- <Length Whitelist ASN.1 BLK> is 2 bytes; the format is LenL LenH, the length of <Whitelist ASN.1 BLK>
- <Whitelist ASN.1 BLK> is N bytes; it is ASN.1 Block data for the White List for aGift Card

11.22 Remove White List

This command is used to remove the Gift Card Definition.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
			<STX><46><4D><02><ETX><CheckLRC>			

Response data:
<NAK>
Or
<ACK>

11.23 Set White List

This command sets the Gift Card Definition. See Appendix L: White List Format for the white list structure.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02			46 4D 03			03

Full Command: <STX><0x46><0x4D><0x03><ETX><LRC>

Command body: <STX><46><4D><03><Length Whitelist ASN.1 BLK><Whitelist ASN.1 BLK><2 bytes MAC Length><MAC Data><ETX><LRC>

Where:

- <Length Whitelist ASN.1 BLK> is 2 bytes; the format is LenL LenH, the length of <Whitelist ASN.1 BLK>
- <Whitelist ASN.1 BLK> is N bytes; it is ASN.1 Block data for the White List for a Gift Card

- <MAC Length> is 2 byte data—the length of <MAC Data>
  - Is fixed to 0x00 0x00 for non-PCI devices
  - Is fixed to 0x1E 0x00 for PCI devices
- <MAC Data>
  - MAC data does not exist For Non-PCI (non-SRED) devices
  - For PCI device, MAC is Fix 30 bytes data:
    - <MAC Value Length> is 2 byte and fixed to 0x10 0x00
    - <MAC Value> is 16 bytes; MAC value is MAC-HOST. The msgX is "46 4D 03<Length Whitelist ASN.1 BLK><Whitelist ASN.1 BLK><2 bytes MAC Length><MAC Value Length>"
    - <MAC Key KSN Length> is 2 byte and fixed to 0x0A 0x00
    - <MAC Key KSN> is 10 bytes – MAC DUKPT Key KSN

Note:
1. The new white list overwrites the old one.
2. The white list is saved in ASN.1 Block format.
3. The total length of <Length Whitelist ASN.1 BLK><Whitelist ASN.1 BLK> is not more than 512.

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00	See below			03
Response Body: <NAK> or <ACK>						

11.24 Review All Settings

This command does not have any function ID. When sent, the unit will respond with all of its setting values.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02			52 1F	4C		03
Full Command: <STX><0x52><0x1F><ETX><0x4C>						

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00	See below			03
Response Body: <NAK> or <ACK><STX><Function Block Number><FuncSETBLOCK1>…<FuncSETBLOCKn><ETX><LRC>						

Each function-setting block <FuncSETBLOCK> has the following format: <FuncID><Len><FuncData>.

Where:

- <Function Block Number> is total function block number.
- <FuncID> is one byte identifying the setting(s) for the function.
- <Len> is a one byte length count for the function-setting block <FuncData>
- <FuncData> is the current setting for this function; it follows the same format as this function's sending command.
- <FuncSETBLOCK> are in the order of their Function ID<FuncID>.

11.25 MSR Setting Command

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		**00**	**See Appendix N: Setting Parameters (Function ID) and Values**			**03**

11.26 Set MSR Reading Characteristics

This command sets the reader's MSR reading characteristics.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	73 53 01 1A 01 <MSR Reading Option>			03

Command Body: 73 53 01 1A 01 <MSR Reading Option>

Where:

MSR Reading Status	Option	Comment
MSR Function Off	0x30	
MSR Function Enable Auto Mode	**0x31**	**Only USB-KB**
MSR Function Enable Buffer Mode & Notification Off	0x32	It is default in USB-HID
MSR Function Enable Buffer Mode & Notification On	0x33	Only USB-HID

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00	06			03

Response Body: 06 = ACK

Note:
1. If MSR Reading Function Off, the response will be 15 6D 11 for below ICC Group command:
- Arm to Read
- Read Buffer Data

2. If MSR Reading Function Off, there is not output data after MSR Card is swiped.

12.0 Command and Response Body with ITP Protocol

Device use ITP protocol to communicate with other device in default mode, only function of swiping magnetic card can be used in this mode, the interface can be USB-KB or RS-232, the commands lists as below.

12.1 Load Certificate for TransArmor RSA Algorithm

This command loads the reader's TransArmor RSA Algorithm certificate.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	<STX> <48> <12> <Length of Certification> <Certification, X.509, PEM format> <ETX> <CheckLRC>			03

Command Body: <STX> <48> <12> <Length of Certification> <Certification, X.509, PEM format> <ETX> <CheckLRC>

Where:

- <Length of Certification> is 2 bytes, format is LenL LenH, it is length of <Certification, X.509, PEM format>
- The Certification Chain includes the following:
  - Root CA Certification
  - Intermediate CA Certification
  - TA Key Certification
- The detailed:
  - Root CA Certification is hardcoded in firmware. The **Certificate** needs to be loaded is **Intermediate CA Certificate** or **TA Key Certificate**.
  - POS software sends Key Update message requesting for Intermediate CA Certificate and it receives the Intermediate CA Certificate in the Key Update response from First Data Server. POS software passes the Intermediate CA Certificate to Card Reader. The Card Reader uses the Root CA Key to validate the Intermediate CA Certificate. After the Intermediate CA Certificate is validated, the Root CA Key extracts the Intermediate CA Key from the Intermediate CA Certificate.
  - POS software sends Key Update message requesting for TA Key Certificate and it receives the TA Key Certificate in the Key Update response from First Data Server. POS software passes the TA Key Certificate to Card Reader. The Card Reader uses the Intermediate CA Key to validate the TA Key Certificate. After the TA Key Certificate is validated, the Intermediate CA Key extracts the TA Key. The TA Key is used to encrypt the sensitive cardholder data in the TA authorization request. TA Key Certificate uses TA Key ID as Common Name.
- The more detailed:
  - If Intermediate CA Certification is loaded successfully, TA Key Certification should be erased immediately.

Note: This command is only Valid in firmware support TransArmor RSA algorithm.

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00				03
Response Body: <NAK> or <ACK>						

12.2 Read Certificate for TransArmor RSA Algorithm

This command reads the reader's TransArmor RSA Algorithm certificate.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	<STX> <48> <13> <Certificate Type> <ETX> <CheckLRC>			03

Command Body: Response Body: <STX> <48> <13> <Certificate Type> <ETX> <CheckLRC>

Where:

- <Certificate Type> - 1 byte

Certificate Type Value	Certificate Type
0x01	Intermediate CA Certificate
0x02	TA Key Certificate

Note: This command is only Valid in firmware support TransArmor RSA algorithm.

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00				03

<NAK> or <ACK> <STX> <Certificate Type> <Length of Certification> <Certification, X.509, PEM format> <ETX> <CheckLRC>

Where:

- <Length of Certification> is 2 bytes, format is LenL LenH, it is length of <Certification, X.509, PEM format>
- The Certification Chain includes the following:
  - Intermediate CA Certification
  - TA Key Certification

13.0 USB-KB Commands

The following commands can be used to set or get parameters in Keyboard Mode. See Appendix E for a list of all function IDs.

13.1 Set/Get for One Byte ID

If the parameter length is one byte, the following command is available.

Set Command Body: <STX><'S'><Function ID><0x01><Setting Value><ETX><LRC>

Response Data:
<NAK> or
<ACK>

Read Command Body: <STX><'R'><Function ID><ETX><LRC>

Response Data:
<NAK> or <0x16> or
<ACK><STX><Function ID><01><Setting Value><ETX><LRC>

13.2 Set/Get for Multi Bytes ID

If the parameter length is more than one byte, following command is available;

Set Command Body:
<STX><'S'><Function ID><Data Length><Array Element Length><Array Element Value><ETX><LRC>

Response Data:
<NAK> or
<ACK>

Read Command Body:
<STX><'R'><Function ID><ETX><LRC>

Response Data:
<NAK> or <0x16> or
<ACK><STX><Function ID><Data Length><Array Element Length><Array Element Value><ETX><LRC>

Where:
<Data Length> is 1 byte for the length of <Array Element Length><Array Element Value>
<Array Element Length> is 1 byte for the length of <Array Element Value>

13.3 Device Enter into Load Important Data State

Command Body: 78 46 F5 <Length of Data>49 44 54 45 43 48 <Work State>

Where: <Work State> is 1 byte; value only is 0x02

Response Body: 15 <Error Code> or 06

13.4 Set Intercharacter Delay for USB-KB interface

This command sets the reader's USB-KB interface intercharacter delay.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	<STX> <'S'> <12> <01> <Setting Value> <ETX> <CheckLRC>			03

Command Body: <STX> <'S'> <12> <01> <Setting Value> <ETX> <CheckLRC>

Where:

Setting Value	Delay Time (ms)
0x30	**1200**
0x31	2000
0x32	5000
0x33	10000
0x34	20000
0x35	50000
0x36	0

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00				03

Response Body: <NAK> or <ACK>

Note:
1. For Mac computer, suggestion of <Setting Value> is 0x33 (10000 ms)
2. For Virtual System of Mac computer, suggestion of <Setting Value> is 0x33 (10000 ms).

13.5 Review Intercharacter Delay for USB-KB interface

This command retrieves the reader's current USB-KB interface intercharacter delay settings.

Command Example

STX	Len Low	Len High	Command Body	LRC	CHK SUM	ETX
02		00	<STX> <'R'> <12> <ETX> <CheckLRC>			03

Command Body: <STX> <'R'> <12> <ETX> <CheckLRC>

Response Example

STX	Len Low	Len High	Response Body	LRC	CHK SUM	ETX
02		00				03

Response Body: <NAK> or <ACK> <STX> <12><01> < Setting Value > <ETX> <CheckLRC>

14.0 Appendix A: LED and Beeper States

LED & Beeper States, from Deactivation State to Activation State.

PK – Public Key (Manufacture Key)
FK – Firmware Key
CV – Check Value
DTV – Date & Timer Value

Device State	Definition	MSR LEDs		ICC LEDs	Beeper State		Note
		FW Control	SW Control	SW Control	FW Control	SW Control	
Load Important Data State	Activation Need Load **PK**, **FK**, **CV**, **and DTV**	Blink Red / Blue	Blink Red / Blue	None	Always beepslow	Always beepslow	Only exist in Manufacture.
Idle State	Device is waiting for Implement MSR or ICC function.	Solid Blue	SW Control	SW Control	None	SW Control	
Load Key State	Want load Data encryption Key, or MAC DUKPT Key, or RKI-KEK.	Blink Red / Green / Blue	Blink Red / Green / Blue	None	None	None	The State only valid in USB-HID interface.
Ready for MSR swipe	Device is ready for MSR payment – Only Supported by MSR BUFFER mode	Blink Blue (Interval – 500ms)	SW Control	None	Not beep	SW Control	Only Supported by MSR BUFFER mode. If MSR work in Auto Mode, the Status is not exist.
MagStripe successful	Magnetic card read correct Then back to idle	Solid Green 2 seconds	SW Control	None	1 beep	SW Control	MagStripe successfully
MagStripe Unsuccessful	Magnetic card read incorrect Then back to idle	Solid Red 2 seconds	SW Control	None	None	SW Control	MagStripe Unsuccessfully
Ready for ICC insertion	The application is ready to interact with the chip card	Blink Blue (Interval – 500ms)	SW Control	SW Control	None	SW Control	Ready for ICC insertion
Card Seated	Card seated	Solid Green	SW Control	SW Control	None	SW Control	
Card Seated incorrectly	Only Valid after receive “Power-On ICC” command and Implement Failed.	Solid Red 2 seconds	SW Control	SW Control	None	SW Control	Finish this State, Device goes into “Card Remove message for incorrectly / unsuccessfully” state.
Successful EMV Transaction	Successfully complete an EMV L2 Transaction	Solid Green 2 seconds	SW Control	SW Control	None	SW Control	Finish this State, Device goes into “Card Remove message” state for "successful" state.
Unsuccessful EMV Transaction	Unsuccessfully complete an EMV L2 Transaction	Solid Red 2 seconds	SW Control	SW Control	None	SW Control	Finish this State, Device work into “Card Remove message for incorrectly / unsuccessfully” state.
Card remove message for incorrect / unsuccessful	“Remove Card” message	Solid Red until Card is removed	SW Control	SW Control	Beeps until card removed	SW Control	If Card was removed, Device goes into Idle State.

Card remove message for successful	“Remove Card” message	Blink Blue (Interval – 1S) until Card is removed	SW Control	SW Control	Beeps until card removed	SW Control	If Card was removed, Device goes into Idle State.

Name	**Tone Note**
"Normal" Tone	beep tone once
"Complete" Tone	beep short tone 2 times
"Invalid" Tone	beep short tone 3 times

15.0 Appendix B: Default Values

The following is a table of default values (and *available* settings, within parentheses) for various function IDs when using USB communications.

Function ID	Hex	Len	Description	Default Setting	Description	Opt
BeepID	11	1	Beep Setting	'2' ('0'~'4')	Beep volume high and frequency high	e
ChaDelayID	12	1	Character Delay	'0' ('0'~'5') '6'	2 ms inter-character delay '6 for 0 mS delay	ke
TrackSelectID	13	1	Track Selection	'0' ('0'~'9')	Any Track '0'-any; '1'-'7'—bit 1 tk1, bit 2 tk2; bit 3 tk3. '8'—tk1-2; '9' tk2-3	e
PollingIntervalID	14	1	Polling Interval	1 (1 ~ 255)	USB HID Polling Interval	ue
TrackSepID	17	1	Track Separator	CR/Enter	CR for commands, Enter for KB any character supported except 00 which means none.	e
SendOptionID	19	1	Send Option	'1' ('0'~3f)	Sentinel and Account number control 0x30 - Not send start/end sentinel and send all data on Track 2, not error notification. Control Key Output. 0x31 - Send start/end sentinel and send all data on Track 2, not send error notification. Control Key Output. 0x32 - Not send start/end sentinel and only send account number on Track 2, not send error notification. Control Key Output. 0x33 - Send start/end sentinel and only send account number on Track 2, not send error notification. Control Key Output. 0x34 - Not send start/end sentinel and send all data on Track 2, send error notification(default). Control Key Output. 0x35 - Send start/end sentinel and send all data on Track 2, send error notification. Control Key Output. 0x36 - Not send start/end sentinel and only send account number on Track 2, send error notification. Control Key Output. 0x37 - Send start/end sentinel and only send account number on Track 2, send error notification. Control Key Output.	e

					0x38 - Not send start/end sentinel and send all data on Track 2, not error notification. Alt Key Output. 0x39 - Send start/endsentineland send all data on Track 2, not send error notification. Alt Key Output. 0x3a - Not send start/end sentinel and only send account number on Track 2, not send error notification. Alt Key Output. 0x3b - Send start/end sentinel and only send account number on Track 2, not send error notification. Alt Key Output. 0x3c - Not send start/end sentinel and send all data on Track 2, send error notification(default). Alt Key Output. 0x3d - Send start/end sentinel and send all data on Track 2, send error notification. Alt Key Output. 0x3e - Not send start/end sentinel and only send account number on Track 2, send error notification. Alt Key Output. 0x3f - Send start/end sentinel and only send account number on Track 2, send error notification. Alt Key Output.	
MSRReadingID	1A	1	MSR Reading	'1' ('0'~'2')	Enable/Disable MSR Reading 0x30 – MSR Reading Disabled 0x31 – MSR Reading Auto Mode Enabled 0x32 – MSR Reading Buffered Mode Enabled	e
DecodingMethodID	1D	1	Decoding Direction	'1' ('0'~'3')	Reading Direction 0x30 – Raw Data Decoding in Both Directions. 0x31 – Decoding in Both directions. 0x32 – Moving Stripe Along Head in Direction of Encoding. 0x33 – Moving Stripe Along Head Against Direction of Encoding.	e
TerminatorID	21	1	Terminator	CR/Enter	CR for KB mode, Enter for KB	e
FmVerID	22	1	Firmware Version		Get firmware version string	nr
USBHIDFmtID	23	1	USB HID Fmt	'0' ('0'-'8')	'0' USB-HID mode '8' USB-KB mode	ur
ForeignKBID	24	1	Foreign KB	'0' ('0' ~ '9')	Foreign Keyboard	ke
CustSetID	30	1		4(00-0x1f)	.0 No SN after hash .4 Enhanced Secured Output will have SN after hash	e

Track1PrefixID	34	6	Track 1 Prefix	0	No prefix for track 1, 6 char max	e
Track2PrefixID	35	6	Track 2 Prefix	0	No prefix for track 2, 6 char max	e
Track3PrefixID	36	6	Track 3 Prefix	0	No prefix for track 3, 6 char max	e
Track1SuffixID	37	6	Track 1 Suffix	0	No suffix for track 1, 6 char max	e
Track2SuffixID	38	6	Track 2 Suffix	0	No suffix for track 2, 6 char max	e
Track3SuffixID	39	6	Track 3 Suffix	0	No suffix for track 3, 6 char max	e
PinKeyID	3E	1		0x00, 0x5A	0x5A– PinKey Can only set at level 1; Won't reset by 53 18;	e
PrePANID	49	1	PAN to not mask	4 (0-6)	# leading PAN digits to display	e
PostPANID	4A	1	PAN to not mask	4 (0-4)	# of trailing PAN digits to display	e
MaskCharID	4B	1	mask the PAN with this character	'*' 20-7E	any printable character	e
CrypTypeID	4C	1	encryption type	'1' ('1'-'2')	'1' : 3DES ; '2' : AES ;	e
SerialNumberID	4E	10	device serial #	10 bytes '0'	10 bytes printable character	r
DispExpDateID,	50	1	mask or display expiration date	'0''0'-'1'	'1' don't mask expiration date	e
Mod10ID	55	1	include mod10 check digit	'0' '0'-'2'	don't include mod10, '1' display mod10, '2' display wrong mod10	e
KeyManageTypeID	58	1	DUKPT or Fixed key	'1' ('0'-'1')	'0' fixed key '1' DUKPT key	e
HashOptID,	5C	1		'3' ('0'-'7')	Send tk1-2 hash bit 0:1 send tk1 hash; bit 1:1 send tk2 hash; bit2:1 send tk3 hash.	e
LRCID	60	1	LRC character	'0' ('0'~'1')	Without LRC in output	e
T17BStartID	61	1	Track 1 7 Bit Start Char	'%'	'%' as Track 1 7 Bit Start Sentinel	e
T16BStartID	62	1	T16B Start	'%'	'%' as Track 1 6 Bit Start Sentinel	e
T15BStartID	63	1	T15B Start	';'	';' as Track 1 5 Bit Start Sentinel	e
T27BStartID	64	1	Track 2 7 Bit Start Char	'%'	'%' as Track 2 7 Bit Start Sentinel	e

T25BStartID	65	1	T25BStart	';'	';' as Track 2 5 Bit Start Sentinel	e
T37BStartID	66	1	Track 3 7 Bit Start Char	'%'	'%' as Track 3 7 Bit Start Sentinel	e
T36BStartID	67	1	T36BStart	'!'	'!' as Track 3 6 Bit Start Sentinel	e
T35BStartID	68	1	T35BStart	';'	';' as Track 3 5 Bit Start Sentinel	e
T1EndID	69	1	Track 1 End Sentinel	'?'	'?' as End Sentinel	e
T2EndID	6A	1	Track 2 End Sentinel	'?'	'?' as End Sentinel	e
T3EndID	6B	1	Track 3 End Sentinel	'?'	'?' as End Sentinel	e
T1ERRSTARTID	6C	1	Track 1 error code	'%'	start sentinel if track 1 error report	e
T2ERRSTARTID	6D	1	Track 2 error code	';'	start sentinel if track 2 error report	e
T3ERRSTARTID	6E	1	Track 3 error code	'+'	start sentinel if track 3 error report	e
SecureLrcID	6F	1	Secured output format Lrc option	'1' ('0'-'1')	'1' to send LRC in secured output data	e
T28BStartID	72	1	JIS T12 SS/ES	7f (00-ff)	Output 00 when connector is USB-KB	e
T38BStartID	73	1	JIS T3 SS/ES	7f (00-ff)	Output 00 when connector is USB-KB	e
SyncCheckID	7B	1	check for track sync bits	'0' ('0'-2')	check leading & trailing sync bits on track data (if poorly encoded card)	
SecurityLevelID	7E	1	Security level	'1' ('1'-'3')	'1' no load Data encryption Key '3' has loaded Data encryption Key	nr
EnOptionID	84	1	Encryption Option (Forced encryption or not)	08	Bit 0: T1 force encrypt Bit 1 : T2 force encrypt Bit 2 : T3 force encrypt Bit3 : T3 force encrypt when card type is 0	e
EnStructID	85	1	Encryption Structure (Enhanced or original)	'0' ('0'-'1')	'0' –Original Encrypt Structure '1' – Enhanced Encrypt Structure	e
MaskOptID	86	1	Masked / clear data sending option	0x07	Bit0: T1 mask allowed Bit1: T2 mask allowed Bit2: T3 mask allowed	e
HashTypeID	88	1	Hash type selection	'0' ('0'-'1')	'0' - SHA-1 20 bytes '1' - SHA-2 32 bytes	e
T3ExpDataPos	89	1	T3 Exp Data Position	'6'('0'-'9')	'6' – Chinese '4' – ISO 4909	e
RkiTimeOutID	AD	1	Remote Key Injection Timeout	2, 2-60	2-60 Remote Key Injection Time Out Setting	e

Equip2ID	AE	1	Equip Setting byte for dif readers or new settings;Not reset by 5318	0(00-0xFF)	.0 OMNI Beep .0 (iMagPro2) 1-Detect V-BUS and control charging. 0-Always allow charging if power is present. (Default 0) .1 Good Beep (Rudi) .2 Bad Beep (Rudi) .3 New beep (Rudi) .4 USB Serial Number	e
CustSet2ID	AF	1	Bitwise customer settings	0(00-0xFF)	.0 1- Block shifted reading .1 1- Read passbook .2 1- Send all tracks in passbook 0- only send 1 trk .3 1- Allow 15 min time out to erase saved PAN. .4 1–support loopPay; 0-not support loopPay. loopPay data will become triple track identical data, will not treat as CC card	e
PrefixID	D2	15	Preamble	0	No Preamble, 15 char max	e
PostfixID	D3	15	Postamble	0	No Postamble, 15 char max	e

Note:

- e: can be *written* (0x53) and *read* (0x52);
- nr: only can be *read* (0x52);
- r : can be set only one time, then always can be read (0x52);
- ue: can be *written* (0x53) and *read* (0x52), it can work when USB-HID connected;
- ke: can be *written* (0x53) and *read* (0x52), it can work when USB-KB connected;

16.0 Appendix C: Error Codes – NGA Protocol

Error Code	Definition
0x0400	Related Key was not loaded
0x0410	Non-SRED Device need Load Manufacture Key and Firmware Key
0x0500	Key Same / Duplicate key detected
0x0702	PAN is Error
0x0F00	Encryption or Decryption Failed
0x1001	file arg invalid
0x1002	file open failed
0x1003	file operation failed
0x2001	memory not enough
0x3001	smartcard OK
0x3002	smartcard fail
0x3003	smartcard init failed
0x3004	fallback situation
0x3005	smartcard absent
0x3006	smartcard timeout
0x3007	MSR card error
0x5001	parsing tags failed
0x5002	card data element duplicate
0x5003	data format incorrect
0x5004	app no term
0x5005	no matching AID
0x5006	mandatory object missing
0x5007	app selection retry
0x5008	amount error get
0x5009	card rejected
0x5010	AIP not received
0x5011	AFL not received
0x5012	AFL len out of range
0x5013	SFI out of range
0x5014	AFL incorrect
0x5015	Exp date incorrect
0x5016	Eff date incorrect
0x5017	ISS code table out of range
0x5018	Cryptogram type incorrect
0x5019	PSE by card not supported
0x5020	user language selected
0x5021	service not allowed
0x5022	no tag found
0x5023	card blocked
0x5024	length incorrect
0x5025	card communications error
0x5026	TSC not increased
0x5027	hash incorrect
0x5028	arc not presenced

0x5029	ARC invalid
0x5030	comm no online
0x5031	tran type incorrect
0x5032	app not supported
0x5033	app not selected
0x5034	lang not selected
0x5035	term data not present
0x5036	Card blocked
0x5500	No RKI-KEK
0x5501	RKI-KEK STOP
0x5504	Validate Authentication Code Error
0x5505	Encrypt or Decrypt data failed
0x5506	Not Support the New Key Type
0x5507	New Key Index is Error
0x5508	Step Error
0x5509	Remote Key Injection Timeout (Latest Command is Timeout)
0x550A	MAC Error
0x550B	Key Usage Error
0x550C	Mode Of Use Error
0x550F	Other Error
0x6000	Save or Config Failed / Or Read Config Error, Flash Error
0x6001	CVM type unknown
0x6002	CVM aip not supported
0x6003	CVM tag 8E missing
0x6004	CVM tag 8E format error
0x6005	CVM code is not supported
0x6006	CVM cond code is not supported
0x6007	CVM no more
0x6008	PIN bypassed before
0x6200	No Serial Number
0x6201	No White List
0x6900	Invalid Command – Protocol is right, but task ID is invalid
0x6A00	Unsupported Command – Protocol and task ID are right, but command is invalid
0x6A01	Unsupported Command – Protocol and task ID are right, but command is invalid – In this State
0x6B00	Unknown parameter in command – Protocol task ID and command are right, but parameter is invalid
0x6B10	ASN.1 Data Error
0x6C00	Unknown parameter in command – Protocol task ID and command are right, but length is out of the requirement.
0x7300	DUKPT is STOP (21 bit 1)
0x8100	Timeout
0x8200	Wrong operate step
0x8300	Decode MSR Error
0x8500	No Swipe MSR Card
0x2C02	No Microprocessor ICC seated

0x2C06	No card seated to request ATR
0x8B10	ICC error on power-up
0xE313	IO line low -- Card error after session start
0x9042	Invalid LCL-KEK
0x9046	Invalid Data encryption Key
0x9047	Do not support this key
0x9052	Invalid RKI-KEK
0x9054	TR31 checks failed
0x9055	DOMAC Verification Failed
0x9057	LCL-KEK exists
0xD00	Unable to turn off encryption
0xF002	ICC communication timeout
0xF003	ICC communication Error
0xF005	ICC Encrypted C-APDU Data Structure Length Error or Format Error.
0xF200	AID List / Application Data does not exist
0xF201	Terminal Data does not exist
0xF202	TLV format is error
0xF203	AID List is full
0xF204	Any CA Key does not exist
0xF205	CA Key RID does not exist
0xF206	CA Key Index it not exist
0xF207	CA Key is full
0xF208	CA Key Hash Value is Error
0xF209	Transaction format error
0xF20A	The command will not be processing
0xF20B	CRL does not exist
0xF20C	CRL number exceed max number
0xF20D	Amount, Other Amount, Transaction Type are missing
0xF20E	The Identification of algorithm is mistake
0xF20F	No Financial Card
0xF210	In Encrypt Result state, TLV total Length is greater than Max Length
0xF211	ICC L2 is not in idle state

17.0 Appendix D: EMV L2 Response Codes

A two-byte response code is returned after each transaction phase (Start Transaction, Authenticate Transaction, Complete Transaction). The first byte may contain flag bits in the bottom two bits (see note further below). For more information, see the EMV_RESULT_CODE enumeration in the Universal SDK's IDTDef.h file. Also see Appendix C for other possible values.

Response Code (2 Bytes)	Meaning
0x00,0x00	APPROVED (offline)
0x00,0x01	DECLINED(offline)
0x00,0x02	APPROVED
0x00,0x03	DECLINED
0x00,0x04	GO ONLINE
0x00,0x05	CALL YOUR BANK
0x00,0x06	NOT ACCEPTED
0x00,0x07	USE MAGSTRIPE
0x00,0x08	TIME OUT
0x00,0x10	(start transaction success)
Error Result Code	TERMINATE

Note:
The first byte of the response code can have the following flags:

Bit 0 --- if transaction has advice, this bit is 1.
Bit 1 --- if transaction has a reversal, this bit is 1.

Example: code 0x0203 means DECLINED (03) with reversal (02).

18.0 Appendix E: Function IDs

72 Series: ICC/EMV Commands

Category	FuncID	Function Description
0x46(Function)	0x01	EMV Level2 - Application Data
	0x02	EMV Level2 - Terminal Data
	0x03	EMV Level2 - AID List
	0x04	EMV Level2 - CA Public Key
	0x05	EMV Level2 - Transaction
	0x06	EMV Level2 - Cancel Transaction
	0x07	EMV Level2 - Retrieve Transaction Result
	0x08	EMV Level2 - Get Version
	0x09	EMV Level2 - Get Kernel / Congifuration Check Value
	0x0A	Remove Transaction Amount Log
	0x20	Control ICC LEDs
	0x23	Get EMV Level One Version
	0x24	Get ICC Reader Status
	0x41	Exchange APDU (Application Protocol Data Unit)
	0x4D	Power off Command
	0x61	Exchange R-APDU Encryption for special case
	0x62	Get KSN
	0x63	Exchange APDU Encryption
	0x6E	Power On (Get ATR) Command
	0x85	EMV Level2 - Certification Revocation List
	0x86	EMV Level2 - Serial Number
	0x87	EMV Level2 - Terminal Identification
0x52/0x53 (setting/review)	0x04	Set Card Type (EMV/ISO)
	0x05	Set ICC L1 Transaction Timeout
	0x11	Set ICC Reading Characteristics
	0x20	Set Pre/Post PAN Ctrl Data Len
	0x21	Set ASCII Mask Data
	0x22	Set BCD Mask Data

73 Series: MSR Commands

Category	FuncID	Function Description
0x46(Function)	0x51	Arm to Read
	0x52	Read MSR Buffer Data
0x52/0x53 (setting/review)	0x00	Review / Default All of this task/group
	0x11	BeepID (FW Control)
	0x12	Character Delay for USB KB
	0x13	Track Selection character
	0x14	Polling Interval for USB interrupt
	0x17	Track Separator character.
	0x19	Send Option
	0x1A	MSR Reading
	0x1D	Decoding Direction option
	0x21	Terminator character
	0x24	Foreign Keyboard
	0x30	Customer setting
	0x34	Track 1 Prefix character
	0x35	Track 2 Prefix character
	0x36	Track 3 Prefix character
	0x37	Track 1 Suffix character
	0x38	Track 2 Suffix character
	0x39	Track 3 Suffix character
	0x49	PAN to not mask, leading PAN digits to display
	0x4A	Leading PAN digits to display
	0x4B	Trailing PAN digits to display
	0x50	Mask or display expiration date
	0x55	Include mod10 check digit
	0x58	DUKPT or Fixed key
	0x5C	Send Track Hash Data
	0x60	Output Clear LRC character or not
	0x61	Track 1 7 Bit Start Character
	0x63	Track 1 5 Bit Start Character
	0x64	Track 2 7 Bit Start Character
	0x65	Track 2 5 Bit Start Character
	0x66	Track 3 7 Bit Start Character
	0x68	Track 3 5 Bit Start Character
	0x69	Track 1 End Sentinel Character

	0x6A	Track 2 End Sentinel Character
	0x6B	Track 3 End Sentinel Character
	0x6C	Track 1 error code character
	0x6D	Track 2 error code character
	0x6E	Track 3 error code character
	0x6F	Secured output format LRC option or not
	0x72	JIS Track1, Track2 SS/ES character
	0x73	JIS Track3 SS/ES character
	0x7B	Check for track sync bits
	0x84	Encryption Option (Forced encryption or not)
	0x85	Enhanced Encrypt Structure
	0x86	Masked / clear data sending option
	0x88	Hash type selection
	0x89	T3 Exp Data Position
	0xAD	Remote Key Injection Timeout
	0xAE	Special settings if bit4 high send serial number during enumeration
	0xAF	Bitwise customer settings
	0xD2	Preamble
	0xD3	Postamble

78 Series: Utility (General Purpose) Commands

Category	FuncID	Function Description
0x46(Function)	0x01	Get Version
	0x04	Beeper Control
	0x06	LED Control
	0x11	Get Bootloader Version
	0x20	Get Model Number
	0x25	Get Key Status
	0x30	Get Key Status
	0x3E	Gets DUKPT KSN
	0x49	Reset
	0x4D	Retrieve/Setting White List
	0x7A	Enter Bootloader
	0xC3	Erase Serial Number Command
	0xF2	Sets HSM DUKPT Key
	0xF3	Sets Key
0x52/0x53 (setting/review)	0x00	Review / Default All of this task/group
	0x01	Remote Key Injection Timeout

	0x02	Account DUKPT Key variant (Data Key/PIN Key)
	0x03	Account DUKPT Key Encryption/Decryption Mode(TDES/AES)
	0x07	Set Encryption Control
	0x10	Set Interface Type
	0x11	Set LED Control
	0x12	Set Beeper Control
	0x50	Set Date Time

19.0 Appendix F: USB-KB Command Set

The following commands are supported when the unit is in its original MSR-only mode of operation (EMV not enabled), communicating via USB-KB. After EMV mode is enabled, USB-HID commands beginning with 72, 73, 75, 78, or 7F (as documented throughout this manual) should be used instead, when applicable.

Command Name	Command; hex value; description	Command ID (hex)	Sub Command	Data	Parameter Range
Arm to Read		50	-	01 + <Arm Setting>	-
Read Buffer Data		51	-	01 + <Track Selection Option>	-
Get DUKPT Key KSN	<F>; 46; function command	3E	-	<Length of Data><KeyNameIndex><Length of Key Slot><Key Slot>	-
White List - Retrieve		4D	01	-	-
White List - Remove			02	-	-
White List - Set			03	<Length Whitelist ASN.1 BLK><Whitelist ASN.1 BLK><2 bytes MAC Length><MAC Data>	-
Reset Device		49	-	-	-
Erase Serial Number		73	-	EraseMSRKey123	-
Get Beep Setting	<R>; 52; review command	11	-	-	0x30~0x34
Get Character Delay		12	-	-	0x30~0x35
Get Track Selection		13	-	-	0x30~0x39
Get Polling Interval		14	-	-	0x01~0xFF

Get Track Separator		17	-	-	0x00~0xFF
Get Send Option		19	-	-	0x30~0x3F
Get MSR Reading		1A	-	-	0x30~0x33
Get Decoding Direction		1D	-	-	0x30~0x33
Review MSR All		1F	-	-	-
Get Terminator		21	-	-	0x00~0xFF
Firmware Version		22	-	-	-
USB Interface Type		23			0X30
Get Foreign Keyboard		24	-	-	0x30~0x39
Get Custom Setting		30	-	-	0x00~0x1F
Track 1 Prefix		34	-	-	0x00~0xFF
Track 2 Prefix		35	-	-	0x00~0xFF
Track 3 Prefix		36	-	-	0x00~0xFF
Track 1 Suffix		37	-	-	0x00~0xFF
Track 2 Suffix		38	-	-	0x00~0xFF
Track 3 Suffix		39	-	-	0x00~0xFF
PIN Key Type		3E	-	-	0x00,0x5A

Preamble PAN to not mask		49	-	-	0x00~0x06
Postamble PAN to not mask		4A	-	-	0x00~0x04
Characte Mask PAN		4B	-	-	0x20~0x7E
Encryption Type		4C	-	-	0x31-0x32
Serial Number		4E	-	-	0x20~0x7E
Mask/Display Expiration Date		50	-	-	0x30~0x31
Include Mod10 Check Digit		55	-	-	0x30~0x32
Key Type		58	-	-	0x31
Hash Option		5C	-	-	0x30~0x37
LRC Character		60	-	-	0x30~0x31
Get Track 1 7 Bit Start Char		61	-	-	0x00~0xFF
Get Track 1 5 Bit Start Char		63	-	-	0x00~0xFF
Get Track 2 7 Bit Start Char		64	-	-	0x00~0xFF
Get Track 2 5 Bit Start Char		65	-	-	0x00~0xFF
Get Track 3 7 Bit Start Char		66	-	-	0x00~0xFF
Get Track 3 5 Bit Start Char		68	-	-	0x00~0xFF
Get Track 1 End Sentinel		69	-	-	0x00~0xFF
Get Track 2 End Sentinel		6A	-	-	0x00~0xFF
Get Track 3 End Sentinel		6B	-	-	0x00~0xFF
Get Track 1 Error Code		6C	-	-	0x00~0xFF

Get Track 2 Error Code		6D	-	-	0x00~0xFF
Get Track 3 Error Code		6E	-	-	0x00~0xFF
Get Secured Output format Lrc		6F	-	-	0x30~0x31
Get JIS T12 SS/ES		72	-	-	0x00~0xFF
Get JIS T3 SS/ES		73	-	-	0x00~0xFF
Check for Track Sync Bits		7B	-	-	0x30~0x32
Security Level		7E	-	-	0x31-0x33
Encryption Option		84	-	-	0x00~0x1F
Get Encryption Structure		85	-	-	0x31
Get Masked /Clear Data Sending Option		86	-	-	0x00~0x07
Get Hash Type Selection		88	-	-	0x30~0x31
Get T3 Exp Data Position		89	-	-	0x30~0x39
Get Remote Key Injection Timeout		AD	-	-	0x02~0x3C
Get Equip Setting Byte		AE	-	-	0x00~0xFF
Get Bitwise Customer Settings		AF	-	-	0x00~0xFF
Get Preamble		D2	-	-	0x00~0xFF
Get Postamble		D3	-	-	0x00~0xFF

Default MSR All	<S>; 53; setting command	18	-	-	-
Set Beep Setting		11	-	01 + <Option>	0x30~0x34
Set Character Delay		12	-	01 + <Option>	0x30~0x35
Set Track Selection		13	-	01 + <Option>	0x30~0x39
Set Polling Interval		14	-	01 + <Option>	0x01~0xFF
Set Track Separator		17	-	01 + <Option>	0x00~0xFF
Set Send Option		19	-	01 + <Option>	0x30~0x3F
Set MSR Reading		1A	-	01 + <Option>	0x30~0x33
Set Decoding Direction		1D	-	01 + <Option>	0x30~0x33
Set Terminator		21	-	01 + <Option>	0x00~0xFF
USB Interface Type		23		01 + <Option>	0X30
Set Foreign Keyboard		24	-	01 + <Option>	0x30~0x39
Set Custom Setting		30	-	01 + <Option>	0x00~0x1F
Set Track 1 Prefix		34	-	<length> + <Option length> + <Option>	0x00~0xFF
Set Track 2 Prefix		35	-	<length> + <Option length> + <Option>	0x00~0xFF
Set Track 3 Prefix		36	-	<length> + <Option length> + <Option>	0x00~0xFF
Set Track 1 Suffix		37	-	<length> + <Option length> + <Option>	0x00~0xFF
Set Track 2 Suffix		38	-	<length> + <Option length> + <Option>	0x00~0xFF
Set Track 3 Suffix		39	-	<length> + <Option length> + <Option>	0x00~0xFF
Set Preamble PAN to not mask		49	-	01 + <Option>	0x00~0x06

Set Postamble PAN to not mask		4A	-	01 + <Option>	0x00~0x04
Set Characte Mask PAN		4B	-	01 + <Option>	0x20~0x7E
Set encrypted type		4C	-	01 + <Option>	0x31-0x32
Set serial number		4E	-	<length> + <Option length> + <Option>	0x20~0x7E
Set Mask/Display Expiration Date		50	-	01 + <Option>	0x30~0x31
Set Include Mod10 Check Digit		55	-	01 + <Option>	0x30~0x32
Set DUKPT Key		58	-	01 + <Option>	0x31
Set Hash Option		5C	-	01 + <Option>	0x30~0x37
Set LRC Character		60	-	01 + <Option>	0x30~0x31
Set Track 1 7 Bit Start Char		61	-	01 + <Option>	0x00~0xFF
Set Track 1 5 Bit Start Char		63	-	01 + <Option>	0x00~0xFF
Set Track 2 7 Bit Start Char		64	-	01 + <Option>	0x00~0xFF
Set Track 2 5 Bit Start Char		65	-	01 + <Option>	0x00~0xFF
Set Track 3 7 Bit Start Char		66	-	01 + <Option>	0x00~0xFF
Set Track 3 5 Bit Start Char		68	-	01 + <Option>	0x00~0xFF
Set Track 1 End Sentinel		69	-	01 + <Option>	0x00~0xFF
Set Track 2 End Sentinel		6A	-	01 + <Option>	0x00~0xFF
Set Track 3 End Sentinel		6B	-	01 + <Option>	0x00~0xFF
Set Track 1 Error Code		6C	-	01 + <Option>	0x00~0xFF

Set Track 2 Error Code		6D	-	01 + <Option>	0x00~0xFF
Set Track 3 Error Code		6E	-	01 + <Option>	0x00~0xFF
Set Secured Output format Lrc		6F	-	01 + <Option>	0x30~0x31
Set JIS T12 SS/ES		72	-	01 + <Option>	0x00~0xFF
Set JIS T3 SS/ES		73	-	01 + <Option>	0x00~0xFF
Set Check for Track Sync Bits		7B	-	01 + <Option>	0x30~0x32
Set Encryption Option		84	-	01 + <Option>	0x00~0x1F
Set Encryption Structure		85	-	01 + <Option>	0x31
Set Masked /Clear Data Sending Option		86	-	01 + <Option>	0x00~0x07
Set Hash Type Selection		88	-	01 + <Option>	0x30~0x31
Set T3 Exp Data Position		89	-	01 + <Option>	0x30~0x39
Set Remote Key Injection Timeout		AD	-	01 + <Option>	0x02~0x3C
Set Equip Setting Byte		AE	-	01 + <Option>	0x00~0xFF
Set Bitwise Customer Settings		AF	-	01 + <Option>	0x00~0xFF
Set Preamble		D2	-	<length> + <Option length> + <Option>	0x00~0xFF
Set Postamble		D3	-	<length> + <Option length> + <Option>	0x00~0xFF

20.0 Appendix G: LCD Foreign Language Mapping Table

ID	Message ID	English	French	Spanish	Chinese
0	MSG_NULL				
1	MSG_AMOUNT	AMOUNT	MONTANT	CANTIDAD	□ □
2	MSG_AMOUNT_OK	AMOUNT OK?	MONTANT OK	MONTO CORRECTO?	□ □ □ □
3	MSG_APPROVED	APPROVED	APPROUVE	APROVADO	□ □
4	MSG_CALL_YOUR_BANK	CALL YOUR BANK	APPE VOTRE BANQE	LLAME A SU BANCO	□ □ □ □ □ □ □
5	MSG_CANCEL_OR_ENTER	CANCEL OR ENTER	ANNULE OU ENTRER	CANCEL O ENTRAR	□ □ □ □ □
6	MSG_CARD_ERROR	CARD ERROR	ERREUR CARTE	ERROR DE TARJETA	□ □ □ □
7	MSG_DECLINED	DECLINED	REFUSE	DECLINADO	□ □ □
8	MSG_ENTER_AMOUNT	ENTER AMOUNT	ENTRER MONTANT	INGRESE MONTO	□ □ □ □
9	MSG_ENTER_PIN	ENTER PIN:	ENTRER PIN:	ENTRAR NPI:	□ □ □ □ □
10	MSG_INCORRECT_PIN	INCORRECT PIN	NIP INCORRECT	NPI INCORRECTO	□ □ □ □
11	MSG_ICC_MSR1	SWIPE OR INSERT	PASSER OU INSERT	MOVER O INSERT	□ □ □ □ □ □
12	MSG_ICC_MSR2	CARD	CARTE	TARJETA	□
13	MSG_INSERT_CARD	INSERT CARD	INSERT CARTE	INSERTAR TARJETA	□ □ □
14	MSG_USE_CHIP_READER	USE CHIP READER	UTI LECTEUR CHIP	USO CHIP LECTOR	□ □ □ □ □
15	MSG_NOT_ACCEPTED	NOT ACCEPTED	PAS ACCEPTE	DENEGADO	□ □ □ □
16	MSG_PIN_OK	GET PIN OK			□ □ □ □
17	MSG_PLEASE_WAIT	PLEASE WAIT...	ATTENDRE...	POR FAVOR ESPERE	□ □ □
18	MSG_PROCESSING_ERROR	PROCESSING ERROR	ERREUR DE TRAITE	ERROR PROCESANDO	□ □ □ □
19	MSG_USE_MAGSTRIPE	USE MAGSTRIPE	USAGE MAGSTRIPE	USO DE MAGSTRIPE	□ □ □ □ □
20	MSG_TRY_AGAIN	TRY AGAIN	REESSAYER	VUELV INTENTARLO	□ □ □
21	MSG_ONLINE	GO ONLINE	GO LIGNE	GO LINEA	□ □
22	MSG_TRANSACTION_ERROR	TRANSACTION ERR	ERREUR DE TRANS	ERROR DE TRANSAC	□ □ □ □
23	MSG_TERMINATE	TERMINATE	RESILIER	TERMINAR	□ □
24	MSG_ADVICE	ADVICE	CONSEILS	CONSEJOS	□ □
25	MSG_TIMEOUT	TIME OUT	TIMEOUT	TIEMPO DE ESPERA	□ □
26	MSG_PROCESSING	PROCESSING...	PROCESSUS...	PROCESANDO...	□ □ □ □ □ □
27	MSG_PIN_TRY_EX	PIN TRY LIMIT EX	PIN TRY DEPASSE	TRY PIN SUPERADA	□ □ □ □ □ □ □ □ □
28	MSG_ISSUER_AUTH_FAIL	ISSUER AUTH FAIL	EMETTEUR FAIL	EMISOR FALLA	□ □ □ □ □ □ □
29	MSG_CONTINUE_PROCESS	CONTINUE PROCESS	CONTINUER LA	CONTINUAR PROCES	□ □ □ □
30	MSG_GET_PIN_ERROR	GET PIN ERROR	GET PIN ERROR	OBTENER PIN ERR	□ □ □ □
31	MSG_GET_PIN_FAIL	GET PIN FAIL	GET PIN FAIL	OBTENER PIN FALL	□ □ □ □ □ □
32	MSG_NOKEY_GET_PIN	NO KEY GET PIN	NO KEY GET PIN	NO CLAVE GET PIN	□ □ □ □ □ □
33	MSG_CANCELLED	CANCELLED	ANNULE	CANCELADO	□ □
34	MSG_LAST_PIN_TRY	LAST PIN TRY			□ □ □ □ □ □ □ □ □
35	MSG_WELCOME	WELCOME	BIENVENUE	BIENVENIDOS	
36	MSG_AMOUNT_OTHER	AMOUNT OTHER	MONTANT AUTRE	IMPORTE OTRAS	
37	MSG_ENTER_AMOUNT_OTHER	ENTER AMOUNT OTHER	ENTRER MONTANT AUTRES	ENTRAR IMPORTE OTRAS	
38	MSG_CAPK_HASH_VALUE_FAIL	CAPK HASH VALUE FAIL	CAPK HASH VALEUR FAIL	CAPK HASH VALOR FAIL	
39	MSG_REMOVE_CARD	REMOVE CARD	RETIRER LA CARTE	RETIRE LA TARJETA	
40-63	RFU	RFU	RFU	RFU	
64	MSG_TRY_MSR_AGAIN	TRY MSR AGAIN	REPASSER VOTRE CARTE	INTENTE MSR NUEVO	
65	MSG_LAST_MSR_TRY	LAST MSR TRY	DERNIERE TENTATIVE	ULTIMO MSR INTENTO	
66	MSG_TRY_ICC_AGAIN	TRY ICC AGAIN	INSERER VOTRE CARTE	INTENTE ICC DE NUEVO	
67	MSG_REMOVE_CARD	REMOVE CARD	RETIRER VOTRE CARTE	QUITE TARJETA	

21.0 Appendix H: Terminal Configurations

Annex A of Book 4 of the EMV specifications categorizes terminal configurations as follows:

Environment	Operational Control Provided By:		
	Financial Institution	Merchant	Cardholder
Attended			
Online only	11	21	--
Offline with online capability	12	22	--
Offline only	13	23	–
Unattended			
Online only	14	24	34
Offline with online capability	15	25	35
Offline only	16	26	36

Augusta's L2 default kernel configuration conforms to Type 21.

The following table shows the various terminal configurations supported by ID TECH's L2 kernel. Terminal Capabilities 2C and 5C are supported by Augusta. To force the use of 2C, issue command stream 02060072530128013 23b2103. To force the use of 5C, issue command stream 02060072530128013 53c2403. Individual values (if marked Minor) can be changed via the 72 46 02 03 (Set Terminal Data) command. Values marked Major should not be changed.

Terminal Capabilities	1C	2C	3C	4C	5C	6C	7C	Major/ Minor
Card Data Input Capability								
Terminal Type	22	21	25	25	21			
Manual Key Entry	Yes	No	No	No	No			Minor
Magnetic Stripe	Yes	Yes	Yes	Yes	Yes			Minor
IC with Contacts	Yes	Yes	Yes	Yes	Yes			N/A
CVM Capability								
Plaintext PIN	Yes	No	Yes	No	No			Major
Online Enciphered PIN	Yes	No	Yes	No	No			Major
Signature (Paper)	Yes	Yes	No	No	Yes			Major
Offline Enciphered PIN	Yes	No	Yes	No	No			Major
No CVM	Yes	Yes	Yes	Yes	Yes			Major
Security Capability								
SDA and DDA	Yes	Yes	Yes	Yes	Yes			Major
Card Capture	No	No	No	No	No			Minor
CDA	Yes	Yes	Yes	Yes	Yes			Major
Transaction Type Capability								
Tran Type – Cash	Yes	Yes	No	No	Yes			Major
Tran Type – Goods	Yes	Yes	Yes	Yes	Yes			Major
Tran Type – Services	Yes	Yes	Yes	Yes	Yes			Major
Tran Type – Cash Back	Yes	Yes	No	No	Yes			Major
Tran Type – Inquiry	No	No	No	No	No			Minor

Terminal Capabilities	1C	2C	3C	4C	5C	6C	7C	Major/ Minor
Tran Type – Transfer	No	No	No	No	No			Minor
Tran Type – Payment	No	No	No	No	No			Minor
Tran Type – Admin	No	No	No	No	No			Minor
Tran Type – Cash Deposit	No	No	No	No	No			Minor
Terminal Data Input Capability Minor								
Keypad	Yes	Yes	Yes	Yes	Yes			
Numeric Keys	Yes	Yes	Yes	Yes	Yes			Minor
Alpha and Special Character Keys	Yes	Yes	Yes	Yes	Yes			Minor
Command Keys	Yes	Yes	Yes	Yes	Yes			Minor
Function Keys	Yes	Yes	Yes	Yes	Yes			Minor
Terminal Data Output Capability								
Print, Attendant	Yes	Yes	No	No	Yes			Minor
Print, Cardholder	No	No	Yes	Yes	No			Minor
Display, Attendant	Yes	Yes	No	No	Yes			Minor
Display, Cardholder	No	No	Yes	Yes	No			Minor
Code Table 10	No	No	No	No	No			If value of supported table changed: Minor If removing all the supported tables or indicating one as supported when previously none were: Major
Code Table 9	No	No	No	No	No			
Code Table 8	No	No	No	No	No			
Code Table 7	No	No	No	No	No			
Code Table 6	No	No	No	No	No			
Code Table 5	No	No	No	No	No			
Code Table 4	No	No	No	No	No			
Code Table 3	No	No	No	No	No			
Code Table 2	No	No	No	No	No			
Code Table 1	Yes	Yes	Yes	Yes	Yes			
Application Selection								
PSE	Yes	Yes	Yes	Yes	Yes			Major
Cardholder Confirmation	Yes	Yes	Yes	No	No			Major
Preferred Display Order	No	No	No	No	No			Major
Partial AID Selection	Yes	Yes	Yes	Yes	Yes			
Multi Language	Yes	Yes	Yes	Yes	Yes			Minor
EMV Language Selection Method	Yes	Yes	Yes	Yes	Yes			Minor
Common Character Set	Yes	Yes	Yes	Yes	Yes			
Data Authentication								
Max CA Public Key	248	248	248	248	248			
Exponents	3 and 2^16 +1	3 and 2^16 +1	3 and 2^16 +1	3and 2^16 +1	3 and 2^16 +1			
Revocation of Issuer PK Certificate	Yes	Yes	Yes	Yes	Yes			Major
Certificate Revocation List Format	RID/CA PK Index Cert SN	RID/CA PK Index Cert SN	RID/CA PK Index Cert SN	RID/CA PK Index Cert SN	RID/CA PK Index Cert SN			Major
Default DDOL	Yes	Yes	Yes	Yes	Yes			Major
Manual Act. when CA PK loading fails	No	No	No	No	No			Major
CA PK verified with Checksum	Yes	Yes	Yes	Yes	Yes			Major
Cardholder Verification Method								
Bypass PIN Entry	Yes	No	No	No	No			Major
Subsequent Bypass PIN Entry	Yes	No	No	No	No			

Terminal Capabilities	1C	2C	3C	4C	5C	6C	7C	Major/ Minor
Get Data for PIN Try Counter	Yes	No	Yes	No	No			Major
Fail CVM	Yes	Yes	Yes	Yes	Yes			Major
Amount known before CVM process	Yes	Yes	Yes	Yes	Yes			Major
Terminal Risk Management								
Floor Limit Checking	Yes	Yes	Yes	Yes	Yes			Major
Random Transaction Selection	Yes	No	Yes	Yes	No			Major
Velocity Checking	Yes	Yes	Yes	Yes	Yes			Major
Transaction Log	Yes	No	Yes	Yes	No			Major
Exception File	No	No	No	No	No			Major
TRM irrespective of AIP setting	Yes	Yes	Yes	Yes	Yes			Minor
Terminal Action Analysis								
Terminal Action Codes Supported	Yes	Yes	Yes	Yes	Yes			Major
TAC can be changed	Yes	Yes	Yes	Yes	Yes			
TAC can be deleted of disabled	No	No	No	No	No			
Default Act. Codes prior to 1st Gen AC	No	No	No	No	No			Major
Default Act. Codes after 1st Gen AC	No	No	No	No	No			Major
TAC/IAC – Default Skipped	No	Yes	No	No	Yes			
TAC/IAC – Default normal processing	Yes	No	Yes	Yes	No			
CDA failure detected prior TA Analysis	Yes	Yes	Yes	Yes	Yes			
Mode 1 (CDA on ARQC and 2GenAC)	Yes	Yes	Yes	Yes	Yes			
Mode 2 (CDA on ARQC only)	No	No	No	No	No			
Mode 3 (No CDA on ARQC or 2GenAC)	No	No	No	No	No			
Mode 4 (CDA on 2GenAC only)	No	No	No	No	No			
Completion Processing								
Forced Online	Yes	N/A	No	No	N/A			Major
Forced Acceptance	No	No	No	No	No			Major
Advices	No	No	No	No	No			Major
Issuer Referrals	Yes	Yes	No	No	Yes			Major
Batch Data Capture	Yes	Yes	Yes	Yes	Yes			Major
Online Data Capture	Yes	Yes	Yes	Yes	Yes			Major
Default TDOL	Yes	Yes	Yes	Yes	Yes			Major
Exception Handling								
POS Entry Mode	80	80	80	80	80			Minor
Miscellaneous								
PIN Pad	Yes	No	Yes	No	No			Minor
Amount and PIN on same keypad	Yes	No	No	No	No			Minor
ICC/Magstripe Reader Combined	No	No	Yes	Yes	No			Minor
If Combined, is Magstripe read first?	N/A	N/A	No	No	N/A			Minor
Supports Account Type selection	Yes	Yes	Yes	Yes	Yes			Minor
Support “on fly” script processing	No	No	No	No	No			
Issuer Script device limit > 128 bytes	No	No	No	No	No			
If limit > 128, value supported?	N/A	N/A	N/A	N/A	N/A			
Internal Date Management	Yes	Yes	Yes	Yes	Yes			

22.0 Appendix I: Transarmor Encryption Support

Augusta-series readers can be configured to support TransArmor public key encryption. The commands in this section apply.

The examples show OUT and IN streams corresponding to commands sent (OUT) to the reader or received (IN) from the reader. All commands stream values are shown as hex.

"Cert Data" portions of command streams are the respective certificates in hex. Certificates need not be decoded from base64 before being encoded as hex.

Erase existing key:

OUT: 0210007846f40b00494454454348140200000c08403
IN: 02010006060603 (ACK)

Load Subordinate CA certificate:

Load Certificate File Command (78 46 12 LenL LenH <Cert Data)>

Certificate File (Base64):
-----BEGIN CERTIFICATE-----
MIIClDCCAXwCAQUwDQYJKoZIhvcNAQEFBQAwDzENMAsGA1UEAxMEVEFDQTAeFw0x
MzA0MTUyMTM2NTFaFw0yMzA0MTUyMTM2NTFaMBExDzANBgNVBAMTBlRBQ0FUMTCC
ASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBALphW/0QEJCAiCXeRgt957Om
DGOGMXfKtQ3mtlNLNd8RyDCzrlaVtGg8NBCA/U2fijUlimGCHBD6ZkAIU3PF7pzJ
LWvStzHpIZd90XX4lHePhCcRHlauw4StFjXCyWt5wS6BYKr9MINoo6T5OWxVVypn
I4+dkEuFnnI3eHAiOdzGs0DMXAzR8mXYmhoE1gAtmHFxuUcSBXX8bTdRuyXkmvHO
XR+U2d8uf3LCcfbSHqnohHKQwmnay76qOUFmPJGDgb6qonOYlqFCyaHhDXUGBWYP
yArVk7DIySWakWfbqFEFNKhNOBYYrEnbwrEVKJRyz+PqMdzssqVFtEygUPOsTw0C
AwEAATANBgkqhkiG9w0BAQUFAAOCAQEATU0W1TuVUFjFAt3tUNFZg9VwagRR/kq+
Ei6B0pTALuHIl+xQA5aQpfPy030vZ0SIgH+pH41JSHPkrp+2qqZAxrEuCYyxbY0P
C1Jg3erPEOBRfwGBSj3tQRodEG/Np9hHOme9QkXCBfEfXucrAChfIFjnrQn1W/o4
nZ8PTRZ84Q08lTdrY/gnbhP4XXf6vXDg8ys02NNsvK0peKJiM0vX2qB/9xk9/yPc
jD7BIn3sQYBbKqC+VQ3rXWrFq85LDQAb6873RRHVFAibhYf0ee6/VFTeu6q7j1ID
VW04VJs2M9IuKEn6T7Z+eYfxvOYKz1XxwzG85WAUPBcuXUzKhHuydA==
-----END CERTIFICATE-----

OUT:
02B103784612AC032D2D2D2D2D424547494E20434552544946494341544 52D2D2D2D2D4D4949436C444343415
8774341515577445159 4A4B6F5A496876634E41514546425141774 47A454E4D41734741315545417 84D45564546
4451544165467730784D7A41304D5455794D544D324E544661467730794D7A41304D5455794D544D324E544661
4D424578447A414E42674E5642414D54426C5242513046554D5443434153497744515 94A4B6F5A496876634E415
1454242514144676745504144434341516F4367674542414C7068572F3051454A434169435865526774393537 4F6
D44474F474D58664B7451336D746C4E4C4E6438527944437A726C6156744767384E4243412F553266696A556C69
6D4743484244365A6B41495533504637707A4A4C577653747A4870495A643930585834 6C486550684363524 86C6
17577345374466A5843795774357753364259 4B72394D494E6F6F3654354F5778565679706E49342B646B4575466
E6E4933654841694F647A477330444D58417A52386D58596D686F45316741746D484678755563534 25858386254
64527579586B6D76484F58522B553264387566334C436366625348716E6F68484B51776D6E61793736714F55466

D504A4744676236716F6E4F596C7146437961486844585547425759507941725 66B37444979535761 6B576662714
645464E4B684E4F42595972456E62777245564B4A52797A2B50714D647A7373715646744579675550 4F73547730
434177454141544 14E42676B71686B6947397730424151554641414F434151454154553057315475565546 6A4641
743374554E465A67395677616752522F6B712B45693642307054414C7548496C2B7851413561517066507930333 0
765A30534967482B704834314A5348506B72702B3271715A41787245754359797862593050433 14A67336572504
54F425266774742536A337451526F6445472F4E703968484F6D6539516B5843426645665875637241436866494 66
A6E72516E31572F6F346E5A385054525A38345130386C546472592F676E6268503458586636765846738797330 3
24E4E73764B3070654B4A694D3076583271422F39786B392F7950636A443742496E337351594262 4B71432B5651
337258577246713835 4C44514162363837335252485646416962685966306565362F5646546575367137 6A314944
56573034564A73324D3949754B456E3654375A2B65596678764F594B7A315878777A47383557415550426375585
57A4B6848757964413D3D2D2D2D2D2D454E44204345525449464943415445 2D2D2D2D2DB2B603
IN: 02010006060603 (ACK)

Load encryption certificate:

Load Certificate File Command (78 46 12 LenL LenH <Cert Data)>

Certificate File:
-----BEGIN CERTIFICATE-----
MIICpTCCAY0CBQIZhYVfMA0GCSqGSIb3DQEBBQUAMBExDzANBgNVBAMTBlRBQ0FU
MTAeFw0xNDA4MTkxMjU5MjdaFw0yNDA4MTYxMjU5MjdaMBwxGjAYBgNVBAMTEVRB
Q0FUMTY2MjU3OTgyNDY0MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA
3H0EX8WSZD4SyW+6ZcY0R42wqAql/skICZSYs9mzjCsgV8wqUckrYUrnzcZ3hPwj
uViC1PRHGcjpDUQ7lmWK01scKXAHMmoYk5+HqrDms8m/ue2rlI9FBVK8N77Juku9
nZ8CB1y99pELv5txU/Qjok/xeZLUxQK2604O0nscK+9bE7LspgU0dgGVsWIYbSXe
4b6Wf8LSdh6XQGr/EiOLbRmNvLtV1tjlcNNBOASn/GGhvBVeOGOBW3SI30GxMPKb
7DpFtRlr463N2i78s2ZMCpTfclkcGaA835RI9rbg65r2J9orrzTrfEv9BQuCmavG
sRJogSCSNQF1sIVlsMfmtQIDAQABMA0GCSqGSIb3DQEBBQUAA4IBAQCAi826Pz2c
gX64BkrpFrfullSuuTO/aOBVCYtdmafmtvmWhmnXcyVXDNG6N4CG6dOxx/NUlU96
yP4P3iQLQJ03gpsfBb7EBJtlX/Re7yP/8bz/WNskCKaUCUvI80Ya9lHsKNQrxy9x
y1x/K/wNv6EmgjJLxg/GdySCMnYwNZv6W39SZMHRDMokICAKuGEWra139Opv1OtF
z7JlwBj7w8Pb7gCcQLDgVnbylF9GB/Bbr3TXLQyewabhBCRCUIwOWhtjSEW6/mHN
Vf/F605YCKC0LFwkNF169/zlKTj+L9QT39aXVzN33nSKrhVknerBXN73qlhKv9Wu
We2YGJRzOhqk
-----END CERTIFICATE-----

OUT:
02C503784612C0032D2D2D2D2D424547494E2043455254494649434154452D2D2D2D2D4D49494370544343415
930434251495A685956664D41304743537147534962334451454242515541 4D424578447A414E42674E5642414D
54426C5242513046554D544165467730784E4441344D546B784D6A55354D6A6461467730794E4441344D545978
4D6A55354D6A64614D427778476A415942674E5642414D5445565242513046554D5459324D6A55334F5467794E
4459304D494942496A414E42676B71686B6947397730424151454641414F43415138414D49494243674B4341514
5413348304558385753 5A44345379572B365A6359305234327771417 16C2F736B49435A535973396D7A6A437367
5638777155636B725955726E7A635A336850776A7556694331505248476 36A70445551376C6D574B303173634B5
841484D6D6F596B352B487172446D73386D2F756532726C49394642564B384E37374A756B75396E5A3843423179
393970454C76357478552F516A6F6B2F78655A4C5578514B323630344F306E73634B2B396245374C737067553064
6747567357495962535865346236576638 4C5364836585147722F45694F4C62526D4E764C745631746A6C634E4E
424F41536E2F474768764256654F474F42573353493330477 84D504B623744704674526C723436334E3269373873
325A4D43705466636C6B6347614138333552493972626736357232 4A396F72727A54726645763942517543 6D617
64773524A6F675343534E5146317349566C734D666D7451494441514142 4D4130474353714753496233445145 42

42515541413449424151434169383236507A326367583634426B7270467266756C6C537575544F2F614F42564359
74646D61666D74766D57686D6E5863795658444E47364E34434736644F78782F4E556C55393679503450336951 4
C514A303367707366426237454 24A746C582F52653779502F38627A2F574E736B434B6155435576493830596139
6C48734B4E5172787939787931782F4B2F774E7636456D676A4A4C78672F47647953434D6E59774E5A763657333
9535A4D4852444D6F6B4943414B7547455772613133394F7076314F74467A374A6C77426A377738506237674363
514C4467566E62796C463947422F4262723354584C5179657761626842435243554977 4F5768746A534557362F6D
484E56662F4636303559434B43304C46776B4E463136392F7A6C4B546A2B4C39515433396158567A4E33336E534
B7268566B6E657242584E3733716C684B7639577557653259474A527A4F68716B2D2D2D2D2D454E44204345525
449464943415445 2D2D2D2D2DE3CD03
IN: 02010006060603 (ACK)

23.0 Appendix J: Quick Chip and M/Chip Fast Support

ID TECH Augusta-series readers are designed to support the Quick Chip and M/Chip Fast EMV modality when operating in USB-KB mode. This is sometimes referred to as Faster EMV.

Quick Chip and M/Chip Fast is a particular way of doing contact-EMV transactions, designed to shorten the amount of time the cardholder spends waiting to remove his or her card from the reader. For most chip cards, the overall "card inserted" time is on the order of 2.0 seconds. (For cards that lack Payment System Environment capability on the chip, the overall time can be as long as 5.0 seconds.)

Quick Chip and M/Chip Fast is an *online-only* style of interaction, hence it is especially appropriate for markets that are traditionally online-only, such as the U.S. retail market. It relies on terminal configuration changes (not kernel changes), and a hard-coded chain of events surrounding Gen AC requests, to achieve its functional goals. For detailed information about how Quick Chip and M/Chip Fast works, be sure to consult the U.S. Payments Forum white paper at:

http://www.emv-connection.com/downloads/2016/09/Optimizing-Txn-Speed-WP-FINAL-February-2017.pdf

ID TECH's Augusta-series readers offer Faster EMV, or Quick Chip and M/Chip Fast capability, in conjunction with USB-Keyboard (USB-KB) mode, which is a patent-pending combination that allows Augusta to provide EMV-readiness in a virtual terminal environment. Combining EMV with Keyboard Mode means it becomes exceptionally easy for virtual terminals to support contact EMV transactions in a browser environment.

When a customer presents a card for a Quick Chip and M/Chip Fast transaction, Augusta reads the card (and carries out all the usual steps of an EMV transaction, except for Issuer Authentication and the processing of Issuer Scripts; these steps are not done in Quick Chip and M/Chip Fast), and then it outputs TLV data as *keystrokes* (which is to say, ASCII-renderable hex values) straight to the host computer. The "keystrokes" will show up in whatever text window (or text field, in a web form) currently has focus. (The payment application can intercept these keystrokes as needed, to parse/filter them, insert appropriate values in appropriate form fields, and/or go online for authorization.)

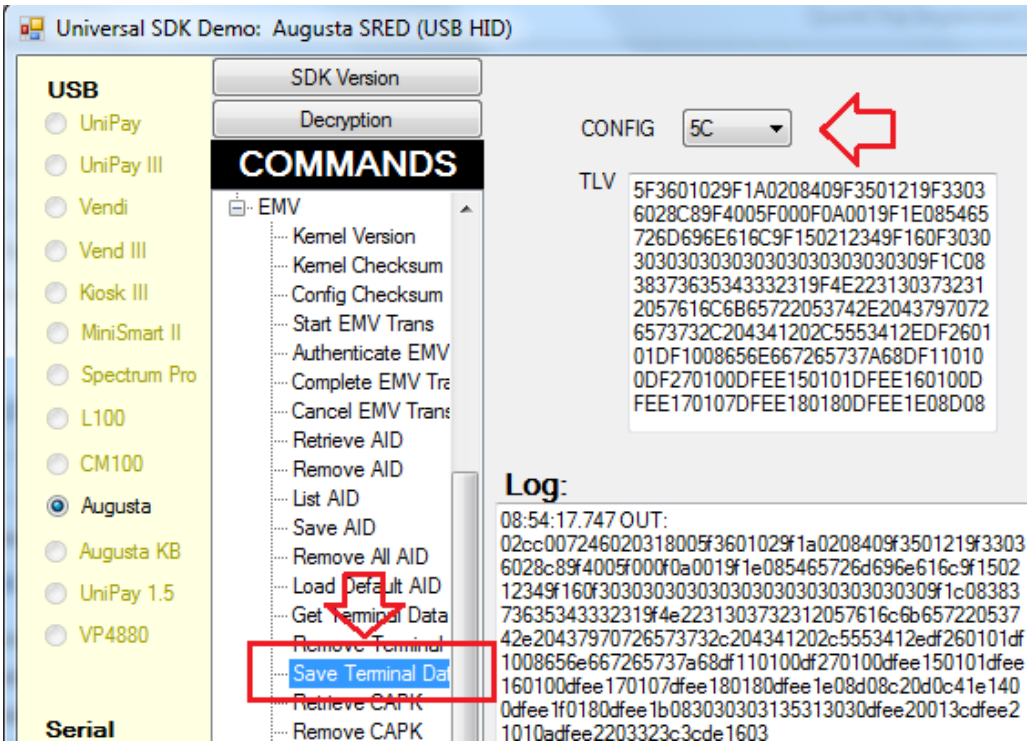
The following section describes the configuration steps needed to put an Augusta reader into Quick Chip and M/Chip Fast Mode. Please note that once the reader is in this mode, all EMV transactions will occur in USB-KB mode using Quick Chip and M/Chip Fast. (MSR transactions will occur in USB-KB mode as well.) It is possible to switch the reader back to USB-HID mode at any time.

How to Enable Quick Chip and M/Chip Fast Mode

To enable Quick Chip and M/Chip Fast, first ensure that the reader's firmware is at V1.01.003 or higher. Then perform the following steps.

1. Enable the 5C configuration of Terminal Capabilities by issuing the 72 53 01 28 01 35 command. (For more about terminal configurations, see Appendix H.)

2. Use the Save Terminal Configuration command in the Universal Demo to set the values of minor-config terminal TLVs as desired, making sure 5C is enabled. (See arrows in the screen shot below.)



3. Set the CA Public Key if it has never been set and load any AIDs or do any other initializations you wish to do, before proceeding.

4. Set the device to Quick Chip and M/Chip Fast Mode by issuing the following command: 72 53 01 29 01 31. (Turn Quick Chip Mode OFF with 72 53 01 29 01 30.)

5. Finally, change the device's USB mode to USB-KB by issuing the following command: 78 53 01 10 02 02 00.

Faster EMV (Quick Chip) Data

The output from a Quick Chip and M/Chip Fast transaction consists mostly of standard EMV tag data, since Quick Chip and M/Chip Fast is, in fact, just EMV. However, you might also encounter certain ID TECH proprietary tags (as follows):

Tag	Meaning
DFEF59	Terminal Data Setting - Default Amount
DFEF5A	Terminal Data Setting - Tags to Return
DFEF5B	Mask for Tag5A
DFEF5C	Mask for Tag56
DFEF5D	Mask for Tag57
DFEF5E	Mask for Tag9F6B
DFEF5F	Mask for TagFFEE13 (track 1)
DFEF60	Mask for TagFFEE14 (track 2)
DFEF61	Error Code
DFEF62	Allow MSR Swipe data from ICC Card

Send tags DFEF59, DFEF5A, and/or DFEF62 as Terminal Configuration values to control (respectively) the Default Amount of the transaction (normally zero, for Quick Chip), the "tags to return" as part of each transaction, and whether or not to enforce a policy of preferring EMV over MSR if a card has a chip. For tag DFEF62, if the value = 0x01, then any MSR swipe with an ICC card requires the card to be inserted instead of returning MSR swipe data. (In other words: MSR swipe data from an ICC card can *only* be captured in a fallback situation.) If the value = 0x00, any card swiped (MSR or ICC) is allowed to return MSR swipe data.

Other tags returned in a Quick Chip transaction will conform to EMVCo standard usage. Some tags may be encrypted: for details, see ID TECH document P/N 80000502-001, *Encrypted Data Output*.

An example of USB-KB data output from a Quick Chip and M/Chip Fast transaction (using Visa ADVT Test Card No. 43) is:

DF EE 25 02 00 02 **DF EE 26 02** 20 00 **DF EE 12 0A** 62 99 49 00 75 00 02 A0 02 90 **DF EF 5D 11** 47 61 CC CC CC CC 04 32 D1 01 22 01 CC CC CC CC CC **5718** D0 18 2C 6B 2D F9 C4 D2 CD 60 B7 74 C7 65 AF 14 0E 7B 53 F7 78 39 62 9A **DF EF 5B 08** 47 61 CC CC CC CC 04 32 **5A 08** D0 18 2C 6B 2D F9 C4 D2 **5F 20 1A** 56 49 53 41 20 41 43 51 55 49 52 45 52 20 54 45 53 54 20 43 41 52 44 20 34 33 **5F 24 03** 10 12 31 **5F 25 03** 95 07 01 **5F 28 02** 08 40 **5F 2A 02**08 40 **5F 2D 00 5F 34 00 5F 57 01** 00 **50 0B** 56 49 53 41 20 43 52 45 44 49 54 **4F 00 82 02** 5C 00 **84 07** A0 00 00 00 03 10 10 **8C 15** 9F 02 06 9F 03 06 9F 1A 02 95 05 5F 2A 02 9A 03 9C 01 9F 37 04 **8D 17** 8A 02 9F 02 06 9F 03 06 9F 1A 02 95 05 5F 2A 02 9A 03 9C 01 9F 37 04 **8E 0E** 00 00 00 00 00 00 00 00 1E 03 02 03 1F 00 **9C 01** 00 **9F 02 06** 00 00 00 00 00 00 **9F 03 06** 00 00 00 00 00 00 **9F 10 07** 06 01 0A 03 21 99 00 **9F 13 00 9F 20 00 9F 26 08**34 BB F5 DC 19 F3

95 24 **9F 27 01** 00 **9F 34 03** 1E 03 00 **9F 36 02** 01 04 **9F 37 04** 9F A2 99 23 **9F 38 00 9F 39 01** 07 **9F 4D 00 9F 4F 00 95 05** 42 C0 00 00 00 **9B 02** E8 00 **8A 02** 5A 33 **99 00 9F 5B 00**

Tags are shown in boldface/blue. The KSN occurs in tag DFEE12. Sensitive fields (such as tag 5A) are encrypted.

24.0 Appendix K: MAC Calculation

Some of the commands used in the Augusta S (SRED) product require MAC authentication. This is a technique in which a special hash is calculated from a data payload that has been concatenated with a (padded) key value. The presence of a key value in the payload means that the receiving party can recreate (validate) the hash only if the receiver also knows the same key value ahead of time. Thus, MACing affords a way to send and receive authenticated commands (where the command is the "data").

For commands requiring the use of MAC, the application programmer must provide a key corresponding to the MAC variant of the DUKPT key. The key will be a one-time-only key value, since it will incorporate a Key Serial Number (KSN) as part of the key. Standard DUKPT key management procedures (ANSI X9.24) apply.

Use HMAC-SHA256 (refer to RFC2104) for MAC commands, retaining only the left-most (high order) **16 bytes** of the calculation for MAC Authentication. The HMAC algorithm relies on the following construction:

$$HMAC(K,m) = H\left((K \oplus opad) | H((K \oplus ipad)|m)\right)$$

Where:

H is a cryptographic hash function (in Augusta's case, SHA256),
K is a secret key, padded to the right with extra zeros to the input block size of the hash function. (The secret key is the MAC key both-way variant, derived from MAC_DUKPT_BDK.)
m is the message to be authenticated,
| denotes concatenation,
⊕ denotes XOR,
opad is the outer padding (0x5c5c5c…5c5c, one-block-long hexadecimal constant),
ipad is the inner padding (0x363636…3636, one-block-long hexadecimal constant).

The final HMAC value consists of the first 16 bytes of the overall hash.

An example of how to construct the HMAC value is shown below.

KSN: 605906b0050106200001

BDK (Base Derivation Key): 0123456789ABCDEFFEDCBA9876543210

MAC-variant key (derived from the preceding values using ANSI X9.24):
8B2B8DB588253CAA2B0D7041A34A84BC

For our message *m* (aka "data"), we will use AABBCCDDEEFF0011AABBCCDDEEFF0011

K (padded key, overall length 64 bytes) is thus
8B2B8DB588253CAA2B0D7041A34A84BC000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000

ipad is
36363636363636363636363636363636363636363636363636363636363636363636363636363636363
636363636363636363636363636363636363636363636

K ⊕ ipad is therefore
BD1DBB83BE130A9C1D3B4677957CB28A3636363636363636363636363636363636363636363636363636
36363636363636363636363636363636363636363636

m is
AABBCCDDEEFF0011AABBCCDDEEFF0011

K' ⊕ ipad || m is
BD1DBB83BE130A9C1D3B4677957CB28A3636363636363636363636363636363636363636363636363636
36363636363636363636363636363636363636363636AABBCCDDEEFF0011AABBCCDDEEFF0011

H(K' ⊕ ipad || m) is
ed9556c53f1149776cea1d2be16755138518de723293e909347cda9c72e2c1d1

opad is
5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5
C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C

K' ⊕ opad is
D777D1E9D47960F677512C1DFF16D8E05C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5
C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C

(K' ⊕ opad) || H((K' ⊕ ipad) || m) is
D777D1E9D47960F677512C1DFF16D8E05C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5
C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5Ced9556c53f1149776cea1d2be16755138518
de723293e909347cda9c72e2c1d1

H((K' ⊕ opad) || H((K' ⊕ ipad) || m)) is
55288deac158d3bd401520858ed741b71cf54634e57fef502674d997a5410874

And thus the first 16 bytes (constituting the final HMAC) is:
55288DEAC158D3BD401520858ED741B7

See the Encrypt/Decrypt tool at http://www.idtechproducts.com/support/downloads/general-reference/946-encryptiondecryptiontool/file for source code. Also see the post at http://www.idtechproducts.com/blog/entry/how-to-decrypt-credit-card-data-part-i.

25.0 Appendix L: White List Format

Whitelist ASN.1 structure::= Sequence

```
{
      Whitelist ASN.1 structure version = 1 (INTEGER)
      BIN Exclusion::= Set
      {
            bininfo::= Sequence
            {
                  binSetName = (PRINTABLESTRING) -- 0 if no name
                  binLow = (INTEGER) -- bin >= binLow
                  binHigh = (INTEGER) -- bin <= binHigh
            }
      }
}
```

Use the **white list format** to change BIN range to <Whitelist ASN.1 BLK>.

Example:
In an example **Set White List** command of
46 4D 03 15 00 3013020101310E300C130002030ACC1B02030ACC73 00 00

3013020101310E300C130002030ACC1B02030ACC73 is <Whitelist ASN.1 BLK>, with a BIN range of *707611* to 707699.

The breakdown is as follows:

- 3013 (Sequence & Length)
- 020101 (Integer & Length & Value（1）)
- 310E (Set & Length)
- 300C (Sequence & Length)
- 1300 (Printablestring& Length)—— example name is 0
- 02030ACC1B (Integer & Length & BIN low)—— example is *707611*, which hex is 0ACC1B
- 02030ACC73(Integer & Length & BIN high) —— example is 707699, which hex is 0ACC73

26.0 Appendix M: US Common AID Support

Some cards may contain 2 AIDs: US Common Debit AID and Global Debit AID. Sometimes the merchant wants to select US Common Debit AID regardless the priority. A tag DFED48 is used to control this behavior.

26.1 US Common Debit AIDs

There currently are for selectable US Common Debit AIDs:

Visa	A0000000980840
M/C	A0000000042203
Discover	A0000001524010
China Union Pay	A000000333010108
Debit Network Alliance	A0000006200620

26.2 Global Debit AIDs

To determine if an AID is a Debit Aid, check to see if the FCI (File Control Information, under Issuer Discretionary Data Tag BF0C) contains tag 42, which is the IIN (Issuer Identification Number). The IIN is the first 6 digits of the BIN, represented by 3 bytes. If there is no tag 42, then that AID is not a debit AID.

To determine if the Global Debit AID can be replaced by US Common Debit AID, the Global AID must have tag 42 (IIN) and must match tag 42 of the US Common Debit AID.

Common Debit Preferring: DFED48 – 1 byte

Flag	Description
Bit 5-7	RFU
Bit 4	Prefer US Common Debit Network Alliance
Bit 3	Prefer US Common Debit China Union Pay
Bit 2	Prefer US Common Debit Discover
Bit 1	Prefer US Common Debit MasterCard
Bit 0	Prefer US Common Debit VISA

This tag determines which US Common AID replaces a US Global Debit AID, if they both exist on the candidate list.

27.0 Appendix N: Setting Parameters (Function ID) and Values

Following is a table of default setting and available settings (value within parentheses) for each function ID.

Function ID	Hex	Len	Description	Default Setting	Description	Opt	Comments
BeepID	11	1	Beep Setting	'2' ('0'~'4')	Beep volume high and frequency high	e	
ChaDelayID	12	1	Character Delay	'0' ('0'~'5') '6'	2 ms inter-character delay '6 for 0 mS delay	pke	It is only used for PS/2 keyboard. It is invalid.
TrackSelectID	13	1	Track Selection	'0' ('0'~'9')	Any Track '0'-any; '1'-'7'—bit 1 tk1, bit 2 tk2; bit 3 tk3. '8'—tk1-2; '9' tk2-3	e	
PollingIntervalID	14	1	Polling Interval	1 (1 ~ 255)	USB KB Polling Interval	ke	If Default command change this value, Need Re-Power it to made this change valid
TrackSepID	17	1	Track Separator	CR/Enter	CR for RS232, Enter for KB any character supported except 00 which means none.	e	
SendOptionID	19	1	Send Option	'1' ('0'~3f)	Sentinel and Account number control 0x30 - Not send start/end sentinel and send all data on Track 2, not error notification. Control Key Output. 0x31 - Send start/end sentinel and send all data on Track 2, not send error notification. Control Key Output. 0x32 - Not send start/end sentinel and only send account number on Track 2, not send error notification. Control Key Output. 0x33 - Send start/end sentinel and only send account number on Track 2, not send error notification. Control Key Output. 0x34 - Not send start/end sentinel and send all data on Track 2, send error notification(default). Control Key Output. 0x35 - Send start/end sentinel and send all data on Track 2, send error notification. Control Key Output. 0x36 - Not send start/end sentinel and only send account number on Track 2, send error notification. Control Key Output. 0x37 - Send start/end sentinel and only send account number on Track 2, send error notification. Control Key Output.	e	

Function ID	Hex	Len	Description	Default Setting	Description	Opt	Comments
					0x38 - Not send start/end sentinel and send all data on Track 2, not error notification. Alt Key Output. 0x39 - Send start/endsentineland send all data on Track 2, not send error notification. Alt Key Output. 0x3a - Not send start/end sentinel and only send account number on Track 2, not send error notification. Alt Key Output. 0x3b - Send start/end sentinel and only send account number on Track 2, not send error notification. Alt Key Output. 0x3c - Not send start/end sentinel and send all data on Track 2, send error notification(default). Alt Key Output. 0x3d - Send start/end sentinel and send all data on Track 2, send error notification. Alt Key Output. 0x3e - Not send start/end sentinel and only send account number on Track 2, send error notification. Alt Key Output. 0x3f - Send start/end sentinel and only send account number on Track 2, send error notification. Alt Key Output.		
MSRReadingID	1A	1	MSR Reading	'1' ('0'~'3')	Enable/Disable MSR Reading 0x30 – MSR Reading Disabled 0x31 – MSR Reading Auto Mode Enabled 0x32 – MSR Reading Buffered Mode Enabled& Notification Off 0x33 – MSR Reading Buffered Mode Enabled & Notification On	e	1. 0x31 is Default Value and only exist in USB-KB interface. 2. 0x32 is Default Value in USB-HID interface. 3. 0x33 is only valid in USB-HID interface for Notification.
DecodingMethodID	1D	1	Decoding Direction	'1' ('0'~'3')	Reading Direction 0x30 – Raw Data Decoding in Both Directions. 0x31 – Decoding in Both directions. 0x32 – Moving Stripe Along Head in Direction of Encoding. 0x33 – Moving Stripe Along Head Against Direction of Encoding.	e	
TerminatorID	21	1	Terminator	CR/Enter	CR for RS232, Enter for KB	e	
FmVerID	22	1	Firmware Version		Get firmware version string	nr	NGA Protocol DO NOT Support
USBHIDFmtID	23	1	USB HID Fmt	'0' ('0'-'8')	'0' USB-HID mode '8' USB-KB mode	ur	NGA Protocol DO NOT Support
ForeignKBID	24	1	Foreign KB	'0' ('0' ~ '9')	Foreign Keyboard	ke	
CustSetID	30	1		4(00-0x1f)	.0 POS-X: Level 3 Non-CC send same as Level1 .1 Level3: No empty pkt when not enough sampling bits	e	

Function ID	Hex	Len	Description	Default Setting	Description	Opt	Comments
					.2 Enhanced Secured Output will have SN after hash		
Track1PrefixID	34	6	Track 1 Prefix	0	No prefix for track 1, 6 char max	e	
Track2PrefixID	35	6	Track 2 Prefix	0	No prefix for track 2, 6 char max	e	
Track3PrefixID	36	6	Track 3 Prefix	0	No prefix for track 3, 6 char max	e	
Track1SuffixID	37	6	Track 1 Suffix	0	No suffix for track 1, 6 char max	e	
Track2SuffixID	38	6	Track 2 Suffix	0	No suffix for track 2, 6 char max	e	
Track3SuffixID	39	6	Track 3 Suffix	0	No suffix for track 3, 6 char max	e	
PinKeyID	3E	1		0x00, 0x5A	0x5A– PinKey Can only set at level 1; Won't reset by 53 18;	e	NGA Protocol DO NOT Support
PrePANID	49	1	PAN to not mask	4 (0-6)	# leading PAN digits to display	e	
PostPANID	4A	1	PAN to not mask	4 (0-4)	# of trailing PAN digits to display	e	
MaskCharID	4B	1	mask the PAN with this character	'*' (0x20-0x7E)	any printable character	e	
CrypTypeID	4C	1	encryption type	'0' ('0'-'2') '3'	'0': None; '1' : 3DES ; '2' : AES; '3' : TransArmor RSA '4' : TransArmor TDES	e	NGA Protocol DO NOT Support
SerialNumberID	4E	10	device serial #	10 bytes '0'(0x20~0x7e)	10 bytes printable character	r	NGA Protocol DO NOT Support
DispExpDateID,	50	1	mask or display expiration date	'0''0'-'1'	'1' don't mask expiration date	e	
Mod10ID	55	1	include mod10 check digit	'0' '0'-'2'	don't include mod10, '1' display mod10, '2' display wrong mod10	e	
KeyManageTypeID	58	1	DUKPT Key	'1' ('1')	'1' DUKPT key	e	
HashOptID,	5C	1		'7' ('0'-'7')	Send tk1-2 hash bit 0:1 send tk1 hash; bit 1:1 send tk2 hash; bit2:1 send tk3 hash.	e	
LRCID	60	1	LRC character	'0' ('0'~'1')	Without LRC in output	e	
T17BStartID	61	1	Track 1 7 Bit Start Char	'%'	'%' as Track 1 7 Bit Start Sentinel	e	
T15BStartID	63	1	T15B Start	';'	';' as Track 1 5 Bit Start Sentinel	e	
T27BStartID	64	1	Track 2 7 Bit Start Char	'%'	'%' as Track 2 7 Bit Start Sentinel	e	
T25BStartID	65	1	T25BStart	';'	';' as Track 2 5 Bit Start Sentinel	e	
T37BStartID	66	1	Track 3 7 Bit Start Char	'%'	'%' as Track 3 7 Bit Start Sentinel	e	
T35BStartID	68	1	T35BStart	';'	';' as Track 3 5 Bit Start Sentinel	e	
T1EndID	69	1	Track 1 End Sentinel	'?'	'?' as End Sentinel	e	
T2EndID	6A	1	Track 2 End Sentinel	'?'	'?' as End Sentinel	e	

Function ID	Hex	Len	Description	Default Setting	Description	Opt	Comments
T3EndID	6B	1	Track 3 End Sentinel	'?'	'?' as End Sentinel	e	
T1ERRSTARTID	6C	1	Track 1 error code	'%'	start sentinel if track 1 error report	e	
T2ERRSTARTID	6D	1	Track 2 error code	';'	start sentinel if track 2 error report	e	
T3ERRSTARTID	6E	1	Track 3 error code	'+'	start sentinel if track 3 error report	e	
SecureLrcID	6F	1	Secured output format Lrc option	'1' ('0'-'1')	'1' to send LRC in secured output data	e	
T28BStartID	72	1	JIS T12 SS/ES	7f (00-ff)	Output 00 when connector is USB-KB	e	
T38BStartID	73	1	JIS T3 SS/ES	7f (00-ff)	Output 00 when connector is USB-KB	e	
SyncCheckID	7B	1	check for track sync bits	'0' ('0'-2')	check leading & trailing sync bits on track data (if poorly encoded card)		
SecurityLevelID	7E	1	Security level	'1' ('1'-'3')	'1' no load Data encryption Key '3' has loaded Data encryption Key	nr	NGA Protocol DO NOT Support
EnOptionID	84	1	Encryption Option (Forced encryption or not)	08	Bit 0: T1 force encrypt Bit 1 : T2 force encrypt Bit 2 : T3 force encrypt Bit3 : T3 force encrypt when card type is 0 bit4: Bank card: T1 and T2 will be encrypted. T3 will check the type. If is iso-4909, will encrypt and allow to send mask, otherwise, will be sent in clear text. Non-bank card: Will be sent in clear text.	e	
EnStructID	85	1	Only Encryption Structure	'1'	'1' – Enhanced Encrypt Structure	e	
MaskOptID	86	1	Masked / clear data sending option	0x07	Bit0: T1 mask allowed Bit1: T2 mask allowed Bit2: T3 mask allowed	e	
HashTypeID	88	1	Hash type selection	'0' ('0'-'1')	'0' - SHA-1 20 bytes '1' - SHA-2 32 bytes	e	DO NOT Support SHA-2
T3ExpDataPos	89	1	T3 Exp Data Position	'6'('0'-'9')	'6' – Chinese '4' – ISO 4909	e	
RkiTimeOutID	AD	1	Remote Key Injection Timeout	2, 2-60	2-60 Remote Key Injection Time Out Setting	e	
Equip2ID	AE	1	Equip Setting byte for dif readers or new settings;Not reset by 5318	0(00-0xFF)	.0 OMNI Beep .0 (iMagPro2) 1-Detect V-BUS and control charging. 0-Always allow charging if power is present. (Default 0) .1 Good Beep (Rudi) .2 Bad Beep (Rudi) .3 New beep (Rudi) .4 USB Serial Number	e	
CustSet2ID	AF	1	Bitwise customer settings	0(00-0xFF)	.0 1- Block shifted reading .1 1- Read passbook .2 1- Send all tracks in passbook 0-only send 1 trk	e	

Function ID	Hex	Len	Description	Default Setting	Description	Opt	Comments
					.3 1- Allow 15 min time out to erase saved PAN. (232 only. Other interface always alow) .4 1–support loopPay; 0-not support loopPay. loopPay data will become triple track identical data, will not treat as CC card		
PrefixID	D2	15	Preamble	0	No Preamble, 15 char max	e	
PostfixID	D3	15	Postamble	0	No Postamble, 15 char max	e	
TransArmorTID	D6	8	TransArmor RSA TID	00	8 character <ASCII code TID>	e	Acceptable range: 0x20~0x7E

Note:

- e: can be written (0x53) and read (0x52);
- nr: only can be read (0x52);
- r : can be set only one time, then always can be read (0x52);
- ue: can be written (0x53) and read (0x52), it can work when USB-HID connected;
- ke: can be written (0x53) and read (0x52), it can work when USB-KB connected;
- pke: can be written (0x53) and read (0x52), it can work when PS/2 Keyboard connected (This device doesn't support PS/2 Keyboard);