



Advanced Card Systems Ltd.
Card & Reader Technologies

ACR1255U-J1

Bluetooth[®]

NFC Reader

Reference Manual V1.00





Table of Contents

1.0.	Introduction	5
1.1.	Symbols and Abbreviations	5
2.0.	Features	6
3.0.	System Block Diagram.....	7
4.0.	Hardware Design	10
4.1.	Battery.....	10
4.1.1.	Battery charging	10
4.1.2.	Battery life	10
4.2.	Bluetooth Interface.....	10
4.3.	USB Interface	10
4.3.1.	Communication Parameters	10
4.3.2.	Endpoints	11
4.4.	NFC Interface	11
4.4.1.	Carrier Frequency	11
4.4.2.	Card Polling.....	11
4.5.	User Interface	11
4.5.1.	Keys	11
4.5.2.	Mode Selection Switch.....	11
4.5.3.	Status LED	12
4.5.4.	Buzzer	13
5.0.	Software Design	14
5.1.	Bluetooth Connection Flow.....	14
5.2.	Profile Selection	15
5.3.	Authentication	16
5.4.	Communication Profile.....	18
5.5.	Frame Format	18
5.5.1.	Data Frame Format.....	18
5.6.	Bluetooth Communication Protocol	19
5.6.1.	Card Power On	20
5.6.2.	Card Power Off	21
5.6.3.	Get Slot Status	22
5.6.4.	APDU Command.....	23
5.6.5.	Notify Card Status Command	24
5.6.6.	Hardware Error Response	25
5.6.7.	Escape Command.....	26
5.7.	Mutual Authentication and Encryption Protocol.....	28
5.7.1.	Bluetooth Authentication Program Flow.....	28
5.7.2.	SPH_to_RDR_ReqAuth	28
5.7.3.	RDR_to_SPH_AuthRsp1	29
5.7.4.	SPH_to_RDR_AuthRsp	30
5.7.5.	RDR_to_SPH_AuthRsp2	31
5.7.6.	RDR_to_SPH_ACK (Error handling)	32
6.0.	Host Programming (PC-linked) API.....	33
6.1.	PC/SC API	33
6.1.1.	SCardEstablishContext.....	33
6.1.2.	SCardListReaders.....	33
6.1.3.	SCardConnect.....	33
6.1.4.	SCardControl	33
6.1.5.	ScardTransmit.....	33
6.1.6.	ScardDisconnect.....	33
6.1.7.	APDU Flow.....	34
6.1.8.	Escape Command Flow	35
6.2.	Contactless Smart Card Protocol	36



6.2.1.	ATR Generation	36
6.3.	Pseudo APDU for Contactless Interface	39
6.3.1.	Get Data	39
6.4.	PICC Commands for MIFARE® Classic (1K/4K) memory cards	40
6.4.1.	Load Authentication Keys	40
6.4.2.	Authentication for MIFARE® Classic (1K/4K)	41
6.4.3.	Read Binary Blocks	44
6.4.4.	Update Binary Blocks	45
6.4.5.	Value Block Operation (INC, DEC, STORE)	46
6.4.6.	Read Value Block	47
6.4.7.	Copy Value Block	48
6.5.	Accessing PC/SC-compliant tags (ISO 14443-4)	49
6.6.	Peripherals Control	51
6.6.1.	Get Firmware Version	52
6.6.2.	Get Serial Number	53
6.6.3.	LED Control (in USB Mode only)	54
6.6.4.	LED Status	55
6.6.5.	Buzzer Control	56
6.6.6.	Set LED and Buzzer Status Indicator Behavior	57
6.6.7.	Read LED and Buzzer Status Indicator Behavior	58
6.6.8.	Set Automatic PICC Polling	59
6.6.9.	Read Automatic PICC Polling	61
6.6.10.	Set PICC Operating Parameter	62
6.6.11.	Read PICC Operating Parameter	63
6.6.12.	Set Auto PPS	64
6.6.13.	Read Auto PPS	65
6.6.14.	Antenna Field Control	66
6.6.15.	Read Antenna Field Status	67
6.6.16.	Card Emulation Mode Conversion	68
6.6.17.	Sleep Mode Option	69
6.6.18.	Change Tx Power command	70
6.6.19.	Read Tx Power Value	71
Appendix A.	Access MIFARE DESFire tags (ISO 14443-3)	72

List of Figures

Figure 1 :	ACR1255U-J1 Architecture	7
Figure 2 :	ACR1255U-J1 USB Communication Architecture	8
Figure 3 :	Bluetooth Smart Protocol Stack	9
Figure 4 :	LED Operation Status	12
Figure 5 :	Bluetooth Connection Flow	14
Figure 6 :	Authentication Procedure	17
Figure 7 :	ACR1255U-J1 APDU Flow	34
Figure 8 :	ACR1255U-J1 Escape Command Flow	35

List of Tables

Table 1 :	Symbols and Abbreviations	5
Table 2 :	Estimated Battery Lifespan	10
Table 3 :	USB Interface Wiring	10
Table 4 :	ACR1255U-J1's Bluetooth Service	15
Table 5 :	ACR1255U-J1 Service Handles and UUID Information List	15
Table 6 :	Command Code Summary	19



Table 7 : Response Code Summary	19
Table 8 : Summary of Mutual Authentication Commands	28
Table 9 : Mutual Authentication Error codes	32
Table 10 : MIFARE® Classic 1K Memory Map	42
Table 11 : MIFARE® Classic 4K Memory Map	43
Table 12 : MIFARE Ultralight® Memory Map	43



1.0. Introduction

ACR1255U-J1 Bluetooth NFC Reader acts as an interface for the communication between a computer/mobile device and a contactless smart card or NFC-enabled device. It establishes a uniform interface from the computer/mobile device to the smart card for a wide variety of cards. By taking care of the card's particulars, it releases the computer software programmer from being responsible with smart card operations' technical details, which in many cases, are not relevant to the implementation of a smart card system.

1.1. Symbols and Abbreviations

Abbreviation	Description
ATR	Attribute Request and Attribute Response
DEP	Data Exchange Protocol Request and Data Exchange Protocol Response
DSL	Deselect Request and Deselect Response
PSL	Parameter Selection Request and Parameter Selection Response
RLS	Release Request and Release Response
WUP	Wakeup Request and Wakeup Response
DID	Device ID
BS	Sending bit duration
BR	Receiving bit duration
PP	Protocol Parameters

Table 1: Symbols and Abbreviations



2.0. Features

- USB 2.0 Full Speed Interface
- Bluetooth Smart Interface
- Plug and Play – CCID support brings utmost mobility
- USB Firmware Upgradeability¹
- Smart Card Reader:
 - Built-in antenna for contactless tag access, with reading distance of up to 60 mm (depending on tag type)
 - Supports ISO 14443 Part 4 Type A and B cards
 - Supports MIFARE® and MIFARE DESFire®
 - Supports FeliCa®
 - Supports ISO 18092 Tags (NFC Tags)
 - Built-in anti-collision feature (only one tag is accessed at any time)
 - NFC Support:
 - Card reader/writer mode
 - Card emulation mode
 - Supports AES-128 encryption algorithm
- Application Programming Interface:
 - Supports PC/SC
 - Supports CT-API (through wrapper on top of PC/SC)
- Supports Android™ 4.3 and above²
- Supports iOS 5.0 and above³
- Built-in Peripherals:
 - Two user-controllable bi-color LEDs
 - User-controllable buzzer
- Compliant with the following standards:
 - EN60950/IEC 60950
 - ISO 18092
 - ISO 14443
 - CE
 - FCC
 - VCCI
 - PC/SC
 - CCID
 - Bluetooth Smart
 - Microsoft® WHQL
 - RoHS 2
 - REACH

¹ Applicable under PC-linked mode

² PC/SC and CCID support are not applicable

³ Same as above

3.0. System Block Diagram

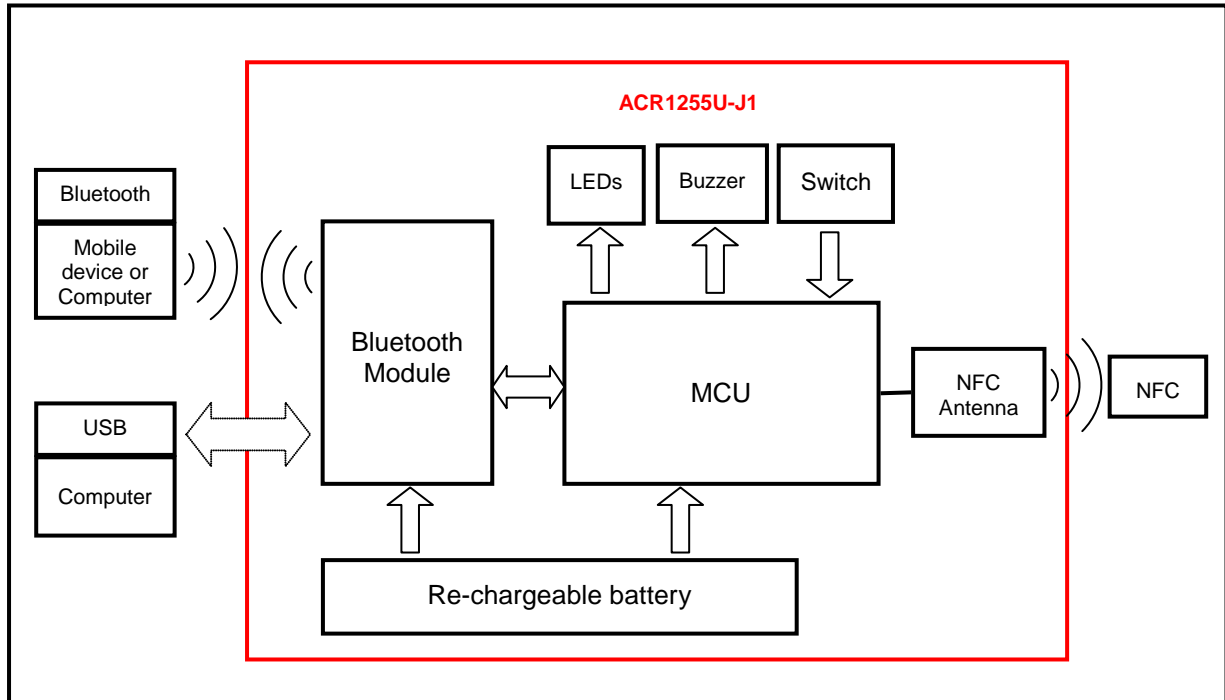


Figure 1: ACR1255U-J1 Architecture

For USB communication architecture, the protocol between ACR1255U-J1 and other devices follow the CCID protocol. All communications for NFC are PC/SC-compliant.

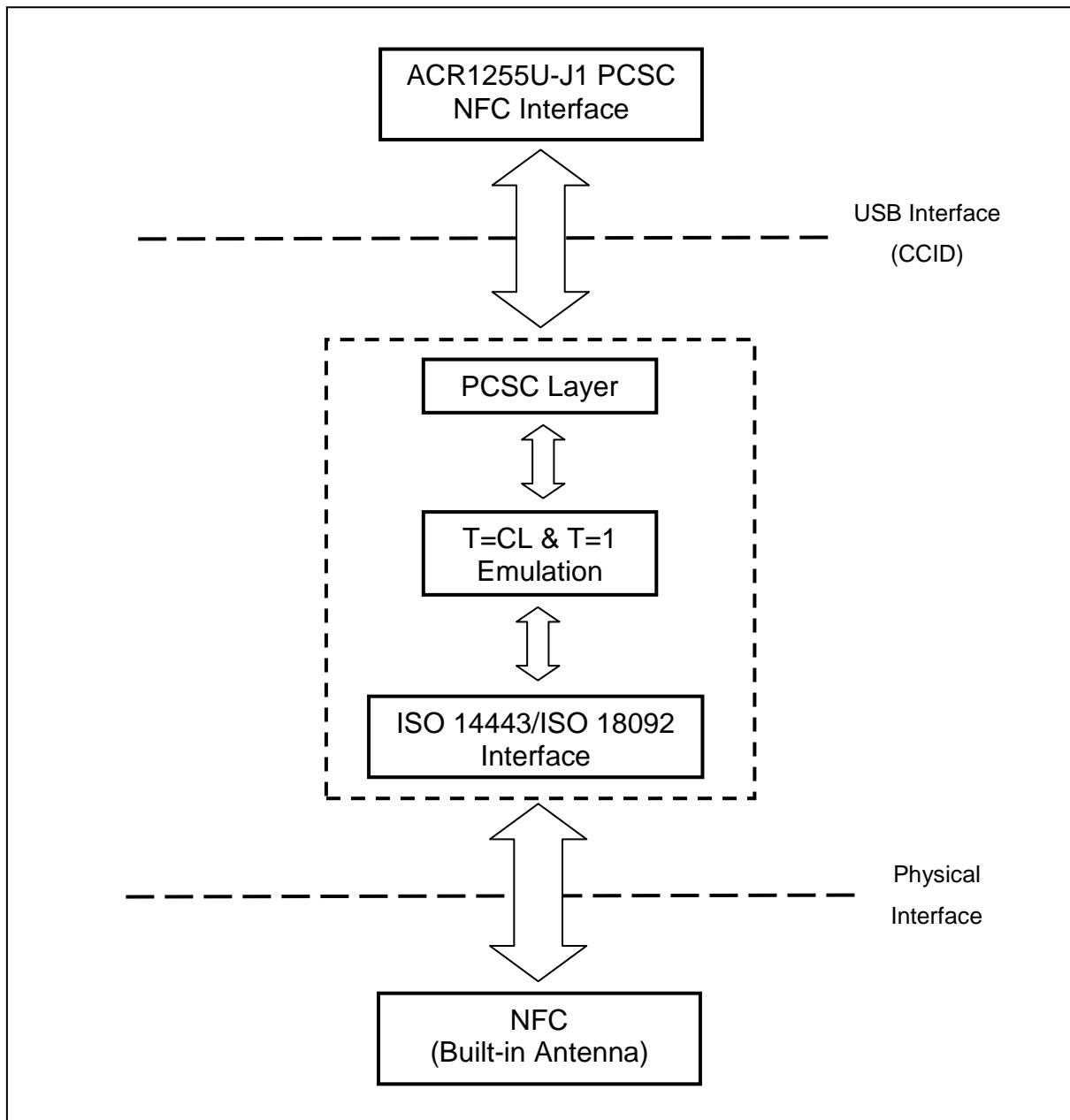


Figure 2: ACR1255U-J1 USB Communication Architecture



Bluetooth Smart protocol stack architecture is as follows:

GAP Peripheral Role Profile	SIM Access Profile	GAP Peripheral Bond Manager
GAP		GATT
SMP		ATT
		L2CAP
HCI		
Link Layer		
Physical Layer		

Figure 3: Bluetooth Smart Protocol Stack

4.0. Hardware Design

4.1. Battery

ACR1255U-J1 uses a rechargeable Lithium-ion battery, which has a capacity of 320 mAh.

4.1.1. Battery charging

Once the battery of ACR1255U-J1 runs out, it may be charged in any of the following modes: OFF, USB, Bluetooth; as long as it is connected to a power outlet.

4.1.2. Battery life

The battery life is dependent on the usage of the device. Below is an estimate of the battery life depending on the various work conditions:

Mode	Estimated Battery Life
Working Mode	10 hours
Standby Mode	7 days
OFF Mode	8 years

Table 2: Estimated Battery Lifespan

4.2. Bluetooth Interface

ACR1255U-J1 uses Bluetooth Smart as the medium to pair the device with computers and mobile devices.

4.3. USB Interface

The micro USB port is used to connect the ACR1255U-J1 to the computer as battery charging port. This port is also used in order for the ACR1255U-J1 to operate in PC-linked mode.

4.3.1. Communication Parameters

ACR1255U-J1 is connected to a computer through USB as specified in the USB Specification 2.0. The ACR1255U-J1 is working in full speed mode, i.e. 12 Mbps.

Pin	Signal	Function
1	V _{BUS}	+5 V power supply for the reader
2	D-	Differential signal transmits data between ACR1255U-J1 and PC
3	D+	Differential signal transmits data between ACR1255U-J1 and PC
4	GND	Reference voltage level for power supply

Table 3: USB Interface Wiring

4.3.2. Endpoints

ACR1255U-J1 uses the following endpoints to communicate with the host computer:

Control Endpoint	For setup and control purpose
Bulk OUT	For command to be sent from host to ACR1255U-J1 (data packet size is 64 bytes)
Bulk IN	For response to be sent from ACR1255U-J1 to host (data packet size is 64 bytes)
Interrupt IN	For card status message to send from ACR1255U-J1 to host (data packet size is 8 bytes)

4.4. NFC Interface

The interface between ACR1255U-J1 and contactless card follows the ISO 14443 or ISO 18092 specifications with certain restrictions or enhancements aimed to increase the practical functionality of the reader.

4.4.1. Carrier Frequency

The carrier frequency of ACR1255U-J1 is 13.56 MHz

4.4.2. Card Polling

ACR1255U-J1 automatically polls the contactless cards that are within the field. ISO 14443-4 Type A, ISO 14443 Type B, FeliCa, Topaz, MIFARE, and NFC tags are supported.

4.5. User Interface

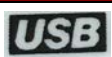


4.5.1. Keys

ACR1255U-J1 has 3 keys:

Key	Description
ANT_KEY	Wake up from sleep mode
MODEL_KEY	Mode selection
UPDATE_KEY	Enable boot loader

4.5.2. Mode Selection Switch

ACR1255U-J1 has three modes: USB, Off and Bluetooth. User can select one mode at a time as a data transmission interface.

Symbol	Switch	Active Mode
	USB	PC-linked
	Off	No power
	Bluetooth	Bluetooth

4.5.3. Status LED

ACR1255U-J1 has two bi-color LEDs to show the various operation statuses, where:

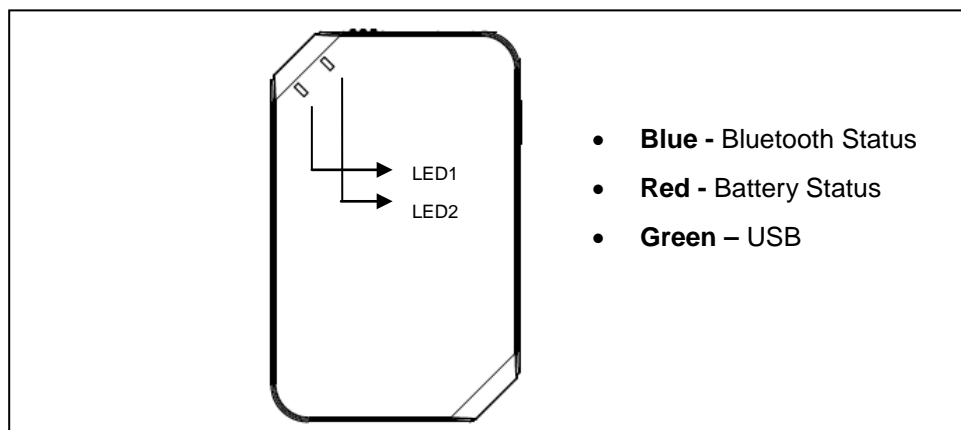


Figure 4: LED Operation Status

Mode	Color	LED Activity	Status
Bluetooth Mode	Blue (LED1)	Off	<ul style="list-style-type: none"> Reader is powered off No Bluetooth device paired Reader is in USB mode
		Slow flash	<ul style="list-style-type: none"> The paired Bluetooth device is authenticated successfully and is waiting for instructions
		Fast flash	<ul style="list-style-type: none"> Data is being transferred between the reader and paired device
		Fast – Slow flash	<ul style="list-style-type: none"> Searching for devices to be paired with
	Red (LED2)	On	<ul style="list-style-type: none"> Bluetooth device is paired with the reader Card is present
		Off	<ul style="list-style-type: none"> Battery is fully charged Reader is powered off The voltage of the battery is greater than 2.8 V and no USB powered is being supplied
		Slow flash	<ul style="list-style-type: none"> Low battery
USB mode	Green (LED2)	On	<ul style="list-style-type: none"> Battery is charging
		Off	<ul style="list-style-type: none"> Reader is powered off
		Slow flash	<ul style="list-style-type: none"> No card presented and the reader is waiting for instruction
		On	<ul style="list-style-type: none"> Card is present and the reader is waiting for instruction
	Red (LED1)	Fast flash	<ul style="list-style-type: none"> There is read/write access between the smart card and reader
		Off	<ul style="list-style-type: none"> Battery is fully charged Reader is powered off The voltage of the battery is greater than 2.8V and no USB powered is being supplied
		Slow flash	<ul style="list-style-type: none"> Low battery
		On	<ul style="list-style-type: none"> Battery is charging



4.5.4. Buzzer

ACR1255U-J1 has a buzzer that is used to notify user of card polling, Bluetooth connection, sleep mode, and low-battery status.

Buzzer Activity	Event
One short beep	Card polling
One long beep	Reader power on (Bluetooth mode)
Two long beeps	Low battery

5.0. Software Design

5.1. Bluetooth Connection Flow

The program flow of the Bluetooth connection is shown below:

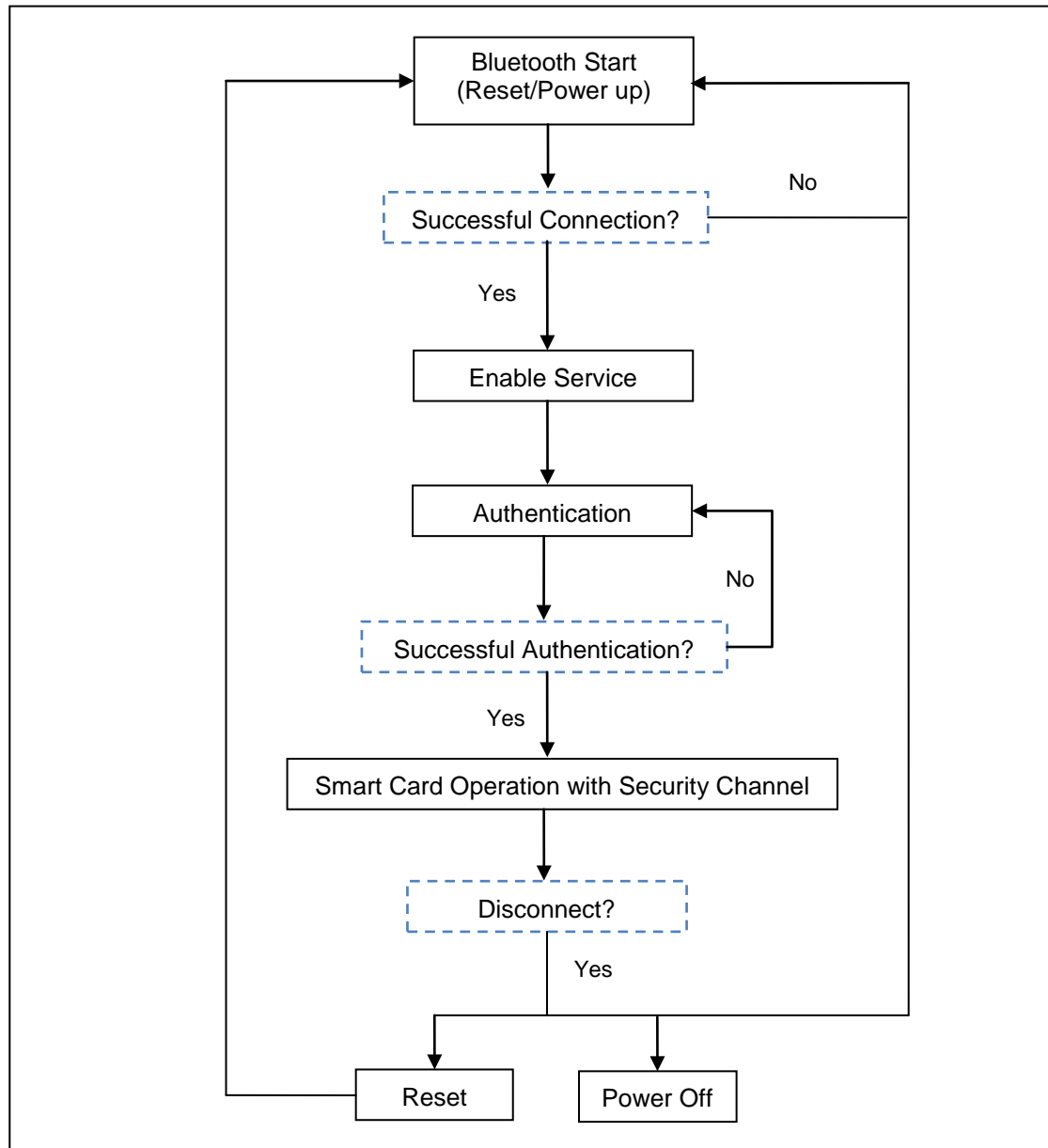


Figure 5: Bluetooth Connection Flow

5.2. Profile Selection

ACR1255U-J1 is a smart card reader that is designed to use Bluetooth technology as an interface to transmit data. A customized service called Commands Communication with three pipes is used: one pipe is used for command request, the second pipe is for command response/card notification, and third is RFU.

Also, the reader's current power consumption is significantly greater when the reader is operating in Bluetooth mode, hence, a standard battery service is used to notify the paired device about the current battery status. When there is a change in the battery status, the reader will notify the paired device through a specific pipe. To simplify, the battery levels are divided into three groups: sufficient battery (≥ 3.78 V), low battery (<3.78 V and ≥ 3.68 V), and no battery (<3.68 V).

Finally, to provide more reader information to the user, a customized Device Information service is added. This can only be read manually, or by an application request. The characteristics include Model Number, Serial Number, Firmware Revision, and Manufacturer Name.

Service	UUID	Pipe
Smart Card	FFF1	Commands Request
	FFF2	Commands Response/Card Notification
	FFF3	RFU
Battery	2A19	Battery Level
Device Information	2A23	System ID
	2A24	Model Number
	2A25	Serial Number
	2A26	Firmware Revision
	2A27	Hardware Revision
	2A29	Manufacturer Name

Table 4: ACR1255U-J1's Bluetooth Service

Attribute Name	UUID	Handle
DeviceName	2A00	03h
Send(Reader → Paired device)	8002	0Bh
Receive(Paired device → Reader)	8003	0Eh
CardStatus	8004	10h
BatteryLevel	2A19	14h
Manufacturer	2A29	18h
SerialNumber	2A25	21h
FW_Version	2A26	1Bh
ModelNumber	2A24	1Eh

Table 5: ACR1255U-J1 Service Handles and UUID Information List



5.3. Authentication

Before any sensitive data can be loaded into ACR1255U-J1, the data processing server must be authenticated by ACR1255U-J1 for the privilege to modify the secured data inside reader. In ACR1255U-J1, a mutual authentication method is being used.

An authentication request is always initiated by either the data processing server or the bridging device, which will then trigger ACR1255U-J1 to return a sequence of 16 bytes of random numbers (RND_A[0:15]). The random numbers are encrypted with the Customer Master Key currently stored in ACR1255U-J1 using the AES-128 CBC ciphering mode before being sent out from ACR1255U-J1. The bridging device must pass this sequence of encrypted random numbers to the data processing server, which will then undergo AES-128 CBC cipher mode decryption using the Customer Master Key that is being used in the data processing server (which should be the same as the one that is being used in ACR1255U-J1 and should be kept securely by the customer). The 16 bytes of decrypted random numbers from ACR1255U-J1 is then padded to the end of another 16 bytes of random numbers generated by the data processing server (RND_B[0:15]). The final sequence of 32 bytes of random numbers (RND_C[0:31]), that is:

$$\text{RND_C}[0:31] = \text{RND_B}[0:15] + \text{RND_A}[0:15],$$

will undergo decryption operation with the Customer Master Key being used in the server and the final output data is sent to ACR1255U-J1 through the bridging device using an authentication response message.

When ACR1255U-J1 receives the authentication response message, the message data will undergo an encryption operation using its own Customer Master Key and will be converted back to the normal 32 bytes of random numbers. In theory, the first 16 bytes of random numbers should be equal to RND_B[0:15] and are generated by the data processing server while the other 16 bytes should be equal to RND_A[0:15] and are originally generated by ACR1255U-J1.

ACR1255U-J1 will first compare if RND_A[0:15] is the same as the original version. If it is the same, then the data processing server is authenticated by ACR1255U-J1. ACR1255U-J1 will then encrypt RND_B[0:15] obtained using the Customer Master Key and the feedback to the data processing server through the bridging device using the answer to the authentication response message.

Upon receiving the answer to the authentication response message, the data processing server will decrypt the data contained in the message and check if the 16 bytes of random numbers are all equal to those originally generated RND_B[0:15]. If they are the same, then ACR1255U-J1 is authenticated by the server. At this moment, the whole authentication process is completed and sensitive data can be injected into ACR1255U-J1.

After successful authentication, a 16-byte Session Key is generated in both ACR1255U-J1 and the data processing server. The Session Key (SK[0:15]) is obtained by padding the first 8 bytes of RND_A at the end of the first 8 bytes of RND_B, that is:

$$\text{SK}[0:15] = \text{RND_B}[0:7] + \text{RND_A}[0:7]$$

All sensitive data leaving out of the Secured Data Processing Server must be encrypted with this Session Key using the AES-128 CBC ciphering mode. Thus, even if the encrypted data may be captured in the bridging mobile device, it is still very difficult to retrieve the original sensitive data without any prior knowledge of the Customer Master Key.

For better illustration, please refer to figure below (the picture below has omitted the bridging device for simplicity and better illustration):

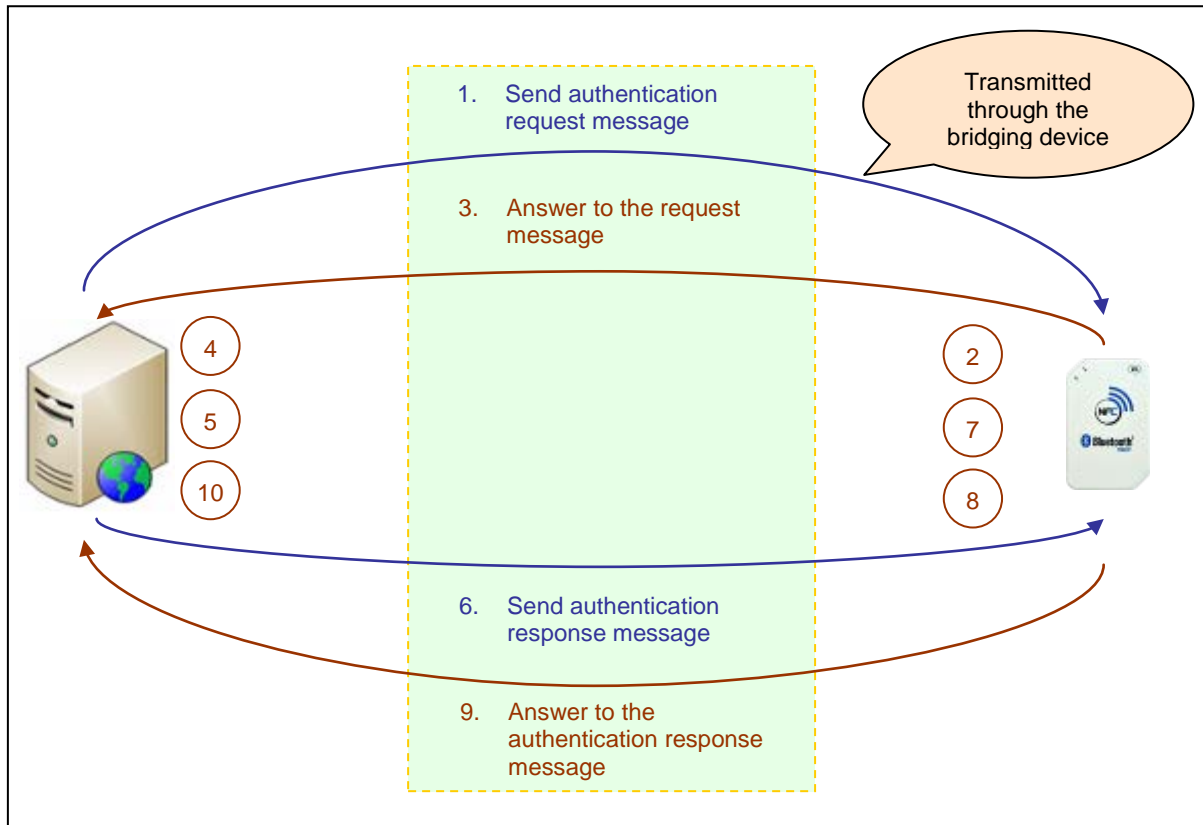


Figure 6: Authentication Procedure

Below is a summary of the above mentioned steps:

1. The data processing server/bridging device initiates an authentication request from ACR1255U-J1 by issuing an authentication request message.
2. Upon receiving the authentication request message, ACR1255U-J1 will generate 16 bytes of random numbers (RND_A[0:15]). The whole 16 bytes of data is encrypted with the Customer Master Key currently being used by ACR1255U-J1.
3. The encrypted version of RND_A[0:15] is then transferred to the data processing server through the answer to the authentication response message.
4. The data processing server will decrypt the data received to recover RND_A[0:15].
5. The data processing server will generate another 16 bytes of random numbers (RND_B[0:15]). RND_A[0:15] will be padded to the end of RND_B[0:15] to form a sequence of 32-byte random numbers (RND_C[0:31] = RND_B[0:15] + RND_A[0:15]). All the 32 bytes of random numbers will undergo a decryption process with the Customer Master Key currently being used in the server.
6. The final output data from the encryption process will be transferred to ACR1255U-J1 through the authentication response message.
7. In ACR1255U-J1, an encryption process will be performed on the received data to recover the 32 bytes of random number. ACR1255U-J1 will check the result RND_A[0:15] to see if they are the same as the original ones. If not, the authentication process will be terminated.
8. ACR1255U-J1 will encrypt the resultant RND_B[0:15] with the Customer Master Key. At the same time, a 16-byte Session Key is created by padding the first 8 bytes of RND_A to the end of the first 8 bytes of RND_B.



9. The encrypted RND_B[0:15] will be transferred to the data processing server through the authentication response message.
10. The data processing server will decrypt the message data and compare if the content is equal to the original RND_B[0:15]. If not, the authentication process will be terminated. Otherwise, the authentication process is completed and a 16-byte Session Key is created by padding the first 8 bytes of RND_A to the end of the first 8 bytes of RND_B.

Unidirectional authentication should be used to establish the binding between ACR1255U-J1 and the paired device. ACR1255U-J1 acts as a peripheral wherein it waits until the device to be paired with establishes a connection. The device to be paired with will need to send a PIN to complete the authentication process. By default, the PIN value is 000000.

5.4. Communication Profile

The communication profile should be:

Start byte + Len + Datablock + Check + Stop byte

Field	Size (bytes)	Description
Start byte	1	Value: 05h
Len	2	Len means the number of bytes in the Datablock field
Datablock	N	Data (the frame format is described in Section 5.5)
Check	1	Check means the XOR values of bytes in Len and Datablock fields
Stop byte	1	Value: 0Ah

5.5. Frame Format

5.5.1. Data Frame Format

Command

HID Frame	Length (bytes)	Description
<i>CmdMessageType</i>	1	Commands
<i>Length</i>	2	Data length
<i>Slot</i>	1	Default value: 00h
<i>Seq</i>	1	Sequence
<i>Param</i>	1	Parameter
<i>Check</i>	1	Checksum
<i>Data</i>	0-N	Data

The frame format should be:

CmdMessageType + Length + Slot + Seq + Param + Check + Data

If the total command length, including identifier, length and payload, is greater than 20 bytes, then the reader or the paired device will automatically divide it into several frames.



Response

HID Frame	Length (bytes)	Description
RspMessageType	1	Commands
Length	2	Data length
Slot	1	Default value: 00h
Seq	1	Sequence
Slot Status/Param	1	Slot Status/Parameter
Checksum	1	Checksum
Data	0-N	Data

Data checksum is used in detecting errors that may have been introduced during wireless data transmission. To calculate the data checksum: XOR { RspMessageType, Length, Slot, Seq, SlotStatus/Param, Data }.

Example: 62010063 => Checksum = 63h

5.6. Bluetooth Communication Protocol

ACR1255U-J1 communicates to the paired device using the Bluetooth interface with a predefined protocol. The protocol is similar to the formats of the CCID Command Pipe and Response Pipe.

Command	Mode Supported	Sender	Description
62h	Authenticated	Paired device	PICC Power On
63h	Authenticated	Paired device	PICC Power Off
65h	Authenticated	Paired device	Get Card Status
6Fh	Authenticated	Paired device	Exchange APDU
6Bh	Authenticated	Paired device	Escape Commands

Table 6: Command Code Summary

Command	Mode Supported	Sender	Description
80h	Authenticated	Reader	Response to Data Block
81h	Authenticated	Reader	Response to Slot Status
83h	Authenticated	Reader	Response to Escape Command
50h	Authenticated	Reader	Notify Card Status
51h	Authenticated	Reader	Response to Hardware Error

Table 7: Response Code Summary



5.6.1. Card Power On

This command is used to send a power on request to the reader.

Command Format

Offset	Field	Size	Value	Description
0	<i>CmdMessageType</i>	1	62h	
1	<i>Length</i>	2	0000h	Data length
3	<i>Slot</i>	1	00h	
4	<i>Seq</i>	1	00h	Sequence
5	<i>Param</i>	1	00h	Parameter
6	<i>Check</i>	1	00h	Check means the XOR values of all bytes in the command
7	<i>Data</i>	N		Data

Response Data Format

Offset	Field	Size	Value	Description
0	<i>CmdMessageType</i>	1	80h	
1	<i>Length</i>	2	0000h	Data length
3	<i>Slot</i>	1	00h	
4	<i>Seq</i>	1	00h	Sequence
5	<i>Param</i>	1	00h	Parameter
6	<i>Check</i>	1	00h	Check means the XOR values of all bytes in the command
7	<i>Data</i>	N		Data

Example:

Response = 80 00 08 00 00 00 3B 3B 83 80 01 41 07 00 44

ATR = 3B 83 80 01 41 07 00 44



5.6.2. Card Power Off

This command is used to send a power off request to the reader.

Command Format

Offset	Field	Size	Value	Description
0	<i>CmdMessageType</i>	1	63h	
1	<i>Length</i>	2	0000h	Data length
3	<i>Slot</i>	1	00h	
4	<i>Seq</i>	1	00h	Sequence
5	<i>Param</i>	1	00h	Parameter
6	<i>Check</i>	1	00h	Check means the XOR values of all bytes in the command
7	<i>Data</i>	N		Data

Response Data Format

Offset	Field	Size	Value	Description
0	<i>CmdMessageType</i>	1	81h	
1	<i>Length</i>	2	0000h	Data length
3	<i>Slot</i>	1	00h	
4	<i>Seq</i>	1	00h	Sequence
5	<i>Param</i>	1	00h	Parameter
6	<i>Check</i>	1	00h	Check means the XOR values of all bytes in the command
7	<i>Data</i>	N		Data



5.6.3. Get Slot Status

This command is used to check the presence of the inserted card.

Command Format

Offset	Field	Size	Value	Description
0	<i>CmdMessageType</i>	1	65h	
1	<i>Length</i>	2	0000h	Data length
3	<i>Slot</i>	1	00h	
4	<i>Seq</i>	1	00h	Sequence
5	<i>Param</i>	1	00h	Parameter
6	<i>Check</i>	1	00h	Check means the XOR values of all bytes in the command
7	<i>Data</i>	N		Data

Response Data Format

Offset	Field	Size	Value	Description
0	<i>CmdMessageType</i>	1	81h	
1	<i>Length</i>	2	0000h	Data length
3	<i>Slot</i>	1	00h	
4	<i>Seq</i>	1	00h	Sequence
5	<i>Param</i>	1		Card Status: 00h = Card present and active 01h = Card present but inactive 02h = Card absent
6	<i>Check</i>	1	00h	Check means the XOR values of all bytes in the command
7	<i>Data</i>	N		Data



5.6.4. APDU Command

This command is used to send an APDU command to the reader.

Command Format

Offset	Field	Size	Value	Description
0	<i>CmdMessageType</i>	1	6Fh	
1	<i>Length</i>	2		Data length
3	<i>Slot</i>	1	00h	
4	<i>Seq</i>	1	00h	Sequence
5	<i>Param</i>	1	00h	Parameter
6	<i>Check</i>	1		Check means the XOR values of all bytes in the command
7	<i>Data</i>	N		Data

Response Data Format

Offset	Field	Size	Value	Description
0	<i>CmdMessageType</i>	1	80h	
1	<i>Length</i>	2		Data length
3	<i>Slot</i>	1	00h	
4	<i>Seq</i>	1	00h	Sequence
5	<i>Param</i>	1	00h	Parameter
6	<i>Check</i>	1		Check means the XOR values of all bytes in the command
7	<i>Data</i>	N		Data



5.6.5. Notify Card Status Command

This command is used to check on the card status.

Offset	Field	Size	Value	Description
0	<i>CmdMessageType</i>	1	50h	
1	<i>Length</i>	2		Data length
3	<i>Slot</i>	1	00h	
4	<i>Seq</i>	1	00h	Sequence
5	<i>Param</i>	1		Status: 02h = Card present 03h = Card absent
6	<i>Check</i>	1	00h	Check means the XOR values of all bytes in the command
7	<i>Data</i>	N		Data



5.6.6. Hardware Error Response

This command is used to check on the hardware error status.

Response Data Format

Offset	Field	Size	Value	Description
0	<i>CmdMessageType</i>	1	51h	
1	<i>Length</i>	2		Data length
3	<i>Slot</i>	1	00h	
4	<i>Seq</i>	1	00h	Sequence
5	<i>Param</i>	1		Status: 01h = Checksum error 02h = Timeout 03h = Command error 04h = Unauthorized 05h = Undefined error 06h = Receive data error
6	<i>Check</i>	1		Check means the XOR values of all bytes in the command
7	<i>Data</i>	N		Data

5.6.7. Escape Command

This command is used to access the extended features of the reader.

5.6.7.1. For PC-linked Mode

Command Format

Offset	Field	Size	Value	Description
0	CmdMessageType	1	6Bh	
1	Length	2		Data length
3	Slot	1	00h	
4	Seq	1	00h	Sequence
5	Param	1	00h	Parameter
6	Check	1		Check means the XOR values of all bytes in the command
7	Data	N		Data

Response Data Format

Offset	Field	Size	Value	Description
0	CmdMessageType	1	83h	
1	Length	2		Data length
3	Slot	1	00h	
4	Seq	1	00h	Sequence
5	Param	1	00h	Parameter
6	Check	1		Check means the XOR values of all bytes in the command
7	Data	N		Data

5.6.7.2. For Bluetooth Mode

1. Open automatic polling

Request command: E0 00 00 40 01

Response command: E1 00 00 00 01

Example:

Request: 6B 00 05 00 00 00 A1 E0 00 00 40 01

Response: 83 00 05 00 00 00 E1 E1 00 00 00 01

2. Close automatic polling

Request command: E0 00 00 40 00

Response command: E1 00 00 00 00

Note: To use a new PIN, the device should rebooted.

Example:

Request: 6B 00 05 00 00 00 A1 E0 00 00 40 00

Response: 83 00 05 00 00 00 E1 E1 00 00 00 00



3. Change PIN of ACR1255U-J1

Request command: **E0 00 00 E0 11 + PINKey(3bytes)**

Where: PINKey = 3 bytes, in decimal

e.g., PINKey in ASCII: 010203

PINKey in decimal: 01 02 03

Response command: **E1 00 00 E0 11**

Example:

Request to change PIN to 000001: 6B 00 08 00 00 00 10 **E0 00 00 E0 11 00 00 01**

Response: 83 00 05 00 00 02 10 **E1 00 00 E0 11**

4. Authentication request

Request command: **E0 00 00 45 00**

Response command: **E1 00 00 45 00 + Data(16 bytes)**

Where: Data = 16 bytes of random numbers

Example:

Request (SPH_to_RDR_ReqAuth): 6B 00 05 00 00 00 CB **E0 00 00 45 00**

Response (RDR_to_SPH_AuthRsp1): 83 00 15 00 00 02 2C **E1 00 00 45 00 77 59 E8 62 B7 80 0D 0A CE 9A 03 9B E9 48 EF 05**

5. Authentication response

Request command: **E0 00 00 46 00 + Data (32 bytes)**

Response command: **E1 00 00 46 00/01 + Data(16 bytes)**

Example:

Request (SPH_to_RDR_AuthRsp): 6B 00 25 00 00 00 EA **E0 00 00 46 00 A6 81 17 91 9F 46 07 AE AE 4E 94 8E 05 14 E8 C8 25 F7 90 05 76 F8 DE 7D 6D ED 55 3F 80 10 C2 CA**

Response (RDR_to_SPH_AuthRsp2): 83 00 15 00 00 02 53 **E1 00 00 46 00 47 D5 50 54 F3 49 D4 17 B1 65 40 21 9B DA C9 B2**

5.7. Mutual Authentication and Encryption Protocol

In Bluetooth mode, the communication protocol in **Section 5.6** will be encrypted and transmitted after a successful mutual authentication.

5.7.1. Bluetooth Authentication Program Flow

As illustrated in **Section 5.3**, mutual authentication is being used to avoid man-in-the-middle attack. The summary of the commands used in mutual authentication is in the following table:

Sequence	Command	Mode Supported	Sender	Description
1	6Bh	Connected	Paired device	SPH_to_RDR_ReqAuth
2	83h	Connected	Reader	RDR_to_SPH_AuthRsp1
3	6Bh	Connected	Paired device	SPH_to_RDR_AuthRsp
4	83h	Connected	Reader	RDR_to_SPH_AuthRsp2

Table 8: Summary of Mutual Authentication Commands

The 16-byte Session Key, SK[0:15], is generated in both ACR1255U-J1 and the data processing server. It is obtained by padding the first 8 bytes of RND_B at the end of the first 8 bytes of RND_A, i.e.

$$SK[0:15] = RND_A[0:7] + RND_B[0:7]$$

5.7.2. SPH_to_RDR_ReqAuth

This command will request ACR1255U-J1 to perform authentication with the paired key-generating device.

For more information on the authentication process, please refer to **Section 5.3**.

Offset	Field	Size	Value	Description	Encrypted
0	<i>bMessageType</i>	1	6Bh		No
1	<i>LEN1 LEN2 (wLength)</i>	2	0005h	The Number of extra bytes in Data field, and is expressed in two bytes long, and LEN1 is MSB while LEN2 is LSB;	
3	<i>Slot Number</i>	1	00h		
4	<i>Sequence</i>	1	00h		
5	<i>Parameter</i>	1	00h	Slot Status	
6	<i>wChecksum</i>	1	CBh	CSUM means the XOR values of all bytes in the command	No
7	<i>Data</i>	5	E0 00 00 45 00h		

The response to this message is RDR_to_SPH_AuthRsp1 if the received command message is error-free. Otherwise, the response message will be RDR_to_SPH_ACK to provide the error information.



5.7.3. RDR_to_SPH_AuthRsp1

This command is sent by ACR1255U-J1 in response to the SPH_to_RDR_ReqAuth.

Offset	Field	Size	Value	Description	Encrypted
0	<i>bMessageType</i>	1	83h		No
1	<i>LEN1 LEN2</i> (<i>wLength</i>)	2	0015h	Number of extra bytes in abRndNum field, and is expressed in two bytes long, and LEN1 is MSB while LEN2 is LSB;	No
3	<i>Slot Number</i>	1	00h		No
4	<i>Sequence</i>	1	00h		No
5	<i>Parameter</i>	1	00h	Slot Status	
6	<i>wChecksum</i>	1		wChecksum means the XOR values of all bytes in the command	No
7	<i>abRndNum</i>	21	E1 00 00 45 00 + 16 bytes random number	abRndNum[0:15] – 16 bytes of random number All the 16-byte data must be encrypted with the Customer Master Key currently stored in ACR1255U-J1. “E1 00 00 45 00” does not need to be encrypted.	Yes



5.7.4. SPH_to_RDR_AuthRsp

This command is the second phase of the authentication process. After the device has initiated the SPH_to_RDR_ReqAuth command to the ACR1255U-J1, the reader will then provide an RDR_to_SPH_AuthRsp1 message if there's no error.

The RDR_to_SPH_AuthRsp1 will contain a sequence of 16-byte random numbers encrypted using the Customer Master Key. The paired key-generating device should decrypt it using the correct Customer Master Key and pads it to the end of the 16-byte of random numbers. The overall 32-byte random numbers will be decrypted using the Customer Master Key and returned to ACR1255U-J1 using this command in order to have a successful authentication.

Offset	Field	Size	Value	Description	Encrypted
0	<i>bMessageType</i>	1	6Bh		No
1	<i>LEN1 LEN2</i> (<i>wLength</i>)	2	0025h	The Number of extra bytes in the abAuthData field, and is expressed in two bytes long, and LEN1 is MSB while LEN2 is LSB;	No
3	<i>Slot Number</i>	1	00h		No
4	<i>Sequence</i>	1	00h		No
5	<i>Parameter</i>	1	00h	Slot Status	No
6	<i>wChecksum</i>	1		wChecksum means the XOR values of all bytes in the command	No
7	<i>abAuthData</i>	37	E0 00 00 46 00 + 32 bytes random number	abAuthData[0:15] – 16 bytes of random number generated by the data processing server abAuthData[16:31] – 16 bytes of decrypted random number received from ACR1255U-J1 All the 32 bytes of data underwent a decryption process with the Customer Master Key using AES128 CBC cipher mode. “E0 00 00 46 00” does not need to be decrypted.	Yes

The response to this message is RDR_to_SPH_AuthRsp2 if the command message received is error-free and the random number generated returned by paired device is correct. Otherwise, the response message will be RDR_to_SPH_ACK to provide the error information.



5.7.5. RDR_to_SPH_AuthRsp2

This command is sent by ACR1255U-J1 in response to the SPH_to_RDR_AuthRsp.

Offset	Field	Size	Value	Description	Encrypted
0	<i>bMessageType</i>	1	83h		No
1	<i>LEN1 LEN2</i> (<i>wLength</i>)	2	0015h	Number of extra bytes in <i>abRndNum</i> field, and is expressed in two bytes long, and <i>LEN1</i> is MSB while <i>LEN2</i> is LSB;	No
3	<i>Slot Number</i>	1	00h		No
4	<i>Sequence</i>	1	00h		No
5	<i>Parameter</i>	1	00h	Slot Status	No
6	<i>wChecksum</i>	1			No
20	<i>abRndNum</i>	21	E1 00 00 46 00 + 16 bytes random number	<i>abRndNum</i> [0:15] – 16 bytes of random number retrieved from the data processing server All the 16-byte data must be encrypted with the Customer Master Key that is currently stored in ACR1255U-J1 “E1 00 00 46 00” does not need to be encrypted.	Yes

5.7.6. RDR_to_SPH_ACK (Error handling)

This is an error handling acknowledgement message sent by the ACR1255U-J1 to the paired device to recognize the acceptance of some command messages. During the communication, any indicated error message will be transmitted using the RDR_to_SPH_ACK. The command is not encrypted.

Command	Mode supported	Sender	Description
51h	Connected, Authenticated	Reader	RDR_to_SPH_ACK (Error handling)

This message will also include the error code whenever appropriate.

Offset	Field	Size	Value	Description	Encrypted
0	<i>bMessageType</i>	1	51h	Error handling response header	No
1	<i>LEN1LEN2 (wLength)</i>	2	00h		
3	<i>Slot Number</i>	1	00h		
4	<i>Sequence</i>	1	00h		
3	<i>bErrorCode</i>	1		Specify the error code for the previously processed command message. The possible error codes are listed in as the table below.	
4	<i>wChecksum</i>	1		CSUM means the XOR values of all bytes in the command	

Field	Value	Description
<i>bErrorCode</i>	01h	Checksum error
	02h	Timeout
	03h	Command error
	04h	Unauthorized
	05h	Undefined error
	06h	Received data error

Table 9: Mutual Authentication Error codes



6.0. Host Programming (PC-linked) API

6.1. PC/SC API

This section describes some of the PC/SC API for application programming usage. For more details, please refer to Microsoft MSDN Library or PC/SC workgroup.

6.1.1. SCardEstablishContext

The *SCardEstablishContext* function establishes the resource manager context within which database operations are performed.

Refer to: <http://msdn.microsoft.com/en-us/library/windows/desktop/aa379479%28v=vs.85%29.aspx>

6.1.2. SCardListReaders

The *SCardListReaders* function provides the list of readers within a set of named reader groups, eliminating duplicates.

The caller supplies a list of reader groups, and receives the list of readers within the named groups. Unrecognized group names are ignored. This function only returns readers within the named groups that are currently attached to the system and available for use.

Refer to: <http://msdn.microsoft.com/en-us/library/windows/desktop/aa379793%28v=vs.85%29.aspx>

6.1.3. SCardConnect

The *SCardConnect* function establishes a connection (using a specific resource manager context) between the calling application and a smart card contained by a specific reader. If no card exists in the specified reader, an error is returned.

Refer to: <http://msdn.microsoft.com/en-us/library/windows/desktop/aa379473%28v=vs.85%29.aspx>

6.1.4. SCardControl

The *SCardControl* function gives you direct control of the reader. You can call it any time after a successful call to *SCardConnect* and before a successful call to *SCardDisconnect*. The effect on the state of the reader depends on the control code.

Refer to: <http://msdn.microsoft.com/en-us/library/windows/desktop/aa379474%28v=vs.85%29.aspx>

Note: Commands from **Section 6.66.6 – Peripherals Control** are using this API for sending.

6.1.5. ScardTransmit

The *SCardTransmit* function sends a service request to the smart card and expects to receive data back from the card.

Refer to: <http://msdn.microsoft.com/en-us/library/windows/desktop/aa379804%28v=vs.85%29.aspx>

Note: APDU Commands (i.e. the commands sent to connected card and **Section 6.3 – Pseudo APDU for Contactless**) are using this API for sending.

6.1.6. ScardDisconnect

The **SCardDisconnect** function terminates a connection previously opened between the calling application and a smart card in the target reader.

Refer to: <http://msdn.microsoft.com/en-us/library/windows/desktop/aa379475%28v=vs.85%29.aspx>

6.1.7. APDU Flow

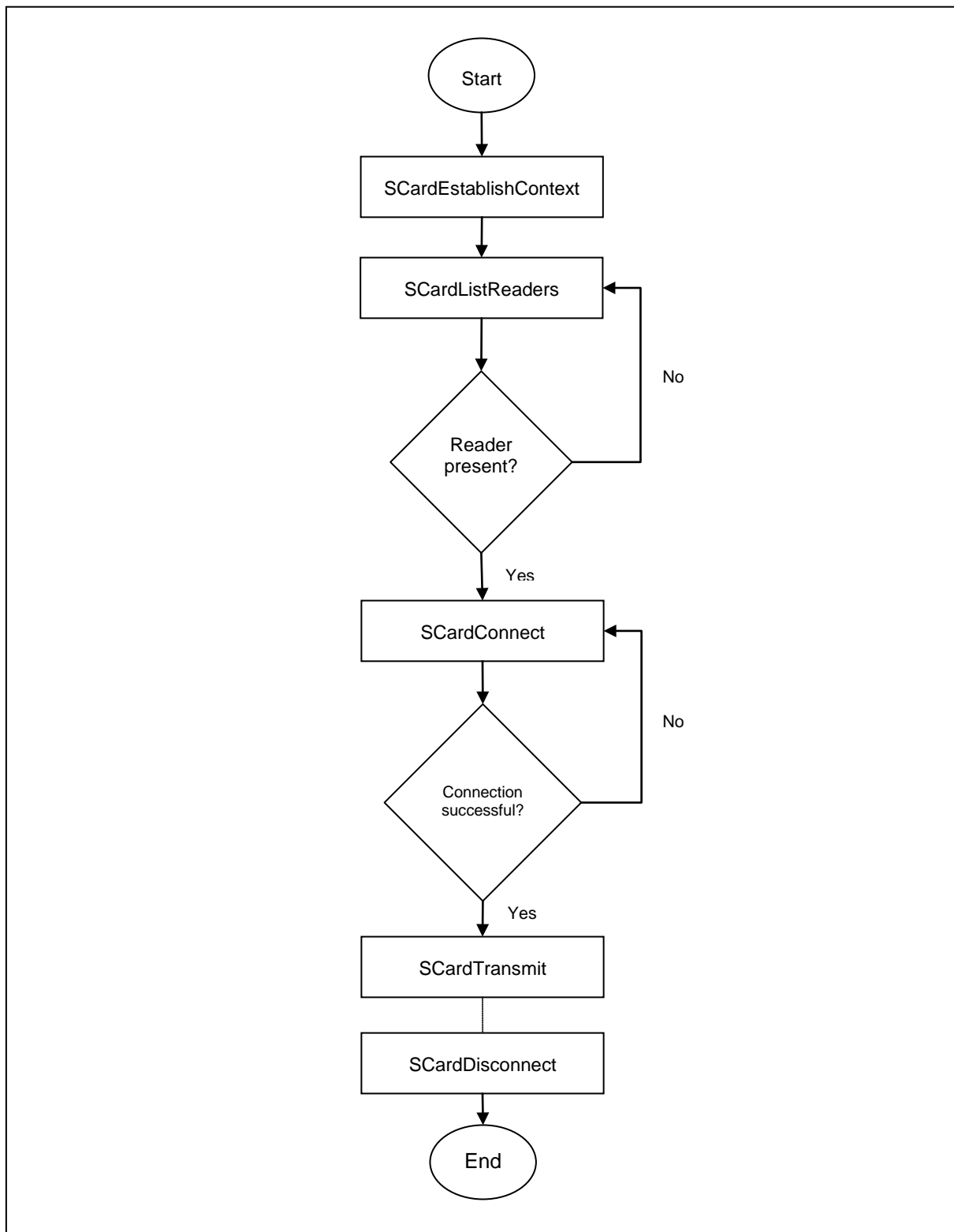


Figure 7: ACR1255U-J1 APDU Flow

6.1.8. Escape Command Flow

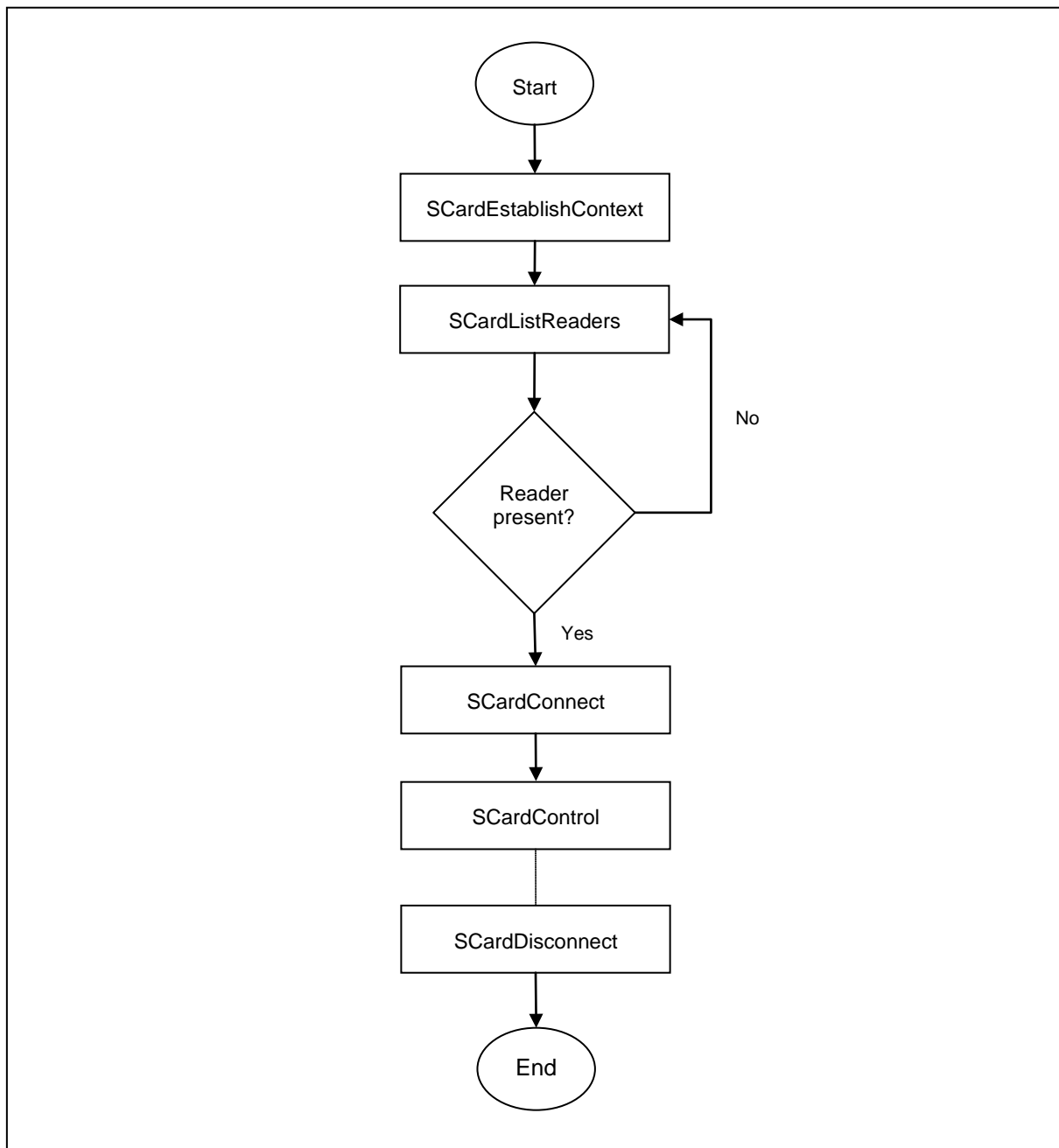


Figure 8: ACR1255U-J1 Escape Command Flow

6.2. Contactless Smart Card Protocol

6.2.1. ATR Generation

If the reader detects a PICC, an ATR will be sent to the PCSC driver for identifying the PICC.

6.2.1.1. ATR Format for ISO 14443 Part 3 PICCs

Byte	Value	Designation	Description
0	3Bh	Initial Header	
1	8Nh	T0	Higher nibble 8 means: no TA1, TB1, TC1 only TD1 is following. Lower nibble N is the number of historical bytes (HistByte 0 to HistByte N-1)
2	80h	TD1	Higher nibble 8 means: no TA2, TB2, TC2 only TD2 is following. Lower nibble 0 means T = 0
3	01h	TD2	Higher nibble 0 means no TA3, TB3, TC3, TD3 following. Lower nibble 1 means T = 1
4 to 3+N	80h	T1	Category indicator byte, 80 means A status indicator may be present in an optional COMPACT-TLV data object.
	4Fh	Tk	Application identifier Presence Indicator.
	0Ch		Length
	RID		Registered Application Provider Identifier (RID) # A0 00 00 03 06
	SS		Byte for standard.
	C0 .. C1h		Bytes for card name.
	00 00 00 00h	RFU	RFU # 00 00 00 00
4+N	UU	TCK	Exclusive-oring of all the bytes T0 to Tk

Example:

ATR for MIFARE® Classic 1K = {3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 01 00 00 00 00 6Ah}

Where:

Length (YY) = 0Ch
RID = A0 00 00 03 06h (PC/SC Workgroup)
Standard (SS) = 03h (ISO 14443A, Part 3)
Card Name (C0 .. C1) = [00 01h] (MIFARE Classic 1K)

Standard (SS) = 03h: ISO 14443A, Part 3
 = 11h: FeliCa



Card Name (C0 .. C1)	00 01: MIFARE Classic 1K	00 30: Topaz and Jewel
	00 02: MIFARE Classic 4K	00 3B: FeliCa
	00 03: MIFARE Ultralight®	FF 28: JCOP 30
	00 26: MIFARE Mini	FF [SAK]: undefined tags

6.2.1.2. ATR Format for ISO 14443 Part 4 PICCs

Byte	Value	Designation	Description						
0	3Bh	Initial Header							
1	8N	T0	Higher nibble 8 means: no TA1, TB1, TC1 only TD1 is following. Lower nibble N is the number of historical bytes (HistByte 0 to HistByte N-1)						
2	80h	TD1	Higher nibble 8 means: no TA2, TB2, TC2 only TD2 is following. Lower nibble 0 means T = 0						
3	01h	TD2	Higher nibble 0 means no TA3, TB3, TC3, TD3 following. Lower nibble 1 means T = 1						
4 to 3 + N	XX	T1	Historical Bytes: ISO 14443-A: The historical bytes from ATS response. Refer to the ISO 14443-4 specification. ISO 14443-B: <table><tr><td>Byte1-4</td><td>Byte5-7</td><td>Byte8</td></tr><tr><td>Application Data from ATQB</td><td>Protocol Info Byte from ATQB</td><td>Higher nibble=MBLI from ATTRIB command Lower nibble (RFU)=0</td></tr></table>	Byte1-4	Byte5-7	Byte8	Application Data from ATQB	Protocol Info Byte from ATQB	Higher nibble=MBLI from ATTRIB command Lower nibble (RFU)=0
	Byte1-4	Byte5-7		Byte8					
	Application Data from ATQB	Protocol Info Byte from ATQB		Higher nibble=MBLI from ATTRIB command Lower nibble (RFU)=0					
	XX XX XX	Tk							
4+N	UU	TCK	Exclusive-oring of all the bytes T0 to Tk						

Example 1:

ATR for MIFARE® DESFire® = {3B 81 80 01 80 80h} // 6 bytes of ATR

Note: Use the APDU "FF CA 01 00 00h" to distinguish the ISO 14443A-4 and ISO 14443B-4 PICCs, and retrieve the full ATS if available. ISO 14443A-3 or ISO 14443B-3/4 PICCs do have ATS returned.

APDU Command = FF CA 01 00 00h

APDU Response = 06 75 77 81 02 80 90 00h

ATS = {06 75 77 81 02 80h}



Example 2: ATR for EZ-link = {3B 88 80 01 1C 2D 94 11 F7 71 85 00 BEh}

Application Data of ATQB = 1C 2D 94 11h

Protocol Information of ATQB = F7 71 85h

MBLI of ATTRIB = 00h

6.3. Pseudo APDU for Contactless Interface

6.3.1. Get Data

This command returns the serial number or ATS of the “connected PICC”.

Get UID APDU Format (5 bytes)

Command	Class	INS	P1	P2	Le
Get Data	FFh	CAh	00h 01h	00h	00h (Max. Length)

If P1 = 00h, Get UID Response Format (UID + 2 bytes)

Response	Data Out					
Result	UID (LSB)	UID (MSB)	SW1	SW2

If P1 = 01h, Get ATS of a ISO 14443 A card (ATS + 2 bytes)

Response	Data Out		
Result	ATS	SW1	SW2

Response Codes

Results	SW1	SW2	Meaning
Success	90h	00h	The operation is completed successfully.
Warning	62h	82h	End of UID/ATS reached before Le bytes (Le is greater than UID Length).
Error	6Ch	XXh	Wrong length (wrong number Le: ‘XX’ encodes the exact number) if Le is less than the available UID length.
Error	63h	00h	The operation is failed.
Error	6Ah	81h	Function not supported

Examples:

To get the serial number of the “connected PICC”:

```
UINT8 GET_UID[5] = {FF, CA, 00, 00, 00};
```

To get the ATS of the “connected ISO 14443 A PICC”:

```
UINT8 GET_ATS[5] = {FF, CA, 01, 00, 00};
```

6.4. PICC Commands for MIFARE® Classic (1K/4K) memory cards

6.4.1. Load Authentication Keys

This command loads the authentication keys to the reader. The authentication keys are used to authenticate the particular sector of the MIFARE Classic (1K/4K) memory card. Two kinds of authentication key locations are provided: volatile and non-volatile key locations.

Load Authentication Keys APDU Format (11 bytes)

Command	Class	INS	P1	P2	Lc	Data In
Load Authentication Keys	FFh	82h	Key Structure	Key Number	06h	Key (6 bytes)

Where:

Key Structure 1 byte.

00h = Key is loaded into the reader volatile memory.

Other = Reserved.

Key Number 1 byte.

00h – 01h = Non-volatile memory for storing keys. The keys are permanently stored in the reader and will be retained in the reader's memory even if the reader is disconnected from the PC. It can store up to 32 keys inside the reader non-volatile memory.

Note: The default value is FF FF FF FF FF FFh.

Key 6 bytes.

The key value loaded into the reader. Example: FF FF FF FF FF FFh

Load Authentication Keys Response Format (2 bytes)

Response	Data Out	
Result	SW1	SW2

Load Authentication Keys Response Codes

Results	SW1	SW2	Meaning
Success	90h	00h	The operation is completed successfully.
Error	63h	00h	The operation is failed.

Example:

// Load a key {FF FF FF FF FF FFh} into the volatile memory location 00h.

APDU = {FF 82 00 00 06 FF FF FF FF FF FFh}

6.4.2. Authentication for MIFARE® Classic (1K/4K)

This command uses the keys stored in the reader to do authentication with the MIFARE Classic (1K/4K) card (PICC). Two types of authentication keys are used: TYPE_A and TYPE_B.

Load Authentication Keys APDU Format (6 bytes) [Obsolete]

Command	Class	INS	P1	P2	P3	Data In
Authentication	FFh	88h	00h	Block Number	Key Type	Key Number

Load Authentication Keys APDU Format (10 bytes)

Command	Class	INS	P1	P2	Lc	Data In
Authentication	FFh	86h	00h	00h	05h	Authenticate Data Bytes

Authenticate Data Bytes (5 bytes)

Byte1	Byte 2	Byte 3	Byte 4	Byte 5
Version 01h	00h	Block Number	Key Type	Key Number

Where:

Block Number 1 byte. The memory block to be authenticated.

For MIFARE Classic 1K card, it has a total of 16 sectors and each sector consists of four consecutive blocks (e.g., Sector 00h consists of blocks {00h, 01h, 02h and 03h}; sector 01h consists of blocks {04h, 05h, 06h and 07h}; the last sector 0Fh consists of blocks {3Ch, 3Dh, 3Eh and 3Fh}. Once the authentication is done successfully, there is no need to do the authentication again provided that the blocks to be accessed belong to the same sector. Please refer to the MIFARE Classic (1K/4K) specification for more details.

Note: Once the block is authenticated successfully, all the blocks belonging to the same sector are accessible.

Key Type 1 byte.

60h = Key is used as a TYPE A key for authentication.

61h = Key is used as a TYPE B key for authentication.

Key Number 1 byte.

00 ~ 01h = Volatile memory for storing keys. The keys will disappear when the reader is disconnected from the PC. Two volatile keys are provided. The volatile key can be used as a session key for different sessions.

Load Authentication Keys Response Format (2 bytes)

Response	Data Out	
Result	SW1	SW2



Load Authentication Keys Response Codes

Results	SW1	SW2	Meaning
Success	90	00h	The operation is completed successfully.
Error	63	00h	The operation is failed.

Examples:

//Authenticate the Block 04h with a {TYPE A, key number 00h}. For PC/SC V2.01, Obsolete.

APDU = {FF 88 00 04 60 00h};

//Authenticate the Block 04h with a {TYPE A, key number 00h}. For PC/SC V2.07

APDU = {FF 86 00 00 05 01 00 04 60 00h}

Sectors (Total 16 sectors. Each sector consists of 4 consecutive blocks)	Data Blocks (3 blocks, 16 bytes per block)	Trailer Block (1 block, 16 bytes)	
Sector 0	00h – 02h	03h	} 1 KB
Sector 1	04h – 06h	07h	
..	
..	
Sector 14	38h – 0Ah	3Bh	
Sector 15	3Ch – 3Eh	3Fh	

Table 10: MIFARE® Classic 1K Memory Map

Sectors (Total 32 sectors. Each sector consists of 4 consecutive blocks)	Data Blocks (3 blocks, 16 bytes per block)	Trailer Block (1 block, 16 bytes)	
Sector 0	00h ~ 02h	03h	} 2 KB
Sector 1	04h ~ 06h	07h	
..			
..			
Sector 30	78h ~ 7Ah	7Bh	
Sector 31	7Ch ~ 7Eh	7Fh	

Sectors (Total 8 sectors. Each sector consists of 16 consecutive blocks)	Data Blocks (15 blocks, 16 bytes per block)	Trailer Block (1 block, 16 bytes)	
Sector 32	80h ~ 8Eh	8Fh	} 2 KB
Sector 33	90h ~ 9Eh	9Fh	
..			
..			
Sector 38	E0h ~ EEh	EFh	
Sector 39	F0h ~ FEh	FFh	

Table 11: MIFARE® Classic 4K Memory Map

Examples:

// To authenticate the Block 04h with a {TYPE A, key number 00h}.

// PC/SC V2.01, Obsolete

APDU = FF 88 00 04 60 00h

// To authenticate the Block 04h with a {TYPE A, key number 00h}.

// PC/SC V2.07

APDU = FF 86 00 00 05 01 00 04 60 00h

Byte Number	0	1	2	3	Page	
Serial Number	SN0	SN1	SN2	BCC0	0	} 512 bits or 64 bytes
Serial Number	SN3	SN4	SN5	SN6	1	
Internal/Lock	BCC1	Internal	Lock0	Lock1	2	
OTP	OPT0	OPT1	OTP2	OTP3	3	
Data read/write	Data0	Data1	Data2	Data3	4	
Data read/write	Data4	Data5	Data6	Data7	5	
Data read/write	Data8	Data9	Data10	Data11	6	
Data read/write	Data12	Data13	Data14	Data15	7	
Data read/write	Data16	Data17	Data18	Data19	8	
Data read/write	Data20	Data21	Data22	Data23	9	
Data read/write	Data24	Data25	Data26	Data27	10	
Data read/write	Data28	Data29	Data30	Data31	11	
Data read/write	Data32	Data33	Data34	Data35	12	
Data read/write	Data36	Data37	Data38	Data39	13	
Data read/write	Data40	Data41	Data42	Data43	14	
Data read/write	Data44	Data45	Data46	Data47	15	

Table 12: MIFARE Ultralight® Memory Map

6.4.3. Read Binary Blocks

This command is used for retrieving a multiple of “data blocks” from the PICC. The data block/trailer block must be authenticated first before executing this command.

Read Binary APDU Format (5 bytes)

Command	Class	INS	P1	P2	Le
Read Binary Blocks	FFh	B0h	00h	Block Number	Number of Bytes to Read

Where:

Block Number 1 byte. The starting block.

Number of Bytes to Read 1 byte.

Multiple of 16 bytes for MIFARE Classic (1K/4K) or Multiple of 4 bytes for MIFARE Ultralight

Maximum of 16 bytes for MIFARE Ultralight.

Maximum of 48 bytes for MIFARE Classic 1K. (Multiple Blocks Mode; 3 consecutive blocks)

Maximum of 240 bytes for MIFARE Classic 4K. (Multiple Blocks Mode; 15 consecutive blocks)

Example 1: 10h (16 bytes). Starting block only. (Single Block Mode)

Example 2: 40h (64 bytes). From the starting block to starting block+3. (Multiple Block Mode)

Note: For security reasons, the Multiple Block Mode is used for accessing Data Blocks only. The Trailer Block is not supposed to be accessed in Multiple Blocks Mode. Please use Single Block Mode to access the Trailer Block.

Read Binary Block Response Format (Multiply of 4/16 + 2 bytes)

Response	Data Out		
Result	Data (Multiple of 4/16 Bytes)	SW1	SW2

Read Binary Block Response Codes

Results	SW1	SW2	Meaning
Success	90h	00h	The operation is completed successfully.
Error	63h	00h	The operation is failed.

Examples:

// Read 16 bytes from the binary block 04h (MIFARE Classic 1K/ 4K)

APDU = FF B0 00 04 10h

// Read 240 bytes starting from the binary block 80h (MIFARE Classic 4K)

// Block 80h to Block 8Eh (15 blocks)

APDU = FF B0 00 80 F0h

6.4.4. Update Binary Blocks

This command is used for writing a multiple of “data blocks” into the PICC. The data block/trailer block must be authenticated first before executing this command.

Update Binary APDU Format (Multiple of 16 + 5 bytes)

Command	Class	INS	P1	P2	Lc	Data In
Update Binary Blocks	FFh	D6h	00h	Block Number	Number of bytes to update	Block Data (Multiple of 16 Bytes)

Where:

Block Number	1 byte. The starting block to be updated.
Number of bytes to update	1 byte. <ul style="list-style-type: none"> Maximum 16 bytes for MIFARE Classic (1K/4K) or 4 bytes for MIFARE Ultralight. Maximum 48 bytes for MIFARE Classic 1K. (Multiple Blocks Mode; 3 consecutive blocks) Maximum 240 bytes for MIFARE Classic 4K. (Multiple Blocks Mode; 15 consecutive blocks)
Block Data	Multiple of 16 + 2 Bytes, or 6 bytes. The data to be written into the binary block/blocks.

Example 1: 10h (16 bytes). The starting block only. (Single Block Mode)

Example 2: 30h (48 bytes). From the starting block to starting block +2. (Multiple Block Mode)

Note: For safety reasons, the Multiple Block Mode is used for accessing data blocks only. The Trailer Block is not supposed to be accessed in Multiple Blocks Mode. Please use Single Block Mode to access the Trailer Block.

Update Binary Block Response Codes (2 bytes)

Results	SW1	SW2	Meaning
Success	90h	00h	The operation is completed successfully.
Error	63h	00h	The operation is failed.

Examples:

// Update the binary block 04h of MIFARE Classic (1K/4K) with Data {00 01 .. 0Fh}

APDU = FF D6 00 04 10 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0Fh

// Update the binary block 04h of MIFARE Ultralight with Data {00 01 02 03h}

APDU = FF D6 00 04 04 00 01 02 03h

6.4.5. Value Block Operation (INC, DEC, STORE)

This command is used for manipulating value-based transactions (e.g., increment a value of the value block).

Value Block Operation APDU Format (10 bytes)

Command	Class	INS	P1	P2	Lc	Data In	
Value Block Operation	FFh	D7h	00h	Block Number	05h	VB_OP	VB_Value (4 Bytes) {MSB .. LSB}

Where:

Block Number 1 byte. The value block to be manipulated.

VB_OP 1 byte.

00h = Store the VB_Value into the block. The block will then be converted to a value block.

01h = Increment the value of the value block by the VB_Value. This command is only valid for value block.

02h = Decrement the value of the value block by the VB_Value. This command is only valid for value block.

VB_Value 4 bytes. The value used for value manipulation. The value is a signed long integer (4 bytes).

Example 1: Decimal -4 = {FFh, FFh, FFh, FCh}

VB_Value			
MSB			LSB
FFh	FFh	FFh	FCh

Example 2: Decimal 1 = {00h, 00h, 00h, 01h}

VB_Value			
MSB			LSB
00h	00h	00h	01h

Value Block Operation Response Format (2 bytes)

Response	Data Out	
Result	SW1	SW2

Value Block Operation Response Codes

Results	SW1	SW2	Meaning
Success	90h	00h	The operation is completed successfully.
Error	63h	00h	The operation is failed.

6.4.6. Read Value Block

This command is used for retrieving the value from the value block. This command is only valid for value block.

Read Value Block APDU Format (5 bytes)

Command	Class	INS	P1	P2	Le
Read Value Block	FFh	B1h	00h	Block Number	04h

Where:

Block Number 1 byte. The value block to be accessed.

Read Value Block Response Format (4 + 2 bytes)

Response	Data Out		
Result	Value {MSB .. LSB}	SW1	SW2

Where:

Value 4 bytes. The value returned from the card. The value is a signed long integer (4 bytes).

Example 1: Decimal -4 = {FFh, FFh, FFh, FCh}

Value			
MSB			LSB
FFh	FFh	FFh	FCh

Example 2: Decimal 1 = {00h, 00h, 00h, 01h}

Value			
MSB			LSB
00h	00h	00h	01h

Read Value Block Response Codes

Results	SW1	SW2	Meaning
Success	90h	00h	The operation is completed successfully.
Error	63h	00h	The operation is failed.

6.4.7. Copy Value Block

This command is used for copying a value from a value block to another value block.

Copy Value Block APDU Format (7 bytes)

Command	Class	INS	P1	P2	Lc	Data In	
Value Block Operation	FFh	D7h	00h	Source Block Number	02h	03h	Target Block Number

Where:

- Source Block Number** 1 byte. The value of the source value block will be copied to the target value block.
- Target Block Number** 1 byte. The value block to be restored. The source and target value blocks must be in the same sector.

Copy Value Block Response Format (2 bytes)

Response	Data Out	
Result	SW1	SW2

Copy Value Block Response Codes

Results	SW1	SW2	Meaning
Success	90h	00h	The operation is completed successfully.
Error	63h	00h	The operation is failed.

Examples:

// Store a value "1" into block 05h

APDU = FF D7 00 05 05 00 00 00 00 01h

// Read the value block 05h

APDU = FF B1 00 05 04h

// Copy the value from value block 05h to value block 06h

APDU = FF D7 00 05 02 03 06h

// Increment the value block 05h by "5"

APDU = FF D7 00 05 05 01 00 00 00 05h

6.5. Accessing PC/SC-compliant tags (ISO 14443-4)

All ISO 14443-4 compliant cards (PICCs) understand the ISO 7816-4 APDUs. ACR1255U-J1 just has to communicate with the ISO 14443-4 compliant cards through exchanging ISO 7816-4 APDUs and responses. ACR1255U-J1 will handle the ISO 14443 Parts 1-4 Protocols internally.

MIFARE Classic (1K/4K), MIFARE Mini and MIFARE Ultralight tags are supported through the T=CL emulation. Just simply treat the MIFARE tags as standard ISO 14443-4 tags. For more information, please refer to **Section 6.4**.

ISO 7816-4 APDU Format

Command	Class	INS	P1	P2	Lc	Data In	Le
ISO 7816 Part 4 Command					Length of the Data In		Expected length of the Response Data

ISO 7816-4 Response Format (Data + 2 bytes)

Response	Data Out		
Result	Response Data	SW1	SW2

Common ISO 7816-4 Response Codes

Results	SW1	SW2	Meaning
Success	90h	00h	The operation is completed successfully.
Error	63h	00h	The operation is failed.

Typical sequence may be:

1. Present the tag and connect the PICC Interface.
2. Read/Update the memory of the tag.

To do this:

1. Connect the tag.

The ATR of the tag is 3B 88 80 01 00 00 00 00 33 81 81 00 3Ah.

Wherein:

Application Data of ATQB = 00 00 00 00

Protocol information of ATQB = 33 81 81.

It is an ISO 14443-4 Type B tag.

2. Send an APDU, Get Challenge.

<< 00 84 00 00 08h

>> 1A F7 F3 1B CD 2B A9 58h [90 00h]

Note: For ISO 14443-4 Type A tags, the ATS can be obtained by using the APDU “FF CA 01 00 00h.”



Example:

// Read 8 bytes from an ISO 14443-4 Type B PICC (ST19XR08E)

APDU = 80 B2 80 00 08h

Class = 80h

INS = B2h

P1 = 80h

P2 = 00h

Lc = None

Data In = None

Le = 08h

Answer: 00 01 02 03 04 05 06 07h [\$9000h]



6.6. Peripherals Control

The reader's peripherals control commands are implemented by using PC_to_RDR_Escape.

Command Format

Offset	Field	Size	Value	Description
0	<i>bMessageType</i>	1	6Bh	
1	<i>dwLength</i>	4		Size of <i>abData</i> field of this message
5	<i>bSlot</i>	1		Identifies the slot number for this command
6	<i>bSeq</i>	1		Sequence number for command
7	<i>abRFU</i>	3		Reserved for Future Use
10	<i>abData</i>	Byte array		Data block sent to the CCID



6.6.1. Get Firmware Version

This command is used for getting the reader's firmware message.

Get Firmware Version Format (5 bytes)

Command	Class	INS	P1	P2	Lc
Get Firmware Version	E0h	00h	00h	18h	00h

Get Firmware Version Response Format (5 bytes + Firmware Message Length)

Response	Class	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	Number of bytes to receive	Firmware Version

Example:

Response = E1 00 00 00 14 41 43 52 31 32 35 35 55 2D 4A 31 20 53 57 56 20 31 2E 30 35

Firmware Version (HEX) = 41 43 52 31 32 35 35 55 2D 4A 31 20 53 57 56 20 31 2E 30 35

Firmware Version (ASCII) = "ACR1255U-J1 SWV 1.05"



6.6.2. Get Serial Number

This command is used for getting the reader's serial number.

Get serial number Format (5 bytes)

Command	Class	INS	P1	P2	Lc
Get Serial Number	E0h	00h	00h	47h	00h

LED Control Response Format (N bytes)

Response	Class	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	Number of bytes to be Received	Serial Number

Example:

Response = E1 00 00 00 0C 52 52 33 33 30 2D 30 30 30 30 31 36

Serial Number (HEX) = 52 52 33 33 30 2D 30 30 30 30 31 36

Serial Number (ASCII) = "RR330-000016"



6.6.3. LED Control (in USB Mode only)

This command is used for controlling the LED's output.

LED Control Format (6 bytes)

Command	Class	INS	P1	P2	Lc	Data In
LED Control	E0h	00h	00h	29h	01h	LED Status

LED Control Response Format (6 bytes)

Response	Class	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	01h	LED Status

LED Status (1 byte)

LED Status	LED	COLOR	Description
Bit 0	1	GREEN	1 = ON; 0 = OFF
Bit 1	1	RED	1 = ON; 0 = OFF
Bit 2	2	BLUE	1 = ON; 0 = OFF
Bit 3	2	RED	1 = ON; 0 = OFF
Bit 4 - 7		RFU	RFU



6.6.4. LED Status

This command is used for checking the existing LED's status.

LED Status Format (5 bytes)

Command	Class	INS	P1	P2	Lc
LED Status	E0h	00h	00h	29h	00h

LED Status Response Format (6 bytes)

Response	Class	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	01h	LED Status

LED Status (1 byte)

LED Status	LED	COLOR	Description
Bit 0	1	GREEN	1 = ON; 0 = OFF
Bit 1	1	RED	1 = ON; 0 = OFF
Bit 2	2	GREEN	1 = ON; 0 = OFF
Bit 3	2	RED	1 = ON; 0 = OFF
Bit 4 - 7		RFU	RFU



6.6.5. Buzzer Control

This command is used for controlling the buzzer output.

Buzzer Control Format (6 bytes)

Command	Class	INS	P1	P2	Lc	Data In
Buzzer Control	E0h	00h	00h	28h	01h	Buzzer On Duration

Where:

Buzzer On Duration 1 byte
00h = Turn OFF
01 to FFh = Duration (unit: 10 ms)

Buzzer Control Response Format (6 bytes)

Response	Class	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	01h	00h

6.6.6. Set LED and Buzzer Status Indicator Behavior

This command is used for setting the behaviors of LEDs and buzzer as status indicators.

Note: The setting will be saved into non-volatile memory.

Set LED and Buzzer Status Indicator Behavior Format (6 bytes)

Command	Class	INS	P1	P2	Lc	Data In
Set LED and Buzzer Status Indicator Behavior	E0h	00h	00h	21h	01h	Default Behavior

Behavior (1 byte)

Behavior	Mode	Description
Bit 0	RFU	RFU
Bit 1	PICC Polling Status LED	To show the PICC Polling Status. 1 = Enable; 0 =Disable
Bit 2	PICC Activation Status LED	To show the activation status of the PICC interface 1 = Enable; 0 =Disable
Bit 3	Card Insertion and Removal Events Buzzer	To make a beep whenever a card insertion or removal event is detected. 1 = Enable; 0 =Disabled
Bit 4 – 6	RFU	RFU
Bit 7	Card Operation Blinking LED	To light up the LED whenever the card is being accessed

Note: Default value of behavior = 8Fh.

Set LED and Buzzer Status Indicator Behavior Response Format (6 bytes)

Response	Class	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	01h	Default Behaviors



6.6.7. Read LED and Buzzer Status Indicator Behavior

This command is used for reading the current default behaviors of LEDs and buzzer.

Read LED and Buzzer Status Indicator Behavior Format (5 bytes)

Command	Class	INS	P1	P2	Lc
Read LED and Buzzer Status Indicator Behavior	E0h	00h	00h	21h	00h

Read LED and Buzzer Status Indicator Behavior Response Format (6 bytes)

Response	Class	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	01h	Behaviors

Behavior (1 byte)

Behavior	Mode	Description
Bit 0	RFU	RFU
Bit 1	PICC Polling Status LED	To show the PICC Polling Status. 1 = Enable; 0 =Disable
Bit 2	PICC Activation Status LED	To show the activation status of the PICC interface 1 = Enable; 0 =Disable
Bit 3	Card Insertion and Removal Events Buzzer	To make a beep whenever a card insertion or removal event is detected. 1 = Enable; 0 =Disabled
Bit 4 – 6	RFU	RFU
Bit 7	Card Operation Blinking LED	To light up the LED whenever the card is being accessed

Note: Default value of Behavior = 8Fh.



6.6.8. Set Automatic PICC Polling

This command is used for setting the reader's polling mode.

Whenever the reader is connected to the PC, the PICC polling function will start the PICC scanning to determine if a PICC is placed on/removed from the built-antenna.

A command may be sent to disable the PICC polling function. The command is sent through the PCSC Escape command interface. To meet the energy saving requirement, special modes are provided for turning off the antenna field whenever the PICC is inactive, or no PICC is found. The reader will consume less current in power saving mode.

Note: The setting will be saved into non-volatile memory.

Set Automatic PICC Polling Format (6 bytes)

Command	Class	INS	P1	P2	Lc	Data In
Set Automatic PICC Polling	E0h	00h	00h	23h	01h	Polling Setting

Set Automatic PICC Polling Response Format (6 bytes)

Response	Class	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	01h	Polling Setting

Polling Setting (1 byte)

Polling Setting	Mode	Description
Bit 0	Automatic PICC Polling	1 = Enable; 0 =Disable
Bit 1	Turn off Antenna Field if no PICC found	1 = Enable; 0 =Disable
Bit 2	Turn off Antenna Field if the PICC is inactive	1 = Enable; 0 =Disable
Bit 3	Activate the PICC when detected.	1 = Enable; 0 =Disable
Bit 5 .. 4	PICC Poll Interval for PICC	<Bit 5 – Bit 4> <0 – 0> = 250 ms <0 – 1> = 500 ms <1 – 0> = 1000 ms <1 – 1> = 2500 ms
Bit 6	RFU	-
Bit 7	Enforce ISO14443A Part 4	1= Enable; 0= Disable

Note: Default value of Polling Setting = 8Fh.



Reminders:

1. It is recommended to enable the option “Turn Off Antenna Field if the PICC is inactive”, so that the “Inactive PICC” will not be exposed to the field all the time to prevent the PICC from “warming up”.
2. The longer the PICC poll interval, the more efficient the energy saving will be. However, the response time of PICC Polling will become longer. The Idle Current Consumption in Power Saving Mode is about 60 mA, while the Idle Current Consumption in Non-Power Saving mode is about 130mA. **Note:** Idle Current Consumption = PICC is not activated.
3. The reader will activate the ISO 14443A-4 mode of the “ISO 14443A-4 compliant PICC” automatically. Type B PICC will not be affected by this option.
4. The JCOP30 card comes with two modes: ISO 14443A-3 (MIFARE Classic 1K) and ISO 14443A-4 modes. The application has to decide which mode should be selected once the PICC is activated.

6.6.9. Read Automatic PICC Polling

This command is used for checking the current PICC polling setting.

Read Automatic PICC Polling Format (5 bytes)

Command	Class	INS	P1	P2	Lc
Read Automatic PICC Polling	E0h	00h	00h	23h	00h

Read the Configure Mode Response Format (6 bytes)

Response	Class	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	01h	Polling Setting

Polling Setting (1 byte)

Polling Setting	Mode	Description
Bit 0	Automatic PICC Polling	1 = Enable; 0 =Disable
Bit 1	Turn off Antenna Field if no PICC found	1 = Enable; 0 =Disable
Bit 2	Turn off Antenna Field if the PICC is inactive	1 = Enable; 0 =Disable
Bit 3	Activate the PICC when detected	1 = Enable; 0 =Disable
Bit 5 .. 4	PICC Poll Interval for PICC	<Bit 5 – Bit 4> <0 – 0> = 250 ms <0 – 1> = 500 ms <1 – 0> = 1000 ms <1 – 1> = 2500 ms
Bit 6	RFU	-
Bit 7	Enforce ISO14443A Part 4	1= Enable; 0= Disable

Note: Default value of Polling Setting = 8Fh.



6.6.10. Set PICC Operating Parameter

This command is used for setting the PICC operating parameter.

Note: The setting will be saved into non-volatile memory.

Set the PICC Operating Parameter Format (6 bytes)

Command	Class	INS	P1	P2	Lc	Data In
Set the PICC Operating Parameter	E0h	00h	00h	20h	01h	Operation Parameter

Set the PICC Operating Parameter Response Format (6 bytes)

Response	Class	INS	P1	P2	Le	Data Out
Result	E1	00h	00h	00h	01h	Operation Parameter

Operating Parameter (1 byte)

Operating Parameter	Parameter	Description	Option
Bit 0	ISO14443 Type A	The Tag Types to be detected during PICC Polling.	1 = Detect 0 = Skip
Bit 1	ISO14443 Type B		1 = Detect 0 = Skip
Bit 2	Felica		1 = Detect 0 = Skip
Bit 3	RFU	RFU	RFU
Bit 4	Topaz	The Tag Types to be detected during PICC Polling.	1 = Detect 0 = Skip
Bit 5 - 7	RFU	RFU	RFU

Note: Default value of Operation Parameter = 17h.



6.6.11. Read PICC Operating Parameter

This command is used for checking the current PICC operating parameter.

Read the PICC Operating Parameter Format (5 bytes)

Command	Class	INS	P1	P2	Lc
Read the PICC Operating Parameter	E0h	00h	00h	20h	00h

Read the PICC Operating Parameter Response Format (6 bytes)

Response	Class	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	01h	Operation Parameter

Operating Parameter (1 byte)

Operating Parameter	Parameter	Description	Option
Bit 0	ISO14443 Type A	The Tag Types to be detected during PICC Polling.	1 = Detect 0 = Skip
Bit 1	ISO14443 Type B		1 = Detect 0 = Skip
Bit 2	Felica		1 = Detect 0 = Skip
Bit 3	RFU	RFU	RFU
Bit 4	Topaz	The Tag Type to be detected during PICC Polling.	1 = Detect 0 = Skip
Bit 5 - 7	RFU	RFU	RFU

6.6.12. Set Auto PPS

Whenever a PICC is recognized, the reader will try to change the communication speed between the PCD and PICC defined by the maximum connection speed. If the card does not support the proposed connection speed, the reader will try to connect the card with a slower speed setting.

Set Auto PPS Format (7 bytes)

Command	Class	INS	P1	P2	Lc	Data In	
Set Auto PPS	E0h	00h	00h	24h	02h	Max Tx Speed	Max Rx Speed

Set Auto PPS Response Format (9 bytes)

Response	Class	INS	P1	P2	Le	Data Out			
Result	E1h	00h	00h	00h	04h	Max Tx Speed	Current Tx Speed	Max Rx Speed	Current Rx Speed

Where:

Max Tx Speed	1 byte. Maximum Transmission Speed.
Max Rx Speed	1 byte. Maximum Receiving Speed.
Current Tx Speed	1 byte. Current Transmission Speed.
Current Rx Speed	1 byte. Current Receiving Speed.

Value can be: 00h = 106 Kbps

01h = 212 Kbps

02h = 424 Kbps

03h = 848 Kbps

FFh = No Auto PPS

Note: Default setting is 00h.

Notes:

1. Normally, the application should be able to know the maximum connection speed of the PICCs being used. The environment can also affect the maximum achievable speed. The reader uses the proposed communication speed to talk with the PICC. The PICC will become inaccessible if the PICC or the environment does not meet the requirement of the proposed communication speed.
2. The reader supports different speed between sending and receiving.



6.6.13. Read Auto PPS

This command is used to check the current Auto PPS Setting.

Read Auto PPS Format (5 bytes)

Command	Class	INS	P1	P2	Lc
Read Auto PPS	E0h	00h	00h	24h	00h

Set Auto PPS Response Format (9 Bytes)

Response	Class	INS	P1	P2	Le	Data Out			
Result	E1	00h	00h	00h	04h	Max Tx Speed	Current Tx Speed	Max Rx Speed	Current Rx Speed

Where:

Max Tx Speed	1 byte. Maximum Transmission Speed.
Max Rx Speed	1 byte. Maximum Receiving Speed.
Current Tx Speed	1 byte. Current Transmission Speed.
Current Rx Speed	1 byte. Current Receiving Speed.

Value can be: 00h = 106 Kbps

01h = 212 Kbps

02h = 424 Kbps

03h = 848 Kbps

FFh = No Auto PPS

Note: Default setting is 02h.



6.6.14. Antenna Field Control

This command is used for turning on/off the antenna field.

Antenna Field Control Format (6 bytes)

Command	Class	INS	P1	P2	Lc	Data In
Antenna Field Control	E0h	00h	00h	25h	01h	Status

Antenna Field Control Response Format (6 bytes)

Response	Class	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	01h	Status

Where:

Status 1 byte.
 01h = Enable Antenna Field
 00h = Disable Antenna Field

Note: Make sure the Auto PICC Polling is disabled first before turning off the antenna field.



6.6.15. Read Antenna Field Status

This command is used to check current Antenna Field Status.

Read Antenna Field Status Format (5 bytes)

Command	Class	INS	P1	P2	Lc
Read Antenna Field Status	E0h	00h	00h	25h	00h

Read Antenna Field Status Response Format (6 bytes)

Response	Class	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	01h	Status

Where:

- Status** 1 byte.
- 00h = PICC Power Off
 - 01h = PICC Idle [Ready to Poll Contactless Tag, but not detected]
 - 02h = PICC Ready [PICC Request (Ref to ISO 14443) Success, i.e. Contactless Tag Detected]
 - 03h = PICC Selected [PICC Select (Ref to ISO 14443) Success]
 - 04h = PICC Activate [PICC Activation (Ref to ISO 14443) Success, Ready for APDU Exchange]



6.6.16. Card Emulation Mode Conversion

This command is used to turn on the card emulation mode, and emulate MIFARE Ultralight.

Set the Card Emulation Format (6 bytes)

Command	Class	INS	P1	P2	Lc	Data In					
Set the Card Emulation	E0h	00h	00h	99h	06h	98h	01h	CardType	1Ah	01h	RegValue

Where:

CardType 1 byte.

01h = MIFARE Ultralight emulation

RegValue 1 byte. Value writer to PN512's GsNoff Register

Set the Card Emulation Response Format (11 bytes)

Response	Class	INS	P1	P2	Le	Data Out					
Result	E1h	00h	00h	00h	06h	98h	01h	CardType	1Ah	01h	Value



6.6.17. Sleep Mode Option

By default, the reader will enter the sleep mode if there is no operation for 60 seconds.

This command is used to set the time interval before entering sleep mode.

Set Sleep Time Interval Format (5 bytes)

Command	Class	INS	P1	P2	Lc
Set Sleep Time Interval	E0h	00h	00h	48h	Time

Where:

Time 1 byte

00h = 60 seconds

01h = 90 seconds

02h = 120 seconds

03h = 180 seconds

Set Sleep Time Interval Response Format (6 bytes)

Response	Class	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	01h	00h



6.6.18. Change Tx Power command

Set Tx Power Format (5 bytes)

Command	Class	INS	P1	P2	Lc
Set Tx power	E0h	00h	00h	49h	Tx Power

Where:

Tx Power 1 byte
00h = -23 dBm
01h = -6 dBm
02h = 0 dBm
03h = 4 dBm

Set Tx Power Response Format (5 bytes)

Response	Class	INS	P1	P2	Lc
Result	E1h	00h	00h	00h	00h



6.6.19. Read Tx Power Value

Read Tx Power Value Format (5 bytes)

Command	Class	INS	P1	P2	Lc
Set Tx power	E0h	00h	00h	50h	00h

Read Tx Power Value Response Format (6 bytes)

Response	Class	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	01h	Tx Power

Where:

Tx Power 1 byte
 00h = -23 dBm
 01h = -6 dBm
 02h = 0 dBm
 03h = 4 dBm



Appendix A. Access MIFARE DESFire tags (ISO 14443-3)

MIFARE DESFIRE supports ISO 7816-4 APDU Wrapping and Native modes. Once the MIFARE DESFire tag is activated, the first APDU command sent to the MIFARE DESFire tag will determine the “Command Mode”. If the first APDU is in “Native Mode”, the rest of the APDU commands must be in “Native Mode” format. Similarly, if the first APDU is “ISO 7816-4 APDU Wrapping Mode”, the rest of the APDUs must be in “ISO 7816-4 APDU Wrapping Mode” format.

Example 1: MIFARE DESFire ISO 7816-4 APDU Wrapping.

// To read 8 bytes random number from an ISO 14443-4 Type A PICC (DESFIRE)

APDU = {90 0A 00 00 01 00 00}

Class = 90h; INS = 0Ah (DESFire Instruction); P1 = 00h; P2 = 00h

Lc = 01h; Data In = 00h; Le = 00h (Le = 00h for maximum length)

Answer: 7B 18 92 9D 9A 25 05 21 [\$91AF]

Status Code {91 AF} is defined in DESFIRE specification. Please refer to the DESFIRE specification for more details.

Example 2: MIFARE DESFire Frame Level Chaining (ISO 7816 wrapping mode)

// In this example, the application has to do the “Frame Level Chaining”.

// To get the version of the DESFIRE card.

Step 1: Send an APDU {90 60 00 00 00} to get the first frame. INS=60h

Answer: 04 01 01 00 02 18 05 91 AF [\$91AF]

Step 2: Send an APDU {90 AF 00 00 00} to get the second frame. INS=AFh

Answer: 04 01 01 00 06 18 05 91 AF [\$91AF]

Step 3: Send an APDU {90 AF 00 00 00} to get the last frame. INS=AFh

Answer: 04 52 5A 19 B2 1B 80 8E 36 54 4D 40 26 04 91 00 [\$9100]

Android is a trademark of Google Inc.

Atmel is a registered trademark of Atmel Corporation or its subsidiaries, in the US and/or other countries.

Infineon is a registered trademark of Infineon Technologies AG.

Microsoft is a registered trademark of the Microsoft group of companies.

MIFARE, MIFARE Classic, MIFARE DESFire, MIFARE Mini and MIFARE Ultralight are registered trademarks of NXP B.V. and are used under license.

The Bluetooth® word, mark and logos are registered trademarks owned by Bluetooth SIG, Inc. and any use of such marks by Advanced Card Systems Ltd. is under license. Other trademarks and trade names are those of their respective owners.