



Advanced Card Systems Ltd.
Card & Reader Technologies

ACR1255U-J1 ACS Secure Bluetooth® NFC Reader

Reference Manual V1.13





Revision History

Release Date	Revision Description	Version Number
2015-09-07	<ul style="list-style-type: none">Initial Release	1.00
2016-02-02	<ul style="list-style-type: none">Updated Section 2.0 FeaturesUpdated Section 4.1.2 Battery LifeUpdated Section 4.5.1 KeysUpdated Section 4.5.3 Status LEDUpdated Section 4.5.4 BuzzerUpdated Section 5.3 AuthenticationUpdated Section 6.7.2 Get Serial NumberUpdated Section 6.7.16 Sleep Mode OptionUpdated Section 6.6.18 Read Tx Power Value	1.01
2016-06-06	<ul style="list-style-type: none">Updated Section 5.6.5 Notify Card Status CommandUpdated Section 6.2.1 ATR GenerationUpdated Section 6.4.1 Load Authentication KeysUpdated Section 6.7.6 Set LED and Buzzer Status Indicator BehaviorUpdated Section 6.7.7 Read LED and Buzzer Status Indicator BehaviorUpdated Section 6.7.8 Set Automatic PICC PollingUpdated Section 6.7.10 Set PICC Operating ParameterUpdated Section 6.7.11 Read PICC Operating ParameterUpdated Section 6.7.12 Set Auto PPSUpdated Section 6.7.13 Read Auto PPSUpdated Section 6.7.16 Sleep Mode OptionUpdated Section 6.7.17 Change Tx Power CommandUpdated Section 6.7.18 Read Tx Power ValueAdded Section 6.7.19 Customer Master Key Rewrite	1.02
2016-09-16	<ul style="list-style-type: none">Updated Product Marketing Name	1.03



Release Date	Revision Description	Version Number
2017-01-10	<ul style="list-style-type: none">• Updated format of commands and examples• Added Example of a Mutual Authentication Communication (section 5.7.1)• Updated Table 5 value• Deleted the PIN passcode description• Updated Section 2.0 Features• Updated Section 4.1.2 Battery Life• Updated Section 5.6.5 Notify Card Status Command• Added Section 6.4 APDU Commands for PCSC 2.0 Part 3• Updated Section 6.7.4 LED Color• Updated Section 6.7.17 Change Tx Power Command• Updated Section 6.7.18 Read Tx Power Value	1.04
2017-10-02	<ul style="list-style-type: none">• Updated Table 5: Service Handles and UUID Information List• Updated Section 5.3 Authentication• Updated Section 5.7 Mutual Authentication• Added Section 6.7 Accessing FeliCa Tags	1.05
2017-12-28	<ul style="list-style-type: none">• Updated Product Photo	1.06
2018-05-24	<ul style="list-style-type: none">• Updated Section 5.2 Profile Selection• Updated Section 5.6 Bluetooth Communication Protocol	1.07



Release Date	Revision Description	Version Number
2018-09-13	<ul style="list-style-type: none">• Updated Formatting• Updated Section 2.0 Features• Updated Section 3.0 System Block Diagram• Updated Section 5.6.4 APDU Command• Updated Section 5.6.6 Hardware Error Response• Updated Section 5.7.6 RDR_to_SPH_ACK (Error handling)• Updated Section 6.3.5 Buzzer Control• Updated Section 6.3.6 Set LED and Buzzer Status Indicator Behavior• Updated Section 6.3.8 Set Automatic PICC Polling• Updated Section 6.3.10 Set PICC Operating Parameter• Updated Section 6.3.11 Read PICC Operating Parameter• Updated Section 6.3.12 Set Auto PPS• Updated Section 6.3.16 Set Sleep Mode Option• Added Section 6.3.17 Read Sleep Mode Option• Updated Section 6.3.18 Set Tx Power Value• Updated Section 6.3.19 Read Tx Power Value• Added Section 6.3.21 Get Battery Level	1.08
2018-10-08	<ul style="list-style-type: none">• Updated Section 6.3.17 Read Sleep Mode Option• Updated Section 6.3.19 Read Tx Power Value• Updated Section 6.3.21 Get Battery Level	1.09
2019-06-14	<ul style="list-style-type: none">• Updated Product Marketing Name• Updated Formatting on Section 5.0 Software Design• Updated Section 5.1.3 Authentication	1.10
2019-07-08	<ul style="list-style-type: none">• Updated Section 5.1.3 Authentication	1.11
2019-08-02	<ul style="list-style-type: none">• Updated Section 6.2.4.7 Copy Value Block• Updated Section 6.3.8 Set Automatic PICC Polling• Updated Section 6.3.9 Read Automatic PICC Polling• Updated Section 6.3.17 Read Sleep Mode Option	1.12



Release Date	Revision Description	Version Number
2020-06-09	<ul style="list-style-type: none">• Added Section 6.2.2.2 Get PICC Data• Added Section 6.3.22 Read PICC Type• Updated Section 6.1 PC/SC API• Updated Section 6.3.6 Set LED and Buzzer Status Indicator Behavior• Updated Section 6.3.7 Read LED and Buzzer Status Indicator Behavior• Updated Section 6.3.9 Read Automatic PICC Polling	1.13



Table of Contents

1.0.	Introduction	8
1.1.	Symbols and Abbreviations	8
2.0.	Features	9
3.0.	System Block Diagram	10
4.0.	Hardware Design	13
4.1.	Battery	13
4.1.1.	Battery Charging	13
4.1.2.	Battery Life	13
4.2.	Bluetooth Interface	13
4.3.	USB Interface	13
4.3.1.	Communication Parameters	13
4.3.2.	Endpoints	14
4.4.	NFC Interface	15
4.4.1.	Carrier Frequency	15
4.4.2.	Card Polling	15
4.5.	User Interface	15
4.5.1.	Keys	15
4.5.2.	Mode Selection Switch	16
4.5.3.	Status LED	16
4.5.4.	Buzzer	17
5.0.	Software Design	18
5.1.	Bluetooth Communication	18
5.1.1.	Bluetooth Connection Flow	18
5.1.2.	Profile Selection	19
5.1.3.	Authentication	20
5.1.4.	Communication Profile	21
5.1.5.	Frame Format	22
5.1.6.	Bluetooth Communication Protocol	23
5.1.7.	Mutual Authentication and Encryption Protocol	33
6.0.	Host Programming API	38
6.1.	PC/SC API (For Windows®)	38
6.1.1.	SCardEstablishContext	38
6.1.2.	SCardListReaders	39
6.1.3.	SCardConnect	40
6.1.4.	SCardControl	41
6.1.5.	SCardTransmit	43
6.1.6.	SCardDisconnect	45
6.1.7.	APDU Flow	46
6.1.8.	Escape Command Flow	47
6.2.	Contactless Smart Card Protocol	48
6.2.1.	ATR Generation	48
6.2.2.	Pseudo APDU for Contactless Interface	51
6.2.3.	APDU commands for PC/SC 2.0 Part 3 (version 2.02 or later)	53
6.2.4.	PICC Commands for MIFARE Classic (1K/4K) memory cards	67
6.2.5.	Accessing PC/SC-compliant tags (ISO 14443-4)	76
6.2.6.	Accessing MIFARE DESFire tags (ISO 14443-3)	78
6.2.7.	Accessing FeliCa tags	79
6.3.	Peripherals Control	80
6.3.1.	Get Firmware Version	81
6.3.2.	Get Serial Number	82
6.3.3.	LED Control	83
6.3.4.	LED Status	84



6.3.5.	Buzzer Control	85
6.3.6.	Set LED and Buzzer Status Indicator Behavior	86
6.3.7.	Read LED and Buzzer Status Indicator Behavior	88
6.3.8.	Set Automatic PICC Polling	89
6.3.9.	Read Automatic PICC Polling	91
6.3.10.	Set PICC Operating Parameter	92
6.3.11.	Read PICC Operating Parameter	93
6.3.12.	Set Auto PPS	94
6.3.13.	Read Auto PPS	95
6.3.14.	Antenna Field Control	96
6.3.15.	Read Antenna Field Status	97
6.3.16.	Set Sleep Mode Option	98
6.3.17.	Read Sleep Mode Option	99
6.3.18.	Change Tx Power	100
6.3.19.	Read Tx Power Value	101
6.3.20.	Customer Master Key Rewrite	102
6.3.21.	Get Battery Level	104
6.3.22.	Read PICC Type	105

List of Figures

Figure 1 :	ACR1255U-J1 Architecture	10
Figure 2 :	ACR1255U-J1 USB Communication Architecture	11
Figure 3 :	Bluetooth Protocol Stack	12
Figure 4 :	Keys on the ACR1255U-J1	15
Figure 5 :	LED Operation Status	16
Figure 6 :	Bluetooth Connection Flow	18
Figure 7 :	Authentication Procedure	20
Figure 8 :	ACR1255U-J1 APDU Flow	46
Figure 9 :	ACR1255U-J1 Escape Command Flow	47

List of Tables

Table 1 :	Symbols and Abbreviations	8
Table 2 :	Estimated Battery Lifespan	13
Table 3 :	USB Interface Wiring	13
Table 4 :	ACR1255U-J1 Bluetooth Service	19
Table 5 :	ACR1255U-J1 Service Handles and UUID Information List	19
Table 6 :	Command Code Summary	23
Table 7 :	Response Code Summary	23
Table 8 :	Summary of Mutual Authentication Commands	33
Table 9 :	Mutual Authentication Error codes	37
Table 10 :	MIFARE Classic 1K Memory Map	69
Table 11 :	MIFARE Classic 4K Memory Map	69
Table 12 :	MIFARE Ultralight® Memory Map	70



1.0. Introduction

The ACR1255U-J1 ACS Secure 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 Full Speed Interface
- Bluetooth Interface
- Plug and Play – CCID support brings utmost mobility
- Smart Card Reader:
 - Contactless Interface:
 - Read/Write speed of up to 424 Kbps
 - Built-in antenna for contactless tag access, with reading distance of up to 50 mm (depending on tag type)
 - Supports ISO 14443 Part 4 Type A and B cards, MIFARE®, FeliCa, and all 4 types of NFC (ISO/IEC 18092) tags
 - Built-in anti-collision feature (only one tag is accessed at any time)
 - Supports AES-128 encryption algorithm (CBC ciphering mode)
 - NFC Mode Supported:
 - Card reader/writer mode
- Application Programming Interface:
 - Supports PC/SC
 - Supports CT-API (through wrapper on top of PC/SC)
- Built-in Peripherals:
 - Two user-controllable bi-color LEDs
 - User-controllable buzzer
- USB Firmware Upgradeability¹
- Supports Android™ 4.3 and later²
- Supports iOS 8.0 and later³
- Compliant with the following standards:
 - EN 60950/IEC 60950
 - ISO 14443
 - ISO 18092
 - Bluetooth
 - PC/SC
 - CCID
 - CE
 - FCC
 - RoHS
 - REACH
 - VCCI (Japan)
 - TELEC (Japan)
 - Microsoft® WHQL

¹ Applicable under PC-linked mode

² Uses an ACS-defined Android Library

³ Uses an ACS-defined iOS Library

3.0. System Block Diagram

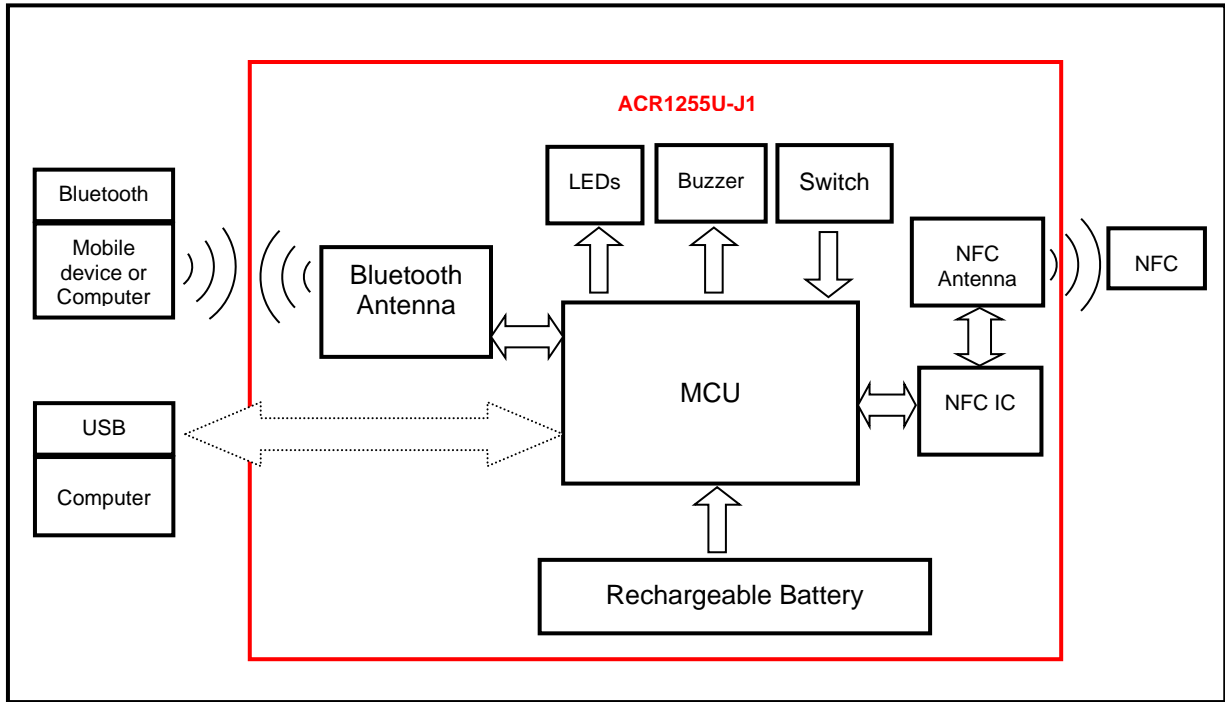


Figure 1: ACR1255U-J1 Architecture

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

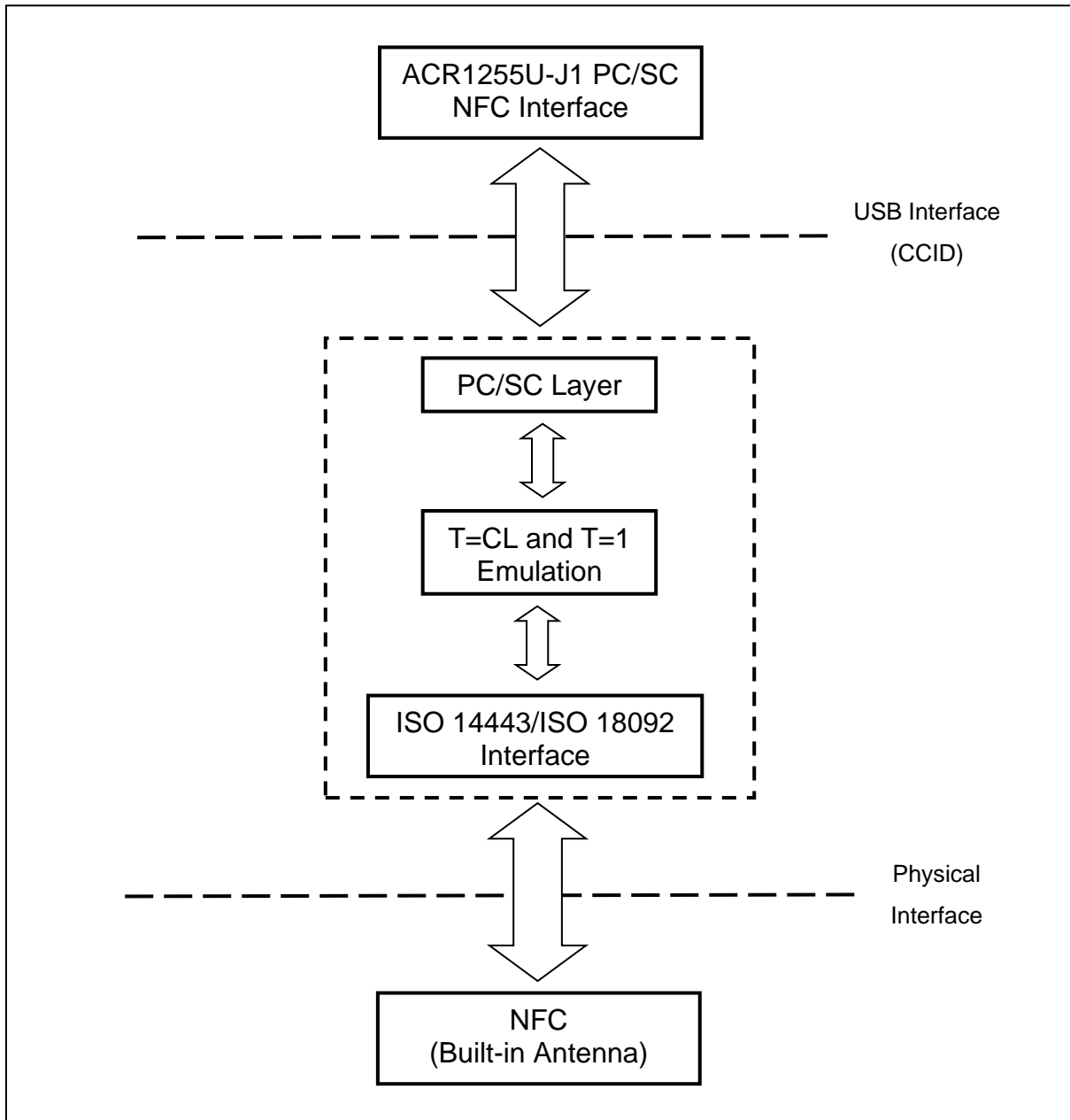


Figure 2: ACR1255U-J1 USB Communication Architecture



The Bluetooth 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 Protocol Stack



4.0. Hardware Design

4.1. Battery

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

4.1.1. Battery Charging

Once the battery of the ACR1255U-J1 runs out, it may be charged in any of the following modes: OFF, USB, or 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	14 days* ⁽¹⁾
Standby Mode	22 days ⁽²⁾
OFF Mode	2 years

Table 2: Estimated Battery Lifespan

Note: Results may vary as it depends on the smart card used.

⁽¹⁾ In Bluetooth mode, run 10 operations per day with 1 minute operation run.

⁽²⁾ In Bluetooth mode, set sleep time as 60 seconds and wake up once per day.

4.2. Bluetooth Interface

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

4.3. USB Interface

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

4.3.1. Communication Parameters

The ACR1255U-J1 connects to computers through USB as specified in the USB Specification 2.0. The ACR1255U-J1 works 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 computer
3	D+	Differential signal transmits data between ACR1255U-J1 and computer
4	GND	Reference voltage level for power supply

Table 3: USB Interface Wiring



4.3.2. Endpoints

The 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 be sent from ACR1255U-J1 to host (data packet size is 8 bytes)

4.4. NFC Interface

The interface between the ACR1255U-J1 and contactless cards 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 the ACR1255U-J1 is 13.56 MHz

4.4.2. Card Polling

The 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

The ACR1255U-J1 has 3 keys with locations shown in the figure below:

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

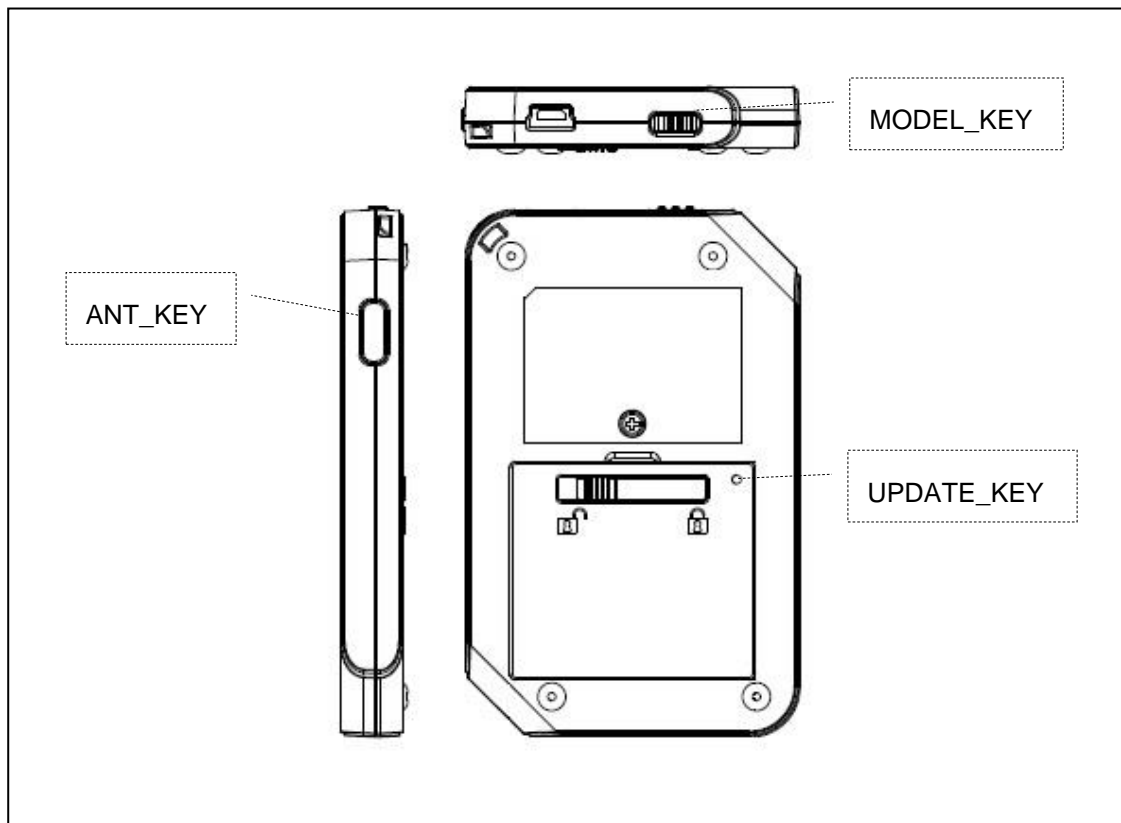





Figure 4: Keys on the ACR1255U-J1

4.5.2. Mode Selection Switch

The ACR1255U-J1 has three modes: USB, Off, and Bluetooth. The 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

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

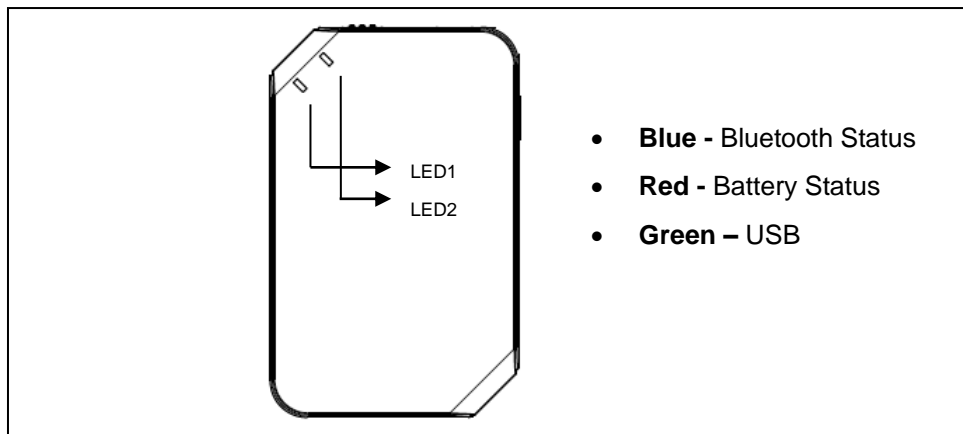


Figure 5: LED Operation Status

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 v1 ⁴	The paired Bluetooth device is authenticated successfully and is waiting for instructions
		Slow flash v2 ⁵	Bluetooth device is paired, card polling is on and no card is present
		Fast flash	Data is being transferred between the reader and paired device
		Fast – Slow flash	<ul style="list-style-type: none"> • Powered on • Searching for devices to be paired with
		On	<ul style="list-style-type: none"> • Bluetooth device is paired with the reader • Card is present

⁴ LED is on for 500 ms, off for 500 ms

⁵ LED is on for 200 ms, off for 1800 ms



Mode	Color	LED Activity	Status
	Red (LED2)	Off	<ul style="list-style-type: none"> Battery is fully charged Reader is powered off The voltage of the battery is greater than 3.5 V and no USB powered is being supplied
		Slow flash	Low battery
		On	Battery is charging
USB Mode	Green (LED2)	Off	Reader is powered off
		Slow flash	No card presented and the reader is waiting for instruction
		Fast flash	There is read/write access between the smart card and reader
		On	Card is present and the reader is waiting for instruction
	Orange ⁶ (LED2)	On	Device is powered on
	Red (LED1)	Off	<ul style="list-style-type: none"> Battery is fully charged Reader is powered off The voltage of the battery is greater than 3.5V and no USB powered is being supplied
		Slow flash	Low battery
On		Battery is charging	

4.5.4. Buzzer

The 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 has been detected or removed
One long beep	Reader has been powered on

⁶ Green and Red both turned on

5.0. Software Design

5.1. Bluetooth Communication

5.1.1. Bluetooth Connection Flow

The program flow of the Bluetooth connection is shown below:

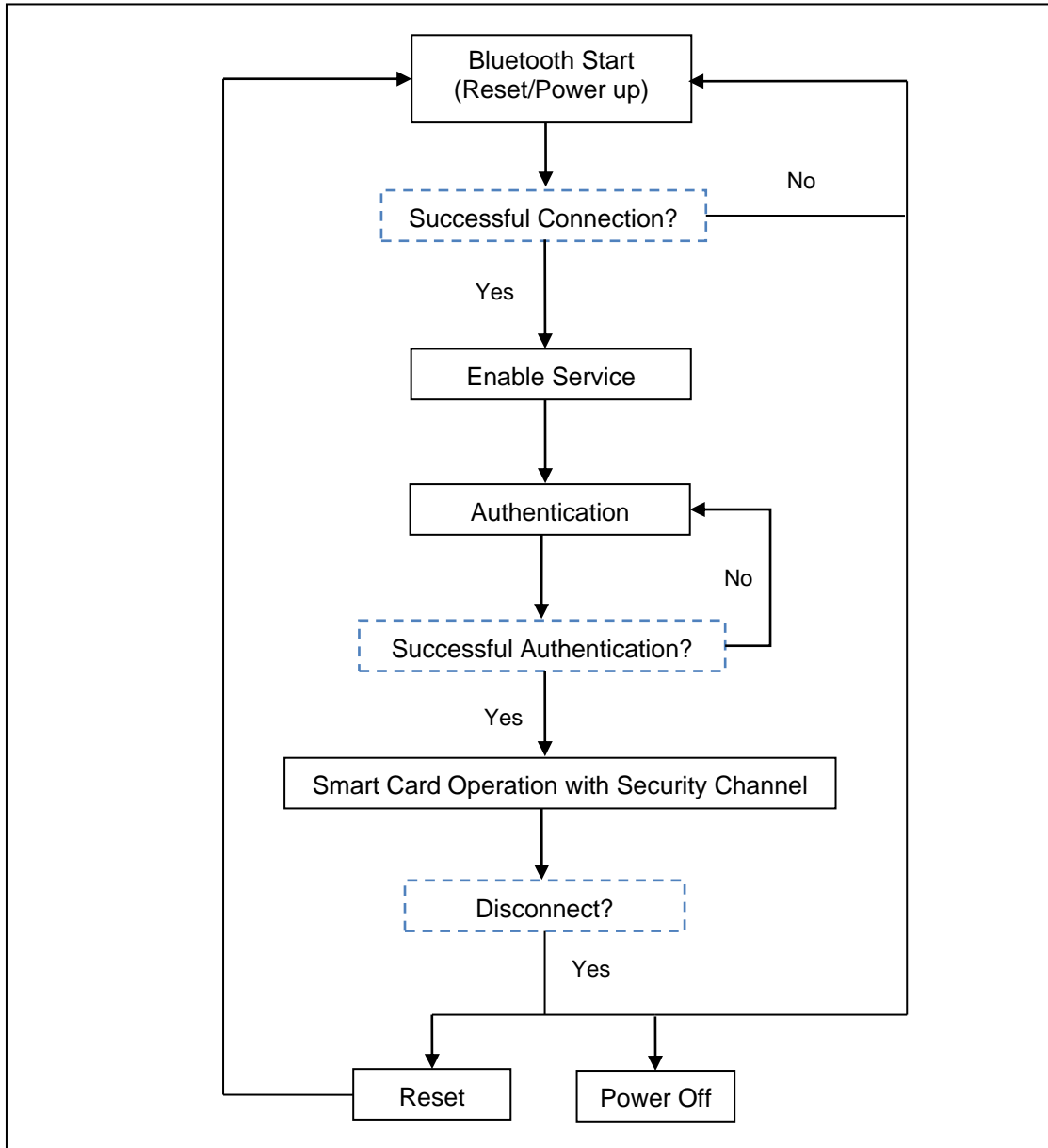


Figure 6: Bluetooth Connection Flow



5.1.2. Profile Selection

The 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: the first pipe is used for command request, the second pipe is for command response/card notification, and the third pipe is for 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	3C4AFF1-4783-3DE5-A983-D348718EF133	Commands Request
	3C4AFF2-4783-3DE5-A983-D348718EF133	Commands Response/Card Notification
	3C4AFF3-4783-3DE5-A983-D348718EF133	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 Bluetooth Service

Attribute Name	UUID	Handle
DeviceName	2A00	03h
Send(Reader → Paired device)	3C4AFF1-4783-3DE5-A983-D348718EF133	2Ah
Receive(Paired device → Reader)	3C4AFF2-4783-3DE5-A983-D348718EF133	2Dh
BatteryLevel	2A19	14h
Manufacturer	2A29	1Eh
SerialNumber	2A25	16h
FW_Version	2A26	18h
ModelNumber	2A24	14h

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

5.1.3. Authentication

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

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

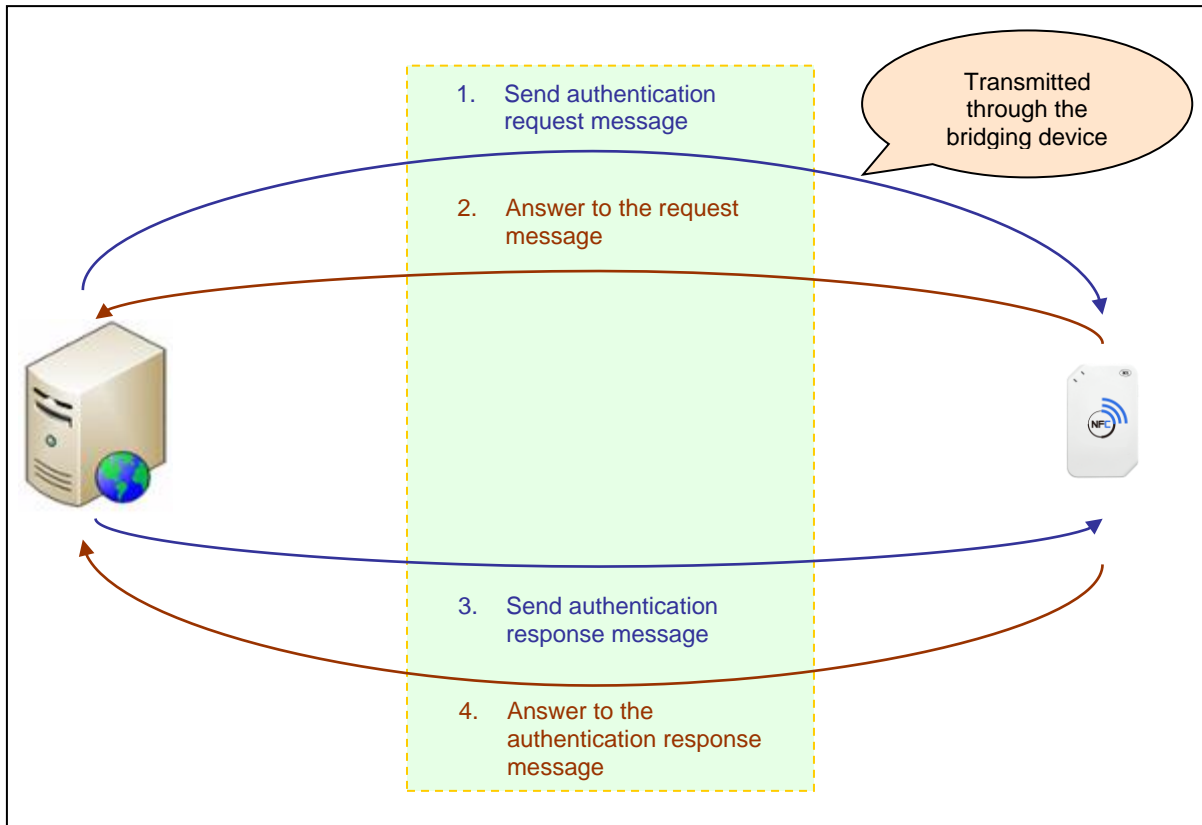


Figure 7: Authentication Procedure

After successful authentication, a 16-byte Session Key is generated in both the ACR1255U-J1 and the data server.

Default Customer Master Key (Hex): **41 43 52 31 32 35 35 55 2D 4A 31 20 41 75 74 68**

Note: The reader will be locked and unusable once incorrect authentication keys are entered more than six (6) times.

For more detailed information, you may contact an ACS sales representative



5.1.4. Communication Profile

The communication profile should be:

Start byte + Len + Datablock + Check + Stop byte

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



5.1.5. Frame Format

5.1.5.1. Data Frame Format

Command

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

The frame format should be:

CmdMessageType + *Length* + *Slot* + *Seq* + *Param* + *Checksum* + *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
<i>RspMessageType</i>	1	Response
<i>Length</i>	2	Data length (Big Endian)
<i>Slot</i>	1	Default value: 00h
<i>Seq</i>	1	Sequence
<i>Slot Status/Param</i>	1	Slot Status/Parameter
<i>Checksum</i>	1	Checksum
<i>Data</i>	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: 62 00 00 00 01 00 63 => Checksum = 63h



5.1.6. Bluetooth Communication Protocol

The 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



5.1.6.1. Card Power On

This command sends 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	00 00h	Data length
3	<i>Slot</i>	1	00h	-
4	<i>Seq</i>	1	00h	Sequence
5	<i>Param</i>	1	00h	Parameter
6	<i>Checksum</i>	1	-	Checksum 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>RspMessageType</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	-	Parameter Bit 0 as LSB Bit 0-1 = 00h = Card present (Active) = 01h = Card present (Inactive) = 02h = Card not present Bit 6 = "0" = No error = "1" = Error e.g. Param = 41h = 0100 0001b Error encountered while card is present (Inactive)
6	<i>Checksum</i>	1	-	Checksum means the XOR values of all bytes in the response
7	<i>Data</i>	N	-	Data

Example:

Response = 80 00 08 00 00 00 00 B3 **3B 83 80 01 41 07 00 44**

ATR = **3B 83 80 01 41 07 00 44**



5.1.6.2. Card Power Off

This command sends 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	00 00h	Data length
3	<i>Slot</i>	1	00h	-
4	<i>Seq</i>	1	00h	Sequence
5	<i>Param</i>	1	00h	Parameter
6	<i>Checksum</i>	1	-	Checksum 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>RspMessageType</i>	1	81h	-
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	-	Parameter Bit 0 as LSB Bit 0-1 = 00h = Card present (Active) = 01h = Card present (Inactive) = 02h = Card not present Bit 6 = "0" = No error = "1" = Error e.g. Param = 41h = 0100 0001b Error encountered while card is present (Inactive)
6	<i>Checksum</i>	1	-	Checksum means the XOR values of all bytes in the response
7	<i>Data</i>	N	-	Data



5.1.6.3. Get Slot Status

This command checks the presence of the inserted card.

Command Format

Offset	Field	Size	Value	Description
0	<i>CmdMessageType</i>	1	65h	-
1	<i>Length</i>	2	00 00h	Data length
3	<i>Slot</i>	1	00h	-
4	<i>Seq</i>	1	00h	Sequence
5	<i>Param</i>	1	00h	Parameter
6	<i>Checksum</i>	1	-	Checksum 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>RspMessageType</i>	1	81h	-
1	<i>Length</i>	2	00 00h	Data length
3	<i>Slot</i>	1	00h	-
4	<i>Seq</i>	1	00h	Sequence
5	<i>Param</i>	1	-	Card Status: Bit 0 as LSB Bit 0-1 = 00h = Card present (Active) = 01h = Card present (Inactive) = 02h = Card not present Bit 6 = "0" = No error = "1" = Error e.g. Param = 41h = 0100 0001b Error encountered while card is present (Inactive)
6	<i>Checksum</i>	1	-	Checksum means the XOR values of all bytes in the response
7	<i>Data</i>	N	-	Data



5.1.6.4. APDU Command

This command sends an APDU command to the reader.

Note: The “Param” option which supports Extended APDU is applicable to firmware version 2.01.00 and later.

Command Format

Offset	Field	Size	Value	Description
0	<i>CmdMessageType</i>	1	6Fh	-
1	<i>Length</i>	2	-	Data length, Max. length: 256 bytes
3	<i>Slot</i>	1	00h	-
4	<i>Seq</i>	1	00h	Sequence
5	<i>Param</i>	1	-	Parameter Short APDU level 00h – default Extended APDU level 00h – the command APDU begins and ends with this command. 01h – the command APDU begins with this command and continues in the next APDU command. 02h – this Data field continues a command APDU and ends the APDU command. 03h – the Data field continues a command APDU and another block is to follow. 10h – empty Data field, continuation of response APDU is expected in the next Response.
6	<i>Checksum</i>	1	-	Checksum 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>RspMessageType</i>	1	80h	-
1	<i>Length</i>	2	-	Data length
3	<i>Slot</i>	1	00h	-
4	<i>Seq</i>	1	00h	Sequence



Offset	Field	Size	Value	Description
5	<i>Param</i>	1	-	Parameter For Bit0-5, Bit 0 as LSB Short APDU level 00h – default Extended APDU level 00h – the response APDU begins and ends in this command. 01h – the response APDU begins with this command and is to continue. 02h – this Data field continues the response APDU and ends the response APDU. 03h – this Data field continues the response APDU and another block is to follow. 10h – empty Data field, continuation of the command APDU is expected in next Command. For Bit 6, it is used as error bit Bit 6 = “0” = No error = “1” = Error
6	<i>Checksum</i>	1	-	Checksum means the XOR values of all bytes in the response
7	<i>Data</i>	N	-	Data

Example:

Sends 600 bytes data to the card

- Command = 6F 01 00 00 XX 01 checksum (256 bytes data)
Response = 80 00 00 00 XX 10 checksum
- Command = 6F 01 00 00 XX 03 checksum (256 bytes data)
Response = 80 00 00 00 XX 10 checksum
- Command = 6F 00 58 00 XX 02 checksum (88 bytes data)
Response = 80 00 02 00 XX 00 checksum 90 00

Receives 600 bytes data from the card

- Command = 6F 00 07 00 XX 00 checksum 00 B0 87 00 00 02 58
Response = 80 01 00 00 XX 01 checksum (256 bytes data)
- Command = 6F 00 00 00 XX 10 checksum
Response = 80 01 00 00 XX 03 checksum (256 bytes data)
- Command = 6F 00 00 00 XX 10 checksum
Response = 80 00 5A 00 XX 02 checksum (88 bytes data) 90 00



5.1.6.5. Notify Card Status Command

This message notifies the card status by the reader.

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

Note: If message “52 00 00 00 00 01 53” is received, it means that the reader has entered sleep mode.



5.1.6.6. Hardware Error Response

This message returns an error message if an incorrect command is received.

Response Data Format

Offset	Field	Size	Value	Description
0	<i>RspMessageType</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 07h = Exceeded authentication retry error
6	<i>Checksum</i>	1	-	Checksum means the XOR values of all bytes in the message
7	<i>Data</i>	N	-	Data



5.1.6.7. Escape Command

This command gives access to the extended features of the reader.

Command Format

Offset	Field	Size	Value	Description
0	<i>CmdMessageType</i>	1	6Bh	-
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>Checksum</i>	1	-	Checksum 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>RspMessageType</i>	1	83h	-
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>Checksum</i>	1	-	Checksum means the XOR values of all bytes in the response
7	<i>Data</i>	N	-	Data

5.1.6.7.1. Escape Command Examples

1. Open automatic polling

Request command: **E0 00 00 40 01**

Response command: **E1 00 00 40 01**

Example:

Request: 6B 00 05 00 00 00 CF **E0 00 00 40 01**

Response: 83 00 05 00 00 00 66 **E1 00 00 40 01**

2. Close automatic polling

Request command: **E0 00 00 40 00**

Response command: **E1 00 00 40 00**

Example:

Request: 6B 00 05 00 00 00 CE **E0 00 00 40 00**

Response: 83 00 05 00 00 00 67 **E1 00 00 40 00**



3. 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 00 21 E1 00 00 45 00 77 59 E8 62
B7 80 0D 0A CE 9A 03 9B E9 48 EF 05

4. Authentication response

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

Response command: E1 00 00 46 00 + 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 00 51 E1 00 00 46 00 47 D5 50 54
F3 49 D4 17 B1 65 40 21 9B DA C9 B2



5.1.7. Mutual Authentication and Encryption Protocol

In Bluetooth mode, the **Bluetooth Communication Protocol** will be encrypted and transmitted after a successful mutual authentication.

5.1.7.1. Bluetooth Authentication Program Flow

As illustrated in **Authentication**, mutual authentication is used to avoid man-in-the-middle attacks. 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

5.1.7.2. SPH_to_RDR_ReqAuth

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

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

Offset	Field	Size	Value	Description	Encrypted
0	<i>bMessageType</i>	1	6Bh	-	No
1	<i>LEN1 LEN2 (wLength)</i>	2	00 05h	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	
7	<i>Data</i>	5	E0 00 00 45 00h	-	No

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.



5.1.7.3. RDR_to_SPH_AuthRsp1

This command is sent by the 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 (wLength)</i>	2	00 15h	Number of extra bytes in <i>abRndNum</i> 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	-	<i>wChecksum</i> 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	<i>abRndNum</i> [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



5.1.7.4. 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 is 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 random numbers. The overall 32-byte random numbers will be decrypted using the Customer Master Key and returned to the ACR1255U-J1 using this command 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	00 25h	The Number of extra bytes in the <i>abAuthData</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	-	<i>wChecksum</i> 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	<i>abAuthData</i> [0:15] – 16 bytes of random number generated by the data processing server. <i>abAuthData</i> [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 the paired device is correct. Otherwise, the response message will be RDR_to_SPH_ACK to provide the error information.



5.1.7.5. RDR_to_SPH_AuthRsp2

This command is sent by the 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 (wLength)</i>	2	00 15h	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	No
6	<i>wChecksum</i>	1	-	wChecksum means the XOR values of all bytes in the command	No
20	<i>abRndNum</i>	21	E1 00 00 46 00 + 16 bytes random number	abRndNum[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



5.1.7.6. 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	00 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	-	wChecksum 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
	07h	Exceeded Authentication retry error

Table 9: Mutual Authentication Error codes



6.0. Host Programming API

6.1. PC/SC API (For Windows®)

This section will describe some of the PCSC API for application programming usage. For more details, please refer to Microsoft MSDN Library or PCSC 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>

This function should be performed first before any other PCSC operation.

Example:

```
#define SCARD_SCOPE_USER 0

SCARDCONTEXT hContext;
int retCode;
void main ()
{
    // To establish the resource manager context and assign it to "hContext"
    retCode = SCardEstablishContext(SCARD_SCOPE_USER,
                                   NULL,
                                   NULL,
                                   &hContext);
    if (retCode != SCARD_S_SUCCESS)
    {
        // Establishing resource manager context failed
    }
    else
    {
        // Establishing resource manager context successful
        // Further PCSC operation can be performed
    }
}
```



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>

Example:

```
#define SCARD_SCOPE_USER 0

SCARDCONTEXT hContext; // Resource manager context
int retCode;
char readerName [256]; // List reader name

void main ()
{
    // To establish the resource manager context and assign to "hContext"
    retCode = SCardEstablishContext(SCARD_SCOPE_USER,
        NULL,
        NULL,
        &hContext);
    if (retCode != SCARD_S_SUCCESS)
    {
        // Establishing resource manager context failed
    }
    else
    {
        // Establishing resource manager context successful
        // List the available reader which can be used in the system
        retCode = SCardListReaders (hContext,
            NULL,
            readerName,
            &size);
        if (retCode != SCARD_S_SUCCESS)
        {
            // Listing reader fail
        }
        if (readerName == NULL)
        {
            // No reader available
        }
        else
        {
            // Reader listed
        }
    }
}
}
```



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>

Example:

```
#define SCARD_SCOPE_USER 0

SCARDCONTEXT      hContext;           // Resource manager context
SCARDHANDLE       hCard;             // Card context handle
unsigned long     dwActProtocol;     // Establish active protocol
int               retCode;
char              readerName [256]; // List reader name
char              rName [256];      // Reader name for connection

void main ()
{
    ...
    if (readerName == NULL)
    {
        // No reader available
    }
    else
    {
        // Reader listed
        rName = "ACS ACR1255U-J1 PICC Reader 0"; // Depends on what
                                                // reader be used
                                                // Should connect to
                                                // PICC interface

        retCode = SCardConnect(hContext,
                                rName,
                                SCARD_SHARE_SHARED,
                                SCARD_PROTOCOL_T0,
                                &hCard,
                                &dwActProtocol);
        if (retCode != SCARD_S_SUCCESS)
        {
            // Connection failed (May be because of incorrect reader
            // name, or no card was detected)
        }
        else
        {
            // Connection successful
        }
    }
}
```




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 **Peripherals Control** are using this API for sending.

Example:

```
#define SCARD_SCOPE_USER    0

#define EscapeCommand 0x310000 + 3500*4
SCARDCONTEXT      hContext;           // Resource manager context
SCARDHANDLE       hCard;              // Card context handle
unsigned long     dwActProtocol;      // Established active protocol
int               retCode;
char              readerName [256];   // Lists reader name
char              rName [256];       // Reader name for connection
BYTE              SendBuff[262],      // APDU command buffer
                 RecvBuff[262];      // APDU response buffer
BYTE              FWVersion [20],     // For storing firmware
                 version message
BYTE              ResponseData[50];   // For storing card response
DWORD             SendLen,            // APDU command length
                 RecvLen;            // APDU response length

void main ()
{
    ...
    rName = "ACS ACR1255U-J1 PICC Reader 0"; // Depends on what
                                              // reader will be used
                                              // Should connect to
                                              // PICC interface

    retCode = SCardConnect(hContext,
                           rName,
                           SCARD_SHARE_DIRECT,
                           SCARD_PROTOCOL_T0| SCARD_PROTOCOL_T1,
                           &hCard,
                           &dwActProtocol);
    if (retCode != SCARD_S_SUCCESS)
    {
        // Connection failed (may be because of incorrect reader
        // name, or no card was detected)
    }
    else
    {
        // Connection successful
        RecvLen = 262;
        // Get firmware version
        SendBuff[0] = 0xE0;
        SendBuff[1] = 0x00;
        SendBuff[2] = 0x00;
        SendBuff[3] = 0x18;
        SendBuff[4] = 0x00;
    }
}
```



```
SendLen = 5;
retCode = SCardControl ( hCard,
    EscapeCommand,
    SendBuff,
    SendLen,
    RecvBuff,
    RecvLen,
    &RecvLen);
if (retCode != SCARD_S_SUCCESS)
{
    // APDU sending failed
    return;
}
else
{
    // APDU sending successful
    // The RecvBuff stores the firmware version message.
    for (int i=0;i< RecvLen-5;i++)
    {
        FWVersion[i] = RecvBuff [5+i];
    }
}
// Connection successful
RecvLen = 262;

// Turn Green LED on, turn Red LED off
SendBuff[0] = 0xE0;
SendBuff[1] = 0x00;
SendBuff[2] = 0x00;
SendBuff[3] = 0x29;
SendBuff[4] = 0x01;
SendBuff[5] = 0x02; // Green LED On, Red LED off
SendLen = 6;
retCode = SCardControl ( hCard,
    EscapeCommand,
    SendBuff,
    SendLen,
    RecvBuff,
    RecvLen,
    &RecvLen);
if (retCode != SCARD_S_SUCCESS)
{
    // APDU sending failed
    return;
}
else
{
    // APDU sending success
}
```



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, **PICC Commands for MIFARE Classic (1K/4K) memory cards**, and **Pseudo APDU for Contactless Interface**) use this API for sending.

Example:

```
#define SCARD_SCOPE_USER      0

SCARDCONTEXT      hContext;          // Resource manager context
SCARDHANDLE       hCard;             // Card context handle
unsigned long     dwActProtocol;     // Established active protocol
int               retCode;
char              readerName [256];  // List reader name
char              rName [256];       // Reader name for connect
BYTE              SendBuff[262];     // APDU command buffer
BYTE              RecvBuff[262];     // APDU response buffer
BYTE              CardID [8];        // For storing the FeliCa IDM/
                                      MIFARE UID
BYTE              ResponseData[50];  // For storing card response
DWORD            SendLen,            // APDU command length
                 RecvLen;           // APDU response length
SCARD_IO_REQUEST  ioRequest;

void main ()
{
    ...
    rName = "ACS ACR1255U-J1 PICC Reader 0"; // Depends on what reader
                                              // should be used
                                              // Should connect to PICC
                                              // interface

    retCode = SCardConnect(hContext,
                           rName,
                           SCARD_SHARE_SHARED,
                           SCARD_PROTOCOL_T0,
                           &hCard,
                           &dwActProtocol);
    if (retCode != SCARD_S_SUCCESS)
    {
        // Connection failed (May be because of incorrect reader
        // name, or no card was detected)
    }
    else
    {
        // Connection successful
        ioRequest.dwProtocol = dwActProtocol;
        ioRequest.cbPciLength = sizeof(SCARD_IO_REQUEST);
        RecvLen = 262;
    }
}
```



```
// Get MIFARE UID/ FeliCa IDM
SendBuff[0] = 0xFF;
SendBuff[1] = 0xCA;
SendBuff[2] = 0x00;
SendBuff[3] = 0x00;
SendBuff[4] = 0x00;
SendLen = 5;
retCode = SCardTransmit( hCard,
                        &ioRequest,
                        SendBuff,
                        SendLen,
                        NULL,
                        RecvBuff,
                        &RecvLen);

if (retCode != SCARD_S_SUCCESS)
{
    // APDU sending failed
    return;
}
else
{
    // APDU sending successful
    // The RecvBuff stores the IDM for FeliCa / the UID for
    MIFARE.
    // Copy the content for further FeliCa access
    for (int i=0;i< RecvLen-2;i++)
    {
        CardID [i] = RecvBuff[i];
    }
}
```



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>

This function is used to end the PCSC Operation.

Example:

```
#define SCARD_SCOPE_USER 0

SCARDCONTEXT      hContext;          // Resource manager context
SCARDHANDLE       hCard;            // Card context handle
unsigned long     dwActProtocol;    // Established active protocol
int               retCode;

void main ()
{
    ...
    // Connection successful
    ...
    retCode = SCardDisconnect(hCard, SCARD_RESET_CARD);
    if (retCode != SCARD_S_SUCCESS)
    {
        // Disconnection failed
    }
    else
    {
        // Disconnection successful
    }
}
}
```

6.1.7. APDU Flow

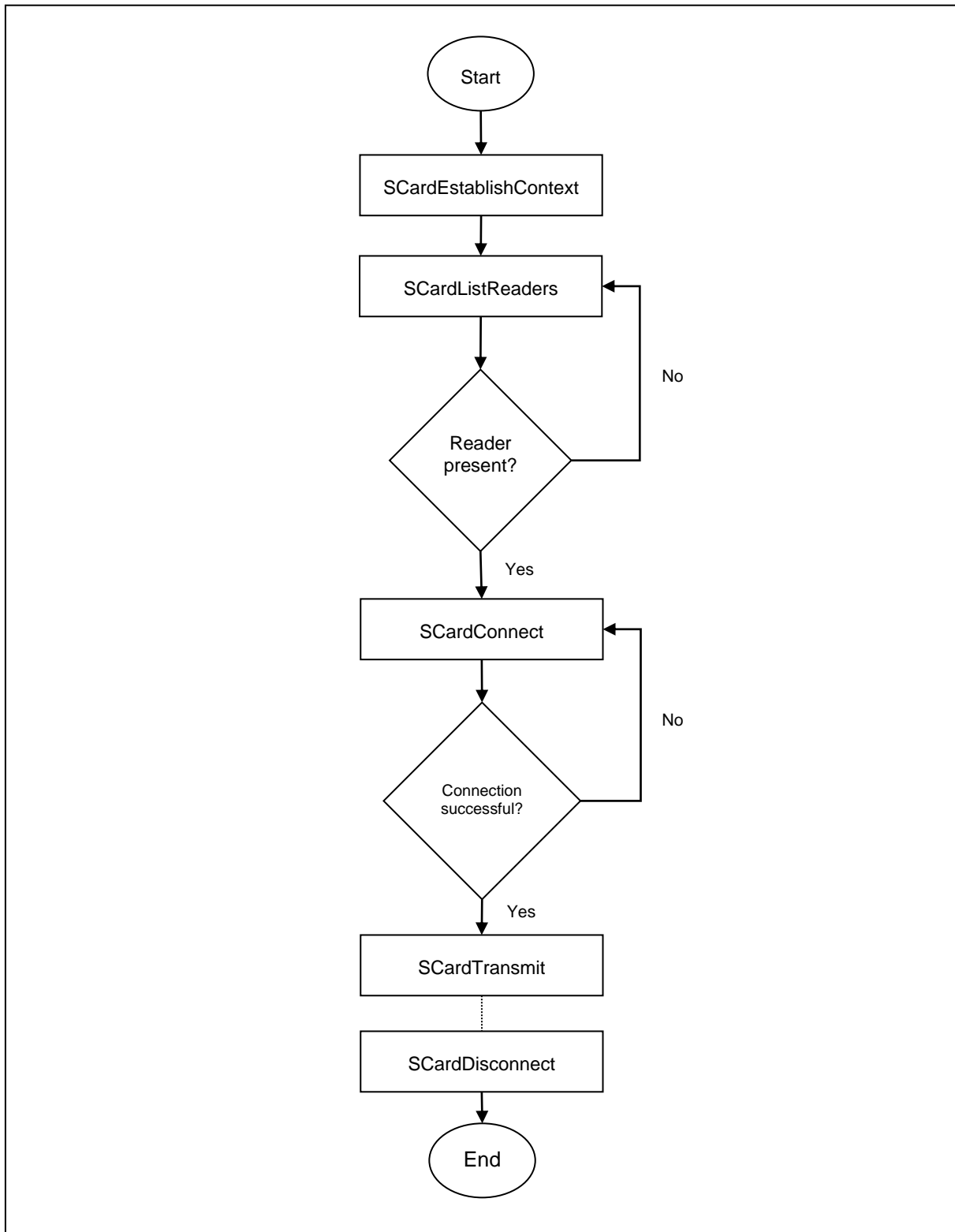


Figure 8: ACR1255U-J1 APDU Flow

6.1.8. Escape Command Flow

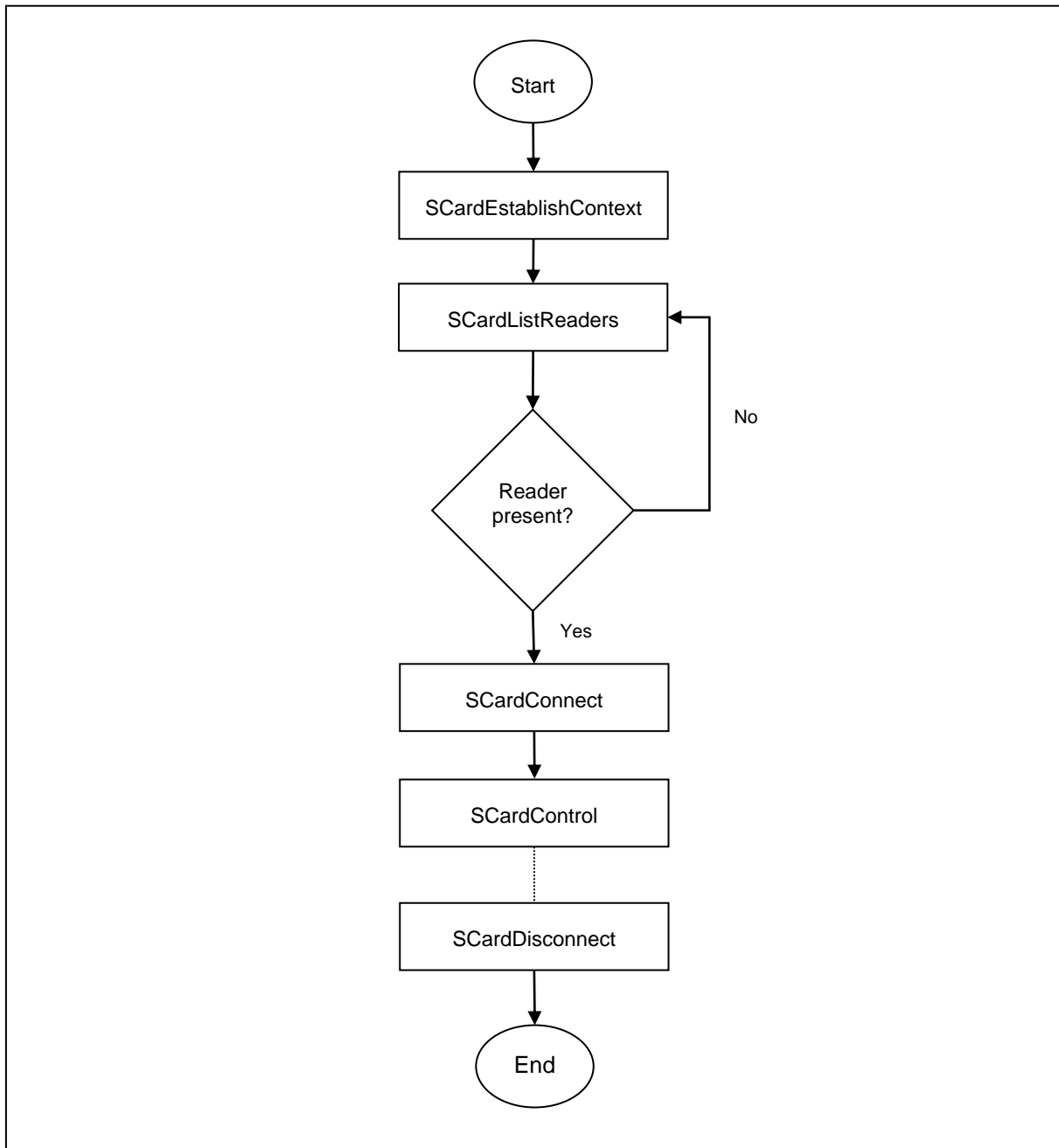


Figure 9: ACR1255U-J1 Escape Command Flow



6.2. Contactless Smart Card Protocol

6.2.1. ATR Generation

If the reader detects a PICC, an ATR will be sent to the PC/SC driver to identify 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
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 38: MIFARE Plus SL2 2K
00 02: MIFARE Classic 4K	00 39: MIFARE Plus SL2 4K
00 03: MIFARE Ultralight®	00 30: Topaz and Jewel
00 26: MIFARE Mini®	00 3B: FeliCa
00 3A: MIFARE Ultralight® C	FF 28: JCOP 30
00 36: MIFARE Plus®	FF [SAK]: undefined tags
SL12K	
00 37: MIFARE Plus SL1 4K	00 07: SRIX512

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 border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Byte1-4</th> <th>Byte5-7</th> <th>Byte8</th> </tr> </thead> <tbody> <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> </tbody> </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



6.2.2. Pseudo APDU for Contactless Interface

6.2.2.1. Get Data

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

Get UID APDU Format (5 bytes)

Command	CLA	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
Successful	90h	00h	The operation was 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 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};
```



6.2.2.2. Get PICC Data

This command returns the PICC data of the connected PICC.

Note: This is applicable only to firmware version 2.04.xx and later.

Get PICC Data APDU Format (5 Bytes)

Command	Class	INS	P1	P2	Le
Get PICC Data	FFh	CAh	00h	02h	00h

If Type A card, Get ATQA + UID + SAK Response Format (2 Bytes + 4/7/10 Bytes + 1 Byte + 2 Bytes)

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

If Type B card, Get ATQB (12 Bytes + 2 bytes)

Response	Data Out				
Result	ATQB			SW1	SW2

Response Codes

Results	SW1	SW2	Meaning
Successful	90h	00h	The operation was completed successfully
Error	63h	00h	The operation failed
Error	6Ah	81h	Function not supported



6.2.3. APDU commands for PC/SC 2.0 Part 3 (version 2.02 or later)

PC/SC 2.0 Part 3 commands are used to transparently pass data from an application to a contactless tag, return the received data transparently to the application and protocol, and switch the protocol simultaneously.

6.2.3.1. Command and Response APDU Format

Command Format

CLA	INS	P1	P2	Lc	Data In
FFh	C2h	00h	Function	DataLen	Data[DataLen]

Where:

Functions 1 byte
 00h = Manage Session
 01h = Transparent Exchange
 02h = Switch Protocol
 Other = RFU

Response Format

Data Out	SW1	SW2
Data Field BER-TLV encoded		

Every command returns SW1 and SW2 together with the response data field (if available). The SW1 SW2 is based on ISO 7816. SW1 SW2 from the C0 data object below should also be used.

C0 data element Format

Tag	Length (1 byte)	SW2
C0h	03h	Error Status

Error Status Description

Error Status	Description
XX SW1 SW2	XX = number of the bad data object in the APDU 00 = general error of APDU 01 = error in the 1 st data object 02 = error in the 2 nd data object
00 90 00h	No error occurred
XX 62 82h	Data object XX warning, requested information not available
XX 63 00h	No information
XX 63 01h	Execution stopped due to failure in other data object
XX 6A 81h	Data object XX not supported
XX 67 00h	Data object XX with unexpected length



Error Status	Description
XX 6A 80h	Data object XX with unexpected vale
XX 64 00h	Data Object XX execution error (no response from IFD)
XX 64 01h	Data Object XX execution error (no response from ICC)
XX 6F 00h	Data object XX failed, no precise diagnosis

The first value byte indicates the number of the erroneous data object XX, while the last two bytes indicate the explanation of the error. SW1 SW2 values based on ISO 7816 are allowed.

If there are more than one data objects in the C-APDU field and one data object failed, IFD can process the following data objects if they do not depend on the failed data objects.



6.2.3.2. Manage Session Command

This command is used to manage the transparent session. This includes starting and ending a transparent session. Through this command, you can also manage the operation environment and the capabilities of the IFD within the transparent session.

Manage Session Command

Command	CLA	INS	P1	P2	Lc	Data In
Manage Session	FFh	C2h	00h	00h	DataLen	DataObject (N bytes)

Where:

Data Object (1 byte)

Tag	Data Object
80h	Version Data Object
81h	Start Transparent Session
82h	End Transparent Session
83h	Turn Off RF Field
84h	Turn On RF Field
5F 46h	Timer
FF 6Dh	Get Parameter
FF 6Eh	Set Parameter

Manage Session Response Data Object

Tag	Data Object
C0h	Generic Error status
80h	Version data object
FF 6Dh	IFD parameter data object

6.2.3.2.1. Start Session Data Object

This command is used to start a transparent session. Once the session has started, auto-polling will be disabled until the session is ended.

Start Session Data Object

Tag	Length (1 byte)	Value
81h	00h	-



6.2.3.2.2. End Session Data Object

This command ends the transparent session. The auto-polling will be reset to the state before the session has started.

End Session Data Object

Tag	Length (1 byte)	Value
82h	00h	-

6.2.3.2.3. Version Data Object

This command returns the version number of the IFD handler.

Version Data Object

Tag	Length (1 byte)	Value		
80h	03h	Major	Minor	Build

6.2.3.2.4. Turn Off the RF Data Object

This command turns off the antenna field.

Turn off RF Field Data Object

Tag	Length (1 byte)	Value
83h	00h	-

6.2.3.2.5. Turn On the RF Data Object

This command turns on the antenna field.

Turn on the RF Field Data Object

Tag	Length (1 byte)	Value
84h	00h	-

6.2.3.2.6. Timer Data Object

This command creates a 32-bit timer data object in unit of 1 μ s.

Example: If there is timer data object with 5000 μ s between RF Turn Off Data Object and RF Turn On Data Object, the reader will turn off the RF field for about 5000 μ s before it is turned on.

Timer Data Object

Tag	Length (1 byte)	Value
5F 46h	04h	Timer (4 bytes)

6.2.3.2.7. Get Parameter Data Object

This command gets the different parameters from the IFD.

Get Parameter Data Object

Tag	Length (1 byte)	Value		
		Tag	Len	Value
FF 6Dh	Var	TLV_Objects		

TLV_Objects

Parameters Requested	Tag	Length
Frame size for IFD integer (FSDI)	01h	00h
Frame size for ICC integer (FSCI)	02h	00h
Frame waiting time integer (FWTI)	03h	00h
Max. Communication Speed supported by the IFD	04h	00h
Communication Speed of the ICC	05h	00h
Modulation Index	06h	00h
PCB for ISO/IEC14443	07h	00h
CID for ISO/IEC14443	08h	00h
NAD for ISO/IEC14443	09h	00h
Param 1 – 4 for for ISO/IEC14443 type B	0Ah	00h

6.2.3.2.8. Set Parameter Data Object

This command sets different parameters from the IFD.

Set Parameter Data Object

Tag	Length (1 byte)	Value		
		Tag	Len	Value
FF 6Eh	Var	TLV_Objects		

TLV_Objects

Parameters Requested	Tag	Length
Frame size for IFD integer (FSDI)	01h	01h
Frame size for ICC integer (FSCI)	02h	01h
Frame waiting time integer (FWTI)	03h	01h
Max. Communication Speed supported by the IFD	04h	01h
Communication Speed of the ICC	05h	01h
Modulation Index	06h	01h
PCB for ISO/IEC14443	07h	01h



Parameters Requested	Tag	Length
CID for ISO/IEC14443	08h	01h
NAD for ISO/IEC14443	09h	01h
Param 1 – 4 for for ISO/IEC14443 type B	0Ah	04h



6.2.3.3. Transparent Exchange Command

This command transmits and receives any bit or bytes from ICC.

Transparent Exchange Command

Command	CLA	INS	P1	P2	Lc	Data In
TranspEx	FFh	C2h	00h	01h	DataLen	DataObject (N bytes)

Where:

Data Object (1 byte)

Tag	Data Object
90h	Transmission and Reception Flag
91h	Transmission Bit Framing
92h	Reception Bit Framing
93h	Transmit
94h	Receive
95h	Transceive – Transmit and Receive
FF 6Dh	Get Parameter
FF 6Eh	Set Parameter

Transparent Exchange Response Data Object

Tag	Data Object
C0h	Generic Error status
92h	Number of valid bits in the last byte of received data
96h	Response Status
97h	ICC response
FF 6Dh	IFD parameter data object

6.2.3.3.1. Transmission and Reception Flag Data Object

This command defines the framing and RF parameters for the following transmission.

Transmission and Reception Flag Data Object

Tag	Length (1 byte)	Value	
		bit	Description
90h	02h	0	0 – append CRC in the transmit data 1 – do not append CRC in the transmit data
		1	0 – discard CRC from the received data 1 – do not discard CRC from the received data (i.e. no CRC checking)
		2	0 – insert parity in the transmit data 1 – do not insert parity
		3	0 – expect parity in received date 1 – do not expect parity (i.e. no parity checking)
		4	0 – append protocol prologue in the transmit data or discard from the response 1 – do not append or discard protocol prologue if any (e.g. PCB, CID, NAD)
		5-15	RFU

6.2.3.3.2. Transmission Bit Framing Data Object

This command defines the number of valid bits of the last byte of data to transmit or transceive.

Transmission bit Framing Data Object

Tag	Length (1 byte)	Value	
		bit	Description
91h	01h	0-2	Number of valid bits of the last byte (0 means all bits are valid)
		3-7	RFU

Transmission bit framing data object shall be together with “transmit” or “transceive” data object only. If this data object does not exist, it means all bits are valid.

6.2.3.3.3. Reception bit Framing Data Object

For the command APDU, this data object defines the number of expected valid bits of the last byte of data received.

For the response APDU, this data object mentions the number of valid bits in the last byte of received data.

Reception bit Framing Data Object

Tag	Length (1 byte)	Value	
		bit	Description
92h	01h	0-2	Number of valid bits of the last byte (0 means all bits are valid)
		3-7	RFU

If this data object does not exist, it means all bits are valid.

6.2.3.3.4. Transmit Data Object

This command transmits the data from IFD to the ICC. No response is expected from the ICC after transmit is complete.

Transmit Data Object

Tag	Length (1 byte)	Value
93h	DataLen	Data (N bytes)

6.2.3.3.5. Receive Data Object

This command forces the reader into receiving mode within the time, given in the following timer object.

Receive Data Object

Tag	Length (1 byte)	Value
94h	00h	-

6.2.3.3.6. Transceive Data Object

This command transmits and receives data from the ICC. After transmission is complete, the reader will wait until the time given in the timer data object.

If no timer data object was defined in the data field, the reader will wait for the duration given in the Set Parameter FWTI Data Object. If no FWTI is set, the reader will wait for about 302 μ s.

Transceive Data Object

Tag	Length (1 byte)	Value
95h	DataLen	Data (N Bytes)



6.2.3.3.7. Response Status Data Object

Inside the response, this command is used to notify the received data status.

Response Status Data Object

Tag	Length (1 byte)	Value		
		Byte 0		Byte 1
		Bit	Description	
96h	02h	0	0 – CRC is OK or no checked 1 – CRC check fail	If a collision is detected, these bytes will tell the collision position. Otherwise, "00h" will be shown.
		1	0 – no collision 1 – collision detected	
		2	0 – no parity error 1 – parity error detected	
		3	0 – no framing error 1 – framing error detected	
		4 - 7	RFU	

6.2.3.3.8. Response Data Object

Inside the response, this command is used to notify the received data status.

Response Data Object

Tag	Length (1 byte)	Value
97h	DataLen	ReplyData (N Byte)

6.2.3.4. Switch Protocol Command

This command specifies the protocol and different layers of the standard within the transparent session.

Switch Protocol Command

Command	CLA	INS	P1	P2	Lc	Data In
SwProtocol	FFh	C2h	00h	02h	DataLen	DataObject (N bytes)

Where:

Data Object (1 byte)

Tag	Data Object
8Fh	Switch Protocol Data Object
FF 6Dh	Get Parameter
FF 6Eh	Set Parameter

Switch Protocol Response Data Object

Tag	Data Object
C0h	Generic Error status
FF 6Dh	IFD parameter data object

6.2.3.4.1. Switch Protocol Data Object

This command specifies the protocol and different layers of the standard.

Switch Protocol Data Object

Tag	Length (1 byte)	Value	
		Byte 0	Byte 1
8Fh	02h	00h – ISO/IEC14443 Type A 01h – ISO/IEC14443 Type B 03h – FeliCa Other – RFU	00h – If no layer separation 02h – Switch to Layer 2 03h – Switch or activate to layer 3 04h – Activate to layer 4 Other - RFU

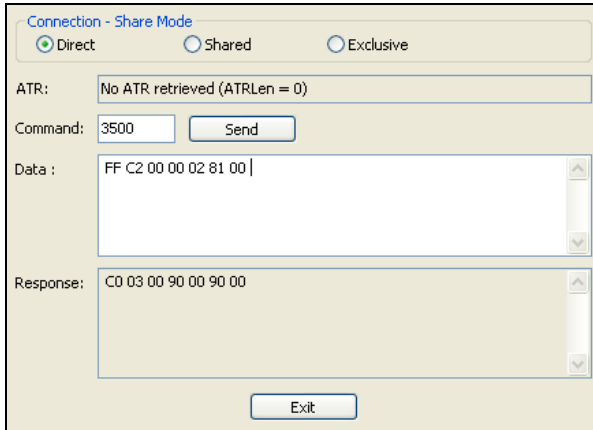


6.2.3.5. PC/SC 2.0 Part 3 Example

1. Start Transparent Session.

Command: **FF C2 00 00 02 81 00**

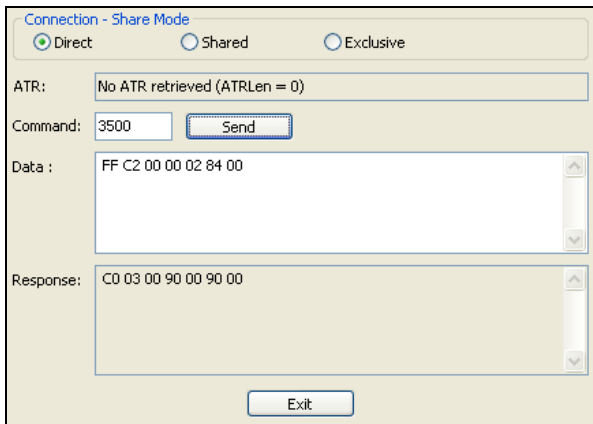
Response: **C0 03 00 90 00 90 00**



2. Turn the Antenna Field on.

Command: **FF C2 00 00 02 84 00**

Response: **C0 03 00 90 00 90 00**



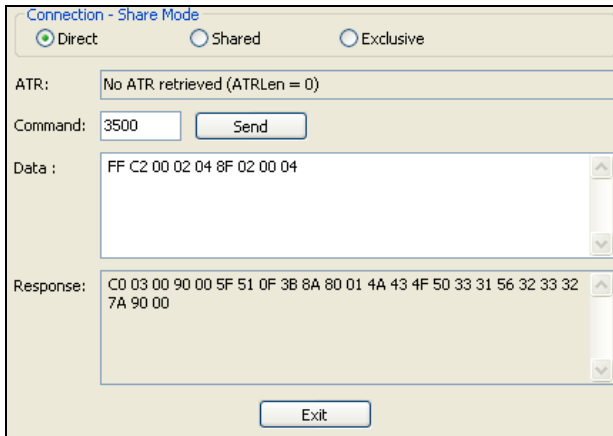


3. ISO 14443-4A Active.

Command: **FF C2 00 02 04 8F 02 00 04**

Response: **C0 03 01 64 01 90 00** (if no card present)

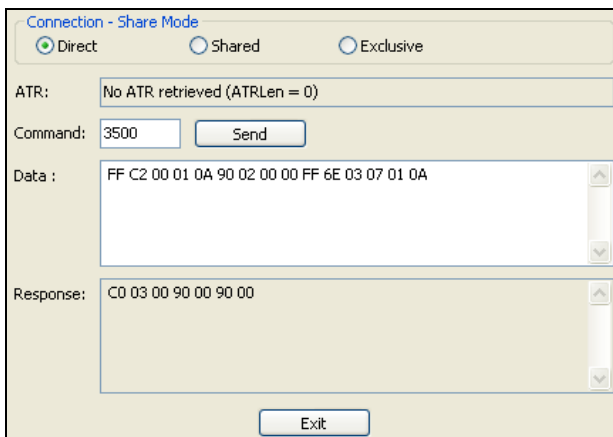
C0 03 00 90 00 5F 51 [ATR] 90 00



4. Set the PCB to 0Ah and enable the CRC, parity and protocol prologue in the transmit data.

Command: **FF C2 00 01 0A 90 02 00 00 FF 6E 03 07 01 0A**

Response: **C0 03 00 90 00 90 00**

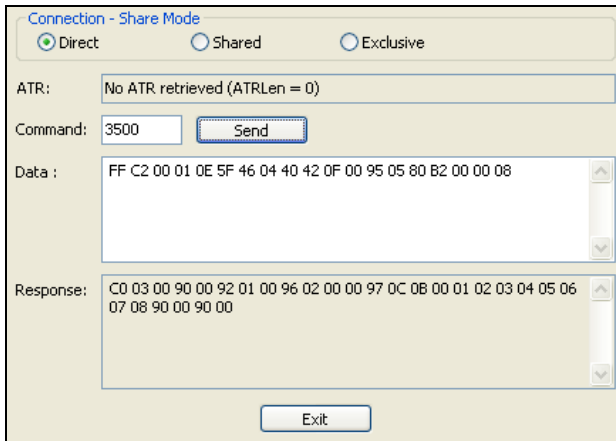




- 5. Send the APDU “80B2000008” to card and get response.

Command: **FF C2 00 01 0E 5F 46 04 40 42 0F 00 95 05 80 B2 00 00 08**

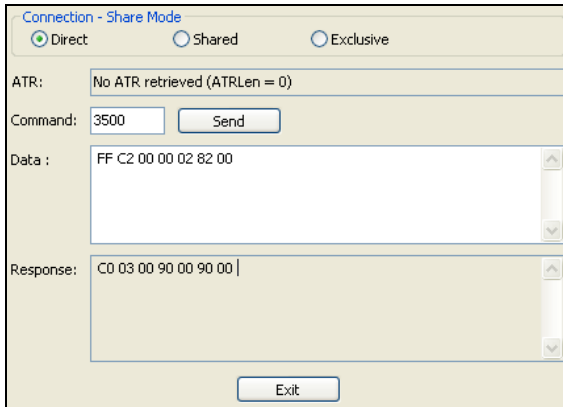
Response: **C0 03 00 90 00 92 01 00 96 02 00 00 97 0C [Card Response] 90 00**



- 6. End Transparent Session.

Command: **FF C2 00 00 02 82 00**

Response: **C0 03 00 90 00 90 00**





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

6.2.4.1. 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.

Load Authentication Keys APDU Format (11 bytes)

Command	CLA	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 = Volatile memory for storing a temporary key. The key will disappear once the reader is disconnected from the computer. Two volatile keys are provided. The volatile key can be used as a session key for different sessions.

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
Successful	90h	00h	The operation was completed successfully.
Error	63h	00h	The operation 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}



6.2.4.2. 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	CLA	INS	P1	P2	P3	Data In
Authentication	FFh	88h	00h	Block Number	Key Type	Key Number

Load Authentication Keys APDU Format (10 bytes)

Command	CLA	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 computer. 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
Successful	90	00h	The operation was completed successfully.
Error	63	00h	The operation failed.

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
Sector 1	04h – 06h	07h
..
..
Sector 14	38h – 0Ah	3Bh
Sector 15	3Ch – 3Eh	3Fh

} 1 KB

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
Sector 1	04h – 06h	07h
..
..
Sector 30	78h – 7Ah	7Bh
Sector 31	7Ch – 7Eh	7Fh

} 2 KB

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
Sector 33	90h – 9Eh	9Fh
..
..
Sector 38	E0h – EEh	EFh
Sector 39	F0h – FEh	FFh

} 2 KB

Table 11: MIFARE Classic 4K Memory Map



Byte Number	0	1	2	3	Page
Serial Number	SN0	SN1	SN2	BCC0	0
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

512 bits
or
64 bytes

Table 12: MIFARE Ultralight® 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}

Note: MIFARE Ultralight® does not need to do any authentication. The memory is free to access.

6.2.4.3. Read Binary Blocks

This command retrieves 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	CLA	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
Successful	90h	00h	The operation was completed successfully.
Error	63h	00h	The operation 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

6.2.4.4. Update Binary Blocks

This command writes 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	CLA	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
- 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
Successful	90h	00h	The operation was completed successfully.
Error	63h	00h	The operation 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



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

This command manipulates value-based transactions (e.g., increment a value of the value block).

Value Block Operation APDU Format (10 bytes)

Command	CLA	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
Successful	90h	00h	The operation was completed successfully.
Error	63h	00h	The operation failed.



6.2.4.6. Read Value Block

This command retrieves the value from a value block. This command is valid only for value blocks.

Read Value Block APDU Format (5 bytes)

Command	CLA	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
Successful	90h	00h	The operation was completed successfully.
Error	63h	00h	The operation failed.



6.2.4.7. Copy Value Block

This command copies a value from a value block to another value block.

Copy Value Block APDU Format (7 bytes)

Command	CLA	INS	P1	P2	Lc	Data In	
Copy Value Block	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
Successful	90h	00h	The operation was completed successfully.
Error	63h	00h	The operation 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



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

All ISO 14443-4 compliant cards (PICCs) understand the ISO 7816-4 APDUs. The ACR1255U-J1 just has to communicate with the ISO 14443-4 compliant cards by exchanging ISO 7816-4 APDUs and responses. The 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. Simply treat the MIFARE tags as standard ISO 14443-4 tags. For more information, please refer to **PICC Commands for MIFARE Classic (1K/4K) memory cards.**

ISO 7816-4 APDU Format

Command	CLA	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
Successful	90h	00h	The operation was completed successfully.
Error	63h	00h	The operation 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]



6.2.6. Accessing 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 (MIFARE® DESFire®)

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

Class = 90h; INS = 0Ah (MIFARE 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 the MIFARE® DESFire® specification. Please refer to the MIFARE® 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 MIFARE® 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]



6.2.7. Accessing FeliCa tags

For FeliCa access, the command is different from that of PCSC-compliant and MIFARE tags. The command follows the FeliCa specification with an added header.

FeliCa Command Format

Command	CLA	INS	P1	P2	Lc	Data In
FeliCa Command	FFh	00h	00h	00h	Length of the Data In	FeliCa Command (start with Length Byte)

FeliCa Response Format (Data + 2 bytes)

Response	Data Out
Result	Response Data

Read Memory Block Example:

1. Connect the FeliCa.

The ATR = 3B 8F 80 01 80 4F 0C A0 00 00 03 06 **11 00 3B** 00 00 00 00 42h

In which, **11 00 3Bh** = FeliCa

2. Read FeliCa IDM.

CMD = FF CA 00 00 00h

RES = [IDM (8bytes)] 90 00h

e.g., FeliCa IDM = 01 01 06 01 CB 09 57 03h

3. FeliCa command access.

Example: "Read" Memory Block.

CMD = FF 00 00 00 10 10 06 **01 01 06 01 CB 09 57 03** 01 09 01 01 80 00h

where:

Felica Command = 10 06 **01 01 06 01 CB 09 57 03** 01 09 01 01 80 00h

IDM = **01 01 06 01 CB 09 57 03h**

RES = Memory Block Data



6.3. Peripherals Control

The reader's peripherals control commands are implemented by using Escape Command (0x6B) in Bluetooth mode or PC_to_RDR_Escape in USB mode.

For PC-Linked Mode:

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

where *abData* is the command

For Bluetooth Mode:

Command Format

Offset	Field	Size	Value	Description
0	<i>CmdMessageType</i>	1	6Bh	-
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>Checksum</i>			Checksum means the XOR values of all bytes in the command
7	<i>Data</i>	N	-	Data

where *Data* is the command



6.3.1. Get Firmware Version

This command gets the reader's firmware message.

Get Firmware Version Format (5 bytes)

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

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

Response	CLA	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"



6.3.2. Get Serial Number

This command gets the reader's serial number.

Get Serial Number Format (5 bytes)

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

Get Serial Number Response Format (N bytes)

Response	CLA	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 34 33 31 2D 30 30 30 30 31 36**

Serial Number (HEX) = **52 52 34 33 31 2D 30 30 30 30 31 36**

Serial Number (ASCII) = "RR431-000016"



6.3.3. LED Control

This command controls the LED's output.

LED Control Format (6 bytes)

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

LED Control Response Format (6 bytes)

Response	CLA	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



6.3.4. LED Status

This command checks for the existing LED's status.

LED Status Format (5 bytes)

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

LED Status Response Format (6 bytes)

Response	CLA	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



6.3.5. Buzzer Control

This command controls the buzzer output.

Buzzer Control Format (6 bytes)

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

Where:

Buzzer On Duration 1 byte

01 to FFh = Duration (unit: 10 ms)

Buzzer Control Response Format (6 bytes)

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



6.3.6. Set LED and Buzzer Status Indicator Behavior

This command sets the behaviors of the LEDs and the buzzer as status indicators.

Note: *The setting will be saved into non-volatile memory. (Starts from firmware version 2.03.xx)
Bit4 and Bit5 behavior options are available only in firmware version 2.04.xx and later.*

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

Command	CLA	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	Battery Charging Status LED	To show Battery Charging Status 1 = Enabled; 0 = Disabled
Bit 1	PICC Polling Status LED	To show the PICC Polling Status 1 = Enabled; 0 = Disabled
Bit 2	PICC Activation Status LED	To show the activation status of the PICC interface 1 = Enabled; 0 = Disabled
Bit 3	Card Insertion Events Buzzer	To make a beep whenever a card insertion event is detected 1 = Enabled; 0 = Disabled
Bit 4	Card Removal Events Buzzer	To make a beep whenever a card removal event is detected 1 = Enabled; 0 = Disabled <i>(Bit 3 should be enabled first, in order for this function to take effect)</i>
Bit 5	Reader Power ON Buzzer	To make a beep when the reader is powered ON 1 = Enabled; 0 = Disabled
Bit 6	RFU	
Bit 7	Card Operation Blinking LED	To light up the LED whenever the card is being accessed

Notes:

- (1) *Default value of behavior = BFh.*
- (2) *When in either USB or BT Mode, Red LED is always ON while charging, unless Charging Status Setting is turned OFF or reader is in fully charged state.*
- (3) *When in BT Mode, Blue LED (LED 2) is always BLINKING and cannot be changed.*
- (4) *When in USB Mode, Green LED (LED 1) is always ON, unless Polling Status Setting is turned OFF.*



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

Response	CLA	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	01h	Default Behavior



6.3.7. Read LED and Buzzer Status Indicator Behavior

This command reads the current default behaviors of the LEDs and the buzzer.

Note: *Bit4 and Bit5 behavior responses are available only in firmware version 2.04.xx and later.*

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

Command	CLA	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	CLA	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	01h	Behavior

Behavior (1 byte)

Behavior	Mode	Description
Bit 0	Battery Charging Status LED	To show Battery Charging Status 1 = Enabled; 0 = Disabled
Bit 1	PICC Polling Status LED	To show the PICC Polling Status 1 = Enabled; 0 = Disabled
Bit 2	PICC Activation Status LED	To show the activation status of the PICC interface 1 = Enabled; 0 = Disabled
Bit 3	Card Insertion Events Buzzer	To make a beep whenever a card insertion event is detected 1 = Enabled; 0 = Disabled
Bit 4	Card Removal Events Buzzer	To make a beep whenever a card removal event is detected 1 = Enabled; 0 = Disabled <i>(Bit 3 should be enabled first, in order for this function to take effect)</i>
Bit 5	Reader Power ON Buzzer	To make a beep when the reader is powered ON 1 = Enabled; 0 = Disabled
Bit 6	RFU	
Bit 7	Card Operation Blinking LED	To light up the LED whenever the card is being accessed

Note: *Default value of Behavior = BFh.*



6.3.8. Set Automatic PICC Polling

This command sets the reader's polling mode.

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

A command may be sent to disable the PICC polling function. The command is sent through the PC/SC 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. (Starts from firmware version 2.03.xx)

Set Automatic PICC Polling Format (6 bytes)

Command	CLA	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	CLA	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 = Enabled; 0 =Disabled
Bit 1	Turn off Antenna Field if no PICC found	1 = Enabled; 0 =Disabled
Bit 2	Turn off Antenna Field if the PICC is inactive	1 = Enabled; 0 =Disabled
Bit 3	RFU	-
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 ISO 14443A Part 4	1= Enabled; 0= Disabled

Note: Default value of Polling Setting = 8Bh.



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 130 mA.*

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.*



6.3.9. Read Automatic PICC Polling

This command checks the current PICC polling setting.

Read Automatic PICC Polling Format (5 bytes)

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

Read Automatic PICC Polling Response Format (6 bytes)

Response	CLA	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 = Enabled; 0 =Disabled
Bit 1	Turn off Antenna Field if no PICC found	1 = Enabled; 0 =Disabled
Bit 2	Turn off Antenna Field if the PICC is inactive	1 = Enabled; 0 =Disabled
Bit 3	RFU	-
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 ISO 14443A Part 4	1= Enabled; 0= Disabled

Note: Default value of Polling Setting = 8Bh.



6.3.10. Set PICC Operating Parameter

This command sets the PICC operating parameter.

Note: The setting will be saved into non-volatile memory. (Starts from firmware version 2.03.xx)

Set the PICC Operating Parameter Format (6 bytes)

Command	CLA	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	CLA	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	ISO 14443 Type A	The Tag Types to be detected during PICC Polling.	1 = Detect 0 = Skip
Bit 1	ISO 14443 Type B		1 = Detect 0 = Skip
Bit 2	FeliCa 212 Kbps		1 = Detect 0 = Skip
Bit 3	FeliCa 424 Kbps		1 = Detect 0 = Skip
Bit 4	Topaz		1 = Detect 0 = Skip
Bit 5	Calypso		1 = Detect 0 = Skip
Bit 6	SRIX		1 = Detect 0 = Skip
Bit 7	RFU	RFU	RFU

Note: Default value of Operation Parameter = 7Fh.



6.3.11. Read PICC Operating Parameter

This command checks the current PICC operating parameter.

Read the PICC Operating Parameter Format (5 bytes)

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

Read the PICC Operating Parameter Response Format (6 bytes)

Response	CLA	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	ISO 14443 Type A	The Tag Types to be detected during PICC Polling.	1 = Detect 0 = Skip
Bit 1	ISO 14443 Type B		1 = Detect 0 = Skip
Bit 2	FeliCa		1 = Detect 0 = Skip
Bit 3	FeliCa 424 Kbps		1 = Detect 0 = Skip
Bit 4	Topaz		1 = Detect 0 = Skip
Bit 5	Calypso		1 = Detect 0 = Skip
Bit 6	SRIX		1 = Detect 0 = Skip
Bit 7	RFU	RFU	RFU

Note: Default value of Operation Parameter = 7Fh.



6.3.12. 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.

Note: The setting will be saved into non-volatile memory. (Starts from firmware version 2.03.xx)

Set Auto PPS Format (7 bytes)

Command	CLA	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	CLA	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

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 exchange information 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.



6.3.13. Read Auto PPS

This command checks the current Auto PPS Setting.

Read Auto PPS Format (5 bytes)

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

Read Auto PPS Response Format (9 bytes)

Response	CLA	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

Note: Default setting is 00h.



6.3.14. Antenna Field Control

This command turns on/off the antenna field.

Antenna Field Control Format (6 bytes)

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

Antenna Field Control Response Format (6 bytes)

Response	CLA	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 before turning off the antenna field.



6.3.15. Read Antenna Field Status

This command checks the current Antenna Field Status.

Read Antenna Field Status Format (5 bytes)

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

Read Antenna Field Status Response Format (6 bytes)

Response	CLA	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) Successful, i.e. Contactless Tag Detected]
 - 03h = PICC Selected [PICC Select (Ref to ISO 14443) Successful]
 - 04h = PICC Activate [PICC Activation (Ref to ISO 14443) Successful, Ready for APDU Exchange]



6.3.16. Set Sleep Mode Option

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

This command sets the time interval before entering sleep mode.

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

Set Sleep Time Interval Format (5 bytes)

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

Set Sleep Time Interval Response Format (6 bytes)

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

Where:

- Time** 1 byte
- 00h = 60 seconds
- 01h = 90 seconds
- 02h = 120 seconds
- 03h = 180 seconds
- 04h = No Sleep



6.3.17. Read Sleep Mode Option

This command checks the sleep mode time interval.

Note: This is applicable only to firmware version 2.03.xx and later.

Read Sleep Time Interval Format (5 bytes)

Command	CLA	INS	P1	P2	Lc
Read Sleep Time Interval	E0h	00h	00h	50h	00h

Read Sleep Time Interval Response Format (6 bytes)

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

Where:

- Time** 1 byte
- 00h = 60 seconds
- 01h = 90 seconds
- 02h = 120 seconds
- 03h = 180 seconds
- 04h = No Sleep



6.3.18. Change Tx Power

This command sets the Bluetooth transmission power.

Note: The setting will be saved into non-volatile memory. (Starts from firmware version 2.03.xx)

Change Tx Power Format (5 bytes)

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

Change Tx Power Response Format (5 bytes)

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

Where:

Tx Power 1 byte

00h = -23 dBm (default), Distance: ~3 meters

01h = -6 dBm, Distance: ~7 meters

02h = 0 dBm, Distance: ~17 meters

03h = 4 dBm, Distance: ~25 meters



6.3.19. Read Tx Power Value

This command checks the Bluetooth transmission power.

Note: This is applicable only to firmware version 2.03.xx and later.

Read Tx Power Value Format (5 bytes)

Command	CLA	INS	P1	P2	Lc
Read Tx power	E0h	00h	00h	51h	00h

Read Tx Power Value Response Format (6 bytes)

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

Where:

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



6.3.20. Customer Master Key Rewrite

This command sets the customer master key.

Customer Master Key reset command format (5 bytes)

Command	CLA	INS	P1	P2	Lc
Customer Master Key reset	E0h	00h	00h	60h	00h

Customer Master Key reset Response format (21 bytes) (successful)

Response	CLA	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	10h	16 bytes random numbers

Customer Master Key rewrite command format (36 bytes)

Command	CLA	INS	P1	P2	Lc
Customer Master Key rewrite	E0h	00h	00h	61h	16 bytes encrypted random numbers + 16 bytes encrypted New Master Key

Customer Master Key rewrite Response format (7 bytes) (successful)

Response	CLA	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	02h	90 00h

Customer Master Key rewrite Response format (5 bytes) (failed)

Response	CLA	INS	P1	P2	Le
Result	E1h	00h	00h	00h	00h

Example: To rewrite customer master key.

The random number generated by the reader is "11 22 33 44 55 66 77 88 99 00 11 22 33 44 55 66"

Peer → Reader: (Customer Master Key Reset Request)

Frame1 "E0 00 00 60 00"

Reader → Peer: (Customer Master Key Reset Command Response)

Frame1 "E1 00 00 00 10 11 22 33 44 55 66 77 88 99 00 11 22 33 44 55 66"



6.3.21. Get Battery Level

This command checks the current battery level.

Note: This is applicable only to firmware version 2.03.xx and later, and when the reader is in Bluetooth mode.

Get Battery Level Format (5 bytes)

Command	CLA	INS	P1	P2	Lc
Get Battery Level	E0h	00h	00h	52h	00h

Get Battery Level Response Format (6 bytes)

Response	CLA	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	01h	Battery Level

Where:

- Battery Level 1 byte
 - 100 = 100% battery
 - 90 = 90% battery
 - 80 = 80% battery
 - 70 = 70% battery
 - 60 = 60% battery
 - 50 = 50% battery
 - 40 = 40% battery
 - 30 = 30% battery
 - 20 = 20% battery
 - 15 = 15% battery



6.3.22. Read PICC Type

This command checks the current PICC Type.

Note: This is applicable only to firmware version 2.04.xx and later.

Read PICC Type Format (5 bytes)

Command	CLA	INS	P1	P2	Lc
Read PICC Type	E0h	00h	00h	35h	00h

Read PICC Type Response Format (6 bytes)

Response	CLA	INS	P1	P2	Le	Data Out	
Result	E1h	00h	00h	00h	02h	Card Type	Card Status

Where:

Card Type

1 byte

CCh = Absent

04h = Topaz

10h = Mifare

11h = Felica 212 Kbps

12h = Felica 424 Kbps

20h = ISO 14443-4 Type A

23h = ISO 14443-4 Type B

25h = Calypso

28h = Srix

Card Status

1 byte

00h = PICC PowerOff [Contactless Tag Not Detected]

Other = PICC Detected [Contactless Tag Detected]

Android is a trademark of Google LLC.

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.

Microsoft is a registered trademark of Microsoft Corporation in the United States and/or other countries.

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