Advanced Card Systems Ltd.
Card & Reader Technologies

ACR1252U
NFC Forum Certified Reader

Application Programming Interface V1.15

info@acs.com.hk
www.acs.com.hk

Revision History

Release Date	Revision Description	Version Number
2014-01-22	• Initial Release	1.00
2014-04-11	• Updated document formatting • Updated Section 1.0: Introduction • Updated Section 2.0: Features • Updated Section 3.0: Architecture • Updated Section 5.0 Software Design, PCSC API, Command Set, Firmware Upgrade Procedure • Updated Sections 5.8.6 and 5.8.7: LED and Buzzer Status Indicator Behavior for PICC Interface • Updated Section 5.9 NFC Peer to Peer Mode Commands • Added Appendix A: SNEP Message • Modified outline for NFC Peer-to-Peer and Card Emulation Related Commands • Added brand trademark attributions	1.01
2014-05-20	• Added Section 3.0: Acronyms and Abbreviations • Updated Section 5.9: NFC Peer to Peer Mode Commands	1.02
2014-08-27	• Added Section 5.10: NFC Card Emulation Commands	1.03
2014-10-17	• Updated Section 2.0: Features	1.04
2014-10-30	• Added Section 5.3.1: Get Data Command for Contactless Interface	1.05
2015-02-09	• Added Section 5.4: PCSC 2.0 commands • Updated Section 5.6: Accessing PCSC-Compliant Tags (ISO14443-4) • Updated Section 5.8: Peripherals Control • Modified WUP Response command • Updated ATR response according to latest PCSC Specification	1.06
2015-03-28	• Updated Document Versioning and Format	1.07

Release Date	Revision Description	Version Number

2015-11-10	• Changed Document Title to API-ACR1252U
	• Changed all instances of ACR1252-A1 to ACR1252
	• Updated photo
	• Added REACH Certification
	• Added Serial Number commands:
	  ○ 5.8.14. Set Serial Number
	  ○ 5.8.15. Set and lock Serial Number
	  ○ 5.8.16. Read Serial Number
	  ○ 5.8.17. Unlock Serial Number
	• Added description on serial number command
		1.08

2016-04-18	• Added Sections 5.8.12 & 5.8.13: Set Auto PPS and Read Auto PPS • Added Appendix B for Escape Command Example • Updated Sections 5.8.8 & 5.8.9: Set and Read Automatic PICC Polling	1.09
2018-09-11	• Updated Section 2.0 Features	1.10
2019-08-05	• Updated Section 5.8.8 Set Automatic PICC Polling • Updated Section 5.8.9 Read Automatic PICC Polling	1.11
2019-09-10	• Updated MIFARE Trademarks • Re-organized Section 5.0 • Added Examples in Section 5.1 PCSC API • Updated Section 5.2.4.7 Copy Value Block • Added Section 5.3 Contact Smart Card Protocol • Added Section 5.3.1 ACOS6-SAM Commands	1.12
2019-10-15	• Updated Section 5.6.4 Set Card Emulation of MIFARE Ultralight UID • Updated Section 5.6.5 Set Card Emulation FeliCa IDm • Updated Section 5.6.6 Set Card Emulation Lock Data in NFC	1.13
2019-12-02	• Updated Section 5.1 PCSC API	1.14

Release Date	Revision Description	Version Number
2020-04-13	• Added section 5.2.2.2 Get PICC Data • Added Section 5.4.12 Set PICC Operating Parameter (Extended) • Added Section 5.4.13 Read PICC Operating Parameter (Extended) • Added Section 5.4.20 Read PICC Type • Updated Section 5.4.6 Set LED and Buzzer Status Indicator Behavior for PICC Interface • Updated Section 5.4.7 Read LED and Buzzer Status Indicator Behavior for PICC Interface • Updated Section 5.4.9 Read Automatic PICC Polling • Updated Section 5.4.10 Set PICC Operating Parameter • Updated Section 5.4.11 Read PICC Operating Parameter • Updated Section 5.4.14 Set Auto PPS • Updated Section 5.4.15 Read Auto PPS • Updated Section 5.4.18 Set and Lock Serial Number	1.15

Table of Contents

List of Figures

List of Tables

1.0. Introduction

The ACR1252U NFC Forum Certified reader is a USB PC-linked contactless card reader/writer with a SAM (Secure Access Module) slot, which can be used together with a SAM card for high level security in contactless transactions. It is an NFC card reader/writer that also supports card emulation and peer-to-peer communication modes. The ACR1252U reader is also NFC library compliant to support Bluetooth and Wi-Fi NFC pairing/log-in.

The ACR1252U is compliant to ISO 14443 Parts 1 to 4 supporting contactless card, MIFARE® cards, FeliCa cards and ISO 18092 NFC tags.

The ACR1252U has two reader interfaces, namely the PICC and SAM interface. Both interfaces follow the PC/SC specifications. This API document will discuss in detail how the PC/SC APDU commands were implemented for the contactless interface and device peripherals of the ACR1252U.

2.0. Features

- USB Full Speed Interface
- CCID-compliant
- Smart Card Reader:
  - Contactless Interface:
    - Read/Write speed of up to 424 Kbps
    - Built-in antenna for contactless tag access, with card reading distance of up to 50 mm (depending on tag type)
    - Supports ISO 14443 Part 4 Type A and B cards, MIFARE Classic®, FeliCa, and all four types of NFC (ISO/IEC 18092 tags)
    - Built-in anti-collision feature (only one tag is accessed at any time)
    - Supports extended APDU (max. 64 KB)
    - NFC Support:
      - Card reader/writer mode
      - Peer-to-Peer mode
      - Card Emulation mode
  - SAM Interface:
    - One SAM Slot
    - Supports ISO 7816-compliant Class A SAM cards
- Built-in Peripherals:
  - User-controllable bi-color LED
  - User-controllable buzzer
- Application Programming Interface:
  - Supports PC/SC
  - Supports CT-API (through wrapper on top of PC/SC)
- USB Firmware Upgradeability
- Supports Android™ 3.1 and later[1]
- Compliant with the following standards:
  - EN 60950/IEC 60950
  - ISO 18092
  - ISO 14443
  - ISO 7816 Class A (for SAM slot)
  - NFC Forum Certification Mark
  - FeliCa Performance Certification
  - PC/SC
  - CCID
  - CE
  - FCC
  - RoHS
  - REACH
  - J-LIS (Japan)
  - VCCI (Japan)
  - MIC (Japan)
  - KC (Korea)
  - Microsoft® WHQL

[1] *Uses an ACS-defined Android Library*

3.0. Acronyms and Abbreviations

Acronym/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
Gi	Optional information field for Initiator
PFB	Control information for transaction
FSL	maximum value for the Frame Length
LLCP	Logical Link Control Protocol

Table 1: Acronyms and Abbreviations

4.0. Architecture

For communication architecture, the protocol used between the ACR1252U and the computer is the CCID protocol. All communications between PICC and SAM are PC/SC compliant.

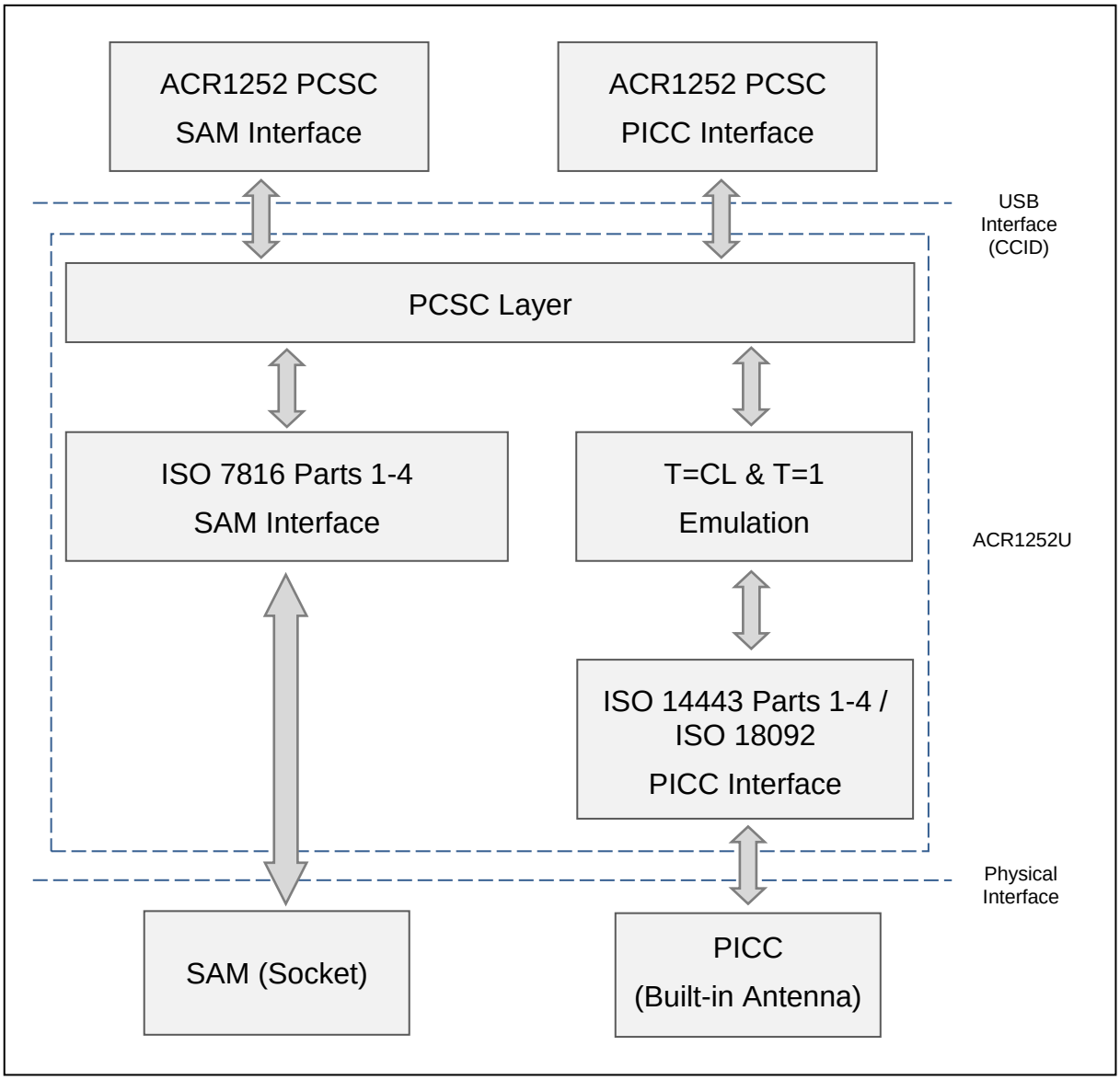


Figure 1: ACR1252U Architecture

5.0. Host Programming (PC-linked) API

5.1. PCSC API

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.

5.1.1. 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
   }
}
```

5.1.2. 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
      }
   }
}
```

5.1.3. 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 ACR1252 1S CL Reader PICC 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
         }
      }
   }
```

5.1.4. 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*** *use 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 ACR1252 1S CL Reader PICC 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
            }
```

5.1.5. 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
                  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 ACR1252 1S CL Reader PICC 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];
   }
}
```

5.1.6. 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
      }
   }
}
```

5.1.7. APDU Flow

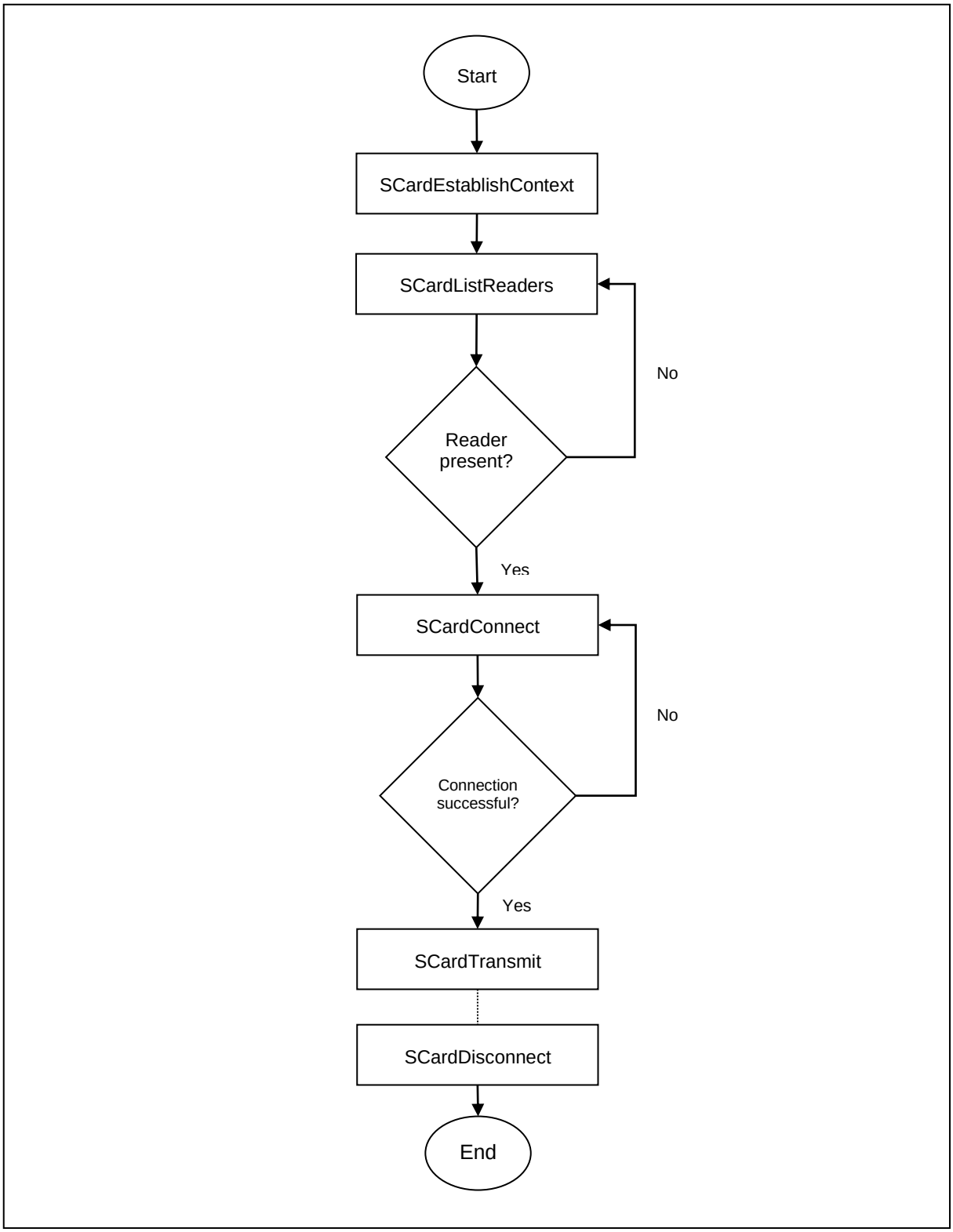


Figure 2: ACR1252U APDU Flow

5.1.8. Escape Command Flow

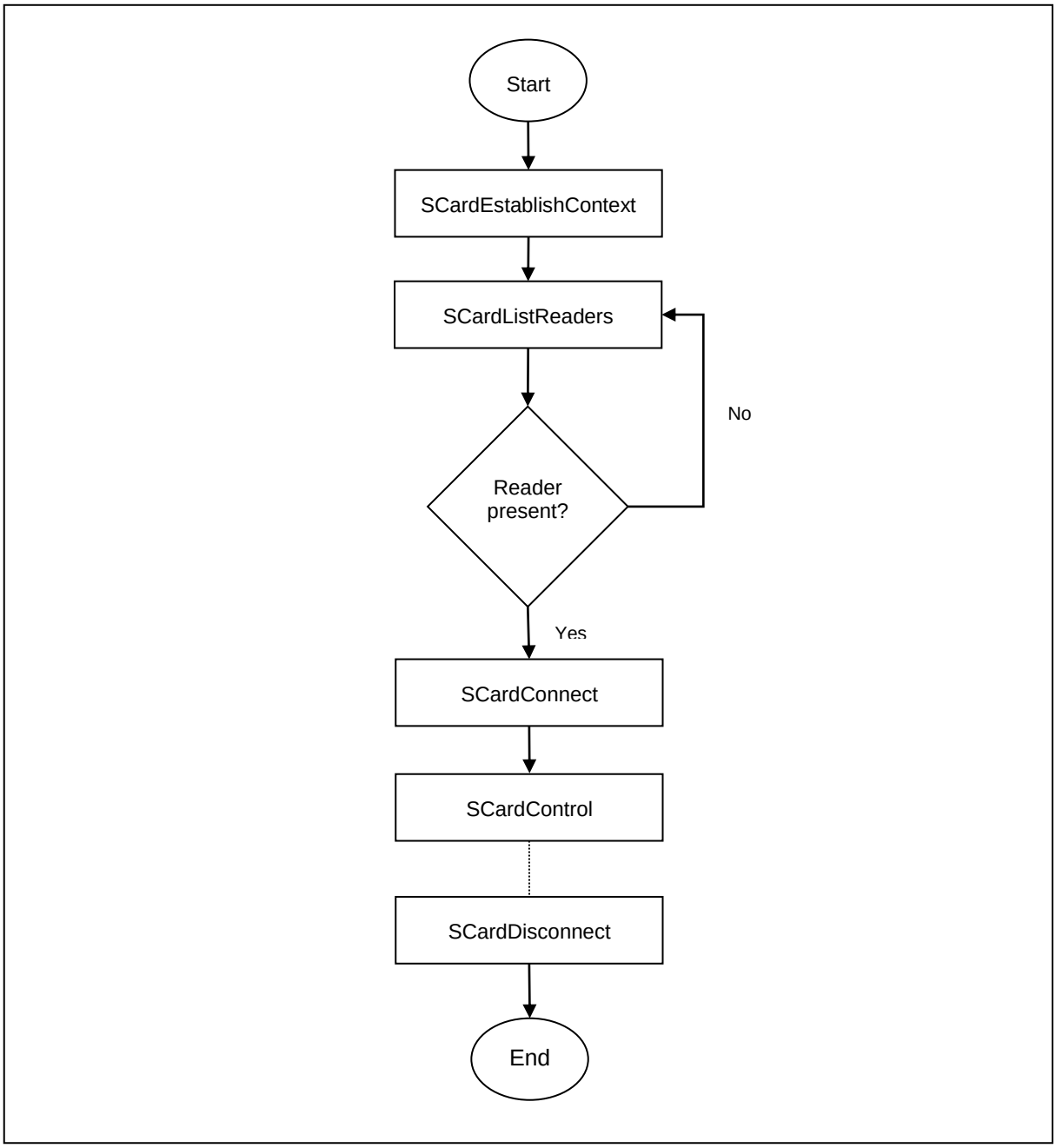


Figure 3: ACR1252U Escape Command Flow

5.2. Contactless Smart Card Protocol

5.2.1. ATR Generation

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

5.2.1.1. ATR Format for ISO 14443 Part 3 PICCs

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

Example:

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

Where:

Length (YY) = 0Ch

RID = {A0 00 00 03 06h} (PC/SC Workgroup)

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

Card Name (C0 .. C1) = {00 01h} (MIFARE Classic 1K)

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

= 11h: FeliCa

Card Name (C0 .. C1)

00 01: MIFARE Classic 1K	00 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® SL1 2K	FF [SAK]: undefined tags
00 37: MIFARE Plus® SL1 4K	

5.2.1.2. ATR Format for ISO 14443 Part 4 PICCs

<table>
<tr><th>Byte</th><th>Value</th><th>Designation</th><th>Description</th></tr>
<tr><td>0</td><td>3Bh</td><td>Initial Header</td><td></td></tr>
<tr><td>1</td><td>8Nh</td><td>T0</td><td>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)</td></tr>
<tr><td>2</td><td>80h</td><td>TD1</td><td>Higher nibble 8 means: no TA2, TB2, TC2 only TD2 is following.
Lower nibble 0 means T = 0</td></tr>
<tr><td>3</td><td>01h</td><td>TD2</td><td>Higher nibble 0 means no TA3, TB3, TC3, TD3 following.
Lower nibble 1 means T = 1</td></tr>
<tr><td rowspan="2">4
to
3 + N</td><td>XX</td><td>T1</td><td rowspan="2">Historical Bytes:

ISO 14443-A:
The historical bytes from ATS response. Refer to the ISO14443-4 specification.

ISO 14443-B:<table><tr><th>Byte1-4</th><th>Byte5-7</th><th>Byte8</th></tr><tr><td>Application Data from ATQB</td><td>Protocol Info Byte from ATQB</td><td>Higher nibble=MBLI from ATTRIB command
Lower nibble (RFU)=0</td></tr></table></td></tr>
<tr><td>XX
XX
XX</td><td>Tk</td></tr>
<tr><td>4+N</td><td>UU</td><td>TCK</td><td>Exclusive-oring of all the bytes T0 to Tk</td></tr>
</table>

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

5.2.2. Pseudo APDU for Contactless Interface

5.2.2.1. Get Data

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

Get UID APDU Format (5 Bytes)

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

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

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

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

Response	Data Out		
Result	ATS	SW1	SW2

Response Codes

Results	SW1	SW2	Meaning
Success	90h	00h	The operation was completed successfully.
Warning	62h	82h	End of UID/ATS reached before Le bytes (Le is greater than UID Length).
Error	6Ch	**XX**h	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};
```

5.2.2.2. Get PICC Data

This command returns the PICC data of the connected PICC.

Note: *This is only applicable to firmware version 110.0 and above.*

Get PICC Data APDU Format (5 Bytes)

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

If TypeA 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 TypeB card, Get ATQB (12 Bytes + 2 Bytes)

Response	Data Out		
Result	ATQB	SW1	SW2

Response Codes

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

5.2.3. APDU commands for PCSC 2.0 Part 3 (version 2.02 or above)

PCSC2.0 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.

5.2.3.1. 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 1st data object 02 = error in the 2nd 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.

5.2.3.2. 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	Class	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

5.2.3.2.1. 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	-

5.2.3.2.2. 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	-

5.2.3.2.3. 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

5.2.3.2.4. 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	-

5.2.3.2.5. 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	-

5.2.3.2.6. Timer Data Object

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

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

Timer Data Object

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

5.2.3.2.7. 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

5.2.3.2.8. 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

Parameters Requested	Tag	Length
PCB for ISO/IEC14443	07h	01h
CID for ISO/IEC14443	08h	01h
NAD for ISO/IEC14443	09h	01h
Param 1 – 4 for for ISO/IEC14443 type B	0Ah	04h

5.2.3.3. Transparent Exchange Command

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

Transparent Exchange Command

Command	Class	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

5.2.3.3.1. 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)

Tag	Length (1 byte)	Value	
		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

5.2.3.3.2. 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.

5.2.3.3.3. 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.

5.2.3.3.4. Transmit Data Object

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

Transmit Data Object

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

5.2.3.3.5. 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	-

5.2.3.3.6. 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 µs.

Transceive Data Object

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

5.2.3.3.7. 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	

Tag	Length (1 byte)	Value		
		3	0 – no framing error 1 – framing error detected	
		4 - 7	RFU	

5.2.3.3.8. 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)

5.2.3.4. Switch Protocol Command

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

Switch Protocol Command

Command	Class	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

5.2.3.4.1. Switch Protocol Data Object

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

Switch Protocol Data Object

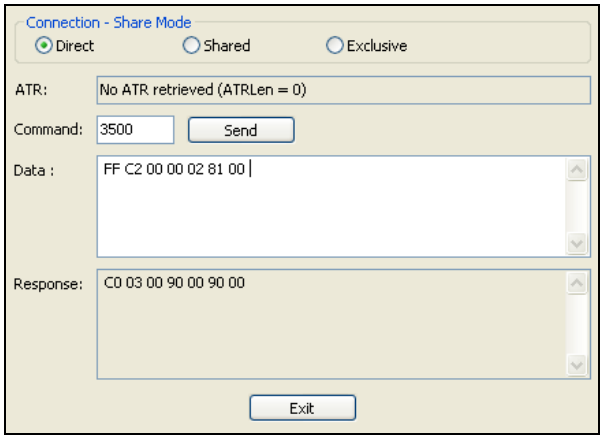
Tag	Length (1 byte)	Value	
		Byte 0	Byte 1
8Fh	02h	00h – ISO/IEC14443 Type A 01h – ISO/IEC14443 Type B 03h – FeliCa Other – RFU	00h – If no layer separation 02h – Switch to Layer 2 03h – Switch or activate to layer 3 04h – Activate to layer 4 Other - RFU

5.2.3.5. PCSC 2.0 Part 3 Example

1. Start Transparent Session.

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

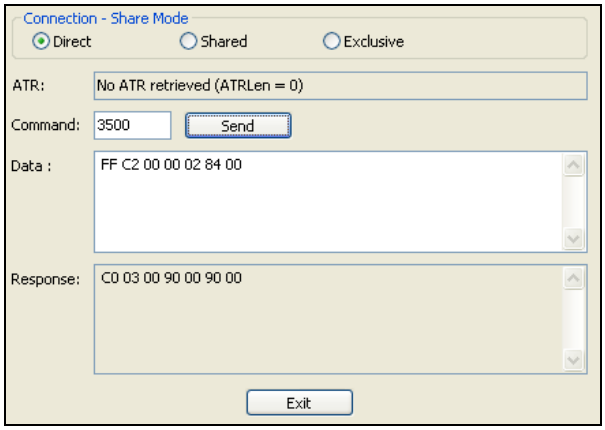
   Response: **C0 03 00 90 00 90 00**



2. Turn the Antenna Field on.

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

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

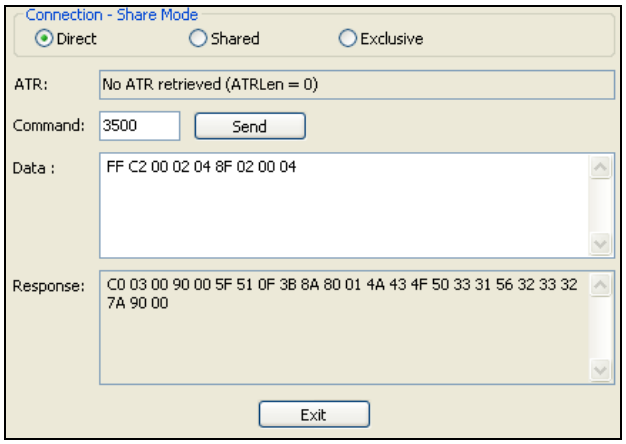


3. ISO 14443-4A Active.

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

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

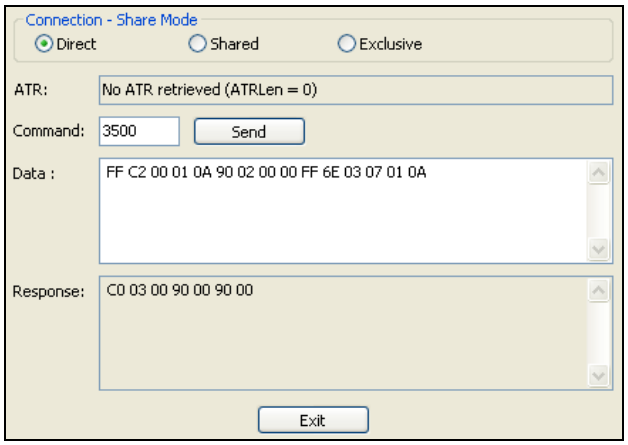
   C0 03 00 90 00 5F 51 [ATR] 90 00



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**

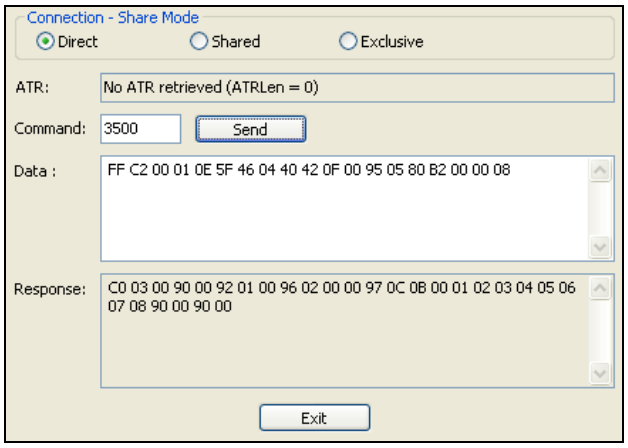
   Response: **C0 03 00 90 00 90 00**



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**

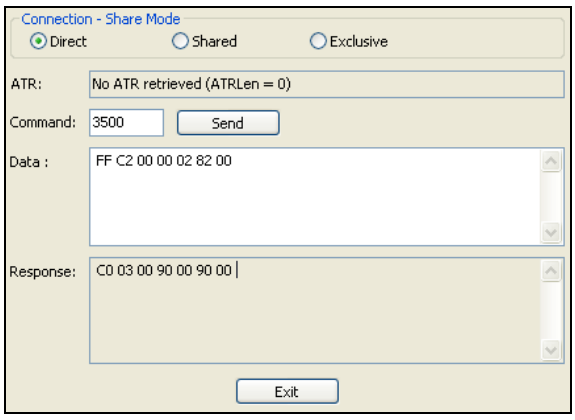
   Response: **C0 03 00 90 00 92 01 00 96 02 00 00 97 0C [Card Response] 90 00**



6. End Transparent Session.

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

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



5.2.4. PICC Commands for MIFARE Classic (1K/4K) Memory Cards

5.2.4.1. Load Authentication Keys

This command loads the authentication keys into 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	Class	INS	P1	P2	Lc	Data In
Load Authentication Keys	FFh	82h	Key Structure	Key Number	06h	Key (6 bytes)

Where:

Key Structure 1 byte.

00h = Key is loaded into the reader 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. *Default Value = {FF FF FF FF FF FFh}*

Key 6 bytes.

The key value loaded into the reader. e.g., {FF FF FF FF FF FFh}

Load Authentication Keys Response Format (2 bytes)

Response	Data Out	
Result	SW1	SW2

Load Authentication Keys Response Codes

Results	SW1 SW2	Meaning
Success	90 00h	The operation was completed successfully.
Error	63 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}

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

Load Authentication Keys APDU Format (10 bytes)

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

Authenticate Data Bytes (5 bytes)

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

Where:

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

For MIFARE Classic 1K card, it has a total of 16 sectors and each sector consists of four consecutive blocks (e.g., Sector 00h consists of blocks {00h, 01h, 02h and 03h}; sector 01h consists of blocks {04h, 05h, 06h and 07h}; the last sector 0Fh consists of blocks {3Ch, 3Dh, 3Eh and 3Fh}. Once the authentication is done successfully, there is no need to do the authentication again provided that the blocks to be accessed are belonging 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
Success	90h	00h	The operation was completed successfully.
Error	63h	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 2: 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 3: 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 4: 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.*

5.2.4.3. Read Binary Blocks

This command retrieves multiple data blocks from the PICC. The data block/trailer block must be authenticated first before executing this command.

Read Binary APDU Format (5 bytes)

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

Where:

Block Number — 1 byte.

The starting block.

Number of Bytes to Read — 1 byte.

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

Maximum of 16 bytes for MIFARE Ultralight.

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

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

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

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

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

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

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

Read Binary Block Response Codes

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

Examples:

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

5.2.4.4. Update Binary Blocks

This command writes multiple data blocks on the PICC. The data block/trailer block must be authenticated first before executing this command.

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

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

Where:

Block Number — 1 byte. The starting block to be updated.

Number of bytes to update — 1 byte.

- Multiple of 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 bytes, or 4 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 Blocks Mode).

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

Update Binary Block Response Codes (2 bytes)

Results	SW1	SW2	Meaning
Success	90	00h	The operation was completed successfully.
Error	63	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}

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

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

Value Block Operation APDU Format (10 bytes)

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

Where:

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

VB_OP 1 byte.

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

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

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

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

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

VB_Value			
MSB			LSB
FFh	FFh	FFh	FCh

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

VB_Value			
MSB			LSB
00h	00h	00h	01h

Value Block Operation Response Format (2 bytes)

Response	Data Out	
Result	SW1	SW2

Value Block Operation Response Codes

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

5.2.4.6. Read Value Block

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

Read Value Block APDU Format (5 bytes)

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

Where:

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

Read Value Block Response Format (4 + 2 bytes)

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

Where:

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

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

Value			
MSB			LSB
FFh	FFh	FFh	FCh

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

Value			
MSB			LSB
00h	00h	00h	01h

Read Value Block Response Codes

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

5.2.4.7. Copy Value Block

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

Copy Value Block APDU Format (7 bytes)

Command	Class	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
Success	90	00h	The operation was completed successfully.
Error	63	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}

5.2.5. Accessing PCSC-compliant tags (ISO 14443-4)

All ISO 14443-4 compliant cards (PICCs) understand the ISO 7816-4 APDUs. The ACR1252U reader just has to communicate with the ISO 14443-4 compliant cards by exchanging ISO 7816-4 APDUs and responses. The ACR1252U will handle the ISO 14443 Parts 1-4 Protocols internally.

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

ISO 7816-4 APDU Format

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

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

Response	Data Out		
Result	Response Data	SW1	SW2

Common ISO 7816-4 Response Codes

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

   In which,

   The 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]

5.2.6. Accessing FeliCa tags

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

FeliCa Command Format

Command	Class	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

5.3. Contact Smart Card Protocol

5.3.1. ACOS6-SAM Commands

This section contains SAM-specific commands.

***Note**: For complete information on ACOS6-SAM Commands and Scenarios, please contact an ACS representative for a copy of the ACOS6-SAM Reference Manual.*

5.3.1.1. Generate Key

This command is used to generate a diversified key to load into the ACOS3/6 card or other cards from deviation data such as a client card serial number. This command is catered for client card issuance purposes.

APDU	Description	
CLA	80h	
INS	88h	
P1	00h	Generate 8 Byte Key
	01h	Generate 16 Byte Key
	02h	Generate 24 Byte Key
P2	Key index of Master Key to generate Derived Key	
P3	08h	
Data	Input Data	

Specific Response Status Bytes

SW1 SW2	Description
69 86h	No DF selected
6A 86h	Invalid P1 or P2
67 00h	Incorrect P3, must be 08h
6A 83h	Referenced key record not found in EF2
69 81h	Invalid EF2 (record size, file type, etc.)
6A 88h	EF2 not found
62 83h	Current DF is blocked; EF2 is blocked
69 83h	Usage counter is zero.
69 82h	Security condition not satisfied
6A 87h	Referenced Master Key is not capable of 3-DES encryption
61 08h	Command completed, issue GET RESPONSE to get the result

5.3.1.2. Diversify (or load) Key Data

This command prepares the SAM card to perform ciphering operations by diversifying and loading the key. It takes the serial number and CBC initial vector as command data input.

APDU	Description								
CLA	80h								
INS	72h								
P1	b7	b6	b5	b4	b3	b2	b1	b0	Description
	-	0	0	0	0	0	0	1	Secret Code (Sc)
	-	0	0	0	0	0	1	0	Account Key (K_{ACCT})
	-	0	0	0	0	0	1	1	Terminal Key
	-	0	0	0	0	1	0	0	Card Key
	-	0	0	0	0	1	0	1	Bulk Encryption Key (Not diversified)
	-	0	0	0	0	1	1	0	Initial vector
	0	-	-	-	-	-	-	-	16-byte Key
	1	-	-	-	-	-	-	-	24-byte Key
P2	Index of Master Key: Bit7: 1 = local Key in current EF2; 0 = global KEY EF2 Bit6-Bit5: 00b - RFU Bit4-Bit0: Key Index								
P3	If P1 = 1-4, P3 = 8/16,(if algo is AES, P3 = 8/16) If P1 = 5, P3 = 0 If P1 = 6, P3 = 8 (Algo of Master Key is DES/ 3DES/ 3KDES) P3 = 16 (Algo of Master Key is AES)								
Data	If P1 = 1-4 Client card's Serial Number, (if algo is AES, Data is Client card's Serial Number or Client card's Serial Number append with "0000000000000000") If P1 = 5, No command data. If P1 = 6, DES/3DES/3KDES/AES CBC initial vector.								

Specific Response Status Bytes

SW1 SW2	Description
69 86h	No DF selected
6A 86h	Wrong P1, P1 must be 1 to 6
67 00h	Wrong P3, P3 must be 8 (or 0)
62 83h	Current DF is blocked, or EF2 is blocked
69 82h	Security condition not satisfied
6A 88h	EF2 not found
6A 83h	Referenced Master Key in EF2 not found

SW1 SW2	Description
69 81h	Invalid EF2 (FDB, MRL, etc., not consistent)
6A 87h	Referenced KEY not capable of authentication
69 83h	Referenced Key is locked
90 00h	Target key generated, and ready in SAM memory

5.3.1.3. Encrypt

This command is used to encrypt data using DES or 3DES with either:

1. The session key created by the mutual authentication procedure with an ACOS3/6, DESFire®, DESFire® EV1 or MIFARE Plus card.
2. A diversified key (secret code).
3. A bulk encryption key.
4. Encrypt the diversified secret code with the session key.
5. Prepare ACOS3 secure messaging command given a non-SM command.

APDU	Description								
CLA	80h								
INS	74h								
P1	b7	b6	b5	b4	b3	b2	b1	b0	Description
	-	0	0	0	0	0	0	-	ECB Mode
	-	0	0	0	0	0	1	-	CBC Mode
	-	0	0	0	0	1	0	-	Retail MAC Mode
	-	0	0	0	0	1	1	-	MAC Mode
	-	0	0	0	1	0	0	-	Prepare ACOS3 SM command.
	-	1	0	0	1	0	1	-	MIFARE DESFire Encryption
	-	1	0	0	1	1	0	-	MIFARE DESFire EV1 Encryption
	-	0	0	0	1	1	1	-	CMAC
	-	0	1	0	0	0	0		MIFARE Plus Command
	-	0	1	0	0	0	1		MIFARE Plus Response
	0	-	-	-	-	-	-	0	3DES
	0	-	-	-	-	-	-	1	DES
	1	-	-	-	-	-	-	0	3K DES
	1	-	-	-	-	-	-	1	AES
	-	-	-	-	-	-	-	-	All other values – RFU

APDU	Description
P2	P2 is derived key in SAM set using Load Key function: 1 – Encrypt Data with Session Key *Ks* 2 – Encrypt Data with Diversified Key *Sc* 3 – Encrypt Data with Bulk Encryption Key 0 – return ENC (*Sc*, *Ks*) If P1.b3 = 1 or b5=1, P2 must be 1 If P2 = 0h, P1 can be either 0 or 1
P3	P3 < 128 If bit 3 of P1 not equal to 1 and bit 5 of P1 not equal to 1 - If P2 = 1-3, multiple of 8 (DES/3DES/3KDES) or 16 (AES) up to 128 bytes - If P2 = 0, 0
Data	Plain text If P2 b6 = 1, The DATA format should be: • Length of Plain text data • Length of Command and Header of DESFire Card • Command and Header of DESFire Card • Plain text P1 = `A1h`, the encryption is for a MIFARE Plus command • if MFP Command is *value* operations command, the DATA format should be Command code(1 BYTE)+BlockNum(2/4 BYTE)+Value(4 BYTE). • if MFP Command is *Proximity Check*, the DATA format should be Command code(1 BYTE)+ PPS1(1 BYTE). • if MFP Command is *Read*, the DATA format should be Command code(1 BYTE)+ BlockNum(2 BYTE) • if MFP Command is *Write*, the DATA format should be Command code(1 BYTE)+ BlockNum(2 BYTE) +plaintext P1=`A3h`, • The data return by ICC (don't include SC code and don't include RMAC if RMAC exist)

Specific Response Status Bytes

SW1 SW2	Description
69 86h	No DF selected
6A 86h	Invalid P1 or P2
67 00h	Incorrect P3
6A 83h	ACOS Target Key is not ready (use Diversify to generate the key)
61 XX	Encryption is done, use GET RESPONSE to get the result

5.3.1.4. Decrypt

This command is used to decrypt data using DES or 3DES or AES with either:

1. The session key created by the mutual authentication procedure with an ACOS3/6, MIFARE DESFire, MIFARE DESFire EV1 or MIFARE Plus card.
2. A diversified key (secret code).
3. A bulk encryption key.
4. Decrypt the diversified secret code with the session key.
5. Verify and Decrypt ACOS3 secure-messaging response.

Verify and Decrypt ACOS3 SM Response:

APDU	Description								
CLA	80h								
INS	76h								
P1	b7	b6	b5	b4	b3	b2	b1	b0	Description
	-	0	0	0	0	0	0	-	ECB Mode
	-	0	0	0	0	0	1	-	CBC Mode
	-	0	0	0	1	0	0	-	Verify and Decrypt ACOS3 SM Response
	-	1	0	0	1	0	1	-	MIFARE DESFire Decryption
	-	1	0	0	1	1	0	-	MIFARE DESFire EV1 Decryption
	-	0	1	0	0	1	0	-	MIFARE Plus Decryption
	0	-	-	-	-	-	-	0	3DES
	0	-	-	-	-	-	-	1	DES
	1	-	-	-	-	-	-	0	3K DES
	1	-	-	-	-	-	-	1	AES
	0	0	0	0	-	-	-	-	All other values - RFU
P2	P2 is derived key in SAM set using Load Key function: 1 – Decrypt Data with Session Key *Ks* 2 – Decrypt Data with Diversified Key *Sc* 3 – Decrypt Data with Bulk Encryption Key 0 – return DEC (*Sc*, *Ks*)								
P3	P3 < 128 If P1 = `A5h`, P3=16/32/48 If bit 3 of P1 not equal to 1 - If P2 = 1-3, multiple of 8 (DES/3DES/3KDES) or 16 (AES) up to 128 bytes - If P2 = 0, 0								
Data	Ciphertext If P1 = `A5h`, The DATA is Encrypted text If P2 b6 = 1, The DATA format should be: • Length of Plain text data, if unknown, use 00 • Length of Command and Header of DESFire Card • Command and Header of DESFire Card • Encrypted text								

Specific Response Status Bytes

SW1 SW2	Description
69 86h	No DF selected
6A 86h	Invalid P1 or P2
67 00h	Incorrect P3
6A 83h	ACOS Target Key is not ready (use Diversify to generate the key)
61 XX	Decryption is done, use GET RESPONSE to get the result

5.3.1.5. Prepare Authentication

This command is used to authenticate the SAM card (as the terminal) to the ACOS 3/6 card or MIFARE Ultralight C/MIFARE DESFire Card/MIFARE Plus card.

APDU	Description
CLA	80h
INS	78h
P1	00h – 3DES 01h – DES 02h – 3KDES (MIFARE DESFire EV1/ACOS3) 03h – AES (MIFARE DESFire EV1/MIFARE Plus/ACOS3) 80h – 3DES (MIFARE DESFire Authenticate only) 81h – DES (MIFARE DESFire Authenticate only) Other – RFU
P2	0h – Verify ACOS3/6 Authenticate Return 01h – MIFARE Ultralight C/DESFire Authenticate by (Diversified) Terminal Key 05h – MIFARE Ultralight C/DESFire Authenticate by Bulk Encryption Key 02h – MIFARE Plus Authenticate. First Authenticate of SL1 to SL3 03h – MIFARE Plus Authenticate. Authentication in SL1 to SL2. 04h – MIFARE Plus Authenticate. Following Authenticate of SL2 to SL3.
P3	8 – (P1 = 00h, 01h, 02h, 80h, 81h) 16 – (P1 = 03h)
Data	Card Challenge Data

Specific Response Status Bytes

SW1 SW2	Description
69 86h	No DF selected
6A 86h	Invalid P1 or P2
67 00h	Incorrect P3, must be 08h
6A 83h	ACOS Key (KT or KC) is not ready (use Diversify to generate this key)
69 82h	Security condition not satisfied
61 10h	Command completed, issue GET RESPONSE to get the result

5.3.1.6. Verify Authentication

This command is used to verify the ACOS 3/6, MIFARE Ultralight C, MIFARE DESFire/MIFARE DESFire EV1 or MIFARE Plus card to the terminal. The Session Key Ks would also be generated internally.

APDU	Description
CLA	80h
INS	7Ah
P1	00h – 3DES (P2 = 0) 01h – DES (P2 = 0) 02h – 3KDES (P2 = 0, ACOS3) 03h – AES (P2 = 0, ACOS3) Other – RFU
P2	00h – Verify ACOS3/6 Authenticate Return 01h – Verify MIFARE Ultralight C®/ DESFire®/ DESFire® EV1 Authenticate Return 02h – Verify MIFARE Plus Authenticate return
P3	08h – (P2 = 0, P2 = 1 and Session Key is DES/3DES) 16h – (P2 = 1 and Session Key is 3KDES/AES) 16h – (P2=02, and MIFARE Plus return data ek(RndA')) 32h – (P2=02, and MIFARE Plus return data ek(TI+PICCcap2+PCDcap2))
Data	ACOS 3/6: DES (K_S, RND_T) MIFARE DESFire/ DESFire EV1 return data: ek(RndA') MIFARE Plus return data ek(RndA') or ek(TI+PICCcap2+PCDcap2)

Specific Response Status Bytes

SW1 SW2	Description
69 86h	No DF selected
6A 86h	Invalid P1 or P2
67 00h	Incorrect P3, must be 08h
6A 83h	ACOS-SAM Session Key or RND_T are not ready. Use PREPARE AUTHENTICATION to build these keys.
69 82h	Data is incorrect
90 00h	Data is correct, ACOS Mutual Authentication is successful

5.3.1.7. Verify ACOS Inquire Account

This command is used to verify the ACOS3/6 card's Inquire Account purse command. It would verify that the MAC checksum returned by ACOS3/6 are correct with the SAM's diversified key.

APDU	Description								
CLA	80h								
INS	7Ch								
P1	b7	b6	b5	b4	b3	b2	b1	b0	Description
	-	0	0	0	0	-	0	-	ACOS INQ_AUT is disabled
	-	0	0	0	0	-	1	-	ACOS INQ_AUT is enabled
	-	0	0	0	0	0	-	-	ACOS INQ_ACC_MAC is disabled
	-	0	0	0	0	1	-	-	ACOS INQ_ACC_MAC is enabled
	0	-	-	-	-	-	-	0	3DES
	0	-	-	-	-	-	-	1	DES
	1	-	-	-	-	-	-	0	3K DES (ACOS3 only)
	1	-	-	-	-	-	-	1	AES (ACOS3 only)
P2	0h								
P3	1Dh								
Data	Data Block returned by INQUIRE ACCOUNT of client ACOS card, see below.								

Specific Response Status Bytes

SW1 SW2	Description
69 86h	No DF selected
6A 86h	Invalid P1 or P2
67 00h	Incorrect P3
6A 83h	ACOS Key K_S or K_{ACCT} are not ready; use DIVERSIFY command to generate K_{ACCT}; if applicable, use "Prepare Authentication" to generate K_S.
6F 00h	Data Block's MAC is incorrect
90 00h	Data Block's MAC is correct

5.3.1.8. Prepare ACOS Account Transaction

To create an ACOS3/6 Credit/Debit command, the MAC must be computed for ACOS3/6 to verify.

APDU	Description								
CLA	80h								
INS	7Eh								
P1	b7	b6	b5	b4	b3	b2	b1	b0	Description
	-	0	0	0	0	0	0	-	ACOS TRNS_AUT is disabled
	-	0	0	0	0	0	1	-	ACOS TRNS_AUT is enabled
	0	-	-	-	-	-	-	0	3DES
	0	-	-	-	-	-	-	1	DES
	1	-	-	-	-	-	-	0	3K DES (ACOS3 only)
	1	-	-	-	-	-	-	1	AES (ACOS3 only)
P2	E2h: Credit E6h: Debit								
P3	0Dh								
Data	Data Block								

Specific Response Status Bytes

SW1 SW2	Description
69 86h	No DF selected
6A 86h	Invalid P1 or P2
67 00h	Incorrect P3, must be 0Dh
6A 83h	ACOS Key K_S or K_{ACCT} are not ready; use DIVERSIFY command to generate K_{ACCT}; if applicable, use “Prepare Authentication” to generate K_S.
61 0Bh	Command completed, issue GET RESPONSE to get the result

5.3.1.9. Verify Debit Certificate

For ACOS3/6, if the DEBIT command has P1 = 1, a debit certificate is returned. The debit certificate can be checked by comparing the ACOS3 response to the result of this command.

APDU	Description
CLA	80h
INS	70h

APDU	Description								
	b7	b6	b5	b4	b3	b2	b1	b0	Description
	-	0	0	0	0	0	0	-	ACOS TRNS_AUT is disabled
	-	0	0	0	0	0	1	-	ACOS TRNS_AUT is enabled
P1	0	-	-	-	-	-	-	0	3DES
	0	-	-	-	-	-	-	1	DES
	1	-	-	-	-	-	-	0	3K DES (ACOS3 only)
	1	-	-	-	-	-	-	1	AES (ACOS3 only)
P2	0h								
P3	14h								
Data	Data Block								

Specific Response Status Bytes

SW1 SW2	Description
69 86h	No DF selected
6A 86h	Invalid P1 or P2
67 00h	Incorrect P3, must be 14h
6A 83h	ACOS Key K_S or K_{ACCT} are not ready; use DIVERSIFY command to generate K_{ACCT}; if applicable, use PREPARE AUTHENTICATION to generate K_S.
69 82h	Security condition not satisfied
6F 00h	DEBIT CERTIFICATE is invalid
90 00h	Success, DEBIT CERTIFICATE is valid

5.3.1.10. Get Key

This command allows secure key injection from the current SAM's Key File (SFI=02h) into another ACOS6/ACOS6-SAM with or without key diversification. Using this ensures that the keys to be injected are protected by encryption and message authentication codes.

The Get Key command also allows secure key injection from the current SAM's Key File (SFI=02h) into ACOS7/10, MIFARE DESFire, MIFARE DESFire EV1 or MIFARE Plus card with key diversification. Using this ensures that the key to be injected is protected by encryption and message authentication codes.

If bit 7 of the Special Function Flag (Key Injection Only Flag) of the **Card Header Block** (Section 3.2 of ACOS6-SAM Reference Manual) has been set and the key file has been activated, Get Key must be used for loading or changing keys in the card. Setting this bit will disable Read Record command for the key file under any circumstances after activation.

Before this command is to be executed, a session key is already established with the target card with the mutual authentication procedure of **Mutual Authentication** (Section 5.3 of ACOS6-SAM Reference Manual) or the MIFARE Plus/MIFARE DESFire mutual authentication procedure.

Note: *The GET KEY command can only get the Key data.*

<table>
<tr><th>APDU</th><th colspan="4">Description</th></tr>
<tr><td>CLA</td><td colspan="4">80h</td></tr>
<tr><td>INS</td><td colspan="4">CAh</td></tr>
<tr><td rowspan="11">P1</td><td colspan="4">Get Key for ACOS card Set Key</td></tr>
<tr><td colspan="4">00h Response data is Key in MSAM
01h Response data is 16-byte Diversify Key
02h Response data is 24-byte Diversify Key
03h Response data is the Change Key command of MIFARE Plus Card</td></tr>
<tr><td colspan="4">Get Key for DESFire card Change Key, Response data for DESFire/DESFire EV1 Change Key</td></tr>
<tr><td></td><td>Card Type</td><td>Authenticate Key No. And Changing Key No.*</td><td>Key Length</td></tr>
<tr><td>80h</td><td>MIFARE DESFire</td><td>Are DIFFERENT in MIFARE DESFire card</td><td>16 bytes</td></tr>
<tr><td>81h</td><td>MIFARE DESFire EV1</td><td>Are DIFFERENT in MIFARE DESFire EV1 card</td><td>16 bytes</td></tr>
<tr><td>82h</td><td>MIFARE DESFire EV1</td><td>Are DIFFERENT in MIFARE DESFire EV1 card</td><td>24 bytes</td></tr>
<tr><td>88h</td><td>MIFARE DESFire</td><td>Are the SAME in MIFARE DESFire card</td><td>16 bytes</td></tr>
<tr><td>89h</td><td>MIFARE DESFire EV1</td><td>Are the SAME in MIFARE DESFire EV1 card</td><td>16 bytes</td></tr>
<tr><td>8Ah</td><td>MIFARE DESFire EV1</td><td>Are the SAME in MIFARE DESFire EV1 card</td><td>24 bytes</td></tr>
<tr><td>P2</td><td colspan="4">Key ID in SAM (New key for change)</td></tr>
<tr><td>P3</td><td colspan="4">If P1 = 00h, P3 is 08h
If P1 = 01/02h, P3 is 10h
If P1 = 03h, P3 is 0Bh
If P1 = 80/81/82/88/89/8Ah: P3 is 0Bh</td></tr>
<tr><td>Data</td><td colspan="4">If P1 = 00h, command data is RND_{Target}
If P1 = 01/02h, command data is RND_{Target} + serial (or batch) number of target card
If P1 = 03h
- Serial Number for target card (8 Byte)
- Write Command (A0 or A1) (1 Byte)
- BNr (2 Byte)
If P1 = 80/81/82/88/89/8Ah:
- Serial Number for target card (8 Byte)
- Original Key ID (Key in SAM card stored the Original key, 00 = Default Key of DESFire - Card)
- Key No. (DESFire Card Key No.)
- Key Version (DESFire Card Key Version, If not used, value = 00)</td></tr>
</table>

* This column points out if the listed cards have a distinct Change Key and Authenticate Key, or if they use the same value for both keys.

Specific Response Status Bytes

SW1 SW2	Description
69 85h	SAM Session Key not ready
62 83h	Current DF is blocked, or Target EF is blocked
69 86h	No DF selected
69 81h	Wrong file type of Key file, it should be Internal Linear Variable File
69 82h	Target file's header block has wrong checksum, or security condition not satisfied
6A 86h	Invalid P1 or P2
67 00h	Incorrect P3
6A 83h	Target Key is not ready or Key Length less than 16
61 1Ch	Success, use GET RESPONSE to get the result

5.4. Peripherals Control

The reader's peripherals control commands are implemented by using ***SCardControl*** with Control Code ***SCARD_CTL_CODE(3500)***.

5.4.1. Get Firmware Version

This command is used to get the reader's firmware message.

Get Firmware Version Format (5 bytes)

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

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

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

Example:

Response = E1 00 00 00 0F 41 43 52 31 32 35 32 55 5F 56 31 30 30 2E 31

Firmware Version (HEX) = 41 43 52 31 32 35 32 55 5F 56 31 30 30 2E 31

Firmware Version (ASCII) = "ACR1252U_V100.1"

5.4.2. LED Control

This command controls the LED's output.

LED Control Format (6 bytes)

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

LED Control Response Format (6 bytes)

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

LED Status (1 byte)

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

5.4.3. LED Status

This command checks the existing LED's status.

LED Status Format (5 bytes)

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

LED Status Response Format (6 bytes)

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

LED Status (1 byte)

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

5.4.4. Buzzer Control

This command controls the buzzer output.

Buzzer Control Format (6 bytes)

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

Where:

Buzzer On Duration 1 byte.

00h = Turn OFF

01 to FFh = Duration (unit: 10 ms)

Buzzer Control Response Format (6 bytes)

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

5.4.5. Buzzer Status

This command checks the existing buzzer status.

Buzzer Status Format (5 bytes)

Command	Class	INS	P1	P2	Lc
Buzzer Status	E0h	00h	00h	28h	00h

Buzzer Status Response Format (6 bytes)

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

5.4.6. Set LED and Buzzer Status Indicator Behavior for PICC Interface

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

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

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

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

Behavior (1 byte)

Behavior	Mode	Description
Bit 0	Card Operation Blinking LED	To blink the LED whenever the PICC card is being accessed
Bit 1	PICC Polling Status LED	To show the PICC Polling Status 1 = Enable; 0 =Disable
Bit 2	PICC Activation Status LED	To show the activation status of the PICC interface 1 = Enable; 0 =Disable
Bit 3	Card Insertion Events Buzzer	To make a beep whenever a card insertion event is detected 1 = Enable; 0 = Disable
Bit 4	Card Removal Events Buzzer	To make a beep whenever a card removal event is detected 1 = Enable; 0 = Disable *(Bit 3 should be enabled first, in order for this function to take effect)*
Bit 5	Reader Power ON Buzzer	To make a beep when the reader is powered ON 1 = Enable; 0 = Disable
Bit 6	Color Select (GREEN)	GREEN LED for status change 1 = Enable; 0 = Disable
Bit 7	Color Select (RED)	RED LED for status change 1 = Enable; 0 = Disable

Note: *Default value of Behavior = 7Fh*

Set LED and Buzzer Status Indicator Behaviors for PICC Interface Response Format (6 bytes)

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

5.4.7. Read LED and Buzzer Status Indicator Behavior for PICC Interface

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

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

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

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

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

Behavior (1 byte)

Behavior	Mode	Description
Bit 0	Card Operation Blinking LED	To blink the LED whenever the PICC card is being accessed
Bit 1	PICC Polling Status LED	To show the PICC Polling Status 1 = Enable; 0 =Disable
Bit 2	PICC Activation Status LED	To show the activation status of the PICC interface 1 = Enable; 0 =Disable
Bit 3	Card Insertion Events Buzzer	To make a beep whenever a card insertion event is detected 1 = Enable; 0 = Disable
Bit 4	Card Removal Events Buzzer	To make a beep whenever a card removal event is detected 1 = Enable; 0 = Disable *(Bit 3 should be enabled first, in order for this function to take effect)*
Bit 5	Reader Power ON Buzzer	To make a beep when the reader is powered ON 1 = Enable; 0 = Disable
Bit 6	Color Select (GREEN)	GREEN LED for status change 1 = Enable; 0 = Disable
Bit 7	Color Select (RED)	RED LED for status change 1 = Enable; 0 = Disable

Note: *Default value of Behavior = 7Fh*

5.4.8. 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.

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

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

Set Automatic PICC Polling Format (6 bytes)

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

Set Automatic PICC Polling Response Format (6 bytes)

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

Polling Setting (1 byte)

Polling Setting	Mode	Description
Bit 0	Auto PICC Polling	1 = Enable; 0 =Disable
Bit 1	Turn off Antenna Field if no PICC is found.	1 = Enable; 0 =Disable
Bit 2	Turn off Antenna Field if the PICC is inactive.	1 = Enable; 0 =Disable
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 14443-A Part 4	1= Enable; 0= Disable.

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 of energy saving. However, the response time of PICC Polling will become longer. The Idle Current Consumption in Power Saving Mode is about 60 mA, while the Idle Current Consumption in Non-Power Saving mode is about 130mA.*

   Note: *Idle Current Consumption = PICC is not activated.*
3. *The reader will activate the ISO 14443A-4 mode of the "ISO 14443A-4 compliant PICC" automatically. Type B PICC will not be affected by this option.*
4. *The JCOP30 card comes with two modes: ISO 14443A-3 (MIFARE Classic 1K) and ISO 14443A-4 modes. The application has to decide which mode should be selected once the PICC is activated.*

5.4.9. Read Automatic PICC Polling

This command checks the current PICC polling setting.

Read Automatic PICC Polling Format (5 bytes)

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

Read Automatic PICC Polling Response Format (6 bytes)

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

Polling Setting (1 byte)

Polling Setting	Mode	Description
Bit 0	Auto PICC Polling	1 = Enable; 0 =Disable
Bit 1	Turn off Antenna Field if no PICC is found.	1 = Enable; 0 =Disable
Bit 2	Turn off Antenna Field if the PICC is inactive.	1 = Enable; 0 =Disable
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 14443-A Part 4	1= Enable; 0= Disable.

Note: *Default value of Polling Setting = 8Bh.*

5.4.10. Set PICC Operating Parameter

This command sets the PICC operating parameter.

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

Set the PICC Operating Parameter Format (6 bytes)

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

Set the PICC Operating Parameter Response Format (6 bytes)

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

Operating Parameter (1 byte)

Operating Parameter	Parameter	Description	Option
Bit 0	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	RFU		RFU
Bit 6	SRIX		1 = Detect 0 = Skip
Bit 7	RFU	RFU	RFU

Note: *Default value of Operation Parameter = 5Fh.*

5.4.11. Read PICC Operating Parameter

This command checks the current PICC operating parameter.

Read the PICC Operating Parameter Format (5 bytes)

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

Read the PICC Operating Parameter Response Format (6 bytes)

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

Operating Parameter (1 byte)

Operating Parameter	Parameter	Description	Option
Bit 0	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	RFU		RFU
Bit 6	SRIX		1 = Detect 0 = Skip
Bit 7	RFU	RFU	RFU

Note: *Default value of Operation Parameter = 5Fh.*

5.4.12. Set PICC Operating Parameter (Extended)

This command sets the PICC operating parameter.

Note: *The setting will be saved into non-volatile memory. This is only applicable to firmware version 110.0 and above.*

Set the PICC Operating Parameter Format (7 bytes)

Command	Class	INS	P1	P2	Lc	Data In	
Set the PICC Operating Parameter	E0h	00h	01h	20h	02h	Operating Parameter1	Operating Parameter2

Set the PICC Operating Parameter Response Format (7 bytes)

Response	Class	INS	P1	P2	Le	Data Out	
Result	E1	00h	01h	00h	02h	Operation Parameter1	Operation Parameter2

Operating Parameter1 (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	RFU		RFU
Bit 6	SRIX		1 = Detect 0 = Skip
Bit 7	RFU	RFU	RFU

Note: *Default value of Operation Parameter1 = 5Fh.*

Operating Parameter2 (1 byte)

Operating Parameter	Parameter	Description	Option
Bit 0	Picopass	The Tag Types to be detected during PICC Polling.	1 = Detect 0 = Skip
Bit 1 - 7	RFU	RFU	RFU

Note: *Default value of Operation Parameter2 = 01h*

5.4.13. Read PICC Operating Parameter (Extended)

This command checks the current PICC operating parameter.

Note: *This is only applicable to firmware version 110.0 and above.*

Read the PICC Operating Parameter Format (5 bytes)

Command	Class	INS	P1	P2	Lc
Read the PICC Operating Parameter	E0h	00h	01h	20h	00h

Read the PICC Operating Parameter Response Format (7 bytes)

Response	Class	INS	P1	P2	Le	Data Out	
Result	E1	00h	01h	00h	02h	Operating Parameter1	Operating Parameter2

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	RFU		RFU
Bit 6	SRIX		1 = Detect 0 = Skip
Bit 7	RFU	RFU	RFU

Note: *Default value of Operation Parameter = 5Fh.*

Operating Parameter2 (1 byte)

Operating Parameter	Parameter	Description	Option
Bit 0	Picopass	The Tag Types to be detected during PICC Polling.	1 = Detect 0 = Skip
Bit 1 - 7	RFU	RFU	RFU

Note: *Default value of Operation Parameter2 = 01h.*

5.4.14. Set Auto PPS

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

Set Auto PPS Format (7 bytes)

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

Set Auto PPS Response Format (9 bytes)

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

Where:

Max Tx Speed Maximum Tx Speed (1 Byte)

Current Tx Speed Current Tx Speed (1 Byte)

Max Rx Speed Maximum Rx Speed (1 Byte)

Current Rx Speed Current Rx Speed (1 Byte)

00h = 106 Kbps; default setting, equal to No Auto PPS

01h = 212 Kbps

02h = 424 Kbps

Notes:

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

5.4.15. Read Auto PPS

This command checks the current auto PPS setting.

Read Auto PPS Format (5 bytes)

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

Read Auto PPS Response Format (9 bytes)

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

Where:

Max Tx Speed Maximum Tx Speed (1 Byte)

Current Tx Speed Current Tx Speed (1 Byte)

Max Rx Speed Maximum Rx Speed (1 Byte)

Current Rx Speed Current Rx Speed (1 Byte)

00h = 106 Kbps; default setting, equal to No Auto PPS

01h = 212 Kbps

02h = 424 Kbps

5.4.16. Set Serial Number

This command sets the serial number.

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

Set the Serial Number Format

Command	Class	INS	P1	P2	Lc	Data In
Set the Serial Number	E0h	00h	00h	DAh	Len	Serial Number (N bytes)

Where:

Serial Number N bytes. Maximum 20 bytes.

Set the Serial Number Response Format

Response	Class	INS	P1	P2	Le	Data Out
Result	E1	00h	00h	00h	Len	Serial Number / Return Code (N bytes) / (2 bytes)

Return Code

Results	SW1 SW2	Meaning
Lock	90 00h	The serial number is already locked.
Error	63 00h	The serial number is too long.

5.4.17. Read Serial Number

This command reads the serial number.

Read the Serial Number Format (5 bytes)

Command	Class	INS	P1	P2	Lc
Read the Serial Number	E0h	00h	00h	33h	00h

Read the Serial Number Response Format

Response	Class	INS	P1	P2	Le	Data Out
Result	E1	00h	00h	00h	Len	Serial Number (N bytes)

5.4.18. Set and Lock Serial Number

This command sets and locks the serial number. The setting will be saved **permanently** into non-volatile memory.

Note: *Please contact us at info@acs.com.hk or contact an Advanced Card Systems Ltd. Sales Representative for the details regarding this command.*

5.4.19. Unlock Serial Number

This command unlocks the serial number if the serial number has been set and locked. This will unlock and re-write a new serial number if the existing serial number has been locked and needs to be reconfigured.

Note: *Please contact us at info@acs.com.hk or contact an Advanced Card Systems Ltd. Sales Representative for the details regarding this command.*

5.4.20. Read PICC Type

This command checks the current PICC Type.

Note: *This is applicable only to firmware version 110.0 and above.*

Read PICC Type Format (5 bytes)

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

Read PICC Type Response Format (7 bytes)

Response	Class	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 TypeA
23h = ISO 14443-4 TypeB
28h = Srix
30h = Picopass

Card Status 1 byte.

00h = PICC Power Off [Contactless Tag Detected]
Others = PICC Detected [Contactless Tag Detected]

5.5. NFC Peer-to-Peer Mode-related Commands

5.5.1. Initiator Mode-related Commands

This section provides the commands that can be used in Initiator Mode. The figure below shows the peer-to-peer flow of commands for Initiator Mode.

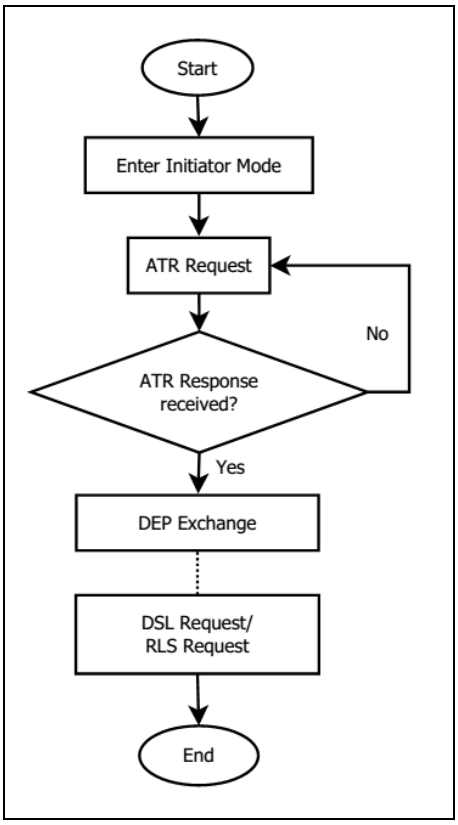


Figure 4: Peer-to-Peer Flow for Initiator Mode

5.5.1.1. Set Initiator Mode Timeout

This command sets the timeout for Initiator Mode.

Set Initiator Mode Timeout Command Format (7 bytes)

Command	Class	INS	P1	P2	Lc	Data In	
Set Initiator Mode	E0h	00h	00h	41h	02h	Timeout (MSB)	Timeout (LSB)

Note: *Unit = 10 ms, default value of Initiator Mode Timeout = 00 64h (100 * 10 ms = 1000 ms).*

Set Initiator Mode Timeout Response Format (7 bytes)

Response	Class	INS	P1	P2	Le	Data Out	
Result	E1h	00h	00h	00h	02h	Timeout (MSB)	Timeout (LSB)

Where:

Timeout 2 bytes. Timeout for Initiator Mode (unit = 10 ms).

5.5.1.2. Enter Initiator Mode

This command sets the reader into Initiator Mode, in order to send out SNEP Message.

Enter Initiator Mode Command Format (8 bytes)

Command	Class	INS	P1	P2	Lc	Data In		
Enter Initiator Mode	E0h	00h	00h	40h	03h	NFCMode	OpMode	Speed

Enter Initiator Mode Response Format (8 bytes)

Response	Class	INS	P1	P2	Le	Data Out		
Result	E1h	00h	00h	00h	03h	NFCMode	OpMode	Speed

Where:

NFCMode 1 byte. NFC Device Mode.
- 01h = MIFARE Ultralight Card Emulation Mode
- 03h = FeliCa Card Emulation Mode
- 08h = Peer-to-Peer Initiator Mode
- 00h = Card Read/Write Mode

OpMode 1 byte. Active Mode/Passive Mode.
- 01h = Active Mode
- 02h = Passive Mode

Speed 1 byte. Communication Speed.
- 01h = 106 Kbps
- 02h = 212 Kbps
- 03h = 424 Kbps

After executing Enter Initiator Mode command, the reader will wait for the NFC device, which in Target Mode, will present and send out the pre-set SNEP Message to it. The reader will stop all other tasks until the SNEP Message is sent successfully.

5.5.1.3. Send ATR Request

This command sends an ATR_REQ to peer-to-peer Target Mode device within the field.

ATR Request Command Format

Command	Class	INS	P1	P2	Lc	Data In				
ATR Request	E0h	00h	00h	42h	Len	11h	Mode (1 byte)	Speed (1 byte)	NFCID (10 bytes)	DID (1 byte)

Data In				
BS (1 byte)	BR (1 byte)	PP (1 byte)	LLCP Parameter	
			GiLen (1 byte)	Gi (GiLen bytes)

ATR Request Response Format

Response	Class	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	Len	ATR Response (Len bytes)

Where:

Mode 1 byte. Operation Mode.
01h = Active
02h = Passive

Speed 1 byte. Communication Speed.
01h = 106 Kbps
02h = 212 Kbps
03h = 424 Kbps

NFCID 10 bytes. Initiator device's NFCID.

DID 1 byte. Initiator device's Device Identification.

BS 1 byte. Initiator device's support send-bit rates.

BR 1 byte. Initiator device's support bit rates.

PP 1 byte. Initiator device's optional parameters.

Gi N bytes. LLCP parameter.

5.5.1.4. Exchange DEP

This command exchanges DEP with target device.

DEP Exchange Command Format

Command	Class	INS	P1	P2	Lc	Data In			
DEP Exchange	E0h	00h	00h	43h	Len	11h	PFB (1 byte)	DepLen (1 byte)	Dep (N bytes)

DEP Exchange Response Format

Response	Class	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	Len	Dep Response (Len bytes)

Where:

PFB 1 byte. Control the data transmission and error recovery.

DepLen 1 byte. DEP message length.

Dep N bytes. DEP message for peer-to-peer communication.

5.5.1.5. Send DSL Request

This command sends a DSL request to target device.

DSL Request Command Format

Command	Class	INS	P1	P2	Lc	Data In	
DSL request	E0h	00h	00h	44h	02h	11h	DID (1 byte)

Where:

DID 1 byte. Device Identification.

DSL Request Response Format

Response	Class	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	02h	Return Code (2 bytes)

Return Code

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

5.5.1.6. Send RLS Request

This command sends an RLS request to target device.

RLS Request Command Format

Command	Class	INS	P1	P2	Lc	Data In	
RLS request	E0h	00h	00h	45h	02h	11h	DID (1 byte)

Where:

DID 1 byte. Device Identification.

RLS Request Response Format

Response	Class	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	02h	Return Code (2 bytes)

Return Code

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

5.5.2. Target Mode-related Commands

This section provides the commands that can be used when in Target Mode. The figure below shows the peer-to-peer flow of commands for Target Mode.

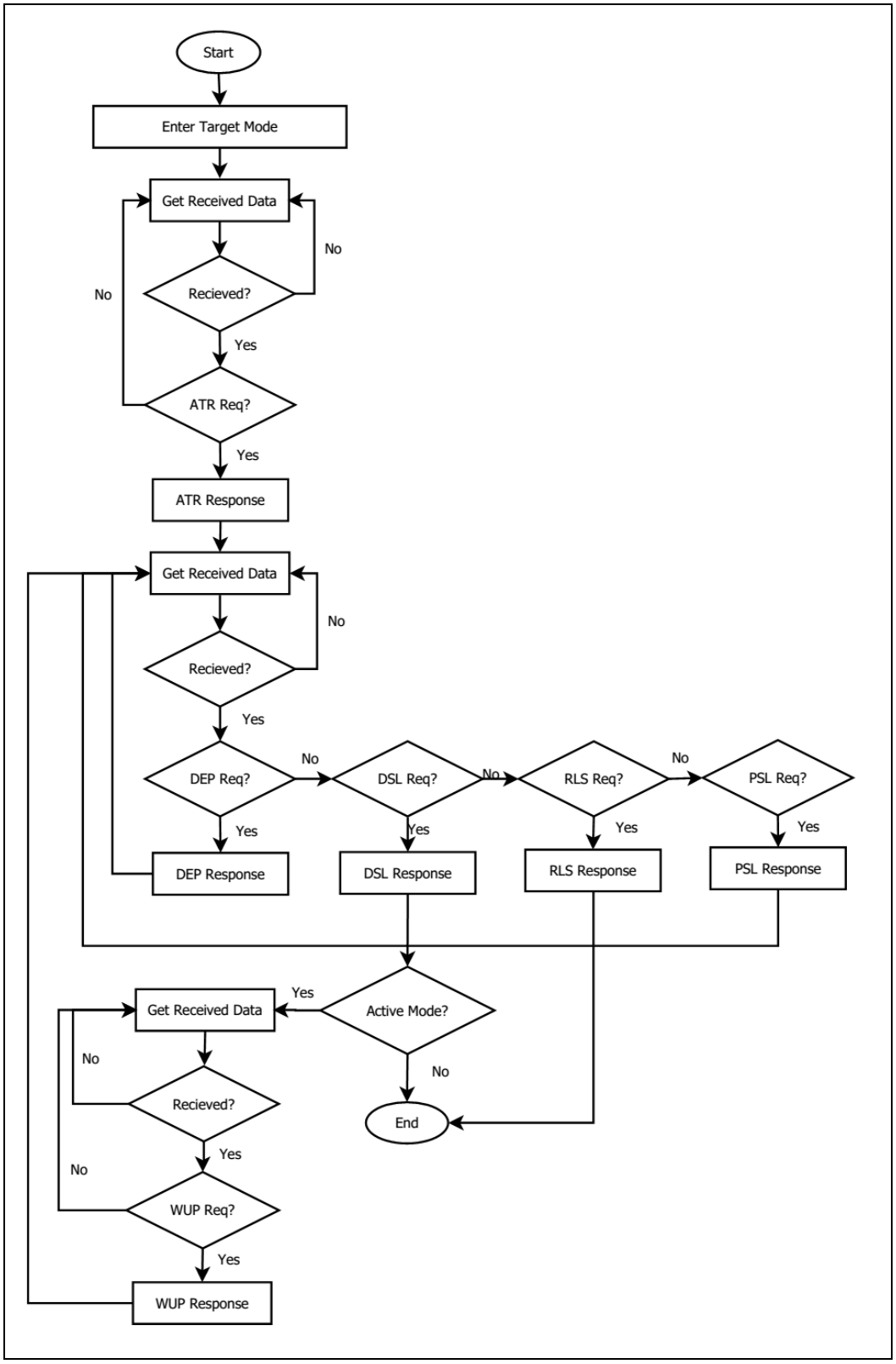


Figure 5: Peer-to-Peer Flow for Target Mode

5.5.2.1. Set Target Mode Timeout

This command sets the timeout for the reader in Target Mode.

Set Target Timeout Command Format

Command	Class	INS	P1	P2	Lc	Data In	
Set Target Timeout	E0h	00h	00h	59h	02h	Timeout (MSB)	Timeout (LSB)

Note: *Unit = 100 µs, default value of Target Timeout = 00 C8h (200 * 100 µs = 20 ms).*

Set Target Timeout Response Format

Response	Class	INS	P1	P2	Le	Data Out	
Result	E1h	00h	00h	00h	02h	Timeout (MSB)	Timeout (LSB)

Where:

Timeout 2 bytes. Timeout for Target Mode (unit =100 µs).

5.5.2.2. Enter Target Mode

This command sets the reader into Target Mode to receive SNEP Message.

Enter Target Mode Command Format (8 bytes)

Command	Class	INS	P1	P2	Lc	Data In	
Enter Target Mode	E0h	00h	00h	51h	02h	Speed	OpMode

Enter Target Mode Response Format (8 bytes)

Response	Class	INS	P1	P2	Le	Data Out	
Result	E1h	00h	00h	00h	02h	Speed	OpMode

Enter Target Mode Response Format

Response	Class	INS	P1	P2	Le	Data Out	
Result	E1h	00h	00h	00h	02h	Speed	OpMode

Where:

Speed 1 byte. Communication Speed.
01h = 106 Kbps
02h = 212 Kbps
03h = 424 Kbps

OpMode 1 byte. Active Mode/Passive Mode.
01h = Active Mode
02h = Passive Mode

After executing Enter Target Mode, the reader will wait for the NFC device, which in Initiator Mode, will present and receive the SNEP message. The reader will stop all other tasks until the SNEP Message is exchanged successfully.

5.5.2.3. Send ATR Response

This command sends an ATR response for the Initiator's ATR request.

ATR Response Command Format

Command	Class	INS	P1	P2	Lc	Data In
ATR Response	E0h	00h	00h	52h	Len	LLCP Parameter (N bytes)

Where:

LLCP Parameter N bytes. ATR response's General Byte.

ATR Response Format

Response	Class	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	02h	Return Code (2 bytes)

Return Code

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

5.5.2.4. Send DEP Response

This command sends a DEP response for the Initiator's DEP request.

DEP Response Command Format

Command	Class	INS	P1	P2	Lc	Data In	
DEP Response	E0h	00h	00h	53h	Len	PFB (1 byte)	DEP Message (N bytes)

Where:

PFB 1 byte. Control the data transmission and error recovery.

DEP Message N bytes. DEP response.

DEP Response Format

Response	Class	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	02h	Return Code (2 bytes)

Return Code

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

5.5.2.5. Send DSL Response

This command sends a DSL response for the Initiator's DSL request.

DSL Response Command Format

Command	Class	INS	P1	P2	Lc
DSL Response	E0h	00h	00h	54h	00h

DSL Response Format

Response	Class	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	02h	Return Code (2 bytes)

Return Code

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

5.5.2.6. Send RLS Response

This command sends an RLS response for the Initiator's RLS request.

RLS Response Command Format

Command	Class	INS	P1	P2	Lc
RLS Response	E0h	00h	00h	55h	00h

RLS Response Format

Response	Class	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	02h	Return Code (2 bytes)

Return Code

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

5.5.2.7. Send PSL Response

This command sends a PSL response for the Initiator's PSL request.

PSL Response Command Format

Command	Class	INS	P1	P2	Lc	Data In	
PSL Response	E0h	00h	00h	56h	02h	BRS (1 byte)	FSL (1 byte)

Where:

BRS 1 byte. BRS Parameter.

FSL 1 byte. FSL Parameter.

PSL Response Format

Response	Class	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	02h	Return Code (2 bytes)

Return Code

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

5.5.2.8. Send WUP Response

This command sends a WUP response for the Initiator's WUP request.

WUP Response Command Format

Command	Class	INS	P1	P2	Lc
WUP Response	E0h	00h	00h	57h	00h

WUP Response Format

Response	Class	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	02h	Return Code (2 bytes)

Return Code

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

5.5.2.9. Get Received Data

This command gets the data received from NFC Initiator device.

Get Received Data Command Format (5 bytes)

Command	Class	INS	P1	P2	Lc
Get Received Data	E0h	00h	00h	58h	00h

Get Received Data Response Format (11 bytes)

Response	Class	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	SNEP Message Len	SNEP Message

Where:

SNEP Message Len 1 byte. Length of the received SNEP Message.

SNEP Message Received SNEP message from Initiator devices.

5.6. NFC Card Emulation Mode-related Commands

5.6.1. Enter Card Emulation Mode

This command sets the reader into card emulation mode in order to emulate a MIFARE Ultralight or a FeliCa Card.

Note: *Lock byte is not supported in emulated MIFARE Ultralight. UID is user programmable.*

Enter Card Emulation Mode Command Format (8 bytes)

Command	Class	INS	P1	P2	Lc	Data In		
Enter Card Emulation Mode	E0h	00h	00h	40h	03h	NFCMode	00h	00h

Enter Card Emulation Mode Response Format (8 bytes)

Response	Class	INS	P1	P2	Le	Data Out		
Result	E1h	00h	00h	00h	03h	NFCMode	01h	01h

Where:

NFCMode 1 byte. NFC Device Mode.

01h = MIFARE Ultralight Card Emulation Mode

03h = FeliCa Card Emulation Mode

06h = Peer-to-Peer Initiator Mode

Other = Card Read/Write Mode

Byte Number	0	1	2	3	Byte Address access by USB
Serial Number	SN0	SN1	SN2	BCC0	Nil
Serial Number	SN3	SN4	SN5	SN6	Nil
Internal/Lock	BCC1	Internal	Lock0	Lock1	Nil
Data read/write	Data0	Data1	Data2	Data3	0-3
Data read/write	Data4	Data5	Data6	Data7	4-7
Data read/write	Data8	Data9	Data10	Data11	8-11
Data read/write	Data12	Data13	Data14	Data15	12-15
Data read/write	Data16	Data17	Data18	Data19	16-19
Data read/write	Data20	Data21	Data22	Data23	20-23
Data read/write	Data24	Data25	Data26	Data27	24-27
Data read/write	Data28	Data29	Data30	Data31	28-31
Data read/write	Data32	Data33	Data34	Data35	32-35
Data read/write	Data36	Data37	Data38	Data39	36-39
Data read/write	Data40	Data41	Data42	Data43	40-43
Data read/write	Data44	Data45	Data46	Data47	44-47
Data read/write	Data48	Data49	Data50	Data51	48-51

Accessible area (52 bytes)

Table 5: MIFARE Ultralight Memory Map (52 bytes)

Where:

Default SN[0-6] {04h, 96h, 50h, 01h, F4h, 02h, 80h}

Default Data[0-3] {E1h, 10h, 06h, 00h} //NFC Type2Tag

Memory	1 Block data (16 Byte)	Byte Address access by USB
Data read/write	Block 0	0-15
Data read/write	Block 1	16-31
Data read/write	Block 2	32-47
Data read/write	Block 3	48-63
Data read/write	Block 4	64-79
Data read/write	Block 5	80-95
Data read/write	Block 6	96-111
Data read/write	Block 7	112-127
Data read/write	Block 8	128-143
Data read/write	Block 9	144-159

Table 6: FeliCa Memory Map (160 bytes)

Where:

Default: Block 0 data: {10h, 01h, 01h, 00h, 09h, 00h, 00h, 00h, 00h, 00h, 01h, 00h, 00h, 00h, 00h, 1Ch}

Default Block 0 data NFC Type3 Tag Attribute Information Block

Notes:

1. *FeliCa card emulation support Read/Write without Encryption*
2. *FeliCa Card Identification Number in IDm is user programmable while Manufacturer Code is fixed at (03 88).*

5.6.2. Read Card Emulation Data (MIFARE Ultralight or FeliCa)

This command reads the emulated card content.

Read Card Emulation Data Command Format (9 bytes)

Command	Class	INS	P1	P2	Lc	Data In			
Read Card Emulation Data	E0h	00h	00h	60h	04h	00h	NFCMode	StartOffset	Length

Read Card Emulation Data Response Format (Data + 5 bytes)

Response	Class	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	Length	Data being read

Where:

NFCMode 1 byte. NFC Device Mode.
01h = MIFARE Ultralight Card Emulation Mode
03h = FeliCa Card Emulation Mode

StartOffset 1 byte. Address start to read.

Length 1 byte. Number of byte going to read.

Data being read The output data.

5.6.3. Write Card Emulation Data (MIFARE Ultralight or FeliCa)

This command writes on the emulated card.

Write Card Emulation Data Command Format

Command	Class	INS	P1	P2	Lc	Data In				
Write Card Emulation Data	E0h	00h	00h	60h	Length + 4	01h	NFCMode	StartOffset	Length	Data to write

Write Card Emulation Data Response Format (8 bytes)

Response	Class	INS	P1	P2	Le	Data Out		
Result	E1h	00h	00h	00h	03h	Length	90h	00h

Where:

NfcMode 1 byte. NFC Device Mode.
01h = MIFARE Ultralight Card Emulation Mode
03h = FeliCa Card Emulation Mode

StartOffset 1 byte. Address start to read.

Length 1 byte. Number of bytes to write.

Data to Write The binary data to write.

5.6.4. Set Card Emulation of MIFARE Ultralight UID

This command sets the UID of the emulated MIFARE Ultralight card.

Set Card Emulation MIFARE Ultralight UID Command Format (12 bytes)

Command	Class	INS	P1	P2	Lc	Data In
Set Card Emulation Mifare Ultralight UID	E0h	00h	00h	61h	07h	7 bytes UID

Set Card Emulation MIFARE Ultralight UID Response Format (7 bytes)

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

Where:

UID 7 bytes. 7 bytes MIFARE UID.

5.6.5. Set Card Emulation FeliCa IDm

This command sets the 6-byte FeliCa Card Identification number on emulated FeliCa card.

Set Card Emulation FeliCa Card Identification number Command Format (11 bytes)

Command	Class	INS	P1	P2	Lc	Data In
Set Card Emulation FeliCa IDm	E0h	00h	00h	64h	06h	IDm

Set Card Emulation FeliCa Card Identification number Response Format (11 bytes)

Response	Class	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	06h	IDm

Where:

IDm 6 bytes.

5.6.6. Set Card Emulation Lock Data in NFC

This command sets the lock for card emulation data in NFC communication. If the data is locked, it is protected from being overwritten via NFC.

Set Card Emulation Lock Data in NFC Command Format (6 bytes)

Command	Class	INS	P1	P2	Lc	Data In
Set Card Emulation Lock Data	E0h	00h	00h	65h	01h	Lock

Set Card Emulation lock data in NFC Response Format (7 bytes)

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

Where:

Lock 1 byte. Protect the data from being overwritten via NFC.

Lock Parameter

Bit	Parameter	Description	Option
7-2	Reserved	Reserved	
1	FeliCa Lock Enable	Data cannot be modified via NFC. The data can still be modified by using the USB escape command.	0: Lock disable 1: Lock enable
0	MIFARE Ultralight Lock Enable		0: Lock disable 1: Lock enable

5.7. ACR122U Compatible Commands

5.7.1. Bi-color LED and Buzzer Control

This command controls the states of the Bi-Color LED and Buzzer.

Bi-Color LED and Buzzer Control Command Format (9 bytes)

Command	Class	INS	P1	P2	Lc	Data In (4 Bytes)
Bi-Color LED and Buzzer Control	FFh	00h	40h	LED State Control	04h	Blinking Duration Control

P2: LED State Control

Bi-color LED and Buzzer Control Format (1 byte)

CMD	Item	Description
Bit 0	Final Red LED State	1 = On; 0 = Off
Bit 1	Final Green LED State	1 = On; 0 = Off
Bit 2	Red LED State Mask	1 = Update the State 0 = No change
Bit 3	Green LED State Mask	1 = Update the State 0 = No change
Bit 4	Initial Red LED Blinking State	1 = On; 0 = Off
Bit 5	Initial Green LED Blinking State	1 = On; 0 = Off
Bit 6	Red LED Blinking Mask	1 = Blink 0 = Not Blink
Bit 7	Green LED Blinking Mask	1 = Blink 0 = Not Blink

Data In: Blinking Duration Control

Bi-color LED Blinking Duration Control Format (4 bytes)

Byte 0	Byte 1	Byte 2	Byte 3
T1 Duration Initial Blinking State (Unit = 100 ms)	T2 Duration Toggle Blinking State (Unit = 100 ms)	Number of repetition	Link to Buzzer

Where:

Byte 3 Link to Buzzer. Control the buzzer state during the LED Blinking.

00h = The buzzer will not turn on.

01h = The buzzer will turn on during the T1 Duration.

02h = The buzzer will turn on during the T2 Duration.

03h = The buzzer will turn on during the T1 and T2 Duration.

Data Out SW1 SW2. Status Code returned by the reader.

Status Code

Results	SW1	SW2	Meaning
Success	90h	Current LED State	The operation was completed successfully.
Error	63h	00h	The operation failed.

Current LED State (1 byte)

Status	Item	Description
Bit 0	Current Red LED	1 = On; 0 = Off
Bit 1	Current Green LED	1 = On; 0 = Off
Bits 2 – 7	RFU	RFU

Reminders:

1. *The LED State operation will be performed after the LED Blinking operation is completed.*
2. *The LED will not change if the corresponding LED Mask is not enabled.*
3. *The LED will not blink if the corresponding LED Blinking Mask is not enabled. Also, the number of repetition must be greater than zero.*
4. *T1 and T2 duration parameters are used for controlling the duty cycle of LED blinking and Buzzer Turn-On duration. For example, if T1=1 and T2=1, the duty cycle = 50%.*

   Note: *Duty Cycle = T1 / (T1 + T2).*
5. *To control the buzzer only, just set the P2 "LED State Control" to zero.*
6. *To make the buzzer operate, the "number of repetition" must be greater than zero.*
7. *To control the LED only, just set the parameter "Link to Buzzer" to zero.*

5.7.2. Get Firmware Version

This command retrieves the firmware version of the reader.

Get Firmware Version Command Format (5 bytes)

Command	Class	INS	P1	P2	Le
Get Firmware	FFh	00h	48h	00h	00h

Get Firmware Version Response Format (X bytes)

Response	Data Out
Result	Firmware Version

Example:

Response = 41 43 52 31 32 35 32 55 5F 56 31 30 30 2E 31h = ACR1252U_V100.1 (ASCII)

5.7.3. Get the PICC Operating Parameter

This command gets the PICC operating parameter of the reader.

Get the PICC Operating Parameter Command Format (5 bytes)

Command	Class	INS	P1	P2	Le
Get PICC Operation Parameter	FFh	00h	50h	00h	00h

Get the PICC Operating Parameter Response Format (2 bytes)

Response	Data Out	
Result	90h	PICC Operating Parameter

PICC Operating Parameter

Bit	Parameter	Description	Option
7	Auto PICC Polling	To enable the PICC Polling	1 = Enable 0 = Disable
6	Auto ATS Generation	To issue ATS Request whenever an ISO14443-4 Type A tag is activated	1 = Enable 0 = Disable
5	Polling Interval	To set the time interval between successive PICC Polling.	1 = 250 ms 0 = 500 ms
4	FeliCa 424 Kbps	The Tag Types to be detected during PICC Polling.	1 = Detect 0 = Skip
3	FeliCa 212 Kbps		1 = Detect 0 = Skip
2	Topaz		1 = Detect 0 = Skip
1	ISO14443 Type B		1 = Detect 0 = Skip
0	ISO14443 Type A ***Note:*** *To detect the MIFARE Tags, the Auto ATS Generation must be disabled first.*		1 = Detect 0 = Skip

5.7.4. Set the PICC Operating Parameter

This command sets the PICC operating parameter of the reader.

Set PICC operation Parameter Command Format (5 bytes)

Command	Class	INS	P1	P2	Le
Set PICC Operation Parameter	FFh	00h	51h	PICC Operating Parameter	00h

Set PICC operation Parameter Response Format (2 bytes)

Response	Data Out	
Result	90h	PICC Operating Parameter

PICC Operating Parameter

Bit	Parameter	Description	Option
7	Auto PICC Polling	To enable the PICC Polling	1 = Enable 0 = Disable
6	Auto ATS Generation	To issue ATS Request whenever an ISO14443-4 Type A tag is activated	1 = Enable 0 = Disable
5	Polling Interval	To set the time interval between successive PICC Polling.	1 = 250 ms 0 = 500 ms
4	FeliCa 424 Kbps	The Tag Types to be detected during PICC Polling.	1 = Detect 0 = Skip
3	FeliCa 212 Kbps		1 = Detect 0 = Skip
2	Topaz		1 = Detect 0 = Skip
1	ISO14443 Type B		1 = Detect 0 = Skip
0	ISO14443 Type A ***Note:*** *To detect the MIFARE tags, the Auto ATS Generation must be disabled first.*		1 = Detect 0 = Skip

Appendix A. SNEP Message

For the data format, please refer to NFC Forum NFC Data Exchange Format (NDEF) Specifications 1.0.

Example:

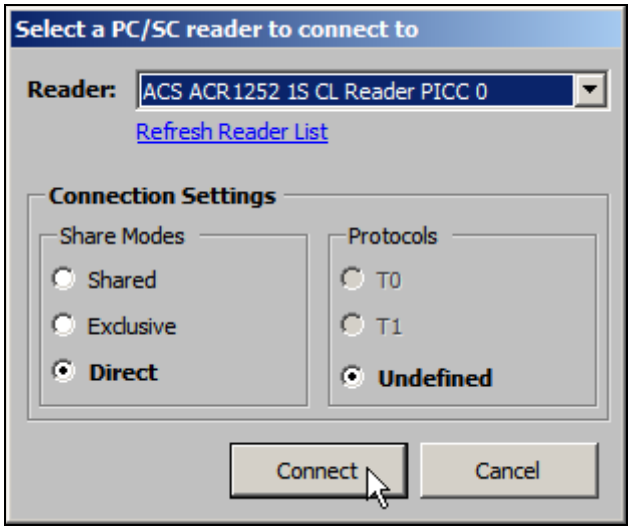
SNEP Message = {D1 02 0F 53 70 D1 01 0B 55 01 61 63 73 2E 63 6F 6D 2E 68 6Bh}

Offset	Content	Length	Description
0	D1	1	NDEF header. TNF = 01h, SR=1, MB=1, ME=1
1	02	1	Record name length (2 bytes)
2	0F	1	Length of the Smart Poster data (15 bytes)
3	53 70 (“Sp”)	2	Record name
5	D1	1	NDEF header. TNF = 01h, SR=1, MB=1, ME=1
6	01	1	Record name length (1 byte)
7	0B	1	The length of the URI payload (11 bytes)
8	55 (“U”)	1	Record type: “U”
9	01	1	Abbreviation: “http://www.”
10	61 63 73 2E 63 6F 6D 2E 68 6B	10	The URL itself. “acs.com.hk”

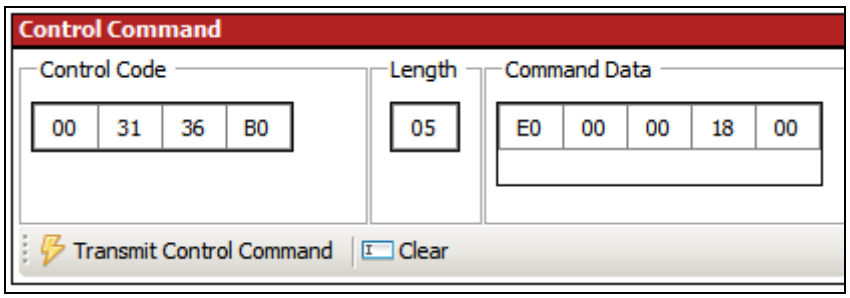
Appendix B. Escape Command Example

Example: Get firmware version using the ACR1252U Reader Tool.

1. Plug in the ACR1252U Reader to your computer.
2. Run the **ACR1252U Reader Tool**.
3. Connect the reader using **Direct Mode**.



4. Go to the **Control Transmit** tab. In the **Length** field, type in *05*.
5. In the **Command Data** field, type in *E0 00 00 18 00* (APDU for Get Firmware Version command).



6. Click **Transmit Control Command**, and then check the Respond Data.

   e.g.,Response Data = E1 00 00 00 0F **41 43 52 31 32 35 32 55 5F 56 31 30 30 2E 31**

   Firmware Version (HEX) = **41 43 52 31 32 35 32 55 5F 56 31 30 30 2E 31**

   Firmware Version (ASCII) = ACR1252U_V100.1