



**Advanced Card Systems Ltd.**  
Card & Reader Technologies

# ACR3901T-W1 Secure Bluetooth® Contact Card Reader



Reference Manual V1.01



## Revision History

Release Date	Revision Description	Version Number
2017-08-31	<ul style="list-style-type: none"><li>Initial Release</li></ul>	1.00
2017-10-02	<ul style="list-style-type: none"><li>Updated Section 5.2.1: Battery Life</li><li>Updated Section 6.3: Authentication</li><li>Updated Section 6.6: Mutual Authentication</li><li>Updated Section 6.6.1: SPH_to_RDR_ReqAuth</li><li>Updated Section 6.6.3: SPH_to_RDR_AuthRsp</li><li>Updated Section 6.6.5: SPH_to_RDR_DataReq</li><li>Updated Section 6.6.6: RDR_to_SPH_DataRsp</li><li>Updated Section 8.1.3 Get Firmware Version Command</li><li>Updated Section 8.1.14: Default Customer Master Key Reset Request</li></ul>	1.01



## Table of Contents

<b>1.0.</b>	<b>Introduction .....</b>	<b>7</b>
1.1.	Reference Documents .....	7
1.2.	Symbols and Abbreviations .....	7
<b>2.0.</b>	<b>Features .....</b>	<b>8</b>
<b>3.0.</b>	<b>Smart Card Support .....</b>	<b>9</b>
3.1.	MCU Cards .....	9
3.2.	Memory-based Smart Cards.....	9
<b>4.0.</b>	<b>System Block Diagram .....</b>	<b>10</b>
<b>5.0.</b>	<b>Hardware Design .....</b>	<b>11</b>
5.1.	Battery.....	11
5.1.1.	Battery charging.....	11
5.1.2.	Battery life .....	11
5.2.	Bluetooth Interface.....	11
5.3.	USB Interface .....	12
5.3.1.	Communication Parameters .....	12
5.3.2.	Endpoints .....	12
5.4.	User Interface .....	13
5.4.1.	Buttons .....	13
5.4.2.	Status LEDs .....	13
5.5.	Smart Card Interface .....	14
5.5.1.	Smart Card Power Supply VCC (C1).....	14
5.5.2.	Programming Voltage VPP (C6) .....	14
5.5.3.	Card Type Selection.....	14
5.5.4.	Interface for Microcontroller-based Cards .....	14
5.5.5.	Card Tearing Protection.....	14
<b>6.0.</b>	<b>Software Design .....</b>	<b>15</b>
6.1.	Bluetooth Connection Program Flow .....	15
6.2.	Profile Selection .....	16
6.3.	Authentication .....	18
6.4.	Frame Format .....	19
6.4.1.	Bluetooth Frame Format .....	19
6.4.2.	Bluetooth Frame Format after Mutual Authentication .....	19
6.5.	Bluetooth Communication Protocol .....	20
6.5.1.	Card Power On .....	21
6.5.2.	Card Power Off .....	22
6.5.3.	Get Card Presence .....	23
6.5.4.	APDU Command.....	24
6.5.5.	Escape Command.....	25
6.5.6.	Card Set Parameters .....	26
6.6.	Mutual Authentication and Encryption Protocol.....	29
6.6.1.	SPH_to_RDR_ReqAuth .....	29
6.6.2.	RDR_to_SPH_AuthRsp1 .....	30
6.6.3.	SPH_to_RDR_AuthRsp .....	31
6.6.4.	RDR_to_SPH_AuthRsp2.....	32
6.6.5.	SPH_to_RDR_DataReq.....	32
6.6.6.	RDR_to_SPH_DataRsp.....	33
<b>7.0.</b>	<b>USB Communication Protocol.....</b>	<b>34</b>
7.1.	CCID Bulk-OUT Messages.....	36
7.1.1.	PC_to_RDR_IccPowerOn.....	36
7.1.2.	PC_to_RDR_IccPowerOff.....	36
7.1.3.	PC_to_RDR_GetSlotStatus .....	36



7.1.4.	PC_to_RDR_XfrBlock.....	37
7.1.5.	PC_to_RDR_GetParameters.....	37
7.1.6.	PC_to_RDR_ResetParameters.....	37
7.1.7.	PC_to_RDR_SetParameters.....	38
7.1.8.	PC_to_RDR_Escape.....	39
7.2.	CCID Bulk-IN Messages.....	40
7.2.1.	RDR_to_PC_DataBlock.....	40
7.2.2.	RDR_to_PC_SlotStatus.....	40
7.2.3.	RDR_to_PC_Parameters.....	41
7.2.4.	RDR_to_PC_Escape.....	41
<b>8.0.</b>	<b>Smart Card Protocol .....</b>	<b>42</b>
8.1.	Peripherals Control.....	42
8.1.1.	Get Serial Number Command.....	42
8.1.2.	Get Random Number Command.....	43
8.1.3.	Get Firmware Version Command.....	44
8.1.4.	Rewrite Master Key Command.....	45
8.1.5.	Sleep Mode Option.....	46
8.1.6.	Get Device Address.....	47
8.1.7.	Set Tx Power.....	48
8.1.8.	Read Tx Power value.....	49
8.1.9.	Set Card Response Time Interval.....	50
8.1.10.	Get Card Response Time Interval.....	51
8.1.11.	Set Card Simulation Option.....	52
8.1.12.	Get Card Simulation Option.....	53
8.1.13.	Check button status.....	54
8.1.14.	Customer Master Key Reset Request.....	55
<b>9.0.</b>	<b>Memory Card Command Set.....</b>	<b>56</b>
9.1.	Memory Card – 1, 2, 4, 8, and 16 kilobit I2C Card.....	56
9.1.1.	SELECT_CARD_TYPE.....	56
9.2.	Memory Card – 32, 64, 128, 256, 512, and 1024 kilobit I2C Card.....	59
9.2.1.	SELECT_CARD_TYPE.....	59
9.2.2.	SELECT_PAGE_SIZE.....	59
9.2.3.	READ_MEMORY_CARD.....	60
9.2.4.	WRITE_MEMORY_CARD.....	60
9.3.	Memory Card – ATMEL AT88SC153.....	62
9.3.1.	SELECT_CARD_TYPE.....	62
9.3.2.	READ_MEMORY_CARD.....	62
9.3.3.	WRITE_MEMORY_CARD.....	63
9.3.4.	VERIFY_PASSWORD.....	64
9.3.5.	INITIALIZE_AUTHENTICATION.....	64
9.3.6.	VERIFY_AUTHENTICATION.....	65
9.4.	Memory Card – ATMEL AT88C1608.....	66
9.4.1.	SELECT_CARD_TYPE.....	66
9.4.2.	READ_MEMORY_CARD.....	66
9.4.3.	WRITE_MEMORY_CARD.....	67
9.4.4.	VERIFY_PASSWORD.....	68
9.4.5.	INITIALIZE_AUTHENTICATION.....	68
9.4.6.	VERIFY_AUTHENTICATION.....	69
9.5.	Memory Card – SLE4418/SLE4428/SLE5518/SLE5528.....	70
9.5.1.	SELECT_CARD_TYPE.....	70
9.5.2.	READ_MEMORY_CARD.....	70
9.5.3.	READ_PRESENTATION_ERROR_COUNTER_MEMORY_CARD (SLE4428 and SLE5528) 71	71
9.5.4.	READ_PROTECTION_BIT.....	71
9.5.5.	WRITE_MEMORY_CARD.....	72
9.5.6.	WRITE_PROTECTION_MEMORY_CARD.....	73
9.5.7.	PRESENT_CODE_MEMORY_CARD (SLE4428 and SLE5528).....	73
9.6.	Memory Card – SLE4432/SLE4442/SLE5532/SLE5542.....	75



9.6.1.	SELECT_CARD_TYPE .....	75
9.6.2.	READ_MEMORY_CARD.....	75
9.6.3.	READ_PRESENTATION_ERROR_COUNTER_MEMORY_CARD (SLE 4442 and SLE 5542) 76	
9.6.4.	READ_PROTECTION_BITS .....	76
9.6.5.	WRITE_MEMORY_CARD.....	77
9.6.6.	WRITE_PROTECTION_MEMORY_CARD .....	77
9.6.7.	PRESENT_CODE_MEMORY_CARD (SLE 4442 and SLE 5542).....	78
9.6.8.	CHANGE_CODE_MEMORY_CARD (SLE 4442 and SLE 5542) .....	79
9.7.	Memory Card – SLE 4406/SLE 4436/SLE 5536/SLE 6636 .....	80
9.7.1.	SELECT_CARD_TYPE .....	80
9.7.2.	READ_MEMORY_CARD.....	80
9.7.3.	WRITE_ONE_BYTE_MEMORY_CARD.....	81
9.7.4.	PRESENT_CODE_MEMORY_CARD .....	82
9.7.5.	AUTHENTICATE_MEMORY_CARD (SLE 4436, SLE 5536 and SLE 6636).....	83
9.8.	Memory Card – SLE 4404 .....	85
9.8.1.	SELECT_CARD_TYPE .....	85
9.8.2.	READ_MEMORY_CARD.....	85
9.8.3.	WRITE_MEMORY_CARD .....	86
9.8.4.	ERASE_SCRATCH_PAD_MEMORY_CARD .....	86
9.8.5.	VERIFY_USER_CODE.....	87
9.8.6.	VERIFY_MEMORY_CODE .....	88
9.9.	Memory Card – AT88SC101/AT88SC102/AT88SC1003.....	89
9.9.1.	SELECT_CARD_TYPE .....	89
9.9.2.	READ_MEMORY_CARD.....	89
9.9.3.	WRITE_MEMORY_CARD.....	90
9.9.4.	ERASE_NON_APPLICATION_ZONE .....	90
9.9.5.	ERASE_APPLICATION_ZONE_WITH_ERASE .....	91
9.9.6.	ERASE_APPLICATION_ZONE_WITH_WRITE_AND_ERASE .....	92
9.9.7.	VERIFY_SECURITY_CODE .....	93
9.9.8.	BLOWN_FUSE .....	94
<b>10.0.</b>	<b>Other Commands Access via PC_to_RDR_XfrBlock.....</b>	<b>96</b>
10.1.	GET_READER_INFORMATION .....	96
<b>Appendix A. Supported Card Types .....</b>		<b>97</b>
<b>Appendix B. Error Codes.....</b>		<b>98</b>

## List of Figures

<b>Figure 1 :</b>	ACR3901T-W1 Architecture .....	10
<b>Figure 2 :</b>	Bluetooth Connection Flow .....	15
<b>Figure 3 :</b>	nRFgo Studio GATT Setting Interface .....	16
<b>Figure 4 :</b>	Authentication Procedure.....	18

## List of Tables

<b>Table 1 :</b>	Symbols and Abbreviations .....	7
<b>Table 2 :</b>	Estimated Battery Lifespan.....	11
<b>Table 3 :</b>	USB Interface Wiring .....	12
<b>Table 4 :</b>	LED Status.....	13
<b>Table 5 :</b>	ACR3901T-W1 Service Handles and UUID Information List .....	17
<b>Table 6 :</b>	Bluetooth Frame Format.....	19
<b>Table 7 :</b>	Encrypted Frame Format after Mutual Authentication.....	19



<b>Table 8</b> : Command Code Summary .....	20
<b>Table 9</b> : Response Code Summary .....	20
<b>Table 10</b> : Summary of Mutual Authentication Commands .....	29
<b>Table 11</b> : Supported Card Types .....	97
<b>Table 12</b> : Error Code .....	98



## 1.0. Introduction

ACR3901T-W1 Secure Bluetooth® SIM-sized Contact Card Reader acts as an interface for the communication between a smart card and a computer/mobile device. The direct communication, in most cases, is prevented by various types of smart cards that have commands and communication protocols different from each other. Through ACR3901T-W1, a uniform interface is established for a wide variety of cards. By taking care of the card's particulars, it releases the computer software programmer from being responsible with smart card operations' technical details, which in many cases, are not relevant to the implementation of a smart card system.

### 1.1. Reference Documents

The following related documents are available from [www.usb.org](http://www.usb.org)

- Universal Serial Bus Specification 2.0 (also referred to as the USB specification), April 27, 2000
- Universal Serial Bus Common Class Specification 1.0, December 16, 1997
- Universal Serial Bus Device Class: Smart Card CCID Specification for Integrated Circuit(s) Cards Interface Devices, Revision 1.1, April 22, 2005

The following related documents can be ordered through [www.ansi.org](http://www.ansi.org)

- ISO/IEC 7816-1; Identification Cards – Integrated circuit(s) cards with contacts - Part 1: Physical Characteristics
- ISO/IEC 7816-2; Identification Cards – Integrated circuit(s) cards with contacts - Part 2: Dimensions and Locations of the contacts
- ISO/IEC 7816-3; Identification Cards – Integrated circuit(s) cards with contacts - Part 3: Electronic signals and transmission protocols

### 1.2. Symbols and Abbreviations

Abbreviation	Description
ATR	Answer-To-Reset
CCID	Chip/Smart Card Interface Device
ICC	Integrated Circuit Cards
IFSC	Information Field Sized for ICC for protocol T=1
IFSD	Information Field Sized for CCID for protocol T=1
NAD	Node Address
PPS	Protocol and Parameters Selection
RFU	Reserved for future use <sup>1</sup>
TPDU	Transport Protocol Data Unit
USB	Universal Serial Bus

**Table 1:** Symbols and Abbreviations

<sup>1</sup> Must be set to zero unless stated differently.



## 2.0. Features

- USB Full Speed Interface
- Bluetooth Smart Interface
- Plug and Play – CCID support brings utmost mobility
- Smart Card Reader:
  - Supports ISO 7816 Class A, B, and C (5 V, 3 V, 1.8 V) SIM-sized cards
  - Supports microprocessor cards with T=0 or T=1 protocol
  - Supports memory cards
  - Supports PPS (Protocol and Parameters Selection)
  - Features Short Circuit Protection
  - Supports AES-128 encryption algorithm
- Application Programming Interface:
  - Supports PC/SC
  - Supports CT-API (through wrapper on top of PC/SC)
- Built-in Peripherals:
  - LEDs
  - Button
- USB Firmware Upgradeability<sup>2</sup>
- Supports Android™ 4.3 and later<sup>3</sup>
- Supports iOS 5.0 and later<sup>4</sup>
- Compliant with the following standards:
  - EN 60950/IEC 60950
  - ISO 7816
  - Bluetooth Smart
  - PC/SC
  - CCID
  - CE
  - FCC
  - RoHS 2
  - REACH
  - Microsoft® WHQL

---

<sup>2</sup> Applicable under PC-linked mode

<sup>3</sup> Uses an ACS-defined Android Library

<sup>4</sup> Uses an ACS-defined iOS Library





## 3.0. Smart Card Support

### 3.1. MCU Cards

ACR3901T-W1 is a PC/SC-compliant smart card reader that supports ISO 7816 Class A, B, and C (5 V, 3 V, and 1.8 V) smart cards. It also works with MCU cards following either the T=0 and T=1 protocol.

The card ATR indicates the specific operation mode (TA2 present; bit 5 of TA2 must be 0) and when that particular mode is not supported by the ACR3901T-W1, it will reset the card to negotiable mode. If the card cannot be set to negotiable mode, the reader will then reject the card.

When the card ATR indicates the negotiable mode (TA2 not present) and communication parameters other than the default parameters, the ACR3901T-W1 will execute the PPS and try to use the communication parameters that the card suggested in its ATR. If the card does not accept the PPS, the reader will use the default parameters (F=372, D=1).

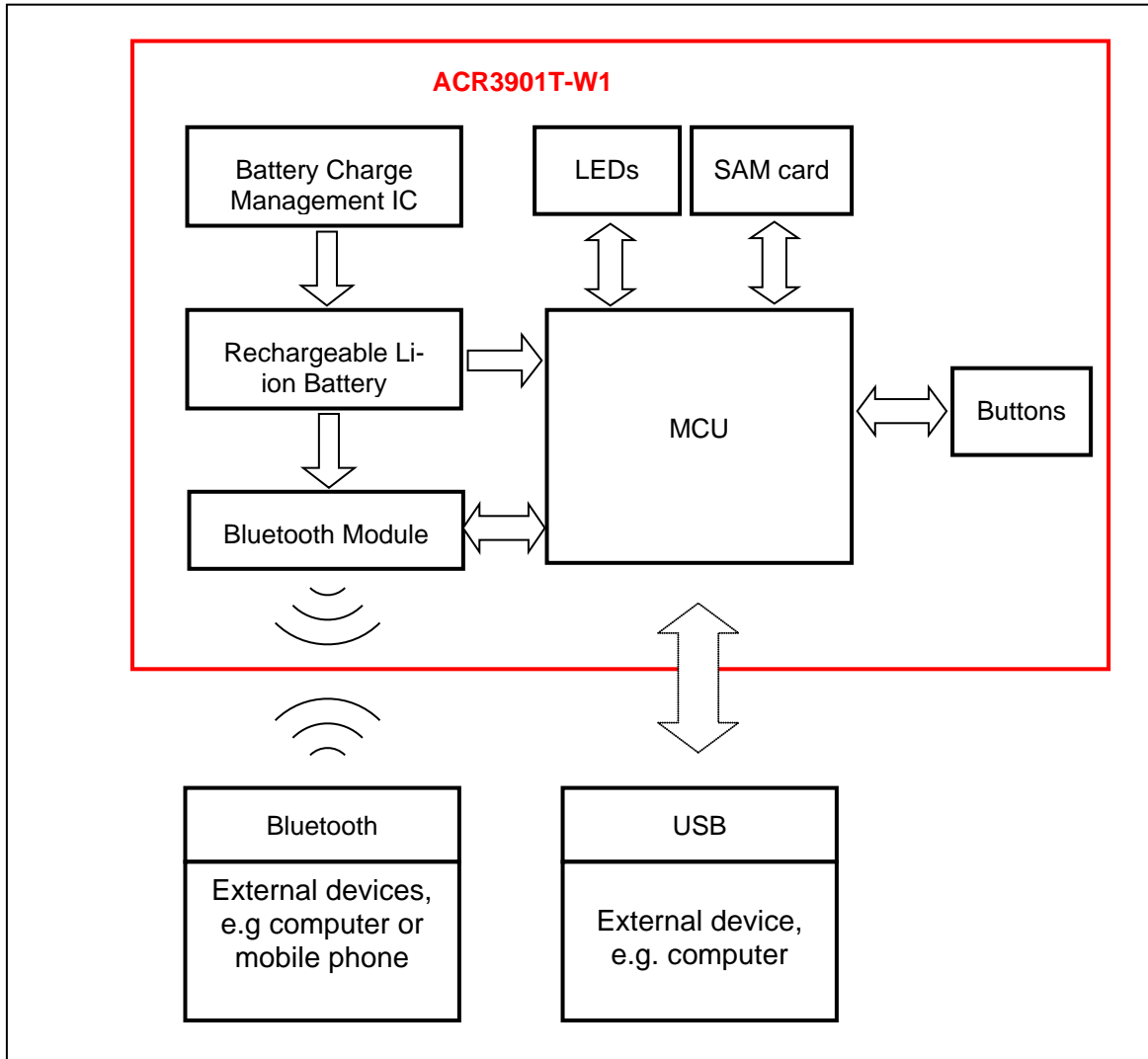
For the meaning of the aforementioned parameters, please refer to ISO 7816-3.

### 3.2. Memory-based Smart Cards

ACR3901T-W1 works with several memory-based smart cards such as:

- Cards following the I2C bus protocol (free memory cards) with maximum 128 bytes page with capability, including:
  - Atmel®: AT24C01/02/04/08/16/32/64/128/256/512/1024
  - SGS-Thomson: ST14C02C, ST14C04C
  - Gemplus: GFM1K, GFM2K, GFM4K, GFM8K
- Cards with secure memory IC with password and authentication, including:
  - Atmel®: AT88SC153 and AT88SC1608
- Cards with intelligent 1 KB EEPROM with write-protect function, including:
  - Infineon®: SLE4418, SLE4428, SLE5518 and SLE5528
- Cards with intelligent 256 bytes EEPROM with write-protect function, including:
  - Infineon®: SLE4432, SLE4442, SLE5532 and SLE5542
- Cards with '104' type EEPROM non-reloadable token counter cards, including:
  - Infineon®: SLE4406, SLE4436, SLE5536 and SLE6636
- Cards with Intelligent 416-bit EEPROM with internal PIN check, including:
  - Infineon®: SLE4404
- Cards with Security Logic with Application Zone(s), including:
  - Atmel®: AT88SC101, AT88SC102 and AT88SC1003

## 4.0. System Block Diagram



**Figure 1:** ACR3901T-W1 Architecture



## 5.0. Hardware Design

### 5.1. Battery

ACR3901T-W1 is using a rechargeable Lithium-ion battery which has a capacity of 70 mAh.

#### 5.1.1. Battery charging

Once the battery of ACR3901T-W1 runs out, it may be charged in any of the following modes: OFF, USB, Bluetooth; as long as it is connected to a power outlet.

#### 5.1.2. Battery life

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

Mode	Estimated Battery Life
Working Mode	4 days* <sup>(1)</sup>
Standby Mode	28 days* <sup>(2)</sup>
OFF Mode	1 year

**Table 2:** Estimated Battery Lifespan

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

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

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

### 5.2. Bluetooth Interface

ACR3901T-W1 uses Bluetooth Low Energy (LE) 4.0 as the medium to pair with computer/mobile device.



### 5.3. USB Interface

The micro-USB port is used to connect the ACR3901T-W1 to the computer as battery charging port. This port is also used in order for the ACR3901T-W1 to operate in PC-linked mode.

#### 5.3.1. Communication Parameters

ACR3901T-W1 is connected to a computer through USB as specified in the USB Specification 2.0. ACR3901T-W1 is working in full speed mode, i.e. 12 Mbps.

Pin	Signal	Function
1	V <sub>BUS</sub>	+5 V power supply for the reader
2	D-	Differential signal transmits data between ACR3901T-W1 and computer
3	D+	Differential signal transmits data between ACR3901T-W1 and computer
4	GND	Reference voltage level for power supply

**Table 3:** USB Interface Wiring

#### 5.3.2. Endpoints

ACR3901T-W1 uses the following endpoints to communicate with the host computer:

- Control Endpoint** For setup and control purpose
- Bulk OUT** For command to be sent from host to ACR3901T-W1 (data packet size is 64 bytes)
- Bulk IN** For response to be sent from ACR3901T-W1 to host (data packet size is 64 bytes)
- Interrupt IN** For card status message to be sent from ACR3901T-W1 to host (data packet size is 8 bytes)



## 5.4. User Interface

ACR3901T-W1 has two buttons and two LEDs (One Single Color: Red; One Bi-color: Green and Blue).

### 5.4.1. Buttons

ACR3901T-W1 has three modes: USB, Off, and Bluetooth. User can perform one mode at a time as a data transmission interface.

- Button (Front) - Power On, Power Off or Wake Up the reader; Simulate SAM card removal event.
  - Once pressed, the reader will power on and the default mode is the Bluetooth mode. Once connected through a computer, it will automatically switch to USB mode. Meanwhile, a long press will power off the device.
- Button (Back) - Reset the reader

### 5.4.2. Status LEDs

ACR3901T-W1 has two LEDs (One Single Color: Red; One Bi-color: Green and Blue) to show various operation status, where:

- **Red LED** - Battery status
- **Blue LED** - Card and reader status under Bluetooth mode
- **Green LED** - Card and reader status under USB mode

Color	LED Activity	Status
Red	On	The battery is charging (will turn OFF after battery is fully charged)
	Slow flash (1 second/flash)	The battery needs to be charged
Blue	Fast-Slow flash (Fast: 250 ms/flash; Slow: 500 ms/flash)	Ready for Bluetooth device pairing
	Slow flash (2 seconds/flash)	Bluetooth device connected and no card operation
	Fast flash	Data transferring between the reader and mobile device
	On	Card is connected and powered on
Green	Slow flash (2 seconds/flash)	No card operation and the reader is waiting for PC instructions
	Fast flash	Data transferring between the reader and PC
	On	Card is connected and powered on

**Table 4:** LED Status

**Note:** When red, blue and green LEDs are OFF, the reader is powered off. Both blue and green LEDs will light for 1 second, and then will turn off when the reader received some critical error codes from the Bluetooth module.



## 5.5. Smart Card Interface

The interface between the ACR3901T-W1 and the inserted smart card follows the specification of ISO 7816-3 with certain restrictions or enhancements to increase the practical functionality of ACR3901T-W1.

### 5.5.1. Smart Card Power Supply VCC (C1)

The current consumption of the inserted card must not be higher than 50 mA.

### 5.5.2. Programming Voltage VPP (C6)

According to ISO 7816-3, the smart card contact C6 (VPP) supplies the programming voltage to the smart card. Since all common smart cards in the market are EEPROM-based and do not require the provision of an external programming voltage, the contact C6 (VPP) has been implemented as a normal control signal in the ACR3901T-W1. The electrical specifications of this contact are identical to those of the signal RST (at contact C2).

### 5.5.3. Card Type Selection

The controlling computer must always select the card type through the proper command sent to the ACR3901T-W1 prior to activating the inserted card. This includes both the memory cards and MCU-based cards.

For MCU-based cards, the reader allows to select the preferred protocol, T=0 or T=1. However, this selection is only accepted and carried out by the reader through the PPS when the card inserted in the reader supports both protocol types. Whenever an MCU-based card supports only one protocol type, T=0 or T=1, the reader automatically uses that protocol type, regardless of the protocol type selected by the application.

### 5.5.4. Interface for Microcontroller-based Cards

For microcontroller-based smart cards, only the contacts C1 (VCC), C2 (RST), C3 (CLK), C5 (GND) and C7 (I/O) are used. A frequency of 4.8 MHz is applied to the CLK signal (C3).

### 5.5.5. Card Tearing Protection

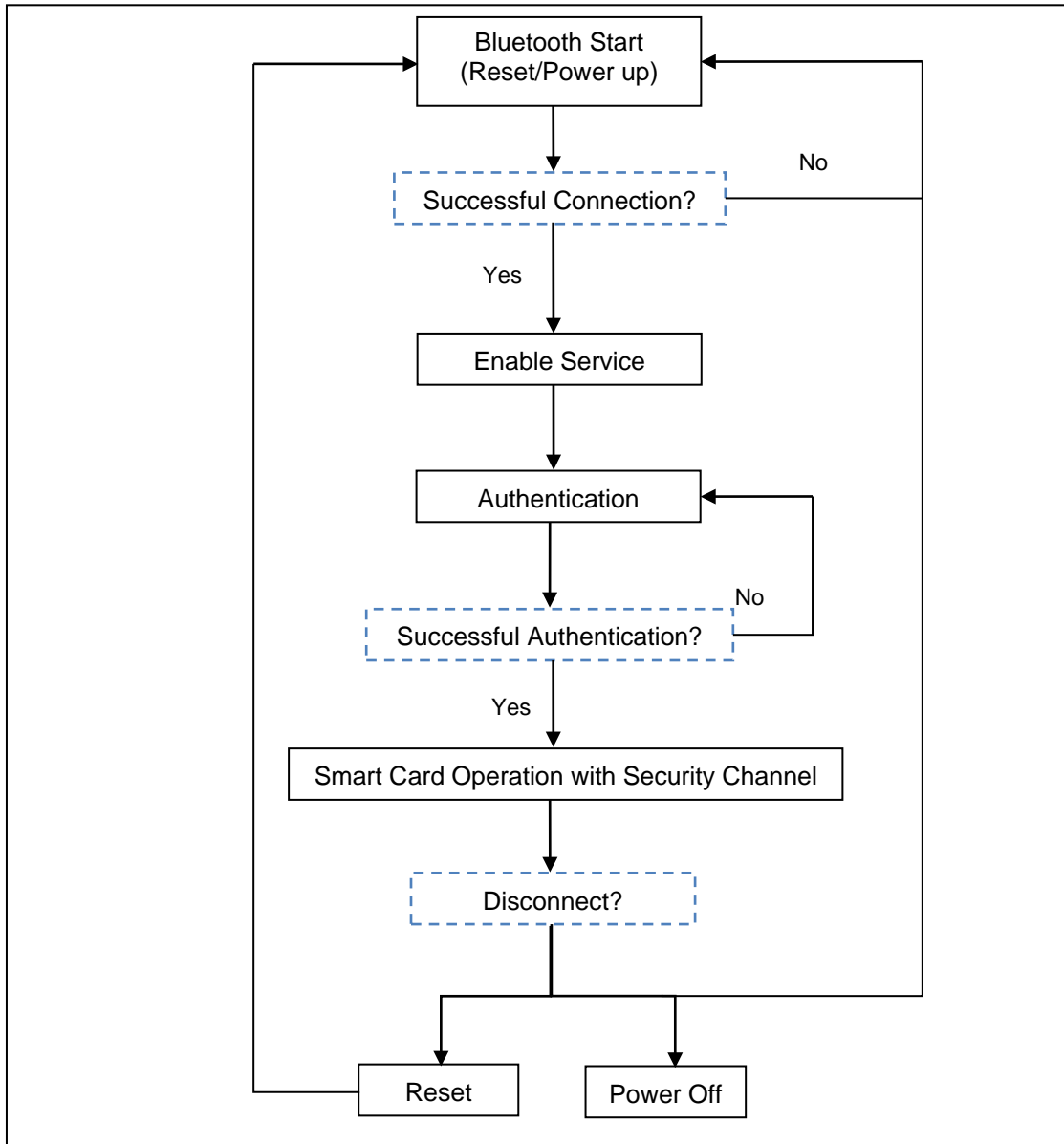
The ACR3901T-W1 provides a mechanism to protect the inserted card when it is suddenly withdrawn while it is powered up. The power supply to the card and the signal lines between the ACR3901T-W1 and the card is immediately deactivated when the card is being removed. However, as a rule to avoid any electrical damage, a card should only be removed from the reader while it is powered down.

**Note:** ACR3901T-W1 never switches on the power supply to the inserted card by itself. The controlling computer through the proper command sent to the reader must explicitly do this.

## 6.0. Software Design

### 6.1. Bluetooth Connection Program Flow

The program flow of a Bluetooth connection is shown below:



**Figure 2:** Bluetooth Connection Flow

## 6.2. Profile Selection

ACR3901T-W1 is a smart card reader that is designed to use Bluetooth technology as an interface to transmit data. A customized service called Commands Communication with three pipes is used: one pipe is used for command request, second pipe is for command response, and the third pipe is used to notify the paired device about the card and sleep mode status.

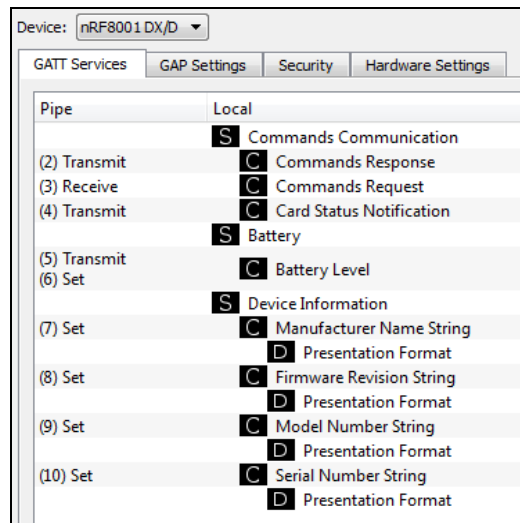
Also, the current reader's battery status is significant when the reader is operating in Bluetooth mode, hence, a customized battery service is used to notify the paired device about the current battery status. When there is a change in the battery status, the reader will notify the paired device through a specific pipe. To simplify, the battery levels are divided into three groups, below is a table summarizing the battery level and its corresponding return value:

Status	Voltage	Return Value
Sufficient battery	≥ 3.3 V	FEh
Low battery	<3.3 V and ≥ 2.9 V	Value other than FFh/FEh/00h
No battery	<2.9 V	00h
USB mode		FFh

In Card Status Notification service, it will notify the paired device on any changes on the card status or when the reader enters sleep mode. Below is a list of the status and the corresponding return value:

Status	Return Value
No card present	50 02h
Card present	50 03h
Reader has entered sleep mode	50 04h

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



**Figure 3:** nRFGo Studio GATT Setting Interface





nRFgo-Studio Configuration adds one service, and there will be totally 10 services:

```
#define PIPE_GAP_DEVICE_NAME_SET 1
#define PIPE_COMMANDS_COMMUNICATION_COMMANDS_RESPONSE_TX 2
#define PIPE_COMMANDS_COMMUNICATION_COMMANDS_REQUEST_RX 3
#define PIPE_COMMANDS_COMMUNICATION_CARD_STATUS_NOTIFICATION_TX 4
#define PIPE_BATTERY_BATTERY_LEVEL_TX 5
#define PIPE_BATTERY_BATTERY_LEVEL_SET 6
#define PIPE_DEVICE_INFORMATION_MANUFACTURER_NAME_STRING_SET 7
#define PIPE_DEVICE_INFORMATION_FIRMWARE_REVISION_STRING_SET 8
#define PIPE_DEVICE_INFORMATION_MODEL_NUMBER_STRING_SET 9
#define PIPE_DEVICE_INFORMATION_SERIAL_NUMBER_STRING_SET 10
#define NUMBER_OF_PIPES 10
```

*#define PIPE\_GAP\_DEVICE\_NAME\_SET* is used to change the device name at runtime by the application controller. So that in Bluetooth mode, the advertising name will be in the format of “ACR3901T-W1XXXXXXX”, where “XXXXXXX” is the last 7 bytes of reader’s serial number.

In order to make the advertising name be “ACR3901T-W1XXXXXXX”, Bluetooth Mode Start operation should be implemented first.

**Bluetooth Mode Start:**

1. Setup (06h) uploads the configuration to Bluetooth module.
2. Use pipe 1 to set the device name in the format of “ACR3901T-W1XXXXXXX” (PIPE\_GAP\_DEVICE\_NAME\_SET)
3. Connect (0Fh).
4. Advertising.

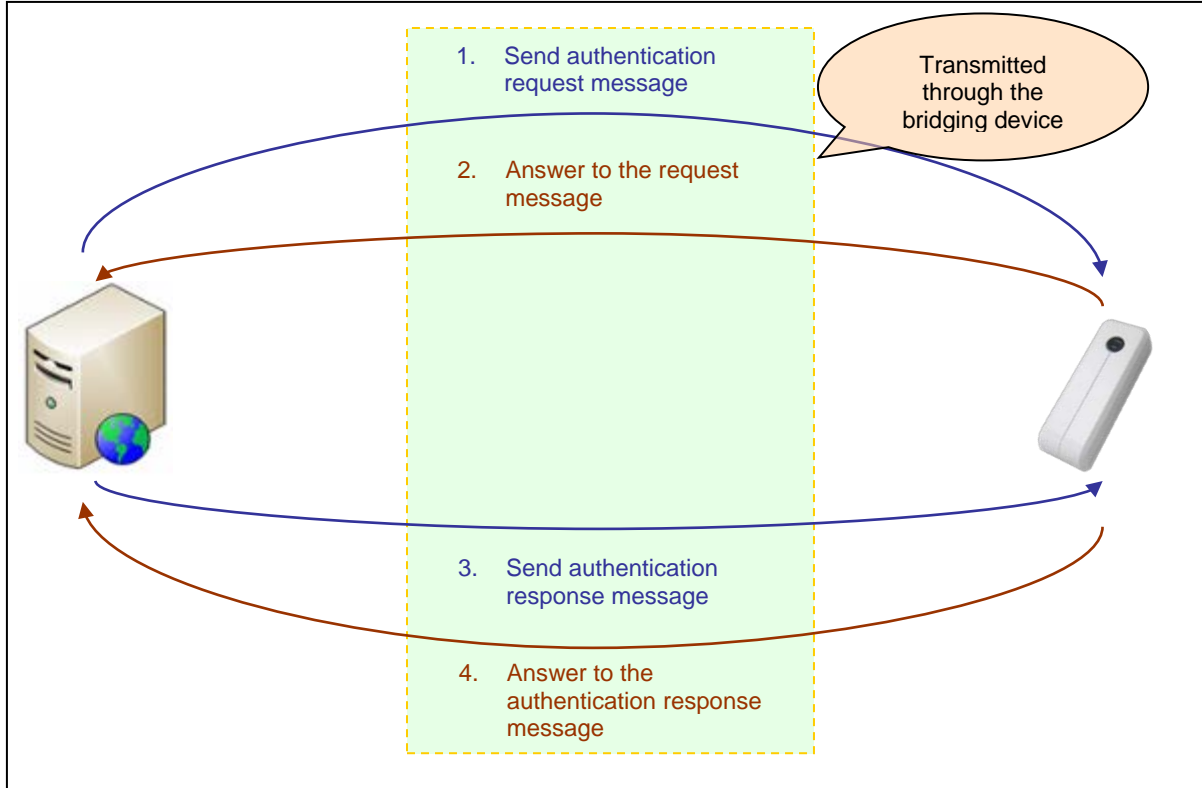
Attribute Name	UUID	Handle
DeviceName	2A00	03h
Send (Reader → Paired device)	8002	0Bh
Receive (Paired device → Reader)	8003	0Eh
CardStatus	8004	10h
BatteryLevel	2A19	14h
Manufacturer	2A29	18h
FW_Version	2A26	1Bh
ModelNumber	2A24	1Eh
SerialNumber	2A25	21h

**Table 5:** ACR3901T-W1 Service Handles and UUID Information List

### 6.3. Authentication

Before any sensitive data can be loaded into ACR3901T-W1, the data processing server must be authenticated by ACR3901T-W1 for the privilege to modify the secured data inside reader. In ACR3901T-W1, a mutual authentication method is being used.

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



**Figure 4:** Authentication Procedure

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

## 6.4. Frame Format

### 6.4.1. Bluetooth Frame Format

HID Frame	Length (Bytes)	Description
Identifiers	1	Commands
Length	2	Length {Payload+Checksum}
Payload	0-N	Data
Checksum	1	XOR {Identifiers,Length,Payload}

**Table 6:** Bluetooth Frame Format

The frame format should be:

Identifier + LEN1 + LEN2 + N-bytes Payload + Checksum

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

Data checksum is used in detecting errors that may have been introduced during wireless data transmission. To calculate the data checksum: XOR {Identifiers,Length,Payload}.

**Example:** 62010063 => Checksum = 63h

### 6.4.2. Bluetooth Frame Format after Mutual Authentication

Mutual authentication was introduced to avoid man-in-the-middle attack through the Bluetooth communication channel. After a successful mutual authentication, the Bluetooth Frame Format in **Table 6** will be encrypted and packed with 1 byte header byte, 2 Len byte, and 1 check byte. The frame format after mutual authentication should look like the structure below:

Header + Len + (Identifiers + Length + Payload + Checksum)\* + Check byte

**Note:** Each 16 bytes of data will be decrypted with the Customer Master Key using the AES-128 CBC cipher mode. The initial vector is 16 bytes (00h) in AES-128 CBC cipher mode.

HID Frame	Length (bytes)	Description	
Header byte	1	Value: 72h / 22h	
Len	2	Length {Identifiers + Length + Payload + Checksum + Check + Stop byte}	
Identifiers	1	Commands	Encrypted data of the Bluetooth frame format; The final data length of this part is 16*N bytes (N>0)
Length	2	Length {Payload + Checksum}	
Payload	0-N	Data	
Checksum	1	XOR {Identifiers, Length, Payload}	
Check byte	1	XOR {Header, Len, Encrypted(Identifiers, Length, Payload, Checksum)}	

**Table 7:** Encrypted Frame Format after Mutual Authentication



## 6.5. Bluetooth Communication Protocol

ACR3901T-W1 communicates to the paired device using the Bluetooth interface with a predefined protocol. The protocol is similar to the formats of the CCID Command Pipe and Response Pipe.

Command	Mode supported	Sender	Description
62h	Authenticated	Paired device	ICC Power On
63h	Authenticated	Paired device	ICC Power Off
65h	Authenticated	Paired device	Get Card Presence
6Fh	Authenticated	Paired device	Exchange APDU
61h	Authenticated	Paired device	Set Parameters
6Bh	Authenticated	Paired device	Peripheral Commands
70h	Connected/Authenticated	Paired device	SPH_to_RDR_ReqAuth*
71h	Connected/Authenticated	Paired device	SPH_to_RDR_AuthRsp*

**Table 8:** Command Code Summary

Command	Mode Supported	Sender	Description
12h	Authenticated	Reader	Response to ICC Power On
13h	Authenticated	Reader	Response to ICC Power Off
14h	Authenticated	Reader	Response to Get Card Presence
11h	Authenticated	Reader	Response to Exchange APDU
...	...	...	...
16h	Authenticated	Reader	Response to Set Parameters
15h	Authenticated	Reader	Response to Peripheral Commands
20h	Connected/Authenticated	Reader	RDR_to_SPH_AuthRsp1*
21h	Connected/Authenticated	Reader	RDR_to_SPH_AuthRsp2*

**Table 9:** Response Code Summary

**\*Note:** These command/response codes are the communication codes being used in Mutual Authentication.



### 6.5.1. Card Power On

This command sends a power on request to the reader.

#### Command Format

Offset	Field	Size	Value	Description
0	<i>bMessageType</i>	1	62h	-
1	<i>LEN1 LEN2 (wLength)</i>	2	0100h	Number of extra bytes starting from the next field for this message, and is expressed in two bytes, and LEN1 is LSB while LEN2 is MSB
3	<i>CSUM (wChecksum)</i>	1	63h	CSUM means the XOR values of all bytes in the command

#### Response Data Format

Offset	Field	Size	Value	Description
0	<i>bMessageType</i>	1	12h	-
1	<i>LEN1 LEN2 (wLength)</i>	2	0100h	Number of extra bytes starting from the next field for this message, and is expressed in two bytes long, and LEN1 is LSB while LEN2 is MS
3	<i>N byte ATR</i>	N		Card Answer-To-Reset
3+N	<i>CSUM (wChecksum)</i>	1		CSUM means the XOR values of all bytes in the command

#### Example:

Request = 62 01 00 63

Response = 12 14 00 3B BE 11 00 00 41 01 38 00 00 00 00 12 34 56 78 01  
90 00 73

ATR = 3B BE 11 00 00 41 01 38 00 00 00 00 12 34 56 78 01



### 6.5.2. Card Power Off

This command sends a power off request to the reader.

#### Command Format

Offset	Field	Size	Value	Description
0	<i>bMessageType</i>	1	63h	-
1	<i>LEN1 LEN2 (wLength)</i>	2	0100h	Number of extra bytes starting from the next field for this message, and is expressed in two bytes long, and LEN1 is LSB while LEN2 is MSB
3	<i>CSUM (wChecksum)</i>	1	62h	CSUM means the XOR values of all bytes in the command

#### Response Data Format

Offset	Field	Size	Value	Description
0	<i>bMessageType</i>	1	13h	-
1	<i>LEN1 LEN2 (wLength)</i>	2	0100h	Number of extra bytes starting from the next field for this message, and is expressed in two bytes long, and LEN1 is LSB while LEN2 is MSB
3	<i>CSUM (wChecksum)</i>	1	12h	CSUM means the XOR values of all bytes in the command

#### Example:

Request = 62 01 00 62

Response = 13 01 00 12



### 6.5.3. Get Card Presence

This checks the presence of the inserted card.

#### Command Format

Offset	Field	Size	Value	Description
0	<i>bMessageType</i>	1	65h	-
1	<i>LEN1 LEN2 (wLength)</i>	2	0100h	Number of extra bytes starting from the next field for this message, and is expressed in two bytes long, and LEN1 is LSB while LEN2 is MSB
3	<i>CSUM (wChecksum)</i>	1	64h	CSUM means the XOR values of all bytes in the command

#### Response Data Format

Offset	Field	Size	Value	Description
0	<i>bMessageType</i>	1	14h	-
1	<i>LEN1 LEN2 (wLength)</i>	2	0200h	Number of extra bytes starting from the next field for this message, and is expressed in two bytes long, and LEN1 is LSB while LEN2 is MSB
3	<i>STA</i>	1	-	Card Status: 00 = Unknown status 01 = No card present 02 = Card present but inactive 03 = Card present and active
4	<i>CSUM (wChecksum)</i>	1	-	CSUM means the XOR values of all bytes in the command

#### Example:

Request = 65 01 00 64

Response = 14 02 00 03 15



### 6.5.4. APDU Command

This command sends an APDU command to the reader.

#### Command Format

Offset	Field	Size	Value	Description
0	<i>bMessageType</i>	1	<b>6Fh</b>	-
1	<i>LEN1 LEN2 (wLength)</i>	2	-	Number of extra bytes starting from the next field for this message, and is expressed in two bytes long, and LEN1 is LSB while LEN2 is MSB
3	<i>APDU CMD</i>	N	-	APDU Command
3+N	<i>CSUM (wChecksum)</i>	1	-	CSUM means the XOR values of all bytes in the command

#### Response Data Format

Offset	Field	Size	Value	Description
0	<i>bMessageType</i>	1	<b>11h</b>	-
1	<i>LEN1 LEN2 (wLength)</i>	2	-	Number of extra bytes starting from the next field for this message, and is expressed in two bytes long, and LEN1 is LSB while LEN2 is MSB
3	<i>APDU Response</i>	N	-	APDU Format Data
3+N	<i>CSUM (wChecksum)</i>	1	-	CSUM means the XOR values of all bytes in the command

#### Example:

Request = **6F** 06 00 80 84 00 00 08 65

Response = **11** 0B 00 C1 7A 3B AA D6 5A FA CE 90 00 18





### 6.5.5. Escape Command

This command gives access the extended features of the reader.

#### Command Format

Offset	Field	Size	Value	Description	
0	<i>bMessageType</i>	1	<b>6Bh</b>	Escape CMD Header	
1	<i>LEN1 LEN2 (wLength)</i>	2	-	Number of extra bytes starting from the next field for this message, and is expressed in two bytes long, and LEN1 is LSB while LEN2 is MSB	
3	<i>abData1</i>	<i>CommandCode</i>	1	-	Command Header
4		<i>Len (CommandLength)</i>	1	-	Number of extra bytes starting from the next field for this message, and is expressed in one byte long
5		<i>Data</i>	N	-	0 =< N <= 255
5+N	<i>CSUM (wChecksum)</i>	1	-	CSUM means the XOR values of all bytes in the command	

#### Response Data Format

Offset	Field	Size	Value	Description	
0	<i>bMessageType</i>	1	<b>15h</b>	Escape Response Header	
1	<i>LEN1 LEN2 (wLength)</i>	2	-	Number of extra bytes starting from the next field for this message, and is expressed in two bytes long, and LEN1 is LSB while LEN2 is MSB	
3	<i>abData2</i>	<i>ResponseCode</i>	1	-	Response Header
4		<i>Len (CommandLength)</i>	1	-	Number of extra bytes starting from the next field for this message, and is expressed in one byte long
5		<i>Data</i>	N	-	0 =< N <= 255
5+N	<i>CSUM (wChecksum)</i>	1	-	CSUM means the XOR values of all bytes in the command	



### 6.5.6. Card Set Parameters

This command changes the parameters for the inserted card after power on.

#### Command Format

Offset	Field		Size	Value	Description
0	<i>bMessageType</i>		1	<b>61h</b>	-
1	<i>LEN1 LEN2 (wLength)</i>		2	-	Number of extra bytes starting from the next field for this message, and is expressed in two bytes long, and LEN1 is LSB while LEN2 is MSB
3	<i>abData1</i>	<i>ProtocolNum</i>	1	-	Card Protocol data structure: 00h = Structure for protocol T=0 01h = Structure for protocol T=1
4		<i>ProtocolDataStructure</i>	N	-	Protocol Data Structure
4+N	<i>CSUM (wChecksum)</i>		1	-	CSUM means the XOR values of all bytes in the command

#### Response Data Format

Offset	Field		Size	Value	Description
0	<i>bMessageType</i>		1	<b>16h</b>	Escape Response Header
1	<i>LEN1 LEN2 (wLength)</i>		2	-	Number of extra bytes starting from the next field for this message, and is expressed in two bytes long, and LEN1 is LSB while LEN2 is MSB
3	<i>abData2</i>	<i>ProtocolNum</i>	1	-	Card Protocol data structure: 00h = Structure for protocol T=0 01h = Structure for protocol T=1
4		<i>ProtocolDataStructure</i>	N	-	Protocol Data Structure
4+N	<i>CSUM (wChecksum)</i>		1	-	CSUM means the XOR values of all bytes in the command



Protocol Data Structure for Protocol T=0 (ProtocolNum = 0, wLength = 0700h)

Offset	Field	Size	Value	Description
4	<i>bmFindexDindex</i>	1	-	B7-4 – FI – Index into the table 7 in ISO/IEC 7816-3:1997 selecting a clock rate conversion factor. B3-0 – DI – Index into the table 8 in ISO/IEC 7816-3:1997 selecting a baud rate conversion factor.
5	<i>bmTCCKST0</i>	1	-	B0 – 0b, B7-2 – 000000b B1 – Convention used (b1=0 for direct, b1=1 for inverse) <b>Note:</b> The CCID ignores this bit.
6	<i>bGuardTimeT0</i>	1	-	Extra Guardtime between two characters. Add 0 to 254 etu to the normal guardtime of 12 etu. FFh is the same as 00h.
7	<i>bWaitingIntegerT0</i>	1	-	WI for T=0 used to define WWT
8	<i>bClockStop</i>	1	-	ICC Clock Stop Support 00h = Stopping the Clock is not allowed 01h = Stop with Clock signal Low 02h = Stop with Clock signal High 03h = Stop with Clock either High or Low

Protocol Data Structure for Protocol T=1 (ProtocolNum = 1, wLength = 0900h)

Offset	Field	Size	Value	Description
4	<i>bmFindexDindex</i>	1	-	B7-4 – FI – Index into the table 7 in ISO/IEC 7816-3:1997 selecting a clock rate conversion factor. B3-0 – DI – Index into the table 8 in ISO/IEC 7816-3:1997 selecting a baud rate conversion factor.
5	<i>BmTCCKST1</i>	1	-	B7-2 – 000100b B0 – Checksum type (b0=0 for LRC, b0=1 for CRC) B1 – Convention used (b1=0 for direct, b1=1 for inverse) <b>Note:</b> The CCID ignores this bit.
6	<i>BGuardTimeT1</i>	1	-	Extra Guardtime (0 to 254 etu between two characters). If value is FFh, then guardtime is reduced by 1 etu.
7	<i>BwaitingIntegerT1</i>	1	-	B7-4 = BWI values 0-9 valid B3-0 = CWI values 0-Fh valid



Offset	Field	Size	Value	Description
8	<i>bClockStop</i>	1	-	ICC Clock Stop Support 00h = Stopping the Clock is not allowed 01h = Stop with Clock signal Low 02h = Stop with Clock signal High 03h = Stop with Clock either High or Low
9	<i>bIFSC</i>	1	-	Size of negotiated IFSC
10	<i>bNadValue</i>	1	00h	Only support NAD = 00h

**Example: (T0 protocol)**

Request = 61 07 00 00 11 00 00 0A 00 7D

Response = 16 07 00 00 11 00 00 0A 00 0A

**Example: (T1 protocol)**

Request = 61 09 00 01 96 10 00 45 00 FE 00 54

Response = 16 09 00 01 96 10 00 45 00 FE 00 23



## 6.6. Mutual Authentication and Encryption Protocol

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

Command	Mode supported	Sender	Description
70h	Connected	Paired device	SPH_to_RDR_ReqAuth
71h	Connected	Paired device	SPH_to_RDR_AuthRsp
72h	Authenticated	Paired device	SPH_to_RDR_DataReq
20h	Connected	Reader	RDR_to_SPH_AuthRsp1
21h	Connected	Reader	RDR_to_SPH_AuthRsp2
22h	Authenticated	Reader	RDR_to_SPH_DataRsp

**Table 10:** Summary of Mutual Authentication Commands

### 6.6.1. SPH\_to\_RDR\_ReqAuth

This command will request ACR3901T-W1 to perform authentication with the paired key-generating device. After a successful authentication, the Customer Master Key can be modified by the paired key-generating device.

For more information on the authentication process, please refer to [Authentication](#).

Offset	Field	Size	Value	Description	Encrypted
0	<i>bMessageType</i>	1	70h	-	No
1	<i>LEN1 LEN2 (wLength)</i>	2	0100h	Number of extra bytes starting from the next field for this message, and is expressed in two bytes long, and LEN1 is LSB while LEN2 is MSB	
3	<i>wChecksum</i>	1	71h	CSUM means the XOR values of all bytes in the command	

The response to this message is *RDR\_to\_SPH\_AuthRsp1* if the received command message is error free.



### 6.6.2. RDR\_to\_SPH\_AuthRsp1

This command is sent by the paired device in response to the *SPH\_to\_RDR\_ReqAuth*.

For more information, please refer to [Authentication](#).

Offset	Field	Size	Value	Description	Encrypted
0	<i>bMessageType</i>	1	20h	-	No
1	<i>LEN1 LEN2</i> ( <i>wLength</i> )	2	1100h	Number of extra bytes starting from the next field for this message, and is expressed in two bytes long, and LEN1 is LSB while LEN2 is MSB	No
4	<i>abRndNum</i>	16	-	<i>abRndNum</i> [0:15] – 16 bytes of random number. All the 16-byte data must be encrypted with the Customer Master Key currently stored in ACR3901T-W1.	Yes
20	<i>wChecksum</i>	1	-	CSUM means the XOR values of all bytes in the command	No



### 6.6.3. SPH\_to\_RDR\_AuthRsp

This command is the second phase of the authentication process. After the device has initiated the *SPH\_to\_RDR\_ReqAuth* command to the ACR3901T-W1, the reader will then provide an *RDR\_to\_SPH\_AuthRsp1* message if there's no error.

The *RDR\_to\_SPH\_AuthRsp1* will contain a sequence of 16-byte random numbers encrypted using the Customer Master Key. The paired key-generating device should decrypt it using the correct Customer Master Key and pads it to the end of the 16-byte of random numbers. The overall 32-byte random numbers will be decrypted using the Customer Master Key and return it to the ACR3901T-W1 using this command in order to have a successful authentication.

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

Offset	Field	Size	Value	Description	Encrypted
0	<i>bMessageType</i>	1	71h	-	No
1	<i>LEN1 LEN2 (wLength)</i>	2	2100h	Number of extra bytes starting from the next field for this message, and is expressed in two bytes long, and LEN1 is LSB while LEN2 is MSB	No
3	<i>abAuthData</i>	32	-	<i>abAuthData</i> [0:15] – 16 bytes of random number generated by the data processing server. <i>abAuthData</i> [16:31] – 16 bytes of decrypted random number received from ACR3901T-W1. All the 32 bytes of data will undergo a decryption process with the Customer Master Key using AES128 CBC cipher mode	Yes
35	<i>wChecksum</i>	1	-	CSUM means the XOR values of all bytes in the command	No

The response to this message is *RDR\_to\_SPH\_AuthRsp2* if the command message received is error free and the random number generated returned by the ACR3901T-W1 is correct.



#### 6.6.4. RDR\_to\_SPH\_AuthRsp2

This command is sent by the paired device in response to the *SPH\_to\_RDR\_AuthRsp*.

For more information, please refer to [Authentication](#).

Offset	Field	Size	Value	Description	Encrypted
0	<i>bMessageType</i>	1	21h	-	No
1	<i>LEN1 LEN2 (wLength)</i>	2	1100h	Number of extra bytes starting from the next field for this message, and is expressed in two bytes long, and LEN1 is LSB while LEN2 is MSB	No
4	<i>abRndNum</i>	16		<i>abRndNum</i> [0:15] – 16 bytes of random number retrieved from the data processing server. All the 16-byte data must be encrypted with the Customer Master Key that is currently stored in ACR3901T-W1.	Yes
20	<i>wChecksum</i>	1		CSUM means the XOR values of all bytes in the command	No

#### 6.6.5. SPH\_to\_RDR\_DataReq

This command is sent from the paired device to the ACR3901T-W1 after the mutual authentication process.

In Bluetooth mode, the communication protocol from [Card Power On](#) to [Card Set Parameters](#) will be encrypted and transmitted after a successful mutual authentication.

Offset	Field	Size	Value	Description	Encrypted
0	<i>bMessageType</i>	1	72h	-	No
1	<i>LEN1 LEN2 (wLength)</i>	2	-	Number of extra bytes starting from the next field for this message, and is expressed in two bytes long, and LEN1 is LSB while LEN2 is MSB	No
3	<i>abEncryptedData</i>	N*16	-	Each 16 bytes of data will undergo a decryption process with the Customer Master Key using AES128 CBC cipher mode	Yes
35	<i>wChecksum</i>	1	-	CSUM means the XOR values of all bytes in the command	No

*abEncryptedData* is N\*16 bytes long. This is the encrypted data of (Identifiers + Length + Payload + Checksum), wherein each byte will undergo a decryption process with the Customer Master Key using the AES128 CBC cipher mode.

The initial vector is 16bytes of 00h in AES-128 CBC cipher mode.





For original data with data length < N\*16, simply pad FFh in the end and make it a 16\*N byte long before encrypting.

HID Frame	Length (bytes)	Description	
Identifiers	1	Commands	The real data is decrypted using abEncryptedData and remove the dummy data
Length	2	Length {Payload+Checksum}	
Payload	0-N	Data	
Checksum	1	XOR {Identifiers,Length,Payload}	

**Example:**

After a successful Mutual Authentication, paired device sends a power on command to the reader, the command will be:

72 11 00 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX

**Where:**

Command header: 72

Encrypted data of the power on command (16 bytes): XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX

The response to this message is the RDR\_to\_SPH\_DataRsp if the command message received is error free.

abData is the encrypted data of the Communication Protocol. Each 16 bytes of data will undergo a decryption process with the Customer Master Key using the AES-128 CBC cipher mode.

**6.6.6. RDR\_to\_SPH\_DataRsp**

This command is sent from the reader to the paired device after a successful mutual authentication.

In Bluetooth mode, the communication protocol from **Card Power On** to **Card Set Parameters** will be encrypted and transmitted after a successful mutual authentication.

Offset	Field	Size	Value	Description	Encrypted
0	<i>bMessageType</i>	1	<b>22h</b>	-	No
1	<i>LEN1 LEN2 (wLength)</i>	2	-	The number of extra bytes starting from the next field for this message, and is expressed in two bytes long, and LEN1 is LSB while LEN2 is MSB	No
3	<i>abEncryptedData</i>	N*16	-	Each 16 bytes of data will undergo a decryption process with the Customer Master Key using AES128 CBC cipher mode	Yes
35	<i>wChecksum</i>	1	-	CSUM means the XOR values of all bytes in the command	No



## 7.0. USB Communication Protocol

ACR3901T-W1 shall interface with the host through the USB connection. A specification, namely CCID, has been released within the industry defining such a protocol for the USB chip-card interface devices. CCID covers all the protocols required for operating smart cards.

The configurations and usage of USB endpoints on ACR3901T-W1 shall follow CCID Rev 1.0 Section 3.

An overview is summarized below:

1. *Control Commands* are sent on control pipe (default pipe). These include class-specific requests and USB standard requests. Commands that are sent on the default pipe report information back to the host on the default pipe.
2. *CCID Events* are sent on the interrupt pipe.
3. *CCID Commands* are sent on BULK-OUT endpoint. Each command sent to ACR3901T-W1 has an associated ending response. Some commands can also have intermediate responses.
4. *CCID Responses* are sent on BULK-IN endpoint. All commands sent to ACR3901T-W1 have to be sent synchronously (e.g., *bMaxCCIDBusySlots* is equal to 01h for ACR3901T-W1).

The ACR3901T-W1 supported CCID features are indicated in its Class Descriptor:

Offset	Field	Size	Value	Description
0	<i>bLength</i>	1	-	Size of this descriptor, in bytes
1	<i>bDescriptorType</i>	1	-	CCID Functional Descriptor type
2	<i>bcdCCID</i>	2	-	CCID Specification Release Number in Binary-coded decimal
4	<i>bMaxSlotIndex</i>	1	-	One slot is available on ACR3901T-W1
5	<i>bVoltageSupport</i>	1	-	ACR3901T-W1 can supply 1.8 V, 3 V, and 5 V to its slot
6	<i>dwProtocols</i>	4	-	ACR3901T-W1 supports T=0 and T=1 protocol
10	<i>dwDefaultClock</i>	4	-	Default ICC clock frequency is 4.8 MHz
14	<i>dwMaximumClock</i>	4	-	Maximum supported ICC clock frequency is 4.8 MHz
18	<i>bNumClockSupported</i>	1	-	Does not support manual setting of clock frequency
19	<i>dwDataRate</i>	4	-	Default ICC I/O data rate is 12903 bps
23	<i>dwMaxDataRate</i>	4	-	Maximum supported ICC I/O data rate is 600 Kbps
27	<i>bNumDataRatesSupported</i>	1	-	Does not support manual setting of data rates
28	<i>dwMaxIFSD</i>	4	-	Maximum IFSD supported by ACR3901T-W1 for protocol T=1 is 254
32	<i>dwSynchProtocols</i>	4	-	ACR3901T-W1 does not support synchronous card
36	<i>dwMechanical</i>	4	-	ACR3901T-W1 does not support special mechanical characteristics



Offset	Field	Size	Value	Description
40	<i>dwFeatures</i>	4	-	ACR3901T-W1 supports the following features: <ul style="list-style-type: none"><li>• Automatic ICC clock frequency change according to parameters</li><li>• Automatic baud rate change according to frequency and FI,DI parameters</li><li>• TPDU level change with ACR3901T-W1</li></ul>
44	<i>dwMaxCCIDMessageLength</i>	4	-	Maximum message length accepted by ACR3901T-W1 is 271 bytes
48	<i>bClassGetResponse</i>	1	-	Insignificant for TPDU level exchanges
49	<i>bClassEnvelope</i>	1	-	Insignificant for TPDU level exchanges
50	<i>wLCDLayout</i>	2	-	No LCD
52	<i>bPINSupport</i>	1	-	With PIN Verification
53	<i>bMaxCCIDBusySlots</i>	1	-	Only 1 slot can be simultaneously busy



## 7.1. CCID Bulk-OUT Messages

### 7.1.1. PC\_to\_RDR\_IccPowerOn

This command activates the card slot and returns ATR from the card.

Offset	Field	Size	Value	Description
0	<i>bMessageType</i>	1	<b>62h</b>	-
1	<i>dwLength</i>	4	00000000h	Size of extra bytes of this message
2	<i>bSlot</i>	1	-	Identifies the slot number for this command
5	<i>bSeq</i>	1	-	Sequence number for command
6	<i>bPowerSelect</i>	1	-	Voltage that is applied to the ICC: 00h = Automatic Voltage Selection 01h = 5 V 02h = 3 V
7	<i>abRFU</i>	2	-	Reserved for future use

The response to this command message is *RDR\_to\_PC\_DataBlock* response message and the data returned is the Answer-to-Reset (ATR) data.

### 7.1.2. PC\_to\_RDR\_IccPowerOff

This command deactivates the card slot.

Offset	Field	Size	Value	Description
0	<i>bMessageType</i>	1	<b>63h</b>	-
1	<i>dwLength</i>	4	00000000h	Size of extra bytes of this message
5	<i>bSlot</i>	1	-	Identifies the slot number for this command
6	<i>bSeq</i>	1	-	Sequence number for command
7	<i>abRFU</i>	3	-	Reserved for future use

The response to this message is the *RDR\_to\_PC\_SlotStatus* message.

### 7.1.3. PC\_to\_RDR\_GetSlotStatus

This command gets the current status of the slot.

Offset	Field	Size	Value	Description
0	<i>bMessageType</i>	1	<b>65h</b>	-
1	<i>dwLength</i>	4	00000000h	Size of extra bytes of this message
5	<i>bSlot</i>	1	-	Identifies the slot number for this command
6	<i>bSeq</i>	1	-	Sequence number for command
7	<i>abRFU</i>	3	-	Reserved for future use

The response to this message is the *RDR\_to\_PC\_SlotStatus* message.



#### 7.1.4. PC\_to\_RDR\_XfrBlock

This command transfers data block to the ICC.

Offset	Field	Size	Value	Description
0	<i>bMessageType</i>	1	<b>6Fh</b>	-
1	<i>dwLength</i>	4	-	Size of <i>abData</i> field of this message.
5	<i>bSlot</i>	1	-	Identifies the slot number for this command.
6	<i>bSeq</i>	1	-	Sequence number for command.
7	<i>bBWI</i>	1	-	Used to extend the CCIDs Block Waiting Timeout for this current transfer. The CCID will timeout the block after “this number multiplied by the Block Waiting Time” has expired.
8	<i>wLevelParameter</i>	2	0000h	RFU (TPDU exchange level).
10	<i>abData</i>	Byte array	-	Data block sent to the CCID. Data is sent “as is” to the ICC (TPDU exchange level).

The response to this message is the *RDR\_to\_PC\_DataBlock* message.

#### 7.1.5. PC\_to\_RDR\_GetParameters

This command gets the slot parameters.

Offset	Field	Size	Value	Description
0	<i>bMessageType</i>	1	<b>6Ch</b>	-
1	<i>DwLength</i>	4	00000000h	Size of extra bytes of this message
5	<i>BSlot</i>	1	-	Identifies the slot number for this command
6	<i>BSeq</i>	1	-	Sequence number for command
7	<i>AbRFU</i>	3	-	Reserved for future use

The response to this message is the *RDR\_to\_PC\_Parameters* message.

#### 7.1.6. PC\_to\_RDR\_ResetParameters

This command resets slot parameters to its default value.

Offset	Field	Size	Value	Description
0	<i>bMessageType</i>	1	<b>6Dh</b>	-
1	<i>DwLength</i>	4	00000000h	Size of extra bytes of this message
5	<i>BSlot</i>	1	-	Identifies the slot number for this command
6	<i>BSeq</i>	1	-	Sequence number for command
7	<i>AbRFU</i>	3	-	Reserved for future use

The response to this message is the *RDR\_to\_PC\_Parameters* message.



### 7.1.7. PC\_to\_RDR\_SetParameters

This command sets slot parameters.

Offset	Field	Size	Value	Description
0	<i>bMessageType</i>	1	<b>61h</b>	-
1	<i>dwLength</i>	4	-	Size of extra bytes of this message
5	<i>bSlot</i>	1	-	Identifies the slot number for this command
6	<i>bSeq</i>	1	-	Sequence number for command
7	<i>bProtocolNum</i>	1	-	Specifies what protocol data structure follows: 00h = Structure for protocol T=0 01h = Structure for protocol T=1  The following values are reserved for future use: 80h = Structure for 2-wire protocol 81h = Structure for 3-wire protocol 82h = Structure for I2C protocol
8	<i>abRFU</i>	2	-	Reserved for future use
10	<i>abProtocolDataStructure</i>	Byte array	-	Protocol Data Structure

Protocol Data Structure for Protocol T=0 (*dwLength*=00000005h)

Offset	Field	Size	Value	Description
10	<i>bmFindexDindex</i>	1	-	B7-4 – FI – Index into the table 7 in ISO/IEC 7816-3:1997 selecting a clock rate conversion factor. B3-0 – DI – Index into the table 8 in ISO/IEC 7816-3:1997 selecting a baud rate conversion factor.
11	<i>bmTCKKST0</i>	1	-	B0 – 0b, B7-2 – 000000b B1 – Convention used (b1=0 for direct, b1=1 for inverse) <b>Note:</b> The CCID ignores this bit.
12	<i>bGuardTimeT0</i>	1	-	Extra Guardtime between two characters. Add 0 to 254 etu to the normal guardtime of 12 etu. FFh is the same as 00h.
13	<i>bWaitingIntegerT0</i>	1	-	WI for T=0 used to define WWT
14	<i>bClockStop</i>	1	-	ICC Clock Stop Support 00h = Stopping the Clock is not allowed 01h = Stop with Clock signal Low 02h = Stop with Clock signal High 03h = Stop with Clock either High or Low



Protocol Data Structure for Protocol T=1 (dwLength=00000007h)

Offset	Field	Size	Value	Description
10	<i>bmFindexDindex</i>	1	-	B7-4 – FI – Index into the table 7 in ISO/IEC 7816-3:1997 selecting a clock rate conversion factor. B3-0 – DI – Index into the table 8 in ISO/IEC 7816-3:1997 selecting a baud rate conversion factor.
11	<i>BmTCKST1</i>	1	-	B7-2 – 000100b B0 – Checksum type (b0=0 for LRC, b0=1 for CRC) B1 – Convention used (b1=0 for direct, b1=1 for inverse) <b>Note:</b> The CCID ignores this bit.
12	<i>BGuardTimeT1</i>	1	-	Extra Guardtime (0 to 254 etu between two characters). If value is FFh, then guardtime is reduced by 1 etu.
13	<i>BwaitingIntegerT1</i>	1	-	B7-4 = BWI values 0-9 valid B3-0 = CWI values 0-Fh valid
14	<i>bClockStop</i>	1	-	ICC Clock Stop Support 00h = Stopping the Clock is not allowed 01h = Stop with Clock signal Low 02h = Stop with Clock signal High 03h = Stop with Clock either High or Low
15	<i>bIFSC</i>	1	-	Size of negotiated IFSC
16	<i>bNadValue</i>	1	00h	Only support NAD = 00h

The response to this message is the *RDR\_to\_PC\_Parameters* message.

### 7.1.8. PC\_to\_RDR\_Escape

This command accesses extended features.

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

The response to this message is the *RDR\_to\_PC\_Escape* message.



## 7.2. CCID Bulk-IN Messages

### 7.2.1. RDR\_to\_PC\_DataBlock

This message is sent by ACR3901T-W1 in response to *PC\_to\_RDR\_IccPowerOn*, and *PC\_to\_RDR\_XfrBlock* messages.

Offset	Field	Size	Value	Description
0	<i>bMessageType</i>	1	<b>80h</b>	Indicates that a data block is being sent from the CCID
1	<i>dwLength</i>	4	-	Size of extra bytes of this message
5	<i>bSlot</i>	1	-	Same value as in Bulk-OUT message
6	<i>bSeq</i>	1	-	Same value as in Bulk-OUT message
7	<i>bStatus</i>	1	-	Slot status register as defined in CCID Rev 1.0 Section 4.2.1
8	<i>bError</i>	1	-	Slot error register as defined in <b>Error Codes</b> and in CCID Rev 1.0 Section 4.2.1
9	<i>bChainParameter</i>	1	00h	RFU (TPDU exchange level)
10	<i>abData</i>	Byte array	-	This field contains the data returned by the CCID

### 7.2.2. RDR\_to\_PC\_SlotStatus

This message is sent by ACR3901T-W1 in response to *PC\_to\_RDR\_IccPowerOff*, and *PC\_to\_RDR\_GetSlotStatus* messages.

Offset	Field	Size	Value	Description
0	<i>bMessageType</i>	1	<b>81h</b>	-
1	<i>dwLength</i>	4	00000000h	Size of extra bytes of this message
5	<i>bSlot</i>	1	-	Same value as in Bulk-OUT message
6	<i>bSeq</i>	1	-	Same value as in Bulk-OUT message
7	<i>bStatus</i>	1	-	Slot status register as defined in CCID Rev 1.0 Section 4.2.1
8	<i>bError</i>	1	-	Slot error register as defined in <b>Error Codes</b> and in CCID Rev 1.0 Section 4.2.1
9	<i>bClockStatus</i>	1	-	Value: 00h = Clock running 01h = Clock stopped in state L 02h = Clock stopped in state H 03h = Clock stopped in an unknown state All other values are RFU





### 7.2.3. RDR\_to\_PC\_Parameters

This message is sent by ACR3901T-W1 in response to *PC\_to\_RDR\_GetParameters*, *PC\_to\_RDR\_ResetParameters* and *PC\_to\_RDR\_SetParameters* messages.

Offset	Field	Size	Value	Description
0	<i>bMessageType</i>	1	<b>82h</b>	-
1	<i>dwLength</i>	4	-	Size of extra bytes of this message
5	<i>bSlot</i>	1	-	Same value as in Bulk-OUT message
6	<i>bSeq</i>	1	-	Same value as in Bulk-OUT message
7	<i>bStatus</i>	1	-	Slot status register as defined in CCID Rev 1.0 Section 4.2.1
8	<i>bError</i>	1	-	Slot error register as defined in <b>Error Codes</b> and in CCID Rev 1.0 Section 4.2.1
9	<i>bProtocolNum</i>	1	-	Specifies what protocol data structure follows: 00h = Structure for protocol T=0 01h = Structure for protocol T=1 The following values are reserved for future use: 80h = Structure for 2-wire protocol 81h = Structure for 3-wire protocol 82h = Structure for I2C protocol
10	<i>abProtocolDataStructure</i>	Byte array	-	Protocol Data Structure as summarized in CCID Rev 1.0 Section 5.2.3

### 7.2.4. RDR\_to\_PC\_Escape

This message is sent by ACR3901T-W1 in response to *PC\_to\_RDR\_Escape* message.

Offset	Field	Size	Value	Description
0	<i>bMessageType</i>	1	<b>83h</b>	-
1	<i>dwLength</i>	4	-	Size of extra bytes of this message
5	<i>bSlot</i>	1	-	Same value as in Bulk-OUT message
6	<i>bSeq</i>	1	-	Same value as in Bulk-OUT message
7	<i>bStatus</i>	1	-	Slot status register as defined in CCID Rev 1.0 Section 4.2.1
8	<i>bError</i>	1	-	Slot error register as defined in <b>Error Codes</b> and in CCID Rev 1.0 Section 4.2.1
9	<i>bRFU</i>	1	-	Reserved for Future Use
10	<i>abData</i>	Byte array	-	Data sent from CCID



## 8.0. Smart Card Protocol

### 8.1. Peripherals Control

The reader's peripherals control commands are implemented by using Escape Command (0x6B) in Bluetooth mode or PC\_to\_RDR\_Escape in USB mode.

#### 8.1.1. Get Serial Number Command

This command reads the unique serial number of the reader.

Command Format

Offset	Field	Size	Value	Description	
0	<i>abData1</i>	<i>CommandCode</i>	1	<b>02h</b>	Command Code of Write Serial Number
1		<i>Len (CommandLength)</i>	1	00h	Number of extra bytes of data
2		<i>Data</i>	0	-	-

Response Format

Offset	Field	Size	Value	Description	
0	<i>abData2</i>	<i>ResponseCode</i>	1	<b>82h</b>	Response Code of Write Serial Number
1		<i>Len (CommandLength)</i>	1	-	Number of extra bytes of data
2		<i>Data</i>	N	-	Number of bytes of Serial Number

#### Example:

Request = 02 00

Response = 82 0A FF FF FF FF FF FF FF FF FF FF

Serial Number = FF FF FF FF FF FF FF FF FF FF



### 8.1.2. Get Random Number Command

This command reads the random number from the reader that is used to encrypt with the Master Key for authentication by the AES Encryption algorithm. *(Used in Bluetooth mode only)*

#### Command Format

Offset	Field		Size	Value	Description
0	abData1	CommandCode	1	03h	Command Code of Get Random Number
1		Len (CommandLength)	1	00h	Number of extra bytes of data
2		Data	0	-	-

#### Response Format

Offset	Field		Size	Value	Description
0	abData2	ResponseCode	1	83h	Response Code of Get Random Number
1		Len (CommandLength)	1	10h	Number of extra bytes of data
2		Data	16	-	16 bytes of Random Number

#### Example:

Request = 03 00

Response = 83 10 F2 8F B7 EF BA 43 C4 6B 85 D8 51 7B 84 08 C3 25



### 8.1.3. Get Firmware Version Command

This command gets the firmware version of the reader.

#### Command Format

Offset	Field	Size	Value	Description	
0	<i>abData1</i>	<i>CommandCode</i>	1	<b>04h</b>	Command Code of Get Firmware Version
1		<i>Len (CommandLength)</i>	1	00h	Number of extra bytes of data
2		<i>Data</i>	0	-	-

#### Response Format

Offset	Field	Size	Value	Description	
0	<i>abData2</i>	<i>ResponseCode</i>	1	<b>84h</b>	Response Code of Get Firmware Version
1		<i>Len (CommandLength)</i>	1	-	Number of extra bytes of data
1		<i>Data</i>	N	-	Number of bytes of Firmware Version in the format of "Vx.xx"

#### Example:

Request = 04 00

Response = 84 08 56 36 2E 30 30 2E 30 30

Firmware Version (HEX) = 56 36 2E 30 30 2E 30 30

Firmware Version (ASCII) = "V6.00.00"



### 8.1.4. Rewrite Master Key Command

This command rewrites the master key to the reader. It is required to be encrypted by the old key using the AES encryption algorithm.

#### Command Format

Offset		Field	Size	Value	Description
0	abData1	CommandCode	1	07h	Command Code of Rewrite Master Key
1		Len (CommandLength)	1	20h	Number of extra bytes of data
2		Data	32	-	Combine the random number (KeyRstRnd[0:15]) encrypted by original Mater Key + 16 byte of new Master Key encrypted by the original Mater Key

#### Response Format

Offset		Field	Size	Value	Description
0	abData2	ResponseCode	1	87h	Response Code of Rewrite Master Key
1		Len (CommandLength)	1	01h	Number of extra bytes of data
2		Data	1	-	00h = Success 01h = Fail

#### Example:

Refer to [Customer Master Key Reset Request](#) for more details.



### 8.1.5. Sleep Mode Option

This command sets the time interval of the device before it enters sleep mode. By default, the reader will enter to sleep mode if the reader is idle for 60 seconds.

#### Command Format

Offset		Field	Size	Value	Description
0	abData1	CommandCode	1	0Dh	Command Code of Sleep Mode Option
1		Len (CommandLength)	1	01h	Number of extra bytes of data
2		Data	1	-	00h = 60 seconds (Default) 01h = 90 seconds 02h = 120 seconds 03h = 180 seconds 04h = Disable

#### Response Format

Offset		Field	Size	Value	Description
0	abData2	ResponseCode	1	8Dh	Response Code of Sleep Mode Option
1		Len (CommandLength)	1	01h	Number of extra bytes of data
2		Data	1	-	00h = Success 01h = Fail

#### Example:

Request to set 90s = 0D 01 01

Response = 8D 01 00



### 8.1.6. Get Device Address

This command gets the device's Bluetooth address. (Used in USB mode only)

#### Command Format

Offset	Field	Size	Value	Description	
0	<i>abData1</i>	<i>CommandCode</i>	1	<b>0Eh</b>	Command Code of Get Device Address
1		<i>Len (CommandLength)</i>	1	00h	Number of extra bytes of data
2		<i>Data</i>	0	-	-

#### Response Format

Offset	Field	Size	Value	Description	
0	<i>abData2</i>	<i>ResponseCode</i>	1	<b>8Eh</b>	Response Code of Get Device Address
1		<i>Len (CommandLength)</i>	1	06h	Number of extra bytes of data
2		<i>Data</i>	6	-	6 bytes of Bluetooth address

#### Example:

Request = 0E 00

Response = 8E 06 AA BB CC DD EE FF

Device address: AA BB CC DD EE FF



### 8.1.7. Set Tx Power

This command sets the Bluetooth transmission power of the reader.

#### Command Format

Offset	Field	Size	Value	Description
0	<i>CommandCode</i>	1	<b>08h</b>	Command Code of Set Tx Power
1	<i>Len (CommandLength)</i>	1	01h	Number of extra bytes of data
2	<i>Data</i>	1	-	00h = -18 dBm (Default), Distance: ~4 meters 01h = -12 dBm Distance: ~7 meters 02h = -6 dBm Distance: ~16 meters 03h = 0 dBm Distance: ~25 meters

#### Response Format

Offset	Field	Size	Value	Description
0	<i>ResponseCode</i>	1	<b>88h</b>	Response Code of Set Tx Power
1	<i>Len (CommandLength)</i>	1	01h	Number of extra bytes of data
2	<i>Data</i>	1	-	00h = Success 01h = Fail

#### Example:

Request = 08 01 00

Response = 88 01 00





### 8.1.8. Read Tx Power value

This command checks the Bluetooth transmission power of the reader.

#### Command Format

Offset	Field	Size	Value	Description	
0	<i>abData1</i>	<i>CommandCode</i>	1	<b>09h</b>	Command Code of Read Tx Power
1		<i>Len (CommandLength)</i>	1	00h	Number of extra bytes of data
2		<i>Data</i>	0	-	-

#### Response Format

Offset	Field	Size	Value	Description	
0	<i>abData2</i>	<i>ResponseCode</i>	1	<b>89h</b>	Response Code of Read Tx Power
1		<i>Len (CommandLength)</i>	1	01h	Number of extra bytes of data
2		<i>Data</i>	1	-	00h = -18 dBm (Default) Distance: ~4 meters 01h = -12 dBm Distance: ~7 meters 02h = -6 dBm Distance: ~16 meters 03h = 0 dBm Distance: ~25 meters

#### Example:

Request = 09 00

Response = 89 01 00



### 8.1.9. Set Card Response Time Interval

This command sets the SAM card response time interval.

#### Command Format

Offset	Field	Size	Value	Description
0	<i>CommandCode</i>	1	<b>18h</b>	Command Code of Set Card Response Time Interval
1	<i>Len (CommandLength)</i>	1	01h	Number of extra bytes of data
2	<i>Data</i>	1	-	00h = 0s 01h = 500 ms 02h = 1000 ms 03h = 1500 ms (Default) 04h = 2000 ms 05h = 2500 ms 06h = 3000 ms

#### Response Format

Offset	Field	Size	Value	Description
0	<i>ResponseCode</i>	1	<b>98h</b>	Response Code of Set Card Response Time Interval
1	<i>Len (CommandLength)</i>	1	01h	Number of extra bytes of data
2	<i>Data</i>	1	-	00h = Success 01h = Fail

#### Example:

Request = 18 01 00

Response = 98 01 00



### 8.1.10. Get Card Response Time Interval

This command gets the SAM card response time interval.

#### Command Format

Offset	Field	Size	Value	Description	
0	<i>abData1</i>	<i>CommandCode</i>	1	<b>19h</b>	Command Code of Get Card Response Time Interval
1		<i>Len (CommandLength)</i>	1	00h	Number of extra bytes of data
2		<i>Data</i>	0	-	

#### Response Format

Offset	Field	Size	Value	Description	
0	<i>abData2</i>	<i>ResponseCode</i>	1	<b>99h</b>	Response Code of Get Card Response Time Interval
1		<i>Len (CommandLength)</i>	1	01h	Number of extra bytes of data
2		<i>Data</i>	1	-	00h = 0s 01h = 500 ms 02h = 1000 ms 03h = 1500 ms (Default) 04h = 2000 ms 05h = 2500 ms 06h = 3000 ms

#### Example:

Request = 19 00

Response = 99 01 03



### 8.1.11. Set Card Simulation Option

This command sets the option of card simulation.

#### Command Format

Offset	Field	Size	Value	Description	
0	<i>abData1</i>	<i>CommandCode</i>	1	<b>1Ah</b>	Command Code of Set Card Simulation Option
1		<i>Len (CommandLength)</i>	1	01h	Number of extra bytes of data
2		<i>Data</i>	1	-	00h = Disable 01h = Enable

#### Response Format

Offset	Field	Size	Value	Description	
0	<i>abData2</i>	<i>ResponseCode</i>	1	<b>9Ah</b>	Response Code of Set Card Simulation Option
1		<i>Len (CommandLength)</i>	1	01h	Number of extra bytes of data
2		<i>Data</i>	1	-	00h = Disable 01h = Enable

#### Example:

Request = 1A 01 01

Response = 9A 01 01



### 8.1.12. Get Card Simulation Option

This command gets the option of card simulation.

#### Command Format

Offset	Field	Size	Value	Description	
0	<i>abData1</i>	<i>CommandCode</i>	1	<b>1Ah</b>	Command Code of Get Card Simulation Option
1		<i>Len (CommandLength)</i>	1	00h	Number of extra bytes of data
2		<i>Data</i>	0	-	

#### Response Format

Offset	Field	Size	Value	Description	
0	<i>abData2</i>	<i>ResponseCode</i>	1	<b>9Ah</b>	Response Code of Get Card Simulation Option
1		<i>Len (CommandLength)</i>	1	01h	Number of extra bytes of data
2		<i>Data</i>	1	-	00h = Disable 01h = Enable

#### Example:

Request = 1A 00

Response = 9A 01 01



### 8.1.13. Check button status

This command checks the current button status.

#### Command Format

Offset	Field	Size	Value	Description	
0	<i>abData1</i>	<i>CommandCode</i>	1	<b>1Bh</b>	Command Code of Check button status
1		<i>Len (CommandLength)</i>	1	00h	Number of extra bytes of data
2		<i>Data</i>	0	-	

#### Response Format

Offset	Field	Size	Value	Description	
0	<i>abData2</i>	<i>ResponseCode</i>	1	<b>9Bh</b>	Response Code of Check button status
1		<i>Len (CommandLength)</i>	1	01h	Number of extra bytes of data
2		<i>Data</i>	1	-	00h = Release 01h = Pressed

#### Example:

Request = 1B 00

Response (Buttons is pressed) = 9B 01 01



### 8.1.14. Customer Master Key Reset Request

This command requests the reader to generate a random number for the Customer Master Key Reset authentication.

#### Command Format

Offset	Field	Size	Value	Description	
0	<i>abData1</i>	<i>CommandCode</i>	1	<b>0Fh</b>	Command Code of Customer Master Key Reset Request
1		<i>Len (CommandLength)</i>	1	00h	Number of extra bytes of data
2		<i>Data</i>	0	-	-

#### Response Format

Offset	Field	Size	Value	Description	
0	<i>abData2</i>	<i>ResponseCode</i>	1	<b>8Fh</b>	Response Code of Rewrite Master Key
1		<i>Len (CommandLength)</i>	1	10h	Number of extra bytes of data
2		<i>Data</i>	16	-	16 bytes of random number (KeyRSTRnd[0:15]) generated by the reader

#### Example:

1. Generate random number.

Customer Master Key Reset Request = 0F 00

Customer Master Key Reset Command Response = 8F 10 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11

2. Encrypt the random number and new master key using the original master key by AES128 CBC. This is done by the application's encryption engine and result will be stored for later use.

Random number: 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11

Encrypted random number: F1 9F D2 D2 BA 1C 22 E1 6D C1 FE 1B 4B 43 D5 30

New Master key: 11 22 33 44 55 66 77 88 11 22 33 44 55 66 77 88

Encrypted new master key: 27 E7 DA BE A6 1E 4B CD 29 F6 9B 36 25 05 8E 41

3. Rewrite the Master Key (see [Rewrite Master Key Command](#)).

Rewrite Master Key Command Request = 07 20 F1 9F D2 D2 BA 1C 22 E1 6D C1 FE 1B 4B 43 D5 30 27 E7 DA BE A6 1E 4B CD 29 F6 9B 36 25 05 8E 41

Rewrite Master Key Command Response = 87 01 00



## 9.0. Memory Card Command Set

### 9.1. Memory Card – 1, 2, 4, 8, and 16 kilobit I2C Card

#### 9.1.1. SELECT\_CARD\_TYPE

This command powers down and up the selected card inserted in the card reader and performs a card reset.

**Note:** This command can only be used after the logical smart card reader communication has been established using the `SCardConnect( )` API. For details of `SCardConnect( )` API, please refer to PC/SC specification.

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU					
CLA	INS	P1	P2	Lc	Card Type
FFh	A4h	00h	00h	01h	01h

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

**SW1 SW2** = 90 00h if no error

#### 9.1.1.1. SELECT\_PAGE\_SIZE

This command chooses the page size to read the smart card. The default value is 8-byte page write. It will reset to default value whenever the card is removed or the reader is powered off.

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU					
CLA	INS	P1	P2	Lc	Page Size
FFh	01h	00h	00h	01h	

Where:

**Page size** = 03h for 8-byte page write  
 = 04h for 16-byte page write  
 = 05h for 32-byte page write  
 = 06h for 64-byte page write  
 = 07h for 128-byte page write





Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

**SW1 SW2** = 90 00h if no error

### 9.1.1.2. READ\_MEMORY\_CARD

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU				
CLA	INS	Byte Address		MEM_L
		MSB	LSB	
FFh	B0h			

Where:

**Byte Address** Memory address location of the memory card

**MEM\_L** Length of data to be read from the memory card

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

BYTE 1	...	...	BYTE N	SW1	SW2

Where:

**BYTE x** Data read from memory card

**SW1 SW2** = 90 00h if no error

### 9.1.1.3. WRITE\_MEMORY\_CARD

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU								
CLA	INS	Byte Address		MEM_L	Byte 1	....	....	Byte n
		MSB	LSB					
FFh	D0h							

Where:

**Byte Address** Memory address location of the memory card

**MEM\_L** Length of data to be written to the memory card

**Byte x** Data to be written to the memory card



Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

**SW1 SW2** = 90 00h if no error



## 9.2. Memory Card – 32, 64, 128, 256, 512, and 1024 kilobit I2C Card

### 9.2.1. SELECT\_CARD\_TYPE

This command powers down and up the selected card that is inserted in the card reader and performs a card reset.

**Note:** This command can only be used after the logical smart card reader communication has been established using the `SCardConnect( )` API. For details of `SCardConnect( )` API, please refer to PC/SC specifications.

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU					
CLA	INS	P1	P2	Lc	Card Type
FFh	A4h	00h	00h	01h	02h

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

**SW1 SW2** = 90 00h if no error

### 9.2.2. SELECT\_PAGE\_SIZE

This command chooses the page size to read the smart card. The default value is 8-byte page write. It will reset to default value whenever the card is removed or the reader is powered off.

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU					
CLA	INS	P1	P2	Lc	Page size
FFh	01h	00h	00h	01h	

Where:

**Data**                    TPDU to be sent to the card

**Page size**            = 03h for 8-byte page write  
                               = 04h for 16-byte page write  
                               = 05h for 32-byte page write  
                               = 06h for 64-byte page write  
                               = 07h for 128-byte page write



Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

**SW1 SW2** = 90 00h if no error

### 9.2.3. READ\_MEMORY\_CARD

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU				
CLA	INS	Byte Address		MEM_L
		MSB	LSB	
FFh				

Where:

**INS** = B0h for 32 kilobit, 64 kilobit, 128 kilobit, 256 kilobit and 512 kilobit iic card

= 1011 000\*b for 1024 kilobit iic card,  
where \* is the MSB of the 17 bit addressing

**Byte Address** Memory address location of the memory card

**MEM\_L** Length of data to be read from the memory card

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

BYTE 1	...	...	BYTE N	SW1	SW2

Where:

**BYTE x** Data read from memory card

**SW1 SW2** = 90 00h if no error

### 9.2.4. WRITE\_MEMORY\_CARD

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU								
CLA	INS	Byte Address		MEM_L	Byte 1	...	...	Byte n
		MSB	LSB					
FFh								

Where:

**INS** = D0h for 32 kilobit, 64 kilobit, 128 kilobit, 256 kilobit, 512 kilobit iic card

= 1101 000\*b for 1024 kilobit iic card,  
where \* is the MSB of the 17 bit addressing



**Byte Address**      Memory address location of the memory card  
**MEM\_L**              Length of data to be written to the memory card  
**Byte x**                Data to be written to the memory card

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

**SW1 SW2** = 90 00h if no error



### 9.3. Memory Card – ATMEL AT88SC153

#### 9.3.1. SELECT\_CARD\_TYPE

This command powers up and down the selected card that is inserted in the card reader and performs a card reset. It will also select the page size to be 8-byte page write.

**Note:** This command can only be used after the logical smart card reader communication has been established using the SCardConnect( ) API. For details of SCardConnect( ) API, please refer to PC/SC specifications.

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU					
CLA	INS	P1	P2	Lc	Card Type
FFh	A4h	00h	00h	01h	03h

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

**SW1 SW2** = 90 00h if no error

#### 9.3.2. READ\_MEMORY\_CARD

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU				
CLA	INS	P1	Byte Address	MEM_L
FFh		00h		

Where:

**INS** = B0h for reading zone 00b  
 = B1h for reading zone 01b  
 = B2h for reading zone 10b  
 = B3h for reading zone 11b  
 = B4h for reading fuse

**Byte Address** Memory address location of the memory card

**MEM\_L** Length of data to be read from the memory card



Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

BYTE 1	...	...	BYTE N	SW1	SW2

Where:

**BYTE x**            Data read from memory card  
**SW1 SW2**        = 90 00h if no error

### 9.3.3. WRITE\_MEMORY\_CARD

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU								
CLA	INS	P1	Byte Address	MEM_L	Byte 1	...	...	Byte n
FFh		00h						

Where:

**INS**                    = D0h for writing zone 00b  
                              = D1h for writing zone 01b  
                              = D2h for writing zone 10b  
                              = D3h for writing zone 11b  
                              = D4h for writing fuse

**Byte Address**        Memory address location of the memory card

**MEM\_L**                Length of data to be written to the memory card

**MEM\_D**                Data to be written to the memory card

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

**SW1 SW2** = 90 00h if no error



### 9.3.4. VERIFY\_PASSWORD

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU							
CLA	INS	P1	P2	Lc	Pw(0)	Pw(1)	Pw(2)
FFh	20h	00h		03h			

Where:

**Pw(0),Pw(1),Pw(2)** Passwords to be sent to memory card  
**P2** = 0000 00rp<sub>b</sub>  
 where the two bits “rp” indicate the password to compare  
 r = 0: Write password,  
 r = 1: Read password,  
 p : Password set number,  
 rp = 01 for the secure code.

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2 ErrorCnt
90h	

Where:

**SW1** = 90h  
**SW2 (ErrorCnt)** = Error Counter. FFh indicates the verification is correct. 00h indicates the password is locked (or exceeded the maximum number of retries). Other values indicate the current verification has failed.

### 9.3.5. INITIALIZE\_AUTHENTICATION

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU								
CLA	INS	P1	P2	Lc	Q(0)	Q(1)	...	Q(7)
FFh	84h	00h	00h	08h				

Where:

**Q(0),Q(1)...Q(7)** Host random number, 8 bytes





Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

**SW1 SW2** = 90 00h if no error

### 9.3.6. VERIFY\_AUTHENTICATION

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU								
CLA	INS	P1	P2	Lc	Ch(0)	Ch(1)	...	Ch(7)
FFh	82h	00h	00h	08h				

Where:

**Ch(0),Ch(1)...Ch(7)** Host challenge, 8 bytes

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

**SW1 SW2** = 90 00h if no error



## 9.4. Memory Card – ATMEL AT88C1608

### 9.4.1. SELECT\_CARD\_TYPE

This command powers down and up the selected card that is inserted in the card reader and performs a card reset. It will also select the page size to be 16-byte page write.

**Note:** This command can only be used after the logical smart card reader communication has been established using the SCardConnect( ) API. For details of SCardConnect( ) API, please refer to PC/SC specifications.

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU					
CLA	INS	P1	P2	Lc	Card Type
FFh	A4h	00h	00h	01h	04h

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

**SW1 SW2** = 90 00h if no error

### 9.4.2. READ\_MEMORY\_CARD

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU				
CLA	INS	Zone Address	Byte Address	MEM_L
FFh				

Where:

**INS** = B0h for reading user zone  
= B1h for reading configuration zone or reading fuse

**Zone Address** = 0000 0A<sub>10</sub>A<sub>9</sub>A<sub>8</sub>b where A<sub>10</sub> is the MSB of zone address  
= don't care for reading fuse

**Byte Address** = A<sub>7</sub>A<sub>6</sub>A<sub>5</sub>A<sub>4</sub> A<sub>3</sub>A<sub>2</sub>A<sub>1</sub>A<sub>0</sub>b is the memory address location of the memory card  
= 1000 0000b for reading fuse

**MEM\_L** Length of data to be read from the memory card



Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

BYTE 1	...	...	BYTE N	SW1	SW2

Where:

**BYTE x** Data read from memory card

**SW1 SW2** = 90 00h if no error

### 9.4.3. WRITE\_MEMORY\_CARD

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU								
CLA	INS	Zone Address	Byte Address	MEM_L	Byte 1	...	...	Byte n
FFh								

Where:

**INS** = D0h for writing user zone

= D1h for writing configuration zone or writing fuse

**Zone Address** = 0000 0A<sub>10</sub>A<sub>9</sub>A<sub>8</sub>b where A<sub>10</sub> is the MSB of zone address

= Don't care for writing fuse

**Byte Address** = A<sub>7</sub>A<sub>6</sub>A<sub>5</sub>A<sub>4</sub> A<sub>3</sub>A<sub>2</sub>A<sub>1</sub>A<sub>0</sub>b is the memory address location of the memory card

= 1000 0000b for writing fuse

**MEM\_L** Length of data to be written to the memory card

**Byte x** Data to be written to the memory card

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

**SW1 SW2** = 90 00h if no error

### 9.4.4. VERIFY\_PASSWORD

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU								
CLA	INS	P1	P2	Lc	Data			
FFh	20h	00h	00h	04h	RP	Pw(0)	Pw(1)	Pw(2)

Where:

**Pw(0),Pw(1),Pw(2)** Passwords to be sent to memory card  
**RP** = 0000 rp<sub>2</sub>p<sub>1</sub>p<sub>0</sub>b  
 where the four bits “rp<sub>2</sub>p<sub>1</sub>p<sub>0</sub>” indicate the password to compare:  
 r = 0: Write password,  
 r = 1: Read password,  
 p<sub>2</sub>p<sub>1</sub>p<sub>0</sub>: Password set number.  
 (rp<sub>2</sub>p<sub>1</sub>p<sub>0</sub> = 0111 for the secure code)

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2 ErrorCnt
90h	

Where:

**SW1** = 90h  
**SW2 (ErrorCnt)** = Error Counter. FFh indicates the verification is correct. 00h indicates the password is locked (or exceeded the maximum number of retries). Other values indicate the current verification has failed.

### 9.4.5. INITIALIZE\_AUTHENTICATION

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU								
CLA	INS	P1	P2	Lc	Q(0)	Q(1)	...	Q(7)
FFh	84h	00h	00h	08h				

Where:

**Byte Address** Memory address location of the memory card  
**Q(0),Q(1)...Q(7)** Host random number, 8 bytes



Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

**SW1 SW2** = 90 00h if no error

### 9.4.6. VERIFY\_AUTHENTICATION

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU								
CLA	INS	P1	P2	Lc	Q1(0)	Q1(1)	...	Q1(7)
FFh	82h	00h	00h	08h				

Where:

**Byte Address**                      Memory address location of the memory card

**Q1(0),Q1(1)...Q1(7)**          Host challenge, 8 bytes

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

**SW1 SW2** = 90 00h if no error



## 9.5. Memory Card – SLE4418/SLE4428/SLE5518/SLE5528

### 9.5.1. SELECT\_CARD\_TYPE

This command powers up and down the selected card that is inserted in the card reader and performs a card reset.

**Note:** This command can only be used after the logical smart card reader communication has been established using the SCardConnect( ) API. For details of SCardConnect( ) API, please refer to PC/SC specifications.

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU					
CLA	INS	P1	P2	Lc	Card Type
FFh	A4h	00h	00h	01h	05h

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

**SW1 SW2** = 90 00h if no error

### 9.5.2. READ\_MEMORY\_CARD

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU				
CLA	INS	Byte Address		MEM_L
		MSB	LSB	
FFh	B0h			

Where:

**MSB Byte Address** = 0000 00A<sub>9</sub>A<sub>8</sub>b is the memory address location of the memory card

**LSB Byte Address** = A<sub>7</sub>A<sub>6</sub>A<sub>5</sub>A<sub>4</sub> A<sub>3</sub>A<sub>2</sub>A<sub>1</sub>A<sub>0</sub>b is the memory address location of the memory card

**MEM\_L** Length of data to be read from the memory card

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

BYTE 1	...	...	BYTE N	SW1	SW2

Where:

**BYTE x** Data read from memory card

**SW1, SW2** = 90 00h if no error



### 9.5.3. READ\_PRESENTATION\_ERROR\_COUNTER\_MEMORY\_CARD (SLE4428 and SLE5528)

This command is used to read the presentation error counter for the secret code.

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU				
CLA	INS	P1	P2	MEM_L
FFh	B1h	00h	00h	03h

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

ERRCNT	DUMMY 1	DUMMY 2	SW1	SW2

Where:

- ERRCNT** Error Counter. FFh indicates that the last verification is correct. 00h indicates that the password is locked (exceeded the maximum number of retries). Other values indicate that the last verification has failed.
- DUMMY** Two bytes dummy data read from the card
- SW1 SW2** = 90 00h if no error

### 9.5.4. READ\_PROTECTION\_BIT

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU				
CLA	INS	Byte Address		MEM_L
		MSB	LSB	
FFh	B2h			

Where:

- MSB Byte Address** = 0000 00A<sub>9</sub>A<sub>8</sub>b is the memory address location of the memory card
- LSB Byte Address** = A<sub>7</sub>A<sub>6</sub>A<sub>5</sub>A<sub>4</sub> A<sub>3</sub>A<sub>2</sub>A<sub>1</sub>A<sub>0</sub>b is the memory address location of the memory card
- MEM\_L** Length of protection bits to be read from the card, in multiples of 8 bits. Maximum value is 32.  
 $MEM\_L = 1 + INT( (number\ of\ bits - 1) / 8 )$

For example, to read 8 protection bits starting from memory 0010h, the following pseudo-APDU should be issued:

FF B2 00 10 01h



Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

PROT 1	...	...	PROT L	SW1	SW2

Where:

- PROT y** Bytes containing the protection bits
- SW1, SW2** = 90 00h if no error

The arrangement of the protection bits in the PROT bytes is as follows:

PROT 1								PROT 2								...									
P8	P7	P6	P5	P4	P3	P2	P1	P16	P15	P14	P13	P12	P11	P10	P9	..	..	..	..	..	..	..	..	P18	P17

Where:

- Px** is the protection bit of BYTE x in the response data
- '0' byte is write protected
- '1' byte can be written

### 9.5.5. WRITE\_MEMORY\_CARD

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU								
CLA	INS	Byte Address		MEM_L	Byte 1	....	....	Byte N
		MSB	LSB					
FFh	D0h							

Where:

- MSB Byte Address** = 0000 00A<sub>9</sub>A<sub>8</sub>b is the memory address location of the memory card
- LSB Byte Address** = A<sub>7</sub>A<sub>6</sub>A<sub>5</sub>A<sub>4</sub> A<sub>3</sub>A<sub>2</sub>A<sub>1</sub>A<sub>0</sub>b is the memory address location of the memory card
- MEM\_L** Length of data to be written to the memory card
- Byte x** Data to be written to the memory card

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

- SW1 SW2** = 90 00h if no error





### 9.5.6. WRITE\_PROTECTION\_MEMORY\_CARD

Each byte specified in the command is used in the card to compare the byte stored in a specified address location. If the data match, the corresponding protection bit is irreversibly programmed to '0'.

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU								
CLA	INS	Byte Address		MEM_L	Byte 1	...	...	Byte N
		MSB	LSB					
FFh	D1h							

Where:

- MSB Byte Address** = 0000 00A<sub>9</sub>A<sub>8</sub>b is the memory address location of the memory card
- LSB Byte Address** = A<sub>7</sub>A<sub>6</sub>A<sub>5</sub>A<sub>4</sub> A<sub>3</sub>A<sub>2</sub>A<sub>1</sub>A<sub>0</sub>b is the memory address location of the memory card
- MEM\_L** Length of data to be written to the memory card
- Byte x** Byte values to be compared with the data in the card starting at *Byte Address*. BYTE 1 is compared with the data at *Byte Address*; BYTE N is compared with the data at (*Byte Address+N-1*).

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

- SW1 SW2** = 90 00h if no error

### 9.5.7. PRESENT\_CODE\_MEMORY\_CARD (SLE4428 and SLE5528)

This command is used to submit the secret code to the memory card to enable the write operation with the SLE4428 and SLE5528 card, the following actions are executed:

1. Search a '1' bit in the presentation error counter and write the bit to '0'.
2. Present the specified code to the card.
3. Try to erase the presentation error counter.

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU						
CLA	INS	P1	P2	MEM_L	CODE	
					Byte 1	Byte 2
FFh	20h	00h	00h	02h		

Where:

- CODE** Two bytes secret code (PIN)



Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2 ErrorCnt
90h	

Where:

**SW1** = 90h

**SW2 (ErrorCnt)** = Error Counter. FFh indicates successful verification. 00h indicates that the password is locked (or exceeded the maximum number of retries). Other values indicate that current verification has failed.



## 9.6. Memory Card – SLE4432/SLE4442/SLE5532/SLE5542

### 9.6.1. SELECT\_CARD\_TYPE

This command powers down and up the selected card that is inserted in the card reader and performs a card reset.

**Note:** This command can only be used after the logical smart card reader communication has been established using the SCardConnect() API. For details of SCardConnect() API, please refer to PC/SC specifications.

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU					
CLA	INS	P1	P2	Lc	Card Type
FFh	A4h	00h	00h	01h	06h

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

**SW1 SW2** = 90 00h if no error

### 9.6.2. READ\_MEMORY\_CARD

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU				
CLA	INS	P1	Byte Address	MEM_L
FFh	B0h	00h		

Where:

**Byte Address** = A<sub>7</sub>A<sub>6</sub>A<sub>5</sub>A<sub>4</sub> A<sub>3</sub>A<sub>2</sub>A<sub>1</sub>A<sub>0</sub>b is the memory address location of the memory card

**MEM\_L** Length of data to be read from the memory card

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

BYTE 1	...	...	BYTE N	SW1	SW2

Where:

**BYTE x** Data read from memory card

**SW1, SW2** = 90 00h if no error



### 9.6.3. READ\_PRESENTATION\_ERROR\_COUNTER\_MEMORY\_CARD (SLE 4442 and SLE 5542)

This command is used to read the presentation error counter for the secret code.

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU				
CLA	INS	P1	P2	MEM_L
FFh	B1h	00h	00h	04h

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

ERRCNT	DUMMY 1	DUMMY 2	DUMMY 3	SW1	SW2

Where:

- ERRCNT** Error counter. 07h indicates that the last verification is correct. 00h indicates that the password is locked (exceeded the maximum number of retries). Other values indicate that the last verification has failed.
- DUMMY** Three bytes dummy data read from the card
- SW1 SW2** = 90 00h if no error

### 9.6.4. READ\_PROTECTION\_BITS

This command is used to read the protection bits for the first 32 bytes.

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU				
CLA	INS	P1	P2	MEM_L
FFh	B2h	00h	00h	04h

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

PROT 1	PROT 2	PROT 3	PROT 4	SW1	SW2

Where:

- PROT y** Bytes containing the protection bits from protection memory
- SW1, SW2** = 90 00h if no error



The arrangement of the protection bits in the PROT bytes is as follows:

PROT 1								PROT 2								...									
P8	P7	P6	P5	P4	P3	P2	P1	P16	P15	P14	P13	P12	P11	P10	P9	..	..	..	..	..	..	..	..	P18	P17

Where:

**Px** is the protection bit of BYTE x in the response data

'0' byte is write protected

'1' byte can be written

### 9.6.5. WRITE\_MEMORY\_CARD

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU								
CLA	INS	P1	Byte Address	MEM_L	Byte 1	...	...	Byte N
FFh	D0h	00h						

Where:

**Byte Address** =  $A_7A_6A_5A_4 A_3A_2A_1A_0b$  is the memory address location of the memory card

**MEM\_L** Length of data to be written to the memory card

**Byte x** Data to be written to the memory card

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

**SW1 SW2** = 90 00h if no error

### 9.6.6. WRITE\_PROTECTION\_MEMORY\_CARD

Each byte specified in the command is internally in the card compared with the byte stored at the specified address and if the data match, the corresponding protection bit is irreversibly programmed to '0'.

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU								
CLA	INS	P1	Byte Address	MEM_L	Byte 1	...	...	Byte N
FFh	D1h	00h						

Where:

**Byte Address** =  $000A_4 A_3A_2A_1A_0b$  (00h to 1Fh) is the protection memory address location of the memory card

**MEM\_L** Length of data to be written to the memory card



**Byte x** Byte values to be compared with the data in the card starting at Byte Address. BYTE 1 is compared with the data at Byte Address; BYTE N is compared with the data at (Byte Address+N-1).

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

**SW1 SW2** = 90 00h if no error

### 9.6.7. PRESENT\_CODE\_MEMORY\_CARD (SLE 4442 and SLE 5542)

To submit the secret code to the memory card to enable the write operation with the SLE 4442 and SLE 5542 card, the following actions are executed:

1. Search a '1' bit in the presentation error counter and write the bit to '0'.
2. Present the specified code to the card.
3. Try to erase the presentation error counter.

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU							
CLA	INS	P1	P2	MEM_L	CODE		
					Byte 1	Byte 2	Byte 3
FFh	20h	00h	00h	03h			

Where:

**CODE** Three bytes secret code (PIN)

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2 ErrorCnt
90h	

Where:

**SW1** = 90h

**SW2 (ErrorCnt)** = Error Counter. 07h indicates that the verification is correct. 00h indicates the password is locked (exceeded the maximum number of retries). Other values indicate that the current verification has failed.



**9.6.8. CHANGE\_CODE\_MEMORY\_CARD (SLE 4442 and SLE 5542)**

This command is used to write the specified data as new secret code in the card.

The current secret code must have been presented to the card with the *PRESENT\_CODE* command prior to the execution of this command.

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU							
CLA	INS	P1	P2	MEM_L	CODE		
					Byte 1	Byte 2	Byte 3
FFh	D2h	00h	01h	03h			

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

**SW1 SW2** = 90 00h if no error



## 9.7. Memory Card – SLE 4406/SLE 4436/SLE 5536/SLE 6636

### 9.7.1. SELECT\_CARD\_TYPE

This command powers down and up the selected card that is inserted in the card reader and performs a card reset.

**Note:** This command can only be used after the logical smart card reader communication has been established using the *SCardConnect()* API. For details of *SCardConnect()* API, please refer to PC/SC specifications.

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU					
CLA	INS	P1	P2	Lc	Card Type
FFh	A4h	00h	00h	01h	07h

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

**SW1 SW2** = 90 00h if no error

### 9.7.2. READ\_MEMORY\_CARD

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU				
CLA	INS	P1	Byte Address	MEM_L
FFh	B0h	00h		

Where:

**Byte Address** = Memory address location of the memory card

**MEM\_L** Length of data to be read from the memory card

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

BYTE 1	...	...	BYTE N	SW1	SW2

Where:

**BYTE x** Data read from memory card

**SW1, SW2** = 90 00h if no error





### 9.7.3. WRITE\_ONE\_BYTE\_MEMORY\_CARD

This command is used to write one byte to the specified address of the inserted card. The byte is written to the card with LSB first, i.e., the bit at card address 0 is regarded as the LSB of byte 0.

Four different WRITE modes are available for this card type, which are distinguished by a flag in the command data field:

- a. **Write**  
The byte value specified in the command is written to the specified address. This command can be used for writing personalization data and counter values to the card.
- b. **Write with carry**  
The byte value specified in the command is written to the specified address and the command is sent to the card to erase the next lower counter stage. Thus, this write mode can only be used for updating the counter value in the card.
- c. **Write with backup enabled** (SLE 4436, SLE 5536 and SLE 6636 only)  
The byte value specified in the command is written to the specified address. This command can be used for writing personalization data and counter values to the card. Backup bit is enabled to prevent data loss when card tearing occurs.
- d. **Write with carry and backup enabled** (SLE 4436, SLE 5536 and SLE 6636 only)  
The byte value specified in the command is written to the specified address and the command is sent to the card to erase the next lower counter stage. Thus, this write mode can only be used for updating the counter value in the card. Backup bit is enabled to prevent data loss when card tearing occurs.

With all write modes, the byte at the specified card address is not erased prior to the write operation and, hence, memory bits can only be programmed from '1' to '0'.

The backup mode available in the SLE 4436 and SLE 5536 card can be enabled or disabled in the write operation.

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU						
CLA	INS	P1	Byte Address	MEM_L	MODE	BYTE
FFh	D0h	00h		02h		

Where:

- Byte Address** = Memory address location of the memory card
- MODE** Specifies the write mode and backup option
  - 00h: Write
  - 01h: Write with carry
  - 02h: Write with backup enabled (SLE 4436, SLE 5536 and SLE 6636 only)
  - 03h: Write with carry and with backup enabled (SLE 4436, SLE 5536 and SLE 6636 only)
- BYTE** Byte value to be written to the card



Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

**SW1 SW2** = 90 00h if no error

#### 9.7.4. PRESENT\_CODE\_MEMORY\_CARD

To submit the secret code to the memory card to enable the card personalization mode, the following actions are executed:

1. Search a '1' bit in the presentation counter and write the bit to '0'.
2. Present the specified code to the card.

The ACR3901T-W1 does not try to erase the presentation counter after the code submission. This must be done by the application software through a separate 'Write with carry' command.

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU								
CLA	INS	P1	P2	MEM_L	CODE			
					ADDR	Byte 1	Byte 2	Byte 3
FFh	20h	00h	00h	04h	09h			

Where:

**ADDR** Byte address of the presentation counter in the card

**CODE** Three bytes secret code (PIN)

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

**SW1 SW2** = 90 00h if no error



**9.7.5. AUTHENTICATE\_MEMORY\_CARD (SLE 4436, SLE 5536 and SLE 6636)**

To read a card authentication certificate from a SLE 5536 or SLE 6636 card, the ACR3901T-W1 executes the following actions:

1. Select Key 1 or Key 2 in the card as specified in the command.
2. Present the challenge data specified in the command to the card.
3. Generate the specified number of CLK pulses for each bit of authentication data computed by the card.
4. Read 16 bits of authentication data from the card.
5. Reset the card to normal operation mode.

The authentication has to be performed in two steps. The first step is to send the Authentication Certificate to the card. The second step is to get back two bytes of authentication data calculated by the card.

**Step 1: Send Authentication Certificate to the Card.**

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU											
CLA	INS	P1	P2	MEM_L	CODE						
					KEY	CLK_CNT	Byte 1	Byte 2	.....	Byte 5	Byte 6
FFh	84h	00h	00h	08h							

Where:

- KEY** Key to be used for the computation of the authentication certificate:  
 00h: Key 1 with no cipher block chaining  
 01h: Key 2 with no cipher block chaining  
 80h: Key 1 with cipher block chaining (SLE 5536 and SLE 6636 only)  
 81h: Key 2 with cipher block chaining (SLE 5536 and SLE 6636 only)
- CLK\_CNT** Number of CLK pulses to be supplied to the card for the computation of each bit of the authentication certificate. Typical value is 160 clocks (A0h)
- BYTE 1...6** Card challenge data

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2
61h	02h

Where:

**SW1 SW2** = 61 02h if no error, meaning two bytes of authentication data are ready. The authentication data can be retrieved by *Get\_Response* command



**Step 2:** Get back the Authentication Data (*Get\_Response*).

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU				
CLA	INS	P1	P2	MEM_L
FFh	C0h	00h	00h	02h

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

CERT	SW1	SW2

Where:

**CERT**            16 bits of authentication data computed by the card. The LSB of BYTE 1 is the first authentication bit read from the card.

**SW1 SW2**        = 90 00h if no error



## 9.8. Memory Card – SLE 4404

### 9.8.1. SELECT\_CARD\_TYPE

This command powers up and down the selected card that is inserted in the card reader and performs a card reset.

**Note:** This command can only be used after the logical smart card reader communication has been established using the SCardConnect() API. For details of SCardConnect() API, please refer to PC/SC specifications.

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU					
CLA	INS	P1	P2	Lc	Card Type
FFh	A4h	00h	00h	01	08h

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

**SW1 SW2** = 90 00h if no error

### 9.8.2. READ\_MEMORY\_CARD

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU				
CLA	INS	P1	Byte Address	MEM_L
FFh	B0h	00h		

Where:

**Byte Address** = Memory address location of the memory card

**MEM\_L** Length of data to be read from the memory card

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

BYTE 1	...	...	BYTE N	SW1	SW2

Where:

**BYTE x** Data read from memory card

**SW1 SW2** = 90 00h if no error



### 9.8.3. WRITE\_MEMORY\_CARD

This command is used to write data to the specified address of the inserted card. The byte is written to the card with LSB first, i.e., the bit at card address 0 is regarded as the LSB of byte 0.

The byte at the specified card address is not erased prior to the write operation and, hence, memory bits can only be programmed from '1' to '0'.

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU								
CLA	INS	P1	Byte Address	MEM_L	Byte 1	...	...	Byte N
FFh	D0h	00h						

Where:

- Byte Address** = Memory address location of the memory card
- MEM\_L** Length of data to be written to the memory card
- BYTE** Byte value to be written to the card

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

- SW1 SW2** = 90 00h if no error

### 9.8.4. ERASE\_SCRATCH\_PAD\_MEMORY\_CARD

This command is used to erase the data of the scratch pad memory of the inserted card. All memory bits inside the scratch pad memory will be programmed to the state of '1'.

To erase error counter or user area, please use the *VERIFY\_USER\_CODE* command as specified in the **VERIFY\_USER\_CODE**.

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU				
CLA	INS	P1	Byte Address	MEM_L
FFh	D2h	00h		00h

Where:

- Byte Address** = Memory byte address location of the scratch pad  
Typical value is 02h



Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

**SW1 SW2** = 90 00h if no error

### 9.8.5. VERIFY\_USER\_CODE

This command is used to submit User Code (2 bytes) to the inserted card. User Code is used to enable the memory access of the card.

The following actions are executed:

1. Present the specified code to the card.
2. Search a '1' bit in the presentation error counter and write the bit to '0'.
3. Erase the presentation error counter. The User Error Counter can be erased when the submitted code is correct.

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU						
CLA	INS	Error Counter LEN	Byte Address	MEM_L	CODE	
					Byte 1	Byte 2
FFh	20h	04h	08h	02h		

Where:

**Error Counter LEN** Length of presentation error counter in bits  
**Byte Address** Byte address of the key in the card  
**CODE** 2 bytes User Code

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

**SW1 SW2** = 90 00h if no error  
= 63 00h if there are no more retries

**Note:** After *SW1SW2 = 9000h* has been received, read back the User Error Counter to check if the *VERIFY\_USER\_CODE* is correct. If User Error Counter is erased and is equal to "FFh," the previous verification is successful.



### 9.8.6. VERIFY\_MEMORY\_CODE

This command is used to submit Memory Code (4 bytes) to the inserted card. Memory Code is used to authorize the reloading of the user memory, together with the User Code.

The following actions are executed:

1. Present the specified code to the card.
2. Search a '1' bit in the presentation error counter and write the bit to '0'.
3. Erase the presentation error counter. Please note that Memory Error Counter cannot be erased.

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU								
CLA	INS	Error Counter LEN	Byte Address	MEM_L	CODE			
					Byte 1	Byte 2	Byte 3	Byte 4
FFh	20h	40h	28h	04h				

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

- SW1 SW2** = 90 00h if no error
- = 63 00h if there are no more retries

**Note:** After SW1SW2 = 9000h has been received, read back the Application Area can check if the VERIFY\_MEMORY\_CODE is correct. If all data in Application Area is erased and is equal to "FFh," the previous verification is successful.





## 9.9. Memory Card – AT88SC101/AT88SC102/AT88SC1003

### 9.9.1. SELECT\_CARD\_TYPE

This command powers down and up the selected card that is inserted in the card reader and performs a card reset.

**Note:** This command can only be used after the logical smart card reader communication has been established using the SCardConnect() API. For details of SCardConnect() API, please refer to PC/SC specifications.

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU					
CLA	INS	P1	P2	Lc	Card Type
FFh	A4h	00h	00h	01h	09h

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

**SW1 SW2** = 90 00h if no error

### 9.9.2. READ\_MEMORY\_CARD

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU				
CLA	INS	P1	Byte Address	MEM_L
FFh	B0h	00h		

Where:

**Byte Address** = Memory address location of the memory card

**MEM\_L** Length of data to be read from the memory card

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

BYTE 1	...	...	BYTE N	SW1	SW2

Where:

**BYTE x** Data read from memory card

**SW1 SW2** = 90 00h if no error



### 9.9.3. WRITE\_MEMORY\_CARD

This command is used to write data to the specified address of the inserted card. The byte is written to the card with LSB first, i.e., the bit at card address 0 is regarded as the LSB of byte 0.

The byte at the specified card address is not erased prior to the write operation and, hence, memory bits can only be programmed from '1' to '0'.

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU								
CLA	INS	P1	Byte Address	MEM_L	Byte 1	....	....	Byte N
FFh	D0h	00h						

Where:

- Byte Address**      Memory address location of the memory card
- MEM\_L**              Length of data to be written to the memory card
- BYTE**                Byte value to be written to the card

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

**SW1 SW2** = 90 00h if no error

### 9.9.4. ERASE\_NON\_APPLICATION\_ZONE

This command is used to erase the data in Non-Application Zones. The EEPROM memory is organized into 16-bit words. Although erases are performed on single bit, the ERASE operation clears an entire word in the memory. Therefore, performing an ERASE on any bit in the word will clear ALL 16 bits of that word to the state of '1'.

To erase Error Counter or the data in Application Zones, please refer to the following:

1. *ERASE\_APPLICATION\_ZONE\_WITH\_ERASE* command as specified in:  
**ERASE APPLICATION ZONE WITH ERASE**
2. *ERASE\_APPLICATION\_ZONE\_WITH\_WRITE\_AND\_ERASE* command as specified in:  
**ERASE APPLICATION ZONE WITH WRITE AND ERASE**
3. *VERIFY\_SECURITY\_CODE* commands as specified in:  
**VERIFY SECURITY CODE**

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU				
CLA	INS	P1	Byte Address	MEM_L
FFh	D2h	00h		00h



Where:

**Byte Address** Memory byte address location of the word to be erased

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

**SW1 SW2** = 90 00h if no error

### 9.9.5. ERASE\_APPLICATION\_ZONE\_WITH\_ERASE

This command can be used in the following cases:

1. AT88SC101: To erase the data in Application Zone with EC Function Disabled.
2. AT88SC102: To erase the data in Application Zone 1.
3. AT88SC102: To erase the data in Application Zone 2 with EC2 Function Disabled.
4. AT88SC1003: To erase the data in Application Zone 1.
5. AT88SC1003: To erase the data in Application Zone 2 with EC2 Function Disabled.
6. AT88SC1003: To erase the data in Application Zone 3.

The following actions are executed for this command:

1. Present the specified code to the card.
  - a. Erase the presentation error counter. The data in corresponding Application Zone can be erased when the submitted code is correct.

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU									
CLA	INS	Error Counter LEN	Byte Address	MEM_L	CODE				
					Byte 1	Byte 2	...	...	Byte N
FFh	20h	00h							

Where:

**Error Counter LEN** Length of presentation error counter in bits. The value should be 00h always.

**Byte Address** Byte address of the Application Zone Key in the card. Please refer to the table below for the correct value.

	Byte Address	LEN
AT88SC101: Erase Application Zone with EC function disabled	96h	04h
AT88SC102: Erase Application Zone 1	56h	06h
AT88SC102: Erase Application Zone 2 with EC2 function disabled	9Ch	04h



	Byte Address	LEN
AT88SC1003: Erase Application Zone 1	36h	06h
AT88SC1003: Erase Application Zone 2 with EC2 function disabled	5Ch	04h
AT88SC1003: Erase Application Zone 3	C0h	06h

**MEM\_L** Length of the Erase Key. Please refer to the table above for the correct value.

**CODE** N bytes of Erase Key

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

**SW1 SW2** = 90 00h if no error

**Note:** After *SW1SW2* = 9000h has been received, read back the data in Application Zone to check if the *ERASE\_APPLICATION\_ZONE\_WITH\_ERASE* is correct. If all data in Application Zone is erased and is equal to "FFh," the previous verification is successful.

### 9.9.6. ERASE\_APPLICATION\_ZONE\_WITH\_WRITE\_AND\_ERASE

This command can be used in the following cases:

1. AT88SC101: To erase the data in Application Zone with EC Function Enabled.
2. AT88SC102: To erase the data in Application Zone 2 with EC2 Function Enabled.
3. AT88SC1003: To erase the data in Application Zone 2 with EC2 Function Enabled.

With EC or EC2 Function Enabled (that is, ECEN or EC2EN Fuse is undamaged and in "1" state), the following actions are executed:

1. Present the specified code to the card.
2. Search a '1' bit in the presentation error counter and write the bit to '0'.
3. Erase the presentation error counter. The data in corresponding Application Zone can be erased when the submitted code is correct.

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU								
CLA	INS	Error Counter LEN	Byte Address	MEM_L	CODE			
					Byte 1	Byte 2	Byte 3	Byte 4
FFh	20h	80h		04h				

Where:

**Error Counter LEN** Length of presentation error counter in bits. The value should be 80h always.



**Byte Address** Byte address of the Application Zone Key in the card

	Byte Address
AT88SC101	96h
AT88SC102	9Ch
AT88SC1003	5Ch

**CODE** 4 bytes Erase Key

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

**SW1 SW2** = 90 00h if no error  
= 63 00h if there are no more retries

**Note:** After *SW1SW2 = 9000h* has been received, read back the data in Application Zone can check whether the *ERASE\_APPLICATION\_ZONE\_WITH\_WRITE\_AND\_ERASE* is correct. If all data in Application Zone is erased and is equal to "FFh," the previous verification is successful.

### 9.9.7. VERIFY\_SECURITY\_CODE

This command is used to submit Security Code (2 bytes) to the inserted card. Security Code is to enable the memory access of the card.

The following actions are executed:

1. Present the specified code to the card
2. Search a '1' bit in the presentation error counter and write the bit to '0'
3. Erase the presentation error counter. The Security Code Attempts Counter can be erased when the submitted code is correct.

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU						
CLA	INS	Error Counter LEN	Byte Address	MEM_L	CODE	
					Byte 1	Byte 2
FFh	20h	08h	0Ah	02h		

Where:

**Error Counter LEN** Length of presentation error counter in bits  
**Byte Address** Byte address of the key in the card  
**CODE** 2 bytes Security Code



Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

- SW1, SW2** = 90 00h if no error Fe
- = 63 00h if there are no more retries

**Note:** After *SW1SW2 = 9000h* has been received, read back the Security Code Attempts Counter (SCAC) to check whether the *VERIFY\_USER\_CODE* is correct. If SCAC is erased and is equal to "FFh," the previous verification is successful.

### 9.9.8. BLOWN\_FUSE

This command is used to blow the fuse of the inserted card. The fuse can be EC\_EN Fuse, EC2EN Fuse, Issuer Fuse or Manufacturer's Fuse.

**Note:** The blowing of fuse is an irreversible process.

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU								
CLA	INS	Error Counter LEN	Byte Address	MEM_L	CODE			
					Fuse Bit Addr (High)	Fuse Bit Addr (Low)	State of FUS Pin	State of RST Pin
FFh	05h	00h	00h	04h			01h	00h or 01h

Where:

- Fuse Bit Addr (2 bytes)** Bit address of the fuse. Please refer to the table below for the correct value.
- State of FUS Pin** State of the FUS pin. Should always be 01h.
- State of RST Pin** State of the RST pin. Please refer to below table for the correct value.

		Fuse Bit Addr (High)	Fuse Bit Addr (Low)	State of RST Pin
AT88SC101	Manufacturer Fuse	05h	80h	01h
	EC_EN Fuse	05h	C9h	01h
	Issuer Fuse	05h	E0h	01h
AT88SC102	Manufacturer Fuse	05h	B0h	01h
	EC2EN Fuse	05h	F9h	01h
	Issuer Fuse	06h	10h	01h
AT88SC1003	Manufacturer Fuse	03h	F8h	00h
	EC2EN Fuse	03h	FCh	00h
	Issuer Fuse	03h	E0h	00h



Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

SW1	SW2

Where:

**SW1 SW2** = 90 00h if no error



## 10.0. Other Commands Access via PC\_to\_RDR\_XfrBlock

### 10.1. GET\_READER\_INFORMATION

This command returns relevant information about ACR3901T-W1 and the current operating status, such as, the firmware revision number, the maximum data length of a command and response, the supported card types, and whether a card is inserted and powered up or not.

**Note:** This command can only be used after the logical smart card reader communication has been established using the `SCardConnect( )` API. For details of `SCardConnect( )` API, please refer to PC/SC specifications.

Command Format (*abData* field in the *PC\_to\_RDR\_XfrBlock*)

Pseudo-APDU				
CLA	INS	P1	P2	Lc
FFh	09h	00h	00h	10h

Response Data Format (*abData* field in the *RDR\_to\_PC\_DataBlock*)

FIRMWARE										MAX_C	MAX_R	C_TYPE	C_SEL	C_STAT

Where:

- FIRMWARE** 10 bytes data for firmware version
- MAX\_C** The maximum number of command data bytes
- MAX\_R** The maximum number of data bytes that can be requested to be transmitted in a response
- C\_TYPE** The card types supported by the ACR3901T-W1. This data field is a bitmap with each bit representing a particular card type. A bit set to '1' means the corresponding card type is supported by the reader and can be selected with the `SELECT_CARD_TYPE` command. The bit assignment is as follows:

Byte	1					2										
card type	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0

Refer to the next section for the correspondence between these bits and the respective card types.

- C\_SEL** The currently selected card type. A value of 00h means that no card type has been selected.
- C\_STAT** Indicates whether a card is physically inserted in the reader and whether the card is powered up:
  - 00h: No card inserted
  - 01h: Card inserted, not powered up
  - 03h: Card powered up





## Appendix A. Supported Card Types

The following table summarizes the card type returned by *GET\_READER\_INFORMATION* correspond with the respective card type.

Byte	Card Type
00h	Auto-select T=0 or T=1 communication protocol
01h	I2C memory card (1, 2, 4, 8 and 16 kilobits)
02h	I2C memory card (32, 64, 128, 256, 512 and 1024 kilobits)
03h	Atmel AT88SC153 secure memory card
04h	Atmel AT88SC1608 secure memory card
05h	Infineon SLE 4418 and SLE 4428
06h	Infineon SLE 4432 and SLE 4442
07h	Infineon SLE 4406, SLE 4436 and SLE 5536
08h	Infineon SLE 4404
09h	Atmel AT88SC101, AT88SC102 and AT88SC1003
0Ch	MCU-based cards with T=0 communication protocol
0Dh	MCU-based cards with T=1 communication protocol

**Table 11:** Supported Card Types



## Appendix B. Error Codes

The following table summarizes all the error codes for ACR3901T-W1:

Error Code	Description
01h	Invalid checksum
02h	Invalid data length
03h	Invalid command format
04h	Invalid command/Unknown command ID
05h	Card operation error
06h	Authentication is required/Authentication error
07h	Low battery
08h	Authentication failed

**Table 12:** Error Code

Android is a trademark of Google Inc.  
Atmel is a registered trademark of Atmel Corporation or its subsidiaries, in the US and/or other countries.  
The Bluetooth® word, mark and logos are registered trademarks owned by Bluetooth SIG, Inc. and any use of such marks by Advanced Card Systems Ltd. is under license. Other trademarks and trade names are those of their respective owners.  
Infineon is a registered trademark of Infineon Technologies AG.  
Microsoft is a registered trademark of Microsoft Corporation in the United States and/or other countries.