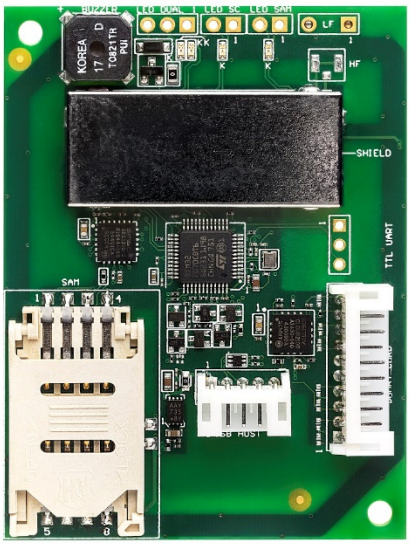
Reference Manual for

uTrust 5501F Multi-Technology Secure Smart Card Reader/Writer Module

For Part #: **905567**



Document Version: 1.0, Last Revised On: 2018-10-23

Abstract

This document contains in-depth information about the hardware and software features of the uTrust 5501 F Multi-Technology Secure Smart Card Reader/Writer Module.

Audience

This document is intended for system integrators and software developers.

Revision History

Rev.	Date	Description
1.0	2018-10-22	Initial Version

Contact Information

For additional information, please visit identiv.com.

Table of Contents

1. Legal Information

1.1. Disclaimers

The content published in this document is believed to be accurate. However, Identiv does not provide any representation or warranty regarding the accuracy or completeness of its content, or regarding the consequences of your use of the information contained herein.

Identiv reserves the right to change the content of this document without prior notice. The content of this document supersedes the content of any previous versions of the same document. This document may contain application descriptions and/or source code examples which are for illustrative purposes only. Identiv gives no representation or warranty that such descriptions or examples are suitable for any application.

Should you notice any problems with this document, please provide your feedback to support@identiv.com.

1.2. Licenses

If the document contains source code examples, they are provided for illustrative purposes only and are subject to the following restrictions:

- You MAY at your own risk use or modify the source code provided in the document in applications you may develop. You MAY distribute those applications ONLY in the form of compiled applications.
- You MAY NOT copy or distribute parts of or the entire source code without prior written consent from Identiv, Inc.
- You MAY NOT combine or distribute the source code provided with Open Source Software or with software developed using Open Source Software in a manner that subjects the source code or any portion thereof to any license obligations of such Open Source Software.

If the document contains technical drawings related to Identiv, Inc. products, they are provided for documentation purposes only. Identiv, Inc. does not grant you any license to its designs.

1.3. Trademarks

MIFARE™ is a registered trademark of NXP Semiconductors BV.

Windows is a trademark of Microsoft Corporation.

2. Introduction to the Manual

2.1. Objective of the Manual

This manual provides an overview of the hardware and software features of the uTrust 5501F Multi-Technology Secure Reader/Writer module.

This manual describes in detail interfaces and supported commands available for developers using uTrust 5501 F in their applications.

2.2. Target Audience

This document describes the technical implementation of uTrust 5501 F.

The manual targets software developers. It assumes knowledge about ISO 7816, 13.56 MHz contactless technologies like ISO/IEC 14443, ISO/IEC 15693, and 125 KHz low-frequency technologies, and commonly used engineering terms.

Should you have questions, you may send them to support@identiv.com.

2.3. Product Version Corresponding to the Manual

Product Component	Version
Hardware	1.00
Firmware	1.00

2.4. Definition of Various Terms and Acronyms

Term or Acronym	Expansion
APDU	Application protocol data unit
ATR	Answer to reset, defined in ISO7816
ATS	Answer to select, defined in ISO/IEC 14443
Byte	Group of 8 bits
CCID	Chip card interface device
CID	Card identifier
DFU	Device firmware upgrade
DR	Divider receive, used to determine the baud rate between the reader to the card
DS	Divider send, used to determine the baud rate between the card to the reader
LED	Light emitting diode
MIFARE	The ISO14443 Type A with extensions for security (NXP)
NA	Not applicable
NAD	Node address
Nibble	Group of 4 bits, 1 digit of the hexadecimal representation of a byte *Example:* 0xA3 is represented in binary as (10100011)b. The least significant nibble is 0x3 or (0011)b and the most significant nibble is 0xA or (1010)b.
PCD	Proximity coupling device
PC/SC	Personal computer/smart card, software interface to communicate between a PC and a smart card
PICC	Proximity integrated chip card
PID	Product ID
Proximity	Distance coverage through ~10 cm
PUPI	Pseudo unique PICC identifier
RF	Radio frequency
RFU	Reserved for future use
USB	Universal serial bus
VID	Vendor ID
(xyz)b	Binary notation of a number x, y, z $\in$ {0,1}
0xYY	The byte value YY is represented in hexadecimal

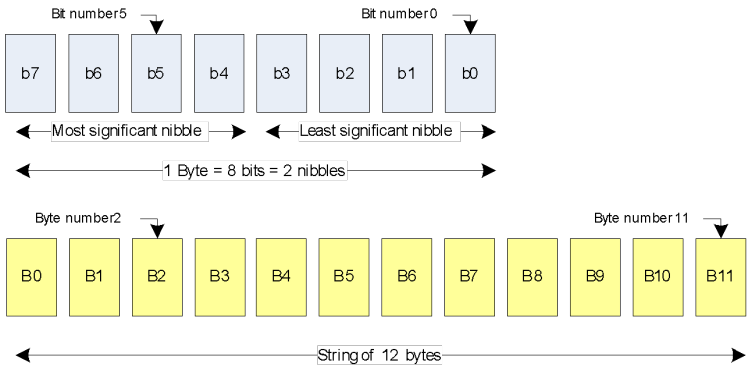
2.5. References

Document Reference in the Manual	Description of the Referenced Document	Document Issuer
ISO/IEC 7816-3	Identification cards — Integrated circuit cards —Part 3: Cards with contacts — Electrical interface and transmission protocols	ISO/IEC
ISO/IEC 7816-4	Identification cards — Integrated circuit(s) cards with contacts — Part 4: Interindustry commands for interchange ISO/IEC 7816-4: 1995 (E)	ISO/IEC
ISO/IEC 14443-3	Identification cards — Contactless integrated circuit(s) cards — Proximity cards —Part 3: Initialization and anti-collision	ISO/IEC
ISO/IEC 14443-4	Identification cards — Contactless integrated circuit(s) cards — Proximity cards — Part 4: Transmission protocol ISO/IEC 14443-4:2001(E)	ISO/IEC
LF_MANUAL	uTrust 55xx LF Reader Manual	Identiv
PC/SC	Interoperability specification for ICCs and personal computer systems v2.01	PC/SC Workgroup
PCSC3	Interoperability specification for ICCs and personal computer systems — Part 3: Requirements for PC-connected interface devices	PC/SC Workgroup
PCSC3-AMD1	Interoperability specification for ICCs and personal computer systems — Part 3: Requirements for PC-connected interface devices — Amendment 1	PC/SC Workgroup
PCSC3-SUP	Interoperability specification for ICCs and personal computer systems — Part 3: Supplemental document	PC/SC Workgroup
PCSC3-SUP2	Interoperability specification for ICCs and personal computer systems — Part 3: Supplemental document for contactless ICCs	PC/SC Workgroup
CCID	Specification for integrated circuit(s) cards interface devices 1.1	USB-IF
USB	Universal serial bus specification 2.0	USB-IF
AN337	Application note describing handling of DESFire EV1 cards	Identiv
AN338	Application note describing handling of MIFARE Plus cards	Identiv

2.6. Conventions for Bits and Bytes

Bits are represented by lower case “b” where followed by a numbering digit.

Bytes are represented by upper case “B” where followed by a numbering digit.



Example:

163 decimal number is represented

- in hexadecimal as 0xA3
- in binary as (10100011)b

The least significant nibble of 0xA3 is

- 0x3 in hexadecimal
- (0011)b in binary

The most significant nibble of xA3 is

- 0xA in hexadecimal
- (1010)b in binary

3. General Information about uTrust 5501 F

3.1. uTrust 5501 F Key Benefits

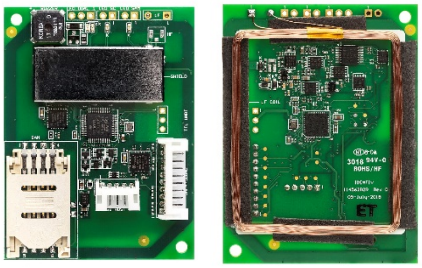
Identiv's uTrust 5501 F Smart Card Reader/Writer Module combines contact and multi-frequency/multi-ISO contactless interface capabilities to support a wide variety of identification applications. uTrust 5501 F is primarily intended for use within a card printer used in card issuance systems. The module supports contact smart card personalization (both synchronous and asynchronous), contactless high-frequency 13.56 MHz cards that are compatible with ISO14443, ISO15693, and FeLiCa, and contactless low-frequency cards that operate at 125 kHz. uTrust 5501 F allows card personalization while other information is printed on its surface.

uTrust 5501 F's optional Secure Access Module (SAM) slot provides enhanced application security via secure authentication, mutual authentication, key management, and smart card cryptographic functions. Secure communication with proprietary card technologies, such as iClass™, iClass Seos™, and more, are supported through SAM.

3.2. uTrust 5501 F Key Features

- 13.56 MHz contactless reader for ISO14443 type A and B and ISO 15693 smart cards
- 125 KHz low-frequency contactless reader/writer for ASK, FSK, and PSK cards
- ISO 7816-compliant contact smart card reader for ID-1 and ID-000 cards
- PC/SC v2.0 compliant
- Full CCID for both contact and contactless interfaces
- Secure in-field SmartOS™ firmware upgrade
- Onboard buzzer and 4 LEDs
- Unique reader serial number which enables uTrust 5501 F to be plugged into any USB slot on a PC without having to re-install the driver. Additionally, the application S/W running on the host can check for exact readers.

3.3. uTrust 5501 F Ordering Information

Item	Part Number	
uTrust 5501 F	905567	

3.4. uTrust 5501 F Customization Options

Upon request, Identiv, Inc. can consider customizing:

- Optional contact slot (ID-1 or ID-000)
- Optional LF circuitry
- Optional HF circuitry
- The USB strings

Terms and conditions apply. Please contact your local Identiv representative or send an email to sales@identiv.com.

3.5. Contactless Communication Principles and uTrust 5501 F Usage Recommendations

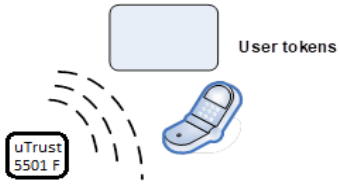
uTrust 5501 F is a dual-interface reader capable of reading both contact smart cards and contactless user tokens. The following focuses on a few specifics of contactless communication to outline usage recommendations in order to ensure the best user experience.

uTrust 5501 F is a contactless reader[1] designed to communicate with user credentials.

User credentials[2] are made of a contactless integrated circuit chip connected to an antenna.

User credentials can take several form factors:

- Credit card-sized smart card
- Key fob
- USB token
- NFC mobile device



Communication between uTrust 5501 F and credentials uses magnetic field inductive coupling.

The magnetic field generated by uTrust 5501 F has a carrier frequency of 13.56 MHz/125 KHz.

3.5.1. Power Supply

When the user credential is placed in the magnetic field of the reader, its antenna couples with the reader and an induction current appears in the antenna, thus providing power to the integrated circuit. The generated current is proportional to the magnetic flux going through the antenna of the user credential.

3.5.2. Data Exchange

The carrier frequency of the magnetic field is used as a fundamental clock signal for the communication between the reader and the credential. It is also used as a fundamental clock input in order for the integrated circuit microprocessor to function.

[1] In the ISO/IEC 14443 standard, the reader is called the proximity coupling device (PCD).

[2] In the ISO/IEC 14443 standard, the user credential is called proximity integrated chip card (PICC).

To send data to the user credential, the reader modulates the amplitude of the field. There are several amplitude modulation and data encoding rules defined in ISO/IEC 14443. The reader should refer to the standard for further details.

To answer the reader, the integrated circuit card of the user credential modulates its way of loading (impedance) the field generated by the reader. Additional details can be found in ISO/IEC 14443.

3.5.3. Recommendations

The communication between the reader and the user credential is sensitive to the presence of materials or objects interfering with the magnetic field generated by the reader.

The presence of conductive materials, like metal, in the vicinity of the reader and the user credential can significantly degrade the communication and even make it impossible. The magnetic field of the reader generates Eddy or Foucault's currents in the conductive materials; the field is literally absorbed by that kind of material.

- It is recommended for proper communication to avoid putting uTrust 5501 F in close proximity of conductive materials.

The presence of multiple user credentials in the field also interferes with communication. When several user credentials are in the field of the reader, load of the field increases, which implies that less energy is available for each of them and that the system is detuned. For this reason, Identiv has implemented only one slot in its driver.

- It is recommended to present only one user credential at a time to uTrust 5501 F.

The communication between the reader and the credential is sensitive to the geometry of the system (reader and credential). Parameters, like the geometry and especially the relative size of the reader's and credential's antennas, directly influence the inductive coupling and, therefore, the communication.

uTrust 5501 F was designed and optimized to function with user credentials of various technologies and sizes.

- It may happen that uTrust 5501 F is not capable of communicating with extremely large or extremely small credentials.
- In order to optimize the coupling between the reader and the credential, it is recommended to put both antennas as parallel as possible to each other.
- In order to optimize transaction speed between the reader and the card, it is recommended to place the credential as close as possible to the reader. This will increase the amount of energy supplied to the user credential which will then be able to use its microprocessor at higher speeds.

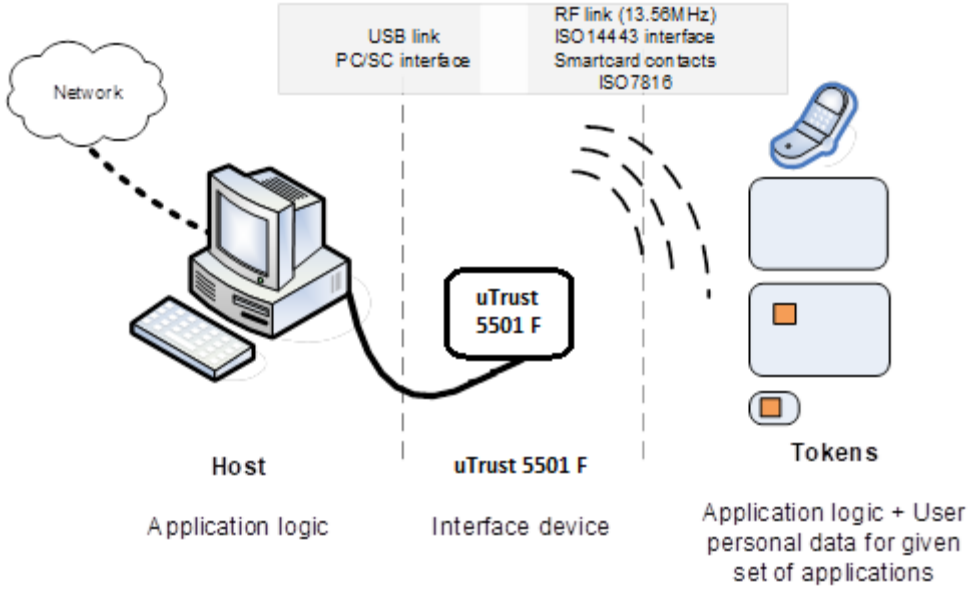
3.6. Applications

3.6.1. General

uTrust 5501 F is a transparent reader/module designed to interface a personal computer host supporting PC/SC interface with 13.56 MHz/125 KHz user tokens, like public transport cards, contactless banking cards, electronic identification documents (e.g., e-passports, e-ID cards, drivers licenses, etc.), ISO 7816 smart cards such as CAC and PKI, health insurance cards, and synchronous memory.

User credentials can have several form factors, including credit cards, key fobs, NFC mobile devices, or USB dongles such as Identiv's uTrust Token family.

uTrust 5501 F incorporates a SAM slot for SIM-sized.



uTrust 5501 F handles the communication protocol with the card but not with the application related to the token or card. The application-specific logic has to be implemented by software developers on the host.

3.6.2. Applications Provided by Identiv, Inc.

Identiv, Inc. does not provide payment, transport, PKI, or CAC applications.

Identiv, Inc. provides a few applications for development and evaluation purposes that can function with uTrust 5501 F. There are many tools provided, including:

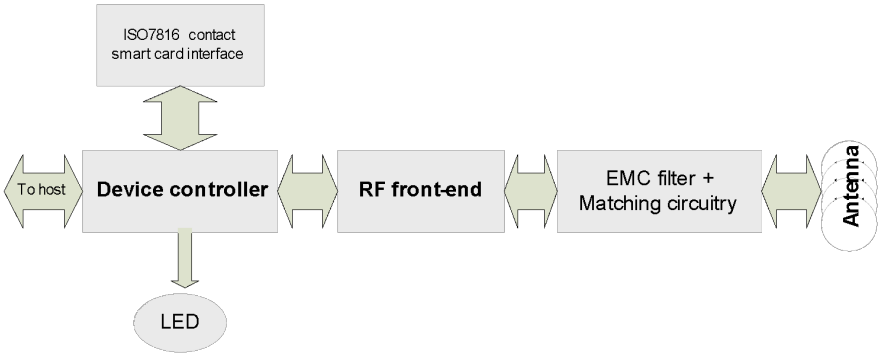
- **The Simple NFC Tag Editor**: Part of Identiv's NFC-NDEF Editor Kit that enables the user to read and write NFC Forum-compliant records from/to NFC Forum-compatible tags. It is an easy-to-use tool to configure NFC Forum tag demonstrations rapidly, available at identiv.com.
- **Smart Card Commander Version 1.3**: Provides capabilities to identify most common cards in the field and display the content and scripting functionality of such cards. This can be very useful for developers in developing and debugging applications. This tool is a part of all Identiv SDKs and is available as a stand-alone product.

4. uTrust 5501 F Characteristics

4.1. uTrust 5501 F High-Level Architecture

4.1.1. Block Diagram

The link between uTrust 5501 F and the host to which it is connected is the USB interface providing both the power and the communication channel.



The device controller has several interfaces available. In the uTrust 5501 F implementation, peripherals are connected to the device controller:

- LED for reader status indication
- Buzzer
- A contact smart card interface
- An RF front-end that handles the RF communication

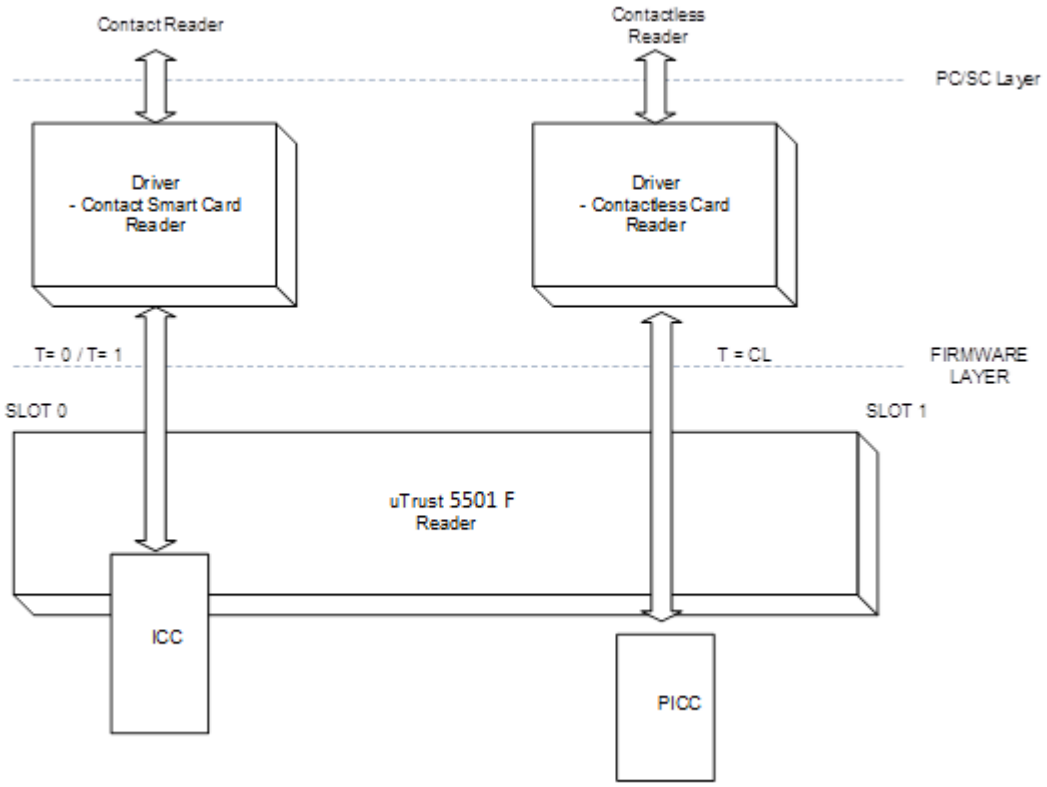
The controller embeds flash memory that contains the firmware developed by Identiv to handle all ISO 7816 contact protocols, RF communication protocols, and the PC/SC communication protocol with the host. The flash can be upgraded once the device is deployed in the field, hence enabling firmware upgrades to add and potentially patch features.

The RF front-end ensures the coding/decoding/framing modulation/demodulation required for RF communication. It is controlled by the device controller through registers.

The matching circuitry provides the transmission and receiver paths adaptation for the antenna to function properly.

4.1.2. Software Architecture

Applications can interface with the driver directly through the PC/SC interface.



uTrust 5501 F leverages a PC/SC CCID driver that is freely available for all supported operating systems (Windows, macOS X, and Linux). With current Windows versions (starting with Windows Vista) and macOS X, this driver is already included in the basic installation.

With the diverse Linux derivatives, there may be distribution-specific drivers that must be installed using the install mechanism of the used distribution.

Please search the web for PC/SC-lite or go to support.identiv.com and navigate to the downloads page for your reader.

4.2. Quick Reference Data

4.2.1. LED Behavior

uTrust 5501 F is equipped with four LEDs. Its current behavior with latest firmware is described in the table below. LED 4 is not implemented in the firmware at this time.

Reader States	LED 1 – Dual GREEN	LED 2 – Dual RED	LED 3 – Smart Card GREEN	Remarks
Just after plug-in	ON	ON	OFF	
During DFU operation	OFF	OFF	OFF	
Just after DFU operation	ON	ON	OFF	
PC sleep/standby	OFF	OFF	OFF	
Reader powered, contact card IN, but not powered	ON	ON	ON-OFF	LED 3 will be ON for 10 seconds, then time-out occurs, and LED will turn OFF
Reader powered, contactless card IN, but not powered	ON	ON	OFF	
Contact card powered/communication	ON	ON	500 msec ON/500 msec OFF	Blinks ON and OFF for 500 milliseconds
Contactless card powered/communication	ON	ON	OFF	LED 1/LED 2 stays ON but does not blink
Reader/card errors	ON	ON	100 msec ON/100 msec OFF	Only LED 3 will blink rapidly when contact card error occurs or card was inserted in reverse
Combi card powered in contact slot	ON	ON	500 msec ON/500 msec OFF	
Combi card powered using RF field	ON	ON	OFF	

4.2.2. Other Data

4.2.2.1. General

Parameter	Value/Description
Clock of the device controller	8 MHz
API	PC/SC 2.0
Operating temperature range	0º to 50ºC
Operating humidity range	Up to 90 %, RH non-condensing
Certifications and compliances	CE, FCC part 15 Subpart C, certified as module, RoHS2, REACH, and WEEE

4.2.2.2. USB

Parameter	Value/Description
DC characteristics	High bus powered (uTrust 5501 F draws power from USB bus), 5V voltage, and average current of 200 mA (RF ON, no cards present) and 270 mA (RF ON, with a CL card)
USB specification	USB 2.0 CCID (USB 1.1/3.0 compliant), 12 Mbps (full speed)
Device Class	CCID
PID	0x5610 for CL and SAM interface, 0x5810 for Contact interface
VID	0x04E6

4.2.2.3. Contact Card and SAM Interface

Parameter	Value/Description
Smart card operating frequency	Up to 12MHz
Maximum supported card baud-rate	600 Kbps
Cards supported	ISO/IEC 7816 smart cards, class A, B and C, synchronous smart cards
ISO-7816 compliant	Yes
CT-API compliant	Yes
Number of slots	Single smart card slot and single SAM slot
Ejection mechanism	Manual for smart card interface and push-pull type for SAM

4.2.2.4. Contactless Interface

Parameter	Value/Description
RF carrier frequency	13.56MHz +/-50ppm
Maximum supported card baud-rate	848 Kbps
Cards supported	• MIFARE Classic 1K and 4K, DESFire, DESFire EV1, Ultralight, Ultralight C, MIFARE mini, and MIFARE Plus • FeliCa™ 212 and 424 Kbps support, FeliCa Standard/Lite • NFC Forum tag type 1, 2, 3, 4 • iCLASS UID support • my-d move – SLE 66RxxP, my-d move NFC – SLE 66RxxPN, SLE 66RxxS, SLE 55RxxE • NFC-enabled smart phones and tablets[3]
ISO-14443A and B compliant	Yes
Number of slots	1
Ejection mechanism	Manual

[3] Tested with available device during development and qualification phase.

5. Software Modules

5.1. Installation

On operating systems with a CCID driver pre-installed, no installation is necessary.

If a CCID driver is not pre-installed (e.g., Linux systems or old Windows systems), the driver must be installed using distribution-specific measures or the available source packages.

Nevertheless, due to some limitations of the available CCID drivers under some circumstances, Identiv does provide a dedicated driver for the reader, which is available through Windows Update or at support.identiv.com. An installer bundled with the signed driver is also provided.

5.2. Utilities

The following utilities are available:

- A tool for testing the resource manager
- A tool called *PCSCDiag* capable of providing basic information about the reader and a card through PC/SC stack

5.3. Driver

5.3.1. uTrust 5501 F Listing

uTrust 5501 F is listed by PC/SC applications as:

- *Identiv uTrust 5501 R Smart Card Reader*
- *Identiv uTrust 5501 Contactless Reader*
- *Identiv uTrust 5501 SAM Reader*

5.3.2. Supported Operating Systems

- Windows Server 2012 RTM and R2, 2016
- Windows 7 (32 and 64 bit)
- Windows 8.1 (32 and 64 bit)
- Windows 10 (32 and 64 bit)
- macOS 10.12, 10.13
- Linux kernel 3.x (32 and 64 bit)

5.3.3. PC/SC 2.0-Compliant ATR for Contactless Interface

When a user credential is placed on the reader, initialization, anti-collision is done. The user credential is automatically activated and an ATR is built as defined in the PC/SC specification. For further information, please refer to section 3.1.3.2.3 of [PCSC3] and [PCSC3-SUP].

5.3.3.1. ATR for Contactless Storage User Tokens

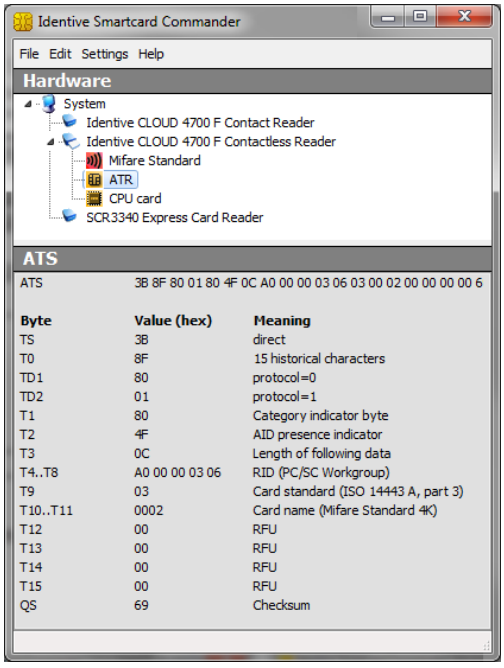
The ATR of the credential is as described in the table below. In order to allow the application to identify the storage card properly, it's standard and card name describing bytes must be interpreted according to the Part 3 Supplemental Document maintained by PC/SC.

Examples include credentials using technology like MIFARE.

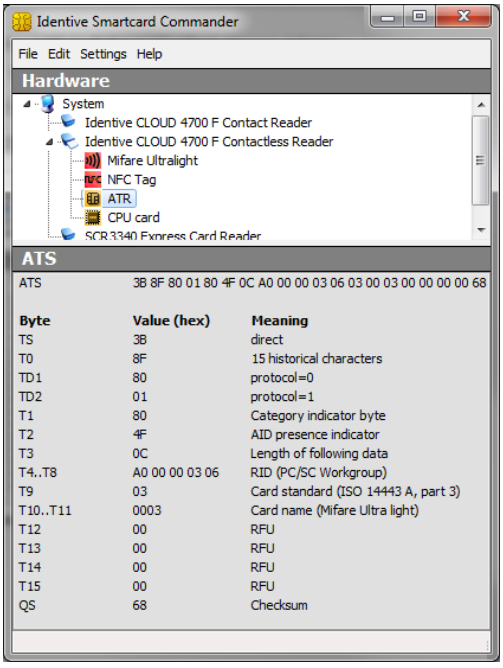
Byte#	Value	Designation	Description
0	0x3B	Initial header	
1	0x8n	T0	n indicates the number of historical bytes in following ATR
2	0x80	TD1	upper nibble 8 indicates no TA2, TB2, TC2 lower nibble 0 means T=0
3	0x01	TD2	upper nibble 0 indicates no TA3, TB3, TC3 lower nibble 1 means T=1
4...3+n	0x80		A status indicator may be present in an optional TLV data object
	0x4F	Optional TLV data object	Tag: Application identifier
	Length		1 byte
	RID		Registered identifier on 5 bytes
	PIX		Proprietary identifier extension on 3 bytes
	0x00 0x000x000x00		4 RFU bytes
4+n		TCK	XOR of all previous bytes

Example of the ATR built for contactless storage tokens:

MIFARE 4K



MIFARE Ultralight



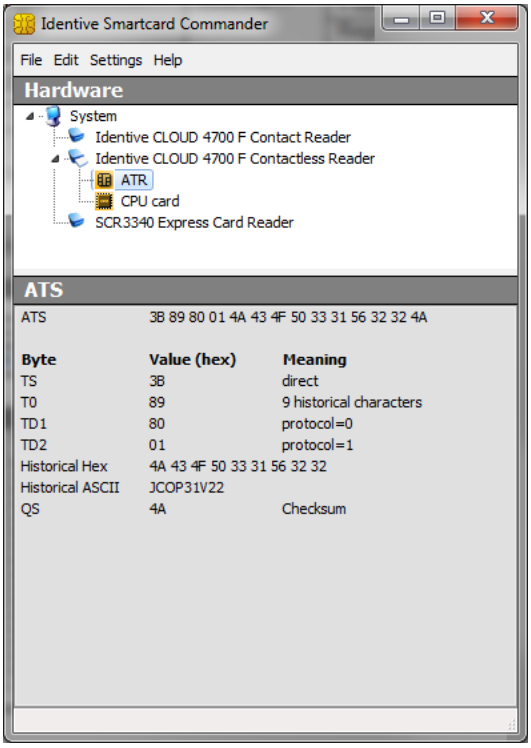
5.3.3.2. ATR for ISO/IEC 14443-4 User Tokens

The credential exposes its ATS or application information which is mapped to an ATR. The table describes how this mapping is done.

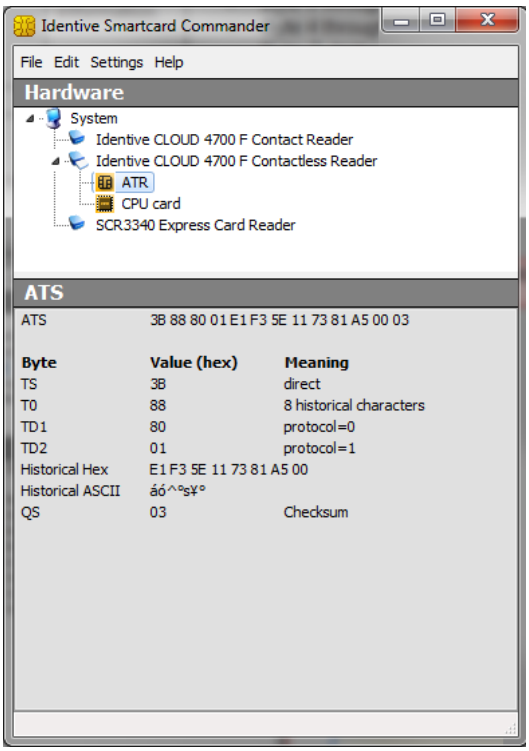
Byte#	Value	Designation	Description
0	0x3B	Initial header	
1	0x8n	T0	n indicates the number of historical bytes in following ATR
2	0x80	TD1	upper nibble 8 indicates no TA2, TB2, TC2 lower nibble 0 means T=0
3	0x01	TD2	upper nibble 0 indicates no TA3, TB3, TC3 lower nibble 1 means T=1
4...3+n		Historical bytes or application information	Type A: The historical bytes from the ATS (up to 15 bytes) Type B (8 bytes): • Byte 0 through 3: application data from ATQB, • Byte 4 through 6: protocol info byte from ATQB, • Byte 7: highest nibble is the MBLI (maximum buffer length index) from ATTRIB, lowest nibble is 0x0
4+n		TCK	XOR of all previous bytes

Example of the ATR built for an ISO 14443-4 credential:

Type A



Type B



5.4. Firmware

5.4.1. CCID Transport Protocol

uTrust 5501 F implements a transport protocol that is compliant with USB Device Class: *Smart Card CCID Specification for Integrated Circuit(s) Cards Interface Devices Revision 1.10*.

This following describes the CCID specification features that are implemented.

5.4.1.1. CCID Class Requests Supported

- Abort

5.4.1.2. CCID Messages Supported

The following CCID messages are supported for the contact interface when received through bulk-out endpoint:

- PC_to_RDR_IccPowerOn
- PC_to_RDR_IccPowerOff
- PC_to_RDR_GetSlotStatus
- PC_to_RDR_XfrBlock
- PC_to_RDR_GetParameters
- PC_to_RDR_ResetParameters
- PC_to_RDR_SetParameters
- PC_to_RDR_Escape
- PC_to_RDR_ICCClock
- PC_to_RDR_T0APDU
- PC_to_RDR_Abort
- PC_to_RDR_SetDataRateAndClockFrequency

5.4.1.3. CCID Error Codes

Extensive error codes are reported on many conditions during all CCID responses. Most of the error messages are reported by the CCID appropriately. Some of the main error codes for the contact interface include:

- HW_ERROR
- XFR_PARITY_ERROR
- ICC_PROTOCOL_NOT_SUPPORTED
- BAD_ATR_TS
- BAD_ATR_TCK
- ICC_MUTE
- CMD_ABORTED
- Command not supported

The following sub-sections discuss when and why these error codes are returned.

5.4.1.3.1. HW_ERROR

This error code is returned when a hardware short circuit condition is detected during application of power to the card or if any other internal hardware error is detected.

5.4.1.3.2. XFR_PARITY_ERROR

This error code is returned when a parity error condition is detected. This error will be reported in the response to a PC_to_RDR_XfrBlock message.

5.4.1.3.3. ICC_PROTOCOL_NOT_SUPPORTED

This error code is returned if the card is signalling to use a protocol other than T=0 or T=1 in its ATR.

5.4.1.3.4. BAD_ATR_TS

This error code is returned if the initial character of the ATR contains invalid data.

5.4.1.3.5. BAD_ATR_TCK

This error code is returned if the check character of the ATR contains is invalid.

5.4.1.3.6. ICC_MUTE

This error code is returned when the card does not respond until the reader time out occurs. This error will be reported in the response to PC_to_RDR_XfrBlock message and PC_to_RDR_IccPowerOn messages.

5.4.1.3.7. CMD_ABORTED

This error code is returned if the command issued has been aborted by the control pipe.

5.4.1.3.8. Command not supported

This error is returned if the command is not supported by the reader.

6. Commands Description

6.1. Generic APDU

6.1.1. Working with DESFire and MIFARE Plus Tokens

To work with DESFire EV1 and MIFARE Plus tokens, please refer to the application notes [AN337] and [AN338], respectively.

Please note that since these application notes contain information available only under NDA with NXP, you will need to sign an NDA with NXP to in order to receive them.

6.1.2. PAPDU_GET_UID

Get UID will retrieve the UID, SNR, or PUPI of the user token. This command can be used for all supported technologies.

Command APDU:

CLA	INS	P1	P2	Lc	Data In	Le
0xFF	0xCA	0x00	0x00	-	-	XX

Setting Le = 0x00 can be used to request the full UID or PUPI is sent back (e.g., for ISO 14443A single 4 bytes, double 7 bytes, triple 10 bytes, or ISO 14443B 4 bytes PUPI).

Response APDU:

Data	Status Word
Requested bytes of UID	SW1, SW2

6.1.3. PAPDU_ESCAPE_CMD

Typically escape commands are transmitted through SCardControl as defined in PCSC API using IOCTL_CCID_ESCAPE. In some environments, the driver will block this IOCTL unless the registry has been edited to allow it. Hence, this vendor-specific APDU was defined to transmit escape commands to the reader, as below.

Command APDU:

CLA	INS	P1	P2	Lc	Data In	Le
0xFF	0xCC	0x00	0x00	Length of data	Escape command buffer	XX

Response APDU:

Data	Status Word
Reader response	SW1, SW2

Example:

1) To issue the “READER_GETIFDTYPE (0x12)” escape command , this pseudo APDU would be used:

   Command APDU: FF CC 00 00 01 12
   Response APDU: 20 57 90 00

2) To issue the “READER_SETMODE (0X01)” escape command, this pseudo APDU would be used:

   Command APDU: FF CC 00 00 02 01 01 (to set to EMV mode)
   Response APDU: 90 00

Note:

1) To send escape commands using this method, the reader should be connected in shared mode using T0 or T1 protocol. Only then will the resource manager allow SCardTransmit.
2) As the escape commands defined using “READER_GENERIC_ESCAPE” have ISO 7816 APDU format, they can be sent using SCardTransmit without needing to prepend “FF CC 00 00 P3”.

6.2. Supported Pseudo APDU (Contactless Interface)

All pseudo APDUs specific to contactless interfaces supported in the reader are explained in this section.

6.2.1. PAPDU_MIFARE_READ_BINARY

This command is used to read data from a MIFARE card. Refer to section 3.2.2.1.8 of [PCSC3] for details.

Command APDU:

Command	CLA	INS	P1	P2	Lc	Data	Le
Read binary	0xFF	0xB0	Addr MSB	Addr LSB	-	-	xx

P1 and P2 represent the block number of the block to be read, starting with 0 for sector 0, block 0, continuing with 4 for sector 1, block 0 (sector no. x 4 + block no.)

Regardless of the value given in Le, this command will always return the entire block content:

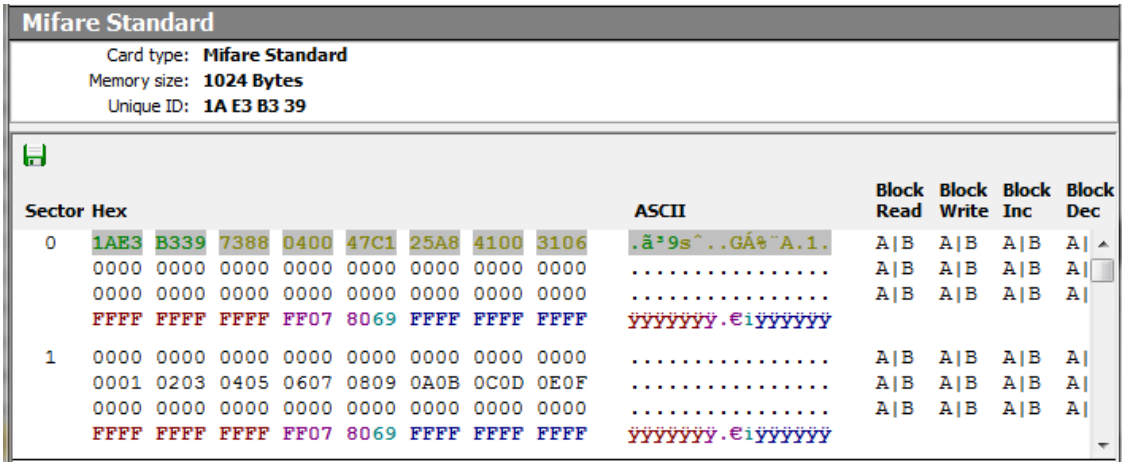
16 bytes for MIFARE Classic
4 bytes for MIFARE UL and UL C

Response APDU:

Data	Status Word
N bytes of block data	SW1, SW2

Example:

For a MIFARE Classic 1K card with the following content:



The following command will read the sixth block and yield the mentioned output:

```
APDU: FF B0 00 05 02
SW12: 9000 (OK)
DataOut: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F (16 bytes)
```

6.2.2. PAPDU_MIFARE_UPDATE_BINARY

This command is used to update the non-volatile memory of a MIFARE card. Refer to section 3.2.2.1.9 of [PCSC3] for further details.

Command APDU:

Command	CLA	INS	P1	P2	Lc	Data	Le
Update Binary	0xFF	0xD6	Addr MSB	Addr LSB	xx	data	-

For a description of P1 and P2, see PAPDU_MIFARE_READ_BINARY.

- Lc must match the block size of the used card
- 16 bytes for MIFARE Classic
- 4 bytes for MIFARE UL and UL C

Response APDU:

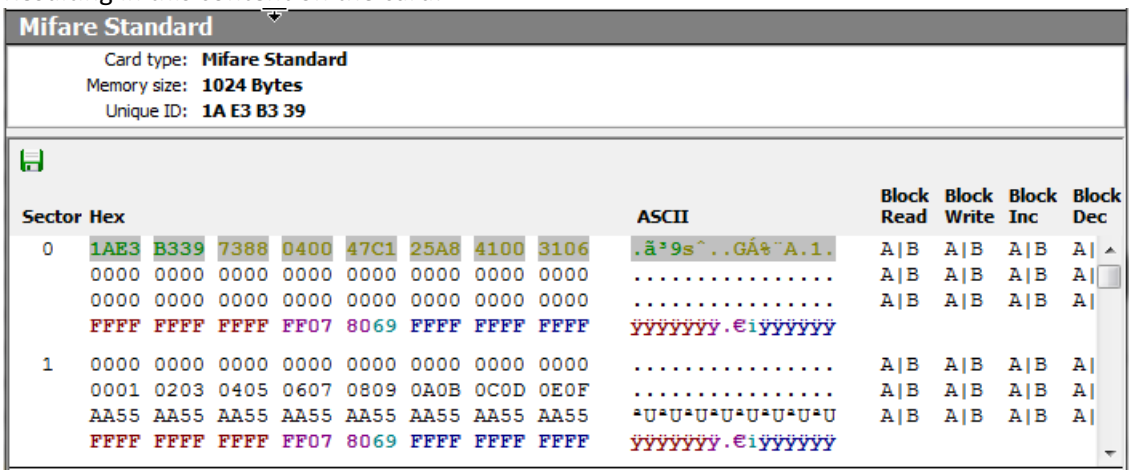
Data	Status Word
-	SW1, SW2

Example:

To write the bytes `AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55` to block 7 of a MIFARE Classic 1K, the following command must be issued:

```
APDU: FF D6 00 06 10 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55
SW12: 9000 (OK)
```

Resulting in this content on the card:



6.2.3. PAPDU_MIFARE_LOAD_KEYS

This command is used to load the key to the volatile memory of the reader. It can be used for all types of contactless cards. Refer to section 3.2.2.1.4 of [PCSC3] for further details.

Command APDU:

Command	CLA	INS	P1	P2	Lc	Data	Le
Load Keys	0xFF	0x82	Key Struct	Key Num	Key data	Key	-

The Key Structure (P1) is defined as follows:

b7	b6	b5	b4	b3	b2	b1	b0	Description
x								0: Card Key; 1 Reader Key
	x							0: Plain Transmission, 1: Secured Transmission
		x						0: Keys are loaded into the volatile memory 1: Keys are loaded into the non-volatile memory.
			x					RFU
				xxxx				If b6 is set, it is the Reader Key number that has been used for the encryption, else it is ignored. Only one reader key (0x00) is supported by uTrust 5501 F

Notes:

1) Card keys can be loaded in both “secure” and “non-secure” mode. Card keys can only be loaded to the volatile memory of the reader.
2) To load card keys in secure mode, the application developer must know the 128 bit AES key of the reader. The default key is “00010203 05060708 0A0B0C0D 0F101112”.
   As a MIFARE key is only 6 bytes in length, data must be padded as per pkcs7 padding scheme (see example below).
3) The reader-key can only be loaded in secure-mode to the non-volatile memory of the reader. The new key is first XORed with the old key and encrypted with the old key. In order to validate the integrity of the processed key data, a 2 byte CRC must be sent following the key data. Refer to the example Load Keys – Reader – Secure for details.
4) The CRC16 is calculated as defined in CRC-16-CCITT (polynomial 0x8408) with an initial value of 0x0000.

Response APDU:

Data	Status Word
-	SW1, SW2

Examples

Load Keys – Card – Non-Secure:

The command to load MIFARE key A "FF FFFFFFFFFF" is
FF82006006 FFFFFFFFFFFF

Load Keys – Card – Secure:

If the default AES128 reader is key is 00010203 05060708 0A0B0C0D 0F101112, then the following explains the steps needed to calculate the key for secure mode.

```
Default reader key          : 00010203 05060708 0A0B0C0D 0F101112
Mifare Key to be loaded     : FFFFFFFF FFFF
Mifare key after padding    : FFFFFFFF FFFF0A0A 0A0A0A0A 0A0A0A0A
AES128 Encrypted            : 10229E33 189403FD A9C14110 B1BB02B4
Load keys command           :FF82406010 10229E33 189403FD A9C14110 B1BB02B4
```

Load Keys – Reader – Secure:

If the default AES128 reader is key is 00010203 05060708 0A0B0C0D 0F101112, then the following explains the steps needed to change the reader key to 10111213 15161718 1A1B1C1D 1F202122.

```
Reader old-key          : A: 00010203 05060708 0A0B0C0D 0F101112
Reader new-key          : B: 10111213 15161718 1A1B1C1D 1F202122
C = XOR (A,B)           : C: 10101010 10101010 10101010 10303030
D = CRC16(C)            : D: 1C5F
E = 0x00 - D            : E: E3A1 (should be appended in LSB order)
F = AES-Encrypt (C)     : F: 886B0872 7BDA4996 D296FB46 09D2C75F
Load-Keys Command       : G: FF82E00012 886B0872 7BDA4996 D296FB46 09D2C75F A1E3
```

6.2.4. PAPDU_MIFARE_AUTHENTICATE

This command is used to authenticate using the key number. Refer to section 3.2.2.1.6 of [PCSC3] for further details.

Command APDU:

Command	CLA	INS	P1	P2	Lc	Data	Le
General Authenticate	0xFF	0x86	0x00	0x00	0x05	data	xx

The data structure is defined as follows:

Byte #	Value	Description
B0	0x01	Version
B1		Block Number MSB (always 0x00 for MIFARE Classic cards)
B2		Block Number LSB
B3	0x60	MIFARE Classic Key A
	0x61	MIFARE Classic Key B
B4		Key number – shall be set to 0x01

Response APDU:

Data	Status Word
-	SW1, SW2

Example:

Load Key A unencrypted and authenticate for block 6 (sector 1, actually) with that key:

```
APDU: FF 82 00 60 06 FF FF FF FF FF FF
SW12: 9000 (OK)
APDU: FF 86 00 00 05 01 00 06 60 01
SW12: 9000 (OK)
```

6.2.5. PAPDU_MIFARE_READ_SECTOR

This command reads the specified sector from a MIFARE Classic card (first 3 blocks of the sector, excluding the key block) or the entire content of MIFARE UL/UL C cards.

Command APDU:

Command	CLA	INS	P1	P2	P3	Data
Read sector	FF	B1	Addr MSB	Addr LSB	0	-

Response APDU:

Data	Status Word
MIFARE Classic: 48 bytes of sector data read from card MIFARE UL: Entire card data is returned (64 bytes)	SW1, SW2

Example:

Read sector 1 of a MIFARE Classic 1K

```
APDU: FF B1 00 01 00
SW12: 9000 (OK)
DataOut: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 (48 bytes)
```

Read entire content of a MIFARE UL:

```
APDU: FF B1 00 01 10
SW12: 9000 (OK)
DataOut: 04 6B 5D BA 09 F8 01 80 70 48 00 00 E1 10 06 00
00 01 02 03 1D 6E 6F 6B 69 61 2E 63 6F 6D 3A 62
74 01 00 11 67 9F 5F B6 04 06 80 30 30 30 30 00
00 00 00 00 00 00 00 00 00 00 02 42 54 FE 00 (64 bytes)
```

6.2.6. PAPDU_MIFARE_READ_SECTOR_EX

This command reads the specified sector from a MIFARE Classic card (all the 4 blocks of the sector, including the key block) or the entire content of MIFARE UL/UL C cards.

Command APDU:

Command	CLA	INS	P1	P2	P3	Data
Read sector extended	FF	B3	Addr MSB	Addr LSB	0	-

Response APDU:

Data	Status Word
MIFARE Classic: 64 bytes of sector data read from card MIFARE UL: Entire card data is returned (64 bytes)	SW1, SW2

Example:

Read sector 1 of a MIFARE Classic 1K

```
APDU: FF B3 00 01 10
SW12: 9000 (OK)
DataOut: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55
00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF (64 bytes)
```

6.2.7. PAPDU_MIFARE_WRITE_SECTOR

This command writes the contained data to the specified sector of a MIFARE Classic or MIFARE UL/UL C card (first blocks of the sector, excluding the key block, are written per MIFARE Classic).

Command APDU:

Command	CLA	INS	P1	P2	P3	Data
Write sector	FF	D7	AddrMsb	AddrLsb	Lc	Data

Lc (P3) must be 0x30 when writing to the small sectors of a MIFARE Classic and 0xF0 when writing to the large sectors of a MIFARE Classic 4K.
Lc must be 0x30 for MIFARE UL; the data will be written from block 4 through the end of the memory.

Response APDU:

Data	Status Word
-	SW1, SW2

6.2.8. PAPDU_MIFARE_VALUE_BLK_OLD

This command increments or decrements the data in a value block on a MIFARE Classic card.

Command APDU:

P2 codes the block number:

Command	CLA	INS	P1	P2	P3	Data
Increment/decrement OLD	FF	F0	00	Block Num	Lc	Data

The data field is structured as follows:

Byte #	Value	Description
B0	0xC0	Decrement
	0xC1	Increment
B1		Block number
B2-B5		Value (LSB first)

Response APDU:

Data	Status Word
-	SW1, SW2

Example:
Decrement block 4 by 1 (key loading and authentication not shown). Block 4 must be set up as a value block prior to executing this command. See datasheet for MIFARE Classic cards.

```
APDU: FF B0 00 04 00 // Read Block 4
SW12: 9000 (OK)
DataOut: A9 AA AA AA 56 55 55 55 A9 AA AA AA 05 FA 05 FA (16 bytes)

APDU: FF F0 00 04 06 C0 04 01 00 00 00 // decrement block 4 by 1
SW12: 9000 (OK)

APDU: FF B0 00 04 00 // Read Block 4
SW12: 9000 (OK)
DataOut: A8 AA AA AA 57 55 55 55 A8 AA AA AA 05 FA 05 FA (16 bytes)
```

6.2.9. PAPDU_MIFARE_VALUE_BLK_NEW

This command increments or decrements the value of a data object if the card supports it. Refer to section 3.2.2.1.10 of [PCSC3-AMD1] for further details.

Command APDU:

Command	CLA	INS	P1	P2	Lc	Data	Le
Increment/ Decrement	FF	C2	00	03	xx	BER-TLV	00

The data object consists of a TLV structure that defines which action should be performed, which block the actions pertain to (the destination(s)), and which value should be applied for the action.

Tags for the action include:
0xA0: Increment
0xA1: Decrement

The tag to define the destination is:
0x80: Destination

The tag to define the value is:
0x81: Value by which to increment or decrement the destination, LSB first

Example:

Increment block 5 by 100

```
FF C2 00 03 0B
A0 09                    increment
80 01 05              block 5
81 04 64 00 00 00 by 100
                  00
```

This command returns a response APDU according to section 2.2 of [PCSC3-SUP2].

Response APDU:

Data	Status Word
C0 03 Error status, see below	SW1, SW2 (card itself will send SW1, SW2)

Error Status	Description
XX SW1 SW2	XX = Number of the bad data object in the APDU 00 = General error of APDU 01 = Error in the 1st data object 02 = Error in the 2nd data object, etc.
00 90 00	No error occurred
XX 62 82	Data object XX warning, requested information not available
XX 63 00	No information
XX 63 01	Execution stopped due to failure in other data object
XX 6A 81	Data object XX not supported
XX 67 00	Data object XX with unexpected length
XX 6A 80	Data object XX with unexpected vale
XX 64 00	Data Object XX execution error (no response from IFD)
XX 64 01	Data Object XX execution error (no response from ICC)
XX 6F 00	Data object XX failed, no precise diagnosis

6.2.10. PAPDU_TCL_PASS_THRU (T=CL Pass Thru)

This command can be used to send raw data using T=CL protocol to a card. Please refer to the status words defined by the PICC manufacturer for a description of the status words

Command APDU:

Command	CLA	INS	P1	P2	P3	Data
Pass-through	FF	FE	00	00	Lc	Data

Response APDU:

Data	Status Word
PICC response data	SW1, SW2 (card itself will send SW1, SW2)

6.2.11. PAPDU_ISO14443_PART3_PASS_THRU (MIFARE Pass Thru)

This command is used to send raw data using type A standard framing to a card. CRC bytes will be appended automatically. The reader will not add transport protocol data to the raw data (e.g., PCB, NAD, CID, etc.)

Command APDU:

Command	CLA	INS	P1	P2	P3	Data
Part 3 Pass-through	FF	EF	Transmit CRC	00	Lc	Data

P1 = 0x00 will transmit the CRC bytes form the card as is to the application
P1 = 0x01 will discard the CRC bytes

Response APDU:

Data	Status Word
Data returned by card	SW1, SW2

6.2.12. PAPDU_ISO14443_PART4_PART3_SWITCH (TCL – Mifare Switch)

This command switches the card state between TCL and MIFARE modes.

Command APDU:

Command	CLA	INS	P1	P2	P3	Data
Part 4-Part 3 Switch	FF	F8	P1	00	00	-

P1 = 0x00 switches from MIFARE mode to TCL mode
P1 = 0x01 switches from TCL mode to MIFARE mode

Response APDU:

Data	Status Word
-	SW1, SW2

NOTE: This command is mainly targeted at MIFARE plus S0 cards. MIFARE plus cards at S0 level will be detected as MIFARE memory cards. In order to personalize these cards, they must first be switched to Part 4 mode. For this purpose, the user command needs to be issued using the SCardTransmit function.

6.2.13. PAPDU_FELICA_REQC

This command Issues REQC as defined in JIS 7.5.1. It is used to detect the presence of an NFC Forum tag type 3 in the field

Command APDU:

Command	CLA	INS	P1	P2	P3	Data
FeliCa REQC	FF	40	00	00	04	2 bytes of system code, 1 byte RFU, 1 byte TSN

Response APDU:

Data	Status Word

16 bytes of NFCID2 + 2 bytes of system code (sent only if the RFU byte is 0x01)	SW1, SW2

6.2.14. PAPDU_FELICA_REQ_SERVICE

This command issues a REQ SERVICE as defined in JIS 9.6.2. P1. On receiving this command, an NFC Forum tag type 3 will respond with the area key version of the specified area and the service key version of the specified service.

Command APDU:

Command	CLA	INS	P1	P2	P3	Data
FeliCa REQ Service	FF	42	Number of services/areas	00	2 * P1	Service code list/ Area code list

Response APDU:

Data	Status Word
8 bytes IDm + number of service or areas(n) + service version or area version list (2*n)	SW1, SW2

6.2.15. PAPDU_FELICA_REQ_RESPONSE

This command issues a REQ RESPONSE as defined in JIS 9.6.1. When an NFC Forum tag type 3 receives this command, it responds with its current mode (0/1/2).

Command APDU:

Command	CLA	INS	P1	P2	P3	Data
FeliCa REQ Response	FF	44	00	00	00	-

Response APDU:

Data	Status Word
8 bytes IDm + Mode	SW1, SW2

6.2.16. PAPDU_FELICA_READ_BLK

This command issues a READ as defined in JIS 9.6.3:

- P1 specifies the number of service
- P2 specifies the number of blocks
- Data buffer specifies the service code and block list

When an NFC Forum tag type 3 receives this command, it responds with the record value of the specified service.

Command APDU:

Command	CLA	INS	P1	P2	P3	Data
FeliCa REQ Response	FF	46	Number of service	Number of blocks	2*(P1 + P2)	Service Code List, Block List

Response APDU:

Data	Status Word
8 bytes IDm + Status Flag 1 + Status Flag 2 + No. of blocks(n) + Block data (n*16)	SW1, SW2

6.2.17. PAPDU_FELICA_WRITE_BLK

This command issues a WRITE as defined in JIS 9.6.4:

- P1 specifies the number of service
- P2 specifies the number of blocks

When an NFC Forum tag type 3 receives this command, it writes the records of the specified service.

Command APDU:

Command	CLA	INS	P1	P2	P3	Data
FeliCa Write Block	0xFF	0x48	Number of service	Number of blocks	2*(P1 + P2) + (16 * P2)	Service code list, block list, block data

Response APDU:

Data	Status Word
8 bytes IDm + Status Flag 1 + Status Flag 2	SW1, SW2

6.2.18. PAPDU_FELICA_SYS_CODE

This command issues a REQ SYSTEM CODE as defined in RC-S850/860 Command-Ref-Manual Section 6.1.7.

Command APDU:

Command	CLA	INS	P1	P2	P3	Data
FeliCa REQ SYSTEM CODE	FF	4A	00	00	00	-

Response APDU:

Data	Status Word
8 bytes IDm + No. of System Codes (n) + System Code List (2n)	SW1, SW2

6.2.19. PAPDU_NFC_TYPE1_TAG_RID

This command issues a RID to get the tag's identification data.

Command APDU:

Command	CLA	INS	P1	P2	P3	Data
TYPE1 Tag RID	FF	50	00	00	00	-

Response APDU:

Data	Status Word
HR0 HR1 UID0 UID1 UID2 UID3	SW1, SW2

- HR0 and HR1 are the 2 bytes header ROM which identify the tag
- UID0 through UID3 are the first 3 bytes of the tag's UID

Topaz tags have a 7 bytes long UID which can be fully fetched using the GET_UID APDU described earlier in this manual.

6.2.20. PAPDU_NFC_TYPE1_TAG_RALL

This command issues a RALL to read the two header ROM bytes and the whole of the static memory blocks 0x0-0xE.

Command APDU:

Command	CLA	INS	P1	P2	P3	Data
TYPE1 Tag RALL	FF	52	00	00	00	-

Response APDU:

Data	Status Word
HR0 HR1 120 bytes (Blocks 0 – E)	SW1, SW2

6.2.21. PAPDU_NFC_TYPE1_TAG_READ

This command issues a READ to read a single EEPROM memory byte within the static memory model area of blocks 0x0-0xE.

Command APDU:

Command	CLA	INS	P1	P2	P3	Data
TYPE1 Tag READ	FF	54	00	Byte Addr	00	-

P2 codes the address of the memory byte in the following way:

Bit numbers	Description
b7 – b3	Block # (value between 0x0 and 0xE)
b2 – b0	Byte # within the block (value between 0 and 7)

Response APDU:

Data	Status Word
Data returned by card	SW1, SW2

6.2.22. PAPDU_NFC_TYPE1_TAG_WRITE_E

This command issues a WRITE to erase and then writes the value of 1 memory byte within the static memory model area of blocks 0x0-0xE.

Command APDU:

Command	CLA	INS	P1	P2	P3	Data
TYPE1 Tag WRITE ERASE	FF	56	00	Byte Addr	01	Data

P2 codes the address of the memory byte in the following way:

Bit Numbers	Description
b7 – b3	Block # (value between 0x0 and 0xE)
b2 – b0	Byte # within the block (value between 0 and 7)

Response APDU:

Data	Status Word
Data returned by card	SW1, SW2

6.2.23. PAPDU_NFC_TYPE1_TAG_WRITE_NE

This command issues a WRITE-NE to write a byte value to one byte within the static memory model area of blocks 0x0-0xE. It does not erase the value of the targeted byte before writing the new data. Execution time of this command for NFC Forum tags type 1 is approximately half that of the normal write command (WRITE-E). Using this command, EEPROM bits can only be set, not reset.

Command APDU:

Command	CLA	INS	P1	P2	P3	Data
TYPE1 Tag WRITE No ERASE	FF	58	00	Byte Addr	01	Data

P2 codes the address of the memory byte in the following way:

Bit numbers	Description
b7 – b3	Block # (value between 0x0 and 0xE)
b2 – b0	Byte # within the block (value between 0 and 7)

Response APDU:

Data	Status Word
Data returned by card	SW1, SW2

6.2.24. PAPDU_NFC_TYPE1_TAG_RSEG

This command issues a RSEG to read out a complete segment (or block) of the memory within dynamic memory model.

Please note that this command works only on specific Topaz tags in the dynamic memory model.

Command APDU:

Command	CLA	INS	P1	P2	P3	Data
TYPE1 Tag READ SEGMENT	FF	5A	00	SegAddr	00	-

P2 segment address:

Bit Numbers	Description
b7 – b4	Segment (0x0 – 0xF)
b2 – b0	0

Response APDU:

Data	Status Word

128 bytes of data	SW1, SW2

6.2.25. PAPDU_NFC_TYPE1_TAG_READ8

This command issues a READ8 to read out a block of eight bytes.

Please note that this command only works on Topaz tags in dynamic memory model.

Command APDU:

Command	CLA	INS	P1	P2	P3	Data
TYPE1 Tag READ BLOCK	FF	5C	00	Block Addr	00	-

P2 block address:

Bit Numbers	Description
b7 – b0	General block (0x00 -0xFF)

Response APDU:

Data	Status Word
8 bytes of data	SW1, SW2

6.2.26. PAPDU_NFC_TYPE1_TAG_WRITE_E8

This command issues a WRITE8 to erase and then write a block of eight bytes.

Please note that this command only works on Topaz tags in dynamic memory model.

Command APDU:

Command	CLA	INS	P1	P2	P3	Data
TYPE1 Tag WRITE and ERASE BLOCK	FF	5E	00	Block Addr	08	Data

P2 block address:

Bit Numbers	Description
b7 – b0	General block (0x00 -0xFF)

Response APDU:

Data	Status Word
8 bytes of written data	SW1, SW2

6.2.27. PAPDU_NFC_TYPE1_TAG_WRITE_NE8

This command issues a WRITE8 to write a block of eight bytes. It does not erase the value of the targeted byte before writing the new data. Using this command, EEPROM bits can be set but not reset.

Please note that this command only works on Topaz tags in dynamic memory model.

Command APDU:

Command	CLA	INS	P1	P2	P3	Data
TYPE1 Tag WRITE and NO ERASE BLOCK	FF	60	00	Block Addr	08	Data

P2 block address:

Bit numbers	Description
b7 – b0	General block (0x00 -0xFF)

Response APDU:

Data	Status Word
8 bytes of data	SW1, SW2

6.2.28. PAPDU_HID_ICLASS_READ_PACS_DATA

This command can be used to read the PACS data from a 15693 HID iClass card. The firmware will internally interpret the PACS data from iClass with the help of HID iClass SAM.

Command APDU:

Command	CLA	INS	P1	P2	P3
Read PACS data	FF	F9	00	00	00

Response APDU:

Byte-0	Byte-1	Byte-2	N-Bytes of data	SW1 SW2
Total Length	Padding Bits Length	Number of data bytes	Actual Data + Padding Bits	90 00

6.3. Escape Commands for uTrust 5501 F

With Amendment 1 of the PC/SC specification, Part 3, a method to define vendor specific commands has been introduced.

uTrust 5501 F provides the command READER_GENERIC_ESCAPE to send commands using this method. However, most of the escape commands listed here are not defined according to this method because of backward compatibility reasons.

All newly defined commands will adhere to this new standard. See the command CONTACT_READ_INSERTION_COUNTER as an example.

6.3.1. Sending Escape Commands to uTrust 5501 F

A developer can use the following methods to send escape commands to uTrust 5501 F:

- SCardControl method defined in PC/SC API
- SCardTransmit method defined in PC/SC API in conjunction with the Escape command APDU defined in this manual

Please note that SCardTransmit will only work when connected to a card.

In Windows, in order to be able to send escape commands to uTrust 5501 F, the feature has must be enabled by setting a REG_DWORD value named 'EscapeCommandEnable' in the registry to a value of '1'.

For Windows 7, Windows 8, Windows 8.1, and Windows 10, the key to hold the value for uTrust 5501F contact:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\USB\VID_04E6&PID_5810\Device-Instance-xxxx \Device Parameters\WUDFUsbccidDriver

For contactless:
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\USB\VID_04E6&PID_5610&MI_01\ Device-Instance-xxxx \Device Parameters\WUDFUsbccidDriver

For SAM:
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\USB\VID_04E6&PID_5610&MI_00\ Device-Instance-xxxx \Device Parameters\WUDFUsbccidDriver

Note: When using the Identiv-supplied driver, this will not be necessary.

Device-Instance-xxxx will be an auto-generated combination of four hexadecimal numbers, separated by '&', so this modification must be made for every physical reader/slot intended to be used on the machine in question. The reader must be plugged in at least once for the mentioned keys to exist and the driver must be restarted for this setting to take effect (unplug and re-plug the reader).

To be able to work with synchronous memory cards using Identiv's MCard API, the same setting must be established.

See appendix B for some sample codes sending escape commands to the reader.

6.3.2. Escape Command Codes

Escape commands can be used by an application to configure uTrust 5501 F to function in a mode that is not its default configured mode or to get specific information. To put uTrust 5501 F back into its default mode, it must be unplugged and re-plugged.

The following escape commands are supported by uTrust 5501 F.

6.3.3. Common for Contact and Contactless Interfaces

ESCAPE COMMAND	ESCAPE CODE
READER_SETMODE	0x01
READER_GETMODE	0x02
READER_GETIFDTYPE	0x12
READER_LED_CONTROL	0x19
READER_GETINFO_EXTENDED	0x1E
READER_LED_CONTROL_BY_FW	0xB2
READER_BUZZER_CONTROL	0x51
READER_GENERIC_ESCAPE	FF 70 04 E6 XX

6.3.3.1. READER_SETMODE

This escape command sets the current mode of the reader. Applications may call this function to set the desired mode. Typically, this call is used to switch between the ISO 7816, EMV, memory card, and NFC test mode operations. Upon power on, the reader will reset to the default ISO 7816 mode.

Input:
The first byte of the input buffer contains the escape code value and the second byte contains the value for the desired mode of operation. The output buffer field shall be NULL.

Byte0	Byte1
Escape code (0x01)	Mode

The following table defines the values for the mode parameter:

Mode	Value	Remarks
ISO 7816	0x00	ISO 7816 mode – Applicable for both contact and contactless slots
EMV	0x01	EMV – Applicable only for contact slot and ignored by contactless interface
Synchronous	0x02	Memory card mode (synchronous) – Applicable only for contact slot and ignored by contactless interface
NFC Test	0x04	NFC test mode – Applicable only for contactless interface

ISO mode uses APDU mode of data transfer and is used for normal operation. This is the default mode of the reader on power up.

EMV mode also uses APDU mode of data transfer and is used for EMV test purposes. This mode has more stringent checks for smartcard detection and communication as per EMV4.2 spec.

Synchronous mode is used for communicating only with memory cards.

NFC test mode is used to ignore deactivate-activate sequence during SCardConnect. (PC_TO_RDR_ICCPOWERON - 0x62, and PC_TO_RDR_ICCPOWEROFF – 0x63).

Output:

Output buffer
NULL

6.3.3.2. READER_GETMODE

This escape command retrieves the current mode of the reader.

Input:
The input buffer contains the escape code value.

Byte0
Escape code(0x02)

Output:
The currently active reader mode will be returned as a byte value

Mode	Value	Remarks
ISO	0x00	ISO 7816 mode
EMV	0x01	EMV mode
Synchronous	0x02	Memory card mode (synchronous)
NFC Test	0x04	NFC Test Mode

6.3.3.3. READER_GET_IFDTYPE

This escape command is used to get the current IFD type from the reader.

Input:
The first byte of the input buffer contains the escape code.

Byte0
Escape code(0x12)

Output:
The reader returns its PID LSB first.

PID Value		Description
B0	B1	
0x10	0x56	Identiv uTrust 5501 Contactless Reader
0x10	0x56	Identiv uTrust 5501 SAM Reader
0x10	0x58	Identiv uTrust 5501 R Smart Card Reader

6.3.3.4. READER_LED_CONTROL

This escape command is used to toggle LED state. LED control by firmware should be disabled using escape command READER_LED_CONTROL_BY_FW to see proper LED change when using this IOCTL.

Input:
The first byte of the input buffer contains the escape code, followed by LED number (if more than one LED is present, otherwise it is set to 0), and then desired LED state. This will be required for production purposes.

Byte0	Byte 1	Byte2
Escape code(0x19)	LED number (0-RED,1-GREEN)	LED state (0-OFF, 1-ON)

Output:

Output buffer
NULL

6.3.3.5. READER_GET_INFO_EXTENDED

This escape command is used to get the reader's firmware version, reader capabilities, and Unicode serial number.

Input:
The first byte of the input buffer contains the escape code.

Byte0
Escape code(0x1E)

Output:
The firmware will return data as per below structure SCARD_READER_GETINFO_PARAMS_EX.

Field Size (Bytes)	Field Name	Field Description	Value/Default
1	byMajorVersion	Major Version in BCD	Based on current firmware version
1	byMinorVersion	Minor Version in BCD	
1	bySupportedModes	Bit map indicating the supported modes. 0x01 => EMV mode 0x02 => Memory card mode 0x04 => NFC test mode	0x07 for Contact + Contactless readers 0x03 for Contact only readers Note: ISO mode is not indicated as it is always supported
2	wSupportedProtocols	Protocols supported by the Reader Bit 0 – T0 Bit 1 – T1	0x0003 Received as LSB first
2	winputDevice	IO_DEV_NONE 0x00 IO_DEV_KEYPAD 0x01 IO_DEV_BIOMETRIC 0x02	0x0000 Received as LSB first
1	byPersonality	Reader Personality (not Used)	0x00
1	byMaxSlots	Maximum number of slots	0x02 (contact and contactless)
1	bySerialNoLength	Serial number length (0x1C)	0x1C

28	abySerialNumber	Unicode serial number	Reader serial number (MSB first)

6.3.3.6. READER_LED_CONTROL_BY_FW

This escape command is used to enable/disable LED control by firmware.

Input:
The first byte of the input buffer contains the escape code. The second byte specifies if LED control by firmware should be disabled or enabled.

Byte0	Byte1	
	Value	Description
Escape code(0xB2)	0	Enable LED control by firmware
	1	Disable LED control by firmware
	FF	Get state: 0 – LED control by firmware enabled 1 – LED control by firmware disabled

Output:
No response is returned for set state. For get state, 1 byte response is received.

Output buffer
Current state

6.3.3.7. READER_BUZZER_CONTROL

This escape command is used to enable/disable BUZZER.

Input:
The first byte of the input buffer contains the escape code. The second byte specifies if buzzer disabled or enabled.

Byte0	Byte1	
	Value	Description
Escape code(0x51)	0	Disable buzzer
	1	Enable buzzer
	FF	Get state: 0 – buzzer is disabled 1 – buzzer is enabled

Output:
1 byte response is always received.

Output buffer
Current state 0 – buzzer is disabled 1 – buzzer is enabled

6.3.3.8. READER_GENERIC_ESCAPE

This escape command is used to invoke newly defined escape functions and send proprietary commands to the reader. It is defined in line with vendor specific generic command defined in [PCSC3-AMD1].

Input:
The first five bytes of the input buffer shall follow APDU structure as per [PCSC3-AMD1]. 6TH byte shall be the command code used to identify the specific command.

Byte0	Byte1	Byte2	Byte3	Byte4	From Byte5 (Up to Lc Bytes)		ByteLc+5
					Byt 5	Byte6 Onwards	
0xFF	0x70	0x04	0xE6	Lc (always > 0)	Cmd Opcode	Command parameters or data	Le (optional)

Output:
Depending on the command, the output shall be Le bytes of data + SW1 + SW2 or SW1+ SW2. The escape message shall at least return 2 bytes status words SW1, SW2. In case of success, SW1=0x90 and SW2=0x00 shall be returned. In error, appropriate error status will be returned (as defined in Error Code section 8.0).

6.3.3.9. READER _CONTROL_CONTACT_SLOT

This escape command is supported through the READER_GENERIC_ESCAPE message.

This command can be used to disable the contact slot until it is re-enabled through the same command or until the reader is re-plugged. When a dual-interface card is placed in the contact slot, it will be detected in both contact and contactless modes. To enable applications to actively switch the detection from contact-only mode to contactless-only mode, this escape command can be used along with CNTLESS_ SWITCH_RF_ON_OFF.

Input
To enable/disable/"get-current-status" of contact slot:

Byte0 CLA	Byte1 INS	Byte2 P1	Byte3 P2	Byte4 Lc	Byte5	Byte6	Byte7	Le
0xFF	0x70	0x04	0xE6	0x03	0x05 (opcode)	0x01	0x00 – to enable	00
							0x01 – to disable	
0xFF	0x70	0x04	0xE6	0x02	0x05 (opcode)	0x00 – to get current contact status		00

Byte2 and Byte3 constitute the world wide unique vendor ID as assigned by the USB organization. For Identiv-based readers, Byte2 = 0x04 and Byte3 = 0xE6, since the USB Vendor ID is 0x04E6

Output:
If the command is successful, a single byte is returned. This byte indicates the status of the contact slot that needs to be interpreted as below:

Byte 0	Description
0x00	Contact slot is enabled

0x01	Contact slot is disabled

6.3.4. Specifics for Contactless Interface

ESCAPE COMMAND	ESCAPE CODE
CNTLESS_GETCARDINFO	0x11
CNTLESS_GET_ATS_ATQB	0x93
CNTLESS_GET_TYPE	0x94
CNTLESS_SET_TYPE	0x95
CNTLESS_CONTROL_PPS	0x99
CNTLESS_RF_SWITCH	0x96
CNTLESS_SWITCH_RF_ON_OFF	0x9C
CNTLESS_CONTROL_848	0x9D
CNTLESS_GET_BAUDRATE	0x9E
CNTLESS_CONTROL_RETRIES	0xA7
CNTLESS_CONTROL_POLLING	0xAC
CNTLESS_FORCE_BAUDRATE	0xAD
CNTLESS_GET_CARD_DETAILS	0xDA
CNTLESS_IS_COLLISION_DETECTED	0xE4
CNTLESS_FELICA_PASS_THRU	0xF3
CNTLESS_CONTROL_KBD_EMULATION	READER_ESCAPE_GENERIC (0x11)
CNTLESS_LF_COMMAND_SET	READER_ESCAPE_GENERIC (0x10)

6.3.4.1. CNTLESS_GET_CARD_INFO

This escape command is used to get information about a contactless card placed in field of the reader.

Input:
The first byte of input buffer contains the escape code.

Byte0
Escape code(0x11)

Output:

Byte0	Byte1	Byte2
Contactless card present (0x01)	Card to Reader communication baud rate (0xNN - see table below for details)	Card Type Info (Upper nibble indicates memory card/T=CL/dual mode card; Lower nibble indicates Type A/ Type B card See Table below for values)

Card to reader communication baud rate BYTE is defined as follows:
b0 – 212kbps supported (direction reader to card)
b1 – 424kbps supported (direction reader to card)
b2 – 848kbps supported (direction reader to card)
b3 – always 0
b4 – 212kbps supported (direction card to reader)
b5 – 424kbps supported (direction card to reader)
b6 – 848kbps supported (direction card to reader)
b7– 1 – indicates same baud rate in both directions
0 – indicates different baud rates in both directions

Example:
If 0xNN = 0x77, the card supports all baud rates, namely 106, 212, 424 and 848 kbps in both directions.
If 0xNN = 0xB3, the card supports 106, 212, and 424 kbps in both directions.

Card Type Info:

Upper Nibble Value	Description
0	Memory card
1	T=CL card
2	Dual mode card
Lower Nibble Value	
0	Type A card
1	Type B card

6.3.4.2. CNTLESS_GET_ATS_ATQB

This escape command retrieves the ATS for type A T= CL or the ATQB for type B cards.

Input:
The first byte of input buffer contains the escape code.

Byte0
Escape code(0x93)

Output:
The output buffer contains the ATS bytes or the ATQB bytes, depending on the type of PICC placed on the reader

6.3.4.3. READER_CNTLESS_GET_TYPE

This escape command retrieves the type of cards for which the reader is configured to poll.

The input buffer shall contain the escape command code in the first byte and an optional extension specifier 0xFF in the second byte.

Byte0	Byte1
0x94	Empty or 0xFF

The output buffer shall point to a BYTE buffer in case the extension specifier is not given and will contain the type value coded as:

Value	Description
0x00	Type A
0x01	Type B
0x02	Type A + type B

The output buffer shall be any array of 4 bytes in case the extension specifier is given in input.

Bitmask for Byte-0 of output

Cards-Type-Bit Mask (Byte-0)								
Bit-7	**Bit-6**	**Bit-5**	**Bit-4**	**Bit-3**	**Bit-2**	**Bit-1**	**Bit-0**	**Card Type**
-	-	-	-	-	-	-	1	Type-A
-	-	-	-	-	-	1	-	Type-B
-	-	-	-	-	1	-	-	B-prime
-	-	-	-	1	-	-	-	B-prime-Sof
-	-	-	1	-	-	-	-	i-Class
-	-	1	-	-	-	-	-	FeliCa 212
-	1	-	-	-	-	-	-	FeliCa 424
1	-	-	-	-	-	-	-	Topaz

Bitmask for Byte-1 of output

Cards-Type-Bit Mask (Byte-1)								
Bit-7	**Bit-6**	**Bit-5**	**Bit-4**	**Bit-3**	**Bit-2**	**Bit-1**	**Bit-0**	**Card Type**
-	-	-	-	-	-	-	1	SRT 512
-	-	-	-	-	-	1	-	15693 VICC
-	-	-	-	-	1	-	-	15693 iClass

-	-	-	-	1	-	-	-	125 KHz FSK
-	-	-	1	-	-	-	-	125 KHz ASK
-	-	1	-	-	-	-	-	125 KHz PSK
-	1	-	-	-	-	-	-	125 KHz EM 42XX
1	-	-	-	-	-	-	-	125 KHz HiTag2

Byte-2 and Byte-3 will be 0x00 each and are reserved for future use.

6.3.4.4. READER_CNTLESS_SET_TYPE

This escape command configures the type of cards for which the reader will poll.

Using this command can improve the polling efficiency for applications where only specific types of cards are expected.

This escape command needs to be used with care. For example, do not disable the polling for a given type of the card after having placed that card on the reader. Since reader will immediately stop polling for the card, it will never detect that the card is removed.

The input buffer shall contain two or three bytes:

Byte0	Byte1	Byte3	Description
Escape code(0x95)	0x00	-	Type A
	0x01	-	Type B
	0x02	-	Type A + type B
	0xFF	Bitmask	See the following table

Cards-Type-Bit Mask (Byte-0)								
Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0	Card Type
-	-	-	-	-	-	-	1	Type-A
-	-	-	-	-	-	1	-	Type-B
-	-	-	-	-	1	-	-	B-prime
-	-	-	-	1	-	-	-	B-prime-Sof
-	-	-	1	-	-	-	-	i-Class
-	-	1	-	-	-	-	-	FeliCa 212
-	1	-	-	-	-	-	-	FeliCa 424
1	-	-	-	-	-	-	-	Topaz

Cards-Type-Bit Mask (Byte-1)								
Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0	Card Type
-	-	-	-	-	-	-	1	SRT 512
-	-	-	-	-	-	1	-	15693 VICC
-	-	-	-	-	1	-	-	15693 iClass
-	-	-	-	1	-	-	-	125 KHz FSK
-	-	-	1	-	-	-	-	125 KHz ASK
-	-	1	-	-	-	-	-	125 KHz PSK
-	1	-	-	-	-	-	-	125 KHz EM 42XX
1	-	-	-	-	-	-	-	125 KHz HiTag2

Byte-2 and Byte-3 will be 0x00 each and are reserved for future use.

Output buffer:

Output Buffer
NULL

6.3.4.5. CNTLESS_CONTROL_PPS

This escape command disables the automatic PPS done by the firmware/device for contactless cards.

Input:
The first byte of input buffer contains the escape code.

The second byte either sets the mode or contains a code to retrieve the setting.

Input			Output
Byte0	**Byte1 - PPS Control Byte**		**Byte0**
Escape code(0x99)	0	Enable	No Output
	1	Disable	No Output
	FF	Get current status	0 – PPS is enabled 1 – PPS is disabled

Output:
No response is returned for set state. For get state, 1 byte response is received.

Output Buffer
NULL or current state

6.3.4.6. CNTLESS_RF_SWITCH

This escape command can be used to switch the RF field ON or OFF.

Input:
The first byte of input buffer contains the escape code.

The second byte either sets the mode or contains a code to retrieve the setting.

Byte0	Byte1		Output
	Value	Description	Byte0
Escape code(0x96)	0x00	Switch RF Field OFF	No Output
	0x01	Switch RF Field ON	No Output
	0xFF	Get current field state	0 – RF field is ON 1 – RF field is OFF

Output:
No response is returned for set state. For get state, 1 byte response is received.

Output buffer
NULL or current state

6.3.4.7. CNTLESS_SWITCH_RF_ON_OFF

This escape command is used to switch the RF field ON or OFF when a contact smart card is inserted into the reader. By default, the RF field is always in the ON state and when a contact smart card is inserted in the reader, the RF field is turned OFF.

Input:
The first byte of input buffer contains the escape code.

The second byte either sets the mode or contains a code to retrieve the setting.

Byte0	Byte1		Output
	Value	Description	Byte 0
Escape code(0x9C)	0x00	Switch RF Field OFF when contact card is present in the reader	No Output
	0x01	Leave RF Field ON when contact card is present in the reader	No Output
	0xFF	Get current field state	0x00 — RF is OFF when contact card is present in the reader 0x01 — RF is ON when contact card is present in the reader

After the RF is turned off, to turn the RF ON again, card connect has to be done in direct mode.

Output:
No response is returned for set state. For get state, 1 byte response is received.

Output buffer
NULL or current state

6.3.4.8. CNTLESS_CONTROL_848

This escape command can be used to enable/disable 848 Kbps support as well as query whether 848 kbps is currently enabled or disabled.

RF communication with a user token will only switch to 848 Kbps if the user token supports this baud rate and provided automatic PPS is ON.

The input buffer shall contain 2 bytes:

Byte0	Byte1	Description
0x9D	0x00	Disable 848 Kbps support
	0x01	Enable 848 Kbps support
	0xFF	Get current status on 848 Kbps support

If B1 of the input buffer is 0x00 or 0x0, then the output buffer is:

Output buffer
NULL

If B1 of the input buffer is 0xFF, the output buffer is a BYTE buffer with following possible values:

Output Buffer	Description
0x00	848 Kbps is disabled
0x01	848 Kbps is enabled

6.3.4.9. CNTLESS_GET_BAUDRATE

This escape command is used to get the current baud rate of card-reader communication.

Input:
The first byte of input buffer contains the escape code.

Byte0
Escape code(0x9E)

Output:
The output contains a byte with the following possible values:

Byte0	Description
0x00	106 Kbps in both directions
0x01	106 Kbps from PICC to PCD, 212 Kbps from PCD to PICC
0x02	106 Kbps from PICC to PCD, 424 Kbps from PCD to PICC
0x03	106 Kbps from PICC to PCD, 848 Kbps from PCD to PICC
0x10	212 Kbps from PICC to PCD, 106 Kbps from PCD to PICC
0x11	212 Kbps in both directions
0x12	212 Kbps from PICC to PCD, 424 Kbps from PCD to PICC
0x13	212 Kbps from PICC to PCD, 848 Kbps from PCD to PICC
0x20	424 Kbps from PICC to PCD, 106 Kbps from PCD to PICC
0x21	424 Kbps from PICC to PCD, 212 Kbps from PCD to PICC
0x22	424 Kbps in both directions
0x23	424 Kbps from PICC to PCD, 848 Kbps from PCD to PICC
0x30	848Kbps from PICC to PCD, 106 Kbps from PCD to PICC
0x31	848 Kbps from PICC to PCD, 212 Kbps from PCD to PICC
0x32	848 Kbps from PICC to PCD, 424 Kbps from PCD to PICC
0x33	848 Kbps in both directions

6.3.4.10. CNTLESS_CONTROL_RETRIES

This escape command is used to enable/disable CRC/PROTOCOL/TIMEOUT error retries which are enabled by default for contactless cards.

Input:
The first byte of input buffer contains the escape code.

The second byte either sets the mode or contains a code to retrieve the setting.

Input			Output
Byte0	**Byte1- Description**		**Byte 0**
Escape code(0xA7)	0x00	Enable RNAK retries	No Output
	0x01	Disable RNAK retries	No Output
	0xFF	Get current state of retries	0x00 - Retries are enabled 0x01 - Retries are disabled

Output:
No response is returned for set state. For get state, 1 byte response is received.

Output buffer
NULL or current state

6.3.4.11. CNTLESS_CONTROL_POLLING

This escape command is used to enable/disable firmware polling for contactless cards.

Input:
The first byte of input buffer contains the escape code.

The second byte either sets the mode or contains a code to retrieve the setting.

Input			Output
Byte0	**Byte1 - Description**		**Byte 0**
Escape code(0xAC)	0x00	Enable polling	No output
	0x01	Disable polling	No output
	0XFF	Get current state of polling	0x00 – Polling enabled 0x01 – Polling disabled

Output:
No response is returned for set state. For get state, 1 byte response is received.

Output buffer
NULL or current state

6.3.4.12. CNTLESS_FORCE_BAUDRATE

This escape command can be used to restrict the baud rate for contactless cards to certain values.

The input buffer is:

Byte #	Value	Description
B0	0xAD	Escape command code
B1	0x00	Use the baud rate specified by the card
	0x01	Only allow baud rates specified in B2
B2	b0- DR=2 supported, if bit is set to 1 b1- DR=4 supported, if bit is set to 1 b2- DR=8 supported, if bit is set to 1 b3- shall be set to 0, 1 is RFU b4- DS=2 supported, if bit is set to 1 b5- DS=4 supported, if bit is set to 1 b6- DS=8 supported, if bit is set to 1 b7- 1 if the same D is required for both communication directions b7- 0 if different D is supported for each communication direction	Encoding of the baud rate to be allowed if B1 value is 0x01. No need to send this byte in case B1 has the value =0x00
	NULL	If B1=0x00

The output buffer is:

Output buffer
NULL

6.3.4.13. CNTLESS_GET_CARD_DETAILS

This escape command is used to get details about the PICC placed in the field of the reader.

Input:
The first byte of input buffer contains the escape code.

Byte0
Escape code(0xDA)

Output:

Byte #	Value	Description
B0	0x00	Type A card
	0x01	Type B card
	0x04	FeliCa 212
	0x08	FeliCa 424
B1	0x00	Memory card
	0x01	T-CL card
	0x02	Dual interface card
	0x43	FeliCa
	0x44	Topaz
	0x45	B-prime
	0x46	i-Class
B2	'xx'	'xx' is the PUPI/UID Length
	0x08	For FeliCa cards
THEN EITHER		
B3-B12		PUPI/UID bytes 0x00 byte padding used if length smaller than 10
B13	0x00	CID not supported
	0x01	CID supported
B14	0x00	NAD not supported
	0x01	NAD supported
B15		Bit Rate Capability
B16		FWI
B17		IFSC
B18		MBLI
B19		SAK
B20		SFGI
OR		
B3–B10		8 Bytes NFCID2
B11		Request service command response time parameter (see JIS-6319 specification)
B12		Request response command response time parameter
B13		Authentication command response time parameter
B14		Read command response time parameter
B15		Write command response time parameter

6.3.4.14. CNTLESS_IS_COLLISION_DETECTED

This escape command is used to identify if multiple type A cards are detected in the field.

Input:

The first byte of input buffer contains the escape code.

Byte0
Escape code(0xE4)

Output:

Value	Description
0x00	Collision is not detected
0x01	Collision is detected

6.3.4.15. CNTLESS_FELICA_PASS_THRU

This escape command is used as a pass through to send FeliCa commands to FeliCa cards.

Input:

The first byte of input buffer contains escape code followed by a FeliCa command to be sent to the card. At least 1 byte of command is required to be sent to the card. Otherwise, an error will be reported.

Byte0	Byte1 onwards
Escape code (0xF3)	FeliCa command bytes

Output:

The response received from the FeliCa card is sent as output for this escape command.

6.3.4.16. CNTLESS_CONTROL_KBD_EMULATION

This command can be used to enable/disable keyboard emulation as well as query whether keyboard emulation is currently enabled or disabled.

Keyboard emulation can be controlled only on selected reader modules

Input:

Byte0 CLA	Byte1 INS	Byte2 P1	Byte3 P2	Byte4 Lc	Byte5	Byte6	Byte7	Le
0xFF	0x70	0x04	0xE6	0x04	0x11	0x01 → SET	0x01 → Enable 0x00 → Disable	00
						0x00 → GET		

Output:

Byte0	
Value	Description
0x00	Keyboard emulation is disabled
0x01	Keyboard emulation is enabled

6.3.4.17. CNTLESS_LF_COMMAND_SET

The LF command set is documented in [LF_MANUAL]

6.3.5. Specific for Contact Interface

ESCAPE COMMAND	Escape code
CONTACT_GET_SET_PWR_UP_SEQUENCE	0x04
CONTACT_EMV_LOOPBACK	0x05
CONTACT_EMV_SINGLEMODE	0x06
CONTACT_EMV_TIMERMODE	0x07
CONTACT_APDU_TRANSFER	0x08
CONTACT_DISABLE_PPS	0x0F
CONTACT_EXCHANGE_RAW	0x10
CONTACT_GET_SET_CLK_FREQUENCY	0x1F
CONTACT_CONTROL_ATR_VALIDATION	0x88
CONTACT_GET_SET_MCARD_TIMEOUT	0x85
CONTACT_GET_SET_ETU	0x80
CONTACT_GET_SET_WAITTIME	0x81
CONTACT_GET_SET_GUARDTIME	0x82

6.3.5.1. CONTACT_GET_SET_PWR_UP_SEQUENCE

This escape command is used to get or set the following parameters:

- Smart card power-on sequence
- Delay between successive activation retires
- Enable/disable any voltage class

As soon as card insertion is detected and power ON message is received from the host, the firmware will start activation with the configured voltage sequence. If the activation fails, it will wait for the configured activation delay and then retry with the next enabled voltage class. If power up succeeds at an operating voltage, the firmware will continue card communication at that voltage. If power up fails in all the enabled operating voltages, then the firmware will report an error.

Input:

The first byte of the input buffer will contain the escape code. The next byte will contain the function to be performed. The third byte will contain the parameter for the function.

Byte0	Byte1		Byte2
	Value	Description	
Escape code(0x04)	0x00	Starts with Class C voltage. (1.8V – 3V – 5V order)	-
	0x01	Starts with Class A voltage. (5V – 3V – 1.8V order)	-
	0x08	Time delay between resets	Delay value in milliseconds
	0x09	Enable/Disable a Voltage Class	Bit Map of all Voltage Classes [Bit0 – Class A; Bit1 – Class B; Bit2 – Class C] Set bit to enable the Voltage class Clear bit to disable the Voltage class
	0xFE	Retrieves all the Activation Configuration	-
	0xFF	Retrieves the current Power up sequence	-

Output:

For retrieving all settings (0xFE), the output will be:

Byte0 Value	Description	Byte 1	Byte2
0x00	Starts with Class C voltage. (1.8V – 3V – 5V order)	Time delay between resets in milliseconds	Bit Map of all Voltage Classes [Bit0 – Class A; Bit1 – Class B; Bit2 – Class C]
0x01	Starts with Class A voltage. (5V – 3V – 1.8V order)		

For retrieving current power-up sequence (0xFF), the output will be:

Byte0 Value	Description
0x00	Starts with Class C voltage. (1.8V – 3V – 5V order)
0x01	Starts with Class A voltage. (5V – 3V – 1.8V order)

Example:
Retrieve all current settings:

```
DataIn = 04 FE
DataOut: 01 0A 07 (3 bytes)
```

00: Starting with Class A
0A: 10ms delay between resets
07: Class A, B, and C enabled

6.3.5.2. CONTACT_EMV_LOOPBACK

This escape command lets the host force the firmware to perform an EMV loop-back application.

Input:
The input buffer contains the escape code value.

Byte0
Escape code(0x05)

Output:

Output buffer
NULL

6.3.5.3. CONTACT_EMV_SINGLEMODE

This escape command allows the host to perform a one-shot EMV loop-back application, as specified in the EMV Level 1 Testing Requirements document.

Input:

Byte0
Escape code(0x06)

Output:

Output buffer
NULL

6.3.5.4. CONTACT_EMV_TIMERMODE

This escape command allows the host to perform a timer mode EMV loop-back application, as specified in the EMV Level 1 Testing Requirements document.

Input:
The input buffer contains the escape code value.

Byte0
Escape code(0x07)

Output:

Output buffer
NULL

6.3.5.5. CONTACT_APDU_TRANSFER

This escape command exchanges a short APDU with the smart card. The user has to ensure that a card is inserted and powered before issuing this escape command.

This escape command is typically used by the MCard API to access synchronous memory cards.

Input:

The input buffer contains the escape code value followed by the short APDU to be sent to the card.

Byte0	Byte1 onwards
Escape code(0x08)	Short APDU to be sent to card

Output:

Output buffer
Response APDU

6.3.5.6. CONTACT_DISABLE_PPS

This escape command disables PPS done by the firmware/device for smart cards. This setting will take effect from the next card connect and remains effective till it is changed again or the next reader power-on. Default mode is PPS-enabled.

Input:
The first byte of the input buffer contains the escape code and the following byte if 1 disables PPS and if 0 enables PPS.

Byte0	Byte1
Escape code(0x0F)	PPS control byte (1-DISABLES PPS, 0-ENABLES PPS)

Output:

Output buffer
NULL

6.3.5.7. CONTACT_EXCHANGE_RAW

This escape command can be used to perform raw exchange of data with the card. The user must ensure that a card is inserted and powered on before issuing this escape command. The card is deactivated upon any reception error.

Input:
The input buffer for this command contains the escape code, low byte of the length of data to be sent, high byte of length of data to be sent, low byte of the length of expected data, high byte of length of expected data, and the command.

Byte0	Byte1	Byte2	Byte3	Byte4	Byte 5 onwards
Escape code(0x10)	LSB of send length	MSB of send length	LSB of expected length	MSB of expected length	Raw data to the card

Output:

Output buffer
Response APDU

6.3.5.8. CONTACT_GET_SET_CLK_FREQUENCY

This escape command is used to instruct the reader to change the smart card clock or to use the current clock divisor. Once set, the change in frequency will take effect immediately. Default divisor value is 10, that is 4.8MHz.

Input:

The first byte of the input buffer contains the escape code; the next byte contains the clock divisor value to set the clock frequency or 0xFF to get the clock frequency.

Byte0	Byte1	
	Value	Description
Escape code(0x1F)	Clock divisor	The value to be set in the smartcard CLK divisor register
	0xFF	Get current clock divisor value

Output:

Set clock frequency: None

Get clock frequency: One byte value indicating the current clock divisor

Output buffer
NULL or current divisor

Clock divisor values:

DIVISOR VALUE	SCCLK Frequency
0x04	4 MHz
0x03	4.8 MHz
0x02	6 MHz
0x01	8 MHz
0x00	12 MHz

```
DataIn = 1F FF

DataOut: 03 (1 byte)
```

6.3.5.9. CONTACT_CONTROL_ATR_VALIDATION

This escape command is used to enable or disable the ATR validation by the firmware in ISO/IEC 7816 mode.

In case the card emits an ATR that is not ISO/IEC 7816 compliant, the card reader may fail to power up the card. In these cases, disabling ATR validation will let you work with the card regardless of ISO conformity of the ATR.

By default, ATR validation is enabled.

Input:
The first byte of the input buffer will contain the escape code; the next byte will contain the control byte.

Byte0	Byte1	
	Value	Description
Escape code(0x88)	0x00	Enable ATR validation
	0x01	Disable ATR validation

Output:

Output buffer
NULL

6.3.5.10. CONTACT_GET_SET_MCARD_TIMEOUT

This escape command is used to get or set the delay, which is applied after a write operation to memory cards. The delay is specified in milliseconds.

Input:
The first byte of the input buffer will contain the escape code; the next byte will contain the memory card write delay in seconds.

Byte0	Byte1	
	Value	Description
Escape code(0x85)	0x01	Delay in milliseconds for memory card Write
	Any value other than 1	Read the current applied delay for memory card Write

Output:
Write delay: No response byte
Read delay value: A byte value specifying the current delay applied during memory card Write in milliseconds

Byte0
Null or Delay in ms

```
DataIn = 85 00
DataOut: 00 (1 byte)
```

6.3.5.11. CONTACT_GET_SET_ETU

This escape command is used by the host to get/set the current ETU for smart cards. Once set, the new ETU value will take effect immediately.

Input:
The input buffer contains the escape code followed by an 8 bit get/set identifier. For set ETU, a DWORD specifying the value to be set follows:

Byte0	Byte1		Byte2	Byte3	Byte4	Byte5
	Value	Description	Wait time			
Escape code(0x80)	0x01	SET ETU	BIT31-BIT24	BIT23-BIT16	BIT15-BIT8	BIT7-BIT0
	0x00	GET ETU	-	-	-	-

Output:
For both set and get ETU, the output will be the following:

Byte0	Byte1	Byte2	Byte3
ETU value			
BIT31-BIT24	BIT23-BIT16	BIT15-BIT8	BIT7-BIT0

```
DataIn = 80 00
DataOut: 00 00 01 40 (4 bytes)
```

6.3.5.12. CONTACT_GET_SET_WAITTIME

This Escape command is used to get/set the Character/Block Waiting Time for smartcards. The wait time is specified in terms of ETU. Once set, the new Wait time will take effect from the next card communication.

Input:
The input buffer contains the escape code followed by an 8 bit get/set identifier, an 8 bit wait time identifier, and a 32 bit wait time value. BWT must be specified in units of 1.25ms, and CWT in units of ETU.

Byte0	Byte1		Byte2		Byte3	Byte4	Byte5	Byte6
	Value	Description	Value	Description	Wait time in ETU			
Escape code (0x81)	0x01	SET Wait time	0x00	CWT	BIT31-BIT24	BIT23-BIT16	BIT15-BIT8	BIT7-BIT0
			0x01	BWT				
	0x00	GET Wait time	0x00	CWT	-	-	-	-
			0x01	BWT				

Output:
For both get/set wait time, the output will be the following:

Byte0	Byte1	Byte2	Byte3
Wait time in ETU			
BIT31-BIT24	BIT23-BIT16	BIT15-BIT8	BIT7-BIT0

```
DataIn = 81 00 01
DataOut: 00 00 03 5D (4 bytes)
```

6.3.5.13. CONTACT_GET_SET_GUARDTIME

This Escape command is used to get/set the character/block guard time of the reader. The guard time is specified in terms of ETU. Once set, the new guard time will take effect immediately.

Input:
The input buffer contains the escape code followed by an 8 bit get/set identifier, an 8 bit guard time identifier, and a 32 bit guard time value in ETU.

Byte0	Byte1		Byte2		Byte3	Byte4	Byte5	Byte6
	Value	Description	Value	Description	Guard time in ETU			
Escape code (0x82)	0x01	SET Guard time	0x00	CGT	BIT31-BIT24	BIT23-BIT16	BIT15-BIT8	BIT7-BIT0
			0x01	BGT				
	0x00	GET Guard time	0x00	CGT	-	-	-	-
			0x01	BGT				

Output:
For get/set guard time, the output will be the character/block guard time value.

Byte0	Byte1	Byte2	Byte3
Character Guard time in ETU			
BIT31-BIT24	BIT23-BIT16	BIT15-BIT8	BIT7-BIT0

```
DataIn = 82 00 01
DataOut: 00 00 00 18 (4 bytes)
```

7. Annexes

7.1. Annex A – Status Words Table

SW1	SW2	Description
0x90	0x00	NO ERROR
0x63	0x00	NO INFORMATION GIVEN
0x65	0x81	MEMORY FAILURE
0x67	0x00	LENGTH INCORRECT
0x68	0x00	CLASS BYTE INCORRECT
0x6A	0x81	FUNCTION NOT SUPPORTED
0x6B	0x00	WRONG PARAMETER P1-P2
0x6D	0x00	INVALID INSTRUCTION BYTE
0x6E	0x00	CLASS NOT SUPPORTED
0x6F	0x00	UNKNOWN COMMAND

7.2. Annex B – Sample Code Using Escape Commands

```
File Name: uTrust 5501 F Escape.h

#ifndef _uTrust_5501F_ESCAPE_H_
#define _uTrust_5501F_ESCAPE_H_

#ifdef __cplusplus
extern"C" {
#endif
```

#pragma pack(1)

typedef struct

```
{
   BYTE byMajorVersion;
   BYTE byMinorVersion;
   BYTE bySupportedModes;
   WORD wSupportedProtocols;
   WORD winputDevice;
   BYTE byPersonality;
   BYTE byMaxSlots;
   BYTE bySerialNoLength;
   BYTE abySerialNumber [28];
} ReaderInfoExtended;
```

#pragma pack()

```
#define IOCTL_CCID_ESCAPE              SCARD_CTL_CODE (0xDAC)

#define READER_SET_MODE                   0x01
#define READER_GET_MODE                   0x02
#define READER_GETIFDTYPE                 0x12
#define READER_LED_CONTROL                0x19
#define READER_LED_CONTROL_BY_FW          0xB2
#define READER_GETINFO_EXTENDED           0x1E
#define READER_RDWR_USR_AREA              0xF0

#define CONTACT_GET_SET_POWERUPSEQUENCE   0x04
#define CONTACT_EMV_LOOPBACK              0x05
#define CONTACT_EMV_SINGLEMODE            0x06
#define CONTACT_EMV_TIMERMODE             0x07
#define CONTACT_APDU_TRANSFER             0x08
#define CONTACT_CONTROL_PPS               0x0F
#define CONTACT_EXCHANGE_RAW              0x10
#define CONTACT_GET_SET_CLK_FREQUENCY     0x1F
#define CONTACT_GET_SET_ETU               0x80
#define CONTACT_GET_SET_WAITTIME          0x81
#define CONTACT_GET_SET_GUARDTIME         0x82
#define CONTACT_GET_SET_MCARD_TIMEOUT     0x85
#define CONTACT_CONTROL_ATR_VALIDATION    0x88

#define CNTLESS_GETCARDINFO               0x11
#define CNTLESS_GET_ATS_ATQB              0x93
#define CNTLESS_CONTROL_PPS               0x99
#define CNTLESS_RF_SWITCH                 0x96
#define CNTLESS_SWITCH_RF_ON_OFF          0x9C
#define CNTLESS_GET_BAUDRATE              0x9E
#define CNTLESS_CONTROL_RETRIES           0xA7
#define CNTLESS_CONTROL_POLLING           0xAC
#define CNTLESS_GET_CARD_DETAILS          0xDA
#define CNTLESS_SET_CONFIG_PARAMS         0xE1
#define CNTLESS_IS_COLLISION_DETECTED     0xE4
#define CNTLESS_FELICA_PASS_THRU          0xF3
#define CNTLESS_P2P_SWITCH_MODES          0xE9
#define CNTLESS_P2P_TARGET_RECEIVE        0xEA
#define CNTLESS_P2P_TARGET_SEND           0xEB
#define CNTLESS_P2P_INITIATOR_TRANSCEIVE  0xE7
#define CNTLESS_NFC_SINGLESHOT            0xEC
#define CNTLESS_NFC_LOOPBACK              0xED
```

```
#ifdef __cplusplus
}
#endif

#endif
File Name: uTrust 5501 F Escape.c


#include <windows.h>
#include <winbase.h>
#include <stdio.h>
#include <conio.h>
#include "winscard.h"
#include "winerror.h"
#include "uTrust 5501F Escape.h"

VOID main(VOID)
{

   SCARDCONTEXT         ContextHandle;
   SCARDHANDLE          CardHandle;
   ReaderInfoExtended   strReaderInfo;
   BYTE                 InByte, i;
   DWORD                BytesRead, ActiveProtocol;
   ULONG                ret;
   char                 *s;
   char                 *ReaderName[] = {"Identiv uTrust 5501 R Smart Card Reader 0",
                                         "Identiv uTrust 5501 Contactless Reader 0",
                                            NULL};

/*************************************************************************************************
***********/

   ContextHandle = -1;

   ret = SCardEstablishContext(SCARD_SCOPE_USER, NULL, NULL, &ContextHandle);

   if (ret == SCARD_S_SUCCESS)
   {
      s = ReaderName[0];
      printf("Connecting to reader %s\n", s);
      ret = SCardConnect(  ContextHandle,
                           s,
                           SCARD_SHARE_DIRECT,
                           SCARD_PROTOCOL_UNDEFINED,
                           &CardHandle,
                           &ActiveProtocol);

      if (ret == SCARD_S_SUCCESS)
      {
         InByte = 0x1E;
         ret = SCardControl(  CardHandle,
                              IOCTL_CCID_ESCAPE,
                              &InByte,
                              1,
                              &strReaderInfo,
                              sizeof(strReaderInfo),
                           &BytesRead);
         if (SCARD_S_SUCCESS == ret) {
            printf("major version:\t\t%d%d\n", (strReaderInfo.byMajorVersion& 0xF0)>> 4,
(strReaderInfo.byMajorVersion& 0x0F));
            printf("minor version:\t\t%d%d\n", (strReaderInfo.byMinorVersion& 0xF0)>> 4,
(strReaderInfo.byMinorVersion& 0x0F));
            printf("modes:\t\t\t%d\n", strReaderInfo.bySupportedModes);
            printf("protocols:\t\t%04x\n", strReaderInfo.wSupportedProtocols);
            printf("input device:\t\t%04x\n", strReaderInfo.winputDevice);
            printf("personality:\t\t%d\n", strReaderInfo.byPersonality);
            printf("maxslots:\t\t%d\n", strReaderInfo.byMaxSlots);
            printf("serial no length:\t%d\n", strReaderInfo.bySerialNoLength);
            printf("serial no:\t\t");
```

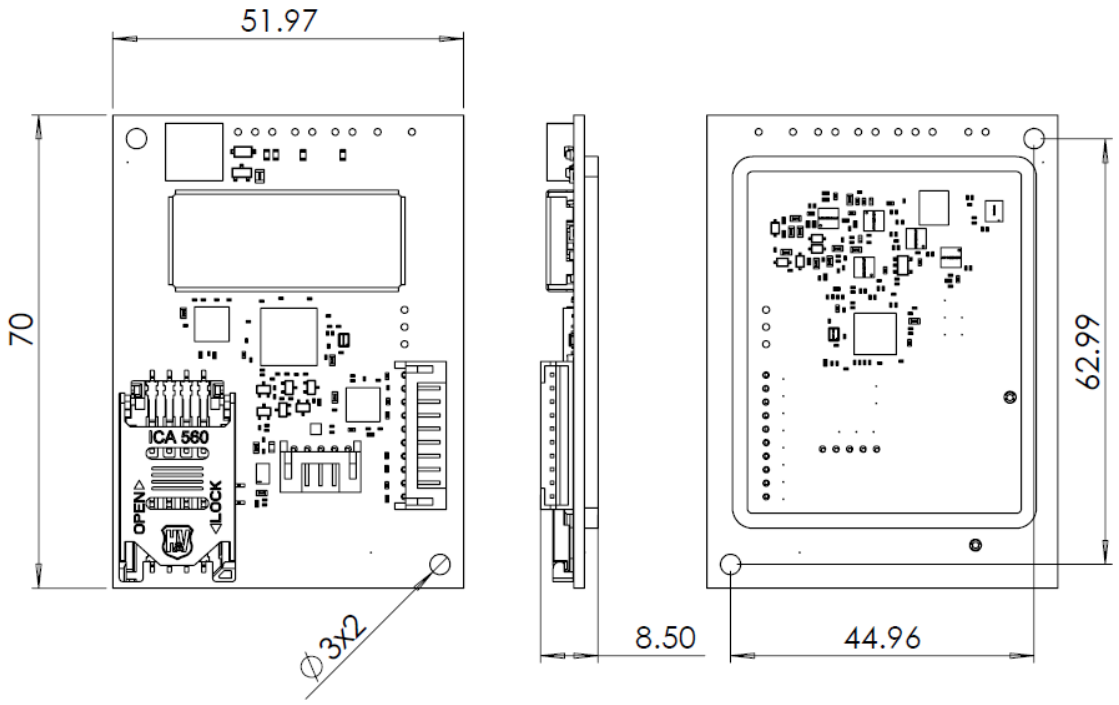
```
            for (i = 0; i <strReaderInfo.bySerialNoLength; i++)
               if (strReaderInfo.abySerialNumber[i] != 0) printf("%c",
strReaderInfo.abySerialNumber[i]);
         } else {
            printf("SCardControl failed: %08X\n", ret);
         }
      }
      else {
         printf("SCardConnect failed: %08X\n", ret);
      }

      ret = SCardReleaseContext(ContextHandle);
   }
   else
   {
      printf("\n SCardEstablishContext failed with %.8lX",ret);
   }

   printf("\npress any key to close the test tool\n");
   getch();
}
```

7.3. Annex C – Mechanical drawings

Product outline



**********End of document**********