HIRSCH

uTrust Software Reference Manual

F Series Smart Card Readers

This document consolidates software reference information for Hirsch's uTrust smart card reader family. It includes shared software details, product-specific differences, and ordering options across several models.

1. Legal Information

1.1. Disclaimers

The content published in this document is believed to be accurate. However, HIRSCH 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.

HIRSCH 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. HIRSCH gives no representation or warranty that such descriptions or examples are suitable for the application that you may want to use them for. Should you notice any problems with this document, please provide your feedback to support@hirschsecure.com

1.2. Licenses

If the document contains source code examples, they are provided for illustrative purposes only and 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 form of compiled applications.
- You MAY NOT copy or distribute parts of or the entire source code without prior written consent from HIRSCH.
- 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 HIRSCH products, they are provided for documentation purposes only. HIRSCH does not grant you any license to its designs.

1.3. Trademarks, logos and brand names

All trademarks, logos, and brand names are the property of their respective owners.

2. Introduction to the Manual

2.1. Objective of the Manual

This manual provides a comprehensive overview of the software features of the *uTrust F Family* Multi-Technology Secure Reader/Writer modules. It details the available interfaces, supported commands, and integration guidelines for developers incorporating uTrust F family readers/writers into their applications.

2.2. Target Audience

This manual is intended for application developers and system integrators with prior experience in smart card technologies. It assumes familiarity with USB CCID protocol, APDU command/response exchange mechanisms, and standards such as ISO/IEC 7816 for contact cards and ISO/IEC 14443/15693 for contactless technologies (13.56 MHz), as well as 125 kHz low-frequency credentials.

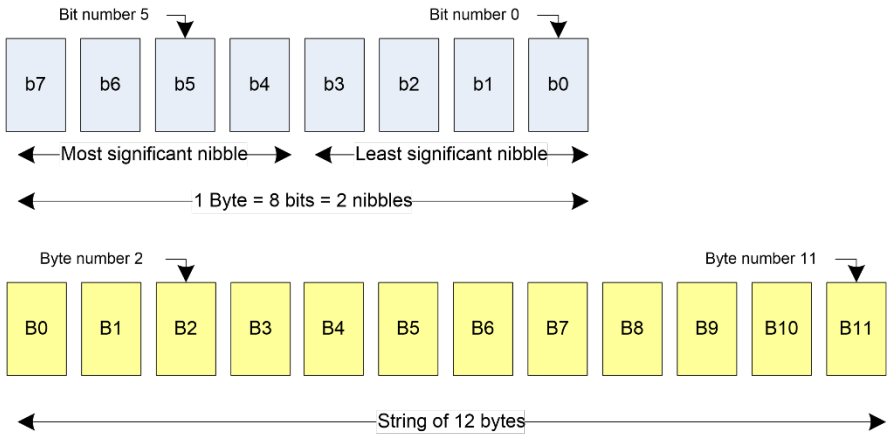
Should you have questions, you may send them to support@hirschsecure.com

Table of Contents

3. Conventions for bits and bytes

Bits are represented by a lower case ‘b’ followed by an ordering digit which indicates its position. Bytes are represented by an upper case ‘B’ followed by one or more ordering digits which indicate their position.

Bit and Byte representation



Example:

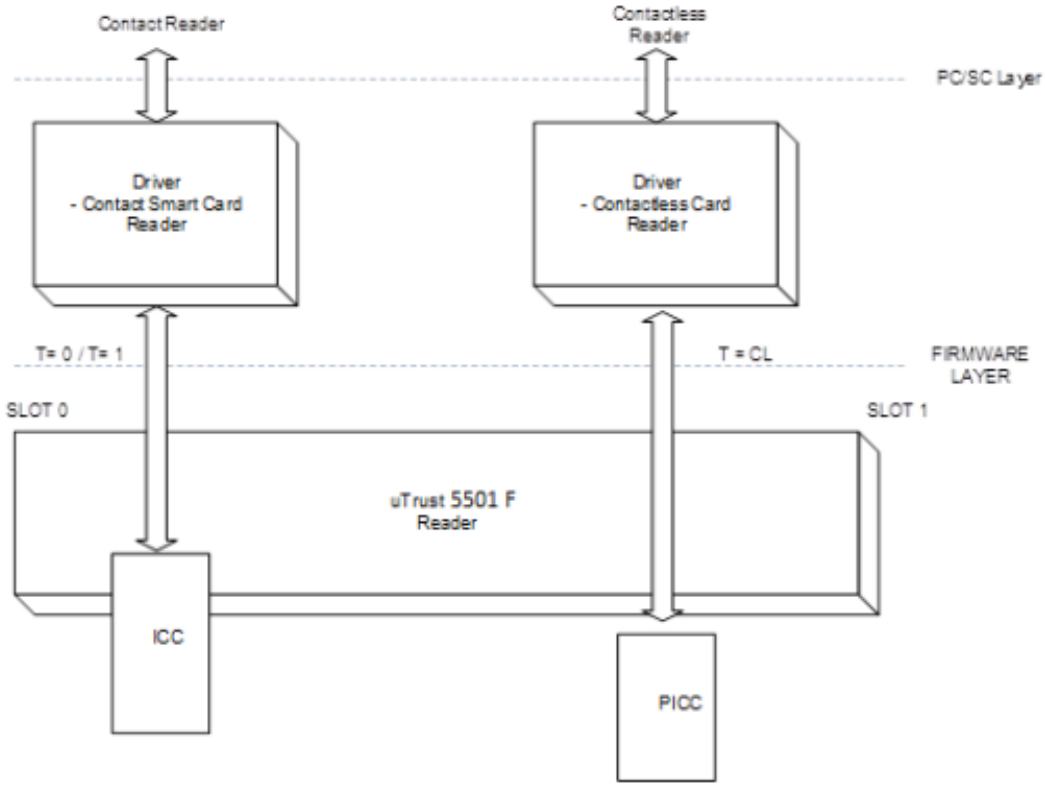
163 decimal number representation

Decimal	HEX	Binary
163	0xA3	10100011
least significant nibble	0x3	0011
most significant nibble	0xA	1010

4. Software Modules and PC/SC Interface

4.1. PC/SC Interface

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



uTrust F family 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@hirschsecure.com and navigate to the downloads page for your reader.

5. 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, Hirsch does provide a dedicated driver for the reader, which is available through Windows Update or at support@hirschsecure.com. An installer bundled with the signed driver is also provided.

6. 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

These tools can be found here: Hirsch | Diagnostic Utilities

7. Driver and Operating System Compatibility

7.1. Driver Listing

Reader Model	PC/SC Listing Names
uTrust 3720 F	• Identiv uTrust 3720 Contactless Reader
uTrust 3721 F	• Identiv uTrust 3721 Contactless Reader
uTrust 47xx F	• Identiv uTrust 47xx F Contact Reader • Identiv uTrust 47xx F Contactless Reader
uTrust 5501 F	• Identiv uTrust 5501 R Smart Card Reader • Identiv uTrust 5501 Contactless Reader • Identiv uTrust 5501 SAM Reader

7.2. 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 and above (32 and 64 bit)
- macOS 10.12, 10.13
- Linux kernel 3.x (32 and 64 bit)

8. CCID

Category	Details

Protocol Standard	USB Device Class: Smart Card CCID Specification for Integrated Circuit(s) Cards Interface Devices, Revision 1.10
CCID Class Requests Supported	Abort
CCID Messages Supported (Contact Interface)	1. PC_to_RDR_IccPowerOn 2. PC_to_RDR_IccPowerOff 3. PC_to_RDR_GetSlotStatus 4. PC_to_RDR_XfrBlock 5. PC_to_RDR_GetParameters 6. PC_to_RDR_ResetParameters 7. PC_to_RDR_SetParameters 8. PC_to_RDR_Escape 9. PC_to_RDR_ICCClock 10. PC_to_RDR_T0APDU 11. PC_to_RDR_Abort 12. PC_to_RDR_SetDataRateAndClockFrequency
CCID Error Codes	HW_ERROR, XFR_PARITY_ERROR, ICC_PROTOCOL_NOT_SUPPORTED, BAD_ATR_TS, BAD_ATR_TCK, ICC_MUTE, CMD_ABORTED, Command not supported
Error Code Descriptions	1. HW_ERROR: Hardware short-circuit, card power failure, or internal hardware issue. 2. XFR_PARITY_ERROR: Parity error detected; reported in response to PC_to_RDR_XfrBlock. 3. ICC_PROTOCOL_NOT_SUPPORTED: Card requests unsupported protocol (not T=0 or T=1). 4. BAD_ATR_TS: Invalid initial character in ATR. 5. BAD_ATR_TCK: Invalid ATR check character. 6. ICC_MUTE: Card does not respond until reader timeout; reported for PC_to_RDR_XfrBlock and PC_to_RDR_IccPowerOn. 7. CMD_ABORTED: Command aborted via control pipe. 8. Command not supported: Issued command not supported by reader.

8.1. Generic APDU Response Description

SW1	SW2	Description
90	00	No error occurred — Command executed successfully.
62	82	Warning: Data object *XX* — Requested information not available.
63	00	No information provided.
63	01	Execution stopped — Failure occurred in another data object.

6A	81	Data object *XX* not supported.
67	00	Wrong length — Data object *XX* has unexpected length.
6A	80	Wrong value — Data object *XX* has unexpected value.
64	00	Execution error: No response from IFD (Interface Device).
64	01	Execution error: No response from ICC (Integrated Circuit Card).
6F	00	General failure — No precise diagnosis available for data object *XX*.

8.2 Product Part Numbers and Descriptions

Product	Description	Part Number(s)
uTrust 3500 F	Contactless HF Module	905502-2, 113259R2R (cable usb-a), 114562R2R (cable usb-c)
uTrust 3502 F	Contactless HF Module	905584
uTrust 3523 F	Contactless HF Module with digital wallet support	905611
uTrust 3720 F	Contactless HF Module	905592-1, 905525 (P/N for Optional Stand Base)
uTrust 3720 F	Contactless LF Module	905592-2, 905525 (P/N for Optional Stand Base)
uTrust 3720 F	HF + LF contactless reader	905592, 905525 (P/N for Optional Stand Base)
uTrust 3721 F	HF + LF contactless reader w/ keyboard input	905593, 905525 (P/N for Optional Stand Base)
uTrust 4501 F	Dual interface (module	905322
uTrust 4511 F		905565-2
uTrust 4701 F	Dual interface (contact + contactless)	905504, 905505, 905506, 905507, 905508, 905320-1
uTrust 4711 F	Contactless only with SAM slot	905565-1
uTrust 5501 F	Dual interface HF + LF reader/writer	905567

uTrust 5501 F	HF + LF contactless reader w/ DTC (Direct-to-Card)	905567-2

9. uTrust 372x Family (3720 F and 3721 F)

With its combination of a modern slim design and its state-of-the-art feature set, uTrust 3720 F is the perfect desktop reader choice for environments where HF/LF contactless card support is required while uTrust 3721 F perfectly fits environments where access to HF/LF contactless cards with data read as keyboard input is required. As for all Hirsch products, uTrust 372x F is designed to offer best in class interoperability.

9.1. uTrust 3500 F

uTrust 3500 F is a compact, contactless USB-CCID smart card reader module designed for seamless integration into kiosks, terminals, and desktop devices. Supporting 13.56 MHz and NFC (ISO/IEC/IEC 14443 & 18092) technologies, it reads a wide range of contactless credentials including MIFARE®, FeliCa™, Calypso, and NFC Forum Tag Types, as well as NFC-enabled mobile devices. With an integrated antenna, high-speed data transfer up to 848 kbps, and SmartOS™ firmware upgradeability, it ensures fast, secure transactions and long-term reliability.

9.2. uTrust 3502 F

uTrust 3502 F is a compact, contactless USB CCID reader module designed for integration with external 50 ohm antennas via a U.FL connector. Supporting 13.56 MHz and NFC technologies, it enables fast, secure transactions with a wide range of contactless credentials, including ID-1 cards, NFC-enabled smart devices with Host Card Emulation (HCE), and mixed credential populations. With SmartOS™ firmware upgradeability, PC/SC compliance, and driver support for Windows®, macOS®, Linux, and Android, it offers flexible deployment in kiosks, terminals, and desktop environments.

9.3. uTrust 3523 F

uTrust 3523 F ECP is a compact, contactless OEM reader module designed for seamless integration into kiosks, terminals, and custom enclosures. It supports 13.56 MHz and NFC technologies, enabling fast, secure interaction with a wide range of contactless credentials, including ID-1 cards, mobile devices with Host Card Emulation (HCE), and digital wallets like Google Wallet and Apple Wallet. With support for external 50 ohm antennas via U.FL or IPEC connectors, USB 2.0 Full Speed interface, and SmartOS™ firmware upgrades, it delivers high-speed performance and flexible deployment across diverse applications.

9.4. 372xF Specific Commands

9.4.1. READ_SINGLE_BLOCK

Reads 4 bytes from specified block.

Command:

CLA	INS	P1	P2	Lc	Data In	Le
0xFF	0xFC	0x00	0x00	0x02	0x20 [Block#]	0x00

Response:

Data field	Status Word
4 Bytes of data from card	SW1 SW2

Example to read block number 0x0F

APDU: FF FC 00 00 02 20 0F 00

RESPONSE: xx xx xx xx 90 00

9.4.2. WRITE_SINGLE_BLOCK

Writes 4 bytes to specified block.

Command:

CLA	INS	P1	P2	Lc	Data In	Le
0xFF	0xFC	0x00	0x00	0x06	0x21 [Block#] [Data0] [Data1] [Data2] [Data3]	-

Response:

Data field	Status Word
-	SW1 SW2

Example to write data bytes “01 02 03 04” to block number 0x0F

APDU: FF FC 00 00 06 21 0F 01 02 03 04

RESPONSE: 90 00

9.4.3. LOCK_BLOCK

Locks specified block; becomes read-only.

Command:

CLA	INS	P1	P2	Lc	Data In	Le
0xFF	0xFC	0x00	0x00	0x02	0x22 [Block#]	-

Response:

Data field	Status Word
-	SW1 SW2

Example to lock block number 0x0F

APDU: FF FC 00 00 02 22 0F

RESPONSE: 90 00

9.4.4. READ_MULTIPLE_BLOCK

Reads multiple consecutive blocks (4 bytes each).

Command:

CLA	INS	P1	P2	Lc	Data In	Le
0xFF	0xFC	0x00	0x00	0x03	0x23 [StartBlock] [NumBlocks-1]	-

Response:

Data field	Status Word
Data from card (number block * 4 Bytes)	SW1 SW2

Example to read 04 consecutive blocks starting at block number 0x10 **APDU:** FF FC 00 00 03 23 10 03 00

RESPONSE: xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx 90 00

9.4.5. WRITE_MULTIPLE_BLOCK

Writes multiple consecutive blocks (4 bytes each).

Command:

CLA	INS	P1	P2	Lc	Data In	Le
0xFF	0xFC	0x00	0x00	3 + 4×NumBlocks	0x24 [StartBlock] [NumBlocks-1] [Data...]	-

Response:

Data field	Status Word
-	SW1 SW2

Example to write “AA AA AA AA BB BB BB BB” to 02 consecutive blocks starting at block number 0x10

APDU: FF FC 00 00 0B 24 10 01 AA AA AA AA BB BB BB BB

RESPONSE: 90 00

9.4.6. WRITE_AFI

Writes AFI value to card memory.

Command:

CLA	INS	P1	P2	Lc	Data In	Le
0xFF	0xFC	0x00	0x00	0x02	0x27 [AFI]	-

Response:

Data field	Status Word
-	SW1 SW2

Example to write AFI value 0xF0
APDU: FF FC 00 00 02 27 F0
RESPONSE: 90 00

9.4.7. LOCK_AFI

Locks AFI value; cannot be updated.

Command:

CLA	INS	P1	P2	Lc	Data In	Le
0xFF	0xFC	0x00	0x00	0x01	0x28	-

Response:

Data field	Status Word
-	SW1 SW2

Example to lock AFI
APDU: FF FC 00 00 01 28
RESPONSE: 90 00

9.4.8. WRITE_DSFID

Writes DSFID value to card memory.

Command:

CLA	INS	P1	P2	Lc	Data In	Le
0xFF	0xFC	0x00	0x00	0x02	0x29 [DSFID]	-

Response:

Data field	Status Word
-	SW1 SW2

Example to write DSFID value 0xF0

APDU: FF FC 00 00 02 29 F0

RESPONSE: 90 00

9.4.9. LOCK_DSFID

Locks DSFID value; cannot be updated.

Command:

CLA	INS	P1	P2	Lc	Data In	Le
0xFF	0xFC	0x00	0x00	0x01	0x2A	-

Response:

Data field	Status Word
-	SW1 SW2

Example to lock DSFID

APDU: FF FC 00 00 01 2A

RESPONSE: 90 00

9.4.10. GET_SYSTEM_INFO

Retrieves system info (UID, DSFID, AFI, memory size, IC ref).

Command:

CLA	INS	P1	P2	Lc	Data In	Le
0xFF	0xFC	0x00	0x00	0x02	0x2B 0x00	-

Response:

Data field	Status Word
System Information Data	SW1 SW2

Note that the size of system information returned can vary between cards depending on the number of fields that are available.

System Information Data is formatted as below:

System Information Data					
Info flags 1 byte	UID 8 bytes	DSFID 1 byte	AFI 1 byte	Memory Size 2 bytes	IC Reference 1 byte

Bits in Info flags byte provide information about the presence or absence of other fields. Info flags is formatted as below:

Bit	Value	Description
b1	0	DSFID not present
	1	DSFID present
b2	0	AFI not present
	1	AFI present
b3	0	Memory size not present
	1	Memory size present
b4	0	IC reference not present
	1	IC reference present
b5-b8	0	RFU

Memory size is interpreted as below: MSB gives the block size in bytes – 1 (add 1 to get number of bytes in block) LSB gives the number of blocks – 1 (add 1 to get number of physical blocks)

Example to lock AFI

APDU: FF FC 00 00 02 2B 00

RESPONSE: xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx 90 00

9.4.11. READ_MULTIPLE_BLOCK_SECURITY

Retrieves security status of consecutive blocks (1 byte each).

Command:

CLA	INS	P1	P2	Lc	Data In	Le
0xFF	0xFC	0x00	0x00	0x03	0x2C [StartBlock] [NumBlocks-1]	-

Response:

Data field	Status Word
Security status bytes (number blocks * 1 bytes)	SW1 SW2

Example to read security status of 04 consecutive blocks starting at block number 0x10

APDU: FF FC 00 00 03 2C 10 03 00

RESPONSE: xx xx xx xx 90 00

9.4.12. TRAVERSE

Sends raw ISO/IEC15693 command to card; returns raw response.

Command:

CLA	INS	P1	P2	Lc	Data In	Le
0xFF	0xFD	FWT code	FrameType	Number of data bytes	Raw Card Command	0x00

The following table provides the FWT codes for P1 parameter and the corresponding wait time.

FWT Code	FWT in uSeconds	FWT Code	FWT in uSeconds
0x00	500	0x0B	750000
0x01	1000	0x0C	1000000
0x02	2000	0x0D	1250000
0x03	5000	0x0E	1500000
0x04	10000	0x0F	1750000
0x05	25000	0x10	2000000
0x06	50000	0x11	2500000
0x07	75000	0x12	3000000
0x08	100000	0x13	4000000
0x09	250000	0x14	5000000
0x0A	500000		

The following table provides the Frame Types for P2 parameter and their description:

Data field	Status Word
0x00	FRAMETYPE_SHORT
0x01	FRAMETYPE_STD
0x02	FRAMETYPE_ACBITORIENTED

Response:

Data field	Status Word
Response from card	SW1 SW2

The response from card is returned raw without any processing. Reception of any response from the card is considered as success irrespective of the content of the response. The calling application needs to process the specific response from card.

9.5. Keyboard Specific Commands

9.5.1. READER_CONTROL_KEYBOARD_SLOT

To enable or disable status of keyboard interface in SRAM, or to get status of keyboard interface.

Command:

Byte 0 CLA	Byte 1 INS	Byte 2 P1	Byte 3 P2	Byte 4 Lc	Byte 5 opcode	Byte 6	Byte 7	Le
0xFF	0x70	0x04	0xE6	0x03	0x11	0x01	0x00 – enable	0x00
0xFF	0x70	0x04	0xE6	0x03	0x11	0x01	0x01 – disable	0x00
0xFF	0x70	0x04	0xE6	0x02	0x11	0x01		0x00

Response:

If the command is successful, 5 bytes are returned indicating the status of keyboard slot.

Byte Offset	Keyboard Description
Byte 0	0x01: Disabled in SRAM 0x00: Enabled in SRAM
Byte 1	00x01: Reader does not include keyboard interface 0x00: Reader includes keyboard interface
Byte 2	0x01: Enabled in OS-Configuration 0x00: Disabled in OS-Configuration
Byte 3	RFU
Byte 4	RFU

- Byte 1 would be 0x01 for readers that do not include keyboard interface, like, uTrust 3720 F. In this case the HID/keyboard interface shall not be enumerated.
- If the HID/keyboard interface is enumerated but disabled in OS-Configuration or SRAM, then keyboard output shall not be emitted until configuration is changed using suitable tool.

9.6. My-d Move Specific Commands

9.6.1. ACCESS

Password verification on my-d card.

Command:

CLA	INS	P1	P2	Lc	opcode	Password	Le
0xFF	0xFD	0x04	0x01	0x05	0xB2	4 bytes	0x00

Response:

Data field	Status Word
-	SW1 SW2

9.6.2. SET PASSWORD

Changes password of my-d card.

Command:

CLA	INS	P1	P2	Lc	opcode	Password	Le
0xFF	0xFD	0x04	0x01	0x05	0xB1	4 bytes	0x00

Response:

Data field	Status Word
-	SW1 SW2

9.6.3. COMPATIBILITY WRITE

Writes 16 bytes to specified block.

Command:

CLA	INS	P1	P2	Lc	opcode	Block No	Data	Le
0xFF	0xFD	0x06	0x01	0x12	0xA0	1 byte	16 bytes	0x00

Response:

Data field	Status Word
-	SW1 SW2

9.6.4. WRITE 2 BLOCKS (8 Bytes)

Writes 8 bytes to specified block.

Command:

CLA	INS	P1	P2	Lc	opcode	Block No	Data	Le
0xFF	0xFD	0x06	0x01	0x0A	0xA1	1 byte	8 bytes	0x00

Response:

Data field	Status Word
-	SW1 SW2

9.6.5. WRITE 1 BLOCKS (4 Bytes)

Writes 4 bytes to specified block.

Command:

CLA	INS	P1	P2	Lc	opcode	Block No	Data	Le
0xFF	0xFD	0x04	0x01	0x06	0xA2	1 byte	4 bytes	0x00

Response:

Data field	Status Word
-	SW1 SW2

9.6.6. READ 4 BLOCKS (16 Bytes)

Reads 16 bytes from specified block.

Command:

CLA	INS	P1	P2	Lc	opcode	Block No	Le
0xFF	0xFD	0x01	0x01	0x02	0x30	1 byte	0x00

Response:

Data field	Status Word
-	SW1 SW2

9.6.7. READ 2 BLOCKS (8 Bytes)

Reads 8 bytes from specified block.

Command:

CLA	INS	P1	P2	Lc	opcode	Block No	Le
0xFF	0xFD	0x01	0x01	0x02	0x31	1 byte	0x00

Response:

Data field	Status Word

8 bytes of data	SW1 SW2

9.6.8. DECREMENT

Decrements counter value of my-d card.

Command:

CLA	INS	P1	P2	Lc	opcode	Decrement Value	Le
0xFF	0xFD	0x06	0x01	0x03	0xD0	2 bytes	0x00

Response:

Data field	Status Word
-	SW1 SW2

9.6.9. HID transport protocol

This is applicable to uTrust 3721 F only. uTrust 3721 F reader supports HID transport protocol besides the CCID transport protocol. The HID transport protocol allows to read information out of card, perform limited formatting and allows it to be fed into host as keystrokes.

10. uTrust 4x01 and 4x11 Family

10.1. uTrust 4x01

It is a versatile 13.56 MHz contactless and contact reader/writer module, offering state-of-the-art interoperability for easy application integration. Future-proof and in-field upgradeable, it protects your investment while combining contact, contactless, and NFC technologies in a single device. Ideal for OEMs, system integrators, and VARs, enables rapid deployment of solutions for security, network log-ons, secure web transactions, and NFC-based customer loyalty programs.

10.2. uTrust 4x11

It is smart card reader/writer module that combines contactless and NFC technologies in a single device, with the option for enhanced security through an onboard SAM socket. This easy-to-integrate solution accelerates time-to-market for OEMs, system integrators, and VARs, enabling the development of innovative solutions for contactless security, network log-ons, secure web transactions, and NFC-based identification.

10.3. CNTLESS_P2P Specific Commands

10.3.1. CNTLESS_P2P_SWITCH_MODES

This Escape command is used to switch the device between the reader/writer and P2P modes of operation and to query the current mode. By default, the device is in the reader/writer mode.

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. Additional data bytes will be needed for Initiator/Target mode.

Offset	Description	Detailed description
0	0xE9	Switch mode
1	0 – P2P Initiator mode 1 – P2P Target mode 2 – Reader / writer mode 0xFF – Get current mode	For the switch to Initiator / Target mode, the bytes from offset 0x02 give additional information as described below

Offset	Initiator Mode Bytes	Detailed description
2		RFU
3		RFU
4		Timeout Low Byte
5		Timeout High Byte
6	N	Number of General Bytes
7 to N+7	General bytes to be sent in ATR-REQUEST	

Offset	Target Mode Bytes (Sample Values)	Detailed description
2	0x00	RFU
3	0x00	RFU
4	0x04	SENS_RES
5	0x03	SENS_RES
6	0x01	NFCID1
7	0xFE	NFCID1
8	0x0F	NFCID1
9	0x40	SEL_RES
10	0x01	NFCID2
11	0xFE	NFCID2
12	0x0F	NFCID2
13	0xBB	NFCID2
14	0xBA	NFCID2
15	0xA6	NFCID2
16	0xC9	NFCID2
17	0x89	NFCID2
18	C0	FeliCa Padding Bytes

19	C1	FeliCa Padding Bytes
20	C2	FeliCa Padding Bytes
21	C3	FeliCa Padding Bytes
22	C4	FeliCa Padding Bytes
23	C5	FeliCa Padding Bytes
24	C6	FeliCa Padding Bytes
25	C7	FeliCa Padding Bytes
26	FF	FeliCa System Code
27	FF	FeliCa System Code
28	0x00	NFCID3 (XOR of 0x08 and 3bytes of NFCID1)
29	0x88	Timeout Low Byte
30	0x13	Timeout High Byte
31	N	Number of G bytes in ATR_RES
32 to N+32	General byte to be sent in ATR_RES	

Output Buffer:

- Initiator Mode: On successful detection of target, the entire ATR_RES buffer from the target device would be given to the host computer
- Target Mode: On successful detection by the initiator the entire ATR_REQ buffer from the initiator device would be given to the host computer
- Reader Mode: The output buffer would be empty
- Get Current Mode: A single byte response indicating the currently selected mode as described below
    - 0x00 => P2P Initiator mode
    - 0x01 => P2P Target mode
    - 0x02 => Reader / Writer mode

10.3.2. CNTLESS_P2P_TARGET_RECEIVE

This Escape command is used to send data to an initiator device. Prior to using this command, the device should have been successfully switched to target mode using CNTLESS_P2P_SWITCH_MODES (E9).

Input:

Offset	Initiator Mode Bytes	Detailed description
0	0xEA	Target Receive
1		RFU
2		RFU
3		RFU
4	0 – No Chaining 1 – Chaining	Chaining byte
5	--	Timeout Low Byte

6	--	Timeout High Byte

Output: On successful reception, the entire data from the initiator device would be returned from offset 0x04

Offset	Initiator Mode Bytes	Detailed description
0		RFU
1		RFU
2		RFU
3	0 – No Chaining 1 – Chaining	Chaining
Offset 4 to offset 4+N	N data bytes	Bytes Received

10.3.3. CNTLESS_P2P_TARGET_SEND

This Escape command is used to send data to an initiator device. Prior to using this command, the device should have been successfully switched to target mode using CNTLESS_P2P_SWITCH_MODES (E9).

Input:

Offset	Initiator Mode Bytes	Detailed description
0	0xEB	Target Send
1	0x00	RFU
2	0x00	RFU
3	0x00	RFU
4	0 – No Chaining 1 – Chaining	Chaining byte
5		Timeout Low Byte
6		Timeout High Byte
Offset 7 to offset 7+N	N data bytes	Bytes to be sent to initiator device

Output: Once the data bytes are sent successfully, the firmware would indicate if it is ready to send more bytes through the chaining byte

Offset	Initiator Mode Bytes	Detailed description
0		RFU
1		RFU
2		RFU
3	0 – No Chaining 1 – Chaining	Chaining

10.3.4. CNTLESS_P2P_INITIATOR_DESELECT

This escape command is used by the application to deselect the target device towards the end of P2P communication.

Input:

Byte 0

Escape code(0xE6)

Output: The deselect response as received from the target will be sent in the response buffer from offset 0x00

10.3.5. CNTLESS_P2P_INITIATOR_TRANCEIVE

This Escape command is used to send data to a target device. Prior to using this command, the device should have been successfully switched to initiator mode using CNTLESS_P2P_SWITCH_MODES (E9).

Input:

Offset	Initiator Mode Bytes	Detailed description
0	0xE7	Initiator transceive
1	0x00	RFU
2	0x00	RFU
3	0x00	RFU
4	0 – No Chaining 1 – Chaining	Chaining
5	--	Timeout Low Byte
6	--	Timeout High Byte
Offset 7 to 7+N	N bytes of data	Bytes to be sent to target device

Output: On successful reception of data from the target, the entire data would be available from offset 0x04. Presence of additional data is indicated by the chaining byte.

Offset	Initiator Mode Bytes	Detailed description
0		RFU
1		RFU
2		RFU
3	0 – No Chaining 1 – Chaining	Chaining
Offset 4 to offset 4+N	N data bytes	Bytes Received

10.4. CNTLESS_NFC Specific Commands

10.4.1. CNTLESS_NFC_SINGLESHOT

This Escape command is used to switch the device to Single-shot mode.

Input:

Offset	Description	Detailed description
0	0xEC	NFC Single-shot

1	0x01	NFC_DEP supported. If a value other than 0x01 is given, NFC_DEP is not supported in the preceding I-Blocks.

Response:

Output buffer
NULL

10.4.2. CNTLESS_NFC_LOOPBACK

This Escape command is used to switch the device to Loop-back mode.

Input:

Offset	Description	Detailed description
0	0xED	NFC Loop-back
1	0x01	NFC_DEP supported. If a value other than 0x01 is given, NFC_DEP is not supported in the preceding I-Blocks.

Response:

Output buffer
NULL

10.4.3. CNTLESS_GET_SET_NFC_PARAMS

This Escape command is supported through the READER_GENERIC_ESCAPE command. During NFC operation, number parameters like DID, LRi, PSL_REQ_BRS and PSL_REQ_FSL can be controlled from application.

Input:

To set the parameters the command syntax is:

Byte 0 CLA	Byte 1 INS	Byte 2 P1	Byte 3 P2	Byte 4 Lc	Byte 5	Byte 6	Byte 7	Byte 8	Le
0xFF	0x70	0x04	0xE6	0x04	0x04 (opcode)	0x01 - SET	NFC Parameter	Value	00

To get the parameters the command syntax is:

Byte 0 CLA	Byte 1 INS	Byte 2 P1	Byte 3 P2	Byte 4 Lc	Byte 5	Byte 6	Byte 7	Le

0xFF	0x70	0x04	0xE6	0x03	0x04 (opcode)	0x01 - GET	NFC Parameter	00

The value of byte 7 is interpreted from this table

Byte 7 Value	Description
0x00 - DID	Device Identification Number
0x01 - LRi	Length Reduction field
0x02 - PSL_REQ_BRS	BRS used in PSL_REQ
0x03 - PS_REQ_FSL	FSL used in PSL_REQ

10.4.4. CNTLESS_GET_P2P_EXTERNAL_RF_STATE

This Escape command is supported through the READER_GENERIC_ESCAPE message. This command is used to check if external RF is reset after the reader got detected in target mode.

Input:

To set the parameters the command syntax is:

Byte 0 CLA	Byte 1 INS	Byte 2 P1	Byte 3 P2	Byte 4 Lc	Byte 5	Le
0xFF	0x70	0x04	0xE6	0x01	0x06 (opcode)	00

Output: If the command is successful, a single byte is returned. This byte indicates the value of parameter.

Bit 0 => Set to logic 1, when a present external RF field is switched off

Bit 1 => Set to logic 1, when an external RF field is detected.

Bit 2 to Bit 7 => RFU bits always read as 0

10.5. CONTACT_EMV Specific Commands

10.5.1 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 shall contain the function to be performed. Third byte shall contain the parameter for the function.

Byte 0	Byte 1		Byte 2
	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:

Byte 0 Value	Description	Byte 1	Byte 2
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:

Byte 0 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 the 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

10.5.2. 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.

Byte 0
Escape code (0x05)

Output:

Output buffer
NULL

10.5.3. CONTACT_EMV_SINGLEMODE

This Escape command lets the host perform a one-shot EMV Loop-back application as specified in the EMV Level 1 Testing Requirements document.

Input:

Byte 0
Escape code (0x06)

Output:

Output buffer
NULL

10.5.4. CONTACT_EMV_TIMERMODE

This Escape command lets the host 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.

Byte 0
Escape code (0x07)

Output:

Output buffer
NULL

11. uTrust 5501 F

The uTrust 5501 F module offers versatile contact and multi-frequency/multi-ISO/IEC contactless interface capabilities, supporting a range of identification applications. It supports contact smart card personalization (synchronous and asynchronous), as well as contactless high-frequency 13.56 MHz cards (ISO/IEC14443, ISO/IEC15693, FeliCa™) and low-frequency 125 kHz cards. For enhanced security, the optional Secure Access Module (SAM) slot provides secure authentication, key management, and cryptographic functions. The SAM supports secure communication with proprietary card technologies like iClass™ and iClass Seos™. ECP Support paired with FCC and IC Modular certifications, the uTrust 5501 F module enables seamless integration into various products without costly re-certification. It ensures flexibility and secure operations across a wide range of applications.

11.1. 5501F Specific Commands

11.1.1. 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.

Byte 0	Byte 1	
	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

11.1.2. 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:

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.

Byte 0 CLA	Byte 1 INS	Byte 2 P1	Byte 3 P2	Byte 4 Lc	Byte 5	Byte 6	Byte 7	Le
0xFF	0x70	0x04	0xE6	0x04	0x11	0x01 → SET	0x01 → Enable 0x00 → Disabled	00
						0x00 → GET		

Output:

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

11.1.3. 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 the LED number (if more than one LED is present, otherwise it is set to 0), and then the desired LED state. This will be required for production purposes.

Byte 0	Byte 1	Byte 2

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

Output:

Output buffer
NULL

11.1.4. 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.

Byte 0	Byte 1	
	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, a 1-byte response is received.

Output buffer
Current state

12. Common APDU and Escape Command Reference

These APDUs are supported across all Hirsch uTrust readers for basic smart card operations (contact and contactless):

12.1. Generic APDU

12.1.1. PAPDU_GET_UID

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

Command:

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 ISO14443A single 4 bytes, double 7 bytes, triple 10 bytes, for ISO14443B 4 bytes PUPI).

Response:

Data	Status Word
Requested bytes of UID	SW1, SW2

12.1.2. PAPDU_ESCAPE_CMD

Usually escape commands are transmitted through SCardControl as defined in PCSC API using IOCTL_CCID_ESCAPE. But on 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:

CLA	INS	P1	P2	Lc	Data in	Le
0xFF	0xCC	0x00	0x00	Length of data	Escape Command Buffer	XX

Response:

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: 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 would 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 having any need to prepend “FF CC 00 00 P3”

12.2. Working with DESFire and MIFARE Plus tokens

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

Please note that, since these application notes contain information available only under NDA with NXP, you’d need to sign an NDA with NXP to be allowed to receive them.

12.3. Generic Reader Control APDU

12.3.1. 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

Byte 0
Escape code (0x12)

Output:

The reader returns its PID LSB first.

PID value		Description
B0	B1	
0x26	0x57	Identiv uTrust 4701 F Dual Interface Reader
0x25	0x57	Identiv uTrust 4711 F Contactless + SAM Reader
0x10	0x57	Identiv uTrust 2700 F Smart Card Reader
0x50	0x57	Identiv uTrust 2910 F Smart Card Keyboard Reader

12.3.2. READER_LED_CONTROL

This Escape command is used to toggle the LED state. LED control by firmware should be disabled using the 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, else set to 0) and then desired LED state. This will be required for production purpose.

Byte 0	Byte 1	Byte 2
Escape code (0x19)	LED number (0-RED, 1-GREEN)	LED state (0-OFF, 1-ON)

Output:

Output buffer
NULL

12.3.3. READER_GET_INFO_EXTENDED

This Escape command is used to get the firmware version, reader capabilities, and Unicode serial number of the reader.

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

Byte 0
Escape code (0x1E)

Output:

The firmware will return data as per structure SCARD_READER_GETINFO_PARAMS_EX mentioned below.

Field Size in 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 of the reader. 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 Received as MSB first

12.3.4. 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. The output buffer is NULL.

Byte 0	Byte 1	
	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
NULL or current state

12.3.5. 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.

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	From Byte 5 (up to Lc bytes)		Byte Lc+5
					Byte 5	Byte 6 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 word SW1, SW2. In case of success, SW1=0x90 and SW2=0x00 shall be returned. In error scenario, appropriate error status shall be returned (as defined in Error Code section 8.0).

12.4. Contactless Control & Info Commands

12.4.1. CNTLESS_GET_CARD_INFO

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

Input:

The first byte of input buffer contains the escape code.

Byte 0
Escape code (0x11)

Output:

Byte 0	Byte 1	Byte 2
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	**Description**
0	Type A card
1	Type B card

12.4.2. 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.

Byte 0
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

12.4.3. READER_CNTLESS_GET_TYPE

This escape command retrieves the type of cards which the reader is configured to poll for. The input buffer shall contain the escape command code in the first byte and an optional extension specifier 0xFF in the second byte.

Byte 0	Byte 1
Escape code (0x93)	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 point to a WORD buffer in case the extension specifier is given and will contain the type value coded as bitmask as

Cards-Type-Bit Mask (Lo Byte)								
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

The Hi Byte will always be 0x00 (RFU).

12.4.4. READER_CNTLESS_SET_TYPE

This escape command configures the type of cards the reader will poll for. 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, we should not disable the polling for a given type of the card, after having placed that card on the reader. Since reader would immediately stop polling for the card, it would never detect that card is removed. The input buffer shall contain two or three bytes

Byte 0	Byte 1	Byte 3	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 (Lo Byte)

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

The Hi Byte will always be 0x00 (RFU).

The output buffer is

Output buffer
NULL

12.4.5. 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
Byte 0	Byte 1 – PPS control byte		Byte 0
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

12.4.6. 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.

Input			Output
Byte 0	Value	Description	Byte 0
Escape code (0x96)	0x00	Enable	No Output
	0x01	Disable	No Output
	0xFF	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

12.4.7. CNTLESS_CONTROL_848

This Escape command can be used to enable/disable 848kbps support as well as query whether 848kbps is currently enabled or disabled. The RF communication with a user token will only switch to 848Kbps if the user token supports this baud rate and provided automatic PPS is ON.

The input buffer shall contain 2 bytes

Byte 0	Byte 1	Description
0x9D	0x00	Disable 848Kbps support
	0x01	Enable 848Kbps support
	0xFF	Get current status on 848Kbps support

If B1 of the input buffer is 0x00 or 0x01 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	848Kbps is disabled
0x01	848Kbps is enabled

12.4.8. 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.

Byte 0
Escape code (0x9E)

Output:

The output contains a byte with the following possible values

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

12.4.9. 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
Byte 0	Byte 1 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

12.4.10. 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
Byte 0	Byte 1 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

12.4.11. 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 – b3 - DR=8 supported, if bit is set to 1 b4 – b5 - shall be set to 0, 1 is RFU b6 – b7 - DS=2 supported, if bit is set to 1 DS=4 supported, if bit is set to 1 b7 – e DS=8 supported, if bit is set to 1 1 if the same D is required for both communication directions 0 if different D is supported for ach 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 =x00
	NULL	If B1=0x00

The output buffer is

Output buffer
NULL

12.4.12. 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.

Byte 0
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		

B18		
B19		
B20		
OR		
B3-B10		8 Bytes NFCID2
B11		Request service command response time parameter (see JIS6319)
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

12.4.13. 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.

Byte 0
Escape code (0xE4)

Output:

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

12.4.14. 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 the escape code followed by 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.

Byte 0	Byte 1 onwards
Escape code (0xE4)	FeliCa command bytes

Output:

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

12.5. MIFARE Classic / MIFARE Ultralight Family

12.5.1. 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:

CLA	INS	P1	P2	Lc	Data in	Le
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:

Data	Status Word
N bytes of block data	SW1, SW2

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)

12.5.2. 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:

CLA	INS	P1	P2	Lc	Data in	Le

0xFF	0xD6	Addr MSB	Addr LSB	XX	data	-

For a description of P1 and P2, see PAPDU_MIFARE_READ_BINARY Lc has got to match the block size of the used card 16 bytes for Mifare Classic 4 bytes for Mifare UL and UL C

Response:

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 has got to 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)

12.5.3. LOAD_KEYS

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

Command:

CLA	INS	P1	P2	Lc	Data in	Le
0xFF	0x82	0x00	Key Num	Key data	Key	-

Response:

Data	Status Word
-	SW1, SW2

Examples

Load Keys

The command to load Mifare key A “FF FFFFFFFFFF” is FF82006006 FFFFFFFFFFFF

12.5.4. 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:

CLA	INS	P1	P2	Lc	Data in	Le

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
	0x00	Use key from non-volatile storage
B4	0x01	when B3=0x60 or B3=0x61
		Key number of key from non-volatile storage when B3=0

For authentication with MIFARE ULC using a key loaded into non-volatile storage use all 0x00 for data.

Response:

Data	Status Word
-	SW1, SW2

Example:

1. 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)

2. Authenticate with MIFARE ULC using key loaded into non-volatile storage:

APDU: FF 86 00 00 05 00 00 00 00 00
SW12: 9000 (OK)

12.5.5. 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:

CLA	INS	P1	P2	P3	Data

0xFF	0xB1	Addr MSB	Addr LSB	0	-

Response:

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 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 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 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 00 00 02 42 54 FE 00 (64 bytes)

12.5.6. 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:

CLA	INS	P1	P2	P3	Data
0xFF	0xB3	Addr MSB	Addr LSB	0	-

Response:

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 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 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)

12.5.7. 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 in case of Mifare Classic).

Command:

CLA	INS	P1	P2	P3	Data
0xFF	0xD7	Addr MSB	Addr LSB	Lc	Data

Lc (P3) has got to 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 has got to be 0x30 for Mifare UL and the data will get written from block 4 till the end of the memory.

Response:

Data	Status Word
-	SW1, SW2

12.5.8. VALUE_BLK_OLD

This command increments or decrements the data in a Value Block on a Mifare Classic card.

Command to Increment / Decrement OLD:

CLA	INS	P1	P2	P3	Data
FF	F0	00	Block Num	Lc	Data

Where P2 codes the block number.

The data field is structured as follows

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

Response:

Data	Status Word
-	SW1, SW2

Example: decrement block 4 by 1 (key loading and authentication not shown) (block 4 has got to be set up as 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)

12.5.9. 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 to Increment / Decrement:

CLA	INS	P1	P2	Lc	Data	Le
FF	C2	00	03	XX	BERTLV	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 to increment or decrement Destination by, 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:

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

Error Status	Description
XX SW1 SW 2	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 value
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

12.5.10. 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:

CLA	INS	P1	P2	P3	Data
0xFF	0xFE	00	00	Lc	Data

Response:

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

12.5.11. 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:

CLA	INS	P1	P2	P3	Data
0xFF	0xEF	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:

Data	Status Word
Data returned by card	SW1, SW2

12.5.12. ISO14443_PART4_PART3_SWITCH (TCL – Mifare Switch)

This command switches the card state between TCL and MIFARE modes

Command:

CLA	INS	P1	P2	P3	Data
0xFF	0xF8	P1	00	00	-

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

Response:

Data	Status Word
-	SW1, SW2

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

12.6. FeliCa (Sony)

12.6.1. FELICA_REQC

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

Command:

CLA	INS	P1	P2	P3	Data
0xFF	40	00	00	04	2 bytes of system code, 1 byte RFU, 1 byte TSN

Response:

Data	Status Word
16 bytes of NFCID2 + 2 bytes of System Code (sent only if the RFU byte is 0x01)	SW1, SW2

12.6.2. 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:

CLA	INS	P1	P2	P3	Data
0xFF	42	Number of services/areas	00	2 * P1	Service Code List / Area Code List

Response:

Data	Status Word
8 bytes IDm + No. of Service or areas(n) + Service version or area version list (2*n)	SW1, SW2

12.6.3. 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:

CLA	INS	P1	P2	P3	Data
0xFF	44	00	00	00	-

Response:

Data	Status Word
8 bytes IDm + Mode	SW1, SW2

12.6.4. 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:

CLA	INS	P1	P2	P3	Data
0xFF	46	Number of service	Number of blocks	2 * (P1 + P2)	Service Code List, Block List

Response:

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

12.6.5. 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:

CLA	INS	P1	P2	P3	Data

0xFF	48	Number of service	Number of blocks	2*(P1 + P2) + (16 * P2)	Service Code List, Block List, Block Data

Response:

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

12.6.6. FELICA_SYS_CODE

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

Command:

CLA	INS	P1	P2	P3	Data
0xFF	4A	00	00	00	-

Response:

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

12.7. NFC Type 1 Tag

12.7.1. NFC_TYPE1_TAG_RID

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

Command:

CLA	INS	P1	P2	P3	Data
0xFF	50	00	00	00	-

Response:

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

Where

- 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.

12.7.2. 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:

CLA	INS	P1	P2	P3	Data
0xFF	52	00	00	00	-

Response:

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

12.7.3. 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:

CLA	INS	P1	P2	P3	Data
0xFF	54	00	Byte Addr	00	-

Where 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:

Data	Status Word
Data returned by card	SW1, SW2

12.7.4. NFC_TYPE1_TAG_WRITE_E

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

Command:

CLA	INS	P1	P2	P3	Data

0xFF	56	00	Byte Addr	01	Data

Where 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:

Data	Status Word
Data returned by card	SW1, SW2

12.7.5. 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:

CLA	INS	P1	P2	P3	Data
0xFF	58	00	Byte Addr	01	Data

Where 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:

Data	Status Word
Data returned by card	SW1, SW2

12.7.6. 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:

CLA	INS	P1	P2	P3	Data

0xFF	5A	00	SegAddr	00	-

Where P2 Segment Address is:

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

Response:

Data	Status Word
128 bytes of data	SW1, SW2

12.7.7. 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:

CLA	INS	P1	P2	P3	Data
0xFF	5C	00	Block Addr	00	-

Where P2 Block Address is:

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

Response:

Data	Status Word
8 bytes of data	SW1, SW2

12.7.8. 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:

CLA	INS	P1	P2	P3	Data
0xFF	5E	00	Block Addr	08	Data

Where P2 Block Address is:

Bit numbers	Description

b7 – b0	General block (0x00 -0xFF)

Response:

Data	Status Word
8 bytes of data that have been written	SW1, SW2

12.7.9. 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:

CLA	INS	P1	P2	P3	Data
0xFF	60	00	Block Addr	08	Data

Where P2 Block Address is:

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

Response:

Data	Status Word
8 bytes of data	SW1, SW2

13. Escape Commands Specific to 47xx and 5501 F Family

13.1. Contact Control & Info Commands

13.1.1. 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 mostly is 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.

Byte 0	Byte 1 onwards
Escape code (0x08)	Short APDU to be sent to card

Output:

Output buffer
Response APDU

13.1.2. 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.

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

Output:

Output buffer
NULL

13.1.3. 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:

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.

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5 onwards
Escape code (0x10)	LSB of send length	MSB of send length	LSB of expected length	MSB expected length	Raw data to the card

Output:

Output buffer
Response APDU

13.1.4. CONTACT_GET_SET_CLK_FREQUENCY

This Escape command is used to instruct the reader to change the clock for the smart card or to get the current Clock divisor used. 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.

Byte 0	Byte 1	
	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:

Output buffer	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)

13.1.5. 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 would emit 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.

Byte 0	Byte 1	
	Value	Description
Escape code (0x88)	0x00	Enable ATR validation
	0x01	Disable ATR validation

Output:

Output buffer
NULL

13.1.6. 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.

Byte 0	Byte 1	
	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

Byte 0
NULL or Delay in ms

DataIn = 85 00

DataOut: 00 (1 byte)

13.1.7. 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 is following.

Byte 0	Byte 1		Byte 2	Byte 3	Byte 4	Byte 5
	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.

Byte 0	Byte 1	Byte 2	Byte 3
ETU value			
BIT31-BIT24	BIT23-BIT16	BIT15-BIT8	BIT7-BIT0

DataIn = 80 00

DataOut: 00 00 01 40 (4 bytes)

13.1.8. 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.

Byte 0	Byte 1		Byte 2		Byte 3	Byte 4	Byte 5	Byte 6
Escape code (0x81)	Value	Description	Value	Description	Wait time in ETU			
	0x01	SET time Wait	0x00	CWT	BIT31-BIT24	BIT23-BIT16	BIT15-BIT8	BIT7-BIT0
			0x01	BWT				
	0x00	GET time Wait	0x00	CWT	-	-	-	-
			0x01	BWT				

Output:

For both Get/Set Wait time, the output will be the following.

Byte 0	Byte 1	Byte 2	Byte 3
Wait time in ETU			

BIT31-BIT24	BIT23-BIT16	BIT15-BIT8	BIT7-BIT0

DataIn = 81 00 01

DataOut: 00 00 03 5D (4 bytes)

13.1.9. 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.

Byte 0	Byte 1		Byte 2		Byte 3	Byte 4	Byte 5	Byte 6
Escape code (0x82)	Value	Description	Value	Description	Guard time in ETU			
	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.

Byte 0	Byte 1	Byte 2	Byte 3
Character Guard time in ETU			
BIT31-BIT24	BIT23-BIT16	BIT15-BIT8	BIT7-BIT0

DataIn = 82 00 01

DataOut: 00 00 00 18 (4 bytes)

13.2. Reader Mode & Control Commands

13.2.1. 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 ISO7816, EMV, Memory card and NFC test mode operations. Upon power on the reader will reset to the default ISO7816 mode.

Input:

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

Byte 0	Byte 1
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 slot and contactless slot
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

13.2.2. READER_GETMODE

This Escape command retrieves the current mode of the reader.

Input:

The input buffer contains the escape code value.

Byte 0
Escape code (0x02)

Output:

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

Mode	Value	Remarks
ISO 7816	0x00	ISO 7816 mode – Applicable for both contact slot and contactless slot
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

13.2.3. 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 get detected in both “contact” and “contactless” mode. 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

Byte 0 CLA	Byte 1 INS	Byte 2 P1	Byte 3 P2	Byte 4 Lc	Byte 5	Byte 6	Byte 7	Le
0xFF	0x70	0x40	0xE6	0x03	0x05 (opcode)	0x01	0x00 – to enable	00
							0x01 – to disable	00
0xFF	0x70	0x40	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 it USB Vendor ID is 0x04E6

Output:

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

Byte 0	Description
0x00	Contact slot is enabled
0x01	Contact slot is disabled

14. Sample Code using Escape Commands

```
// File Name: uTrust 47xx F Escape.h
#ifndef _uTrust_47xxF_ESCAPE_H_
#define _uTrust_47xxF_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 47xx F Escape.c
#include <windows.h>
#include <winbase.h>
#include <stdio.h>
#include <conio.h>
```

```
#include "winscard.h"
#include "winerror.h"
#include "uTrust 47xxF 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 4701 F Contact Reader 0",
      "Identiv uTrust 4701 F 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();
}
```