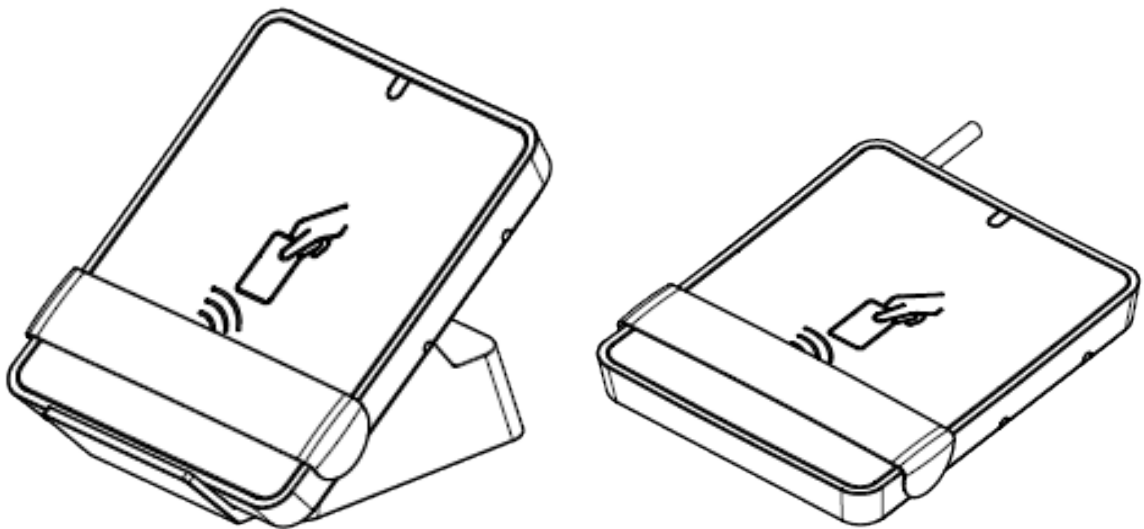




Reference Manual for the CLOUD 370x F Contactless Desktop Readers

For Part #: 905502 CLOUD 3700 F (global version) and 905503 CLOUD 3701 F (Japan-only version)



Abstract

This document contains in-depth information about the hardware and software features of the CLOUD 370x family of contactless desktop readers.

Audience

This document is intended for system integrators and software developers.

Revision History

Rev.	Date	Description
1.0	2014-07-22	First published external version, corresponding to firmware version 1.00

Contact Information

For additional information, please visit <http://www.identiv.com/>.

Table of Contents

1: LEGAL INFORMATION	6
1.1: Disclaimers	6
1.2: FCC Rules	6
1.2.1: Section 15.21 Information to User	6
1.2.2: Class B, Section 15, Subpart C, Section 15.225	6
1.3: Software Licenses	7
1.4: Trademarks	7
2: INTRODUCTION TO THIS DOCUMENT	8
2.1: Objective of this Document	8
2.2: Target Audience	8
2.3: Product Version Corresponding to this Document	8
2.4: Definition of Various Terms and Acronyms	9
2.5: References to Other Documents	10
2.6: Conventions for Bits and Bytes	11
3: GENERAL INFORMATION ABOUT CLOUD 370X F READERS	12
3.1: Key Benefits	12
3.2: Key Features	12
3.3: Ordering Information	13
3.4: Customization Options	14
3.5: Contactless Communication Principles and CLOUD 370x F Usage Recommendations	14
3.5.1: Power Source for the User Credential	14
3.5.2: Data Exchange	14
3.5.3: Recommendations	15
3.6: Applications	16
3.6.1: General	16
3.6.2: Applications Provided by Identiv	16
4: CHARACTERISTICS OF THE CLOUD 370X F READERS	17
4.1: High-Level Architecture	17
4.1.1: Block Diagram	17
4.1.2: Software Architecture	18
4.2: Quick Reference Data	19
4.2.1: Dimensions of the CLOUD 370x F Readers	19

4.2.2: LED Behavior	20
4.2.3: Other Data.....	21
5: SOFTWARE MODULES	23
5.1: Installation	23
5.2: Utilities.....	23
5.3: Driver	23
5.3.1: Device Description Text	23
5.3.2: Supported Operating Systems.....	23
5.3.3: PC/SC 2.0 Compliant Answer To Reset (ATR) for Contactless Interface	24
5.4: Firmware.....	27
5.4.1: CCID Transport Protocol	27
6: COMMAND DESCRIPTIONS	28
6.1: Generic Application Protocol Data Unit (APDU).....	28
6.1.1: Working with DESFire and MIFARE Plus Tokens	28
6.1.2: PAPDU_GET_UID.....	28
6.1.3: PAPDU_ESCAPE_CMD	28
6.2: Supported Pseudo APDU	30
6.2.1: PAPDU_MIFARE_READ_BINARY.....	30
6.2.2: PAPDU_MIFARE_UPDATE_BINARY	31
6.2.3: PAPDU_MIFARE_LOAD_KEYS.....	32
6.2.4: PAPDU_MIFARE_AUTHENTICATE.....	34
6.2.5: PAPDU_MIFARE_READ_SECTOR	35
6.2.6: PAPDU_MIFARE_READ_SECTOR_EX	36
6.2.7: PAPDU_MIFARE_WRITE_SECTOR.....	36
6.2.8: PAPDU_MIFARE_VALUE_BLK_OLD	37
6.2.9: PAPDU_MIFARE_VALUE_BLK_NEW	38
6.2.10: PAPDU_TCL_PASS_THRU (T=CL Pass Thru)	39
6.2.11: PAPDU_ISO14443_PART3_PASS_THRU (MIFARE Pass Thru).....	40
6.2.12: PAPDU_ISO14443_PART4_PART3_SWITCH (TCL – MIFARE Switch).....	40
6.2.13: PAPDU_FELICA_REQC.....	41
6.2.14: PAPDU_FELICA_REQ_SERVICE.....	41
6.2.15: PAPDU_FELICA_REQ_RESPONSE.....	42
6.2.16: PAPDU_FELICA_READ_BLK.....	42
6.2.17: PAPDU_FELICA_WRITE_BLK	43
6.2.18: PAPDU_FELICA_SYS_CODE.....	43

6.2.19: <i>PAPDU_NFC_TYPE1_TAG_RID</i>	44
6.2.20: <i>PAPDU_NFC_TYPE1_TAG_RALL</i>	44
6.2.21: <i>PAPDU_NFC_TYPE1_TAG_READ</i>	45
6.2.22: <i>PAPDU_NFC_TYPE1_TAG_WRITE_E</i>	45
6.2.23: <i>PAPDU_NFC_TYPE1_TAG_WRITE_NE</i>	46
6.2.24: <i>PAPDU_NFC_TYPE1_TAG_RSEG</i>	47
6.2.25: <i>PAPDU_NFC_TYPE1_TAG_READ8</i>	47
6.2.26: <i>PAPDU_NFC_TYPE1_TAG_WRITE_E8</i>	48
6.2.27: <i>PAPDU_NFC_TYPE1_TAG_WRITE_NE8</i>	48
6.3: <i>Escape Commands for the CLOUD 370x F</i>	49
6.3.1: <i>Sending Escape Commands to CLOUD 370x F</i>	49
6.3.2: <i>Escape Command Codes</i>	49
6.3.3: <i>Reader-Specific Escape Commands</i>	50
6.3.4: <i>Specific Escape Commands for Contactless Interface</i>	57
7: APPENDIXES	77
7.1: <i>Appendix A: Status Words table</i>	77
7.2: <i>Appendix B: Sample Code Using Escape Commands</i>	78
7.3: <i>Appendix C: Mechanical Drawings</i>	81
7.3.1: <i>Reader (without stand)</i>	81
7.3.2: <i>Reader on Stand</i>	82

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 the application that you may want to use them for.

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

1.2: FCC Rules

1.2.1: Section 15.21 Information to User

Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

1.2.2: Class B, Section 15, Subpart C, Section 15.225

NOTE: This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation.

If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

1.3: Software Licenses

Where this 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 this document in applications you may develop. You may distribute those applications **ONLY** in the form of compiled applications.
- You may **NOT** copy or distribute any portion of the source code without prior written consent from Identiv.
- You may **NOT** combine or distribute the source code provided in this document 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 products, they are provided for documentation purposes only. Identiv does not grant you any license to its designs.

1.4: Trademarks

MIFARE™ is a registered trademark of NXP Semiconductors BV.

Windows is a trademark of Microsoft Corporation.

2: Introduction to this Document

2.1: Objective of this Document

This document provides an overview of the hardware and software features of the CLOUD 370x F Multiprotocol contactless desktop readers (CLOUD 3700 F and CLOUD 3701 F).

This manual describes in detail the interfaces and supported commands available for developers using CLOUD 370x F in their applications.

2.2: Target Audience

This document describes the technical implementation of CLOUD 370x F, and targets software developers. It assumes knowledge about ISO 7816, 13.56 MHz contactless technologies like ISO/IEC 14443, and commonly used engineering terms.

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

2.3: Product Version Corresponding to this Document

Product Component	Version
Hardware	0.1
Firmware	1.00

2.4: Definition of Various Terms and Acronyms

Term or Acronym	Definition
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 from the reader to the card.
DS	Divider Send: used to determine the baud rate from 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
NFC	Near Field Communication: a set of standards for smartphones (and similar devices) to establish radio communication with each other by touching them together or bringing them into close proximity.
nibble	Group of 4 bits. 1 digit of the hexadecimal representation of a byte. <i>Example:</i> 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 personal computer and a smart card.
PICC	Proximity Integrated Chip Card
PID	Product ID
proximity	Distance coverage till ~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.

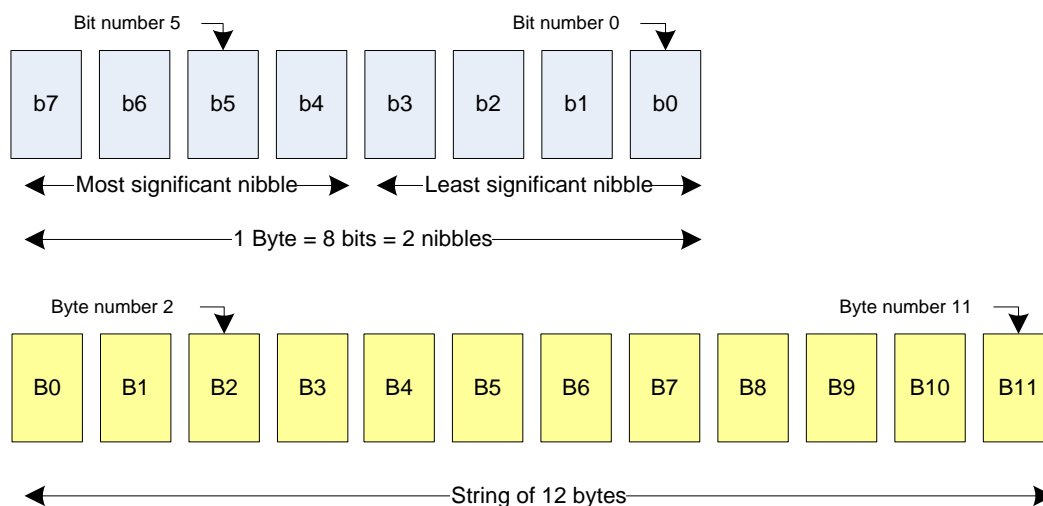
2.5: References to Other Documents

Short reference used in this document	Description of the other referenced document	Document issuer
ISO/IEC 7816-3	Identification cards — Integrated circuit cards with contacts — Part 3: Cards with contacts — Electrical interface and transmission protocols	ISO/IEC
ISO/IEC 7816-4	Identification cards — Integrated circuit 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 anticollision	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
JIS (also known as FeliCa)	Japanese Industrial Standard JIS X 6319-4: Specification of implementation for integrated circuit(s) cards, Part 4: High speed proximity cards	JICSAP (Japan IC Card System Application Council)
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 the handling of DESFire EV1 cards	Identiv
AN338	Application Note describing the handling of MIFARE Plus cards	Identiv

2.6: 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 its position.



Examples:

Decimal number 163 is represented in

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

The least significant nibble of 0xA3 is

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

The most significant nibble of 0xA3 is

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

3: General Information About CLOUD 370x F Readers

3.1: Key Benefits

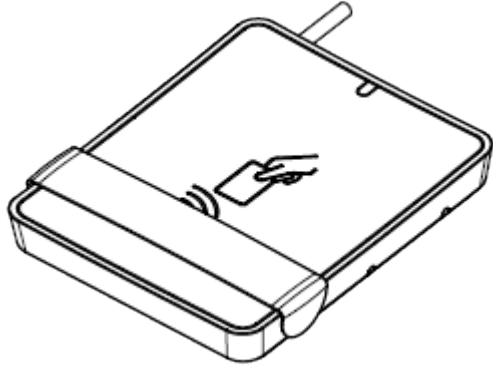

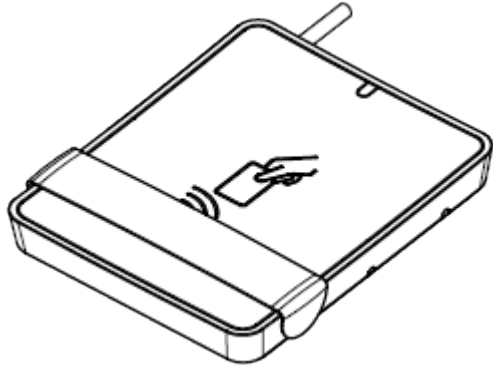

With its combination of a modern slim design and its state of the art multi-protocol feature set, a CLOUD 370x F desktop reader is the perfect choice to support various contactless applications such as electronic ID, payment, and public transportation schemes, and to interact with NFC-enabled devices.

The CLOUD 370x F readers are designed to offer best in class interoperability with various formats of tokens such as cards, dongles, watches, or NFC mobile phones.

3.2: Key Features

- 13.56 MHz contactless reader:
 - ISO14443 type A & B
 - MIFARE™
 - FeliCa™
 - Topaz (NFC Forum tag type 1)
 - Type B Innovatron protocol support (Calypso cards) (CLOUD 3700 F only)
 - NFC Peer-to-peer communication
- PC/SC v2.0 compliant
- Secure in-field firmware upgrade
- Unique reader serial number which enables a CLOUD 370x F reader to be plugged into any USB 2.0 slot on a PC without having to re-install the driver. Additionally, the application software running on the host can check for this serial number.
- Communication speed up to 848 Kbps for the CLOUD 3700 F reader (424 Kbps for the CLOUD 3701 F reader)

3.3: Ordering Information

Item	Part number	Product image
CLOUD 3700 F	905502	
CLOUD 3700 F with preassembled Stand foot kit	905502_5000	
CLOUD 3701 F	905503	
CLOUD 3701 F with preassembled Stand foot kit	905503_5001	

3.4: Customization Options

Upon request and based on a minimum order quantity, Identiv can consider customizing the reader's:

- Product label
- Identification text strings
- Logo
- Casing colors

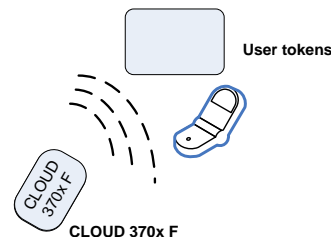
For more information about customizing this product, please contact your local Identiv representative or send an email to sales@identiv.com.

3.5: Contactless Communication Principles and CLOUD 370x F Usage Recommendations

The CLOUD 370x F is a contactless reader¹ designed to communicate with user tokens, which are also known as user credentials. User tokens² contain a contactless integrated circuit card connected to an antenna.

User tokens can take several form factors, including:

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



Communication between a CLOUD 370x F reader and user credentials uses magnetic field inductive coupling. The magnetic field generated by a CLOUD 370x F reader has a carrier frequency of 13.56 MHz.

3.5.1: Power Source for the User Credential

When the user credential is placed within the magnetic field of the reader, its antenna couples with the reader and an induction current appears in the credential's 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 for the integrated circuit microprocessor to function.

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, which you can refer to 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. Further details about this can be found in ISO/IEC 14443.

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 token is called proximity integrated chip card (PICC).

3.5.3: Recommendations

The communication between the reader and the user credential is sensitive to the presence of material or objects interfering with the magnetic field generated by the reader. The presence of electrically conductive materials such as 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 currents (also known as Foucault's currents) in the conductive materials; the field is literally absorbed by that kind of material.

- For proper communication, avoid putting your CLOUD 370x F reader in close proximity to conductive materials. A minimum distance of 2.5 cm (1") should be kept.
- To prevent the fields of adjacent readers from interfering with each other, a minimum distance between readers of 2.5 cm (1") should be kept.

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

- It is recommended that you present only one user credential at a time to a CLOUD 370x F reader.

The communication between the reader and the credential is sensitive to the geometry of the system {reader, 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. The CLOUD 370x F reader was designed and optimized to function with user credentials of various technologies and sizes. Even so:

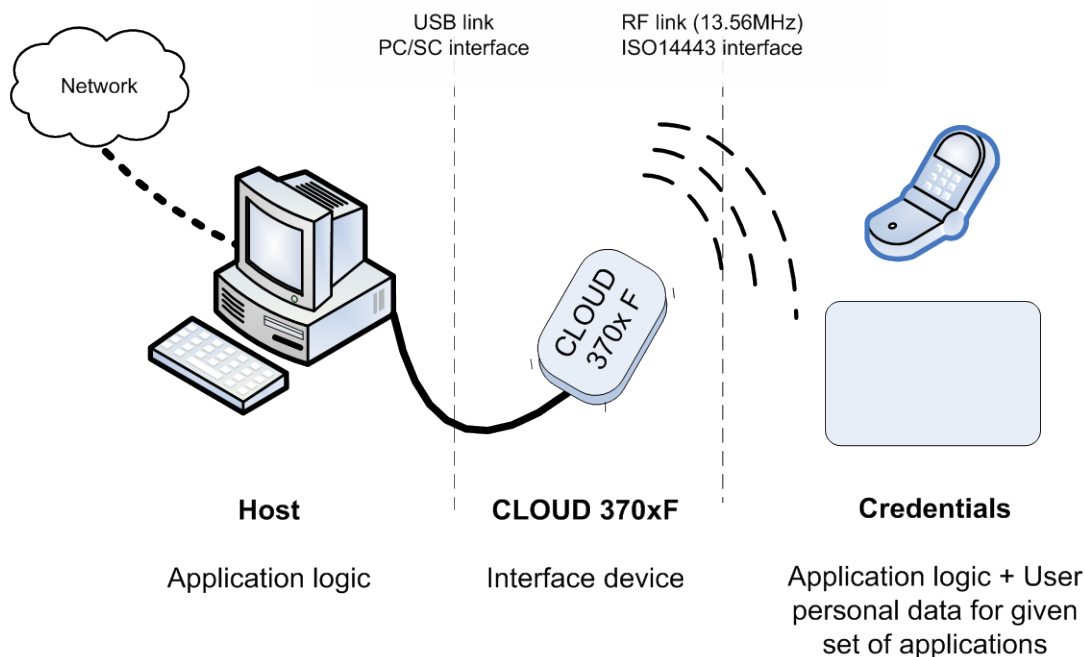
- A CLOUD 370x F reader might not be capable of communicating with extremely large or extremely small credentials.
- To optimize the coupling between the reader and the credential, put both antennas as parallel as possible to each other.
- To optimize transaction speed between the reader and the card, 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

The CLOUD 370x F reader is a transparent reader module that provides an interface between a personal computer host supporting the PC/SC interface, and 13.56 MHz user tokens like public transport cards, employee badges, loyalty cards, NFC enabled smart phones, etc.

User credentials can have several form factors including cards, key fobs, NFC mobile phones, or USB dongles (such as Identiv's SCTxxxx or @MAXX products).



The CLOUD 370x F reader handles the communication protocol, but not the application related to the credential or card. The application-specific logic has to be implemented on the host by software developers.

3.6.2: Applications Provided by Identiv

Identiv does not provide payment or transport applications. Identiv provides a few applications for development and evaluation purposes that can function with the CLOUD 370x F reader. There are many tools available; here are two of them:

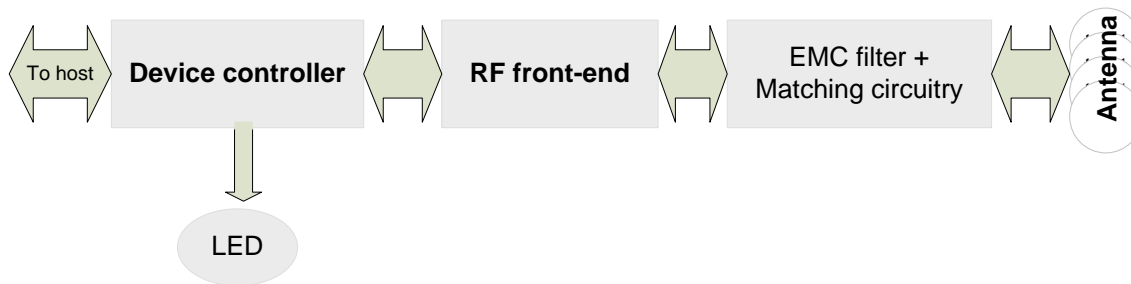
- The **Simple NFC Tag Editor** is part of our NFC-NDEF Editor Kit that enables the user to read and write NFC forum compliant records from/to NFC forum compatible tags. It is an easy to use tool to configure NFC forum tag demonstrations rapidly, and is available in our [Web shop](#).
- **Smart Card Commander** version 1.3 provides capabilities to identify most common cards in the field and display their content, as well as scripting functionality, which can be very useful for developers to develop and debug their applications. This tool is part of all our SDKs and is also available as a [stand-alone product](#).

4: Characteristics of the CLOUD 370x F Readers

4.1: High-Level Architecture

4.1.1: Block Diagram

The link between a CLOUD 370x F reader and the host computer to which it is connected is the standard Universal Serial Bus (USB) 2.0 interface, which provides both the power and the communication channel.



The device controller has several interfaces available. In the CLOUD 370x F implementation two peripherals are connected to the device controller:

- LED for reader status indication
- A radio frequency (RF) front-end that handles the RF communication

The device controller contains the firmware developed by Identiv to handle all the RF communication protocols and the PC/SC communication protocol with the host. The flash memory can be upgraded after the device controller has been deployed in the field, hence enabling firmware upgrades to add and patch features.

The RF front-end ensures the coding/decoding/framing modulation/demodulation required for the RF communication. It is managed 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.

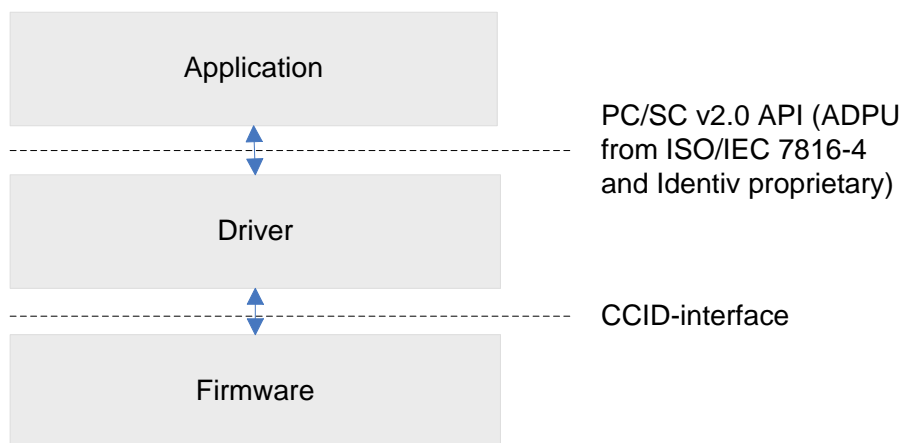
The CLOUD 370x F reader leverages a PC/SC CCID driver that is freely available for all supported operating systems (Windows, MacOSX, and Linux). This driver is already included in the basic installation of the current versions of Windows (starting with Windows Vista) and MacOSX.

Because there are so many variations of Linux, there may be distribution-specific drivers that need to be installed using the install mechanism of your particular variation.

If there is none, the driver may always be downloaded from the Web page of the maintainer, Ludovic Rousseau, https://alioth.debian.org/frs/?group_id=30105.

Alternatively, Identiv provides a proprietary driver for all the supported operating systems.

For Android-based systems, Identiv offers a CCID Library, which enables every application running on Android version 4.0 and higher to communicate with the reader using the CCID protocol.



4.2: Quick Reference Data

4.2.1: Dimensions of the CLOUD 370x F Readers

Item	Characteristic	Value
CLOUD 3700 F	Weight	44g / 0.097 lb \pm 5%
	External dimensions	91 x 75 x 13 mm / 3.583 x 2.953 x 0.472 in 86 x 77 x 69 mm / 3.386 x 3.031 x 2.717 in (with Stand foot kit)
	Default label	
CLOUD 3701 F	Weight	44g / 0.097 lb \pm 5%
	External dimensions	91 x 75 x 13 mm / 3.583 x 2.953 x 0.472 in 86 x 77 x 69 mm / 3.386 x 3.031 x 2.717 in (with Stand foot kit)
	Default label	

Drawings with dimensions of the CLOUD 370x F readers can be found in section 7.3 (at the end of this document).

4.2.2: LED Behavior

4.2.2.1: CLOUD 3700 F

The CLOUD 3700 F reader is equipped with a two-color LED, which has the following behavior:

Reader State	GREEN	RED
Just after plug-in (with drivers already installed)	ON	OFF
Just after device firmware upgrade	ON	OFF
Suspend / standby	OFF	OFF
Reader powered, and contactless card in reader's field but not yet powered	ON	ON
Contactless card powered and communicating with reader	500 ms ON 500 ms OFF	500 ms ON 500 ms OFF
Reader / card errors	OFF	100 ms ON 100 ms OFF
Dual interface card powered using RF field	500 ms ON 500 ms OFF	500 ms ON 500 ms OFF

4.2.2.2: CLOUD 3701 F

The CLOUD 3701 F reader is equipped with a one-color LED, which has the following behavior:

Reader State	GREEN
Just after plug-in (with drivers already installed)	ON
Just after device firmware upgrade	ON
Suspend / standby	OFF
Reader powered, and contactless card in reader's field but not yet powered	ON
Contactless card powered and communicating with reader	500 ms ON 500 ms OFF
Reader / card errors	100 ms ON 100 ms OFF
Dual interface card powered using RF field	500 ms ON 500 ms OFF

4.2.3: Other Data

4.2.3.1: General Information

Parameter	Value/Description
Application Programming Interface (API)	PC/SC 2.0
Operating temperature range	For CLOUD 3700 F: -20°C to 60°C (-4°F to 140°F) For CLOUD 3701 F: 0°C to 50°C (32°F to 122°F)
Storage temperature range	-25°C to 85°C (-13°F to 185°F)
Operating humidity range	Up to 95% relative humidity non-condensing
Certifications	<ul style="list-style-type: none"> • USB 2.0 • CE • FCC • UL • RoHS2 • REACH • WHQL

4.2.3.2: USB 2.0 Interface

Parameter	Value/Description
DC characteristics	High bus powered (CLOUD 370x F draws power from USB bus) Voltage: 5V Average Current for CLOUD 3700 F: 170 mA (RF on, no cards present) Average Current for CLOUD 3701 F: 50 mA (RF on, no cards present) Suspend Current: 600 µA
USB specification	USB 2.0 full speed device
USB speed	Full speed (12 Mbits per second)
Device class	CCID
Product ID value	0x5790 for a CLOUD 3700 F; 0x5791 for a CLOUD 3701 F
Vendor ID value	0x04E6

4.2.3.3: Contactless Interface

Parameter	Value/Description
RF carrier frequency	13.56 MHz +/-50 ppm
Supported card baud-rates	For the CLOUD 3700 F: 106 / 212 / 424 / 848 Kbps For the CLOUD 3701 F: 106 / 212 / 424 Kbps
Card types supported	<p>By the CLOUD 3700 F reader:</p> <ul style="list-style-type: none"> • 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 • Calypso CD21 • 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 Tablets¹ <p>By the CLOUD 3701 F reader:</p> <ul style="list-style-type: none"> • Lascom cards • 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 • Calypso CD21 • 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 Tablets¹
ISO-14443A and B compliant	Yes
Number of slots	1
Ejection mechanism	Manual

¹ Tested with a Google Nexus 5 smart phone running Android 4.4

5: Software Modules

5.1: Installation

On operating systems with a CCID driver pre-installed, no software installation is necessary. Where there's no CCID driver pre-installed (such as Linux systems or pre-Vista Windows systems), the driver has to be installed using distribution-specific measures or the available source packages.

Due to some limitations of the available CCID drivers under some circumstances, Identiv also provides a dedicated driver for this reader, which is available through Windows Update or on the [Identiv support pages](#).

5.2: Utilities

The following utilities are available:

- A tool for testing the resource manager (TestResMan)
- A tool called PCSCDiag, which is capable of providing basic information about the reader and a card through a PC/SC stack

5.3: Driver

5.3.1: Device Description Text

A CLOUD 370x F reader is described by PC/SC applications as either:

- *Identiv CLOUD 3700 F Contactless Reader*
- *Identiv CLOUD 3701 F Contactless Reader*

5.3.2: Supported Operating Systems

- Windows XP (32 & 64 bit)
- Windows 2003 Server (32 & 64 bit)
- Windows Vista (32 & 64 bit)
- Windows Server 2008 (32 & 64 bit)
- Windows 7 (32 & 64 bit)
- Windows 8 (32 & 64 bit)
- Windows Embedded 7 and higher
- MacOS X 10.5 and higher
- Linux 2.6 and higher (32 & 64 bit)
- Android 4.0 and higher (through CCID library)

5.3.3: PC/SC 2.0 Compliant Answer To Reset (ATR) for Contactless Interface

When a user credential is placed near the reader, initialization, anti-collision is done. The user credential is automatically activated and an Answer To Reset (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 to [PCSC3-SUP].

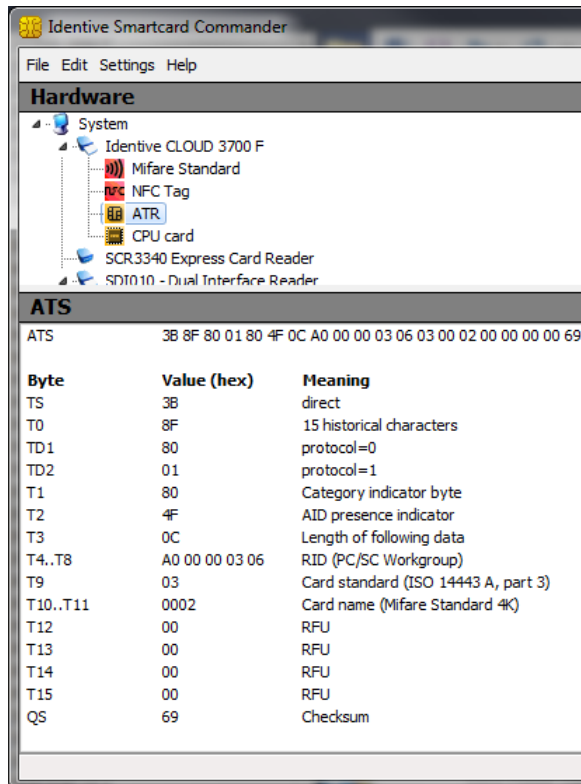
5.3.3.1: ATR for Contactless Storage User Tokens

The ATR of the user credential is composed as described in the following table. For the application to identify the storage card properly, its Standard and Card name describing bytes must be interpreted according to the Part 3 Supplemental Document, which is maintained by PC/SC. Credentials using technology like MIFARE are examples of this.

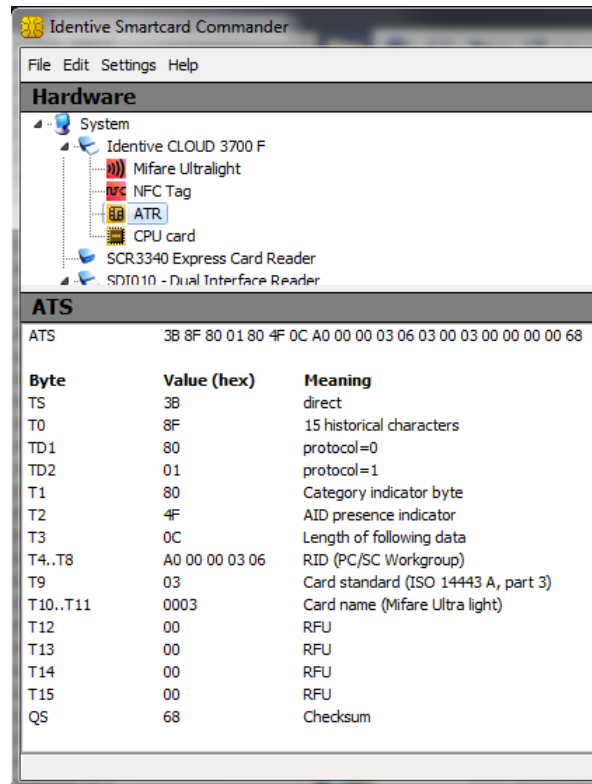
Byte#	Value	Designation	Description
0	0x3B	Initial header	
1	0x8n	T0	n indicates the number of historical bytes in the following ATR
2	0x80	TD1	upper nibble 8 indicates no TA2, TB2, TC2 lower nibble 0 means T=0
3	0x01	TD2	upper nibble 0 indicates no TA3, TB3, TC3 lower nibble 1 means T=1
4...3+n	0x80		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

Examples of the ATR built for contactless storage tokens:

MIFARE Classic 4K



MIFARE Ultralight



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

The user credential exposes its ATS or application information, which is mapped to an ATR. The following table describes the structure of this mapping.

Byte#	Value	Designation	Description
0	0x3B	Initial header	
1	0x8n	T0	n indicates the number of historical bytes in the 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): <ul style="list-style-type: none"> 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

Examples of the ATR built for an ISO14443-4 credential:

Type A

The screenshot shows the 'Identive Smartcard Commander' application. The 'Hardware' tree on the left lists 'System', 'Identive CLOUD 3700 F', 'ATR' (selected), 'CPU card', 'SCR3340 Express Card Reader', and 'SDIO10 - Dual Interface Reader'. The 'ATS' section displays the following data:

ATS 3B 89 80 01 4A 43 4F 50 33 31 56 32 32 4A

Byte	Value (hex)	Meaning
TS	3B	direct
T0	89	9 historical characters
TD1	80	protocol=0
TD2	01	protocol=1
Historical Hex	4A 43 4F 50 33 31 56 32 32	
Historical ASCII	JCOP31V22	
QS	4A	Checksum

Type B

The screenshot shows the 'Identive Smartcard Commander' application. The 'Hardware' tree on the left lists 'System', 'Identive CLOUD 3700 F', 'ATR' (selected), 'CPU card', 'SCR3340 Express Card Reader', and 'SDIO10 - Dual Interface Reader'. The 'ATS' section displays the following data:

ATS 3B 88 80 01 00 00 00 00 11 81 B1 00 28

Byte	Value (hex)	Meaning
TS	3B	direct
T0	88	8 historical characters
TD1	80	protocol=0
TD2	01	protocol=1
Historical Hex	00 00 00 00 11 81 B1 00	
Historical ASCII	00000±0	
QS	28	Checksum

5.4: Firmware

5.4.1: CCID Transport Protocol

The CLOUD 370x F reader 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 section 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 contactless 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_Escape
- PC_to_RDR_Abort

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.

The following table summarizes when and why some of the main error codes are returned:

CCID Error Code	Brief Description
HW_ERROR	Returned when any internal hardware error is detected.
XFR_PARITY_ERROR	Returned when a parity error condition is detected. This error is reported in the response to a PC_to_RDR_XfrBlock message.
ICC_MUTE	Returned when the card does not respond until the reader time out occurs. This error is reported in the response to the PC_to_RDR_XfrBlock and the PC_to_RDR_IccPowerOn messages.
CMD_ABORTED	Returned if the command issued was aborted by the control pipe.
Command not supported	Returned if the command is not supported by the reader.

6: Command Descriptions

6.1: Generic Application Protocol Data Unit (APDU)

6.1.1: Working with DESFire and MIFARE Plus Tokens

For information about working with DESFire EV1, see Identiv's application note [AN337].

For information about working with MIFARE Plus, see Identiv's application note [AN338].

NOTE: Because these application notes contain information available only under a Non-Disclosure Agreement (NDA) with NXP, you must sign an NDA with NXP before you can receive them.

6.1.2: 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 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. (For example, for ISO14443A: single 4 bytes, double 7 bytes, triple 10 bytes; For ISO14443B: 4 bytes PUPI).

Response APDU:

Data	Status Word
Requested bytes of UID	SW1, SW2

6.1.3: PAPDU_ESCAPE_CMD

Usually Escape commands are transmitted through ScardControl, as defined in the 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. Therefore this vendor-specific APDU was defined to transmit Escape commands to the reader.

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

Examples:

- 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 04 (to set to NFC test mode)

Response APDU: 90 00

NOTES:

- 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 data to be sent using SCardTransmit.
- 2) Because the Escape commands defined using "[READER_GENERIC_ESCAPE](#)" have ISO 7816 APDU format, they can be sent using SCardTransmit without having to prepend "FF CC 00 00 P3".

6.2: Supported Pseudo APDU

This section describes all the Pseudo APDUs specific to the Contactless Interface supported by the reader.

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:

Mifare Standard										
Card type: Mifare Standard										
Memory size: 1024 Bytes										
Unique ID: 1A E3 B3 39										
Sector	Hex	ASCII								Block Read Write Inc Dec
0	1AE3 B339 7388 0400 47C1 25A8 4100 3106	.ã³9s^..GÁ#~A.1.								A B A B A B A
	0000 0000 0000 0000 0000 0000 0000 0000								A B A B A B A
	0000 0000 0000 0000 0000 0000 0000 0000								A B A B A B A
	FFFF FFFF FFFF FF07 8069 FFFF FFFF FFFF	ÿÿÿÿÿÿ.ëiÿÿÿÿÿÿ								
1	0000 0000 0000 0000 0000 0000 0000 0000								A B A B A B A
	0001 0203 0405 0607 0809 0A0B 0C0D 0E0F								A B A B A B A
	0000 0000 0000 0000 0000 0000 0000 0000								A B A B A B A
	FFFF FFFF FFFF FF07 8069 FFFF FFFF FFFF	ÿÿÿÿÿÿ.ëiÿÿÿÿÿÿ								

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 to match the block size of the used card, which is:

- 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 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:

Mifare Standard									
Card type: Mifare Standard									
Memory size: 1024 Bytes									
Unique ID: 1A E3 B3 39									
Sector	Hex	ASCII							Block Read
0	1AE3 B339 7388 0400 47C1 25A8 4100 3106	.ã*9s^..GÃ%A.1.							A B
	0000 0000 0000 0000 0000 0000 0000 0000							A B
	0000 0000 0000 0000 0000 0000 0000 0000							A B
	FFFF FFFF FFFF FF07 8069 FFFF FFFF FFFF	ÿÿÿÿÿÿÿ.€iÿÿÿÿÿÿÿ							A B
1	0000 0000 0000 0000 0000 0000 0000 0000							A B
	0001 0203 0405 0607 0809 0A0B 0C0D 0E0F							A B
	AA55 AA55 AA55 AA55 AA55 AA55 AA55 AA55	*U*U*U*U*U*U*U*U							A B
	FFFF FFFF FFFF FF07 8069 FFFF FFFF FFFF	ÿÿÿÿÿÿÿ.€iÿÿÿÿÿÿÿ							A B

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 kinds 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					Reserved for future use
				xxxx				If b6 is set, then 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 CLOUD 370x 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 the 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”. Because a MIFARE key is only 6 bytes in length, data needs to be padded per the 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. To validate the integrity of the processed key data, a 2-byte CRC must be sent following the key data. Refer to the “Load Keys – Reader – Secure” example 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 FF FF FF FF FF” is:

```
FF82006006 FFFFFFFFFFFFFFFF
```

Load Keys – Card – Secure

If the default AES128 reader 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 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, which excludes 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
For MIFARE Classic: 48 bytes of sector data read from card For MIFARE UL: Entire card data is returned (64 bytes)	SW1, SW2

Examples:

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 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 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 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 4 blocks of the sector, which includes 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
For MIFARE Classic: 48 bytes of sector data read from card For 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
 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 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 in the case of MIFARE Classic).

Command APDU:

Command	CLA	INS	P1	P2	P3	Data
Write Sector	FF	D7	Addr MSB	Addr LSB	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, and 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, where P2 codes the block number.

Command APDU:

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 are not shown). Note that block 4 must be set up as the value block before executing this command; see the 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 (Tag - Length - Value) structure that defines which action should be performed, which block the actions pertain to (the destinations), 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 the 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 APDU:

Data	Status Word
C0 03 Error status; see the next table	SW1, SW2 (The 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 the APDU; 01 = error in the 1 st data object; 02 = error in the 2 nd 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

6.2.10: PAPDU_TCL_PASS_THRU (T=CL Pass Thru)

This command can be used to send raw data using the 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 (The 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 ISO 14443-4 Type A standard framing to a card. CRC bytes will be appended automatically. The reader will not add transport protocol data – such as Protocol Control Byte (PCB), Node Address (NAD), Card Identifier (CID), etc. – to the raw data.

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 from 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 targeted mainly at MIFARE Plus S0 cards. A MIFARE Plus card at S0 level is detected as a MIFARE memory card. To personalize one of these cards first, it needs to be switched to Part 4 mode. For this purpose, this 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 a NFC Forum tag type 3 in the reader's field.

Command APDU:

Command	CLA	INS	P1	P2	P3	Data
FeliCa REQC	FF	40	00	00	04	2 bytes of system code, 1 byte reserved for future use (RFU), 1 byte for time slot number (TSN)

Response APDU:

Data	Status Word
16 bytes of NFCID2, and 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 of Manufacture ID (IDm) + No. of services 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 of Manufacture ID (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 services
- 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 services	Number of blocks	2*(P1 + P2)	Service Code List, Block List

Response APDU:

Data	Status Word
8 bytes of Manufacture ID (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 services
- 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 services	Number of blocks	$2 * (P1 + P2) + (16 * P2)$	Service Code List, Block List, Block Data

Response APDU:

Data	Status Word
8 bytes of Manufacture ID (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 of Manufacture ID (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

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](#) command 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	—

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 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 write 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

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 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 with erase (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

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 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 the 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	—

Where the P2 Segment Address is:

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 the dynamic memory model.

Command APDU:

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

Where the P2 Block Address is:

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 the 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

Where the P2 Block Address is:

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

Response APDU:

Data	Status Word
8 bytes of data that have been written	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 the 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

Where the P2 Block Address is:

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

Response APDU:

Data	Status Word
8 bytes of data	SW1, SW2

6.3: Escape Commands for the CLOUD 370x F

Amendment 1 of the PC/SC specification, Part 3, introduced a method to define vendor-specific commands. CLOUD 370x 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 6.3.4.1. [CNTLESS_GET_SET_NFC_PARAMS](#) as an example.

6.3.1: Sending Escape Commands to CLOUD 370x F

A developer can use the following methods to send Escape commands to CLOUD 370x F:

- **SCardControl** method defined in PC/SC API
- **SCardTransmit** method defined in PC/SC API in conjunction with the [Escape command APDU](#). Please note that SCardTransmit will only work when connected to a card.

When using the CCID driver provided with Windows, to send Escape commands for the CLOUD 370x F, this feature must be enabled by setting a REG_DWORD value named '**EscapeCommandEnable**' in the registry to a value of '1'. (When using the Identiv-supplied driver, this is not necessary.)

- For **Windows XP** and **Windows Vista**, the key to hold the value for CLOUD **3700 F** is:
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\USB\VID_04E6&PID_5790\Device-Instance-xxxx\Device Parameters](#)

while the value for CLOUD **3701 F** is:

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\USB\VID_04E6&PID_5791\Device-Instance-xxxx\Device Parameters](#)

- For **Windows 7** and **Windows 8**, the value for CLOUD **3700 F** is:
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\USB\VID_04E6&PID_5790\Device-Instance-xxxx\Device Parameters\WUDFUsbccidDriver](#)

while the value for CLOUD **3701 F** is:

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\USB\VID_04E6&PID_5791\Device-Instance-xxxx\Device Parameters\WUDFUsbccidDriver](#)

Device-Instance-xxxx must be equal to the serial number of the reader used, so this modification must be made for every physical reader intended to be used on the machine in question. The reader must be plugged in at least once for the mentioned key to exist, and the driver must be restarted (by unplugging and re-plugging the reader) for this setting to take effect.

See Appendix B for some sample code sending Escape commands to the reader.

6.3.2: Escape Command Codes

Escape commands can be used by an application to configure the CLOUD 370x F to function in a mode that is not its default configured mode, or to get specific information. To return the CLOUD 370x F back to its default mode, it either has to be unplugged and plugged again, or the application can send the same Escape command again.

The Escape commands supported by CLOUD 370x F readers are shown in the next table.

6.3.3: Reader-Specific Escape Commands

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_RDWR_USR_AREA	0xF0
READER_RD_WR_CUSTOMER_ID	0xF0
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 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 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 slot and contactless slot
NFC Test	0x04	NFC Test Mode – Applicable only for contactless interface

- ISO mode uses the APDU mode of data transfer and is used for normal operation. This is the default mode of the reader on Power up.
- NFC test mode is used to ignore the deactivate-activate sequence (PC_TO_RDR_ICCPOWERON - 0x62, and PC_TO_RDR_ICCPOWEROFF – 0x63) during SCardConnect.
- Any other value sent as the Mode is reserved for future use.

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
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	
0x90	0x57	Identiv CLOUD 3700 F Contactless Reader
0x91	0x57	Identiv CLOUD 3701 F Contactless Reader

6.3.3.4: 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 the LED number (when more than one LED is present; otherwise set to 0), and then the desired LED state.

The CLOUD 3700 F reader provides a two-color (Red or Green) LED:

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

The CLOUD 3701 F reader provides only a one-color (Green) LED:

Byte0	Byte 1	Byte2
Escape code (0x19)	LED number (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 firmware version, reader capabilities, and Unicode serial number of the reader.

Input:

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

Byte0

Escape code (0x1E)

Output:

The firmware will return data in this SCARD_READER_GETINFO_PARAMS_EX data structure:

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	0x04 for Contactless only readers. 0x07 for Contact + Contactless readers. 0x03 for Contact only readers. NOTE: ISO mode is not indicated because it is always supported.
2	wSupportedProtocols	Protocols supported by the Reader: Bit 0 = T0 Bit 1 = T1	0x0003 Received as LSB first.
2	wInputDevice	0x00 = IO_DEV_NONE 0x01 = IO_DEV_KEYPAD 0x02 = IO_DEV_BIOMETRIC	0x0000 Received as LSB first.
1	byPersonality	Reader Personality (Not Used)	0x00
1	byMaxSlots	Maximum number of slots	0x01
1	bySerialNoLength	Serial number length (0x1C)	0x1C
28	bySerialNumber	Unicode serial number	Reader serial number Received as 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 whether LED control by firmware should be disabled or enabled. The output buffer is NULL.

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, a 1-byte response is received.

Output buffer
NULL, or current state

6.3.3.7: READER_RD_WR_USER_AREA

This Escape command is used to access the user data area in the reader. The user area is located in the non-volatile memory of the reader, so data will be retained even after a power cycle.

NOTES:

- Frequent writes should be avoided, because the non-volatile memory supports only 100K writing cycles.
- A maximum of 249 bytes can be read and written. The sector can be read and written only as a whole.
- If complete data (249 bytes) is not given during a write operation, then random data will be padded to the given data and then written. If you want to modify only part of the data, read the entire 249 bytes, modify the data you want to change, and then write it back to the reader.

Input:

The first byte of the input buffer contains the escape code. The second byte specifies whether the user area is to be read or written.

Byte0	Byte1 Value	Description	Byte2 to Byte251
Escape code (0xF0)	1	Read 249 bytes of user data	None
	2	Write 249 bytes of user data	Data to be written

Output:

Operation	Data (Byte0 - BYTE248)
Read	249 bytes of user data
Write	No bytes returned

6.3.3.8: READER_RD_WR_CUSTOMER_ID

This Escape command is used to read or write the customer ID from/to the user area in the reader. The user area is in the non-volatile memory of the reader, so data will be retained even after a power cycle.

NOTES:

- Frequent writes should be avoided, because the non-volatile memory supports only 100K writing cycles.
- A maximum of 8 bytes can be read and written. The sector can be read and written only as a whole.
- If complete data (8 bytes) is not given during a write operation, then the operation will fail and return an error. If you want to modify only partial data in the reader, read the entire 8 bytes, modify the data you want, and then write it back to the reader.

Input:

The first byte of the input buffer contains the escape code. The second byte specifies whether the customer ID is to be read or written.

Byte0	Byte1 Value	Description	Byte2 to Byte9
Escape code (0xF0)	3	Write 8 bytes of customer ID	Data to be written
	4	Read 8 bytes of customer ID	None

Output:

Operation	Data (Byte0-BYTE8)
Read	8 bytes of customer ID
Write	No bytes returned

6.3.3.9: READER_GENERIC_ESCAPE

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

Input:

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

Byte0	Byte1	Byte2	Byte3	Byte4	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 return at least 2 bytes status word SW1, SW2.

- In case of success, SW1=0x90 and SW2=0x00 shall be returned.
- In case of an error, the appropriate error status shall be returned (as defined in Error Code section 8.0).

6.3.4: Specific Escape Commands for Contactless Interface

Escape Code	Escape Command
0x11	CNTLESS_GET_CARD_INFO
0x93	CNTLESS_GET_ATS_ATQB
0x94	CNTLESS_GET_TYPE
0x95	CNTLESS_SET_TYPE
0x96	CNTLESS_RF_SWITCH
0x99	CNTLESS_CONTROL_PPS
0x9D	CNTLESS_CONTROL_848
0x9E	CNTLESS_GET_BAUDRATE
0xA7	CNTLESS_CONTROL_RETRIES
0xAC	CNTLESS_CONTROL_POLLING
0xAD	CNTLESS_FORCE_BAUDRATE
0xDA	CNTLESS_GET_CARD_DETAILS
0xE1	CNTLESS_SET_CONFIG_PARAMS
0xE4	CNTLESS_IS_COLLISION_DETECTED
0xF3	CNTLESS_FELICA_PASS_THRU
0xE9	CNTLESS_P2P_SWITCH_MODES
0xEA	CNTLESS_P2P_TARGET_RECEIVE
0xEB	CNTLESS_P2P_TARGET_SEND
0xE6	CNTLESS_P2P_INITIATOR_DESELECT
0xE7	CNTLESS_P2P_INITIATOR_TRANSCEIVE
0xEC	CNTLESS_NFC_SINGLESOT
0xED	CNTLESS_NFC_LOOPBACK
0xFF	CNTLESS_GET_SET_NFC_PARAMS CNTLESS_GET_P2P_EXTERNAL_RF_STATE

6.3.4.1: CNTLESS_GET_CARD_INFO

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

Input:

The first byte of the 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 details in the next table)	Card Type Info: Upper nibble indicates memory card, T=CL, or dual mode card; Lower nibble indicates Type A or Type B card. (See the Card Type Info table below for values.)

Card to Reader communication baud rate (Byte1) settings:

bit	Description
b0	1 = 212 Kbps supported from reader to card
b1	1 = 424 Kbps supported from reader to card
b2	1 = 848 Kbps supported from reader (CLOUD 3700 F only) to card
b3	Always 0
b4	1 = 212 Kbps supported from card to reader
b5	1 = 424 Kbps supported from card to reader
b6	1 = 848 Kbps supported from card to reader (CLOUD 3700 F only)
b7	0 = different baud rates used for the two directions (reader to card versus card to reader) 1 = same baud rate used for both directions (reader to card and card to reader)

Examples:

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	Meaning
0	Memory card
1	T=CL card
2	Dual mode card

Lower Nibble Value	Meaning
0	Type A card
1	Type B card

6.3.4.2: CNTLESS_GET_ATS_ATQB

This Escape command retrieves the ATS (Answer To Select) for Type A T=CL cards, or the ATQB (Answer to Request, Type B) for Type B cards.

Input:

The first byte of input buffer contains the escape code.

Byte0
Escape code (0x93)

Output:

The output buffer contains either the ATS bytes or the ATQB bytes, depending on the type of PICC placed within the field of the reader.

6.3.4.3: CNTLESS_GET_TYPE

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

Input:

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
Escape code (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

Output:

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)								Card Type
Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0	
—	—	—	—	—	—	—	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 (reserved for future use).

6.3.4.4: CNTLESS_SET_TYPE

This Escape command configures the types 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. The default is Type A + Type B (0x02).

Input:

The input buffer shall contain either 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 (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 (reserved for future use).

Output:

The output buffer is:

Output buffer
NULL

6.3.4.5: CNTLESS_RF_SWITCH

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

Input:

The first byte of the 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, a 1-byte response is received.

Output buffer
NULL or current state

6.3.4.6: CNTLESS_CONTROL_PPS

This Escape command disables the automatic Protocol Parameter Selection (PPS) done by the firmware/device for contactless cards.

Input:

The first byte of the 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, a 1-byte response is received.

Output buffer
NULL or current state

6.3.4.7: CNTLESS_CONTROL_848

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

The RF communication with a user token will only switch to 848 Kbps if the user token supports this baud rate and automatic Protocol Parameter Selection (PPS) is ON.

NOTE:

Only the CLOUD 3700 F reader supports 848 Kbps; the CLOUD 3701 F reader's maximum baud rate is 424 Kbps.

Input:

The input buffer shall contain 2 bytes.

Byte0	Byte1	Description
Escape code (0x9D)	0x00	Disable 848 Kbps support
	0x01	Enable 848 Kbps support
	0xFF	Get current status of 848 Kbps support

Output:

- 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, then the output buffer is a BYTE buffer with the following possible values:

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

6.3.4.8: CNTLESS_GET_BAUDRATE

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

Input:

The first byte of the 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	848 Kbps 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

NOTE:

Only the CLOUD 3700 F reader supports 848 Kbps; the CLOUD 3701 F reader's maximum baud rate is 424 Kbps.

6.3.4.9: CNTLESS_CONTROL_RETRIES

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

Input:

The first byte of the 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, a 1-byte response is received.

Output buffer
NULL or current state

6.3.4.10: CNTLESS_CONTROL_POLLING

This Escape command is used to enable or disable firmware polling for contactless cards.

Input:

The first byte of the 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, a 1-byte response is received.

Output buffer
NULL or current state

6.3.4.11: CNTLESS_FORCE_BAUDRATE

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

Input:

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	When bit 0 = 1, DR=2 is supported	Encoding of the baud rate to be allowed when the B1 value is 0x01. (There is no need to send this byte when B1 has the value =x00.)
	When bit 1 = 1, DR=4 is supported	
	When bit 2 = 1, DR=8 is supported	
	Bit 3 shall be set to 0; the value of 1 is reserved for future use	
	When bit 4 = 1, DR=2 is supported	
	When bit 5 = 1, DR=4 is supported	
	When bit 6 = 1, DR=8 is supported	
	When bit 7 = 0, different baud rates can be used for the two directions. When bit 7 = 1, the same baud rate must be used for both directions.	
	NULL	When B1=0x00

Output:

The output buffer is:

Output buffer
NULL

6.3.4.12: 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 the 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 is 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 the 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

6.3.4.13: CNTLESS_SET_CONFIG_PARAMS

This Escape command is used to configure RXGAIN and RXTHRESHOLD of the RF receiver for different baud rates and card types. All configured parameters are volatile.

Input:

The first byte of the input buffer contains the escape code. The following 16 bytes contain these parameters:

Byte #	Value	Description
B0	0xE1	Escape code
B1	Type A RXGAIN for polling or 106 Kbps	
B2	Type A RXGAIN for 212 Kbps	
B3	Type A RXGAIN for 424 Kbps	
B4	Type A RXGAIN for 848 Kbps	
B5	Type A RX THRESHOLD for polling or 106 Kbps	
B6	Type A RX THRESHOLD for 212 Kbps	
B7	Type A RX THRESHOLD for 424 Kbps	
B8	Type A RX THRESHOLD for 848 Kbps	
B9	Type B RXGAIN for polling or 106 Kbps	
B10	Type B RXGAIN for 212 Kbps	
B11	Type B RXGAIN for 424 Kbps	
B12	Type B RXGAIN for 848 Kbps	
B13	Type B RX THRESHOLD for polling or 106 Kbps	
B14	Type B RX THRESHOLD for 212 Kbps	
B15	Type B RX THRESHOLD for 424 Kbps	
B16	Type B RX THRESHOLD for 848 Kbps	

Output:

Output buffer

NULL

6.3.4.14: CNTLESS_IS_COLLISION_DETECTED

This Escape command is used to identify whether multiple Type A cards are detected in the reader's field.

Input:

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

Byte0
Escape code (0xE4)

Output:

Byte0	
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 the input buffer contains the escape code, and the following bytes contain 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_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 the input buffer contains the escape code. The second byte either sets the mode, or contains a code to retrieve the setting.

Additional data bytes are used for Initiator or Target mode.

Offset	Description	Detailed description
0	Escape code (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 provide additional information, as described below.

Offset	Initiator Mode Bytes	Detailed description
2		Reserved for future use
3		Reserved for future use
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	Reserved for future use
3	0x00	Reserved for future use
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 3 bytes of NFCID1)
29	0x88	Timeout Low Byte
30	0x13	Timeout High Byte
31	N	Number of General bytes in ATR_RES
32 to N+32	General bytes to be sent in ATR_RES	

Output:

- 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:
 - 0x00 = P2P Initiator mode
 - 0x01 = P2P Target mode
 - 0x02 = Reader / Writer mode

6.3.4.17: CNTLESS_P2P_TARGET_RECEIVE

This Escape command is used to receive data from the 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	Description	Detailed description
0	Escape code (0xEA)	Target Receive
1		Reserved for future use
2		Reserved for future use
3		Reserved for future use
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	Description	Detailed description
0		Reserved for future use
1		Reserved for future use
2		Reserved for future use
3	0 = No Chaining 1 = Chaining	Chaining
Offset 4 to offset 4+N	N data bytes	Bytes Received

6.3.4.18: 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	Description	Detailed description
0	Escape code (0xEB)	Target Send
1	0x00	Reserved for future use
2	0x00	Reserved for future use
3	0x00	Reserved for future use
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:

After the data bytes are sent successfully, the firmware will use the Chaining byte to indicate if it is ready to send more bytes.

Offset	Description	Detailed description
0		Reserved for future use
1		Reserved for future use
2		Reserved for future use
3	0 = No Chaining 1 = Chaining	Chaining

6.3.4.19: CNTLESS_P2P_INITIATOR_DESELECT

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

Input:

Byte0
Escape code (0xE6)

Output:

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

6.3.4.20: CNTLESS_P2P_INITIATOR_TRANSCIVE

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	Description	Detailed description
0	Escape code (0xE7)	Initiator transceive
1	0x00	Reserved for future use
2	0x00	Reserved for future use
3	0x00	Reserved for future use
4	0 = No Chaining 1 = Chaining	Chaining
5	—	Timeout Low Byte
6	—	Timeout High Byte
Offset 7 to offset 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 is available from offset 0x04. The value of the Chaining byte indicates whether additional data is present.

Offset	Description	Detailed description
0		Reserved for future use
1		Reserved for future use
2		Reserved for future use
3	0 = No Chaining 1 = Chaining	Chaining
Offset 4 to offset 4+N	N data bytes	Bytes received

6.3.4.21: CNTLESS_NFC_SINGLESOT

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

Input:

Offset	Description	Detailed description
0	Escape code (0xEC)	NFC Single-shot mode. To perform test cases as defined in the NFC Test-Cases-For-Digital-Protocol Tag 4 (Type A and Type B) test cases.
1	0x01	NFC_DEP supported. To perform test cases in Peer2Peer mode, as defined in the NFC Test-Cases-For-Digital-Protocol. If a value other than 0x01 is given, NFC_DEP is not supported in the preceding I-Blocks.

Output:

Output buffer
NULL

6.3.4.22: CNTLESS_NFC_LOOPBACK

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

Input:

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

Output:

Output buffer
NULL

6.3.4.23: 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 the application.

Input:

To **set** the parameters, the command syntax is:

Byte0 CLA	Byte1 INS	Byte2 P1	Byte3 P2	Byte4 Lc	Byte5	Byte6	Byte7	Byte 8	Le
0xFF	0x70	0x04	0xE6	0x04	0x04 (opcode)	0x01 = SET	NFC Parameter	Value	00

To **get** the parameters, the command syntax is:

Byte0 CLA	Byte1 INS	Byte2 P1	Byte3 P2	Byte4 Lc	Byte5	Byte6	Byte7	Le
0xFF	0x70	0x04	0xE6	0x03	0x04 (opcode)	0x00 = GET	NFC Parameter	00

The value of byte 7 (NFC Parameter) 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 = PSL_REQ_FSL	FSL used in PSL_REQ

6.3.4.24: CNTLESS_GET_P2P_EXTERNAL_RF_STATE

This Escape command is supported through the [READER GENERIC ESCAPE](#) message. This command is used to check whether the external RF was reset after the reader was detected in target mode.

Input:

Byte0 CLA	Byte1 INS	Byte2 P1	Byte3 P2	Byte4 Lc	Byte5	Le
0xFF	0x70	0x04	0xE6	0x01	0x06 (opcode)	00

Output:

If the command is successful, a single byte is returned which indicates the value of the parameter:

- Bit 0 = 1 when a present external RF field is switched off.
- Bit 1 = 1 when an external RF field is detected.
- Bit 2 through Bit 7 always read as 0, and are reserved for future use.

7: Appendixes

7.1: Appendix 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: Appendix B: Sample Code Using Escape Commands

File Name: CLOUD 370x F Escape.h

```
#ifndef _CLOUD_370xF_ESCAPE_H_
#define _CLOUD_370xF_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 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_SINGLESOT                         0xEC
#define CNTLESS_NFC_LOOPBACK                         0xED

#ifdef __cplusplus
}
#endif
#endif
```

File Name: CLOUD 370x F Escape.c

```
#include <windows.h>
#include <winbase.h>
#include <stdio.h>
#include <conio.h>
#include "winscard.h"
#include "winerror.h"
#include "CLOUD 370xF Escape.h"

VOID main(VOID)
{
    SCARDCONTEXT          ContextHandle;
    SCARDHANDLE            CardHandle;
    ReaderInfoExtended     strReaderInfo;
    BYTE                   InByte, i;
    DWORD                  BytesRead, ActiveProtocol;
    ULONG                  ret;
    char                   *s;
    char                   *ReaderName[] = {"Identive CLOUD 3700 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\n", (strReaderInfo.byMajorVersion& 0xF0)>> 4,
                (strReaderInfo.byMajorVersion& 0x0F));
                printf("minor version:\t\t%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.wininputDevice);
                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.abbySerialNumber[i] != 0) printf("%c",
strReaderInfo.abbySerialNumber[i]);
        } else {
            printf("SCardControl failed: %08X\n", ret);
        }
    }
    else {
        printf("SCardConnect failed: %08X\n", ret);
    }

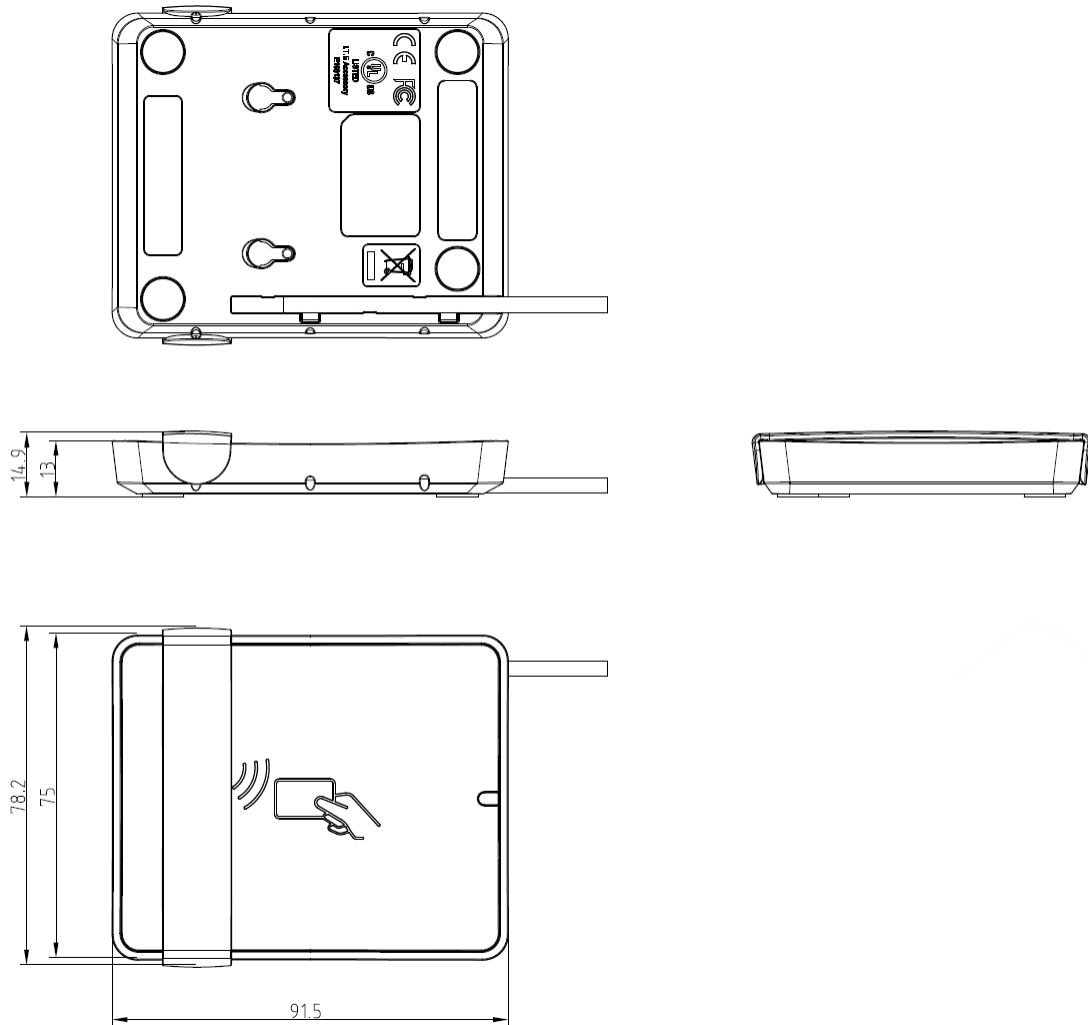
    ret = SCardReleaseContext(ContextHandle);
}
else
{
    printf("\n SCardEstablishContext failed with %.8lX",ret);
}

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


7.3: Appendix C: Mechanical Drawings

7.3.1: Reader (without stand)

NOTE: All dimensions on these mechanical drawings are in millimeters.



7.3.2: Reader on Stand

