



Reference Manual for SCR3310 V2 Contact Desktop Reader

For Part #: 905331



Document Version: 1.1, Last Revised On: 2018-07-19

Abstract

This document contains in-depth information about the hardware and software features of the SCR3310 V2 contact smart card reader.

Audience

This document is intended for system integrators and software developers.

Revision History

Rev.	Date	Description
1.0	2014-06-02	First published external version
1.1	2017-07-06	Minor modifications and company name change

Contact Information

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

Table of Contents

1: LEGAL INFORMATION	5
1.1: Disclaimers.....	5
1.2: Licenses	5
1.3: Trademarks	5
2: INTRODUCTION TO THE MANUAL.....	6
2.1: Objective of the manual	6
2.2: Target audience	6
2.3: Product version corresponding to the manual	6
2.4: Definition of various terms and acronyms	7
2.5: References	7
2.6: Conventions	8
3: GENERAL INFORMATION ABOUT SCR3310 V2.....	9
3.1: SCR3310 V2 key benefits	9
3.2: SCR3310 V2 key features	9
3.3: SCR3310 V2 ordering information.....	9
3.4: SCR3310 V2 customization options	9
Applications.....	10
3.4.1: General.....	10
3.4.2: Applications provided by Identiv	10
4: SCR3310 V2 CHARACTERISTICS	11
4.1: SCR3310 V2 high level architecture.....	11
4.1.1: Block diagram.....	11
4.1.2: Software architecture	12
4.2: Quick reference data.....	13
4.2.1: SCR3310 V2 dimensions	13
4.2.2: LED behavior	14
4.2.3: Other data	14
4.2.3.1: General.....	14
4.2.3.2: USB	14
4.2.3.3: Card interface.....	15
5: SOFTWARE MODULES.....	16
5.1: Installation	16
5.2: Utilities	16
5.3: Driver	16
5.3.1: SCR3310 V2 listing	16
5.3.2: Supported operating systems.....	17
5.4: CT-API.....	17
5.5: MCard-API.....	17
5.6: Firmware	17

5.6.1:	CCID transport protocol.....	17
5.6.1.1:	CCID class requests supported	17
5.6.1.2:	CCID messages supported	18
5.6.1.3:	CCID Error Codes	18
6:	COMMANDS DESCRIPTION	20
6.1:	Escape commands for the SCR3310 V2.....	20
6.1.1:	Sending Escape commands to SCR3310 V2	20
6.1.2:	Escape command codes	21
6.1.2.1:	CONTACT_SETMODE	22
6.1.2.2:	CONTACT_GETMODE	22
6.1.2.3:	CONTACT_GET_SET_POWER_UP_SEQUENCE	23
6.1.2.4:	CONTACT_EMV_LOOPBACK.....	24
6.1.2.5:	CONTACT_EMV_SINGLEMODE	25
6.1.2.6:	CONTACT_APDU_TRANSFER	25
6.1.2.7:	CONTACT_CONTROL_PPS	26
6.1.2.8:	CONTACT_EXCHANGE_RAW	26
6.1.2.9:	READER_GET_IFDTYPE	27
6.1.2.10:	READER_LED_CONTROL	27
6.1.2.11:	READER_LED_CONTROL_BY_FW.....	28
6.1.2.12:	READER_GET_INFO_EXTENDED	29
6.1.2.13:	CONTACT_GET_SET_CLK_FREQUENCY.....	30
6.1.2.14:	CONTACT_GET_SET_ETU	31
6.1.2.15:	CONTACT_GET_SET_WAITTIME	32
6.1.2.16:	CONTACT_GET_SET_GUARDTIME	33
6.1.2.17:	CONTACT_GET_SET_MCARD_TIMEOUT.....	34
6.1.2.18:	CONTACT_CONTROL_ATR_VALIDATION	35
7:	ANNEXES	36
7.1:	A – Status words table	36
7.2:	Annex B – Sample code using Escape commands through Escape IOCTL	37
7.3:	Annex C – Mechanical drawings	39
7.3.1:	Top Casing	39
7.3.2:	Bottom Casing	40

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: Licenses

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

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

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

1.3: Trademarks

Windows is a trademark of Microsoft Corporation.

2: Introduction to the manual

2.1: Objective of the manual

This manual provides an overview of the hardware and software features of the SCR3310 V2 smart card reader.

This manual describes in details interfaces and supported commands available for developers using SCR3310 V2 in their applications.

2.2: Target audience

This document describes the technical implementation of SCR3310 V2.

The manual targets software developers. It assumes knowledge about ISO/IEC 7816 and commonly used engineering terms.

2.3: Product version corresponding to the manual

This document has been updated for the below Hardware and Firmware versions.

Item	Version
Hardware	3.1
Firmware	6.01

2.4: Definition of various terms and acronyms

Term	Expansion
APDU	Application Protocol Data Unit
ATR	Answer to Reset, defined in ISO/IEC 7816
Byte	Group of 8 bits
CCID	Chip Card Interface Device
CID	Card Identifier
LED	Light emitting diode
NA	Not applicable
NAD	Node Address
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
PC/SC	Personal Computer/Smart Card: software interface to communicate between a PC and a smart card
PID	Product ID
RFU	Reserved for future use
USB	Universal Serial Bus
VID	Vendor ID
(xyz) _b	Binary notation of a number x, y, z ∈ {0,1}
0xYY	The byte value YY is represented in hexadecimal

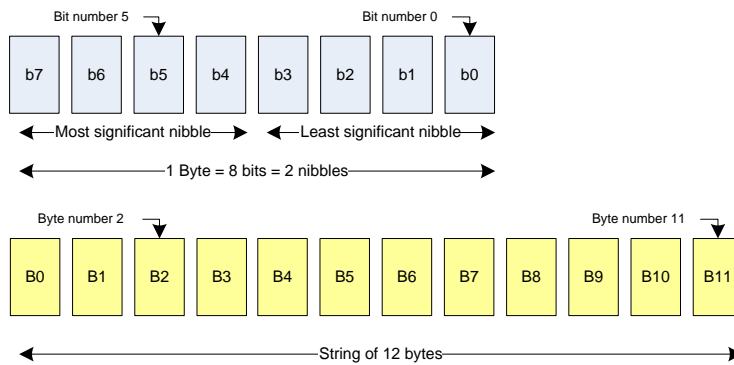
2.5: References

Doc ref in the manual	Description	Issuer
ISO/IEC 7816-3	Identification cards — Integrated circuit cards — Part 3: Cards with contacts — Electrical interface and transmission protocols	ISO / IEC
ISO/IEC 7816-4	Identification cards - Integrated circuit(s) cards with contacts Part 4: Interindustry commands for interchange ISO/IEC 7816-4: 2005 (E)	ISO / IEC
PC/SC	Interoperability Specification for ICCs and Personal Computer Systems v2.01	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

2.6: Conventions

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

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



Example:

163 decimal number is represented

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

The least significant nibble of 0xA3 is

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

The most significant nibble of =xA3 is

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

3: General information about SCR3310 V2


3.1: SCR3310 V2 key benefits

With its combination of a modern slim design and its state of the art feature set, SCR3310 V2 is the perfect desktop reader choice for environments where smart card support is required. Such environments may be corporations or government authorities where use applications like network log-in, Windows authentication & Single Sign-On are implemented. As for all Identiv products, SCR3310 V2 is designed to offer best in class interoperability.

3.2: SCR3310 V2 key features

- ISO/IEC 7816 compliant smart card reader
- PC/SC v2.0 compliant
- Unique serial number which enables that SCR3310 V2 can be plugged into any USB slot on a PC without having to re-install the driver.

3.3: SCR3310 V2 ordering information

Item	Part number	
SCR3310 V2	905331 (and derived products)	

3.4: SCR3310 V2 customization options

Upon request and based on a minimum order quantity, Identiv can customize:

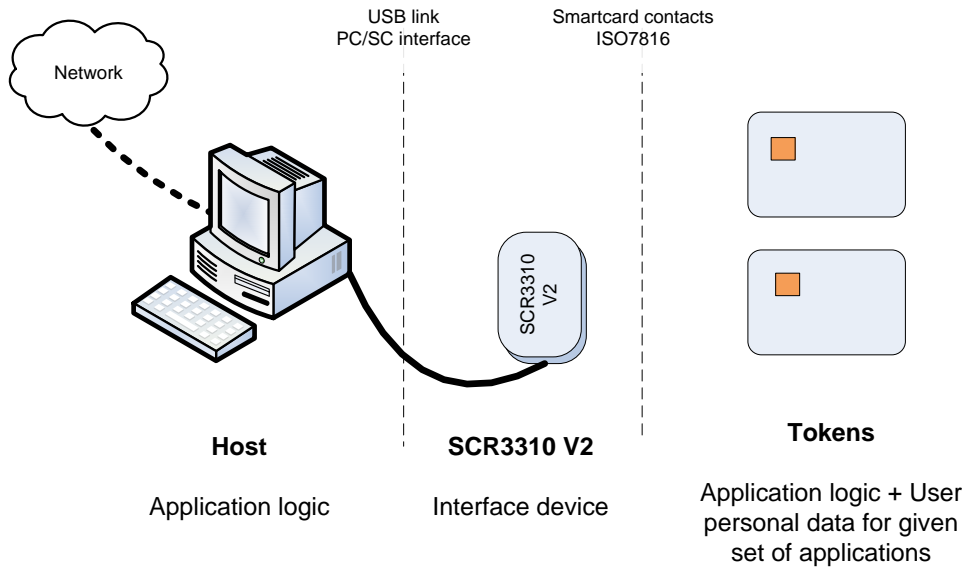
- The color of the casing
- The logo
- The product label
- The USB strings

*Terms and conditions apply, please contact your local Identiv representative.

3.5: Applications

3.5.1: General

SCR3310 V2 is a transparent reader designed to interface with a personal computer host supporting PC/SC interface with smart cards according to ISO/IEC 7816 as well as synchronous memory cards like CAC and PKI cards, banking cards and health insurance cards.



SCR3310 V2 itself handles the communication protocol but not the application related to the token. The application-specific logic has to be implemented by software developers on the host.

3.5.2: Applications provided by Identiv

Identiv provides a few applications for development and evaluation purposes that can function with SCR3310 V2. There are some tools provided; here is one of them:

- Smart card commander version 1.3 provides capabilities to identify most commonly used cards in the field and display the content of them as well as scripting functionality which can be very useful for developers to develop and debug their applications.

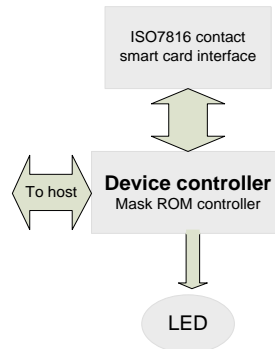
Identiv does not provide PKI or CAC applications.

4: SCR3310 V2 characteristics

4.1: SCR3310 V2 high level architecture

4.1.1: Block diagram

The link between SCR3310 V2 and the host to which it is connected is the USB interface providing both the power and the communication channel.



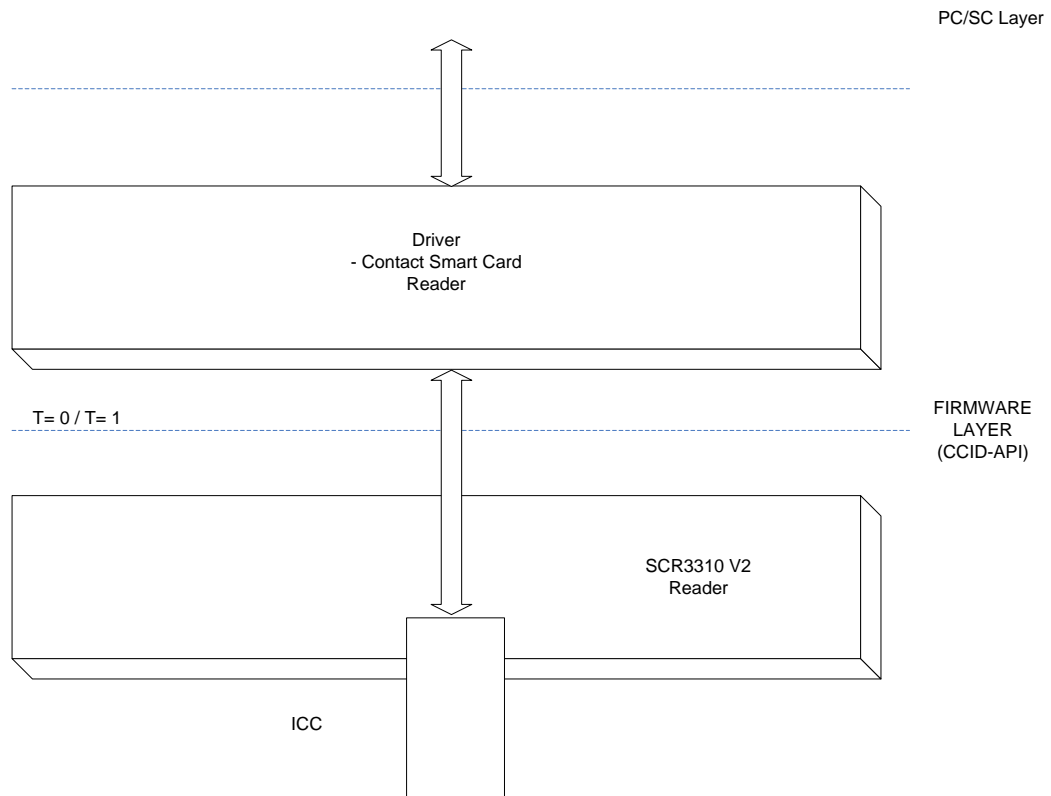
The SCR3310 V2 device controller has several interfaces available. In the SCR3310 V2 implementation 2 peripherals are connected to the device controller:

- LED for reader status indication
- A smart card interface

The micro-Controller contains the firmware developed by Identiv to handle all the ISO/IEC 7816 contact protocol and the PC/SC communication protocol with the host.

4.1.2: Software architecture

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



The SCR3310 V2 leverages a PC/SC CCID driver that is freely available for all supported operating systems (Windows, macOS X and Linux). With current Windows versions (starting with Windows Vista) and macOS X, this driver is already included in the basic installation.

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

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

Additionally, Identiv provides a proprietary driver for all the supported OSs.

4.2: Quick reference data

4.2.1: SCR3310 V2 dimensions

Item	Characteristic	Value
SCR3310 V2	Weight	60g
	External dimensions	66x69x15 mm
	Cable length	1.5 meter long with USB type A connector
	Default color	black and grey
	Default label	

Drawing with dimensions of the SCR3310 V2 can be found in annex.

4.2.2: LED behavior

SCR3310 V2 is equipped with a green LED. Its behavior is described in the table below.

SCR3310 V2 states	LED Indication
Reader powered, card out	OFF
Reader powered, card in but not powered	OFF
Card powered	ON
Card access	Blinking (500ms ON/500ms OFF)
error condition	Blinking (100ms ON/100ms OFF)

4.2.3: Other data

4.2.3.1: General

Parameter	Value/Description
Clock of the device controller	48 MHz
API	PC/SC 2.0
Operating temperature range	0° to 50°C
Operating humidity range	Up to 95%RH non-condensing
Certifications	USB CE UL-60950 FCC ICES-003 WEEE RoHS2 REACH WHQL Listed in GSA APL

4.2.3.2: USB

Parameter	Value/Description
DC characteristics	Low bus powered (SCR3310 V2 draws power from USB bus) Voltage: 5V Max. Current: 6mA + current consumed by an inserted card Suspend current: 500uA
USB specification	USB 2.0 FS device
USB Speed	Full Speed Device (12Mbit/s)
Device Class	CCID
PID	0x5116
VID	0x04E6

4.2.3.3: Card interface

Parameter	Value/Description
Smart card operating frequency	up to 16MHz
Maximum supported card baud-rate	600Kbps
Cards supported	Class A, B and C asynchronous smart cards (T=0, T=1) Synchronous smart cards (2wire, 3wire, I ² C)
ISO/IEC 7816 compliant	Yes
EMV 4.2 compliant	Yes
CT-API compliant	Yes
Number of slots	Single smart card slot
Ejection mechanism	Manual

5: Software modules

5.1: Installation

On Operating Systems with a PC/SC CCID driver preinstalled, no installation is necessary.

Where there's no PC/SC CCID driver preinstalled (Linux systems) the driver has to be installed using distribution specific measures or installed using the available source packages.

5.2: Utilities

The following utilities are available:

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

Operating systems supported by the tools:

- Windows Server 2012 (32 & 64 bit)
- Windows 7, 8.1 and 10 (32 & 64 bit)

5.3: Driver

5.3.1: SCR3310 V2 listing

SCR3310 V2 is listed by PC/SC applications as

- *Identiv SCR3310 V2*

SCR3310 V2 uses the PC/SC CCID class driver readily available for all the supported operating systems.

On Windows XP systems, this driver is not preinstalled and will not be loaded from Windows Update. The Hardware Wizard will need to be allowed to load the driver from the internet if it is not already present on the system (first time installation).

Starting with Windows Vista, the OS does have the driver preinstalled, so no additional driver installation is necessary.

macOS X systems do have the PC/SC CCID driver preinstalled.

On Linux systems, the distribution specific installation mechanism should be used.

5.3.2: Supported operating systems

- Windows Server 2012 (32 & 64 bit)
- Windows 7, 8.1 and 10 (32 & 64 bit)
- MacOS X (10.12)
- Linux 3.x (32 & 64 bit)

5.4: CT-API

A CT-API interface that mostly is used in German banking applications and in conjunction with health insurance cards, is available for the reader.

5.5: MCard-API

With this proprietary Identiv API, it is possible to access a vast majority of synchronous memory cards.

Cards supported are:

- SLE4404
- SLE4428
- SLE4432
- SLE4436
- SLE6636
- SLE4442
- SLE5532
- SLE5536
- SLE5542
- AT24C01ASC
- AT24C02SC
- AT24C04SC
- AT24C08SC
- AT24C16SC
- AT24C32SC
- AT24C64SC
- AT24C128SC
- AT24C256SC
- AT24C512SC
- AT88SC153
- AT88SC1608
- ST14C02

5.6: Firmware

5.6.1: CCID transport protocol

SCR3310 V2 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 paragraph describes the CCID specification features that are implemented.

5.6.1.1: CCID class requests supported

- Abort

5.6.1.2: CCID messages supported

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

- PC_to_RDR_IccPowerOn
- PC_to_RDR_IccPowerOff
- PC_to_RDR_GetSlotStatus
- PC_to_RDR_XfrBlock
- PC_to_RDR_GetParameters
- PC_to_RDR_SetParameters
- PC_to_RDR_Escape
- PC_to_RDR_Abort
- PC_to_RDR_NotifySlotChange
- PC_to_RDR_ResetParameters
- PC_to_RDR_T0APDU
- PC_to_RDR_SetDatarateAndClockFrequency

5.6.1.3: CCID Error Codes

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

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

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

1.1.1.1.1. HW_ERROR

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

1.1.1.1.2. XFR_PARITY_ERROR

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

1.1.1.1.3. ICC_PROTOCOL_NOT_SUPPORTED

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

1.1.1.1.4. BAD_ATR_TS

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

1.1.1.1.5. BAD_ATR_TCK

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

1.1.1.1.6. ICC_MUTE

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

1.1.1.1.7. CMD_ABORTED

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

1.1.1.1.8. Command not supported

This error would be returned, if the command would not be supported by the reader.

6: Commands description

6.1: Escape commands for the SCR3310 V2

6.1.1: Sending Escape commands to SCR3310 V2

A developer can use the following method to send Escape commands to SCR3310 V2

- SCardControl method defined in PC/SC API

In Windows, in order to be able to send Escape commands to the SCR3310 V2, the feature has got to be enabled by setting a REG_DWORD value named 'EscapeCommandEnable' in the registry to a value of '1'.

For Windows XP and Windows Vista, the key to hold the value would be

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\USB\VID_04E6&PID_5816\Device-Instance-xxxx \Device Parameters](#)

For Windows 7 and Windows 8, that would be

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\USB\VID_04E6&PID_5816\Device-Instance-xxxx \Device Parameters\WUDFUsbccidDriver](#)

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

To be able to work with synchronous memory cards using our MCard API, the same setting will have to be established.

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

6.1.2: Escape command codes

Escape commands can be used by an application to configure SCR3310 V2 to function in a mode that is not its default configured mode or to get specific information. To put the SCR3310 V2 back into its default mode, it either has to be unplugged and plugged again or the application can send the same Escape command again.

The following Escape commands are supported by SCR3310 V2

Escape command	Code
CONTACT_SETMODE	0x01
CONTACT_GETMODE	0x02
CONTACT_GET_SET_POWERUPSEQUENCE	0x04
CONTACT_EMV_LOOPBACK	0x05
CONTACT_EMV_SINGLEMODE	0x06
CONTACT_APDU_TRANSFER	0x08
CONTACT_CONTROL_PPS	0x0F
CONTACT_EXCHANGE_RAW	0x10
READER_GET_IFDTYPE	0x12
READER_LED_CONTROL	0x19
READER_LED_CONTROL_BY_FW	0xB2
READER_GET_INFO_EXTENDED	0X1E
READER_CONTROL_XFER_MODE	0x70
CONTACT_GET_SET_CLK_FREQUENCY	0x1F
CONTACT_GET_SET_ETU	0x80
CONTACT_GET_SET_WAITTIME	0x81
CONTACT_GET_SET_GUARDTIME	0x82
CONTACT_GET_SET_MCARD_TIMEOUT	0x85
CONTACT_CONTROL_ATR_VALIDATION	0x88

6.1.2.1: CONTACT_SETMODE

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

Input:

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

Byte0	Byte1
Escape code (0x01)	Mode

Following table gives the value of modes as interpreted by the firmware:

Mode	Value	Remarks
ISO	0x00	ISO/IEC 7816 mode
EMV	0x01	EMV
Synchronous	0x02	memory card mode (Synchronous)

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

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

Synchronous mode is used for communicating only with memory cards.

Any other value sent as mode is invalid.

The output buffer is

Output buffer
NULL

6.1.2.2: CONTACT_GETMODE

This Escape command may be used to retrieve the current mode of the reader.

The input buffer is

Byte0
Escape code(0x02)

Output:

Current active reader mode will be returned as a BYTE value.

Following table gives the value of modes as interpreted by reader firmware

Mode	Value	Remarks
ISO	0x00	ISO/IEC 7816 mode
EMV	0x01	EMV
Synchronous	0x02	memory card mode (synchronous)

6.1.2.3: CONTACT_GET_SET_POWER_UP_SEQUENCE

This Escape command is used by the application/driver to get/set the following parameters:

- Smart card Power-on sequence
- Delay between successive Activation retries
- Enable/Disable any Voltage Class

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

Input:

The first byte of the input buffer contains the Escape code. The next byte contains the function to be performed. Third byte contains the parameter for the function.

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

Output:

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

Byte0 Value	Description	Byte1	Byte2
0x00	Starts with Class C voltage. (1.8V – 3V – 5V order)	Time delay between resets in milliseconds	Bit Map of all Voltage Classes [Bit0 – Class A; Bit1 – Class B; Bit2 – Class C]
0x01	Starts with Class A voltage. (5V – 3V – 1.8V order)		

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

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

Example: retrieve all the current settings:

DataIn = **04 FE**
 DataOut: **01 0A 07** (3 bytes)
 00: Starting with Class A
 0A: 10ms delay between resets
 07: Class A, B, and C enabled

6.1.2.4: CONTACT_EMV_LOOPBACK

This Escape command lets the host force the firmware to perform an EMV Loop-back application.

The input buffer is

Byte0
Escape code(0x05)

The output buffer is

Output buffer
NULL

6.1.2.5: CONTACT_EMV_SINGLEMODE

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

The input buffer is

Byte0
Escape code(0x06)

The output buffer is

Output buffer
NULL

6.1.2.6: CONTACT_APDU_TRANSFER

This Escape command exchanges a short APDU with the smart card. The user has to ensure that a card is inserted and powered before issuing this Escape command. This Escape command mostly is used by the MCard API to access synchronous memory cards.

Input:

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

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

The output buffer contains the response APDU.

Output buffer
Response APDU

6.1.2.7: CONTACT_CONTROL_PPS

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

Input:

The first byte of input buffer contains the Escape code and the following byte, if 1 disables the PPS and if 0 enables the PPS.

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

The output buffer is

Output buffer
NULL

6.1.2.8: CONTACT_EXCHANGE_RAW

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

Input:

The input buffer for this command will contain the Escape code, low byte of the length of data to be sent, high byte of length of data to be sent, low byte of the length of expected data, high byte of length of expected data and the command.

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

Output:

Output buffer
Response APDU

6.1.2.9: READER_GET_IFDTYPE

This Escape command is used to get the current PID type from the reader.

Input:

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

Byte0
Escape code(0x12)

Output:

The reader returns the PID of the firmware which can be used to identify the reader.

PID value		Description
B0	B1	
0x16	0x51	USB PID of Identiv SCR3310 v2 USB Smart Card Reader

6.1.2.10: READER_LED_CONTROL

This Escape command may be 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 while using this IOCTL else the LED state will be overwritten by the FW LED behavior.

Input:

The first byte of the input buffer contains the Escape code, followed by LED number always set to 0 (just one LED) and then the desired LED state.

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

Output:

Output buffer
NULL

6.1.2.11: READER_LED_CONTROL_BY_FW

This command is used to enable/disable LED control by firmware. Default setting is: LED is controlled by firmware.

Input:

The first byte of the input buffer contains the Escape code. The second byte specifies if LED control by the 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
Get State	FF	0 -- LED control by firmware enabled 1 -- LED control by firmware disabled

Output:

No response is returned for set state. For Get State 1 byte response is received.

Output buffer
NULL or current state

6.1.2.12: READER_GET_INFO_EXTENDED

This Escape command may be used to retrieve extended information about the reader and supported features.

Input:

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

Byte0
Escape code(0x1E)

Output:

The firmware returns data as per structure SCARD_READER_GETINFO_PARAMS_EX mentioned below. This Escape command is used to get the firmware version, reader capabilities, and Unicode serial number of the reader.

Field Size in Bytes	Field Name	Field Description	Default value
1	byMajorVersion	Major Version in BCD	Based on current firmware version
1	byMinorVersion	Minor Version in BCD	
1	bySupportedModes	Total no of supported modes in the reader	0x03 (ISO, EMV and MCard modes)
2	wSupportedProtocols	Protocols supported by the Reader Bit 0 – T0 Bit 1 – T1	0x0300 (LSB first)
2	winputDevice	IO_DEV_NONE 0x00 IO_DEV_KEYPAD 0x01 IO_DEV_BIOMETRIC 0x02	0x0000 (LSB first)
1	byPersonality	Reader Personality (Not Used)	0x00
1	byMaxSlots	Maximum number of slots	0x01 (Single slot device)
1	bySerialNoLength	Serial number length	0x1C
28	abySerialNumber [28]	Unicode serial number	Reader serial number(MSB first)

DataIn = 1E

DataOut: 01 00 03 03 00 00 00 00 01 1C 35 00 33 00 36 00
 39 00 31 00 33 00 30 00 31 00 32 00 30 00 30 00
 30 00 36 00 32 00 (38 bytes)

6.1.2.13: CONTACT_GET_SET_CLK_FREQUENCY

In case when an application wants to get or set the smart card clock frequency, this Escape command is used to instruct the reader to change the clock or to get the current Clock divisor used. Once set, the change in frequency will take effect immediately. Default divisor value is 10, that is 4.8MHz.

Input:

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

Byte0	Byte1 Value	Description
Escape code(0x1F)	Clock divisor	The value to be Set in the smart card CLK divisor register
	0xFF	Get current Clock divisor value

Output:

Set clock frequency: None

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

Output buffer
NULL or current divisor

Clock Divisor values:

DIVISOR VALUE	SCCLK Frequency
12	4 MHz
10	4.8 MHz
8	6 MHz
7	6.8 MHz
6	8 MHz
5	9.6 MHz
4	12 MHz
3	16 MHz

DataIn = 1F FF

DataOut: 0A (1 byte)

6.1.2.14: CONTACT_GET_SET_ETU

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

Input:

The input buffer contains the Escape followed by an 8 bit GET/SET identifier. For SET ETU, a DWORD specifying the value to be set is following.

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

Output:

For both Set and Get ETU, the output will be the following.

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

DataIn = 80 00

DataOut: 00 00 01 40 (4 bytes)

6.1.2.15: CONTACT_GET_SET_WAITTIME

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

Input:

The input buffer contains the Escape code followed by an 8 bit GET/SET identifier, an 8 bit Wait time identifier and a 32 bit Wait time value. BWT must be specified in units of 1.25ms and CWT in units of ETU.

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

Output:

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

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

DataIn = 81 00 01

DataOut: 00 00 03 5D (4 bytes)

6.1.2.16: CONTACT_GET_SET_GUARDTIME

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

Input:

The input buffer contains the Escape code followed by an 8 bit GET/SET identifier, an 8 bit guard time identifier and a 32 bit guard time value in ETU.

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

Output:

For Get/Set guard time, the output will be the Character/Block Guard Time value.

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

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

6.1.2.17: CONTACT_GET_SET_MCARD_TIMEOUT

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

Input:

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

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

Output:

Write delay: No response byte

Read delay value: A byte value specifying the current delay applied during memory card Write in milliseconds

Byte0
Delay in ms

DataIn = 85 00
 DataOut: 00 (1 byte)

6.1.2.18: CONTACT_CONTROL_ATR_VALIDATION

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

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

By default, ATR validation is enabled.

Input:

The first byte of the input buffer will contain the Escape code; the next byte will contain the control byte.

Byte0	Byte 1	
	Value	Description
Escape code(0x88)	0x00	Enable ATR validation
	0x01	Disable ATR validation

Output:

Output buffer
NULL

7: Annexes

7.1: A – Status words table

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

7.2: Annex B – Sample code using Escape commands through Escape IOCTL

File Name : SCR3310 V2 Escape.h

```
#ifndef _SCR3310V2_ESCAPE_H_
#define _SCR3310V2_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 CONTACT_GET_SET_POWERUPSEQUENCE 0x04
#define CONTACT_EMV_LOOPBACK            0x05
#define CONTACT_EMV_SINGLEMODE          0x06
#define CONTACT_APDU_TRANSFER           0x08
#define CONTACT_CONTROL_PPS             0x0F
#define CONTACT_EXCHANGE_RAW            0x10
#define READER_GETIFDTYPE                0x12
#define READER_LED_CONTROL               0x19
#define READER_LED_CONTROL_BY_FW        0xB2
#define READER_GETINFO_EXTENDED         0x1E
#define CONTACT_GET_SET_CLK_FREQUENCY   0x1F
#define CONTACT_GET_SET_ETU              0x80
#define CONTACT_GET_SET_WAITTIME        0x81
#define CONTACT_GET_SET_GUARDTIME       0x82
#define CONTACT_GET_SET_MCARD_TIMEOUT   0x85
#define CONTACT_CONTROL_ATR_VALIDATION   0x88

#ifdef __cplusplus
}
#endif

#endif
```

File Name : SCR3310 V2 Escape.c

```
#include <windows.h>
#include <winbase.h>
#include <stdio.h>
#include <conio.h>
#include "winscard.h"
#include "winerror.h"
#include "SCR3310 V2 Escape.h"

VOID main(VOID)
{
    SCARDCONTEXT          ContextHandle;
    SCARDHANDLE           CardHandle;
    ReaderInfoExtended    strReaderInfo;
    BYTE                  InByte, i;
```

```

DWORD          BytesRead, ActiveProtocol;
ULONG          ret;
char           *ReaderName[] = {"Identiv SCR3310 V2 Smart Card Reader 0",
                                NULL};

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

ContextHandle = -1;

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

if (ret == SCARD_S_SUCCESS)
{
    ret = SCardConnect( ContextHandle,
                        ReaderName[0],
                        SCARD_SHARE_DIRECT,
                        SCARD_PROTOCOL_UNDEFINED,
                        &CardHandle,
                        &ActiveProtocol);

    if (ret == SCARD_S_SUCCESS)
    {
        InByte = 0x1E;
        ret = SCardControl( CardHandle,
                            IOCTL_CCID_ESCAPE,
                            &InByte,
                            1,
                            &strReaderInfo,
                            sizeof(strReaderInfo),
                            &BytesRead);
        if (SCARD_S_SUCCESS == ret) {
            printf("major version:\t\t%d%d\n", (strReaderInfo.byMajorVersion & 0xF0)>>
4, (strReaderInfo.byMajorVersion & 0x0F));
            printf("minor version:\t\t%d%d\n", (strReaderInfo.byMinorVersion & 0xF0)>>
4, (strReaderInfo.byMinorVersion & 0x0F));
            printf("modes:\t\t\t%d\n", strReaderInfo.bySupportedModes);
            printf("protocols:\t\t%04x\n", strReaderInfo.wSupportedProtocols);
            printf("input device:\t\t%04x\n", strReaderInfo.winInputDevice);
            printf("personality:\t\t%d\n", strReaderInfo.byPersonality);
            printf("max slots:\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++)
                printf("%c", strReaderInfo.abSerialNumber[i]);

        } else {
            printf("SCardControl failed: %08X\n", ret);
        }
    }
    else {
        printf("SCardConnect failed: %08X\n", ret);
    }

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

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

```

7.3: Annex C – Mechanical drawings

7.3.1: Top Casing

