Reference Manual for SPR332 v2
Class 2 Secure PINpad Reader

For Part #: **905127-1**

Document Version: 1.1, Last Revised On: 2018-07-20

Abstract

This document contains in-depth information about the hardware and software features of the SPR332 v2 Contact Secure PINpad Reader.

Audience

This document is intended for system integrators and software developers.

Revision History

Rev.	Date	Description
1.1	2017-09-22	First published external version, corresponding to firmware version 7.00

Contact Information

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

Table of Contents

1. Legal Information

1.1. Disclaimers

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

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

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

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 Secure PINPad reader (SPR332 v2).

This manual describes in detail the interfaces and supported commands available for developers using SPR332 v2 in their applications.

2.2. Target Audience

This document describes the technical implementation of SPR332 v2, and targets software developers. It assumes knowledge about ISO 7816, 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	1.00
Firmware	7.00

2.4. Definition of Various Terms and Acronyms

Term or Acronym	Definition
APDU	Application Protocol Data Unit
ATR	Answer To Reset, defined in ISO7816.
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
NA	Not Applicable
NAD	Node Address
Nibble	Group of 4 bits. 1 digit of the hexadecimal representation of a byte. *Example:* 0xA3 is represented in binary as (10100011)b. The least significant nibble is 0x3 or (0011)b and the most significant nibble is 0xA or (1010)b.
PC/SC	Personal Computer/Smart Card: software interface to communicate between a personal computer 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 $\in$ {0,1}.
0xYY	The byte value YY is represented in hexadecimal.

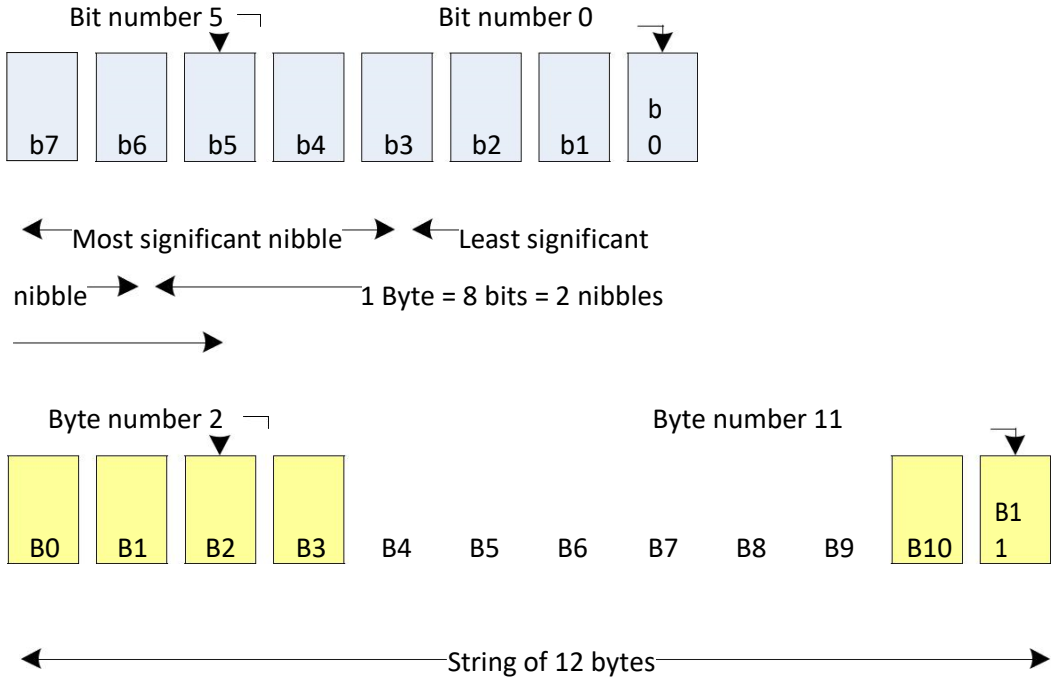
2.5. References to Other Documents

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: 2005 (E)	ISO/IEC
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

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 0xA3is

- 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 SPR332 v2 Readers

3.1. Key Benefits

The SPR332 v2 is a Class 2 contact smart card reader, featuring a secure PIN entry. The reader allows authentication processes to be securely executed within the device protecting the entered data from various attacks. The reader is extremely useful with secure applications where securing of private data is the key.

The SPR332 v2 is compatible with the software standards CT-API and PC/SC as well as the Microsoft USB CCID protocol. With this reader you can use all ISO7816- and EMV-compatible cards, e.g. the German HBCI card and value card as well as all signature -law-conformal smart cards.

3.2. Key Features

- ISO/IEC 7816 compliant smart card reader
- PC/SC v2.0 compliant
- Secure PIN entry to allow securely executed authentication processes within the device, protecting the entered PIN from various attacks (PCSC2.0 part 10 compliant APDU and IOCTL)
- Secure in-field firmware upgrade
- Unique reader serial number which enables a SPR332 v2 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 600 Kbps with support for TA1=97 cards

3.3. Ordering Information

Item	Part number	Product Image
SPR332 v2	905127-1	

3.4. SPR332 v2 customization options

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

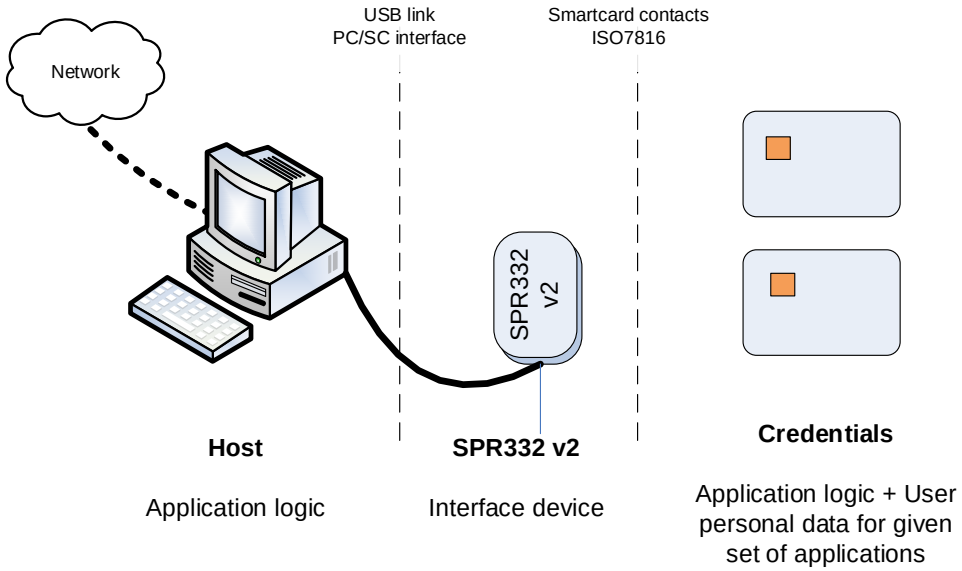
- The color of the casing
- The logo
- The product label
- The USB strings

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

3.5. Applications

3.5.1. General

SPR332 v2 is a transparent reader designed to interface a personal computer host supporting PC/SC interface with smart cards according to ISO/IEC 7816 like CAC viz., PKI cards, banking cards and health insurance cards.



SPR332 v2 itself handles the communication protocol but not the application related to the credential. 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 SPR332 v2.

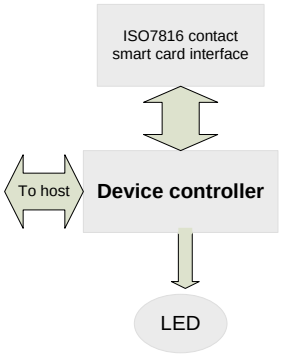
Identiv does not provide PKI or CAC applications.

4. SPR332 v2 characteristics

4.1. SPR332 v2 high level architecture

4.1.1. Block diagram

The link between SPR332 v2 and the host to which it is connected is the USB interface providing both the power and the communication channel.



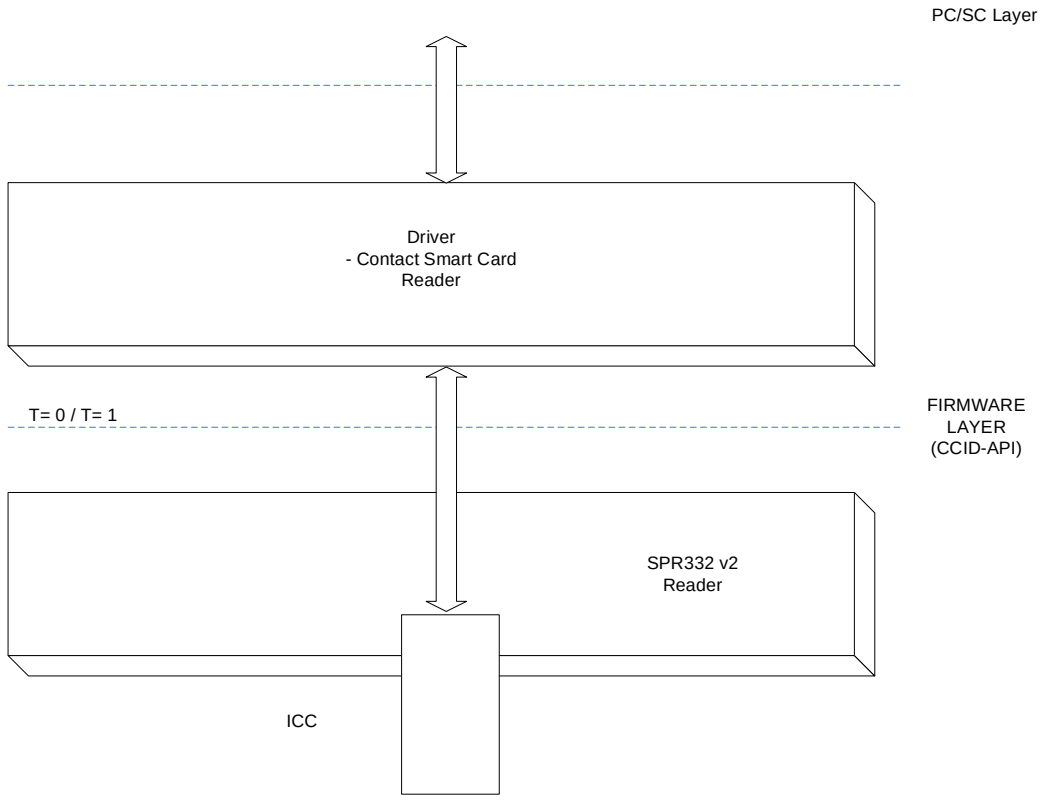
The SPR332 v2 device controller has several interfaces available. In the SPR332 v2 implementation three peripherals are connected to the device controller:

- LED for reader status indication
- A smart card interface
- Secure PINpad

The µController 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 SPR332 v2 leverages a PC/SC CCID driver that is freely available for all supported operating systems (Windows, macOS X and Linux).
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.

5. Secure PIN-entry feature

Using the integrated keypad, the SPR332 v2 allows the user a secure PIN-entry and is therefore suitable for HBCI banking and signature-law-conformal applications in the home and office environment. This feature must be supported by the application.
The secure PIN-entry ensures that the PIN is only transmitted to the smart card and never leaves the reader towards the PC (host). Contact your bank for further information regarding HBCI and the required smart card.

The secure PIN-entry is only guaranteed when using the numeric keypad. During the entry, you can verify the secure PIN-entry mode by checking the LED-status of the SPR332 v2 (table "LED-status and acoustic signals"). Confirm the PIN with [✓] or cancel the entry with **[✖]**. You can erase and re-enter the PIN by pressing the [**C**] key.

Important: The PC keyboard is not suitable for a secure PIN-entry. Always use the integrated keypad of the smart card reader.

6. Security Features

Before setting up the SPR332 v2 examine whether the housing and the seal on the lower surface are intact. The hardware is developed in such a way that when opening the housing, the internal mounting plates break off and the seal is destroyed. Thus, any kind of tampering can easily be recognized. In case of suspicion of tampering please refer to your dealer and/or to your local distributor.

6.1. Security functions of the SPR332 v2

6.1.1. Secure PIN-entry

The SPR332 v2 is a class 2 card reader. This means that the PIN verification is performed between the smart card reader and the smart card. As the PIN never leaves the smart card reader towards the host PC it is impossible for other applications (e.g. spy software) to get knowledge of the PIN. To guarantee a maximum level of security the PIN is erased from the SPR332 v2 memory immediately after the session.

6.1.2. Tampering

On the bottom, the SPR332 v2 is equipped with a falsification safe seal. This allows you to recognize if someone opened the housing to perform any manipulations of the hardware. Verify regularly if the seal is still intact (chapter 5.2.3). Also make sure that the SPR332 is directly attached with the respective USB Port. There must be no other devices between the PC and the smart card reader with the exception of a potential USB hub.

6.2. Usage security recommendations

To get the most out of the security functions of the SPR332 v2, please read the following recommendations carefully. Aside from keeping the PIN secret and a secure installation environment, the user is also responsible for a regular examination of all security functions of the SPR332 v2. In the case of suspected tampering, the user takes full responsibility to keep using the device as it may affect

the security features. It is recommended not to use it anymore and to contact your dealer or your local Identiv office. The SPR332 v2 is exclusively designed for home and office usage, therefore not suitable for any kind of public accessible areas.

6.2.1. Secure environment

Install the SPR332 v2 in a way that any access by unauthorized people is avoided and an unobserved PIN-entry is guaranteed.

6.2.2. Verification of the seal

The SPR332 v2 is equipped with a falsification safe seal on its bottom with four predetermined breaking points. Any attempts to remove it will result in an irreversible damage of the seal. A damaged seal exhibits a regular hexagon structure, which can be best recognized by a lateral view of the foil. In order to detect any unauthorized manipulations of the device, please regularly verify if the seal is still intact.

fig. seal on bottom of the SPR332 v2
left – normal seal; right – tampered seal

6.2.3. Secure handling of the PIN

The PIN as a personal secret allows you to access your smart card. Therefore, it is important that no one but you has knowledge of the PIN. Always ensure that the PIN is entered unobserved. Never write the PIN down on the smart card or on any other places that are accessible by other people. Do not enter the PIN via the keyboard of the PC. Always use the keypad of the SPR332 v2.

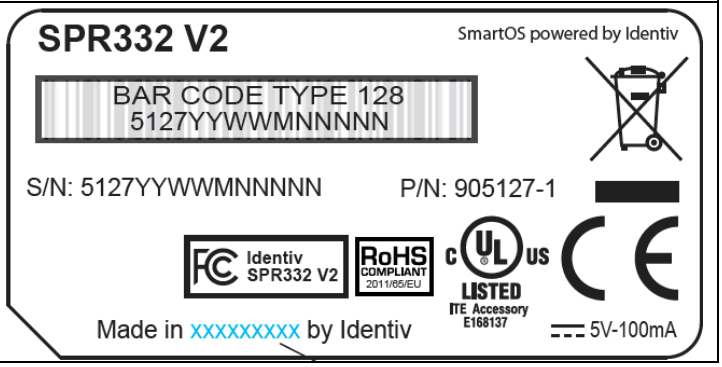
During the PIN-entry ensure that the SPR332 v2 is working in "Secure PIN-entry mode". You can verify this by checking the LED status (table "LED-status and acoustic signals").

6.2.4. HBCI and signature cards

When using the SPR332 v2 with home banking, other finance software or digital signature applications, remove your smart card immediately after finishing the transaction or signature. Also keep your smart card in a safe place.

6.3. Quick reference data

6.3.1. SPR332 v2 dimensions

Item	Characteristic	Value
SPR332 v2	Weight	250g
	External dimensions	120 x 70 x 40 mm
	Cable length	1.5 meter long with USB type A connector
	Default color	white
	Default label	

6.3.2. LED behavior and acoustic signals (buzzer)

The SPR332 v2 has two LEDs and a buzzer to indicate the actual operating mode or possible malfunctions. Always install the SPR332 v2 in a way that you can see both LEDs. Together with the buzzer, they indicate the following states:

Status	LED1 (green)	LED2 (orange)	buzzer	
Power on, DFU* finished	off	off	740 Hz/25 ms	
Reader active	on 0.5 s off 4.5 s	off	off	
Smart card active	on	off	off	
Smart card communication	on 0.5 s off 0.5 s	off	off	
Secure PIN-entry active	on	on 0.5 s off 0.5 s	0-9	2400 Hz/25 ms
				1100 Hz/25 ms
				1100 Hz/25 ms
				1100 Hz/25 ms
			PIN-entry successful	900 Hz/100 ms 1200 Hz/100 ms
			PIN-entry failed	300 Hz/100 ms
PIN-entry successful, smart card active	on	on	off	
PIN-entry successful, smart card communication	on 0.5 s off 0.5 s	on	off	
smart card communication failure	on 100 ms off 100 ms	previous state	off	
self-test failed during boot	off	off	off	
DFU* running	off	off	off	

*Device firmware upgrade

6.3.3. Other data

6.3.3.1. General

Parameter	Value/Description
Clock of the device controller	48 MHz
API	PC/SC 2.0
Operating temperature range	0º to 50ºC
Operating humidity range	Up to 95%RH non-condensing
Certifications	USB CE UL FCC WEEE RoHS2 REACH WHQL

6.3.3.2. USB

Parameter	Value/Description
DC characteristics	Low bus powered (SPR332 v2 draws power from USB bus) Voltage: 5V Max. Current: 12mA + current consumed by an inserted card Suspend current: 800uA
USB specification	USB 2.0 FS device
USB Speed	Full Speed Device (12Mbit/s)
Device Class	CCID
PID	0xE003
VID	0x04E6

6.3.3.3. Card interface

Parameter	Value/Description
Smart card operating frequency	up to 12MHz
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^2C)
ISO/IEC 7816 compliant	Yes
EMV 4.2 compliant	Yes
CT-API compliant	Yes
Number of slots	Single smart card slot
Ejection mechanism	Manual

6.3.3.4. PINpad interface

Function	Value/Description
Numeric Keypad 	Numeric Keys 0 - 9
	Clear
	Cancel
	Confirmation

7. Software modules

7.1. 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.

7.2. Utilities

The following utilities are available:

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

Operating systems supported by the tools:

- Windows Server 2012
- Windows 7, Windows 8.1, Windows 10

7.3. Driver

7.3.1. SPR332 v2 listing

SPR332 v2 is listed by PC/SC applications as

- *"Identiv Inc. SPR332 USB Smart Card Reader"*
- *"SCM Microsystems Inc. SPRx32 USB Smart Card Reader" (legacy support)*

Identiv SPR332 v2 uses the PC/SC CCID class driver readily available for all the supported operating systems.

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.

7.3.2. Supported operating systems

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

7.4. 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.

7.5. MCard-API

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

S.No	Synchronous Card	S.No	Synchronous Card	S.No	Synchronous Card
1	SLE4404	9	SLE5542	17	AT24C128SC
2	SLE4428	10	AT24C01ASC	18	AT24C256SC
3	SLE4432	11	AT24C02SC	19	AT24C512SC
4	SLE4436	12	AT24C04SC	20	AT88SC153
5	SLE6636	13	AT24C08SC	21	AT88SC1608
6	SLE4442	14	AT24C16SC	22	ST14C02
7	SLE5532	15	AT24C32SC		
8	SLE5536	16	AT24C64SC		

7.6. Firmware

7.6.1. CCID transport protocol

SPR332 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.

7.6.1.1. CCID class requests supported

- Abort

7.6.1.2. 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

7.6.1.3. 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:

7.6.1.3.1. HW_ERROR

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

7.6.1.3.2. XFR_PARITY_ERROR

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

7.6.1.3.3. 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.

7.6.1.3.4. BAD_ATR_TS

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

7.6.1.3.5. BAD_ATR_TCK

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

7.6.1.3.6. ICC_MUTE

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

7.6.1.3.7. CMD_ABORTED

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

7.6.1.3.8. Command not supported

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

8. Commands description

8.1. Escape commands for the SPR332 v2

A developer can use the “SCardControl” to send Escape commands defined in PC/SC API. When using the Identiv supplied driver, special registry setting will not be necessary.

However with the inbox CCID driver for Windows, in order to be able to send Escape commands to the SPR332 v2, this 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_E003\ Device-Instance-xxxx \Device Parameters

For Windows 7 and above, that would be HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\USB\VID_04E6&PID_E003\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.

8.1.1. Escape command codes

Escape commands can be used by an application to configure SPR332 v2 to function in a mode that is not its default configured mode or to get specific information. To put the SPR332 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 SPR332 v2

Escape command	Code
READER_SETMODE	0x01
READER_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_GETIFDTYPE	0x12
READER_LED_CONTROL	0x19
BUZZER_CONTROL	0x51
READER_LED_CONTROL_BY_FW	0xB2
READER_GETINFO_EXTENDED	0X1E
CONTACT_GET_SET_CLK_FREQUENCY	0x1F
CONTACT_CONTROL_ATR_VALIDATION	0x88
READER_GENERIC_ESCAPE	0xFF 0x70 0x04 0xE6 XX
CONTACT_GET_SET_ETU	READER_GENERIC_ESCAPE(0x80)
CONTACT_GET_SET_WAITTIME	READER_GENERIC_ESCAPE(0x81)
CONTACT_GET_SET_GUARDTIME	READER_GENERIC_ESCAPE(0x82)
CONTACT_GET_SET_MCARD_TIMEOUT	READER_GENERIC_ESCAPE(0x85)

8.1.1.1. READER_SETMODE

This Escape command sets the current mode of the reader. Applications may call this function, to set the desired mode. Typically, this call is used to switch between the ISO/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

8.1.1.2. READER_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)

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

Output:

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

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

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

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

Example: retrieve all 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

8.1.1.4. 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

8.1.1.5. 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

8.1.1.6. 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

8.1.1.7. 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

8.1.1.8. 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

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

PID value		Description
B0	B1	
0x03	0xE0	USB PID of Identiv SPR332 v2 smart card Reader

8.1.1.10. 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 and then the desired LED state.

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

Output:

Output buffer
NULL

8.1.1.11. 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

8.1.1.12. 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 and hence data will be retained even after power cycle.

Note:

- Frequent writes should be avoided (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 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 if user area is to be read or written as described below.

Byte0	Byte1		Byte2 to Byte251
	Value	Description	
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

8.1.1.13. 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 01 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)
```

8.1.1.14. 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
0x04	4 MHz
0x03	4.8 MHz
0x02	6 MHz
0x01	8 MHz
0x00	12 MHz

```
DataIn = 1F FF
DataOut: 03 (1 byte)
```

8.1.1.15. 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	Byte6		Byte7	Byte 8	Byte 9	Byte10
						Value	Description	Wait time			
0xFF	0x70	0x04	0xE6	0x07 (LEN) or 0x02	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 = FF 70 04 E6 02 80 00
DataOut: 00 00 01 40 (4 bytes)
```

8.1.1.16. 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.

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte6		Byte7		Byte8	Byte9	Byte 10	Byte 11
						Value	Description	Value	Description	Wait time in ETU			
0xFF	0x70	0x04	0xE6	0x07 (LEN) or 0x03	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 = FF 70 04 E6 03 81 00 01
DataOut: 00 00 03 5D (4 bytes)
```

8.1.1.17. CONTACT_GET_SET_GUARDTIME

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

Input:

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

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte5	Byte6		Byte7		Byte8	Byte9	Byte 10	Byte 11
						Value	Description	Value	Description	Guard time in ETU			
0xFF	0x70	0x04	0xE6	0x07 (LEN) or 0x03	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 = FF 70 04 E6 03 82 00 01
DataOut: 00 00 00 18 (4 bytes)
```

8.1.1.18. 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	Byte2	Byte3	Byte4	Byte5	Byte6	
						Value	Description
0xFF	0x70	0x04	0xE6	0x03 (LEN) or 0x02	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 = FF 70 04 E6 02 85 00
DataOut: 00 (1 byte)
```

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

Output:

Output buffer
NULL

8.1.1.20. CONTACT_READ_INSERTION_COUNTER

This Escape command is supported through the READER_GENERIC_ESCAPE command and retrieves the number of times a contact smart card has been inserted into the reader.

Input:

The first five bytes of the input buffer follow APDU structure as per [PCSC3-AMD1]. The 6thbyte is the Escape code 0x00 to identify the command.

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Le
0xFF	0x70	0x04	0xE6	0x01	0x00 (Escape code)	4 (Insertion counter is a 4 byte value)

Output:

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5
Insertion counter value				SW1	SW2
BIT31-BIT24	BIT23-BIT16	BIT15-BIT8	BIT7-BIT0	0x90	0x00

In case of any error, only SW1 and SW2 set with error status will be returned.

8.1.1.21. READER_GENERIC_ESCAPE

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

Input:

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

Byte0	Byte1	Byte2	Byte3	Byte4	From Byte5 (up to Lc bytes)		ByteLc+5
					Byte 5	Byte 6 onwards	
0xFF	0x70	0x04	0xE6	Lc (always > 0)	Cmd Opcode	Command parameters or data	Le (optional)

Output:

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

8.1.1.22. BUZZER_CONTROL

This Escape command is used to enable or disable the buzzer in spr332.By default, Buzzer is enabled at power on.

Input:
The first byte of the input buffer will contain the Escape code followed by the control byte to enable or disable buzzer.

Byte0	Byte1	
	Value	Description
Escape code(0x51)	0x00	Disable Buzzer
	0x01	Enable Buzzer

Output:

Output buffer
NULL

9. Annexes

9.1. 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

9.2. Annex B – Sample code using Escape commands through Escape IOCTL

```
File Name :  uTrust_Escape.h

#ifndef _uTrust_ESCAPE_H_
#define _uTrust_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 :  uTrust_Escape.c

#include <windows.h>
#include <winbase.h>
#include <stdio.h>
#include <conio.h>
#include "winscard.h"
#include "winerror.h"
#include "uTrust_Escape.h"

VOID main(VOID)
{
   SCARDCONTEXT         ContextHandle;
   SCARDHANDLE          CardHandle;
   ReaderInfoExtended   strReaderInfo;
   BYTE                 InByte, i;
   DWORD                BytesRead, ActiveProtocol;
   ULONG                ret;
   char                 *ReaderName[] = {"SCM Microsystems Inc. SPRx32 USB 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.winputDevice);
            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.abySerialNumber[i]);

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

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

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