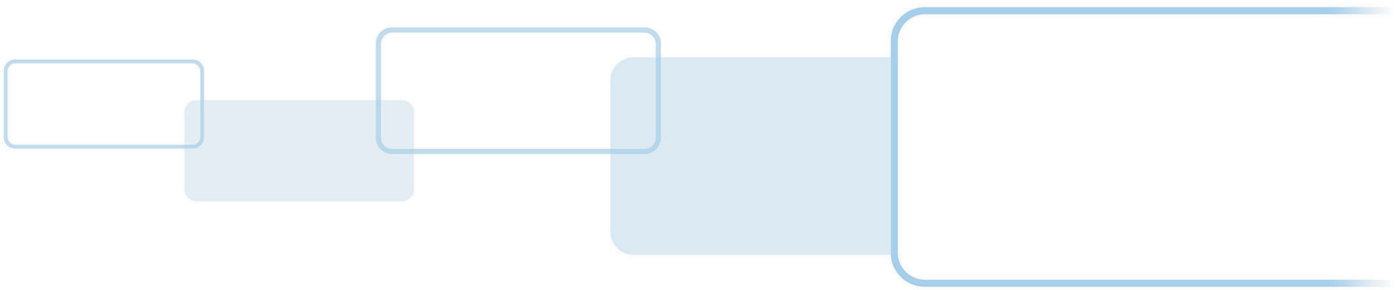
OMNIKEY® 5027
Software Developer Guide

PLT-03824, Rev. A.0

May 2018



hidglobal.com

Copyright

Trademarks

Revision history

Date	Description	Revision
May 2018	Initial release.	A.0

Contacts

For additional offices around the world, see **www.hidglobal.com/contact/corporate-offices**

Americas and Corporate

611 Center Ridge Drive
Austin, TX 78753
USA
Phone: 866 607 7339
Fax: 949 732 2120

Asia Pacific

19/F 625 King's Road
North Point, Island East
Hong Kong
Phone: 852 3160 9833
Fax: 852 3160 4809

Europe, Middle East and Africa (EMEA)

Haverhill Business Park Phoenix Road
Haverhill, Suffolk CB9 7AE
England
Phone: 44 (0) 1440 711 822
Fax: 44 (0) 1440 714 840

Brazil

Condomínio Business Center
Av. Ermano Marchetti, 1435
Galpão A2 - CEP 05038-001
Lapa - São Paulo / SP
Brazil
Phone: +55 11 5514-7100

HID Global Technical Support: www.hidglobal.com/support

Contents

Section 1: Introduction 9

1.1 Product description 9

1.1.1 Key features 9

1.1.2 Reference documents 10

1.1.3 Abbreviations and definitions 11

1.1.4 Driver installation 11

1.1.5 HID OMNIKEY Workbench 11

Section 2: Host interfaces and human interfaces 13

2.1 USB 13

2.1.1 Endpoints assignment 13

2.1.2 Enumeration mode switching mechanism 13

2.2 Human interfaces 14

2.2.1 LEDs 14

2.2.2 Buzzer 14

Section 3: Keyboard 15

3.1 Keyboard wedge 15

3.1.1 Output generation process 15

3.1.2 Multiple output 15

3.1.3 Data manipulation 15

3.1.3.1 Bit reverse 15

3.1.3.2 Byte reverse 15

3.1.3.3 Range limit 15

3.1.3.4 Offset 16

3.1.3.5 Example of PACS data manipulation 17

3.1.4 Data formatting and output string generation 17

3.1.5 Character mapping into keystrokes 17

3.1.6 Special command characters 18

3.1.7 Entering extended ASCII characters 18

Section 4: Configuration card 19

4.1 Security 19

4.2 Key management 19

4.3 Configuration download 19

Section 5: Contactless card interface 21
5.1 Polling mode 21
5.2 Polling and power consumption 21
Section 6: Contactless credential support 23
6.1 Contactless protocols 23
6.2 Supported credentials 23
6.3 Physical access control (PACS) data 24
6.4 Dual/combo cards support 24
6.4.1 Cards with two interfaces using different protocols 24
6.4.2 Cards with MIFARE emulation 24
6.4.3 Cards with two interfaces using same protocol 24
Section 7: PC/SC 25
7.1 How to access the reader through PC/SC 25
7.1.1 Step 1: Establish context 25
7.1.2 Step 2: Get status change 25
7.1.3 Step 3: List readers 26
7.1.4 Step 4: Connect to the card 26
7.1.5 Step 5: Exchange data and commands with the card or the reader 26
7.1.6 Step 6: Disconnect 27
7.1.7 Step 7: Release 27
7.2 Vendor-specific commands 28
7.2.1 Command APDU 28
7.2.2 Response APDU 28
7.2.3 Error response 29
Section 8: Communicating with the reader 31
8.1 OMNIKEY specific commands 31
8.1.1 OMNIKEY specific command APDU format 31
8.1.2 Response for OMNIKEY specific commands 31
8.1.2.1 Vendor payload command types 31
8.1.3 Response APDU 32
8.1.4 Error response 32
8.1.5 Reader information API 33
Section 9: Reader configuration 35
9.1 APDU commands 35
9.2 Accessing configuration 36
9.2.1 Reader information structure 36
9.2.2 Example: Get reader information 37

9.3 Reader capabilities . . . 38
9.3.1 tlvVersion . . . 38
9.3.2 deviceID . . . 39
9.3.3 productName . . . 39
9.3.4 productPlatform . . . 39
9.3.5 enabledCLFeatures . . . 40
9.3.6 firmwareVersion . . . 40
9.3.7 hfControllerVersion . . . 41
9.3.8 hardwareVersion . . . 41
9.3.9 hostInterfaceFlags . . . 41
9.3.10 numberOfContactSlots . . . 42
9.3.11 numberOfContactlessSlots . . . 42
9.3.12 numberOfAntennas . . . 42
9.3.13 humanInterfaces . . . 43
9.3.14 vendorName . . . 43
9.3.15 exchangeLevel . . . 43
9.3.16 serialNumber . . . 44
9.3.17 hfControllerType . . . 44
9.3.18 sizeOfUserEEPROM . . . 44
9.3.19 firmwareLabel . . . 45
9.3.20 configCardVerSupport . . . 45
9.4 Keyboard wedge configuration . . . 46
9.4.1 Keyboard wedge output configuration . . . 47
9.4.1.1 KBWCardType . . . 47
9.4.1.2 KBWOutputFormat . . . 48
9.4.1.3 KBWFlags . . . 49
9.4.1.4 KBWRangeStart . . . 50
9.4.1.5 KBWRangeLen . . . 50
9.4.1.6 KBWPostStrokeStart . . . 51
9.4.1.7 KBWPrePostStrokes . . . 51
9.4.2 Keyboard country definition . . . 52
9.4.2.1 UseSecondKeyboardLayout . . . 52
9.4.2.2 charactersDiff . . . 53
9.4.3 Extended ASCII character support . . . 54
9.4.3.1 OSforExtendedChars . . . 54
9.5 Contactless slot configuration . . . 55
9.5.1 Contactless slot configuration structure . . . **55**
9.5.2 Baud rates . . . 55

9.5.2.1 Examples . . . 55
9.5.2.2 Default values . . . 55
9.5.3 Common parameters . . . 56
9.5.3.1 sleepModePollingFrequency . . . 56
9.5.3.2 sleepModeCardDetectionType . . . 56
9.5.3.3 emdSupressionEnabled . . . 57
9.5.3.4 configCardEnable . . . 57
9.5.4 ISO/IEC 14443 Type A . . . 58
9.5.4.1 iso14443aRxTxBaudRate . . . 58
9.5.4.2 mifareClassicEmulationPreferred . . . 58
9.5.5 ISO/IEC 14443 Type B . . . 59
9.5.5.1 iso14443bRxTxBaudRate . . . 59
9.5.6 FeliCa . . . 59
9.5.6.1 felicaRxTxBaudRate . . . 59
9.5.7 iCLASS . . . 60
9.5.7.1 iClass15693DelayTime . . . 60
9.5.7.2 iClass15693Timeout . . . 60
9.5.7.3 iClassActallTimeout . . . 61
9.6 Hardware configuration . . . 62
9.6.1 LED . . . 62
9.6.1.1 defaultLEDstate . . . 62
9.7 Reader EEPROM . . . 63
9.7.1 Read . . . 63
9.7.2 Write . . . 63
9.8 Reader configuration control . . . 64
9.8.1 applySettings . . . 64
9.8.2 restoreFactoryDefaults . . . 64
9.8.3 rebootDevice . . . 64
Section 10: Device specific commands . . . 65
10.1 Configuration card API . . . 65
10.1.1 Configuration card data structure . . . 65
10.1.2 configCardPrepare . . . 66
10.1.3 configCardWrite . . . 67
10.1.4 configCardLoadKey . . . 67
Section 11: ICAO test commands . . . 69
11.1 Command set . . . 69
11.1.1 ICAO commands . . . 69
11.1.2 0x92 - ISO/IEC 14443-2: ISO/IEC 14443-2 command APDU . . . 69

11.1.2.1 ISO/IEC 14443-2 P1 coding . . . 70
11.1.2.2 ISO/IEC 14443-2 response . . . 70
11.1.3 0x94 - transmit pattern command APDU . . . 70
11.1.3.1 ICAO transmit pattern P1 coding . . . 70
11.1.3.2 ICAO transmit pattern P2 coding . . . 71
11.1.3.3 ICAO transmit pattern SW1SW2 response bytes . . . 71
11.1.4 0x96 - ISO/IEC 14443-3 command APDU . . . 72
11.1.4.1 ISO/IEC 14443-3 P1 coding . . . 72
11.1.4.2 ISO/IEC 14443-3 P2 coding . . . 72
11.1.4.3 ISO/IEC 14443-3 SW1SW2 response bytes . . . 72
11.1.4.4 Cases for which data out is command dependent . . . 72
11.1.5 0x9A: ICAO miscellaneous command APDU . . . 73
11.1.5.1 ICAO miscellaneous P1 coding . . . 73
11.1.6 ICAO miscellaneous P2 coding . . . 73
11.1.6.1 ICAO miscellaneous response . . . 73
Section 12: Using the PC_to_RDR_Escape command . . . 75

This page is intentionally left blank.

Section 1

1 Introduction

1.1 Product description

The OMNIKEY® 5027 keyboard wedge smart card reader opens new horizons in easy integration and capabilities for incorporation of credential usage into IT and security systems. The simple reader integration on keyboard wedge interface, the wide variety of supported HF credentials and, the easy configurability via a Windows based tool or through configuration cards makes this an ideal device for quick access to contactless identification cards.

The OMNIKEY 5027 reader features include supporting the common high frequency card technologies, including ISO/IEC 14443 A/B, iCLASS®, MIFARE®, FeliCa and therefore most NFC credentials card technologies. With the newly implemented low power mode and system wakeup support, the OMNIKEY 5027 makes an ideal device for easy credential usage on mobile phones, tablets or mother battery powered computer devices. Field firmware upgradeability, multiple language keyboard support the OMNIKEY 5027 is an easy to use device for basic contactless credentials access.

1.1.1 Key features

- HID (Human Interface Device) support removes the requirement to install drivers on standard operating systems to fully support capabilities of the reader as it's working with a standard keyboard driver already provided by the operating system.
- High frequency card technologies. Supports common high frequency card technologies, including ISO/IEC 14443 A/B, iCLASS® 15693 and MIFARE including DESFire EV1/EV2.
- Rapid and easy integration. No special driver installation is required.
- Multiple language keyboard support. Easy configuration of different keyboard languages.
- Advanced power management. Fully compliant with low power modes specified by USB, allowing the reader to wake up the host device and also go to a state of low power consumption.

1.1.2 Reference documents

Document number	Description
USB 2.0 Specification	Universal Serial Bus Revision 2.0 specification provides the technical details to understand USB requirements and design USB compatible products. Refer to **http://www.usb.org/developers/docs/usb20_docs/**
CCID Specification	Specification for Integrated Circuit(s) Cards Interface Devices. Revision 1.1
PC/SC	PC/SC Workgroup Specifications version 2.01.9 April 2010
PC/SC-3	PC/SC – Part 3 – Requirements for PC Connected Interface Devices V2.01.09, June 2007
ISO/IEC 14443-3	Identification cards – Contactless integrated circuit cards – Proximity cards – Part 3: Initialization and anti-collision
5027 Reader Data Sheet	Provides a summary of the OMNIKEY 5027 Reader's features
NIST Special Publication 800-108	Recommendation for Key Derivation Using Pseudorandom Functions

Note: HID Global is not allowed to support proprietary card layer protocols that may be implemented in the host device/application. For example, FeliCa application developers must contact Sony and MIFARE branded products must contact NXP to obtain these card layer protocols. HID Global is constantly expanding credential support in the reader, so some card technologies support only the chip UID. Contact HID Global Technical Support for further information: **https://www.hidglobal.com/support**.

1.1.3 Abbreviations and definitions

Abbreviation	Description
APDU	Application Protocol Data Unit
ATR	Answer To Reset
ATS	Answer To Select
CCID	Integrated Circuit(s) Cards Interface Device
CE	Conformité Européenne
CSN	Card Serial Number
EMD	Electromagnetic Disturbance
FCC	Federal Communications Commission
ICAO	International Civil Aviation Organization
IDm	Manufacture ID (FeliCa UID)
IFD	Interface Device
ISO/IEC	International Organization for Standardization / International Electrotechnical Commission
KBW	Keyboard Wedge
OID	Unique Object Identifier
PC/SC	Personal Computer / Smart Card
PCDs	Proximity Coupling Device
PICC	Proximity Integrated Circuit Card
RFU	Reserved for Future Use
UID	Universal ID
USB	Universal Serial Bus
VCD	Vicinity Coupling Device
VICC	Vicinity Integrated Circuit Card

1.1.4 Driver installation

No extra driver installation is necessary. Every HID Keyboard and CCID compliant driver should work with the reader. However, Microsoft's CCID driver prevents the execution of CCID Escape commands. If an application uses those commands, apply the procedure described in *Section 12 Using the PC_to_RDR_Escape command*.

1.1.5 HID OMNIKEY Workbench

The product is supported by HID OMNIKEY Workbench (version 1.7 and later).

This page is intentionally left blank.

Section 2

2 Host interfaces and human interfaces

The OMNIKEY® 5027 reader supports the USB 2.0 Full Speed (12 Mbit/s) Device Port Host Interface.

2.1 USB

The OMNIKEY 5027 enumerates as a HID Keyboard Device in normal operation mode. To enable device configuration via CCID interface, it can be switched to composite device mode. See *Section 2.1.2 Enumeration mode switching mechanism*. When the device enumerates as a composite device, the OMNIKEY 5027 USB protocol stack implements the following two device classes:

- HID Keyboard
- CCID (Integrated Circuit Cards Interface Device, v1.1) - used only for reader configuration.

The USB CCID interface can be used to send Application Protocol Data Units (APDU) to the reader. The OMNIKEY 5027 supports the standard PC/SC API (for example, `SCardConnect`, `SCardDisconnect`, `SCardTransmit`). Consequently, any application software using the PC/SC API commands should be able to communicate with the reader. However, the CCID slot allows only configuration of the reader; it cannot be used to communicate with the smart card directly.

2.1.1 Endpoints assignment

The table below lists the USB protocol stack endpoints, their parameters and functional assignment:

Endpoint	Description	Type	Max. packet size
EP0	Control Endpoint	Interrupt IN/OUT	64 bytes
EP1	Keyboard	Interrupt IN	8 bytes
EP2	Smart card	BULK OUT	64 bytes
EP3	Smart card	BULK IN	64 bytes
EP4	Smart card	INTERRUPT	8 bytes

2.1.2 Enumeration mode switching mechanism

To switch on CCID mode, send a HID Set Feature Report request with report ID 0x00 and two bytes: 0xA5, 0x5A. The device will re-enumerate in composite device mode with VID 0x076B and PID 0x5A27. It will stay in this mode unless it is physically disconnected from the PC, or a CCID mode off request is sent.

To switch off CCID mode, send a HID Set Feature Report request with report ID 0x00 and two bytes: 0x5A, 0xA5. In keyboard only mode, the device enumerates with VID 0x076B and PID 0x5027.

For implementation examples, see *Section 9 Reader configuration*.

2.2 Human interfaces

2.2.1 LEDs

The OMNIKEY 5027 is equipped with one LED to indicate the current status of the smart card reader. It blinks several times after power-up. The LED also signals communication with the smart card by blinking during a transaction. Its default idle state can be set to ON or OFF.

2.2.2 Buzzer

The OMNIKEY 5027 is equipped with one buzzer with a fixed tone. It is used to signal the completion of configuration card reading, or an error during configuration card reading. The buzzer action can also be triggered by putting special character 0x0A in pre or post strokes, to signal successful card reading.

Section 3

3 Keyboard

The OMNIKEY® 5027 reader supports a keyboard interface according to the USB Device Class Definition for Human Interface Devices (HID Specification) version 1.11. It implements the “Keyboard with BIOS support” USB protocol stack interface.

3.1 Keyboard wedge

3.1.1 Output generation process

The process of generating keyboard wedge output consists of the following steps:

1. Get card type and CSN/UID.
2. Find matching configurations.
3. Obtain PACS data if required.
4. Manipulate data according to configurations.
5. Format data and generate output string.
6. Map string characters into keystrokes.

3.1.2 Multiple output

When a card is presented, the reader checks for all matching configurations. If any of them require PACS data, it is retrieved once per whole reading process. All matching configurations are then output one after another.

Note: If a matching configuration requires PACS data, but the card does not have it, no pre or post strokes will be provided for this configuration.

3.1.3 Data manipulation

Data obtained from the card (either CSN/UID or PACS data) can be manipulated before being sent by keyboard wedge. The following operations can be applied.

3.1.3.1 Bit reverse

Reverses the order of all bits in the data. This operation is applied before range limit or offset.

3.1.3.2 Byte reverse

Reverses the order of all bytes. It is possible that the number of PACS data bits is not a multiple of 8. In this case, data is padded with zeros on the left. This operation is applied after range and offset.

3.1.3.3 Range limit

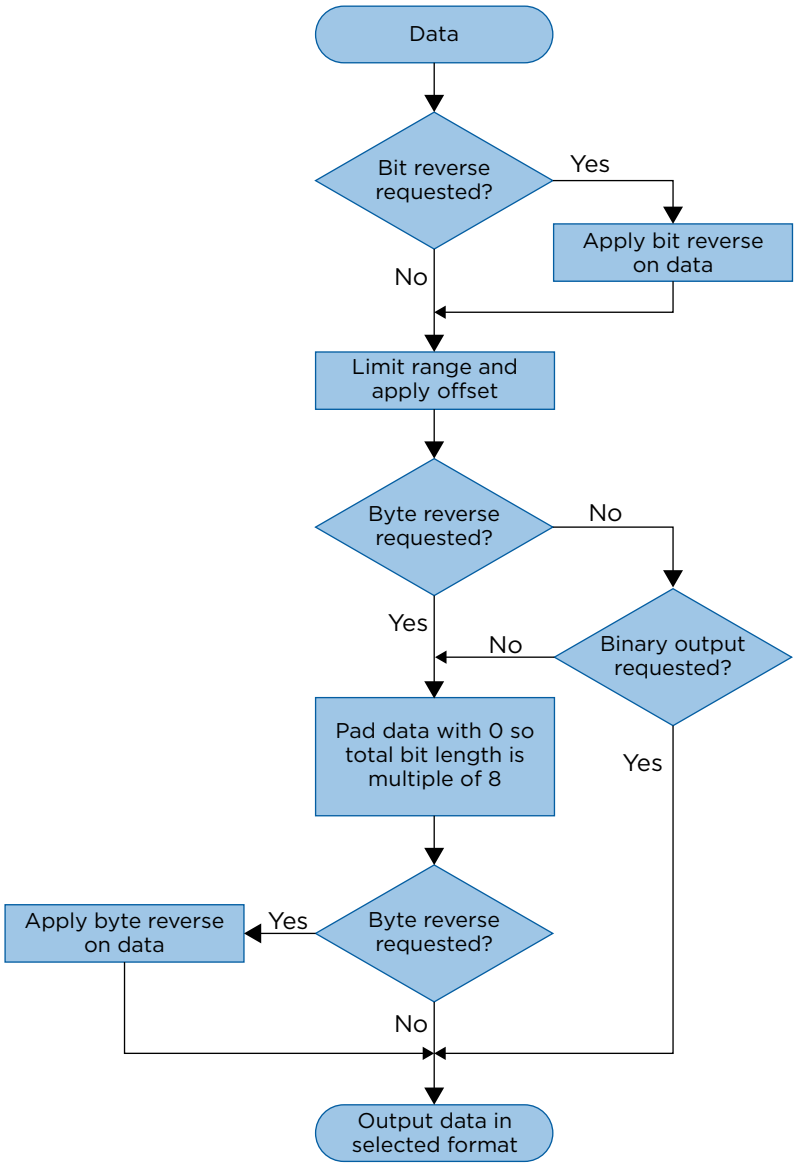
Limits the length of presented data. When requesting PACS information, this parameter is interpreted as bit length, whereas for CSN/UID it is interpreted as bytes.

3.1.3.4 Offset

Offset allows you to skip a certain amount of data. When requesting PACS information, this parameter is interpreted as a number of bits, whereas for CSN/UID it is interpreted as bytes.

Note: Bit and byte reverse operations are mutually exclusive.

The following diagram describes the order of data manipulation operations:



3.1.3.5 Example of PACS data manipulation

	Binary data after manipulation	Comment
PACS data raw	`001 11111111 11111011 10010101 11011111`	All PACS data.
Bit reverse	`11111011 10101001 11011111 11111111 100`	Bits output in reverse order.
Byte reverse	`11011111 10010101 11111011 11111111 00000001`	Bits padded with zeros, then byte order reversed (groups of 8 bits).
Offset 5	`111111 11111011 10010101 11011111`	First 5 bits skipped.
Offset 5, range 15	`111111 11111011 1`	First 5 bits skipped and length limited to 15 bits.
Offset 5, range 15, bit reverse	`011 10101001 1101`	Offset and range applied after bit reverse.
Offset 5, range 15, byte reverse	`11110111 01111111`	Offset and range applied before byte reversing. Note that right before byte reverse data was padded with 0.

Note: Data manipulation for CSN/UID would give similar effect except that offset and range are defined in bytes (not bits)

3.1.4 Data formatting and output string generation

Card data (PACS or UID/CSN) can be printed using following formats:

1. ASCII
2. Decimal
3. BCD - Binary Coded Decimal
4. Binary
5. Hexadecimal (either uppercase or lowercase)

For every output (except binary without byte reverse) data is padded with zeros from the left side to get full length bytes. The pre stroke string is then added in front of the formatted data, and the post stroke string is added at the end of it.

3.1.5 Character mapping into keystrokes

To generate keyboard strokes from the output string correctly, a character to keystroke mapping process is performed. When creating keyboard wedge output, every byte from pre strokes, post strokes or payload data is mapped into keyboard scan codes. The following steps are used, in the order shown, to get keyboard scan codes:

1. If the character is equal to “0”, reject.
2. If a custom keyboard layout is active, try to find a matching entry in character difference array.
3. If the character is part of the payload data (not pre or post strokes) and the character value is lower than DEC: 32, then dot will be sent (scan code 55).
4. If a character is part of the pre or post strokes, check for a special command character; see *Section 3.1.6 Special command characters*.
5. If the character is part of the pre or post strokes, and its value is in range 11 – 31, then dot will be sent (scan code 55).

6. If the character is extended ASCII (value from 127 to 254), carriage return or line feed, custom code entry will be performed. This behavior depends on the selected operating system; see *Section 3.1.7 Entering extended ASCII characters*.
7. Find keystroke for standard characters (Aa – Zz, 0-9, +-*/.. etc.)

3.1.6 Special command characters

Special command characters are specific byte values reserved for outputting other keyboard functions, such as cursor move, tab, etc.

Byte value	Action	Comment
0x00	Null terminator	
0x01	Enter	
0x02	Cursor down	
0x03	Cursor left	
0x04	Cursor right	
0x05	Cursor up	
0x06	Space	
0x07	Carriage return	Printed as extended ASCII
0x08	Line feed	Printed as extended ASCII
0x09	Tab	
0x0A	Led & buzzer	No keyboard output for this character

3.1.7 Entering extended ASCII characters

Characters in the range 127 to 254, LF (line feed – ASCII 10) and CR (carriage return – ASCII 13) are entered using a combination of keys. The input method depends on the operating system and is as follows:

Windows:

1. Press and hold **Alt**.
2. Enter the ASCII decimal code using the numeric keyboard.
3. Release **Alt**.

Linux:

1. Press and release **Ctrl** + **Shift** + **U**.
2. Enter UTF8 hexadecimal code.
3. Press **Space bar**.

macOS:

1. Press **Option**.
2. Enter UTF8 hexadecimal code.
3. Release **Option**.

Note: To enter a Unicode character using the **Option** key, you must switch the keyboard to Unicode Hex Input.

Section 4

4 Configuration card

The OMNIKEY® 5027 reader allows parts of the configuration to be stored on a DESFire EV1 Smart Card. This allows the configuration to be distributed over many readers, speeding up overall system setup. For configuration card order numbers, please refer to the *Readers and Credentials How to Order Guide* (PLT-2630) on the **HID Global website**.

4.1 Security

The configuration card is secured using DESFire EV1 encryption methods, therefore it cannot be read or modified by third parties. All encryption and decryption operations are managed by a Secure Element chip.

4.2 Key management

HID Global provides a default key in the Secure Element chip, loaded using secure infrastructure. A configuration card custom key can be set using HID OMNIKEY Workbench software, or by sending a proper APDU command as described in *Section 10.1 Configuration card API*.

Note: If the key was changed prior to programming a new configuration card, the same key must be loaded to every reader so that it matches the key on the newly programmed card.

4.3 Configuration download

Downloading a new configuration from the configuration card is a simple process. Presenting any DESFire EV1 smart card will start the validation procedure. If a card is verified as valid, the new configuration will be stored in non-volatile memory. The reader will indicate the programming process using the LED and buzzer according to the following table.

Event	LED	Buzzer
Start	blink	-
Stop - success	blink	~1000 ms
Stop - error	blink	~300 ms (2x)

If the card is removed prematurely, it will result in an error from the reader. If the card had the configuration stored in multiple files, then part of it could be saved in non-volatile memory.

Note: If the configuration card and Secure Element card keys are different, then no error is signaled. The card is detected as regular smart card.

This page is intentionally left blank.

Section 5

5 Contactless card interface

The OMNIKEY® 5027 reader supports sleep mode for low power applications. When in low power mode, the reader is able to detect a HF credential by detecting subtle changes in magnetic field using Low Power Card Detection (LPCD) feature.

5.1 Polling mode

OMNIKEY 5027 supports a single polling mode. This polling mode operates as follows:

1. The reader polls for cards automatically, using a set of configured credentials (see *Section 6.2 Supported credentials*).
2. When a card is found, the reader matches it with a proper configuration.
3. When a match is found, the reader downloads data from the card into internal buffers.
4. The data is processed and formatted according to the description in the configuration.
5. Output is sent over HID (Human Interface Device) to the host machine.
6. On card removal, the cycle is repeated.

5.2 Polling and power consumption

When the reader is in low power mode, it constantly monitors for magnetic field disturbances while using very little power. In sleep mode, the reader will wake up using any supported credential even if it is not configured for data output.

This page is intentionally left blank.

Section 6

6 Contactless credential support

6.1 Contactless protocols

The OMNIKEY® 5027 reader supports retrieving CSN/UID or PACS (for specified credentials) from cards compliant with the following protocols:

- ISO/IEC 14443 Type A
- ISO/IEC 14443 Type B
- ISO/IEC 15693
- ISO 15693-iCLASS
- FeliCa

Individual protocols are retrieved only when there is keyboard wedge configuration for the specific card type. The only exception is protocol ISO 14443 Type A, which is also retrieved when configuration card reading is enabled.

When the host powers up the card, the relevant anti-collision procedure is executed to achieve the selection of a single card.

By default, the card normally is switched to the highest possible speed supported by both the reader and the card. The reader's supported communication speeds for a particular protocol can be configured using appropriate APDU commands.

6.2 Supported credentials

The table below shows supported card types and the matching keyboard wedge configuration.

No.	KBW credential type	UID/CSN	PACS
1	MIFARE Classic	Yes	Yes
2	MIFARE Ultralight	Yes	No
3	MIFARE DESFire (0.6, EV1, EV2)	Yes	Yes (only DESFire 0.6 and DESFire EV1)
4	HID iCLASS® Seos®	Yes	Yes
5	HID iCLASS	Yes	Yes
6	FeliCa	Yes	No
7	ISO/IEC 15693	Yes	No
8	ISO/IEC 14443 Type A - Generic	Yes	No
9	ISO/IEC 14443 Type B	Yes	No

6.3 Physical access control (PACS) data

The OMNIKEY 5027 reader is capable of retrieving PACS data from a card. Data of maximum length 96 bits is processed. If the presented card's data exceeds 96 bits, PACS data will not be printed. You can set a range limit to print part of the card data.

Note: If the card is presented for too short a period, it can cause only UID/CSN to be accessed.

6.4 Dual/combo cards support

6.4.1 Cards with two interfaces using different protocols

If the card has different protocols implemented, you can access one of them by enabling retrieval of this interface and disabling retrieval of the other. The interface is disabled when there is no keyboard wedge configuration for a card type using this protocol. To turn off ISO 14443A, it is also necessary to disable configuration card reading, since this protocol is also used by a MIFARE DESFire EV1 configuration card.

6.4.2 Cards with MIFARE emulation

If the card supports MIFARE emulation, it can be detected by the OMNIKEY 5027 reader as a MIFARE card. This feature has to be enabled for the ISO 14443 Type A protocol using the appropriate APDU described in *Section 9.5.4.1 iso14443aRxTxBaudRate*.

6.4.3 Cards with two interfaces using same protocol

If the card supports two interfaces, but both of them are using same ISO protocol, it is not possible to determine which interface will be used to communicate with the OMNIKEY 5027.

Note: The keyboard wedge output will randomly present one of the embedded contents.

Section 7

7 PC/SC

The OMNIKEY® 5027 reader allows you to access the reader configuration through the framework defined in PC/SC. This simplifies card integration for any developer who is already familiar with this framework.

The Microsoft Developer Network (MSDN) Library contains valuable information and complete documentation of the SCard API within the MSDN Platform SDK. Search for *Smart Card Resource Manager API* on **http://www.microsoft.com**.

7.1 How to access the reader through PC/SC

The following steps provide a guideline to create your first smart card application using industry standard, PC/SC compliant API function calls. The function definitions provided are taken verbatim from the MSDN Library [MSDNLIB]. For additional descriptions of these and other PC/SC functions provided by the Microsoft Windows PC/SC smart card components, search for *The Smart Card Cryptographic Service Provider Cookbook* on **http://www.microsoft.com**.

7.1.1 Step 1: Establish context

This step initializes the PC/SC API and allocates all resources necessary for a smart card session. The `SCardEstablishContext` function establishes the resource manager context (scope) within which database operations are performed.

```
LONG SCardEstablishContext( IN DWORD dwScope,
                            IN LPCVOID pvReserved1,
                            IN LPCVOID pvReserved2,
                            OUT LPSCARDCONTEXT phContext);
```

7.1.2 Step 2: Get status change

Check the status of the reader for card insertion, removal, or availability of the reader. The `SCardGetStatusChange` function blocks execution until the current availability of the cards in a specific set of readers change. The caller supplies a list of monitored readers and the maximum wait time (in milliseconds) for an action to occur on one of the listed readers.

```
LONG SCardGetStatusChange( IN SCARDCONTEXT hContext,
                           IN DWORD dwTimeout,
                           IN OUT LPSCARD_READERSTATE rgReaderStates,
                           IN DWORD cReaders);
```

7.1.3 Step 3: List readers

To acquire a list of all PC/SC readers, use the `SCardListReaders` function. Look for HID Global OMNIKEY Smart Card Reader in the returned list. If multiple Contact Smart Card readers are connected to your system, they will be enumerated.

Example:

HID Global OMNIKEY 5027 Smartcard Reader 0, and HID Global OMNIKEY 5027CL Smartcard Reader 0.

```
LONG SCardListReaders( IN SCARDCONTEXT hContext,
                       IN LPCTSTR mszGroups,
                       OUT LPTSTR mszReaders,
                       IN OUT LPDWORD pcchReaders);
```

7.1.4 Step 4: Connect to the card

The `SCardConnect` function establishes a connection (using a specific resource manager context) between the calling application and a smart card contained by a specific reader. If no card exists in the specified reader, an error is returned.

```
LONG SCardConnect( IN SCARDCONTEXT hContext,
                   IN LPCTSTR szReader,
                   IN DWORD dwShareMode,
                   IN DWORD dwPreferredProtocols,
                   OUT LPSCARDHANDLE phCard,
                   OUT LPDWORD pdwActiveProtocol);
```

7.1.5 Step 5: Exchange data and commands with the card or the reader

You exchange commands and data through APDUs. The `SCardTransmit` function sends a service request to the smart card, expecting to receive data back from the card.

```
LONG SCardTransmit( IN SCARDHANDLE hCard,
                    IN LPCSCARD_I0_REQUEST pioSendPci,
                    IN LPCBYTE pbSendBuffer,
                    IN DWORD cbSendLength,
                    IN OUT LPSCARD_IO_REQUEST pioRecvPci,
                    OUT LPBYTE pbRecvBuffer,
                    IN OUT LPDWORD pcbRecvLength);
```

Note: The application communicates through `SCardControl()` in:

- environments that do not allow `SCardTransmit()` without an ICC
- environments that do not allow `SCardTransmit()` for any other reasons
- when developers prefer the application to communicate through `SCardControl()`.

The application retrieves the control code corresponding to `FEATURE_CCID_ESC_COMMAND` (see part 10, rev.2.02.07). In case this feature is not returned, the application may try `SCARD_CTL_CODE (3500)` as the control code to use.

```
LONG SCardControl( IN SCARDHANDLE hCard,
                   IN DWORD dwControlCode,
                   IN LPCVOID lpInBuffer,
                   IN DWORD nInBufferSize,
                   OUT LPVOID lpOutBuffer,
                   IN DWORD nOutBufferSize,
                   OUT LPDWORD lpBytesReturned);
```

7.1.6 Step 6: Disconnect

It is not necessary to disconnect the card after the completion of all transactions, but it is recommended. The `SCardDisconnect` function terminates a connection previously opened between the calling application and a smart card in the target reader.

```
LONG SCardDisconnect( IN SCARDHANDLE hCard,
                      IN DWORD dwDisposition);
```

7.1.7 Step 7: Release

This step ensures all system resources are released. The `SCardReleaseContext` function closes an established resource manager context, freeing any resources allocated under that context.

```
LONG SCardReleaseContext( IN SCARDCONTEXT hContext);
```

7.2 Vendor-specific commands

The card reader supports features outside the specified commands of PC/SC. To allow applications to control these features, a generic command needs to be used. Use of such a generic command prevents conflicts of reserved INS values used by certain card readers. This command allows applications to control device specific features provided by the reader.

7.2.1 Command APDU

CLA	INS	P1	P2	Lc	Data field	Le
FFh	70h	07h	6Bh	xx	DER TLV coded PDU (Vendor Payload)	xx

The IFD supports the INS Byte 70h for vendor-specific proprietary commands.

P1 and P2 constitute the vendor ID. For OMNIKEY 5027 Smart Card Reader product is the VID = 076Bh.

The data field is constructed as ASN.1 objects/items, whereby every OMNIKEY Smart Card Reader object is identified by a unique Object Identifier (OID).

OIDs are organized as a leaf tree under an invisible root node. The following table shows the first root nodes.

Tag value	Vendor payload branch
A2h (constructed)	readerInformationApi - see *Section 9.3 Reader capabilities*.
A4h (constructed)	contactlessSlotConfiguration - see *Section 9.5 Contactless slot configuration*.
A7h (constructed)	readerEEPROM - see *Section 9.7 Reader EEPROM*.
9Dh (primitive) BDh (constructed)	response
9Eh (primitive)	errorResponse

7.2.2 Response APDU

For all commands encapsulated in the generic 70h APDU, the IFD returns a response frame constructed as follows.

Data field	SW1SW2
DER TLV coded PDU (Vendor Payload)	See ISO 7816-4

The last two bytes of the response frame are always the return code, SW1SW2.

In case of an ISO 7816 violation, the return code is according to ISO 7816-4 and the data field may be empty.

In case of positive processing or internal errors, the IFD returns SW1SW2 = 9000 and the data field is encapsulated in the response tag (9Dh or BDh) or error response tag (9Eh).

The response includes more than one leaf, depending on the request. Each leaf is encapsulated in the leaf tag.

7.2.3 Error response

The error response tag caused by the firmware core is 9Eh (Class Context Specific) + (Primitive) + (1Eh). Length is 2 bytes. The first byte is the cycle in which the error occurred, and the second byte is the exception type.

9E 02 xx yy 90 00

Value	Description
9Eh	Tag = Error Response (0Eh) + (Class Context Specific) + (Primitive)
02h	Len = 2
cycle	Value byte 1: Cycle in which the error is occurred; see error cycle table, below.
error	Value byte 2: Error code; see error code table, below.
SW1	90
SW2	00

Error cycle:

First value byte	
Cycle	**Description**
0	HID Proprietary Command APDU
1	HID Proprietary Response APDU
2	HID Read or Write EEPROM Structure
3	RFU
4	RFU
5	RFU

Error code:

Second value byte		
Exception	**Description**	
3	03h	NOT_SUPPORTED
4	04h	TLV_NOT_FOUND
5	05h	TLV_MALFORMED
6	06h	ISO_EXCEPTION
11	0Bh	PERSISTENT_TRANSACTION_ERROR
12	0Ch	PERSISTENT_WRITE_ERROR
13	0Dh	OUT_OF_PERSISTENT_MEMORY
15	0Fh	PERSISTENT_MEMORY_OBJECT_NOT_FOUND
17	11h	INVALID_STORE_OPERATION
19	13h	TLV_INVALID_SETLENGTH
20	14h	TLV_INSUFFICIENT_BUFFER
21	15h	DATA_OBJECT_READONLY

Second value byte		
Exception	**Description**	
31	1Fh	APPLICATION_EXCEPTION (Destination Node ID mismatch)
42	2Ah	MEDIA_TRANSMIT_EXCEPTION (Destination Node ID mismatch)
43	2Bh	SAM_INSUFFICIENT_MSGHEADER (Secure Channel ID not allowed)
47	2Fh	TLV_INVALID_INDEX
48	30h	SECURITY_STATUS_NOT_SATISFIED
49	31h	TLV_INVALID_VALUE
50	32h	TLV_INVALID_TREE
64	40h	RANDOM_INVALID
65	41h	OBJECT_NOT_FOUND

Section 8

8 Communicating with the reader

The OMNIKEY® 5027 reader does not support direct communication with the card using PC/SC. The reader automatically handles all the communication to get the desired data from the smart card and output it through the keyboard interface. However, the CCID interface is used for sending pseudo APDUs with configuration commands.

8.1 OMNIKEY specific commands

The reader supports features outside the specified commands of PC/SC-3; see *Section 1.1.2 Reference documents*. Vendor-specific proprietary commands allow applications to control the device specific features provided by the reader. Use of such a generic command prevents conflicts of reserved INS values used by certain card readers.

8.1.1 OMNIKEY specific command APDU format

CLA	INS	P1	P2	Lc	Data field	Le
FF	70	07	6B	xx	DER TLV coded PDU (Vendor Payload)	xx

The IFD supports the INS byte 70 for vendor-specific commands. P1 and P2 constitute the vendor ID (VID). For OMNIKEY products, the VID = 0x076B. The data field is constructed as ASN.1 objects/items.

8.1.2 Response for OMNIKEY specific commands

Data field	SW1 SW2
DER TLV Response PDU	See ISO 7816-4

OIDs are organized as a leaf tree under an invisible root node. The following table shows the first root nodes.

8.1.2.1 Vendor payload command types

Vendor payload	Tag value (hex)	Description
readerInformationApi	0x02	Reader information API
response	0x1D	Response
errorResponse	0x1E	Error Response

The following description explains the DER TLV coded data field.

Note: The L field uses the definite form. For the definite form the length octets consist of one or more octets, short form or long form. For the long form, the IFD uses the version with two subsequent octets.

8.1.3 Response APDU

For all commands encapsulated in a generic 70h APDU, the IFD returns a response frame constructed as follows.

Data field	SW1 SW2
DER TLV coded Response PDU	See ISO 7816-4

The last two bytes of the response frame are always the return code, SW1SW2.

In cases of an ISO 7816 violation, the return code is according to ISO 7816-4 and the data field may be empty.

In cases of positive processing or internal errors, the IFD returns SW1SW2 = 9000 and the data field is encapsulated in the response tag (9Dh or BDh) or error response tag (9Eh).

The response includes more than one leaf, depending on the request. Each leaf is encapsulated in the leaf tag.

8.1.4 Error response

The error response tag caused by the firmware core is 9Eh (Class Context Specific) + (Primitive) + (1Eh). Length is 2 bytes. The first byte is the cycle in which the error occurred, and the second byte is the exception type.

9E 02 xx yy 90 00

Value	Description
9Eh	Tag = Error Response (0Eh) + (Class Context Specific) + (Primitive)
02h	Len = 2
cycle	Value byte 1: Cycle in which the error is occurred; see error cycle table below.
error	Value byte 2: Error code. See error code table, below.
SW1	90
SW2	00

Error cycle:

First value byte	
Cycle	**Description**
0	HID Proprietary Command APDU
1	HID Proprietary Response APDU
2	HID Read or Write EEPROM Structure
3	RFU
4	RFU
5	RFU

Error code:

Second value byte		
Exception	**Description**	
3	03h	NOT_SUPPORTED
4	04h	TLV_NOT_FOUND
5	05h	TLV_MALFORMED
6	06h	ISO_EXCEPTION
11	0Bh	PERSISTENT_TRANSACTION_ERROR
12	0Ch	PERSISTENT_WRITE_ERROR
13	0Dh	OUT_OF_PERSISTENT_MEMORY
15	0Fh	PERSISTENT_MEMORY_OBJECT_NOT_FOUND
17	11h	INVALID_STORE_OPERATION
19	13h	TLV_INVALID_SETLENGTH
20	14h	TLV_INSUFFICIENT_BUFFER
21	15h	DATA_OBJECT_READONLY
31	1F	APPLICATION_EXCEPTION (Destination Node ID mismatch)
42	2Ah	MEDIA_TRANSMIT_EXCEPTION (Destination Node ID mismatch)
43	2Bh	SAM_INSUFFICIENT_MSGHEADER (Secure Channel ID not allowed)
47	2Fh	TLV_INVALID_INDEX
48	30h	SECURITY_STATUS_NOT_SATISFIED
49	31h	TLV_INVALID_VALUE
50	32h	TLV_INVALID_TREE
64	40h	RANDOM_INVALID
65	41h	OBJECT_NOT_FOUND

8.1.5 Reader information API

This command group is reserved for GET and SET of reader specific information.

This page is intentionally left blank.

Section 9

9 Reader configuration

All OMNIKEY® 5027 reader configurable items are identified by a unique ASN.1 leaf. A full description is given below, including default values and example APDU commands to get and set.

9.1 APDU commands

If the host implements a PC/SC environment, the OMNIKEY 5027 ASN.1 leafs are accessible using proprietary APDU commands sent through the CCID USB device class. The APDU commands are used to set and get the configuration items, to control the reader, and to apply, store or reset the changes.

APDUs supported by the OMNIKEY 5027 reader fall into the following groups:

- ICAO (International Civil Aviation Organization) test commands as defined in Appendix C of “RF Protocol and Application Test Standard for e-Passport - Part 4”
- OMNIKEY 5027 commands - these include APDUs to manage the reader, to directly access the configuration items.

9.2 Accessing configuration

OMNIKEY specific commands include the reader information API command group, which provides access to reader configuration and provides control of the reader.

9.2.1 Reader information structure

Root	Request	Branch
readerInformationApi (0x02)	Get (0x00)	readerCapabilities (0x00) tlvVersion (0x00) deviceID (0x01) productName (0x02) productPlatform (0x03) enabledCLFeatures (0x04) firmwareVersion (0x05) hfControllerVersion (0x08) hardwareVersion (0x09) hostInterfaces (0x0A) numberOfContactSlots (0x0B) numberOfContactlessSlots (0x0C) numberOfAntennas (0x0D) humanInterfaces (0x0E) vendorName (0x0F) exchangeLevel (0x11) serialNumber (0x12) hfControllerType (0x13) sizeOfUserEEProm (0x14) firmwareLabel (0x16)
	Get (0x00) Set (0x01)	contactlessSlotConfiguration (0x04) contactlessCommon (0x00) sleepModePollingFrequency(0x0D) sleepModeCardDetectionEnable (0x0E) pollingSearchOrder (0x09) emdSuppresionEnable (0x07) configCardEnable (0x11) iso14443aConfig (0x02) iso14443aRxTxBaudRate (0x01) mifareClassicEmulationPreferred (0x04) iso14443bConfig (0x03) iso14443bRxTxBaudRate (0x01) felicaConfig (0x05) felicaRxTxBaudRate (0x01) iClassConfig (0x06) iClass15693Timeout (0x05) iClassActallTimeout (0x06)
	Get (0x00) Set (0x01)	readerEEPROM (0x07) eepromOffset (0x01) eepromRdLength (0x02) eepromWrData (0x03)
	Set (0x01)	readerConfigurationControl (0x09) applySettings (0x00) restoreFactoryDefaults (0x01) rebootDevice (0x03)

Note: After SET requests, apply settings to apply the changes.

9.2.2 Example: Get reader information

With Get Reader Information, the host application gets specific information about the reader. The following example shows how to read a single item.

DER TLV PDU to retrieve single IFD information (productName):

```
A2 06                       // CHOICE ReaderInformationAPI
  A0 04                     // CHOICE GetRequest
    A0 02                   // CHOICE ReaderCapabilities
    82 00                   // SEQUENCE productName
```

The reply of single information is TLV coded:

```
BD 0F 82 0D 4F 4D 4E 49 4B 45 59 20 35 34 32 32 00 90 00 // 'OMNIKEY 5027' + return code
```

For a reader information GET request, the response tag (1D) is always CONSTRUCTED. The response can include more than one leaf, depending on the request.

DER TLV PDU to retrieve single IFD information (productPlatform):

```
A2 06                       // CHOICE ReaderInformationAPI
  A0 04                     // CHOICE GetRequest
    A0 02                   // CHOICE ReaderCapabilites
    83 00                   // SEQUENCE productplatform
```

The reply of single information is TLV coded:

```
BD 0A 83 08 41 56 69 61 74 6F 52 00 90 00          // 'AViatoR' + return code success
```

9.3 Reader capabilities

Reader capabilities structure:

Root	Branch
readerCapabilities (0x00)	tlvVersion (0x00) deviceID (0x01) productName (0x02) productPlatform (0x03) enabledCLFeatures (0x04) firmwareVersion (0x05) hfControllerVersion (0x08) hardwareVersion (0x09) hostInterfaces (0x0A) numberOfContactSlots (0x0B) numberOfContactlessSlots (0x0C) numberOfAntennas (0x0D) humanInterfaces (0x0E) vendorName (0x0F) exchangeLevel (0x11) serialNumber (0x12) hfControllerType (0x13) sizeOfUserEEProm (0x14) firmwareLabel (0x16) configCardsVerSupport (0x1B)

9.3.1 tlvVersion

Tag	0x00
Access	Read-only
Type	INTEGER
Length	1 byte
Value	0x00 - 0xFF
Description	The version of the TLV encoding used by APDUs
Get APDU	FF 70 07 6B 08 A2 06 A0 04 A0 02 80 00 00
Sample response	BD 03 80 **01** 01 90 00

9.3.2 deviceID

Tag	0x01
Access	Read-only
Type	OCTET STRING
Length	2 bytes
Value	0x0000 - 0xFFFF
Description	Product ID
Get APDU	FF 70 07 6B 08 A2 06 A0 04 A0 02 81 00 00
Sample response	BD 04 81 02 **00 08** 90 00

9.3.3 productName

Tag	0x02
Access	Read-only
Type	OCTET STRING
Length	Variable
Value	Null terminated string
Description	The name of the reader
Get APDU	FF 70 07 6B 08 A2 06 A0 04 A0 02 82 00 00
Sample response	BD 0F 82 0D **4F 4D 4E 49 4B 45 59 20 35 30 32 37 00** 90 00

9.3.4 productPlatform

Tag	0x03
Access	Read-only
Type	OCTET STRING
Length	Variable
Value	Null terminated string
Description	The name of the software Platform on which the product is based
Get APDU	FF 70 07 6B 08 A2 06 A0 04 A0 02 83 00 00
Sample response	BD 0A 83 08 **41 56 69 61 74 6F 52 00** 90 00

9.3.5 enabledCLFeatures

Access	Read-only
Type	OCTET STRING
Length	2 bytes
Value	Bit mask, see below
Description	Provides information about the contactless protocols that are supported
Get APDU	FF 70 07 6B 08 A2 06 A0 04 A0 02 84 00 00
Sample response	BD 04 84 02 **0F 99** 90 00

CL features:
0x0001 – FeliCa support
0x0002 – EMVCo support
0x0004 – Calypso support
0x0008 – NFC P2P support
0x0010 – SIO processor available
0x0020 – SDR (LF processor) available
0x0040 – Native FW Secure Engine
0x0080 – T=CL support
0x0100 – ISO 14443 A support
0x0200 – ISO 14443 B support
0x0400 – ISO 15693 support
0x0800 – PicoPass 15693-2 support
0x1000 – PicoPass 14443B-2 support
0x2000 – PicoPass 14443A-3 support
0x4000 – RFU
0x8000 – RFU

9.3.6 firmwareVersion

Tag	0x05
Access	Read-only
Type	OCTET STRING
Length	3 bytes
Value	Null terminated string
Description	The version number of the reader's firmware. 1st byte is major, 2nd byte is minor, 3rd byte is revision number.
Get APDU	FF 70 07 6B 08 A2 06 A0 04 A0 02 85 00 00
Sample response	BD 05 85 03 **01 00 00** 90 00

9.3.7 hfControllerVersion

Tag	0x08
Access	Read-only
Type	OCTET STRING
Length	1 byte
Value	Version number
Description	The version of the HF front-end used for controlling high frequency credentials
Get APDU	FF 70 07 6B 08 A2 06 A0 04 A0 02 88 00 00
Sample response	BD 03 88 01 **1A** 90 00

9.3.8 hardwareVersion

Tag	0x09
Access	Read-only
Type	OCTET STRING
Length	Variable
Value	Null terminated string
Description	The version of the reader hardware used
Get APDU	FF 70 07 6B 08 A2 06 A0 04 A0 02 89 00 00
Sample response	BD 11 89 0F **50 43 42 2D 30 30 31 39 36 20 52 45 56 42 00** 90 00

9.3.9 hostInterfaceFlags

Tag	0x0A
Access	Read-only
Type	OCTET STRING
Length	1 byte
Value	Bit mask, see below
Description	Provides information on the interfaces supported by the reader for communication with the host
Get APDU	FF 70 07 6B 08 A2 06 A0 04 A0 02 8A 00 00
Sample response	BD 03 8A 01 **02** 90 00

Host interface flags:
0x01 – Ethernet available
0x02 – USB available
0x04 – Serial RS232 available
0x08 – SPI available
0x10 – I^2C available

9.3.10 numberOfContactSlots

Tag	0x0B
Access	Read-only
Type	INTEGER
Length	1 byte
Value	Number of contact slots
Description	Number of contact slots supported by the reader
Get APDU	FF 70 07 6B 08 A2 06 A0 04 A0 02 8B 00 00
Sample response	BD 03 8B 01 **00** 90 00

9.3.11 numberOfContactlessSlots

Tag	0x0C
Access	Read-only
Type	INTEGER
Length	1 byte
Value	Number of contactless slots
Description	The number of contactless PCSC slots supported by the reader
Get APDU	FF 70 07 6B 08 A2 06 A0 04 A0 02 8C 00 00
Sample response	BD 03 8C 01 **01** 90 00

9.3.12 numberOfAntennas

Tag	0x0D
Access	Read-only
Type	INTEGER
Length	1 byte
Value	Number of antennas
Description	The number of antennas in the reader
Get APDU	FF 70 07 6B 08 A2 06 A0 04 A0 02 8D 00 00
Sample response	BD 03 8D 01 **01** 90 00

9.3.13 humanInterfaces

Tag	0x0E
Access	Read-only
Type	INTEGER
Length	1 byte
Value	List of tags describing human interfaces
Description	Provides information on the human interfaces supported by the reader
Get APDU	FF 70 07 6B 08 A2 06 A0 04 A0 02 8E 00 00
Sample response	BD 05 8E 03 80 01 **00** 90 00

9.3.14 vendorName

Tag	0x0F
Access	Read-only
Type	OCTET STRING
Length	Variable
Value	Null terminated string
Description	The vendor of the reader
Get APDU	FF 70 07 6B 08 A2 06 A0 04 A0 02 8F 00 00
Sample response	BD 0D 8F 0B **48 49 44 20 47 6C 6F 62 61 6C 00** 90 00

9.3.15 exchangeLevel

Tag	0x11
Access	Read-only
Type	OCTET STRING
Length	1 byte
Value	Bit mask, see below
Description	Provides information about the different APDU levels supported by the reader
Get APDU	FF 70 07 6B 08 A2 06 A0 04 A0 02 91 00 00
Sample response	BD 03 91 01 **02** 90 00

Exchange level flags:
0x01 – TPDU
0x02 – APDU
0x04 – Extended APDU

9.3.16 serialNumber

Tag	0x12
Access	Read-only
Type	OCTET STRING
Length	Variable, max 32 bytes
Value	Serial number
Description	The serial number of the reader
Get APDU	FF 70 07 6B 08 A2 06 A0 04 A0 02 92 00 00
Sample response	BD 19 92 17 **4B 54 2D 30 38 36 33 30 30 33 30 2D 31 36 31 30 2D 30 30 30 31 31 34** 90 00

9.3.17 hfControllerType

Tag	0x13
Access	Read-only
Type	OCTET STRING
Length	Variable, max 32 bytes
Value	Null terminated chip name
Description	The IC used for control of HF credentials
Get APDU	FF 70 07 6B 08 A2 06 A0 04 A0 02 93 00 00
Sample response	BD 08 93 06 **52 43 36 36 33 00** 90 00

9.3.18 sizeOfUserEEPROM

Tag	0x14
Access	Read-only
Type	OCTET STRING
Length	2 bytes
Value	Size in bytes
Description	The amount of user EEPROM memory available
Get APDU	FF 70 07 6B 08 A2 06 A0 04 A0 02 94 00 00
Sample response	BD 04 94 02 **04 00** 90 00

9.3.19 firmwareLabel

Tag	0x16
Access	Read-only
Type	OCTET STRING
Length	Variable
Value	Firmware unique ID as string
Description	Detailed information about the firmware version
Get APDU	FF 70 07 6B 08 A2 06 A0 04 A0 02 96 00 00
Sample response	BD 35 96 3E **4F 4B 35 30 32 37 2D 31 2E 30 2E 30 2E 32 34 34 2D 32 30 31 38 30 32 30 35 54 30 36 34 33 30 37 2D 41 44 33 34 46 42 46 34 37 35 30 44 2D 46 4C 41 53 48 5F 50 52 4F 44 55 43 54 49 4F 4E** 90 00

9.3.20 configCardVerSupport

Tag	0x1B
Access	Read-only
Type	INTEGER
Length	2 bytes
Value	Version in bytes
Description	The amount of user EEPROM memory available
Get APDU	FF 70 07 6B 08 A2 06 A0 04 A0 02 9B 00 00
Sample response	BD 04 9B 02 **01 00** 90 00

9.4 Keyboard wedge configuration

OMNIKEY 5027 offers three keyboard wedge configuration slots, plus additional settings regarding keyboard language and extended characters support.

Root	Branch	
contactlessSlotConfiguration (0x04)	KBWConfiguration1 (0x08)	KBWCardType (0x00) KBWOutputFormat (0x01) KBWFlags (0x02) KBWRangeStart (0x03) KBWRangeLen (0x04) KBWPostStrokeStart (0x05) KBWPrePostStrokes (0x06)
	KBWConfiguration2 (0x09)	KBWCardType (0x00) KBWOutpuFormat (0x01) KBWFlags (0x02) KBWRangeStart (0x03) KBWRangeLen (0x04) KBWPostStrokeStart (0x05) KBWPrePostStrokes (0x06)
	KBWConfiguration3 (0x0A)	KBWCardType (0x00) KBWOutpuFormat (0x01) KBWFlags (0x02) KBWRangeStart (0x03) KBWRangeLen (0x04) KBWPostStrokeStart (0x05) KBWPrePostStrokes (0x06)
	KeyboardCountryDefinition (0x0B)	UseSecondKeyboardLayout (0x00) CharactersDiff (0x01)
	ExtendedCharacterSupport (0xC)	OSforExtendedChars (0x00)

9.4.1 Keyboard wedge output configuration

9.4.1.1 KBWCardType

Tag	0x00
Access	Read/Write
Type	INTEGER
Length	1 byte
Value	yy – configuration slot, xx – selected card type, see below
Description	Card type selected for data output
Set APDU	FF 70 07 6B 0B A2 09 A1 07 A4 05 **yy** 03 80 01 **xx** 00
Sample response	BD 00 90 00
Get APDU	FF 70 07 6B 0A A2 08 A0 06 A4 04 **yy** 02 80 00 00
Sample response	BD 03 80 01 **xx** 90 00

Configuration slot:
0xA8 – Slot 1
0xA9 – Slot 2
0xAA – Slot 3

Supported card types:
0x00 – Slot not used
0x01 – MIFARE Classic
0x02 – MIFARE Ultralight, NFC Tag Type 2
0x03 – MIFARE DESFire, NFC Tag Type 4
0x04 – HID iCLASS® Seos®
0x05 – HID iCLASS
0x06 – Sony FeliCa, NFC Tag Type 3
0x08 – ISO15693, NFC Tag Type 5
0x09 – Generic ISO14443B
0x0A – Generic ISO14443A

9.4.1.2 KBWOutputFormat

Tag	0x01
Access	Read/Write
Type	INTEGER
Length	1 byte
Value	yy – configuration slot, xx - output format, see below
Description	The output format for the selected configuration slot
Set APDU	FF 70 07 6B 0B A2 09 A1 07 A4 05 **yy** 03 81 01 **xx** 00
Sample response	BD 00 90 00
Get APDU	FF 70 07 6B 0A A2 08 A0 06 A4 04 **yy** 02 81 00 00
Sample response	BD 03 81 01 **xx** 90 00

Configuration slot:
0xA8 – Slot 1
0xA9 – Slot 2
0xAA – Slot 3

Output formats:
0x00 – ASCII
0x01 – BCD
0x02 – Binary
0x03 – Hexadecimal lowercase
0x04 – Decimal
0x05 – Hexadecimal uppercase

9.4.1.3 KBWFlags

Tag	0x02
Access	Read/Write
Type	INTEGER
Length	1 byte
Value	yy – configuration slot, xx - Keyboard wedge flags, see below
Description	Keyboard wedge output flags
Set APDU	FF 70 07 6B 0B A2 09 A1 07 A4 05 **yy** 03 82 01 **xx** 00
Sample response	BD 00 90 00
Get APDU	FF 70 07 6B 0A A2 08 A0 06 A4 04 **yy** 02 82 00 00
Sample response	BD 03 82 01 **xx** 90 00

Configuration slot:
0xA8 – Slot 1
0xA9 – Slot 2
0xAA – Slot 3

Keyboard wedge flags:
b1 = 0 defines UID/CSN output
b1 = 1 defines PACS output
b2 = 1 defines data bit reverse
b3 = 1 defines data byte reverse

Note: Bit and byte reverse are mutually exclusive, so cannot be set at the same time. If you try to set bit and byte reverse at the same time, the reader will respond with an invalid TLV error code exception: 9E 02 00 31 90 00

b8	b7	b6	b5	b4	b3	b2	b1
RFU					Byte reverse	Bit reverse	PACS/UID

9.4.1.4 KBWRangeStart

Tag	0x03
Access	Read/Write
Type	INTEGER
Length	1 byte
Value	yy – configuration slot, xx - Keyboard wedge data offset, see below
Description	Data offset of data retrieved from card
Set APDU	FF 70 07 6B 0B A2 09 A1 07 A4 05 **yy** 03 83 01 **xx** 00
Sample response	BD 00 90 00
Get APDU	FF 70 07 6B 0A A2 08 A0 06 A4 04 **yy** 02 83 00 00
Sample response	BD 03 83 01 **xx** 90 00

Configuration slot:
0xA8 – Slot 1
0xA9 – Slot 2
0xAA – Slot 3

Data offset:
If PACS data is selected for output, the offset value is the number of bits.
If UID/CSN data is selected for output, the offset value is the number of bytes.

9.4.1.5 KBWRangeLen

Tag	0x04
Access	Read/Write
Type	INTEGER
Length	1 byte
Value	yy – configuration slot, xx - Keyboard wedge data length, see below
Description	PACS/UID/CSN data length
Set APDU	FF 70 07 6B 0B A2 09 A1 07 A4 05 **yy** 03 84 01 **xx** 00
Sample response	BD 00 90 00
Get APDU	FF 70 07 6B 0A A2 08 A0 06 A4 04 **yy** 02 84 00 00
Sample response	BD 03 84 01 **xx** 90 00

Configuration slot:
0xA8 – Slot 1
0xA9 – Slot 2
0xAA – Slot 3

Data length:
If PACS data is selected for output, the length is the number of bits selected for output.
If UID/CSN data is selected for output, the length is the number of bytes selected for output.

9.4.1.6 KBWPostStrokeStart

Tag	0x05
Access	Read/Write
Type	INTEGER
Length	1 byte
Value	yy – configuration slot, xx - Position in KBWPrePostStrokes string where post strokes start
Description	Post strokes start in pre/post strokes string
Set APDU	FF 70 07 6B 0B A2 09 A1 07 A4 05 **yy** 03 85 01 **xx** 00
Sample response	BD 00 90 00
Get APDU	FF 70 07 6B 0A A2 08 A0 06 A4 04 **yy** 02 85 00 00
Sample response	BD 03 85 01 **xx** 90 00

Configuration slot:
0xA8 – Slot 1
0xA9 – Slot 2
0xAA – Slot 3

KBWPostStrokeStart:
Position in KBWPrePostStrokes string of post strokes start.

9.4.1.7 KBWPrePostStrokes

Tag	0x06
Access	Read/Write
Type	INTEGER
Length	32 bytes
Value	yy – configuration slot, xx - Pre/post strokes string
Description	The string containing pre and post strokes for keyboard wedge output
Set APDU	FF 70 07 6B 2A A2 28 A1 26 A4 24 **yy** 22 86 20 **xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx** 00
Sample response	BD 00 90 00
Get APDU	FF 70 07 6B 0A A2 08 A0 06 A4 04 AA 02 86 00 00
Sample response	BD 22 86 20 **xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx** 90 00

Configuration slot:
0xA8 – Slot 1
0xA9 – Slot 2
0xAA – Slot 3

Pre/post strokes string:
String with pre and post strokes desired for selected keyboard wedge configuration slot.

9.4.2 Keyboard country definition

9.4.2.1 UseSecondKeyboardLayout

Tag	0x00
Access	Read/Write
Type	INTEGER
Length	1 byte
Value	xx – use second keyboard layout
Description	Enables or disables using second keyboard layout defined in tag 0x01
Set APDU	FF 70 07 6B 0B A2 09 A1 07 A4 05 AC 03 80 01 **xx** 00
Sample response	BD 00 90 00
Get APDU	FF 70 07 6B 0A A2 08 A0 06 A4 04 AC 02 80 00 00
Sample response	BD 03 84 01 **xx** 90 00

9.4.2.2 charactersDiff

Tag	0x01
Access	Read/Write
Type	INTEGER
Length	192 bytes
Value	xx – character differences array
Description	Enables or disables using second keyboard layout defined in tag 0x01
Set APDU	FF 70 07 6B CF A2 81 CA A1 81 C8 A4 81 C6 AB 81 C3 81 81 C0 **xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx** **xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx** **xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx** **xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx** **xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx** **xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx** **xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx** **xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx** **xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx** **xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx** **xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx** **xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx** 00
Sample response	BD 00 90 00
Get APDU	FF 70 07 6B 0A A2 08 A0 06 A4 04 AB 02 81 00 00
Sample response	BD 82 00 C4 81 82 00 C0 **xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx** **xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx** **xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx** **xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx** **xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx** **xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx** **xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx** **xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx** **xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx** **xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx** **xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx** **xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx** 90 00

Character differences array:
63 structures consisting of three bytes, encoding ASCII character, modifier key and HID Keyboard Value plus three zero bytes terminating the structure. Available key modifiers (can be combined using bitwise alternative):

0x00 None

0x01 Left Control

0x02 Left Shift

0x04 Left Alt

0x08 Left GUI

0x10 Right Control

0x20 Right Shift

0x40 Right Alt

0x80 Right GUI

Example:
To output the “Z” character properly on a German keyboard, the array should contain the following structure 0x5A021C and unused spaces should be filled with zeroes:

Mapped character	Modifier key	Keyboard scan code
0x5A ("Z")	0x02 (Left Shift)	0x1C

Note: Since version 1.7, OMNIKEY Workbench offers out-of-box arrays containing character differences for German, French and UK keyboard layouts. You can also import custom layouts created in Microsoft Keyboard Layout Creator. See *OMNIKEY 5027 User Guide* (PLT-03827).

9.4.3 Extended ASCII character support

9.4.3.1 OSforExtendedChars

Tag	0x00
Access	Read/Write
Type	INTEGER
Length	1 byte
Value	xx – Operating system, see below
Description	Selects operating system for extended ASCII character support
Set APDU	FF 70 07 6B 0B A2 09 A1 07 A4 05 AC 03 80 01 **xx** 00
Sample response	BD 00 90 00
Get APDU	FF 70 07 6B 0A A2 08 A0 06 A4 04 AC 02 80 00 00
Sample response	BD 03 84 01 **xx** 90 00

Operating system:
0x00 – Windows
0x01 – Linux
0x02 - macOS

9.5 Contactless slot configuration

9.5.1 Contactless slot configuration structure

Root	Branch	
contactlessSlotConfiguration (0x04)	contactlessCommon (0x00)	sleepModePollingFrequency(0x0D) sleepModeCardDetectionEnable (0x0E) mdSuppresionEnable (0x07) configCardsEnable (0x11)
	iso14443aConfig (0x02)	iso14443aRxTxBaudRate (0x01) mifarePreferred (0x04)
	iso14443bConfig (0x03)	iso14443bRxTxBaudRate (0x01)
	FeliCa (0x05)	felicaRxTxBaudRate (0x01)
	iClassConfig (0x06)	iClass15693DelayTime (0x04) iClass15693Timeout (0x05) iClassActallTimeout (0x06)

9.5.2 Baud rates

OMNIKEY 5027 allows setting of the maximum baud rate to and from a card for ISO/IEC 14443 Type A and ISO/IEC 14443 Type B protocols.

Commands `iso14443aRxTxBaudRate`, `iso14443bRxTxBaudRate` and `felicaRxTxBaudRate` use the same format. A one byte argument defines separate baud rates for receiving (Rx) and transmitting (Tx) data. The first 4 bits are used to set Rx baud rate, the other for Tx baud rate. The resulting value is a combination of bits:

Bit 0 (0x01) – 212 kbps
Bit 1 (0x02) – 424 kbps
Bit 2 (0x04) – 848 kbps

The reader always supports 106 kbps regardless of bit settings. If a card does not support a specific transmission speed, the reader will use the other value.

For example, 0x77 means the reader supports 106, 212, 424, 848 kbps for Rx and Tx. If a card supports only 106 kbps and 424 kbps, the reader will use 424 kbps or 106 kbps (in case card activation at 424 kbps fails).

Please note that doubling the baud rate does not double the transmission speed. In an extreme example, changing the baud rate from 424 kbps to 848 kbps increases transmission speed by less than 10%. The number may vary depending on the amount of data transmitted. The worst ratio is for short packets.

Note: Increasing the maximum baud rate may cause transmission problems and shorten maximum effective distance between a card and the reader.

9.5.2.1 Examples

0x00 – 106 kbps for Rx and Tx
0x23 – 106 and 424 kbps for Rx and 106, 212, 424 kbps for Tx
0x71 – 106, 212, 424, 848 kbps for Rx and 106, 212 kbps for Tx

9.5.2.2 Default values

ISO/IEC 14443 Type A: 0x77 – 106, 212, 424, 848 kbps for Rx and Tx
ISO/IEC 14443 Type B: 0x77 – 106, 212, 424, 848 kbps for Rx and Tx
FeliCa: 0x11 – 106, 212 kbps for Rx and Tx

9.5.3 Common parameters

9.5.3.1 sleepModePollingFrequency

Tag	0x0D
Access	Read/Write
Type	INTEGER
Length	1 byte
Value	Frequency index, see below
Description	The frequency of card insertion check in sleep mode
Set APDU	FF 70 07 6B 0B A2 09 A1 07 A4 05 A0 03 8D 01 **xx** 00
Sample response	BD 00 90 00
Get APDU	FF 70 07 6B 0A A2 08 A0 06 A4 04 A0 02 8D 00 00
Sample response	BD 03 8D 01 **xx** 90 00

Frequency index:
0x00 – 41 Hz (24 ms)
0x01 – 20 Hz (48 ms)
0x02 – 10 Hz (96 ms)
0x03 – 5 Hz (0.2 s)
0x04 – 2.5 Hz (0.4 s)
0x05 – 1.3 Hz (0.8 s)
0x06 – 0.7 Hz (1.4 s)
0x07 – 0.3 Hz (3.1 s)
0x08 – 0.15 Hz (6.2 s)
0x09 – 0.08 Hz (12.3 s)

9.5.3.2 sleepModeCardDetectionType

Tag	0x0E
Access	Read/Write
Type	INTEGER
Length	1 byte
Value	0x02 LPCD, 0x01 polling, 0x00 disable
Description	Enable and choose or disable card detection in sleep mode
Set APDU	FF 70 07 6B 0B A2 09 A1 07 A4 05 A0 03 8E 01 **xx** 00
Sample response	BD 00 90 00
Get APDU	FF 70 07 6B 0A A2 08 A0 06 A4 04 A0 02 8E 00 00
Sample response	BD 03 8E 01 **xx** 90 00

9.5.3.3 emdSupressionEnabled

Tag	0x07
Access	Read/Write
Type	INTEGER
Length	1 byte
Value	0x01 enable, 0x00 disable
Description	Enables or disables EMD suppression
Set APDU	FF 70 07 6B 0B A2 09 A1 07 A4 05 A0 03 87 01 **xx** 00
Sample response	BD 00 90 00
Get APDU	FF 70 07 6B 0A A2 08 A0 06 A4 04 A0 02 87 00 00
Sample response	BD 03 87 01 **xx** 90 00

9.5.3.4 configCardEnable

Tag	0x07
Access	Read/Write
Type	INTEGER
Length	1 byte
Value	0x01 enable, 0x00 disable
Description	Enables or disables configuration card support
Set APDU	FF 70 07 6B 0B A2 09 A1 07 A4 05 A0 03 91 01 **xx** 00
Sample response	BD 00 90 00
Get APDU	FF 70 07 6B 0A A2 08 A0 06 A4 04 A0 02 91 00 00
Sample response	BD 03 91 01 **xx** 90 00

9.5.4 ISO/IEC 14443 Type A

9.5.4.1 iso14443aRxTxBaudRate

Tag	0x01
Access	Read/Write
Type	INTEGER
Length	1 byte
Value	See *Section 9.5.2 Baud rates*
Description	Sets supported baud rates for ISO/IEC 14443 Type A cards
Set APDU	FF 70 07 6B 0B A2 09 A1 07 A4 05 A2 03 81 01 **xx** 00
Sample response	BD 00 90 00
Get APDU	FF 70 07 6B 0A A2 08 A0 06 A4 04 A2 02 81 00 00
Sample response	BD 03 81 01 **xx** 90 00

9.5.4.2 mifareClassicEmulationPreferred

Tag	0x04
Access	Read/Write
Type	INTEGER
Length	1 byte
Value	0x01 enable, 0x00 disable
Description	Enables or disables emulated MIFARE Classic preferred mode
Set APDU	FF 70 07 6B 0B A2 09 A1 07 A4 05 A2 03 84 01 **xx** 00
Sample response	BD 00 90 00
Get APDU	FF 70 07 6B 0A A2 08 A0 06 A4 04 A2 02 84 00 00
Sample response	BD 03 84 01 **xx** 90 00

9.5.5 ISO/IEC 14443 Type B

9.5.5.1 iso14443bRxTxBaudRate

Tag	0x01
Access	Read/Write
Type	INTEGER
Length	1 byte
Value	See *Section 9.5.2 Baud rates*
Description	Sets supported baud rates for ISO/IEC 14443 Type B cards
Set APDU	FF 70 07 6B 0B A2 09 A1 07 A4 05 A3 03 81 01 **xx** 00
Sample response	BD 00 90 00
Get APDU	FF 70 07 6B 0A A2 08 A0 06 A4 04 A3 02 81 00 00
Sample response	BD 03 81 01 **xx** 90 00

9.5.6 FeliCa

9.5.6.1 felicaRxTxBaudRate

Tag	0x00
Access	Read/Write
Type	INTEGER
Length	1 byte
Value	Sets supported baud rates for FeliCa cards
Description	See *Section 9.5.2 Baud rates*
Set APDU	FF 70 07 6B 0B A2 09 A1 07 A4 05 A5 03 81 01 **xx** 00
Sample response	BD 00 90 00
Get APDU	FF 70 07 6B 0A A2 08 A0 06 A4 04 A5 02 81 00 00
Sample response	BD 03 81 01 **xx** 90 00

9.5.7 iCLASS

9.5.7.1 iClass15693DelayTime

Tag	0x04
Access	Read/Write
Type	INTEGER
Length	4 bytes
Value	Timeout
Description	Sets or gets minimum chip response to reader command delay
Set APDU	70 07 6B 0E A2 0C A1 0A A4 08 A6 06 84 04 **xx xx xx xx** 00
Sample response	BD 00 90 00
Get APDU	FF 70 07 6B 0A A2 08 A0 06 A4 04 A6 02 84 00 00
Sample response	BD 06 84 04 **xx xx xx xx** 90 00

9.5.7.2 iClass15693Timeout

Tag	0x05
Access	Read/Write
Type	INTEGER
Length	4 bytes
Value	Timeout
Description	Sets or gets time to wait for response to a command
Set APDU	70 07 6B 0E A2 0C A1 0A A4 08 A6 06 85 04 **xx xx xx xx** 00
Sample response	BD 00 90 00
Get APDU	FF 70 07 6B 0A A2 08 A0 06 A4 04 A6 02 85 00 00
Sample response	BD 06 85 04 **xx xx xx xx** 90 00

9.5.7.3 iClassActallTimeout

Tag	0x06
Access	Read/Write
Type	INTEGER
Length	4 bytes
Value	Timeout
Description	Sets or gets time to wait for response to ACT/ACTALL
Set APDU	70 07 6B 0E A2 0C A1 0A A4 08 A6 06 86 04 **xx xx xx xx** 00
Sample response	BD 00 90 00
Get APDU	FF 70 07 6B 0A A2 08 A0 06 A4 04 A6 02 86 00 00
Sample response	BD 06 86 04 **xx xx xx xx** 90 00

9.6 Hardware configuration

9.6.1 LED

9.6.1.1 defaultLEDstate

Tag	0x01
Access	Read/Write
Type	INTEGER
Length	1 bytes
Value	Default LED state in idle
Description	Sets or gets default LED idle state
Set APDU	FF 70 07 6B 0B A2 09 A1 07 AF 05 A0 03 81 01 **xx** 00
Sample response	BD 00 90 00
Get APDU	FF 70 07 6B 0A A2 08 A0 06 AF 04 A0 02 81 00 00
Sample response	BD 01 81 01 **xx** 90 00

9.7 Reader EEPROM

OMNIKEY 5027 provides a user available area (1024 bytes) in internal EEPROM memory. The content of this memory is preserved even the power is off.

When specifying command to read or write data offset must be specified (Tag 0x01; 2 bytes).

Root	Branch
readerEEPROM (0x07)	eepromOffset (0x01) eepromRdLength (0x02) eepromWrData (0x03)

9.7.1 Read

Tag	0x02
Access	Read-only
Type	OCTET STRING
Length	variable
Value	yy yy – address (0x0000–0x03FF) ss – number of bytes to read xx – read out data
Description	The version of the TLV encoding used by APDUs
Get APDU	FF 70 07 6B 0D A2 0B A0 09 A7 07 81 02 **yy yy** 82 01 **ss** 00
Sample response	9D **ss xx** ... 90 00

9.7.2 Write

Tag	0x03
Access	Write-only
Type	OCTET STRING
Length	variable
Value	yy yy – address (0x0000–0x03FF) ss – number of bytes to write (0x01–0xEF) xx – data to write
Description	Writes data to user EEPROM area.
Sample Set APDU	FF 70 07 6B **0C+ss** A2 **0A+ss** A1 **08+ss** A7 **06+ss** 81 02 **yy yy** 83 **ss xx** ... 00 Write 1 byte: FF 70 07 6B **0D** A2 **0B** A1 **09** A7 **07** 81 02 **yy yy** 83 01 **xx** 00 Write 16 bytes: FF 70 07 6B **1C** A2 **1A** A1 **18** A7 **16** 81 02 **yy yy** 83 10 **xx** ... 00 Write 64 bytes: FF 70 07 6B **4C** A2 **4A** A1 **48** A7 **46** 81 02 **yy yy** 83 10 **xx** ... 00 Write 115 bytes: FF 70 07 6B **7F** A2 **7D** A1 **7B** A7 **79** 81 02 **yy yy** 83 10 **xx** ... 00 For number of bytes to write (ss value) bigger than 115 bytes, the tag's length is coded with two bytes according to DER TLV coding: FF 70 07 6B **10+ss** A2 **81 0D+ss** A1 **81 0A+ss** A7 **07+ss** 81 02 **yy yy** 83 81 **ss xx** ... 00 Write 116 bytes: FF 70 07 6B **84** A2 **81 81** A1 **81 7E** A7 **81 7B** 81 02 **yy yy** 83 81 74 **xx** ... 00 Write 239 bytes: FF 70 07 6B **FF** A2 **81 FC** A1 **81 F9** A7 **81 F6** 81 02 **yy yy** 83 81 EF **xx** ... 00
Sample response	9D 00 90 00

9.8 Reader configuration control

9.8.1 applySettings

Tag	0x00
Access	Write-only
Type	
Length	
Value	None
Description	Apply settings. This command must be used to accept changes in the reader configuration. The only settings that take changes immediately are `iso14443aRxTxBaudRate` and `iso14443bRxTxBaudRate`. The command resets the device.
Set APDU	FF 70 07 6B 08 A2 06 A1 04 A9 02 80 00 00
Sample response	9D 00 90 00

9.8.2 restoreFactoryDefaults

Tag	0x01
Access	Write-only
Type	
Length	
Value	None
Description	Sets reader configuration to factory defaults. The command resets the device.
Set APDU	FF 70 07 6B 08 A2 06 A1 04 A9 02 81 00 00
Sample response	9D 00 90 00

9.8.3 rebootDevice

Tag	0x03
Access	Write-only
Type	
Length	
Value	None
Description	Reboots the reader
Set APDU	FF 70 07 6B 08 A2 06 A1 04 A9 02 83 00 00
Sample response	9D 00 90 00

Section 10

10 Device specific commands

This section contains a list of device specific commands applicable only to the OMNIKEY 5027.

Device specific command structure:

Root	Branch
deviceSpecific (0x1C)	configCardAPI (0x01) configCardPrepare (0x00) configCardWrite (0x01) configCardLoadKey (0x05)

10.1 Configuration card API

The configuration card programming interface allows the storing of a reader configuration on a Smart Card, allowing it to be easily distributed over many devices. The configuration can be stored only on a DESFire EV1 card. The stored data is protected against unauthorized access and modification by using a default configuration card key preloaded by HID in each OMNIKEY 5027 reader. Optionally, the key can be changed using either HID OMNIKEY Workbench or the `configCardLoadKey` APDU command.

10.1.1 Configuration card data structure

Before writing any configuration data, the card must be formatted by invoking the `configCardPrepare` command. The whole configuration is stored in a key-secured DESFire EV1 application which contains two types of data files: header file and data file.

A header file always has an ID = 0 and describes the general layout of configuration data.

Bytes	Value	Comment	Section
2	0x50, 0x27	Product identification, fixed value	HEADER
2	0x06, 0x00	Length (DATA section + CRC), fixed value	
1	0xXX	Configuration card API major version	DATA
1	0xXX	Configuration card API minor version	
2	RFU		
2	CRC	CRC X.25 of entire DATA section	CRC

Files with IDs greater than 0 are data files. Each data file must contain a set of ASN.1 commands, as described in *Section 9 Reader configuration*. If your configuration exceeds the size of a single file, then some ASN.1 strings must be moved to the next file. The data file internal structure is defined below:

Bytes	Value	Comment	Section
2	0x50, 0x27	Product identification, fixed value	HEADER
2	0xXX, 0xXX	Length (data section + CRC), LSB	
Data Length	...	Configuration payload	DATA
2	CRC	CRC X.25 of entire DATA section	

10.1.2 configCardPrepare

Tag	0x00
Access	Write-only
Type	INTEGER
Length	1 byte
Value	0x01 - 0x1E
Description	xx - number of files to be created
Sample APDU	FF70076B 09 BC 07 A1 05 A0 03 80 10 **xx** Prepare configuration card with 5 data files: FF70076B 09 BC 07 A1 05 A0 03 80 10 **05**
Sample response	Success: 90 00 Error: 69 01

This command prepares the configuration card with a sequence of commands to comply with the desired data structure. The number of files must be specified, but this does not include the header file. The size of each file is predefined and equal to 237 bytes. During the preparation process, all existing files on the card are lost.

10.1.3 configCardWrite

Tag	0x01
Access	Write-only
Type	OCTET STRING
Length	variable
Value	0x01 - 0x1E
Description	xx – destination file number yyyy – destination file offset (MSB) ss – bytes to write (0x01 – 0x80)
Sample APDU	FF70076B **0F+ss** BC 0D+ss A1 **0B+ss** A1 **09+ss** 82 01 **xx** 83 02 **yy yy** 84 **ss** … 00 Write 1 byte to file number 01 at offset 0: FF70076B 10 BC 0E A1 0C A1 0A 82 01 **01** 83 02 **00 00** 84 **01** AA 00 Write sample configuration that enables LED state: a) Write header file number 0 FF70076B19BC17A115A113820100830200008 40A5027060001000000E065 b) Write data file number 1 FF70076B20BC1EA11CA11A 8201018302000084 1150270D00A209A107AF05A0038101010AF1
Sample response	Success: 90 00 Error: 69 01

10.1.4 configCardLoadKey

Tag:	0x05
Access:	Write-only
Type:	OCTET STRING
Length:	16 bytes
Value:	0x00 - 0xFF
Description:	xx – key value byte
Sample APDU	FF70076B 18 BC 16 A1 14 A5 12 80 10 **xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx**
Sample response	Success: 90 00 Error: 69 01

All OMNIKEY 5027 readers are preloaded with a default key to encrypt and decrypt a configuration card. This feature secures against reading and unauthorized modification by 3rd parties. However, if more security is required, you can upload your own custom key. The configuration card key is set on the card during the preparation phase, therefore it must be loaded before the programming procedure and to all readers on which the card will be read. This command should be sent using the PC_to_RDR_Escape CCID message; see *Section 12 Using the PC_to_RDR_Escape command*.

This page is intentionally left blank.

Section 11

11 ICAO test commands

11.1 Command set

The International Civil Aviation Organization (ICAO) has defined a set of APDUs for testing e-Passport readers. These are defined in Annex C of the technical report *RF Protocol and Application Test Standard for e-Passport - Part 4*, available from the ICAO website **www.icao.int**. The standard APDU syntax and standard SCardTransmit API are used with the reserved value FF for the CLA byte; the values of the INS byte are also reserved (in the range of 0x9x). The commands supported by this reader are as follows:

11.1.1 ICAO commands

Instruction	Description	Comments
0x92	ISO/IEC 14443-2	Fully supported
0x94	Transmit Pattern	Partially supported
0x96	ISO/IEC 14443-3	Partially supported
0x98	ISO/IEC 14443-4	Not supported
0x9A	Miscellaneous	Partially supported

All of the ICAO test commands are attempted regardless of card presence or type.

11.1.2 0x92 - ISO/IEC 14443-2: ISO/IEC 14443-2 command APDU

CLA	INS	P1	P2	Lc	Data In	Le
0xFF	0x92	XX	RFU	XX	Lc bytes	-

Note: Any data received back from the card is ignored in this test.

11.1.2.1 ISO/IEC 14443-2 P1 coding

b7	b6	b5	b4	b3	b2	b1	b0	Description
0	0	------						Turn off RF field
0	1	------						Turn on RF field with no sub-carrier
1	0	------						Turn on RF field and transmit Lc bytes
1	1	------						RFU
--		RFU	0	----				ISO/IEC 14443 Type A transmission
--		RFU	1	----				ISO/IEC 14443 Type B transmission
----				RFU		0	0	106 kbps
----						0	1	212 kbps
----						1	0	424 kbps
----						1	1	848 kbps

11.1.2.2 ISO/IEC 14443-2 response

Data out	SW1SW2	
-	0x9000	Operation successful
	0x6700	Wrong length (e.g. Lc absent when P1 b7 = 1)
	0x6401	Internal error (e.g. protocol setup failed)

11.1.3 0x94 - transmit pattern command APDU

CLA	INS	P1	P2	Lc	Data In	Le
0xFF	0x94	XX	XX	XX	Lc bytes	XX

Note: This test can be used to transmit and/or receive data to/from the card. No parity bit or CRC bytes are added, but framing (that is, start/stop bits, SOF/EOF) *will* be added. This is *not* fully compliant with the ICAO test standard.

11.1.3.1 ICAO transmit pattern P1 coding

b7	b6	b5	b4	b3	b2	b1	b0	Description
RFU				---			0	ISO/IEC 14443 Type A transmission
				---			1	ISO/IEC 14443 Type B transmission
				xxx			-	No of bits in last byte to be transmitted

11.1.3.2 ICAO transmit pattern P2 coding

b7	b6	b5	b4	b3	b2	b1	b0	Description
RFU		---			RFU	0	0	Tx - 106 kbps
		---				0	1	Tx - 212 kbps
		---				1	0	Tx - 424 kbps
		---				1	1	Tx - 848 kbps
		RFU	0	0	---			Rx - 106 kbps
			0	1	---			Rx - 212 kbps
			1	0	---			Rx - 424 kbps
			1	1	---			Rx - 848 kbps

11.1.3.3 ICAO transmit pattern SW1SW2 response bytes

Data out	SW1SW2	
XX bytes (if Le present)	0x9000	Operation successful
-	0x6700	Wrong length (e.g. Lc and Le are both absent)
	0x6A8A	Modulation index not supported (P1 b7:b4)
	0x6401	Internal error (e.g. protocol setup failed or transceive failed)

11.1.4 0x96 - ISO/IEC 14443-3 command APDU

CLvA	INS	P1	P2	Lc	Data In	Le
0xFF	0x96	XX	XX	XX	Lc bytes	XX

11.1.4.1 ISO/IEC 14443-3 P1 coding

b7	b6	b5	b4	b3	b2	b1	b0	Description
--			ISO/IEC 14443 Type A commands					
			0	0	0	0	1	REQA
			0	0	0	1	0	WUPA
			0	0	0	1	1	HLTA
			ISO/IEC 14443 Type B commands					
			1	0	0	0	1	REQB (Number of slots in P2)
			1	0	0	1	0	WUPB (Number of slots in P2)

11.1.4.2 ISO/IEC 14443-3 P2 coding

b7	b6	b5	b4	b3	b2	b1	b0	Description
Number of slots for REQB/WUPB command								
RFU					xxx			$N = 2^{(b2b1b0)}$ (that is, for b2b1b0 = 0, N = 1)

11.1.4.3 ISO/IEC 14443-3 SW1SW2 response bytes

Data out	SW1SW2	
XX bytes (if Le present)	0x9000	Operation successful
-	0x6700	Wrong length (e.g. Lc absent when P1 b7 is set)
	0x6400	Execution error (e.g. command timeout)
	0x6401	Internal error (e.g. protocol setup failed or transceive failed)
	0x6A88	Requested buffer size too big

11.1.4.4 Cases for which data out is command dependent

Command	Data out
REQA	ATQA (2 bytes)
WUPA	ATQA (2 bytes)
HLTA	-
REQB	ATQB (14 bytes)
WUPB	ATQB (14 bytes)

11.1.5 0x9A: ICAO miscellaneous command APDU

CLA	INS	P1	P2	Lc	Data In	Le
0xFF	0x9A	XX	XX	XX	Lc bytes	XX

11.1.5.1 ICAO miscellaneous P1 coding

b7	b6	b5	b4	b3	b2	b1	b0	Description
0	0	0	0	0	1	0	0	Reader control (coded in P2)

Note: All other values of P1 are RFU.

11.1.6 ICAO miscellaneous P2 coding

b7	b6	b5	b4	b3	b2	b1	b0	Description
Coding for Reader control (Lc and Le both absent)								
0	0	0	0	0	0	0	0	Turn off polling for card (enter test mode)
0	0	0	0	0	0	0	1	Turn on polling for card (leave test mode)

Note: All other values of P2 are either RFU or not supported.

11.1.6.1 ICAO miscellaneous response

Data out	SW1SW2	
XX bytes (if Le present)	0x9000	Operation successful
-	0x6700	Wrong length (e.g. Le absent for reader information)
	0x6A82	Function not supported
	0x6A89	Information not available
	0x6A90	Trigger signal not available

This page is intentionally left blank.

Section 12

12 Using the PC_to_RDR_Escape command

The PC/SC layer does not allow the use of the SCardTransmit API unless the reader has previously signaled the presence and activation of a card. This prevents the use of commands such as the ICAO test commands or the HID commands without being able to properly recognize and activate a card. To be able to use these commands even without a previous card activation, the same functionality of pseudo-APDUs (CLA='FF') is provided through the PC_to_RDR_Escape command.

To use the PC_to_RDR_Escape command with the default Microsoft CCID driver, the functionality must be first enabled in the Windows registry.

To issue the PC_to_RDR_Escape command without a card being present, the reader must first be opened using the `SCardConnect` function with the following settings:

```
dwShareMode = SCARD_SHARE_DIRECT
dwPreferredProtocols = 0
```

The vendor IOCTL for the Escape command is defined as follows:

```
#define IOCTL_CCID_ESCAPE SCARD_CTL_CODE(3500)
```

The call will look something like one of the following:

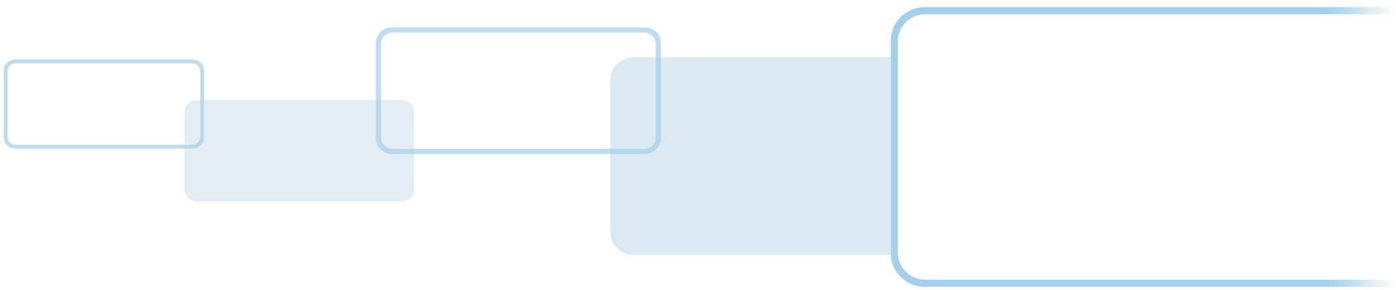
```
SCardControl(hCard, IOCTL_CCID_ESCAPE, …)
```

or:

```
SCardControl(hCard, SCARD_CTL_CODE(3500), …)
```

The data in the lpInBuffer parameter, of the length given in nInBufferSize, are copied to the abData field of the PC_to_RDR_Escape command. All the data in the response in RDR_to_PC_Escape abData field are copied back to the lpOutBuffer.

The abData field of the PC_to_RDR_Escape must contain the pseudo-APDU to be executed (typically, an ICAO test command or reader configuration). The maximum allowed size of abData in PC_to_RDR_Escape is currently 262 bytes. The maximum size of the response data in the abData field in the RDR_to_PC_Escape response is 464 bytes. The PC_to_RDR_Escape and RDR_to_PC_Escape do not support any form of chaining to extend the lengths of the supported parameters.



hidglobal.com