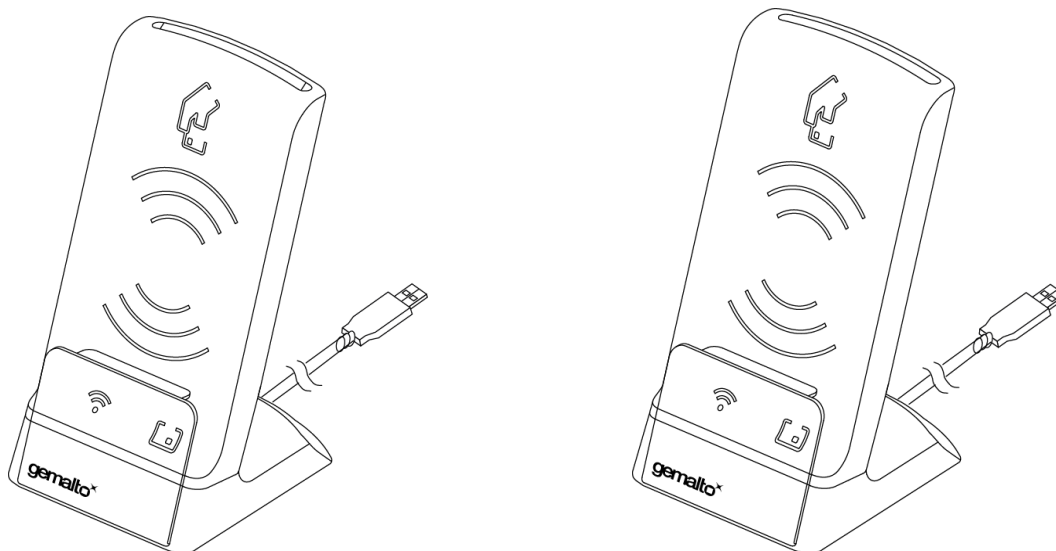


Prox-DU & Prox-SU

Dual interface USB smart card reader

Reference Manual



Prox-DU & Prox-SU

All information herein is either public information or is the property of and owned solely by Gemalto NV. and/or its subsidiaries who shall have and keep the sole right to file patent applications or any other kind of intellectual property protection in connection with such information.

Nothing herein shall be construed as implying or granting to you any rights, by license, grant or otherwise, under any intellectual and/or industrial property rights of or concerning any of Gemalto's information.

This document can be used for informational, non-commercial, internal and personal use only provided that:

- The copyright notice below, the confidentiality and proprietary legend and this full warning notice appear in all copies.
- This document shall not be posted on any network computer or broadcast in any media and no modification of any part of this document shall be made.

Use for any other purpose is expressly prohibited and may result in severe civil and criminal liabilities.

The information contained in this document is provided "AS IS" without any warranty of any kind. Unless otherwise expressly agreed in writing, Gemalto makes no warranty as to the value or accuracy of information contained herein.

The document could include technical inaccuracies or typographical errors. Changes are periodically added to the information herein. Furthermore, Gemalto reserves the right to make any change or improvement in the specifications data, information, and the like described herein, at any time.

Gemalto hereby disclaims all warranties and conditions with regard to the information contained herein, including all implied warranties of merchantability, fitness for a particular purpose, title and non-infringement. In no event shall Gemalto be liable, whether in contract, tort or otherwise, for any indirect, special or consequential damages or any damages whatsoever including but not limited to damages resulting from loss of use, data, profits, revenues, or customers, arising out of or in connection with the use or performance of information contained in this document.

Gemalto does not and shall not warrant that this product will be resistant to all possible attacks and shall not incur, and disclaims, any liability in this respect. Even if each product is compliant with current security standards in force on the date of their design, security mechanisms' resistance necessarily evolves according to the state of the art in security and notably under the emergence of new attacks. Under no circumstances, shall Gemalto be held liable for any third party actions and in particular in case of any successful attack against systems or equipment incorporating Gemalto products. Gemalto disclaims any liability with respect to security for direct, indirect, incidental or consequential damages that result from any use of its products. It is further stressed that independent testing and verification by the person using the product is particularly encouraged, especially in any application in which defective, incorrect or insecure functioning could result in damage to persons or property, denial of service or loss of privacy.

© Copyright 2011 Gemalto N.V. All rights reserved. Gemalto and the Gemalto logo are trademarks and service marks of Gemalto N.V. and/or its subsidiaries and are registered in certain countries. All other trademarks and service marks, whether registered or not in specific countries, are the property of their respective owners.

GEMALTO, B.P. 100, 13881 GEMENOS CEDEX, FRANCE.

Tel: +33 (0)4.42.36.50.00 Fax: +33 (0)4.42.36.50.90

Printed in France.

REVISION HISTORY

Date	Release	Comments
January 2010	A	First release for pre-production run samples V1.01-GXD14 version and before
March 2010	B	Update for serial product V1.04-GXD01 version and later Dimensions & weight correction EEPROM table is updated Pictures are updated Installation popup Windows update PC/SC names are updated MIFARE® Mini support Load Keys command is updated Interfacing with DESFire card paragraph added Requesting contactless info command added Read Binary and Update Binary commands updated USB strings updated Microsoft USB CCID Class Driver Details paragraph added Unused escape command removed: Card movement notification enable GemCore POS Pro chip escape command Linux and Mac USB CCID Class Driver Details paragraph added http://catalog.update.microsoft.com web link added Web link for HID libraries added
September 2010	C	Linux & MAC OS support correction Mac OS X Snow Leopard (10.6) is now supported RF parameters considered only when both RF Parameters Usage = 01h & RF Power Attenuation = 0Fh Gem_PC/SC V2 picture upgrade
February 2011	D	Upgrade after BSI TR-03119 certification Linux packages are now available Linux OpenSUSE now available New EEPROM parameters to support BSI test features A PC/SC Guide is now available PC/SC reader name updated for Linux and Mac OS A Release Note is now available for known issues and limitations Extended APDU supported by the contactless CCID interface HID report descriptor correction Warning related to the use of the native commands of the MIFARE® DESFire smart card BSI TR-03119 conformity paragraph added

TABLE OF CONTENTS

INTRODUCTION	12
OVERVIEW	13
DESCRIPTION	13
MAIN FEATURES	14
BSI TR-03119 CONFORMITY	15
PROX-DU AND PROX-SU DIFFERENCES	16
SMART CARD PROTECTION AND SWITCH FEATURE	17
USING SMART CARDS	18
ENVIRONMENTAL CHARACTERISTICS	20
INTERFACE CAPABILITY	21
INTERFACE FEATURES	22
USB SERIAL INTERFACE	22
CONTACTLESS INTERFACE	22
CONTACT INTERFACE	23
LED INTERFACE	24
INSTALLING THE READER/WRITER	25
WINDOWS XP INSTALLATION	27
Windows XP installation without the Windows Update procedure	27
Windows XP installation using the Windows Update procedure	28
CHECKING THE INSTALLATION	30
CHECKING THE SMART CARD DETECTION	30
CONFIGURING THE READER/WRITER	32
EEPROM PARAMETERS CONTENTS	32
Control parameters	33
EEPROM structure version	33
General parameters	33
Dual interface card protection	33
Card notification delay	34
Communication time out with GemCore POS Pro	34
Load MIFARE® Keys security option	34
Contactless automaton parameters	34
Automaton timing	34
Extended ATQB support	34
Allowed bit rates	34
T=CL card presence check behavior	35
Card type polling enable/disable	35
Deactivation / Reactivation behavior	35
Miscellaneous parameters	36
Overwrite FWI	36
Other bytes	36
General parameters	36
RF Reset time	36
RF On Delay	36
RF Parameters Usage	36
RF Power Attenuation	36
RF ISO level 2 control for BSI analog tests	36
RF parameters for ISO14443-A cards	37
RF parameters for ISO14443-B cards	37
EEPROM Parameters Validity	37

CRC control	37
MAD CRC calculation program	37
USING PC/SC APPLICATION	39
PC/SC OVERVIEW	39
GEM_PC/SC SOFTWARE TOOL	42
PROX-DU AND PROX-SU PC/SC READER NAME	43
WINDOWS OPERATING SYSTEMS	43
LINUX AND MAC OS X OPERATING SYSTEMS	44
PC/SC LIMITATIONS	45
INTERFACING WITH CONTACTLESS CARDS	46
DETECTING AN INSERTION	46
DETECTING A REMOVAL	46
ATR FOR CONTACTLESS SMART CARDS	47
INTERFACING WITH MIFARE® DESFIRE CARDS	49
REQUESTING CONTACTLESS SMART CARD INFORMATION	51
INTERFACING WITH MIFARE® CARDS	52
ATR FOR MIFARE® CARDS	53
GET DATA COMMAND	55
LOAD KEYS COMMAND	56
GENERAL AUTHENTICATE COMMAND	58
READ BINARY COMMAND	60
UPDATE BINARY COMMAND	61
ERROR CODE LIST SUMMARY	62
INTERFACING WITH CONTACT CARDS	64
DETECTING AN INSERTION	64
DETECTING A REMOVAL	64
ATR FOR CONTACT SMART CARDS	64
Structures and content	66
Structure of the subsequent characters in the ATR	66
Format character T0	66
Interface characters T _{Ai} , T _{Bi} , T _{CI} , T _{Di}	67
Historical characters T ₁ , T ₂ , ... , T _K	67
Check character TCK	68
Protocol type T	68
Specifications of the global interface bytes	68
TA1	68
TB1 and TB2	69
TC1	69
TA2	69
The first TA1 for T=15	70
The first TB for T=15	70
CCID DEVICES	71
CCID OVERVIEW	71
CCID communication pipes	71
CCID protocol and parameters selection	72
TPDU level of exchange	72
APDU level of exchange	73
Character level of exchange	73
Suspend behavior	74
CCID DEVICE FOR THE CONTACT INTERFACE	74

Prox-DU & Prox-SU

Command pipe bulk-out message for the contact card interface	74
PC_to_RDR_IccPowerOn command	75
PC_to_RDR_IccPowerOff command	75
PC_to_RDR_GetSlotStatus command	76
PC_to_RDR_XfrBlock command	76
PC_to_RDR_GetParameters command	77
PC_to_RDR_ResetParameters command	77
PC_to_RDR_SetParameters command	77
PC_to_RDR_Escape command	79
Switch interface	79
PC_to_RDR_Abort command	80
Response pipe bulk-in for the contact card interface	80
RDR_to_PC_DataBlock	80
RDR_to_PC_SlotStatus	81
RDR_to_PC_Parameters	81
RDR_to_PC_Escape	82
Reporting slot error and slot status registers in bulk-in messages for the contact interface	83
Interrupt in messages for the contact card interface	85
RDR_to_PC_NotifySlotChange	85
CCID DEVICE FOR THE CONTACTLESS INTERFACE	87
Command pipe bulk-out messages for the contactless interface	87
PC_to_RDR_IccPowerOn command	87
PC_to_RDR_IccPowerOff command	88
PC_to_RDR_GetSlotStatus command	88
PC_to_RDR_XfrBlock command	88
PC_to_RDR_GetParameters command	90
PC_to_RDR_ResetParameters Command	90
PC_to_RDR_SetParameters command	90
PC_to_RDR_Escape command	92
Switch interface	92
Get firmware version	92
PC_to_RDR_Abort command	92
Response pipe bulk-in messages for the contactless interface	93
RDR_to_PC_DataBlock Command	93
RDR_to_PC_SlotStatus Command	94
RDR_to_PC_Parameters Command	94
RDR_to_PC_Escape Command	96
Reporting slot error and slot status registers in bulk-in messages for the contactless interface	97
Interrupt in messages for the contactless card interface	97
RDR_to_PC_NotifySlotChange message	97
USB CCID CLASS DRIVER DETAILS	99
Overview	99
Microsoft CCID class driver	99
Enabling the CCID Escape Command feature into the Microsoft driver	100
CCID Escape Control Code for Microsoft Operating Systems	102
Linux and Mac CCID class driver	102
CCID Escape Control Code for Linux and Mac Operating Systems	103
HID DEVICES	104
GEMALTO PROPRIETARY COMMANDS	104
Proprietary commands	105
Firmware version request command	105
Read EEPROM parameters command	106
Write EEPROM parameters command	106

Prox-DU & Prox-SU

Switch interface command	107
Read switch interface state command	107
Reset reader command	108
Start download command	108
Download firmware file command	108
End download command	109
HID LIBRARY	109
HID COMMANDS ERROR CODES	109
FIRMWARE VERSIONING RULES	110
READER FIRMWARE STRING VERSION	110
BOOT-LOADER STRING VERSION	110
USB DESCRIPTORS	111
STANDARD USB DESCRIPTORS	111
Device descriptor	111
Configuration descriptor	111
Interfaces descriptors	112
DEVICE CLASS DESCRIPTORS	113
HID class descriptor	113
HID interface endpoint descriptor	113
HID report descriptor	113
Contactless smart card device class descriptor	115
Contactless smart card interface endpoint descriptors	116
Contact smart card device class descriptor	117
Contact smart card interface endpoint descriptors	118
STRING DESCRIPTORS	119
LangID string descriptor	119
Manufacturer string descriptor	119
Product string descriptor	119
Serial number string descriptor	120
HID interface string descriptor	121
Contactless smart card interface string descriptor	121
Contact smart card interface string descriptor	122
BOOT-LOADER	123
HARDWARE REQUIREMENT	123
BOOT-LOADER START UP OPERATIONS	123
BOOT-LOADER DOWNLOAD OPERATIONS	124
Start download command	124
Download firmware file command	124
End download command	125
Boot-loader version request command	125
Reset reader command	126
Boot-loader error codes	126
TYPICAL DOWNLOAD OPERATIONS	126
DOWNLOADED FILE FORMAT	126
BOOT-LOADER USB DESCRIPTORS	127
Device Descriptor	127
Configuration Descriptor	127
Interface descriptor	127
HID class descriptor	127
HID endpoint descriptor	127
HID report descriptor	127
String descriptors	128
LangID string descriptor	128

Prox-DU & Prox-SU

Manufacturer string descriptor	128
Product string descriptor	128
Serial number string descriptor	128
HID interface string descriptor.....	128
LEDS STATES FOR THE BOOT-LOADER.....	129
DOWNLOADING A FIRMWARE.....	130
DOWNLOAD TOOL OPERATIONS	130
MIFARE® CARDS MAPPING	134
MIFARE® 1K MEMORY MAPPING	134
MIFARE® MINI MEMORY MAPPING	135
MIFARE® 4K MEMORY MAPPING	136
MIFARE® UL MEMORY MAPPING	138
Serial Number Area	138
Lock Bytes Area.....	139
OTP Bytes Area.....	139
Data Bytes Area.....	139
MIFARE® UL Read/Write Operation.....	139
MIFARE® MEMORY ORGANIZATION	140
Sector Trailer	140
Authentication Keys.....	140
Access Bits.....	141
Data Block Access Conditions	141
Sector Trailer Access Conditions	143
FOR MORE INFORMATION.....	145
STANDARDS AND SPECIFICATIONS.....	145

TABLE LIST

Table 1 – Dual interface USB smart card reader/writer models.....	12
Table 2 – Prox-DU and Prox-SU differences.....	16
Table 3 – Environmental Characteristics	20
Table 4 – Interface capability	21
Table 5 – Supported Operating Systems	26
Table 6 – EEPROM parameters contents.....	33
Table 7 – Smart Card Database Query Functions.....	40
Table 8 – Smart Card Database Management Functions.....	40
Table 9 – Resource Manager Context Functions	40
Table 10 – Resource Manager Support Function	41
Table 11 – Smart Card Tracking Functions	41
Table 12 – Smart Card and Reader Access Functions.....	41
Table 13 – Direct Card Access Functions.....	41
Table 14 – ATR for contactless Smart cards	47
Table 15 – ATR for MIFARE® cards	53
Table 16 – SS Byte for Standard	54
Table 17 – NN Bytes for Card Name	54
Table 18 – Memory card error codes	63
Table 19 – ATR for contact smart cards	66
Table 20 – Clock rate conversion factor F	68
Table 21 – Bit rate adjustment factor D.....	69
Table 22 – clock stop indicator X	70

Prox-DU & Prox-SU

Table 23 – class indicator Y	70
Table 24 – Slot error register when bmCommandStatus = 1	84
Table 25 – Slot Status register	84
Table 26 – Common error codes	109
Table 27 – USB Device Descriptor	111
Table 28 – USB Configuration Descriptor	112
Table 29 – USB HID Interface Descriptor	112
Table 30 – USB Contactless Smart Card Interface Descriptor	112
Table 31 – USB Contact Smart Card Interface Descriptor	113
Table 32 – USB HID Class Descriptor	113
Table 33 – USB HID Interface Endpoint Descriptor	113
Table 34 – USB HID Report Descriptor	114
Table 35 – USB Contactless Smart Card Device Class Descriptor	116
Table 36 – USB Contactless Smart Card Interface Endpoint Descriptor (Bulk Out)	116
Table 37 – USB Contactless Smart Card Interface Endpoint Descriptor (Bulk In)	116
Table 38 – USB Contactless Smart Card Interface Endpoint Descriptor (Interrupt In).....	117
Table 39 – USB Contact Smart Card Device Class Descriptor	118
Table 40 – USB Contact Smart Card Interface Endpoint Descriptor (Bulk Out).....	118
Table 41 – USB Contact Smart Card Interface Endpoint Descriptor (Bulk In).....	118
Table 42 – USB Contact Smart Card Interface Endpoint Descriptor (Interrupt In)	119
Table 43 – USB LangID String Descriptor	119
Table 44 – USB Manufacturer String Descriptor.....	119
Table 45 – USB Product String Descriptor.....	120
Table 46 – USB Serial Number String Descriptor.....	120
Table 47 – USB HID Interface String Descriptor.....	121
Table 48 – USB Contactless Smart Card Interface String Descriptor.....	122
Table 49 – USB Contact Smart Card Interface String Descriptor	122
Table 50 – Boot-loader HID error codes	126
Table 51 – USB Boot-loader Configuration Descriptor	127
Table 52 – USB Boot-loader Interface String Descriptor	129
Table 53 – LEDs states for the Boot-loader LEDs	129
Table 54 – Memory Sectors of MIFARE [®] 1K.....	134
Table 55 – Memory Sectors of MIFARE [®] Mini	135
Table 56 – Memory Sectors of MIFARE [®] 4K.....	137
Table 57 – Memory mapping of MIFARE [®] UL	138
Table 58 – Access to Data Blocks	143
Table 59 – Access to Sector Trailer	144

FIGURE LIST

Figure 1 – Prox-DU view	13
Figure 2 – Prox-SU view.....	13
Figure 3 – Prox-DU with the stand for vertical use.....	14
Figure 4 – Prox-DU ID-1 size slot.....	16
Figure 5 – Prox-SU ID-000 size slot.....	16
Figure 6 – Dual interface smart card view.....	17
Figure 7 – The contactless smart card is put near or over the Prox-DU landing zone	18
Figure 8 – The contactless smart card is not fully inserted into the Prox-DU slot.....	18
Figure 9 – The contactless smart card is fully inserted into the Prox-DU slot (after the switch activation)	18
Figure 10 – The contact smart card is fully inserted into the Prox-DU slot.....	19
Figure 11 – The contactless smart card is put near or over the Prox-SU landing zone.....	19
Figure 12 – The contact SIM/SAM card is inserted into the Prox-SU connector	19
Figure 13 – USB devices (Windows XP example).....	22
Figure 14 – Contactless logo of the landing zone	22
Figure 15 – Contact card slot (Prox-DU and Prox-SU)	23

Prox-DU & Prox-SU

Figure 16 – ID-1 and ID-000 card size 23

Figure 17 – Visual indicators 24

Figure 18 – Prox-DU Installation popup dialog boxes 28

Figure 19 – Windows XP Installation wizard: first window 28

Figure 20 – Windows XP Installation wizard: second window 29

Figure 21 – Windows XP Installation wizard: third window 29

Figure 22 – Windows XP Installation wizard: final window 29

Figure 23 – USB smart card reader icons in the Device Manager window (Windows XP) .. 30

Figure 24 – USB HID icons in the Device Manager window (Windows XP) 30

Figure 25 – Contactless smart card check 30

Figure 26 – Contact smart card check 31

Figure 27 – PC/SC Architecture 39

Figure 28 – Gem_PCSC window 42

Figure 2 – Prox-DU PC/SC name (Windows) 43

Figure 3 – Prox-SU PC/SC name (Windows) 43

Figure 4 – Prox-DU and Prox-SU PC/SC names (Windows) 43

Figure 5 – Two Prox-DU PC/SC names (Windows) 43

Figure 6 – Prox-DU PC/SC name (Linux) 44

Figure 7 – Prox-SU PC/SC name (Linux) 44

Figure 8 – Prox-DU and Prox-SU PC/SC names (Linux) 44

Figure 9 – Two Prox-DU PC/SC names (Linux) 44

Figure 33 – Information provided by T0 67

Figure 34 – Information provided by TDi 67

Figure 35 – Push button PCB location (S1) 123

Introduction

This reference manual provides information on the use of the Prox-DU and the Prox-SU dual interface (contactless and contact) USB smart card reader/writer.

This document is applicable to following reference, revision C and later:

Model	Reference	Comments
Prox-DU	HWP118184	Dual interface USB smart card reader Contact & contactless
Prox-SU	HWP118185	Contactless interface USB smart card reader With optional SIM/SAM slot
Prox-DU with stand	HWP118830	Prox-DU with a stand for vertical use
Prox-SU with stand	HWP118831	Prox-SU with a stand for vertical use

Table 1 – Dual interface USB smart card reader/writer models

For information on installation, please refer to the “*Installation Guide*” document.

Who Should Read This Book

This reference manual is designed for developers of PC/SC smart card application or driver. For driver design, familiarity with the USB protocol is recommended.

Conventions

Bit Numbering

A byte consists of 8 bits, b7 to b0, where b7 is the most significant bit and b0 is the least significant bit.

One byte	b7	b6	b5	b4	b3	b2	b1	b0
----------	----	----	----	----	----	----	----	----

Byte Numbering

A string of n bytes consists of n number of concatenated bytes: Bn...B3...B0.

Bn is the most significant byte and B0 is the least significant byte:

A string of n bytes	Bn	Bn-1	-	-	-	B2	B1	B0
------------------------	----	------	---	---	---	----	----	----

Contact Our Hotline

If you do not find the information you need in this document, or if you find errors, contact the Gemalto hotline at <http://support.gemalto.com/>.

Please note the document reference number, your job function, and the name of your company. (You will find the document reference number at the bottom of the document.)

Overview

Description

The Prox-DU and the Prox-SU are Gemalto smart card reader/writers embedding the Prox and the GemCore technologies developed by Gemalto to interface contactless and contact smart cards:

- The Prox-DU is a **dual interface (contact and contactless)** USB smart card reader/writer:

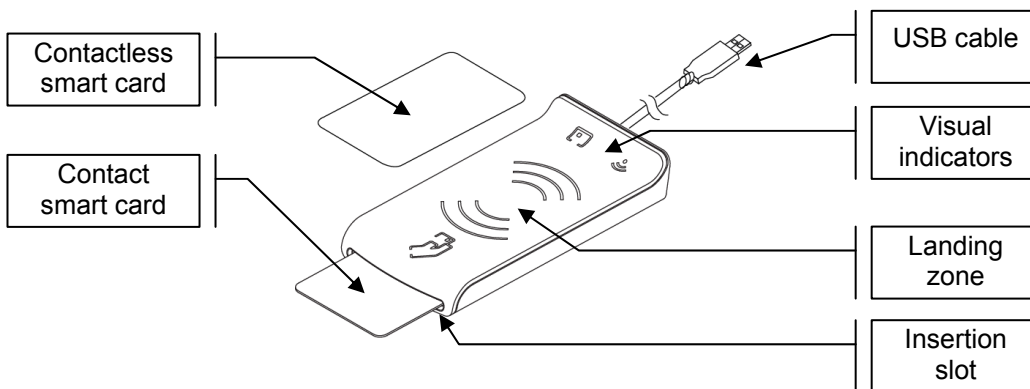


Figure 1 – Prox-DU view

- The Prox-SU is a **contactless interface** USB smart card reader including an internal SIM/SAM card slot:

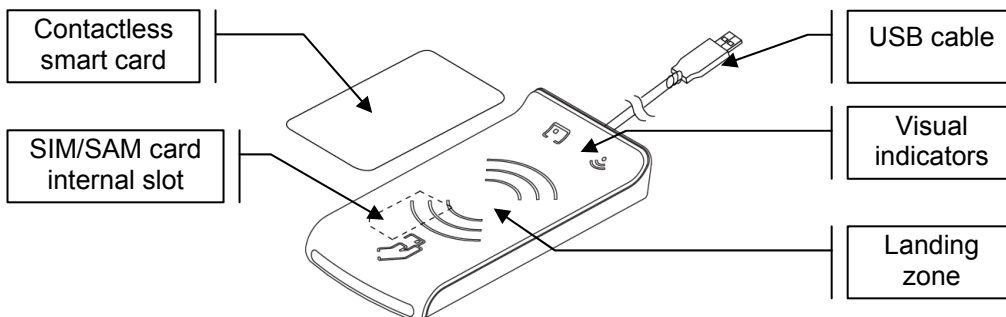


Figure 2 – Prox-SU view

The Prox technology complies with ISO14443 standard related to proximity cards applicable to type A and type B contactless smart cards.

The Prox technology uses MIFARE® (a registered trademark of NXP) technology as part of its integrated solution.

The GemCore technology complies with ISO7816 and EMV standard related to contact smart cards.

Both technologies also provide visual feedback for each smart card interface.

Main Features

The Prox-DU and The Prox-SU have the following common features:

- Up-to-date architecture using the Gemalto Prox and GemCore technologies
- Ability to drive any type of ISO14443-A&B T=CL contactless smart cards
- Ability to drive any type of MIFARE[®] contactless smart cards
- Ability to drive any type of ISO7816 contact smart card or SIM/SAM card
- Support for smart card with a higher baud rate (contact and contactless)
- Easily upgradeable download of the latest features
- Standard USB Full speed interface, bus powered (no external power supply required)
- Unique USB serial number which enables that the device can be plugged into any USB slot on a computer without having to re-install the driver
- Standard CCID interface for both smart card slot (contact and contactless)
- Standard HID interface for device administration
- No need of a proprietary USB driver. The standard CCID and HID drivers of the computer can be used
- PC/SC V2.0 compliant
- Embedded protection against dual interface smart card damage and switch feature to select the active interface (contact or contactless)

Prox and GemCore are Gemalto proprietary technologies developed for contactless and contact reader/writers. It is based on a specific operating system that interfaces with contactless and contact smart cards.

A stand can be attached to the reader/writer for vertical use:

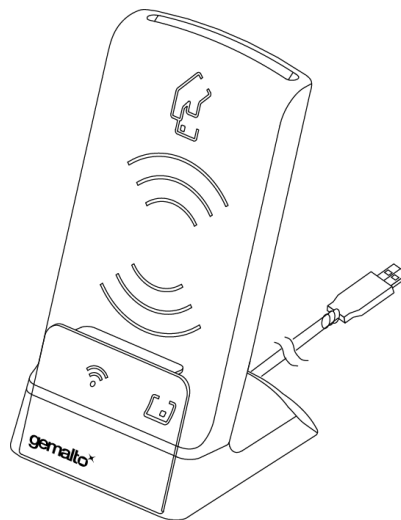


Figure 3 – Prox-DU with the stand for vertical use

BSI TR-03119 Conformity

The BSI TR-03119 certificate N° BSI-K-TR-0078-2010 recognizes the ability of the Gemalto Prox-SU and Prox-DU smart card readers to interface with the new German electronic identity cards called nPA (neue Personalausweis) as a “Basic Chip Card Reader Category B”.

This certification includes a compliance with the next specifications:

- BSI TR-03105 Part 4 specification related to the test plan for ICAO compliant Proximity Coupling Device (PCD) on layer 2-4,
- Additional environmental and safety tests according to BSI TR-03119 attachment B.1,
- Functional tests according to BSI TR-03119 attachment B.2:
 - Installation of the smart card reader on different operating systems
 - Functional tests related to the use of the nPA smart card as card recognition, secret code input or change, or online authentication.

The conformity of the product Prox-SU / Prox-DU (with or without the stand) to the Technical Guideline BSI TR-03119 has been evaluated by evaluation facilities recognized according to DIN ISO/IEC 17025 and was confirmed by the German Federal Office for Information Security (BSI).



The following Test Standards have been applied for the performance of the conformity evaluation:

BSI TR-03119 – Anforderungen an Chipkartenleser mit ePA Unterstützung (Requirements for Chipcard Reader Devices with ePA support), Version 1.1.

The product meets the requirements of the Technical Guideline BSI TR-03119.

Prox-DU and Prox-SU differences

The main difference between the Prox-DU and Prox-SU models is related to the smart card slot:

- Prox-DU: the smart card slot located in the top cabinet is open. The user can insert or remove its **ID-1 size** smart card directly into or from the reader slot

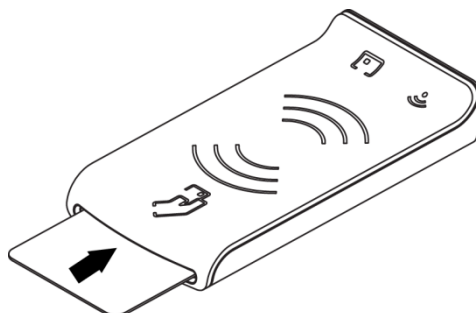


Figure 4 – Prox-DU ID-1 size slot

- Prox-SU: the smart card slot located in the top cabinet is **closed**. The user should open the casing before inserting its **ID-000 size** smart card into the dedicated connector. When the casing is closed the SIM/SAM card cannot be removed.

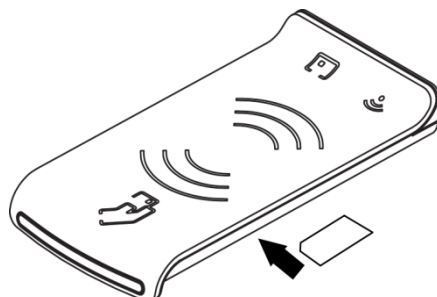


Figure 5 – Prox-SU ID-000 size slot

The following table shows all the different features supported by the Prox-DU and the Prox-SU reader/writers:

Feature	Prox-DU	Prox-SU
Contact card interface	ID-1 size format (smart card) Removable	ID-000 size format (SIM/SAM) Not removable
Dual interface protection	Managed Can be disabled according to device configuration	No management
Switch interface command	Available	Not useful
Product name in the string version	Gemalto Prox-DU	Gemalto Prox-SU
Product string in USB descriptor	Prox Dual USB PC Link Reader	Prox SU USB PC Link Reader
LEDs	The LED of the contact interface is blinking when no card is inserted	The LED of the contact interface is Off when no SIM/SAM is present

Table 2 – Prox-DU and Prox-SU differences

Smart card protection and switch feature

As the Prox-DU can interface two smart cards simultaneously, a dedicated protection system is included into the device to avoid to damage dual interface smart cards (both contact and contactless).

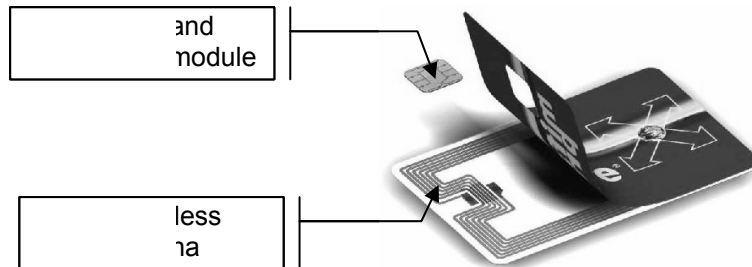


Figure 6 – Dual interface smart card view

To protect the smart card from undesired operation only one interface will be activated at a given time:

- The contact interface is deactivated when a contactless smart card is detected by the reader. The contact smart card power supply will be turned off to avoid powering the contact smart card.
- The contactless interface is deactivated when a contact card is detected by the reader. The RF field is turned off to avoid powering the contactless smart card.

The contactless interface will be activated again when the smart card is removed from the slot or when a dedicated **switch command** is send to the device by the application.

This feature will enable the user to **communicate with the two interfaces without moving the smart card from the slot.**

The dual interface smart card protection is enabled by default into the Prox-DU device. It can be disabled if needed by changing the reader/writer configuration. When disabled the two smart card interfaces are available simultaneously.

Note: As it is not possible to have a dual interface SIM/SAM card, the smart card protection is not active with the Prox-SU device.

Using Smart Cards

The Prox-DU and Prox-SU reader/writers must be used with contactless and contact smart cards. Depending on the model, the following pictures show the different ways to use the smart cards:

The Prox-DU and a contactless smart card:

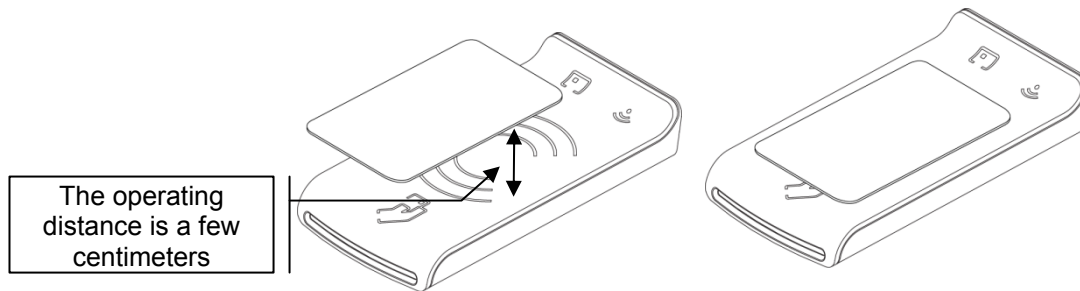


Figure 7 – The contactless smart card is put near or over the Prox-DU landing zone



Figure 8 – The contactless smart card is **not fully** inserted into the Prox-DU slot

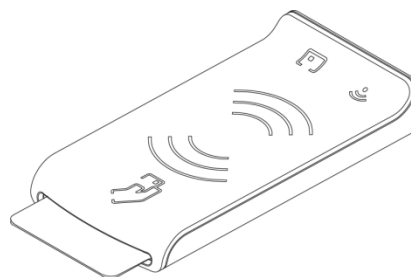


Figure 9 – The contactless smart card is **fully** inserted into the Prox-DU slot (after the switch activation)

Note: if the contactless smart card is fully inserted into the slot, the contactless interface will be deactivated because the contact interface has a higher priority than the contactless interface. To activate the contactless smart card when it is fully inserted into the slot, the switch feature of the device should be activated. Refer to the switch feature paragraph below for more information.

Prox-DU & Prox-SU

The Prox-DU and a contact smart card:

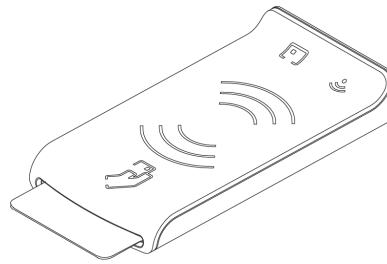


Figure 10 – The contact smart card is fully inserted into the Prox-DU slot

The Prox-SU and a contactless smart card:

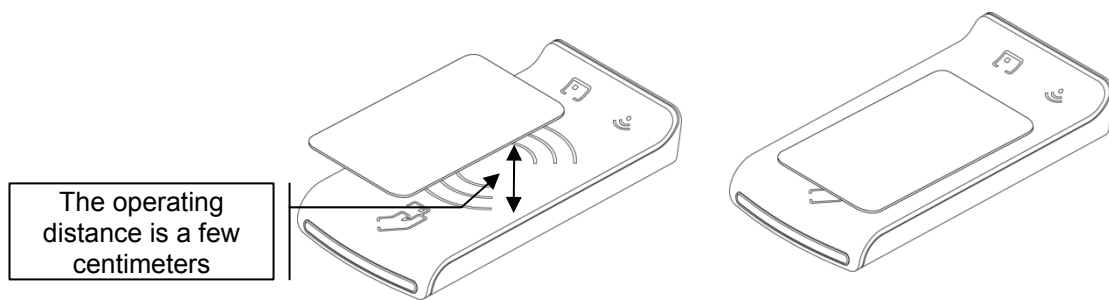


Figure 11 – The contactless smart card is put near or over the Prox-SU landing zone

The Prox-SU and a contact SIM/SAM card:

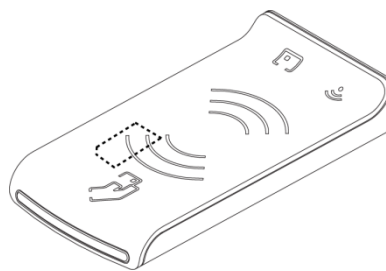


Figure 12 – The contact SIM/SAM card is inserted into the Prox-SU connector

Note: the SIM/SAM card cannot be installed or removed without opening the casing. The SIM/SAM card is permanently installed into the Prox-SU reader.

Environmental Characteristics

For an optimal performance, operate the Prox-DU and the Prox-SU under the following environment conditions:

Description	Value or Range
Operating Temperature	0°C to +50°C (+32°F to +122°F)
Storage Temperature	-20°C to +60°C (-4°F to +140°F)
MTBF reliability	900,000 hours at 20°C (MIL-HDBK-217F grade GB)
Humidity Range	0% to 95% non-condensing
Protection Index	Prox-DU: IP20 (open-case device) Prox-SU: IP40 (dustproof device)
EEPROM data	10 years minimum retention 100000 erase/write cycles
Physical Dimensions	Without stand: 26 mm x 69 mm x 126 mm max. (height x width x depth) With stand: 132 mm x 69 mm x 79 mm max. (height x width x depth)
Weight	Without stand: 145 g max. With stand: 255 g max.
Cabling Distance	1.80 m USB cable
Power Supply Voltage	USB bus powered
Power Supply Current Operating RF On	< 200 mA
Power Supply Current Operating RF Off	< 50 mA
Power Supply Current Suspend	< 2.5 mA
EMC Regulations	CE FCC Part 15 Class B
Safety	UL 60950 Recognized
Table 3 – Environmental Characteristics	

Interface capability

The Prox-DU and the Prox-SU devices support the following interfaces:

- One USB interface
- One contact smart card interface
- One contactless smart card interface
- Two visual indicators

Depending on the smart card type, the connectors used will be as shown in the next table:

Interface	Prox-DU Connector type	Prox-SU Connector type
USB	USB A plug	USB A plug
Contact card	ISO7810 ID-1 size ISO7816-2 8 pins	ISO7810 ID-000 size ISO7816-2 8 pins
Contactless card	Landing zone (No connector)	Landing zone (No connector)
Visual Indicators	One blue LED One yellow LED	One blue LED One yellow LED

Table 4 – Interface capability

Interface Features

USB serial interface

The USB interface is available with the Prox-DU and the Prox-SU.

The USB interface is USB 2.0 full speed compliant.

The USB interface is a **composite device** composed of the following devices:

- One USB Smart Card reader for the contact interface
- One USB Smart Card reader for the contactless interface
- One USB Human Interface Device for the reader administration

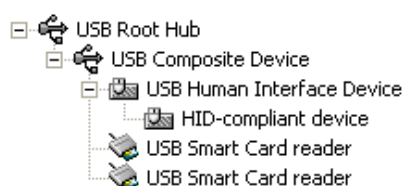


Figure 13 – USB devices (Windows XP example)

The USB interface does not require a specific driver for use with various operating systems.

The standard USB CCID driver included into the operating system of the computer is used for the smart card interface.

The standard USB HID driver included into the operating system of the computer is used for the device administration.

The selective suspend is not supported by the USB interface. Only standard suspend is supported.

Contactless interface

The contactless interface is available with the Prox-DU and the Prox-SU and is composed of a landing zone located in the front cabinet defined by the following contactless logo:



Figure 14 – Contactless logo of the landing zone

The contactless antenna is integrated into the device. It consists of inductive loops and a matching circuit mounted into the printed circuit board.

This contactless interface complies with the ISO14443-A&B standard.

The characteristics for the contactless interface are as follows:

Prox-DU & Prox-SU

- Contactless type:
 - ISO14443-A Memory cards (MIFARE®)
 - ISO14443-A Microprocessor cards (T=CL)
 - ISO14443-B Microprocessor cards (T=CL)
 - Automatic scan between ISO14443-A and ISO14443-B
- Contactless baud rate:
 - 106 kbps - 212 kbps - 424 kbps - 848 kbps
- Contactless protocol:
 - MIFARE® classic
 - ISO14443-4 (T=CL)
- Contactless commands:
 - Compliant with PC/SC V2.0 Part 3 Revision 2.01.09 specifications

Contact interface

The contact interface is available with the Prox-DU and the Prox-SU and is composed of a smart card connector with a smart card slot located in the front cabinet for the Prox-DU and located inside the casing for the Prox-SU.

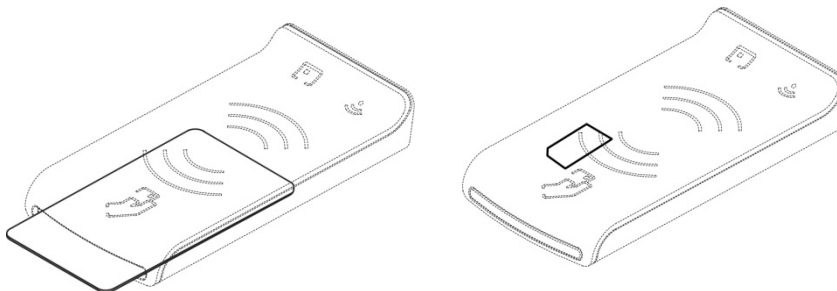


Figure 15 – Contact card slot (Prox-DU and Prox-SU)

The characteristics for the contact interface are as follows:

- Card type:
 - Asynchronous (Microcontroller based)
- Card size:
 - ID-1 for the Prox-DU (full size form factor)
 - ID-000 for the Prox-SU (SIM/SAM form factor)

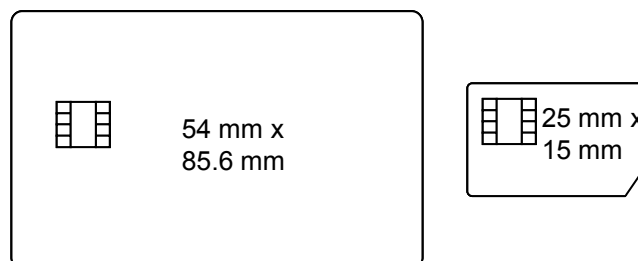


Figure 16 – ID-1 and ID-000 card size

- Card baud rate:
 - up to 500 kbps (TA1 = 97h for a clock frequency of 4 MHz)

- Card protocol:
 - T=0 and T=1 protocols
- Card compliance:
 - EMV and PC/SC modes. The interface is compliant with the EMV version 4.0 specifications.
 - ISO 7816-3 and -4 and ability to supply the cards with 5 V, 3 V, or 1.8 V (Class A, B, or C cards respectively)
 - IAS version 1.0 certified (Identification, Authentication & Signature)

LED interface

Two LEDs are available as visual indicators:

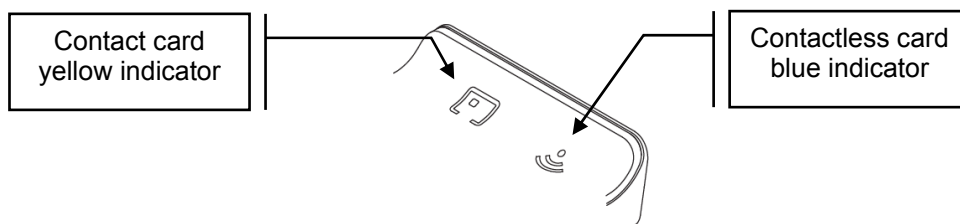


Figure 17 – Visual indicators

The indicators give information about the smart card interface as described in the next table:

Card Indicator	Description
Off	The card interface is deactivated
Slow Blink (0,2 s / 2 s)	The card interface is ready
Blink (0,5 s / 1 s)	The card is present and inactive (powered off)
On	The card is present and active (powered on)
Flashing	The card communication is in progress

Note: When no SIM/SAM card is inserted into the Prox-SU, the yellow indicator will always be off.












Installing the reader/writer

When the USB cable is plugged into the USB port of the computer:

- The two LED indicators will flash shortly to indicate the user that it has started properly,
- The installation wizard of the operating system will appear.

Follow the on-screen instructions, the wizard will automatically install the drivers required by the operating system.

The following table shows the driver to consider regarding the operating system:

Operating system	CCID driver	HID driver
Microsoft Windows 2000 	Use Windows Update	Inbox
Microsoft Windows XP 32/64 bits 	Use Windows Update	Inbox
Microsoft Windows Vista 32/64 bits 	Inbox	Inbox
Microsoft Windows 7 32/64 bits 	Inbox	Inbox
Microsoft Windows CE 5.0 & 6.0 	Upon request	Inbox
Microsoft Windows CE 6.0R2 	Inbox	Inbox
Linux Debian distribution Release 5.0x and higher (32 and 64 bit versions) 	Use the latest Debian installation package available in the web site http://support.gemalto.com	Inbox
Linux Ubuntu distribution Release 9.04 and higher (32 and 64 bit versions) 	Use the latest Ubuntu installation package available in the web site http://support.gemalto.com	Inbox
Linux OpenSUSE distribution Release 11.1 and higher (32 and 64 bit versions) 	Use the latest OpenSUSE installation package available in the web site http://support.gemalto.com	Inbox
Linux Red Hat distribution Release 5 and higher (32 and 64 bit versions) 	Use the latest CCID package (CCID driver V1.4.0 minimum). If not operating, use the Debian source code available on the following web site: http://pcsc-lite.alioth.debian.org/ccid.html	Inbox
Mac OS X Tiger (10.4) 32 bits edition, for Intel and Power PC platforms 	Use the latest Mac OS 10.4 installation package available in the web site http://support.gemalto.com	Inbox

<p>Mac OS X Leopard (10.5) 32 bits edition, for Intel and Power PC platforms</p>		<p>Use the latest Mac OS 10.5 installation package available in the web site http://support.gemalto.com</p>	<p>Inbox</p>
<p>Mac OS X Snow Leopard (10.6) 32/64 bits edition, for Intel platforms</p>		<p>Use the latest Mac OS 10.6 installation package available in the web site http://support.gemalto.com</p>	<p>Inbox</p>

Table 5 – Supported Operating Systems

Note that all the drivers needed for the Prox-DU and the Prox-SU reader/writer are the standard drivers available into the operating system. No Gemalto proprietary drivers are needed, Microsoft Windows CE R5.0 & R6.0 except.

For Windows operating system the following web link can be used to get a cabinet containing the driver files:

<http://catalog.update.microsoft.com/v7/site/Search.aspx?q=Microsoft%20ccid>

If needed the following web link <http://support.gemalto.com> will give instructions how to get these drivers.

The next paragraph will detail the installation wizard for Microsoft Windows XP operating system.

For other operating systems, please refer to the “*Computer Installation Guide*” for more information.

Windows XP installation

The HID driver is always available in all the operating systems and the HID device will be installed automatically.

If the CCID driver is available in the operating system, the two CCID devices will be installed automatically, as described in the next paragraph “Windows XP installation without the Windows Update procedure”.

If the CCID driver is not available in the operating system, the two CCID devices will be installed after the Windows Update procedure, as described in the next paragraph “Windows XP installation using the Windows Update procedure”.

Windows XP installation without the Windows Update procedure

These installation steps will be effective only if the USB CCID driver is available in the operating system.

When the USB cable is plugged into the USB port of the computer the following popup dialog boxes will be successively displayed over the task bar:



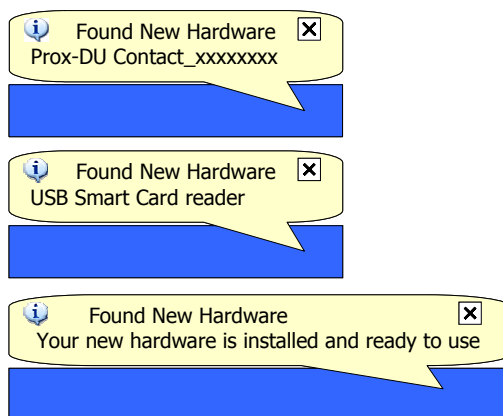


Figure 18 – Prox-DU Installation popup dialog boxes

xxxxxxx is the serial number printed on the label located on the back of the reader/writer.

Your Prox-DU or Prox-SU device is now ready to use.

Note: the popup dialog boxes will only appear the first time the device is connected to the computer.

Windows XP installation using the Windows Update procedure

These installation steps will be effective if the USB CCID driver not available in the operating system.

When the USB cable is plugged into the USB port of the computer the previous popup windows and the following wizard will appear:



Figure 19 – Windows XP Installation wizard: first window

- Click the **“Yes, this time only”** button to start the Windows Update procedure

Prox-DU & Prox-SU

- Click the “**Next**” button to continue (in the picture below 09A00235 is the serial number printed on the label located on the back of the reader/writer)

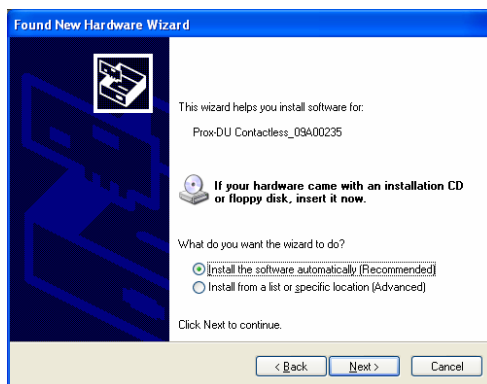


Figure 20 – Windows XP Installation wizard: second window

- Click the “**Install the software automatically (Recommended)**” button
- Click the “**Next**” button to continue

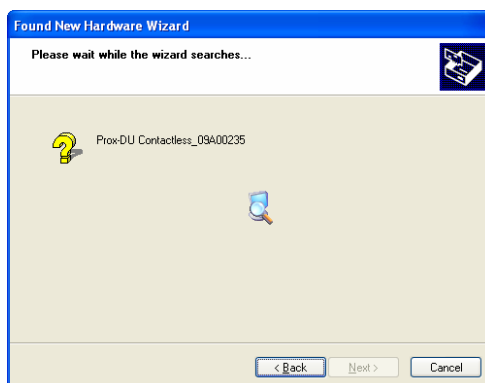


Figure 21 – Windows XP Installation wizard: third window

The Windows Update procedure will be running. Wait until the installation is completed:



Figure 22 – Windows XP Installation wizard: final window

Now the installation is finished. Your Prox-DU or Prox-SU device is ready to use.

Note: depending on the network configuration, the Windows Update procedure can take a **long** time. Please wait until the end of the procedure.

Checking the installation

To check if all the drivers have been properly installed, perform the following steps:

Check that the devices are recognized by the Device Manager (Windows XP):

- Right click the **“My Computer”** icon on the Desktop
- Select the **“Properties”** menu
- Select the **“Hardware”** tab
- Click the **“Device Manager”** button
- Click the **“Smart card readers”** icon

Two **“USB Smart Card reader”** icons should be displayed as shown in the next figure:

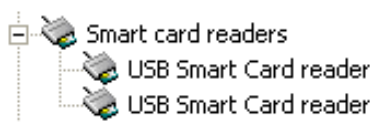


Figure 23 – USB smart card reader icons in the Device Manager window (Windows XP)

- Click the **“Human Interface Devices”** icon

Two **“HID devices”** icons should be displayed as shown in the next figure:

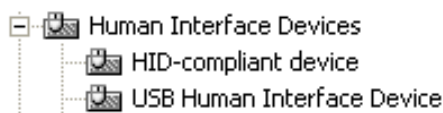


Figure 24 – USB HID icons in the Device Manager window (Windows XP)

Checking the smart card detection

To check if the Prox-DU or the Prox-SU reader/writer is able to detect contactless smart cards put a smart card near the reader/writer antenna:

The blue LED should be set to an enlightened state.

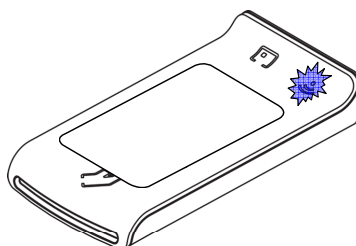


Figure 25 – Contactless smart card check

Note: The blue LED should return to the blinking state after a short time depending on the computer operating system. The smart card used for the check should be of course ISO14443-A or ISO14443-B compliant. Otherwise, no LED change will occur.

To check if the Prox-DU reader/writer is able to detect contact smart cards insert a smart card into the reader/writer slot:

The yellow LED should be set to an enlightened state.



Figure 26 – Contact smart card check

Note: The yellow LED should return to the blinking state after a short time depending on the computer operating system. The smart card used for the check should be of course ISO7816-3 compliant. Otherwise, no LED change will occur. The second test is not available for the Prox-SU model.

Configuring the reader/writer

The Prox-DU and Prox-SU device's configuration is stored into an internal EEPROM memory specifying various parameters used by the devices for their operation.

For normal use it is not needed to modify the reader/writer configuration.

These parameters can be modified if needed using the "Write EEPROM" and "Read EEPROM" commands supported by the HID interface. Refer to the "Proprietary commands on the HID interface" paragraph for more information.

Note: Special care should be considered on the understanding of the parameters. A modified value may result in an unexpected operation of the Prox-DU and Prox-SU device if the parameter is not known by the user.

EEPROM parameters contents

The next table shows the EEPROM parameters contents:

Offset	Usage	Default Value
Control parameters		
0	EEPROM structure version	08h
General parameters		
1	Dual interface card protection	00h
2	Card notification delay (x 100 ms)	01h
3	Communication time-out with GemCore POS Pro (x10 sec)	11h
4	Load MIFARE® keys security option	00h
5 to 8	RFU (4 bytes)	00h
Contactless automaton parameters		
9	Automaton timing	11h
10	Extended ATQB support	00h
11	RFU (1 byte)	00h
12	Allowed bit rates list	77h
13	T=CL card presence check behavior	00h
14	Card type polling enable/disable	80h
15	Deactivation / Reactivation behavior	02h
16	RFU (1 byte)	00h
Miscellaneous parameters		
17	Overwrite FWI	FFh
18 to 24	RFU (7 bytes)	00h
General RF parameters		
25	RF reset time	00h
26	RF on delay	00h
27	RF parameters use	00h
28	RF power attenuation	00h
29 to 31	RFU (3 bytes)	00h
32	RF ISO level 2 control for BSI analog tests	00h
RF parameters for ISO14443-A cards		

33	RxThreshold for 106 kbps	84h
34	RxThreshold for 212 kbps	84h
35	RxThreshold for 424 kbps	84h
36	RxThreshold for 848 kbps	84h
37	RFCfg	58h
38	TypeB	00h
39	GsN	F8h
40	CWGsP	3Fh
41	ModGsP	3Fh
42 to 48	RFU (7 bytes)	00h
RF parameters for ISO14443-B cards		
49	RxThreshold for 106 kbps	84h
50	RxThreshold for 212 kbps	84h
51	RxThreshold for 424 kbps	84h
52	RxThreshold for 848 kbps	84h
53	RFCfg	58h
54	TypeB	92h
55	GsN	F8h
56	CWGsP	3Fh
57	ModGsP	28h
58 to 64	RFU (7 bytes)	00h
EEPROM parameters Validity		
65	CRC control	5Bh

Table 6 – EEPROM parameters contents

Control parameters

EEPROM structure version

This byte defines the structure of the parameters in the EEPROM.

At start up, if the structure version is not the same than the structure version known by the firmware, the EEPROM is reinitialized.

General parameters

These bytes define the general behavior of the device.

Dual interface card protection

This byte defines the dual interface card protection (for Prox-DU only):

00h: protection on

01h: protection off

When the Dual interface card protection is on:

The contact smart card cannot be supplied with a VCC voltage while the RF field is on.

The RF field cannot be set on while a contact smart card is supplied with a VCC voltage.

Note: The Prox-SU reader does not use this parameter.

Card notification delay

This byte defines the minimum time between the notifications of card movement:
Time unit = 100 ms

Communication time out with GemCore POS Pro

This byte defines the internal communication time-out with the GemCore POS Pro chip controlling the contact interface:
Time unit = 10 seconds

Load MIFARE[®] Keys security option

This byte defines the Load MIFARE[®] keys security option:
00h: security option off
01h: security option on

When the Load MIFARE[®] keys security option is set on, the PC/SC “Load Keys” command must be used with an additional secret transport key. Refer to the “Load Keys” paragraph for more information.

Contactless automaton parameters

Automaton timing

This byte defines the timing of the contactless automaton scanning for contactless smart cards:

- b0-b3: periodic time to search or check for a card presence (unit = 100 ms)
0.1 sec to 1.6 sec (0 is not allowed)
- b7-b4: release time after the last host command (unit = 1 second)
1 to 16 sec (0 is not allowed)

Extended ATQB support

This byte defines the Extended ATQB support option as defined in the ISO14443 standard:
00h: Extended ATQB is not supported
01h: Extended ATQB is supported

When this byte = 01h, the information “Extended ATQB supported” is transmitted to the ISO14443-B contactless card.

Allowed bit rates

This byte defines the list of bit rates allowed for the contactless interface to perform a PPS (ISO14443-A card) or an ATTRIB command (ISO14443-B card).

The reader will select the highest allowed bit rate that is also supported by the card.

A PPS will be executed only if the ISO14443-A card is compliant to ISO level 4 and if the selected bit rate is higher than 106 kbps.

b7	b6	b5	b4	b3	b2	b1	b0	Bit rate
x	x	x	x	0	x	x	1	212 kbps reader to card allowed
x	x	x	x	0	x	1	x	424 kbps reader to card allowed
x	x	x	x	0	1	x	x	848 kbps reader to card allowed
x	x	x	1	0	x	x	x	212 kbps card to reader allowed

Prox-DU & Prox-SU

x	x	1	x	0	x	x	x	424 kbps card to reader allowed
x	1	x	x	0	x	x	x	848 kbps card to reader allowed
1	x	x	x	x	x	x	x	Only the same bit rate for both direction is allowed
0	x	x	x	x	x	x	x	Different bit rate for both direction is allowed (do not used this setting)

Note: The list of bit rate reader to card (b2-b0) and card to reader (b6-b4) can be different.

T=CL card presence check behavior

This byte is used only for test purpose:

00h: the first dummy APDU command to check a T=CL card presence is send (normal behavior)

01h: the first dummy APDU command to check a T=CL card presence is not used

Note: the first APDU command has the following format: 00h A4h 00h 00h 00h

Card type polling enable/disable

This byte is used to inhibit the reader to poll for a specific card type:

b0 is used to stop the polling of ISO14443-A smart cards (when set to 1)

b1 is used to stop the polling of ISO14443-B smart cards (when set to 1)

b7 is used to perform a RF reset before each REQ command (when set to 1)

b7	b6	b5	b4	b3	b2	b1	b0	Bit rate
RFU	RFU	RFU	RFU	RFU	RFU	X	1	Type A card polling is disabled
RFU	RFU	RFU	RFU	RFU	RFU	1	X	Type B card polling is disabled
0	RFU	RFU	RFU	RFU	RFU	X	X	A RF reset is not performed before each REQ command
1	RFU	RFU	RFU	RFU	RFU	X	X	A RF reset is performed before each REQ command

RFU bit must be set to 0.

Deactivation / Reactivation behavior

This byte is used to control the behavior when the T=CL card is deactivated and reactivated.

b7	b6	b5	b4	b3	b2	b1	b0	Behavior
RFU	RFU	RFU	RFU	RFU	RFU	0	0	Deactivation with a Deselect The card serial number is checked on reactivation
RFU	RFU	RFU	RFU	RFU	RFU	0	1	Deactivation with a Deselect The card serial number is not checked on reactivation
RFU	RFU	RFU	RFU	RFU	RFU	1	0	Deactivation with a Deselect and a RF reset The card serial number is not checked on reactivation
RFU	RFU	RFU	RFU	RFU	RDU	1	1	Deactivation with a RF reset The card serial number is not checked on reactivation

RFU bit must be set to 0

Miscellaneous parameters

Overwrite FWI

This byte is used to force the FWI parameter used for T=CL card communication instead of the card parameters.

FFh: The card parameters is used

0Xh: X = 0h to Eh (Fh is RFU). The value X is used for FWI parameters (as defined in the ISO14443 standard)

Other values are reserved for future use.

Other bytes

These bytes are reserved for test purpose. Do not modify them.

General parameters

RF Reset time

This byte defines the time while the RF field is turn off for a RF reset:

00h : default value is used

01h to FFh : time = 5 to 1275 ms (unit = 5 ms)

RF On Delay

This byte defines the delay for the first card command after the RF field is turn on:

00h : default value is used

01h to FFh : time = 5 to 1275 ms (unit = 5 ms)

RF Parameters Usage

This byte defines the usage of the RF Parameters for ISO14443-A and ISO14443-B:

00h: default values are used

01h: user defined values in RF parameters for type A and Type B card are used.
(RxThreshold, RFCfg, TypeB)

Note: The user defined values must be used only for tuning purpose.

RF Power Attenuation

The byte defines the RF power Attenuation or the user defined values for the antenna output driver conductance:

00h: 0 dB

01h: -1 dB

02h: -2 dB

03h: -3 dB

0Fh: user defined values for the antenna output driver conductance are used.
(GsN, CWGsP and ModGsP)

Note: The user defined values must be used only for tuning purpose.

RF ISO level 2 control for BSI analog tests

This byte is only use to perform the analogue tests for the BSI certification.

00h : ISO normal behavior

X0h : force bit rate = 106 kbps

X1h : force bit rate = 212 kbps

X2h : force bit rate = 424 kbps

X3h : force bit rate = 848 kbps

Note: The user defined values must be used only for certification test purpose.

RF parameters for ISO14443-A cards

These parameters are used only for tuning purpose to communicate with an ISO14443-A card.

RxThreshold, RFCfg, TypeB, GsN, CWGsP and ModGsP parameters are considered only when RF Parameters Usage = 01h **and** RF Power Attenuation = 0Fh.

To configure these parameters refer to the MFRC523 contactless controller IC documentation.

RF parameters for ISO14443-B cards

These parameters are used only for tuning purpose to communicate with an ISO14443-B card.

RxThreshold, RFCfg, TypeB, GsN, CWGsP and ModGsP parameters are considered only when RF Parameters Usage = 01h **and** RF Power Attenuation = 0Fh.

To configure these parameters refer to the MFRC523 contactless controller IC documentation

EEPROM Parameters Validity

CRC control

This byte controls the EEPROM parameters validity:

MAD CRC algorithm is used to compute the CRC value from offset 0 to 64.

At startup, if the EEPROM parameters validity is not correct, all the parameters are set to their default value.

That CRC should be updated at each modification into the EEPROM.

MAD CRC calculation program

The following lines are an example of MAD CRC calculation program:

```
/* *****  
* Prototype : unsigned char ucMadCrc( unsigned char _uc_len,  
*           unsigned char *_puc_in,  
*           unsigned char *_puc_out );  
* Description : This is the function to calculate the CRC  
* If the last byte on input is the supposed CRC of the preceding bytes :  
* the result will be 0 if this CRC is correct.  
* Parameters :  
* unsigned char _uc_len - number of bytes to compute CRC  
* unsigned char *_puc_in - pointer to first byte  
* unsigned char *_puc_out - pointer to store CRC computed  
* Response :  
* ERR_OK compute CRC is OK  
* ERR_MAD_CRC compute CRC is not OK
```

* (last byte on input must be the supposed CRC of the preceding bytes)

Globals :

Ressources

(Use) :

(Modify):

(Call) Internals :

Externals :

*Remarks :

*****/

```

unsigned char ucMadCrc( unsigned char _uc_len,
                        unsigned char *_puc_in,
                        unsigned char *_puc_out )
{
    unsigned char u_i;
    unsigned char u_j;
    unsigned char uc_status;

    uc_status = ERR_OK;

    *_puc_out = 0xC7; // bit-swapped 0xE3

    for (u_j = 0; u_j < _uc_len; u_j++)
    {
        *_puc_out = *_puc_out ^ _puc_in[u_j];

        for (u_i = 0; u_i < 8; u_i++)
        {
            if (*_puc_out & 0x80)
            {
                *_puc_out = (*_puc_out << 1) ^ 0x1D;
            }
            else
            {
                *_puc_out = *_puc_out << 1;
            }
        }
    }
    if (*_puc_out)
    {
        uc_status = ERR_MAD_CRC;
    }

    return (uc_status); // 0x00 if last byte is the CRC of the previous bytes
}

```

Using PC/SC application

PC/SC Overview

The PC/SC specification describes the minimum functionality required of smart cards, smart card readers, and PCs to allow interoperability among compliant elements as provided by a variety of vendors.

The specification as a whole seeks to achieve the following objectives:

- Maintain consistency with existing smart card-related and PC-related standards while expanding upon them where necessary and practical.
- Enable interoperability among components running on various platforms (platform neutral).
- Enable applications to take advantage of products and components from multiple manufacturers (vendor neutral).
- Enable the use of advances in technology without rewriting application-level software (application neutral).
- Facilitate the development of standards for application-level interfaces to smart card services in order to enhance the fielding of a broad range of smart card-based applications in the PC environment.
- Support an environment that encourages the widest possible use of smart cards as an adjunct to the PC environment.

The next figure shows the PC/SC architecture:

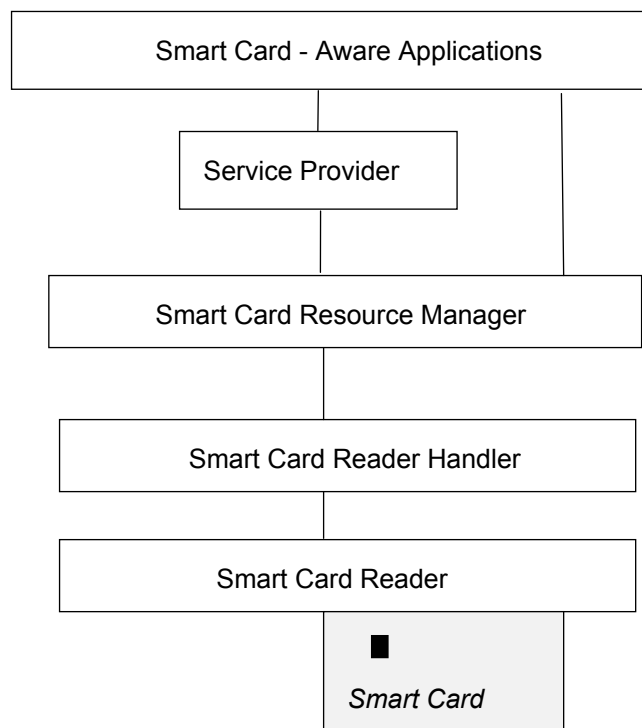


Figure 27 – PC/SC Architecture

Prox-DU & Prox-SU

The Prox-DU and the Prox-SU devices were designed to be fully compliant with the latest PC/SC specification.

The software design considerations presented in the PC/SC specification address the development of applications built on the architecture presented in the figure below.

This paragraph describes the way Smart Card-aware applications can use the functionality provided by the smart card. By using the Smart Card Resource Manager and the Service Provider layers, an application can use smart card functionality with some level of independence from a specific reader, or to some extent, from a specific smart card.

Standard PC/SC functions are listed the following tables:

Smart Card Database Query Functions: Query the smart card database	
SCardGetProviderId	Retrieve the identifier (GUID) of the primary service provider for the given card
SCardListCards	Retrieve a list of cards previously introduced to the system by a specific user
SCardListInterfaces	Retrieve the identifiers (GUIDs) of the interfaces supplied by a given card
SCardListReaderGroups	Retrieve a list of reader groups that have previously been introduced to the system
SCardListReaders	Retrieve the list of readers within a set of named reader groups

Table 7 – Smart Card Database Query Functions

Smart Card Database Management Functions: Manage the smart card database	
SCardAddReaderToGroup	Add a reader to a reader group
SCardForgetCardType	Remove a smart card from the system
SCardForgetReader	Remove a reader from the system
SCardForgetReaderGroup	Remove a reader group from the system
SCardIntroduceCardType	Introduce a new card to the system
SCardIntroduceReader	Introduce a new reader to the system
SCardIntroduceReaderGroup	Introduce a new reader group to the system
SCardRemoveReaderFromGroup	Remove a reader from a reader group

Table 8 – Smart Card Database Management Functions

Resource Manager Context Functions: Manage the context of the resource manager's database operations	
SCardEstablishContext	Establishes a context for accessing the smart card database
SCardReleaseContext	Closes an established context

Table 9 – Resource Manager Context Functions

Prox-DU & Prox-SU

Resource Manager Support Function: Release allocated memory	
SCardFreeMemory	Release memory returned through the use of SCARD_AUTOALLOCATE

Table 10 – Resource Manager Support Function

Smart Card Tracking Functions: Track smart cards within readers	
SCardLocateCards	Search for a card whose ATR string matches a supplied card name
SCardGetStatusChange	Block execution until the current availability of cards changes
SCardCancel	Terminate outstanding actions

Table 11 – Smart Card Tracking Functions

Smart Card and Reader Access Functions: Connect to and communicate with a smart card, including transferring data using <i>T=0</i> , <i>T=1</i> , and raw protocols	
SCardConnect	Connect to a card
SCardReconnect	Reestablish a connection
SCardDisconnect	Terminate a connection
SCardBeginTransaction	Start a transaction, blocking other applications from accessing a card
SCardEndTransaction	End a transaction, allowing other applications to access a card
SCardStatus	Provide the current status of the reader
SCardTransmit	Requests service and receives data back from a card using <i>T=0</i> , <i>T=1</i> , and raw protocols

Table 12 – Smart Card and Reader Access Functions

Direct Card Access Functions: Communicate with cards that may not conform to the ISO 7816 specifications	
SCardControl	Provide direct control of the reader
SCardGetAttrib	Get reader attributes
SCardSetAttrib	Set reader attribute

Table 13 – Direct Card Access Functions

To get more information about these commands please refer to the “*Prox-DU & Prox-SU PC/SC Guide*”.

Gem_PC/SC software tool

The Gemalto Gem_PCSC tool may help to become familiar with the PC/SC environment:



Figure 28 – Gem_PCSC window

The Gem_PCSC tool is available for download in the Gemalto support website: <http://support.gemalto.com>.

Prox-DU and Prox-SU PC/SC reader name

The Prox-DU and the Prox-SU device will be recognized using their PC/SC name. The string name depends on the operating system.

Windows operating systems

The name will comply with the following string format:

- “Gemalto Prox-DU Contactless_xxxxxxxx N1” for the Prox-DU contactless interface
- “Gemalto Prox-DU Contact_xxxxxxxx N2” for the Prox-DU contact interface
- “Gemalto Prox-SU Contactless_yyyyyyyy N3” for the Prox-SU contactless interface
- “Gemalto Prox-SU Contact_yyyyyyyy N4” for the Prox-SU contact interface

N1, N2, N3, N4 are numbers delivered by the computer. xxxxxxxx or yyyyyyyy is the 8-byte reader/writer’s serial number printed on the label located on the rear cabinet.

The next figure displays the name for one Prox-DU connected to the computer:

```
Gemalto Prox-DU Contactless_xxxxxxxx 0
Gemalto Prox-DU Contact_xxxxxxxx 1
```

Figure 29 – Prox-DU PC/SC name (Windows)

The next figure displays the name for one Prox-SU connected to the computer:

```
Gemalto Prox-SU Contactless_xxxxxxxx 0
Gemalto Prox-SU Contact_xxxxxxxx 1
```

Figure 30 – Prox-SU PC/SC name (Windows)

The next figure displays the name for one Prox-DU and one Prox-SU both connected to the computer:

```
Gemalto Prox-DU Contactless_xxxxxxxx 0
Gemalto Prox-DU Contact_xxxxxxxx 1
Gemalto Prox-SU Contactless_yyyyyyyy 2
Gemalto Prox-SU Contact_yyyyyyyy 3
```

Figure 31 – Prox-DU and Prox-SU PC/SC names (Windows)

The next figure displays the name for two Prox-DU devices both connected to the computer:

```
Gemalto Prox-DU Contactless_xxxxxxxx 0
Gemalto Prox-DU Contact_xxxxxxxx 1
Gemalto Prox-DU Contactless_yyyyyyyy 2
Gemalto Prox-DU Contact_yyyyyyyy 3
```

Figure 32 – Two Prox-DU PC/SC names (Windows)

The two first names belong to the first Prox-DU device. The two next names belong to the second Prox-DU device.

Note: The application should use the name of the device for connecting the appropriate smart card interface.

Linux and Mac OS X operating systems

The name will comply with the following string format:

- “Gemalto Prox-DU (xxxxxxx) N1 00” for the Prox-DU contactless interface
- “Gemalto Prox-DU (xxxxxxx) N1 01” for the Prox-DU contact interface
- “Gemalto Prox-SU (yyyyyyy) N2 00” for the Prox-SU contactless interface
- “Gemalto Prox-SU (yyyyyyy) N2 01” for the Prox-SU contact interface

N1, N2 are numbers delivered by the computer. xxxxxxx or yyyyyy is the 8-byte reader/writer’s serial number printed on the label located on the rear cabinet.

The next figure displays the name for one Prox-DU connected to the computer:

```
Gemalto Prox-DU (xxxxxxx) 00 00
Gemalto Prox-DU (xxxxxxx) 00 01
```

Figure 33 – Prox-DU PC/SC name (Linux)

The next figure displays the name for one Prox-SU connected to the computer:

```
Gemalto Prox-SU (xxxxxxx) 00 00
Gemalto Prox-SU (xxxxxxx) 00 01
```

Figure 34 – Prox-SU PC/SC name (Linux)

The next figure displays the name for one Prox-DU and one Prox-SU both connected to the computer:

```
Gemalto Prox-DU (xxxxxxx) 00 00
Gemalto Prox-DU (xxxxxxx) 00 01
Gemalto Prox-SU (yyyyyyy) 01 00
Gemalto Prox-SU (yyyyyyy) 01 01
```

Figure 35 – Prox-DU and Prox-SU PC/SC names (Linux)

The next figure displays the name for two Prox-DU devices both connected to the computer:

```
Gemalto Prox-DU (xxxxxxx) 00 00
Gemalto Prox-DU (xxxxxxx) 00 01
Gemalto Prox-DU (yyyyyyy) 01 00
Gemalto Prox-DU (yyyyyyy) 01 01
```

Figure 36 – Two Prox-DU PC/SC names (Linux)

The two first names belong to the first Prox-DU device. The two next names belong to the

second Prox-DU device.

Note: The application should use the name of the device for connecting the appropriate smart card interface.

PC/SC limitations

The Prox-DU and the Prox-SU devices have the following limitations:

- The contactless interface will only support the T=1 protocol.

Consequently any connection requiring the T=0 protocol will not be accepted by the contactless interface.

- Multi-activation of contactless smart cards is not supported.

Consequently the first smart card detected in front of the reader/writer will be activated. The remaining smart cards will be ignored.

- The communication with the contactless interface and the contact interface shall be exclusive.

Consequently the application shall not use the two interfaces simultaneously. Else communication errors can occur.

For more information about the known issues and limitations please refer to the “Prox-DU and Prox-SU Release Notes” document.

Interfacing with Contactless Cards

As defined in the PC/SC V2.0 specifications, the Prox-DU and the Prox-SU devices handle all the ISO7816-4 Inter Industry commands to interface ISO14443 contactless smart cards.

The Prox-DU and the Prox-SU devices support both type ISO14443-A and ISO14443-B cards.

In addition the Prox-DU and the Prox-SU devices will poll the field for the following smart card events:

- Insertion
- Removal

Detecting an insertion

The contactless reader/writer periodically sends out commands to poll the RF field. If a smart card comes within the range of the RF field, the contactless reader/writer detects the card and activates it.

When the card is activated, its properties are recorded and an insertion event is generated.

The ISO14443 contactless smart card will be activated using the reader parameters stored into the device's configuration EEPROM.

ISO14443-A and ISO14443-B cards are polled with a default periodic rate of 100 ms.

Note: Multi-activation of contactless smart cards is not supported by the Prox-DU and the Prox-SU devices. The first smart card detected in front of the reader/writer will be activated.

When a smart card insertion is detected, a CCID insertion notification message will be generated and the blue LED of the contactless reader/writer will be set to an enlightened steady state.

Detecting a removal

A smart card being removed from the field is detected by the contactless reader/writer.

The contactless reader/writer polls for an ISO14443-3 (MIFARE[®]) smart card by periodically accessing the smart card during periods when there is no communication between the reader/writer and the card.

The contactless reader/writer polls for an ISO14443-4 (T=CL) smart card by periodically sending negative acknowledge frames to the smart card expecting, either a positive acknowledge or the last I-block to be repeated (according to the ISO14443-4 standards).

When a smart card removal is detected, a CCID removal notification message will be generated and the blue LED of the contactless reader/writer will blink slowly.

ATR for contactless smart cards

The Answer To Request (ATR) returned by an ISO14443-A or ISO14443-B smart card is compliant with the PC/SC V2.0 Part 3 Revision 2.01.09 specifications.

The ATR is as follows:

Byte Number	Value	Designation	Description
0	3Bh	Initial header	
1	8Nh	T0	<ul style="list-style-type: none"> Higher nibble 8 means no TA1, TB1, TC1 only TD1 is following. Lower nibble N is the number of historical bytes (HistByte 0 to HistByte N-1).
2	80h	TD1	<ul style="list-style-type: none"> Higher nibble 8 means no TA2, TB2, TC2 only TD2 is following. Lower nibble 0 means T = 0.
3	01h	TD2	<ul style="list-style-type: none"> Higher nibble 0 means no TA3, TB3, TC3, TD3 following. Lower nibble 1 means T = 1.
4 to 3+N	XX XX XX	T1 Tk	Historical bytes: <ul style="list-style-type: none"> ISO14443A: The historical bytes from ATS response. Refer to the ISO14443-4 specification ISO14443B: Byte 1-4: Application Data from ATQB Byte 5-7: Protocol Info Byte from ATQB Byte 8: Higher nibble = MBLI from ATTRIB command. Lower nibble (RFU) = 0 Refer to the ISO14443-3 specification
4 + N	UU	TCK	Exclusive-OR of bytes T0 to Tk

Table 14 – ATR for contactless Smart cards

The contactless smart card exposes its ATS or information bytes not directly, but via a specific ATR mapping. For those cards that provide such information, optionally with Historical Bytes (or Application Information respectively), the mapping in the table above applies.

- The ATR returned by a DESFire smart card is:

3Bh 8Fh 80h 01h 80h 80h 65h B0h 07h 02h 02h 89h 83h 00h 90h 00h 00h 00h 00h 46h

With:

n = Fh (15 historical bytes)

Historical bytes from the ATS response = 80h 80h 65h B0h 07h 02h 02h 89h 83h 00h 90h 00h 00h 00h 00h

UU = 46h (TCK)

- The ATR returned by a GemCombi Xpresso Lite R2 STD smart card will be:

3Bh 8Bh 80h 01h 80h 31h 80h 65h B0h 07h 02h 02h 89h 83h 00h E3h

With:

n = Bh (11 historical bytes)

Historical bytes from the ATS response = 80h 31h 80h 65h B0h 07h 02h 02h 89h
83h 00h

UU = E3h (TCK)

- The ATR returned by a GemCombi CDLite smart card will be:

3Bh 80h 80h 01h 01h

With:

n = 0h (no historical byte)

UU = 01h (TCK)

Interfacing with MIFARE[®] DESFire Cards

The MIFARE[®] DESFire smart card is based on open global standards for both air interface and cryptographic methods. It is compliant to all 4 levels of ISO14443-A and uses optional ISO 7816-4 commands.

The native MIFARE[®] DESFire commands are non ISO7816-4 commands. A proprietary APDU command is implemented into the Prox-DU and Prox-SU reader/writer in order to send and receive these native commands.

Warning: Note that the EEPROM parameter byte “T=CL card presence check behavior” should be set to value “01h” to avoid the reader to send a dummy APDU command during the selection process, because when the MIFARE[®] DESFire smart card receives an APDU command after the selection process the native commands are no more available.

The command is formatted as follows:

CLA	INS	P1	P2	Lc	Data In
FFh	DEh	00h	00h	N	DESFire Command
1 byte	1 byte	1 byte	1 byte	1 byte	N bytes

The response is formatted as follows:

Data Out	[SW1	SW2]
M bytes	1 byte	1 byte

Where:

N	Length of the Data In field	Length of the native command
Data In	DESFire native command	Refer to the DESFire datasheet
Data Out	DESFire native response	Refer to the DESFire datasheet
[SW1-SW2]	Command execution status	Optional field: SW1 SW2 = 90h 00h is added by the reader when the DESFire smart card response is only one byte Status.
	Command executed successfully	90h 00h
	Others	
	67h 00h	Wrong length
	6Ah 81h	Function not supported
	6Bh 00h	Wrong P1 or P2

As an example, to get the version of the DESFire smart card, the following native command should be send: 60h

The proprietary command to consider is the following:

FFh DEh 00h 00h 01h 60h

The response will be:

Prox-DU & Prox-SU

AFh 04h 01h 01h 00h 02h 18h 05h (example)

Refer to the DESFire datasheet for more information about the response.

Requesting contactless smart card information

This proprietary APDU command is used to retrieve the contactless smart card parameters returned by the smart card during the contactless selection process.

The command is formatted as follows:

CLA	INS	P1	P2	Lc	Data In	Le
FFh	FCh	Param	00h	-	-	00h
1 byte	1 byte	1 byte	1 byte	-	-	1 byte

The response is formatted as follows:

Data Out	SW1	SW2
M bytes	1 byte	1 byte

Where:

Param	ISO14443-A or ISO14443-B requested information	00h : ATQA + SN + SAK 01h : complete ATS 02h : ATQB 03h :complete ATTRIB response
Data Out	Requested information	Refer to the ISO14443 standard
SW1-SW2	Command execution status	added by the reader
	Command executed successfully	90h 00h
	Others	
	67h 00h	Wrong length
	6Bh 00h	Wrong P1 or P2
	6Ch xxh	Wrong length (XX is required)
	62h 82h	End of data reach before Le bytes

Note: When the requested information does not correspond to the current smart card type (ISO14443-A or ISO14443-B) an error is reported.
For ISO14443-A3 smart cards, the ATS field is empty.

As an example, to get the ATQA + SN + SAK of the DESFire smart card, the proprietary command to consider is the following:

FFh FCh 01h 00h 00h

The response will be:

44h 03h 04h 26h 47h 09h 48h E8h 10h 20h 90h 00h (example)

ATQA = 44h 03h 04h

SN = 26h 47h 09h 48h E8h 10h (7 bytes)

SAK = 20h

Refer to the DESFire datasheet for more information about the response.

Interfacing with MIFARE[®] Cards

As defined in PC/SC V2.0 Part 3 Revision 2.01.09 specifications, the Prox-DU and the Prox-SU devices perform the appropriate mapping for memory smart card commands that consist of Inter Industry commands (and the exposed data structures) to memory card commands (and the associated data structures defined for the MIFARE[®] contactless memory smart cards).

The Prox-DU and the Prox-SU devices will handle the following ISO7816-4 Inter Industry commands to interface with MIFARE[®] 1K, MIFARE[®] 4K, MIFARE[®] Ultralight and MIFARE[®] Mini memory smart cards:

- **Get Data:** retrieves the UID or the historical bytes of the ATS of the inserted smart card.
- **Load Keys:** Load MIFARE[®] secret into the contactless reader/writer.
- **General Authenticate:** Perform an authentication between the contactless reader/writer and the MIFARE[®] memory smart cards.
- **Read Binary:** Read data from the MIFARE[®] memory smart cards.
- **Update Binary:** Write data to the MIFARE[®] memory smart cards.

The MIFARE[®] 1K is a 8-Kbit (1 Kbyte) MIFARE[®] memory contactless smart card arranged as 64 memory blocks as shown in the appendix “MIFARE[®] cards mapping”.

The MIFARE[®] 4K is a 32-Kbit (4 Kbytes) MIFARE[®] memory contactless smart card arranged as 256 memory blocks as shown in the appendix “MIFARE[®] cards mapping”.

The MIFARE[®] Ultralight is a 512-bit (64 bytes) MIFARE[®] memory contactless smart card arranged as 16 memory pages as shown in the appendix “MIFARE[®] cards mapping”.

The MIFARE[®] Mini is a 2.5-Kbit (320 bytes) MIFARE[®] memory contactless smart card arranged as 20 memory blocks as shown in the appendix “MIFARE[®] cards mapping”.

Important note regarding contactless smart cards including both MIFARE[®] and ISO14443-A4 (T=CL) modes:

When the smart card is connected, the ISO14443-A4 (T=CL) mode will be selected. The corresponding ATR will be returned.

When a MIFARE[®] command is send to the smart card an automatic switch to the MIFARE[®] mode is done and the command will be processed accordingly.

When an ISO14443-A4 (T=CL) command is send to the smart card an automatic switch to the ISO14443-A4 (T=CL) mode is done and the command will be processed accordingly.

When the smart card is in the MIFARE[®] mode, the only way to retrieve the MIFARE[®] type (1K-4K-UL-Mini) is to reconnect the smart card. The appropriate MIFARE[®] ATR will then be returned.

ATR for MIFARE[®] cards

The Answer To Request (ATR) returned by a MIFARE[®] card is compliant with PC/SC V2.0 Part 3 Revision 2.01.09 specifications.

The ATR will be as follows:

Byte Number	Value	Designation	Description		
0	3Bh	Initial header			
1	8Nh	T0	<ul style="list-style-type: none"> Higher nibble 8 means no TA1, TB1, TC1 only TD1 is following. Lower nibble N is the number of historical bytes (HistByte 0 to HistByte N-1). 		
2	80h	TD1	<ul style="list-style-type: none"> Higher nibble 8 means no TA2, TB2, TC2 only TD2 is following. Lower nibble 0 means T = 0. 		
3	01h	TD2	<ul style="list-style-type: none"> Higher nibble 0 means no TA3, TB3, TC3, TD3 following. Lower nibble 1 means T = 1. 		
4 to 2+n	80h	T1	Category indicator byte, 80h means a status indicator may be present in an optional COMPACT-TLV data object.		
	4Fh		Application identifier presence indicator		
	LL		Length		
	A0h 00h 00h 03h 06h		5 bytes for registered application provider identifier (RID)		
	SS		1 byte for Standard		
	NN NN		2 bytes for Card Name		
	00h 00h 00h 00h		RFU: Shall be set to zero. Assigned by PC/SC for future extensions.		
	3 + n		UU	TCK	Exclusive-OR of bytes T0 to Tk

Table 15 – ATR for MIFARE[®] cards

The ATR of a contactless storage card is structured in the manner described in the table above. In order to allow the application to identify a storage card type properly, its Standard and Card Name bytes must be interpreted according to the following tables:

b7	b6	b5	b4	b3	b2	b1	b0	Description
0	0	0	0	0	0	0	0	No information given
0	0	0	0	0	0	1	1	ISO14443-A, part 3
0	0	0	0	0	1	0	0	ISO14443-A, part 4
...								RFU
1	1	1	1	1	1	1	1	RFU

Table 16 – SS Byte for Standard

Card Name	Two bytes identifier
MIFARE [®] Standard 1K	00h 01h
MIFARE [®] Standard 4K	00h 02h
MIFARE [®] Ultralight	00h 03h
MIFARE [®] Mini	00h 26h

Table 17 – NN Bytes for Card Name

- The ATR returned by a MIFARE[®] Standard 1K will be:

3Bh 8Fh 80h 01h 80h 4Fh 0Ch A0h 00h 00h 03h 06h 03h 00h 01h 00h 00h 00h 00h 6Ah

With:

LL = 0Ch (12 bytes)

SS = 03h (ISO14443-A, part 3)

NN NN = 00h 01h (MIFARE[®] Standard 1K)

UU = 6Ah (TCK)
- The ATR returned by a MIFARE[®] Standard 4K will be:

3Bh 8Fh 80h 01h 80h 4Fh 0Ch A0h 00h 00h 03h 06h 03h 00h 02h 00h 00h 00h 00h 69h

With:

LL = 0Ch (12 bytes)

SS = 03h (ISO14443-A, part 3)

NN NN = 00h 02h (MIFARE[®] Standard 4K)

UU = 69h (TCK)
- The ATR returned by a MIFARE[®] Ultralight will be:

3Bh 8Fh 80h 01h 80h 4Fh 0Ch A0h 00h 00h 03h 06h 03h 00h 03h 00h 00h 00h 00h 68h

With:

LL = 0Ch (12 bytes)

SS = 03h (ISO14443-A, part 3)

NN NN = 00h 03h (MIFARE® Ultralight)

UU = 68h (TCK)

- The ATR returned by a MIFARE® Mini will be:

3Bh 8Fh 80h 01h 80h 4Fh 0Ch A0h 00h 00h 03h 06h 03h 00h 26h 00h 00h 00h 00h 4Dh

With:

LL = 0Ch (12 bytes)

SS = 03h (ISO14443-A, part 3)

NN NN = 00h 26h (MIFARE® Mini)

UU = 4Dh (TCK)

Get Data command

This command is used to retrieve information about the inserted smart card. This command can be used for all kinds of contactless cards.

The command is formatted as follows:

CLA	INS	P1	P2	Lc	Data	Le
FFh	CAh	INF	00h	-	-	NN
1 byte	1 byte	1 byte	1 byte	-	-	1 byte

The response is formatted as follows:

Data	SW1	SW2
NN bytes	1 byte	1 byte

Where:

INF Info type

INF = 00h means: Card serial number (UID or PUPI) is returned
 INF = 01h means: All historical bytes from the ATS of a ISO14443 A card without CRC are returned

NN Expected length of the data

Prox-DU & Prox-SU

Data	<p>Serial Number</p> <p>For ISO14443-A smart cards: UID0-UID1-UID2-UID3 or UID0-UID1-UID2-UID3-UID4-UID5- UID6 or UID0-UID1-UID2-UID3-UID4-UID5- UID6-UID7-UID8-UID9</p> <p>For ISO14443-B smart cards: PUPI3-PUPI2-PUPI1-PUPI0</p>	<p>NN = 00h means: Return full length of the UID (e.g. for ISO14443-A single 4 bytes, double 7 bytes, triple 10 bytes, for ISO14443-B 4 bytes PUPI)</p> <p>The UID is exposed as a string of the expected length. If the expected length is greater than the actual length the rest of the string is filled with zero-value padding bytes. No cast must be done over the UID or parts of it. For example, casting four bytes of the UID to a 32-bit Integer is illegal.</p> <p>The order of the bytes within the string matches the order of bytes received from the card during the anti-collision process. Consequently, the first byte received will be at index zero.</p> <p>The bit order of the string bytes must be such that the LSB (MSB) matches with the LSB (MSB) of the card-defined UID.</p> <p>Refer to ISO14443-A standard.</p>
	<p>Historical bytes of the ATS</p>	<p>For a MIFARE® or ISO14443-B card that command is not supported.</p>
SW1-SW2	<p>Command execution status</p> <p>Command executed successfully 90h 00h</p> <p>Others Refer to the error codes table below</p>	

Load Keys command

This command is used to load the MIFARE® secret keys into the contactless reader/writer.

Up to 160 keys can be loaded to support all the keys pairs needed for the Mifare 4K cards (2 keys for each sector):

- 80 keys stored in the reader/writer's EEPROM
- 80 keys stored in the reader/writer's RAM

The command is formatted as follows:

CLA	INS	P1	P2	Lc	Data
FFh	82h	KS	KN	KL	Key
1 byte	1 byte	1 byte	1 byte	1 byte	6 bytes

If the Load MIFARE® key security bit is set to one in the configuration EEPROM, a Transport secret key should be added to the MIFARE® key:

The command is formatted as follows:

Prox-DU & Prox-SU

CLA	INS	P1	P2	Lc	Data
FFh	82h	KS	KN	KL	Key
1 byte	1 byte	1 byte	1 byte	1 byte	12 bytes

The response is formatted as follows:

SW1	SW2
1 byte	1 byte

Where:

KS	Key Structure	00h for key location storage in RAM 20h for key location storage in EEPROM
KN	Key Number	160 keys are available MIFARE® Key Number 0 to 159 (00h to 9Fh) The key number which will be used for the authentication The key number 0 to 79 (00h to 4Fh) are reserved for the non volatile key stored in EEPROM The key number 80 to 159 (50h to 9Fh) are reserved for the volatile key stored in RAM
KL	Key Length	KL = 06h means: 6 bytes long KL = 0Ch means: 12 bytes long if the Load MIFARE® key security bit is set on.
Key	MIFARE® Secret Key	The MIFARE® key value Should be followed by the Gemalto Transport key if the Load MIFARE® key security bit is set on. The byte order must be the same as the byte order in the card sector trailer (A0h first for the key A0h A1h A2h A3h A4h A5h) Gemalto default Key A A0h A1h A2h A3h A4h A5h Gemalto default Key B B0h B1h B2h B3h B4h B5h Gemalto Transport Key T0 T1 T2 T3 T4 T5 Other values User Key

SW1-SW2 Command execution status

Command executed successfully 90h 00h

Others Refer to the error codes table below

Warning: If the Load MIFARE[®] key security bit is set to one, a 6 bytes transport key must be added to the Data field and the total key length must be equal to 12.

The Transport keys are secret and are available upon request.

To load the secret key A0h-A1h-A2h-A3h-A4h-A5h into the key number KN using location storage in RAM the following APDU command should be used:

CLA	INS	P1	P2	Lc	Data
FFh	82h	00h	KN	06h	A0h A1h A2h A3h A4h A5h

With KN = 80 to 159 (50h to 9Fh)

To load the secret key A0h-A1h-A2h-A3h-A4h-A5h into the key number KN using location storage in EEPROM the following APDU command should be used:

CLA	INS	P1	P2	Lc	Data
FFh	82h	20h	KN	06h	A0h A1h A2h A3h A4h A5h

With KN = 0 to 79 (00h to 4Fh)

Note: Loading key number 0 to 79 in RAM is forbidden. Loading key number 80 to 159 in EEPROM is forbidden. The error code SW1-SW2 69h 88h will be returned (Key number not valid)

Note: After delivery the non volatile keys stored in EEPROM (number 0 to 79) are initialized to a default value:

The keys number 00 to 39 are initialized with value A0h A1h A2h A3h A4h A5h

The keys number 40 to 79 are initialized with value B0h B1h B2h B3h B4h B5h

Note: Each time the Prox-DU and the Prox-SU is powered, the volatile keys stored in RAM (number 80 to 159) are initialized to a default value:

The keys number 80 to 119 are initialized with value A0h A1h A2h A3h A4h A5h

The keys number 120 to 159 are initialized with value B0h B1h B2h B3h B4h B5h

General Authenticate command

The General Authenticate command is used to perform an authentication between the contactless reader/writer and a MIFARE[®] memory block.

For MIFARE[®] 1K, MIFARE[®] 4K and MIFARE[®] Mini it is mandatory to perform the General Authenticate command before each read or write memory block operation. Otherwise, an authentication error will occur.

For MIFARE[®] Ultralight the General Authenticate operation is not required.

Prox-DU & Prox-SU

This command is formatted as follows:

CLA	INS	P1	P2	Lc	Data In				
FFh	86h	00h	00h	05h	VER	ABLM	ABLL	KT	KN
1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	1 byte

The response is formatted as follows:

SW1	SW2
1 byte	1 byte

Where:

VER	Version	01h	VER is used in the future to differentiate different version of this command.
ABLM	Address Block MSB	00h	
ABLL	Address Block LSB		
	MIFARE® 1K	00h – 3Fh	
	MIFARE® 4K	00h – FFh	
	MIFARE® Mini	00h – 13h	
KT	Key Type		
	Key A	60h	
	Key B	61h	
KN	Key Number		
	MIFARE® Key Number	0 to 159 (00h to 9Fh)	The key number 0 to 79 are reserved for the non volatile key stored in EEPROM The key number 80 to 159 are reserved for the volatile key stored in RAM
SW1-SW2	Command execution status		
	Command executed successfully	90h 00h	
	Others		Refer to the error codes table below

The authentication is performed for a memory sector. As each memory sector is composed of four memory blocks, the authentication will be done for all the four memory blocks.

The authentication operation is not required for a MIFARE® Ultralight chip.

If an authentication fails, the smart card is automatically deselected (MIFARE® specification). However the reader/writer automatically recovers the communication with the smart card.

Read Binary command

The Read Binary command is used to read data from a MIFARE® memory area.

Data consist of a memory block (16 bytes) or a memory page (4 bytes).

This command is formatted as follows:

CLA	INS	P1	P2	Lc
FFh	B0h	ABLM	ABLL	Size
1 byte	1 byte	1 byte	1 byte	1 byte

The response is formatted as follows:

Data	SW1	SW2
16 bytes	1 byte	1 byte

Where:

ABLM	Address Block MSB	00h
ABLL	Address Block LSB	
	MIFARE® 1K	00h – 3Fh
	MIFARE® 4K	00h – FFh
	MIFARE® Mini	00h – 13h
	MIFARE® Ultralight	00h – 0Fh
Size	Size of the memory area	
	MIFARE® 1K, 4K, Mini	10h (size of the memory block)
	MIFARE® Ultralight	04h (size of the memory page)
		If this parameters is 00h then all the bytes of the block will be returned (16 bytes or 4 bytes)
Data		
	MIFARE® 1K, 4K, Mini	16-byte of data
	MIFARE® Ultralight	4-byte of data
		The first byte of the block is byte 0
		Present only when there is no error in the status report.
SW1-SW2	Command execution status	

Prox-DU & Prox-SU

Command executed successfully 90 00h

Others Refer to the error codes table below

Note:

For MIFARE® 1K, MIFARE® 4K and MIFARE® Mini, it is mandatory to perform the General Authenticate command before each read memory block operation. Otherwise, an authentication error will occur.

For MIFARE® Ultralight the General Authenticate operation is not required. Refer to the appendix for the MIFARE® Ultralight read operation.

Update Binary command

The Update Binary command is used to write data into a MIFARE® memory area.

Data consist of a memory block (16 bytes) or a memory page (4 bytes).

This command is formatted as follows:

CLA	INS	P1	P2	Lc	DATA
FFh	D6h	ABLM	ABLL	Size	Data
1 byte	1 byte	1 byte	1 byte	1 byte	16 bytes

The response is formatted as follows:

SW1	SW2
1 byte	1 byte

Where:

ABLM	Address Block MSB	00h
ABLL	Address Block LSB	
	MIFARE® 1K	00h – 3Fh
	MIFARE® 4K	00h – FFh
	MIFARE® Mini	00h – 13h
	MIFARE® Ultralight	00h – 0Fh (*)
Size	Size of the memory area	
	MIFARE® 1K, 4K, Mini	10h (size of the memory block)
	MIFARE® Ultralight	04h (size of the memory page)
Data		
	MIFARE® 1K, 4K, Mini	16-byte of data
	MIFARE® Ultralight	4-byte of data
		The first byte of the block is byte 0
		Present only when there is no error

in the status report.

SW1-SW2 Command execution status

Command executed successfully 90 00h

Others Refer to the error codes table below

Note:

For MIFARE® 1K, MIFARE® 4K and MIFARE® Mini, it is mandatory to perform the General Authenticate command before each write memory block operation. Otherwise, an authentication error will occur.

For MIFARE® Ultralight the General Authenticate operation is not required. Refer to the appendix for the MIFARE® Ultralight write operation.

Error code list summary

The error codes returned by the commands listed above are defined in the following table:

SW1	SW2	Meaning
Get Data error codes		
62h	82h	End of data reach before Le bytes (Le is greater than data length)
67h	00h	Wrong length
6Ah	81h	Function not supported
6Bh	00h	Wrong parameter P1-P2
6Ch	XXh	Wrong length (wrong number Le; XX is the exact number) if Le is less than the available data length
6Dh	00h	Instruction code not supported
Load Keys error codes		
65h	81h	Memory failure
67h	00h	Wrong length
69h	83h	Reader key not supported
69h	85h	Secure transmission not supported
69h	88h	Key number not valid
69h	89h	Key length is not correct
General Authenticate error codes		
67h	00h	Wrong length
69h	82h	Security status not satisfied
69h	83h	Authentication cannot be done
69h	85h	Secure transmission not supported
69h	86h	Key type not known
69h	88h	Key number not valid
6Ah	81h	Function not supported
6Bh	00h	Wrong parameter P1-P2
6Dh	00h	Instruction code not supported
Read Binary error codes		
62h	82h	End of data reach before Le bytes (Le is greater than data length)

Prox-DU & Prox-SU

67h	00h	Wrong length
68h	00h	Class byte is not correct
69h	82h	Security not satisfied
69h	85h	Address out of range
6Ah	81h	Function not supported
6Ch	XX	Wrong length (wrong number Le; XX is the exact number) if Le is less than the available data length
Update Binary error codes		
67h	00h	Wrong length
69h	82h	Security not satisfied
69h	85h	Address out of range
6Ah	81h	Function not supported

Table 18 – Memory card error codes

Interfacing with Contact Cards

ISO7816 asynchronous smart cards are accessible via standard PC/SC using Microsoft's library "winscard.dll". This type of cards supports at least one of the asynchronous protocols T=0 or T=1. No additional libraries or third-party software components are necessary to integrate ISO7816 smart cards.

As defined in the PC/SC specifications, the Prox-DU and the Prox-SU devices handle all the ISO7816-4 Inter Industry commands to interface ISO7816 asynchronous contact smart cards.

In addition the Prox-DU device will support the following smart card events:

- Insertion
- Removal

As the Prox-SU has no capability to detect a smart card insertion or removal, the SIM/SAM card will always be considered as inserted when the SIM/SAM card is into its connector.

Detecting an Insertion

The contact reader/writer will check if a smart card is inserted into the slot.

When a smart card insertion is detected, its properties are recorded and a CCID insertion notification message will be generated.

Detecting a Removal

A smart card being removed from the slot is detected by the contact reader/writer.

When a smart card removal is detected, a CCID removal notification message will be generated.

ATR for Contact Smart Cards

The Answer To Request (ATR) returned by a contact smart card is compliant with the ISO7816-3 specifications.

The Prox-DU and the Prox-SU will return the smart card ATR after a smart card power up.

The ATR is as follows:

Byte Number	Value	Designation	Description
0	3Bh or 3Fh	TS	Initial header (Mandatory) Direct or inverse convention

Prox-DU & Prox-SU

1	Y1-K	T0	Format character (Mandatory) Encodes Y1 and K Y1 indicator for the presence of the interface characters TA1-TB1-TC1-TD1 K=number of historical bytes
2	Fi-Di	TA1	Interface characters (Optional) Global, encodes Fi and Di
3	XX	TB1	Interface characters (Optional) Global, deprecated
4	N	TC1	Interface characters (Optional) Global, encodes N
5	Y2-T	TD1	Interface characters (Optional) Structtural, encodes Y2 and T
6	XX	TA2	Interface characters (Optional) Global, specific mode byte
7	XX	TB2	Interface characters (Optional) Global, deprecated
8	XX	TC2	Interface characters (Optional) Specific to T=0
9	Y3-T	TD2	Interface characters (Optional) Structural, encodes Y3 and T

- For $i > 2$

	Yi-T	TDi-1	Interface characters (Optional) Structural, encodes Yi and T
	XX	TAi	Interface characters (Optional) Specific to T after T from 0 to 14 in TDi-1 Global after T=15 in TDi-1
	XX	TBi	
	XX	TCi	
	Yi+1-T	TDi	Interface characters (Optional) Interface characters (Optional) Structural, encodes Yi+1 and T

-

	XX	T1	Historical characters (Optional): max 15 bytes
--	----	----	--

	XX	...	
	XX	..	
		Tk	
N	UU	TCK	Check character (Conditional): Exclusive-OR of bytes T0 to Tk

Table 19 – ATR for contact smart cards

Structures and content

A reset operation results in the answer from the smart card consisting of the initial character TS followed by at most 32 characters in the following order:

- T0 Format character (Mandatory)
- T_{Ai}, T_{Bi}, T_{Ci}, T_{Di} Interface characters (Optional)
- T1, T2, ... ,TK Historical characters (Optional)
- TCK Check character (Conditional)

The interface characters specify physical parameters of the integrated circuit in the smart card and logical characteristics of the subsequent exchange protocol.

The historical characters designate general information, for example, the smart card manufacturer, the chip inserted in the smart card, the masked ROM in the chip, the state of the life of the smart card. The specification of the historical characters falls outside the scope of this part of ISO7816.

For simplicity, T0, T_{Ai}, ... ,TCK will designate the bytes as well as the characters in which they are contained.

Structure of the subsequent characters in the ATR

The initial character TS is followed by a variable number of subsequent characters in the following order: The format character T0 and, optionally the interface characters T_{Ai}, T_{Bi}, T_{Ci}, T_{Di} and the historical characters T1, T2, ... , TK and conditionally, the check character TCK.

The presence of the interface characters is indicated by a bit map technique explained below.

The presence of the historical characters is indicated by the number of bytes as specified in the format character defined below.

The presence of the check character TCK depends on the protocol type(s) as defined as below.

Format character T0

The T0 character contains two parts:

- The most significant half byte (b4, b5, b6, b7) is named Y1 and indicates with a logic level ONE the presence of subsequent characters T_{A1}, T_{B1}, T_{C1}, T_{D1} respectively.
- The least significant half byte (b3 to b0) is named K and indicates the number (0 to 15) of historical characters.

b7	b6	b5	b4	b3	b2	b1	b0
----	----	----	----	----	----	----	----

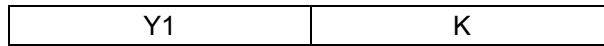


Figure 37 – Information provided by T0

Y1: indicator for the presence of the interface characters

- TA1 is transmitted when b4=1
- TB1 is transmitted when b5=1
- TC1 is transmitted when b6=1
- TD1 is transmitted when b7=1
- K: number of historical characters

Interface characters T_{Ai}, T_{Bi}, T_{Ci}, T_{Di}

T_{Ai}, T_{Bi}, T_{Ci} (i=1, 2, 3, ...) indicate the protocol parameters.

Each interface byte TA, TB or TC is either global or specific:

- Global interface bytes refer to parameters of the integrated circuit within the smart card,
- Specific interface bytes refer to parameters of a transmission protocol offered by the smart card.

T_{Di} indicates the protocol type T and the presence of subsequent characters.

Bits b4, b5, b6, b7 of the byte containing Y_i (T₀ contains Y₁; T_{Di} contains Y_{i+1}) state whether character T_{Ai} for b4, character T_{Bi} for b5, character T_{Ci} for b6, character T_{Di} for b7 are or are not (depending on whether the relevant bit is 1 or 0) transmitted subsequently in this order after the character containing Y_i.

When needed, the interface device shall attribute a default value to information corresponding to a non transmitted interface character.

When T_{Di} is not transmitted, the default value of Y_{i+1} is null, indicating that no further interface characters T_{Ai+j}, T_{Bi+j}, T_{Ci+j}, T_{Di+j} will be transmitted.

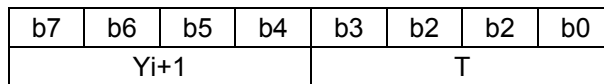


Figure 38 – Information provided by T_{Di}

Y_{i+1}: indicator for the presence of the interface characters

- T_{Ai+1} is transmitted when b4=1
- T_{Bi+1} is transmitted when b5=1
- T_{Ci+1} is transmitted when b6=1
- T_{Di+1} is transmitted when b7=1

T: Protocol type for subsequent transmission.

If T_{D1}, T_{D2} and so on are present, the encoded types T shall be in ascending numerical order. If present, T=0 shall be first, T=15 shall be last. T=15 is invalid in T_{D1}.

Historical characters T₁, T₂, ... , T_K

When K is not null, the answer to reset is continued by transmitting K historical characters T₁, T₂, ... , T_K.

Check character TCK

The value of TCK shall be such that the exclusive-oring of all bytes from T0 to TCK included is null.

Protocol type T

The four least significant bits of any interface character TDi indicate a protocol type T, specifying rules to be used to process transmission protocols. When TDi is not transmitted, T=0 is used.

- T=0 is the asynchronous half duplex character transmission protocol.
- T=1 is the asynchronous half duplex block transmission protocol.
- T=2 and T=3 are reserved for future full duplex operations.
- T=4 is reserved for an enhanced asynchronous half duplex character transmission protocol.
- T=5 to T=13 are reserved for future use.
- T=14 is reserved for protocols not standardized by ISO.
- T=15 does not refer to a transmission protocol, but only qualifies global interface bytes.

Note: If only T=0 is indicated, TCK shall not be sent. In all other cases TCK shall be sent.

Specifications of the global interface bytes

Among the interface bytes possibly transmitted by the smart card in answering to reset, this subclaus defines only the global interface bytes TA1, TB1, TC1, TA2, TB2, the first TA for T=15 and the first TB for T=15.

These global interface bytes convey information to determine parameters which the interface device shall take into account.

TA1

TA1 encodes the indicated value of the clock rate conversion integer (Fi), the indicated value of the baud rate adjustment integer (Di) and the maximum value of the frequency supported by the smart card (f (max.)). The default values are Fi = 372, Di = 1 and f (max.) = 5 MHz.

Fi	0000	0001	0010	0011	0100	0101	0110	0111
F	372	372	558	744	1116	1488	1860	RFU
Fs (max) MHz	4	5	6	8	12	16	20	-

Fi	1000	1001	1010	1011	1100	1101	1110	1111
F	RFU	512	768	1024	1536	2048	RFU	RFU
Fs (max) MHz	-	5	7.5	10	15	20	-	-

Table 20 – Clock rate conversion factor F

Prox-DU & Prox-SU

Di	0000	0001	0010	0011	0100	0101	0110	0111
D	RFU	1	2	4	8	16	32	64

Di	1000	1001	1010	1011	1100	1101	1110	1111
D	12	20	RFU	RFU	RFU	RFU	RFU	RFU

Table 21 – Bit rate adjustment factor D

TB1 and TB2

TB1 and TB2 are deprecated. The smart card should not transmit them. The interface device shall ignore them.

Note: The first two editions of ISO 7816-3 specified TB1 and TB2 to fix electrical parameters of the integrated circuit for the deprecated use of contact C6.

TC1

TC1 encodes the extra guard time integer (N) from 0 to 255 over the eight bits. The default value is N = 0.

If N = 0 to 254, then before being ready to receive the next character, the smart card requires the following delay from the leading edge of the previous character (transmitted by the smart card or the interface device):

$$GT = 12 \text{ etu} + R \times N/f$$

- If T=15 is absent in the Answer-to-Reset, then $R = F / D$, i.e., the integers used for computing the etu.
- If T=15 is present in the Answer-to-Reset, then $R = Fi / Di$, i.e., the integers defined above by TA1.

No extra guard time is used to transmit characters from the card: $GT = 12 \text{ etu}$.

The use of N = 255 is protocol dependent: $GT = 12 \text{ etu}$ in PPS and in T=0. For the use of N = 255 in T=1, refer to the ISO7816-3 standard.

TA2

TA2 is the specific mode byte. For the use of TA2 refer to the ISO7816-3 standard.

Bit 7 indicates the ability for changing the negotiable/specific mode:

- capable to change if bit 7 is set to 0,
- unable to change if bit 7 is set to 1.

Bits 6 and 5 are reserved for future use (set to 0 when not used).

Bit 4 indicates the definition of the parameters F and D.

- If bit 4 is set to 0, then the integers Fi and Di defined above by TA1 shall apply.
- If bit 4 is set to 1, then implicit values (not defined by the interface bytes) shall apply.

Bits 3 to 0 encode a type T.

The first TA1 for T=15

The first TA for T=15 encodes the clock stop indicator (X) and the class indicator (Y). The default values are X = “clock stop not supported” and Y = “only class A supported”. For the use of clock stop and for the use of the classes of operating conditions refer to the ISO7816-3 standard.

- According to the next table, bits 7 and 6 indicate whether the smart card supports clock stop ($\neq 00$) or not ($= 00$) and, when supported, which state is preferred on the electrical circuit CLK when the clock is stopped.

Bits 7 and 6	00	01	10	11
X	Clock stop not supported	State L	State H	No preference

Table 22 – clock stop indicator X

- According to the next table 10, bits 5 to 1 indicate the classes of operating conditions accepted by the smart card. Each bit represents a class: bit 1 for class A, bit 2 for class B and bit 3 for class C.

Bits 5 to 0	00 0001	00 0010	00 0100	00 0011
Y	A only (5V)	B only (3V)	C only (1.8V)	A and B
Bits 5 to 0	00 0110	00 0111	Any other value	
Y	B and C	A, B and C	RFU	

Table 23 – class indicator Y

The first TB for T=15

The first TB for T=15 indicates the use of standard or proprietary use contact (SPU) by the smart card. The default value is “SPU not used”.

Coded over bits 6 to 0, the use is either standard (bit 7 set to 0), or proprietary (bit 7 set to 1). The value '00' indicates that the smart card does not use SPU. Any other value where bit 7 is set to 0 are reserved for future use.

For additional information about the ATR contents please refer to the ISO7816-3 standard.

CCID Devices

The CCID device is a standardized USB smart card reader. The Prox-DU and the Prox-SU reader/writers include two CCID devices:

- One CCID device for the contactless interface
- One CCID device for the contact interface

Each CCID device complies with the standard CCID specification Revision 1.1.

CCID Overview

When a CCID device is connected to a USB host, it may or may not have a smart card inserted. The CCID device identifies to the host its capabilities and requirements, and the host prepares to communicate with it. The CCID device may then, at any time, detect the presence of a smart card, at which time it communicates that information to the host. As soon as the host receives information about the attached smart card, further communications may then take place between the host and the smart card through the CCID device.

The CCID model assumes that a smart card is or can be inserted into the device. This is the purpose for the “slot change” interrupt message.

Also this model applies to devices that integrate CCID and smart card functionalities and may be viewed as containing a permanently inserted smart card.

CCID communication pipes

The CCID device uses the following USB communication pipes:

- A control pipe
 - To monitor the USB device as described in the USB specification
 - To transport 3 class-specific requests
 - ABORT to stop running CCID commands
 - GET_CLOCK_FREQUENCIES to retrieve the list of frequencies for the card clock provided by the reader
 - GET_DATA_RATES to get the baud rates available on the reader
- An interrupt pipe to notify asynchronous events
 - To notify card movements
 - To report hardware problems

This interrupt pipe is mandatory for a CCID that supports card insertion/removal as the Prox-DU, and optional for a CCID with cards that are always inserted and not removable as the Prox-SU.

- A bulk-in and bulk-out pipe
 - The host command is sent on the bulk-out endpoint
 - The device sends the answer on the bulk-in endpoint

The features of the bulk-in and bulk-out pipes implemented into the Prox-DU and the Prox-SU reader/writers are the following:

- The CCID can address one card. Each card is called a 'slot'
- The host can send only one command at a time to a slot
- The host cannot send a new command to a slot until the ending response to the last command to that slot is received
- At each bulk out command sent by the host, there is only bulk-in answer sent by the reader and possibly time extension requested by the card

CCID protocol and parameters selection

A CCID device announces its level of exchanges with the host, TPDU, APDU (Short and Extended), or Character.

The Prox-DU and the Prox-SU reader/writers announce the following level of exchanges:

- **Extended APDU, T=1 for the contactless interface**
- **TPDU, T=0 & T=1 for the contact interface**

Note: the character level of exchanges is not supported by both interfaces.

TPDU level of exchange

For TPDU level exchanges, the CCID provides the transportation of host's TPDU to the smart card's TPDU. The TPDU format changes according to the protocol or for PPS exchange.

TPDU for PPS exchange has the following format:

Command TPDU:

FF PPS0 PPS1 PPS2 PPS3 PCK, with PPS1, PPS2, PPS3 optional.

Response TPDU:

FF PPS0_R PPS1_R PPS2_R PPS3_R PCK_R, with PPS1_R, PPS2_R, PPS3_R optional.

The CCID implements and verifies timings and protocol according to its parameters settings to assume ISO7816-3. No check on frame format is mandatory on request, and on response the only recommended analysis is the most significant nibble of PPS0_R to compute the number of bytes left to receive.

A CCID that implements automatic PPS should not accept TPDU for PPS exchange and must check for PPS response validity.

T = 0 TPDU can have three formats:

- Form 1, no data to exchange with the smart card, only header:

Command TPDU = CLA INS P1 P2, the CCID is responsible to add P3=00h.

Response TPDU = SW1 SW2

- Form 2, data expected from the smart card:

Command TPDU = CLA INS P1 P2 Le, Le=P3 from 00h to FFh (00h means 100h)

Response TPDU = Data(Le) SW1 SW2, Data(Le) is for the Le data

received from the smart card or empty if the smart card rejects the command.

- Form 3, data are to be sent to the smart card:

Command TPDU = CLA INS P1 P2 Lc Data(Lc), Lc=P3 from 01h to FFh and Data(Lc) for the Lc data to send to the smart card.

Response TPDU = SW1 SW2

The CCID device, for T=0 TPDU, is in charge of managing procedure bytes and character level.

The procedure bytes are not mapped into the response TPDU except for the SW1 SW2 bytes. The CCID implements and verifies timings according to its parameters settings to assume ISO7816-3 (work waiting time, extra guard time ...). If the smart card uses NULL procedure byte (60h) the CCID informs the host of this request for time extension.

T = 1 TPDU command and response use the frame format. The CCID expects the respect of the character frame. But no check on frame format is mandatory on sending, and on receiving. The only recommended checks are:

- Expecting LEN byte as third byte
- Wait for LEN bytes as INF field
- Wait for an EDC field which length complies with parameter bmTCKST1

The CCID implements and verifies timing according to its parameters settings to assume ISO7816-3 (CWT, BWT, BGT ...).

The detection of parity error on character received is optional. The interpretation of first bytes received as NAD and PCB to manage VPP is optional and depends on CCID capabilities.

APDU level of exchange

For APDU level exchanges, the CCID provides the transportation of host's APDU to the smart card's TPDU.

APDU commands and responses are defined in ISO7816-4.

Two APDU levels are defined, short APDU and extended APDU. Short APDU and extended APDU are defined in ISO/IEC 7816-4

A CCID that indicates a short APDU exchange only accepts short APDU. A CCID that indicates an extended APDU exchange accepts both short APDU and extended APDU.

If the smart card requests time extension, by using a NULL procedure byte (60h) in T=0 protocol or S(WTX) in T=1 protocol, the CCID informs the host of this request.

A CCID supporting APDU level of exchanges implements a high level of automatism in smart card communications. It shall also provide a high level of automatism in ATR treatment and implement one of the following automatisms: automatic parameters negotiation (proprietary algorithm), or automatic PPS according to the current parameters. At least two standards of transportation for APDU are defined, ISO 7816-4 and EMV, which standard to implement is out of the scope of this specification.

Character level of exchange

Character level of exchanges is selected when none of the TPDU, Short APDU or Short and extended APDU is selected.

The CCID sends the characters in the command (maybe none) then waits for the number of

characters (if not null) indicated in the command.

For character level exchange between the host and the CCID, the CCID supports asynchronous characters communication with the smart card as per ISO7816-3 including timings defined in ISO7816-3 for T = 0 and for T = 1. To respect timing the CCID shall use the defined parameters.

The CCID implements the character frame and character repetition procedure when T = 0 is selected.

Suspend behavior

When resuming from a USB suspend, the host/driver will assume that all the smart cards have been deactivated (powered down).

When the USB bus suspends, CCIDs are not required to deactivate inserted smart cards, but may do so; however, after the USB bus resumes, CCIDs must respond to the host as if all of the inserted smart cards had been deactivated and newly inserted.

After resuming, the CCID will do two things in no particular order.

1. Send the RDR_to_PC_NotifySlotChange message to inform the driver which slots have "newly inserted" cards.
2. The CCID will reactivate the smart cards only from a PC_to_RDR_IccPowerOn message from the driver or automatically if the CCID has the "automatic activation on insertion" feature. Note: When reactivating, all slot parameters initially revert back to the defaults.

CCID device for the contact interface

Command pipe bulk-out message for the contact card interface

The following CCID commands are implemented for the contact interface:

- PC_to_RDR_IccPowerOn
- PC_to_RDR_IccPowerOff
- PC_to_RDR_GetSlotStatus
- PC_to_RDR_XfrBlock
- PC_to_RDR_SetParameters
- PC_to_RDR_GetParameters
- PC_to_RDR_ResetParameters
- PC_to_RDR_Escape
- PC_to_RDR_Abort

The following CCID commands are not implemented:

- PC_to_RDR_IccClock
- PC_to_RDR_T0APDU
- PC_to_RDR_Secure
- PC_to_RDR_Mechanical
- PC_to_RDR_SetDataRateAndClockFrequency

In the following paragraphs for all the command messages:

The field bSlot must be set to 00h.

The field bSeq is not checked.

PC_to_RDR_IccPowerOn command

This command powers on the smart card. A cold and a warm resets are allowed.

This command is rejected if no contact smart card is currently declared inserted.

The command processing depends on the slot mode (APDU/EMV mode or TPDU mode).

- Slot in APDU/EMV Mode

In EMV mode, the reader powers on the card; it also checks that the ATR is compliant with the EMV standard and sets the card interface transmission parameters according to the response from the card.

Because it is required by the EMV specifications, if the reader succeeds in retrieving the response from the card (no time-out error, parity error or TCK error), but the ATR does not meet EMV specifications, the reader tries a warm reset.

If the ATR does not comply with EMV requirements, the reader deactivates the card.

If the reader encounters a transmission error, it deactivates the card and makes no further attempts to obtain a response from the card.

- Slot in TPDU Mode

The command is compliant with the ISO7816-3 standard. If the command fails, the card is powered off.

Because it does not parse the ATR, the reader does not store parameters.

To meet card requirements, the host must send a PC_to_RDR_SetParameters command to set the baud rate, the protocol, etc.

Refer to the PC_to_RDR_SetParameters command for a complete overview.

Offset	Field	Size	Value	Description
0	bMessageType	1	62h	PC_to_RDR_IccPowerOn
1	bwLength	4	00000000h	
5	bSlot	1	00h	Slot 0
6	bSeq	1	00-FFh	Sequence number for the command
7	bPowerSelect	1	01h	Voltage that is applied to the ICC TPDU mode and APDU mode 01h – 5.0V
8	abRFU	2	0000h	Reserved for future used

The contact smart card interface does not support automatic voltage selection.

The data returned is the card ATR.

The response to this command message is RDR_to_PC_DataBlock response message.

PC_to_RDR_IccPowerOff command

This command powers off the smart card.

Offset	Field	Size	Value	Description
0	bMessageType	1	63h	PC_to_RDR_IccPowerOff
1	bwLength	4	00000000h	
5	bSlot	1	00h	Slot 0
6	bSeq	1	00-FFh	Sequence number for the

				command
7	abRFU	3	000000h	Reserved for future used

The response to this command message is the RDR_to_PC_SlotStatus response message.

PC_to_RDR_GetSlotStatus command

This command is used to retrieve the current slot status:

- Card not present
 - no card is inserted
- Card present and not active
 - a card is inserted but is not powered (the PC_to_RDR_PowerOn command was not executed).
- Card present and active
 - a card is detected and powered. (the PC_to_RDR_PowerOn command was successfully executed).

Offset	Field	Size	Value	Description
0	bMessageType	1	65h	PC_to_RDR_GetSlotStatus
1	bwLength	4	00000000h	
5	bSlot	1	00h	Slot 0
6	bSeq	1	00-FFh	Sequence number for the command
7	abRFU	3	000000h	Reserved for future used

The response to this command message is RDR_to_PC_SlotStatus response message.

PC_to_RDR_XfrBlock command

This command will be rejected if no contact smart card is declared present and active.

This command is handled differently depending on what mode the slot is in:

- Slot in APDU/EMV Mode

The command is exchanged between the reader and the host, using APDU commands. As the reader exchanges TPDU commands with the card, it formats the command using the T=0 or T=1 protocol, depending on the fields of the ATR. If necessary, the reader chains the data in T=1; it attempts recovery in the event of a problem, etc. The host receives the result of the command in the APDU format.
- Slot in TPDU mode

The command is sent in TPDU mode. The data is sent to the card as it was received by the reader. The reader returns the card response to the Host in TPDU format.

When the command follows an ATR and its format is a PPS exchange, the reader starts a sequence of PPS exchanges.

The Data format for a PPS exchange is:

PPSS PPS0 [PPS1] [PPS2] [PPS3] PCK

With PPSS = FFh

Refer to ISO 7816-3 for more information.

Offset	Field	Size	Value	Description
0	bMessageType	1	6Fh	PC_to_RDR_XfrBlock
1	bwLength	4		Size of the abData field

Prox-DU & Prox-SU

5	bSlot	1	00h	Slot 0
6	bSeq	1	00-FFh	Sequence number for the command
7	bBWI	1	00-FFh	Used to extend the CCIDs Block Waiting Timeout for this current transfer. The CCID will timeout the block after “this number multiplied by the block Waiting Time” has expired.
8	wLevelParameter	2	0000h	Use changes depending of the exchange level reported by the class descriptor in dwFeature field Short APDU level, RFU = 0000h
10	abData	Byte Array		Short APDU command

The parameter bBWI is only use in TPDU mode and only for T=1 transfers.

The command message length = 10 + Card command length.

The maximum length in case of Short ADPU = 10 + 261 = 271 bytes.

The response to this command message is RDR_to_PC_DataBlock response message.

PC_to_RDR_GetParameters command

This command is used to retrieve the current slot parameters. It is always accepted.

Offset	Field	Size	Value	Description
0	bMessageType	1	6Ch	PC_to_RDR_GetParameters
1	bwLength	4	00000000h	
5	bSlot	1	00h	Slot 0
6	bSeq	1	00-FFh	Sequence number for the command
7	abRFU	3	000000h	Reserved for future used

The response to this command message is the RDR_to_PC_Parameters response message.

PC_to_RDR_ResetParameters command

This command is used to reset the current slot parameters.

The command is allowed in TPDU mode. In APDU mode, the command is rejected.

The slot resets the T=0 and T=1 parameters, but the slot is set to T=0 protocol.

Offset	Field	Size	Value	Description
0	bMessageType	1	6Dh	PC_to_RDR_ResetParameters
1	bwLength	4	00000000h	
5	bSlot	1	00h	Slot 0
6	bSeq	1	00-FFh	Sequence number for the command
7	abRFU	3	000000h	Reserved for future used

The response to this command message is the RDR_to_PC_Parameters response message.

PC_to_RDR_SetParameters command

This command is used to change the slot parameters such as the baud rate, the protocol, etc.

Prox-DU & Prox-SU

The command is allowed with the in TPDU mode. In APDU mode, the command is rejected.

Offset	Field	Size	Value	Description
0	bMessageType	1	61h	PC to RDR_SetParameters
1	bwLength	4		Size of the abProtocolDataStructure field
5	bSlot	1	00h	Slot 0
6	bSeq	1	00-FFh	Sequence number for the command
7	bProtocolNum	1	01h	Specifies what protocol data structure follows. 00h = structure for protocol T=0 01h structure for protocol T=1
8	abRFU	2		Reserved for future use
10	abProtocolDataStructure	Byte array		Protocol Data Structure

Protocol data structure for protocol T=0 (bProtocolNum = 0) (dwLength = 00000005h):

Offset	Field	Size	Value	Description
10	bmFindexDindex	1		b7-4 – FI selecting a clock rate conversion factor b3-0 – DI selecting a baud rate conversion factor see table Fi/Di of ISO 7816-3
11	bmTCCKST0	1	00h, 02h	For T=0, b0 – 0b b7-2 – 000000b b1 = 0 direct convention b1 = 1 inverse convention CCID ignores bit b1
12	bGuardTimeT0	1	00h-FFh	Extra Guard Time between two characters. Add 0 to 254 etu to the normal guard time 12 etu. FFh is the same as 00h
13	bWaitingIntegerT0	1	00h-FFh	WI for T=0 used to define WWT
14	bClockStop	1	00h	ICC clock stop support 00h = Stopping the clock is not allowed

bClockStop: Stopping the clock is not allowed for this parameter.

Protocol data structure for protocol T=1 (bProtocolNum = 1) (dwLength = 00000007h):

Offset	Field	Size	Value	Description
10	bmFindexDindex	1		b7-4 – FI selecting a clock rate conversion factor b3-0 – DI selecting a baud rate conversion factor see table Fi/Di of ISO 7816-3
11	bmTCCKST1	1	10h, 11h, 12h, 13h	For T=1, b7-2 = 000100b b0 = 0b – Checksum LRC b0 = 1b – Checksum CRC b1 = 0 - direct convention b1 = 1 - inverse convention CCID ignores bit b1
12	bGuardTimeT1	1	00-FFh	Extra Guard Time between two characters. If value FFh, then

Prox-DU & Prox-SU

13	bWaitingIntegersT1	1	00-9Fh	guard time is reduced by 1 etu. b7-b4 = BWI value 0-9 valid b3-b0 = CWI value 0-F valid
14	bClockStop	1	00h	ICC clock stop support 00h = Stopping the clock is not allowed
15	blfsc	1	00-FEh	Size of negotiated IFSC
16	bNadValue	1		Value = 00h if CCID doesn't support a value other than the default value. Else value respects ISO/IEC 7816-3 9.4.2.1

bClockStop: Use only stopping the clock is not allowed for this parameter.

bNadValue: The value is ignored.

The response to this message is the RDR_to_PC_Parameters response message.

PC_to_RDR_Escape command

This command allows the CCID manufacturer to define and access extended features.

The information sent via this command is processed by the CCID control logic.

Offset	Field	Size	Value	Description
0	bMessageType	1	6Bh	PC_to_RDR_Escape
1	bwLength	4		Size of the abData field
5	bSlot	1	00h	Slot 0
6	bSeq	1	00-FFh	Sequence number for the command
7	abRFU	3	000000h	Reserved for Future Used
10	abData	Byte array		Data block sent to the CCID

The response to this message is the RDR_to_PC_Escape response message.

Note: the Microsoft CCID USB driver parameters should be customized to process the CCID Escape Command because this feature is not enabled by default. Refer to the "Enabling the CCID Escape Command feature into the driver" paragraph for more information.

The Linux and Mac CCID USB driver support the next following Escape commands.

Switch interface

This command is use to switch to the contactless interface after a contact card is inserted and to switch back to the contact card interface.

The command format is the following:

Offset	Field	Size	Value	Description
0	bCommandFamily	1	52h	Reader command
1	bCommandType	1	F8h	Management
2	bCommand	1	04h	04h = Switch interface
3, 4	wLength	2	0001h	Size of the Data field
5	blnferface	1	01h,02h	01h = Switch to contactless interface 02h = Switch to contact interface

The response format is the following:

Offset	Field	Size	Value	Description
0,1	abStatus	2	XX XX	Command Status execution

2, 3	wLength	2	0000h	Size of the Data field
------	---------	---	-------	------------------------

The abStatus field can report a possible execution error.

PC_to_RDR_Abort command

This command is used with the control pipe Abort request to tell the CCID to stop any current transfer at the specified slot and return to a state where the slot is ready to accept a new command pipe Bulk-Out message.

Offset	Field	Size	Value	Description
0	bMessageType	1	72h	PC_to_RDR_Escape
1	bwLength	4	00000000h	
5	bSlot	1	00h	Slot 0
6	bSeq	1	00-FFh	Sequence number for the command
7	abRFU	3	000000h	Reserved for Future Used

The response to this message is the RDR_to_PC_SlotStatus response message.

Response pipe bulk-in for the contact card interface

The following CCID messages are implemented for the contact interface:

- RDR_to_PC_DataBlock
- RDR_to_PC_SlotStatus
- RDR_to_PC_Parameters
- RDR_to_PC_Escape

The following CCID message is not implemented:

- RDR_to_PC_DataRateAndClockFrequency

RDR_to_PC_DataBlock

This message is the response to the PC_to_RDR_IccPowerOn and PC_to_RDR_XfrBlock – commands.

For the PC_to_RDR_PowerOn command this response message contains the ATR of the card.

For the PC_to_RDR_XfrBlock command this response message contains the card response.

Offset	Field	Size	Value	Description
0	bMessageType	1	80h	RDR_to_PC_DataBlock
1	bwLength	4		Size of the abData field
5	bSlot	1	00h	Slot number : Same as Bulk-OUT message
6	bSeq	1	00-FFh	Sequence number : Same as Bulk-OUT message
7	bStatus	1	00h, 01h, 02h, 40h, 41h, 42h, 80h	Slot status register: 0Xh = no error 4Xh = command failed 80h Time extension request X = 0 card present and active X = 1 card present and inactive X = 2 card not present
8	bError	1	00h-FFh	Slot error register: Error when bStatus = 4Xh Time multiplier when bStatus =

Prox-DU & Prox-SU

				0x80h
9	bChainParameter	1	00h	Depends on the exchange level reported by the class descriptor in dwFeature field. TPDU level, Short APDU level, this field is RFU =00h.
10	abData	Byte array		This field contains the data returned by the CCID

The response message length = 10 + Card card response length (or ATR length).

The maximum length in case of Short ADPU = 10 + 258 = 268 bytes.

RDR_to_PC_SlotStatus

This message is the response to the PC_to_RDR_IccPowerOff, PC_to_RDR_GetSlotStatus and PC_to_RDR_Abort commands.

Offset	Field	Size	Value	Description
0	bMessageType	1	81h	RDR_to_PC_SlotStatus
1	bwLength	4	00000000h	
5	bSlot	1	00h	Slot number : Same as Bulk-OUT message
6	bSeq	1	00-FFh	Sequence number : Same as Bulk-OUT message
7	bStatus	1	00h, 01h, 02h, 40h, 41h, 42h	Slot status register: 0Xh = no error 4Xh = command failed X = 0 card present and active X = 1 card present and inactive X = 2 card not present
8	bError	1	00h-FFh	Slot error register: Error when bStatus = 4Xh
9	bClockStatus	1	00h	Value = 00h – Clock running

RDR_to_PC_Parameters

This message is the response to the PC_to_RDR_GetParameters, PC_to_RDR_ResetParameters and PC_to_RDR_SetParameters commands.

Offset	Field	Size	Value	Description
0	bMessageType	1	82h	RDR_to_PC_Parameters
1	bwLength	4		Size of the abProtocolDataStructure field
5	bSlot	1	00h	Slot number : Same as Bulk-OUT message
6	bSeq	1	00-FFh	Sequence number : Same as Bulk-OUT message
7	bStatus	1	00h, 01h, 02h, 40h, 41h, 42h	Slot status register: 0Xh = no error 4Xh = command failed X = 0 card present and active X = 1 card present and inactive X = 2 card not present
8	bError	1	00h-FFh	Slot error register: Error when bStatus = 4Xh
9	bProtocolNum	1	00h, 01h	Specifies what protocol data structure follows.

Prox-DU & Prox-SU

				00h = structure for protocol T=0 01h = structure for protocol T=1
10	abProtocolDataStructure	Byte array	00h, 01h	Protocol data structure

Protocol data structure for protocol T=0 (bProtocolNum = 0) (dwLength = 00000005h):

Offset	Field	Size	Value	Description
10	bmFindexDindex	1		b7-4 – FI selecting a clock rate conversion factor b3-0 – DI selecting a baud rate conversion factor
11	bmTCCKST0	1	00h, 02h	For T=0, b0 – 0b b7-2 – 000000b b1 = 0 direct convention used b1 = 1 inverse convention used
12	bGuardTimeT0	1	00h-FFh	Extra Guard Time between two characters. Add 0 to 254 etu to the normal guard time 12 etu. FFh is the same as 00h
13	bWaitingIntegerT0	1	00h-FFh	WI for T=0 used to define WWT
14	bClockStop	1	00h	ICC clock stop support 00h = Stopping the clock is not allowed

Protocol data structure for protocol T=1 (bProtocolNum = 1) (dwLength = 00000007h):

Offset	Field	Size	Value	Description
10	bmFindexDindex	1		b7-4 – FI selecting a clock rate conversion factor b3-0 – DI selecting a baud rate conversion factor
11	bmTCCKST1	1	10h, 11h, 12h, 13h	For T=1, b7-2 = 000100b b0 = 0b – Checksum type LRC b0 = 1b – Checksum type CRC b1 = 0 - direct convention used b1 = 1 - inverse convention used
12	bGuardTimeT1	1	00-FFh	Extra Guard Time between two characters. If value FFh, then guard time is reduced by 1 etu.
13	bWaitingIntegersT1	1	00-9Fh	b7-b4 = BWI value 0-9 valid b3-b0 = CWI value 0-F valid
14	bClockStop	1	00h	ICC clock stop support 00h = Stopping the clock is not allowed
15	blfsc	1	00-FEh	Size of negotiated IFSC
16	bNadValue	1		NAD value used by CCID

RDR_to_PC_Escape

This message is the response to the PC_to_RDR_Escape command.

Offset	Field	Size	Value	Description
0	bMessageType	1	83h	RDR_to_PC_Escape
1	bwLength	4		Size of the abData field
5	bSlot	1	00h	Slot number : Same as Bulk-OUT message
6	bSeq	1	00-FFh	Sequence number : Same as Bulk-OUT message
7	bStatus	1	00h, 01h, 02h, 40h, 41h, 42h	Slot status register: 0Xh = no error 4Xh = command failed X = 0 card present and active X = 1 card present and inactive X = 2 card not present
8	bError	1	00h-FFh	Slot error register: Error when bStatus = 4Xh
9	bRFU	1	00h	Reserved for Future Used
10	abData	Byte array		Data sent from CCID

Reporting slot error and slot status registers in bulk-in messages for the contact interface

Each bulk-in message contains the values of the Slot Error register (bError) and the Slot Status register (bStatus).

Slot error register: when bmCommandStatus = 1

Error Code	Error Name	Possible cases
FFh	CMD_ABORTED	Host aborted the current activity
FEh	ICC_MUTE	CCID time out while talking to the ICC
FDh	XFR_PARITY_ERROR	Parity error while talking to the ICC
FCh	XFR_OVERRUN	Overrun error while talking to the ICC
FBh	HW_ERROR	An all inclusive hardware error occurred
F8h	BAD_ATR_TS	
F7h	BAD_ATR_TCK	
F6h	ICC_PROTOCOL_NOT_SUPPORTED	
F5h	ICC_CLASS_NOT_SUPPORTED	
F4h	PROCEDURE_BYTE_CONFLICT	
F3h	DEACTIVATED_PROTOCOL	
F2h	BUSY_WITH_AUTO_SEQUENCE	Automatic Sequence Ongoing
F0h	PIN_TIMEOUT	
EFh	PIN_CANCELED	

E0h	CMD_SLOT_BUSY	A second command was sent to a slot which was already processing a command
C0h to 81h	User Defined	Low byte of error code listed in Annex
80h and those filling the gaps	Reserved for future used	
7Fh to 01h	Index of not supported / incorrect message parameter	
00h	Command not supported	

Table 24 – Slot error register when bmCommandStatus = 1

Slot Status register:

Offset	Field	Size	Value	Description
0	bmICCStatus	2 bits	0, 1, 2	0 – An ICC is Present and active (power is on and stable, RST is Inactive) 1 – An ICC is present and inactive (not activated or shut down by hardware error) 2 – No ICC is Present 3 – RFU
2	bmRFU	4bits	0	RFU
6	bmCommandStatus	2 bits	0, 1, 2	0 – Processed without error 1 – Failed (error code provided by error register) 2 – Time Extension is requested 3 – RFU

Table 25 – Slot Status register

When the bmCommandStatus field is 0 indicating the command processed without error or when the bmCommand field is an RFU value, then the slot's error register is RFU.

When the bmCommandStatus field is 1 indicating the command failed, then the slot's error register is set with a signed 8-bit value.

When the bmCommandStatus field is 2, indicating a time extension is requested, then the slot's error register contains the multiplier value of BWT when the protocol is T=1 or the multiplier value of WWT when the protocol is T=0.

Command processed without error: Slot Status register = 0xh

Command failed (error code provided by the error register): Slot Status register = 4xh

Time extension is requested (slot error register = time multiplier): Slot Status register = 80h

x = 0 : Card present and active

x = 1 : Card present and inactive

x = 2 : Card not present

For the contactless interface:

An ICC is present and active when a contactless card was detected and a PC_to_RDR_PowerOn command was processed without error.

An ICC is Present and inactive when contactless card was detected and no PC_to_RDR_PowerOn was processed or processed with error or the ICC was shut down by a PC_to_RDR_PowerOff command or after an error.

For the contact smart card interface:

An ICC is present and active when a card is inserted and powered (the PC_to_RDR_PowerOn command was processed without error).

An ICC is Present and inactive when card is inserted and not powered (the PC_to_RDR_PowerOn was processed or processed with error or the ICC was shut down by a PC_to_RDR_PowerOff command or after an error).

Interrupt in messages for the contact card interface

The Interrupt-In endpoint is used to notify the host of events that may occur asynchronously and outside the context of a command-response exchange between host and CCID. If the host has sent a Bulk-Out message and is waiting for a Bulk-In message in response, and one of these events occurs, then the Bulk-In message may have duplicate information related to the event.

Only the RDR_to_PC_NotifySlotChange message is implemented.

The RDR_to_PC_HardwareError message is not implemented.

RDR_to_PC_NotifySlotChange

This message is sent whenever the CCID device detects a change in the insertion status of an ICC slot. If an ICC is either inserted or removed from a slot, this message must be sent. The presence of this message means to the host driver that a change has occurred. It is possible for more than one change to occur between deliveries of RDR_to_PC_NotifySlotChange messages.

When the USB bus is resumed from a suspended state, both the CCID and the host driver must make identical assumptions about the state of the ICC slots. For simplicity, the specification requires that both CCID and host driver shall presume that all slots are empty. Therefore, after resumption from suspend, the CCID shall report all occupied ICC slots using this message.

Offset	Field	Size	Value	
0	bMessageType	1	50h	
1	bSlotICCState			This field is reported on byte granularity. The size is (2 bits * number of slots) rounded up to the nearest byte. Each slot has 2 bits. The least significant bit reports the current state of the slot (0b = no ICC present, 1b = ICC present). The most significant bit reports whether the slot has changed state since the last RDR_to_PC_NotifySlotChange message was sent (0b = no change, 1b = change). If no slot exists for a given location, the field returns 00b in those 2 bits. Example: A 3 slot CCID reports a single byte with the following format:

				Bit 0 = Slot 0 current state Bit 1 = Slot 0 changed status Bit 2 = Slot 1 current state Bit 3 = Slot 1 changed status Bit 4 = Slot 2 current state Bit 5 = Slot 2 changed status Bit 6 = 0b Bit 7 = 0b
--	--	--	--	---

For the **contact smart card Interface** only slot 0 is defined. Therefore, bSlotICCState can have the following value:

- 00h: no ICC present, no change since the last RDR_to_PC_NotifySlotChange message was sent
- 01h: ICC present, no change since the last RDR_to_PC_NotifySlotChange message was sent
- 02h: no ICC present, the slot has changed state since the last RDR_to_PC_NotifySlotChange message was sent
- 03h: ICC present, the slot has changed state since the last RDR_to_PC_NotifySlotChange message was sent

CCID device for the contactless interface

Command pipe bulk-out messages for the contactless interface

The following CCID commands are implemented for the contactless interface:

For driver compatibility reasons, some of these commands or their parameters will do anything but the response will be like the functionality works.

- PC_to_RDR_IccPowerOn
- PC_to_RDR_IccPowerOff
- PC_to_RDR_GetSlotStatus
- PC_to_RDR_XfrBlock
- PC_to_RDR_GetParameters
- PC_to_RDR_ResetParameters
- PC_to_RDR_SetParameters
- PC_to_RDR_Escape
- PC_to_RDR_Abort

The following CCID command messages are not implemented

- PC_to_RDR_IccClock
- PC_to_RDR_T0APDU
- PC_to_RDR_Secure
- PC_to_RDR_Mechanical
- PC_to_RDR_SetDataRateAndClockFrequency

In the following paragraphs for all the command messages:

- bSlot must be set to 00h
- bSeq is not checked

PC_to_RDR_IccPowerOn command

This command acts like a power on of a contact card.

Cold reset and warm reset are possible but the pseudo ATR will be always the same.

This command is rejected if no contactless card is currently declared present.

The bPowerSelect parameter is checked to verify its integrity but is not used.

Offset	Field	Size	Value	Description
0	bMessageType	1	62h	PC_to_RDR_IccPowerOn
1	bwLength	4	00000000h	
5	bSlot	1	00h	Slot 0
6	bSeq	1	00-FFh	Sequence number for the command
7	bPowerSelect	1	00h-03h	Voltage that is applied to the ICC 00h – automatic voltage selection 01h – 5.0V 02h – 3.0v 03h – 1.8V

				For a contactless card this parameter don't care
8	abRFU	2	0000h	Reserved for future used

The response to this command message is the RDR_to_PC_DataBlock response message.

The data returned is the pseudo ATR computed according to the PC/SC specification.

After a successful Power On, the T=1 protocol is automatically selected.

The PC_to_RDR_GetParameters command message will return the corresponding parameters.

PC_to_RDR_IccPowerOff command

This command acts like a power off of a contact card.

Offset	Field	Size	Value	Description
0	bMessageType	1	63h	PC_to_RDR_IccPowerOff
1	bwLength	4	00000000h	
5	bSlot	1	00h	Slot 0
6	bSeq	1	00-FFh	Sequence number for the command
7	abRFU	3	000000h	Reserved for future used

The response to this command message is the RDR_to_PC_SlotStatus response message.

PC_to_RDR_GetSlotStatus command

This command is used to retrieve the current slot status:

- No ICC is present
 - No card detected in the RF field
- An ICC is present and inactive
 - A card is present but the PC_to_RDR_PowerOn command was not executed.
- An ICC is present and active
 - A card is present and the PC_to_RDR_PowerOn command was successfully executed.

Offset	Field	Size	Value	Description
0	bMessageType	1	65h	PC_to_RDR_GetSlotStatus
1	bwLength	4	00000000h	
5	bSlot	1	00h	Slot 0
6	bSeq	1	00-FFh	Sequence number for the command
7	abRFU	3	000000h	Reserved for future used

The response to this command message is the RDR_to_PC_SlotStatus response message.

PC_to_RDR_XfrBlock command

This command will be rejected if no contactless card is declared present and powered.

The parameter bBWI is not managed because this parameter is only use by CCIDs which use the character level and TPDU level of exchange (as reported in the dwFeature parameter in the CCID functional descriptor) and only for T=1 transfers.

Prox-DU & Prox-SU

dwFeature defines Extended APDU for the contactless interface.

Full Extended APDU command length is supported ($7+65535+2 = 65544$ bytes).

The Extended APDU command must be split into several PC_to_RDR_XfrBlock messages according to the CCID specification (the PC_toRDR_XfrBlock message length is limited to dwCCIDMessageLength).

For a Mifare cards the data must be a Short APDU command as defined in PC/SC specification.

For ISO14443-4 cards the data are send “as it” using the T=CL protocol.

For other proprietary APDU commands, the data must be a Short APDU command.

Offset	Field	Size	Value	Description
0	bMessageType	1	6Fh	PC_to_RDR_XfrBlock
1	bwLength	4		Size of the abData field
5	bSlot	1	00h	Slot 0
6	bSeq	1	00-FFh	Sequence number for the command
7	bBWI	1	00-FFh	Used to extend the CCIDs Block Waiting Timeout for this current transfer. The CCID will timeout the block after “this number multiplied by the block Waiting Time” has expired
8	wLevelParameter	2	xxxxh	Use changes depending of the exchange level reported by the class descriptor in dwFeature field For the contactless interface: Extended APDU level Indicate if an APDU command begins and ends with this command 0000h :The APDU command begins and ends with this command 0001h :The APDU command begins with this command and continues in the next PC_to_RDR_XfrBlock 0002h :This abData field continues an APDU command and ends the APDU command. 0003h :The abData field continues an APDU command and another block is to follow. 0010h :Empty abDatafield continuation of response APDU is expected in the next RDR_to_PC_DataBlock
10	abData	Byte Array	4 to 502	Data block sent to the CCID. For a T=CL card this data are send “as it” using the T=CL protocol. For a Mifare card: the data must be a Short APDU command as defined in PC/SC specification.

The command message length = 10 + Card command length.

Prox-DU & Prox-SU

The maximum length in case of Short ADPU = 10 + 261 = 271 bytes.

The maximum length in case of Extended ADPU = 10 + 502 = 512 bytes

The response to this command message is the RDR_to_PC_DataBlock response message.

PC_to_RDR_GetParameters command

This command is used to retrieve the current slot parameters.

Offset	Field	Size	Value	Description
0	bMessageType	1	6Ch	PC_to_RDR_GetParameters
1	bwLength	4	00000000h	
5	bSlot	1	00h	Slot 0
6	bSeq	1	00-FFh	Sequence number for the command
7	abRFU	3	000000h	Reserved for future used

All parameters have no meaning for a contactless card, except the bNadValue parameter in the Protocol data structure for protocol T=1.

This value is used for the NAD field of the T=CL protocol.

Other parameters are not used but are stored with the command PC_to_RDR_SetParameter or set to default value with the command PC_toRDR_ResetParameter to be send back in the response message RDR_to_PC_Parameters.

The response to this command message is the RDR_to_PC_Parameters response message.

PC_to_RDR_ResetParameters Command

This command is used to reset the current slot parameters.

Offset	Field	Size	Value	Description
0	bMessageType	1	6Dh	PC_to_RDR_ResetParameters
1	bwLength	4	00000000h	
5	bSlot	1	00h	Slot 0
6	bSeq	1	00-FFh	Sequence number for the command
7	abRFU	3	000000h	Reserved for future used

All parameters have no meaning for a contactless card, except the bNadValue parameter in the Protocol data structure for protocol T=1.

This value is used for the NAD field of the T=CL protocol.

Other parameters are not used but are and stored to be send back in the response message RDR_to_PC_Parameters.

The response to this command message is the RDR_to_PC_Parameters response message.

PC_to_RDR_SetParameters command

This command is used to change the slot parameters.

Offset	Field	Size	Value	Description
--------	-------	------	-------	-------------

Prox-DU & Prox-SU

0	bMessageType	1	61h	PC_to_RDR_SetParameters
1	bwLength	4		Size of the abProtocolDataStructure field
5	bSlot	1	00h	Slot 0
6	bSeq	1	00-FFh	Sequence number for the command
7	bProtocolNum	1	01h	Specifies what protocol data structure follows. 00h = structure for protocol T=0 01h structure for protocol T=1 Only T=1 structure will be used for the contactless interface
8	abRFU	2		Reserved for future use
10	abProtocolDataStructure	Byte array		Protocol Data Structure

All parameters have no meaning for a contactless card, except the bNadValue parameter in the Protocol data structure for protocol T=1.

This value is used for the NAD field of the T=CL protocol.

Other parameters are not used but are be stored to be sent back in the response message RDR_to_PC_Parameters.

Protocol data structure for protocol T=0 (bProtocolNum = 0) (dwLength = 00000005h):

Offset	Field	Size	Value	Description
10	bmFindexDindex	1		b7-4 – FI selecting a clock rate conversion factor b3-0 – DI selecting a baud rate conversion factor
11	bmTCKKST0	1	00h, 02h	For T=0, b0 – 0b b7-2 – 000000b b1 = 0 direct convention CCID ignores bit b1
12	bGuardTimeT0	1	00h-FFh	Extra Guard Time between two characters. Add 0 to 254 etu to the normal guard time 12 etu. FFh is the same as 00h
13	bWaitingIntegerT0	1	00h-FFh	WI for T=0 used to define WWT
14	bClockStop	1	00h- 03h	ICC clock stop support 00h = Stopping the clock is not allowed

Protocol data structure for protocol T=1 (bProtocolNum = 1) (dwLength = 00000007h):

Offset	Field	Size	Value	Description
10	bmFindexDindex	1		b7-4 – FI b3-0 – DI
11	bmTCKKST1	1	10h, 11h, 12h, 13h	For T=0, b7-2 = 000100b b0 = 0b – Checksum LRC b0 = 1b – Checksum CRC b1 = 0 - direct convention CCID ignores bit b1
12	bGuardTimeT1	1	00-FFh	Extra Guard Time between two characters. If value FFh, then guard time is reduced by 1 etu.
13	bWaitingIntegersT1	1	00-9Fh	b7-b4 = BWI value 0-9 valid

Prox-DU & Prox-SU

14	bClockStop	1	00h- 03h	b3-b0 = CWI value 0-F valid ICC clock stop support 00h = Stopping the clock is not allowed
15	bfsc	1	01-FEh	Size of negotiated IFSC
16	bNadValue	1		Value = 00h if CCID doesn't support a value other than the default value. Else the value respects ISO/IEC 7816-3 9.4.2.1

The response to this message is the RDR_to_PC_Parameters response message.

PC_to_RDR_Escape command

This command allows the CCID manufacturer to define and access extended features.

Information sent via this command is processed by the CCID control logic.

Offset	Field	Size	Value	Description
0	bMessageType	1	6Bh	PC_to_RDR_Escape
1	bwLength	4		Size of the abData field
5	bSlot	1	00h	Slot 0
6	bSeq	1	00-FFh	Sequence number for the command
7	abRFU	3	000000h	Reserved for Future Used
10	abData	Byte array		Data block sent to the CCID

The detail of the commands in the adData field is described in the following paragraph.

The response to this message is the RDR_to_PC_Escape response message.

Note: the Microsoft CCID USB driver parameters should be customized to process the CCID Escape Command because this feature is not enabled by default. Refer to the "Enabling the CCID Escape Command feature into the driver" paragraph for more information.

The Linux and Mac CCID USB driver support the next following Escape commands.

Switch interface

This command is used to switch to the contactless interface after a contact card is inserted and to switch back to the contact card interface.

This command act as the "Switch interface" command defined in the "Proprietary Commands on the HID Interface" and the format in the abData field is the same.

Get firmware version

This command is useful for the Gemalto CCID driver.

The format is the same than the command of the GemCore POS Pro chip.

abData field = 0x02

The response is the Prox-DU or SU string version as defined in the "Firmware Versioning Rules" paragraph.

PC_to_RDR_Abort command

This command is used with the control pipe Abort request to tell the CCID to stop any

current transfer at the specified slot and return to a state where the slot is ready to accept a new command pipe Bulk-Out message.

Offset	Field	Size	Value	Description
0	bMessageType	1	72h	PC to RDR_Escape
1	bwLength	4	00000000h	
5	bSlot	1	00h	Slot 0
6	bSeq	1	00-FFh	Sequence number for the command
7	abRFU	3	000000h	Reserved for Future Used

The response to this message is the RDR_to_PC_SlotStatus response message.

Response pipe bulk-in messages for the contactless interface

The following CCID messages are implemented for the contactless interface:

- RDR_to_PC_DataBlock
- RDR_to_PC_SlotStatus
- RDR_to_PC_Parameters
- RDR_to_PC_Escape

The following CCID message is not implemented:

- RDR_to_PC_DataRateAndClockFrequency

RDR_to_PC_DataBlock Command

This message is the response to the PC_to_RDR_IccPowerOn and the PC_to_RDR_XfrBlock command messages.

For the PC_to_RDR_PowerOn command message this response message contains the pseudo ATR data associated with the contactless card.

For the PC_to_RDR_XfrBlock command message this response message contains the card response.

If the card is a T=CL card:

- The card response is send “as it”. (Only the concatenated data of the INF field of the T=CL block).
- The full extended APDU length response is supported (65536+2 = 65538 bytes). The APDU response will be split into several RDR_to_PC_XfrBlock messages according to CCID specification.

If the card is a MIFARE® card: the card response is the data read in the card followed by a status word SW1SW2 or only the status, according to the PC/SC specification.

Offset	Field	Size	Value	Description
0	bMessageType	1	80h	RDR_to_PC_DataBlock
1	bwLength	4		Size of the abData field
5	bSlot	1	00h	Slot number : Same as Bulk-Out message
6	bSeq	1	00-FFh	Sequence number : Same as Bulk-Out message
7	bStatus	1	00h, 01h, 02h, 40h, 41h, 42h, 80h	Slot status register: 0Xh = no error 4Xh = command failed 80h Time extension request X = 0 card present and active

Prox-DU & Prox-SU

				X = 1 card present and inactive X = 2 card not present
8	bError	1	00h-FFh	Slot error register: Error when bStatus = 4Xh Time multiplier when bStatus = 0x80h
9	bChainParameter	1	00h	Depends on the exchange level reported by the class descriptor in dwFeature field. Short APDU level, this field is RFU = 00h
10	abData	Byte array		This field contains the data returned by the CCID

The response message length = 10 + Card response length (or ATR length).

The maximum length in case of Short ADPU = 10 + 258 = 268 bytes.

The maximum length in case of extended ADPU response = 10 + 259 = 269 bytes.

RDR_to_PC_SlotStatus Command

This message is the response to the PC_to_RDR_IccPowerOff, PC_to_RDR_GetSlotStatus and PC_to_RDR_Abort commands.

Offset	Field	Size	Value	Description
0	bMessageType	1	81h	RDR_to_PC_SlotStatus
1	bwLength	4	00000000h	
5	bSlot	1	00h	Slot number : Same as Bulk-Out message
6	bSeq	1	00-FFh	Sequence number : Same as Bulk-Out message
7	bStatus	1	00h, 01h, 02h, 40h, 41h, 42h	Slot status register: 0Xh = no error 4Xh = command failed X = 0 card present and active X = 1 card present and inactive X = 2 card not present
8	bError	1	00h-FFh	Slot error register: Error when bStatus = 4Xh
9	bClockStatus	1	00h	Value = 00h – Clock running

RDR_to_PC_Parameters Command

This message is the response to the PC_to_RDR_GetParameters, PC_to_RDR_ResetParameters and PC_to_RDR_SetParameters commands.

All parameters have no meaning for a contactless card, except the bNadValue parameter in the Protocol data structure for protocol T=1.

This value is used for the NAD field of the T=CL protocol.

Other parameters are not used but are stored with the command PC_to_RDR_SetParameter or set to default value with the command PC_toRDR_ResetParameter to be send back in the response message RDR_to_PC_Parameters.

Offset	Field	Size	Value	Description
--------	-------	------	-------	-------------

Prox-DU & Prox-SU

0	bMessageType	1	82h	RDR_to_PC_Parameters
1	bwLength	4		Size of the abProtocolDataStructure field
5	bSlot	1	00h	Slot number : Same as Bulk-Out message
6	bSeq	1	00-FFh	Sequence number : Same as Bulk-Out message
7	bStatus	1	00h, 01h, 02h, 40h, 41h, 42h	Slot status register: 0Xh = no error 4Xh = command failed X = 0 card present and active X = 1 card present and inactive X = 2 card not present
8	bError	1	00h-FFh	Slot error register: Error when bStatus = 4Xh
9	bProtocolNum	1	00h, 01h	Specifies what protocol data structure follows. 00h = structure for protocol T=0 01h = structure for protocol T=1
10	abProtocolDataStructure	Byte array	00h, 01h	Protocol data structure

Protocol data structure for protocol T=0 (bProtocolNum = 0) (dwLength = 00000005h):

Offset	Field	Size	Value	Description
10	bmFindexDindex	1		b7-4 – FI selecting a clock rate conversion factor b3-0 – DI selecting a baud rate conversion factor
11	bmTCKKST0	1	00h, 02h	For T=0, b0 – 0b b7-2 – 000000b b1 = 0 direct convention used
12	bGuardTimeT0	1	00h-FFh	Extra Guard Time between two characters. Add 0 to 254 etu to the normal guard time 12 etu. FFh is the same as 00h
13	bWaitingIntegerT0	1	00h-FFh	WI for T=0 used to define WWT
14	bClockStop	1	00h- 03h	ICC clock stop support 00h = Stopping the clock is not allowed

The default values are the following:

bmFindexDindex : 11h
 bmTCKKST0 : 00h
 bGuardTimeT0 : 00h
 bWaitingIntegerT0 : 0Ah
 bClockStop : 00h

Protocol data structure for protocol T=1 (bProtocolNum = 1) (dwLength = 00000007h):

Offset	Field	Size	Value	Description
10	bmFindexDindex	1		b7-4 – FI selecting a clock rate conversion factor b3-0 – DI selecting a baud rate conversion factor

Prox-DU & Prox-SU

11	bmTCCKST1	1	10h, 11h, 12h, 13h	For T=1, b7-2 = 000100b b0 = 0b – Checksum type LRC b0 = 1b – Checksum type CRC b1 = 0 - direct convention used
12	bGuardTimeT1	1	00-FFh	Extra Guard Time between two characters. If value FFh, then guard time is reduced by 1 etu.
13	bWaitingIntegersT1	1	00-9Fh	b7-b4 = BWI value 0-9 valid b3-b0 = CWI value 0-F valid
14	bClockStop	1	00h- 03h	ICC clock stop support 00h = Stopping the clock is not allowed
15	blfsc	1	01-FEh	Size of negotiated IFSC
16	bNadValue	1		NAD value used by CCID

The default values are the following:

```

bmFindexDindex      : 11h
bmTCCKST1           : 10h
bGuardTimeT1        : 00h
bWaitingIntegersT1  : 4Dh
bClockStop           : 00h
blfsc                : 20h
bNadValue            : 00h
    
```

RDR_to_PC_Escape Command

This message is the response to the PC_to_RDR_Escape command.

Offset	Field	Size	Value	Description
0	bMessageType	1	83h	RDR_to_PC_Escape
1	bwLength	4		Size of the abData field
5	bSlot	1	00h	Slot number : Same as Bulk-Out message
6	bSeq	1	00-FFh	Sequence number : Same as Bulk-Out message
7	bStatus	1	00h, 01h, 02h, 40h, 41h, 42h	Slot status register: 0Xh = no error 4Xh = command failed X = 0 card present and active X = 1 card present and inactive X = 2 card not present
8	bError	1	00h-FFh	Slot error register: Error when bStatus = 4Xh
9	bRFU	1	00h	Reserved for Future Used
10	abData	Byte array		Data sent from CCID

Reporting slot error and slot status registers in bulk-in messages for the contactless interface

Each bulk-in message contains the values of the Slot Error register (bError) and the Slot Status register (bStatus).

These values are the same as defined in the “Reporting slot error and slot status registers in bulk-in messages for the contact interface” paragraph.

Interrupt in messages for the contactless card interface

The Interrupt-In endpoint is used to notify the host of events that may occur asynchronously and outside the context of a command-response exchange between host and CCID. If the host has sent a Bulk-Out message and is waiting for a Bulk-In message in response, and one of these events occurs, then the Bulk-In message may have duplicate information related to the event.

Only the RDR_to_PC_NotifySlotChange message is implemented.

The RDR_to_PC_HardwareError message is not implemented.

RDR_to_PC_NotifySlotChange message

This message is sent whenever the CCID device detects a change in the insertion status of an ICC slot. If an ICC is either inserted or removed from a slot, this message must be sent. The presence of this message means to the host driver that a change has occurred. It is possible for more than one change to occur between deliveries of RDR_to_PC_NotifySlotChange messages.

When the USB bus is resumed from a suspended state, both the CCID and the host driver must make identical assumptions about the state of the ICC slots. For simplicity, the specification requires that both CCID and host driver shall presume that all slots are empty. Therefore, after resumption from suspend, the CCID shall report all occupied ICC slots using this message.

Offset	Field	Size	Value	
0	bMessageType	1	50h	
1	bSlotICCState			<p>This field is reported on byte granularity. The size is (2 bits * number of slots) rounded up to the nearest byte. Each slot has 2 bits. The least significant bit reports the current state of the slot (0b = no ICC present, 1b = ICC present). The most significant bit reports whether the slot has changed state since the last RDR_to_PC_NotifySlotChange message was sent (0b = no change, 1b = change). If no slot exists for a given location, the field returns 00b in those 2 bits.</p> <p>Example: A 3 slot CCID reports a single byte with the following format:</p> <ul style="list-style-type: none"> Bit 0 = Slot 0 current state Bit 1 = Slot 0 changed status Bit 2 = Slot 1 current state Bit 3 = Slot 1 changed status Bit 4 = Slot 2 current state Bit 5 = Slot 2 changed status Bit 6 = 0b

Prox-DU & Prox-SU

				Bit 7 = 0b
--	--	--	--	------------

For the **Contactless Interface** only slot 0 is defined. Therefore, bSlotICCState can have the following value:

- 00h: no ICC present, no change since the last RDR_to_PC_NotifySlotChange message was sent
- 01h: ICC present, no change since the last RDR_to_PC_NotifySlotChange message was sent
- 02h: no ICC present, the slot has changed state since the last RDR_to_PC_NotifySlotChange message was sent
- 03h: ICC present, the slot has changed state since the last RDR_to_PC_NotifySlotChange message was sent

USB CCID Class Driver Details

This paragraph provides USB smart card class-driver release information for devices that are compliant with the USB CCID specification.

Overview

Microsoft CCID class driver

The Microsoft CCID class driver is compatible with the USB Chip/Smart Card Interface Devices (CCID) Specification (revision 1.0 or later), which specifies a protocol that a host (computer) can use to interact with CCID class devices or interface (on a composite device). Neither the mechanics of the smart-card interface or the content of the data are described in the CCID specification. However, the CCID specification does provide detailed information with respect to the USB-related configuration and communication channels.

The current release of the Microsoft CCID class driver implements a majority of the features defined in the USB CCID specification. The Microsoft CCID class driver will support the following items that are based on the USB CCID Class specification:

- 5.0V, 3.0V, and 1.8V cards.
- Both T=0 and T=1 protocols.
- Variable clock frequencies and data rates.
- All features in the `dwFeatures` field, including Character, TPDU, APDU, and extended APDU levels, although TPDU is the preferred exchange level.
- In order to send or receive an Escape command to a reader, the `DWORD` registry value `EscapeCommandEnable` must be added and set to a non-zero value under the `HKLM\SYSTEM\CCS\Enum\USB\Vid*Pid*\Device Parameters` key.

Then the vendor IOCTL for the Escape command is defined as follows: `#define IOCTL_CCID_ESCAPE SCARD_CTL_CODE(3500)`.

With the enabled Escape Command, security against malicious escape commands becomes the reader's responsibility.

- USB CCID readers should implement the `GET_CLOCK_FREQUENCIES` and `GET_DATA_RATES` properties, even if bit 20h in `dwFeatures` is set. The values of the `bNumDataRatesSupported` and `bNumClockSupported` functions should also be non-zero accordingly. This is due to a problem with the USB CCID Class specification where the driver is supposed to send a PPS request; however, if bit 20h and the values associated with the PPS request are zero, the driver does not know what values to set in the PPS request. If `bNumDataRatesSupported` and/or `bNumClockSupported` are set to zero, the driver will make a guess as to what baud rates the reader supports, which may or may not be correct.

The following features are not currently supported in this initial release of the Microsoft CCID class driver:

- Keypad or LCD display support.
- Vendor/device-specific string name support in the device manager. Since the INF to load USBCCID is not included with Windows, the INF will be renamed as `OEM*.INF`. Vendors can not refer to this INF through `Include/Needs`.
- Support for multiple slots on readers. If the reader has multiple slots, only slot 0 will be used. Devices that wish to expose multiple readers may develop a composite device (a CCID-compliant interface would then be required for each reader).

Prox-DU & Prox-SU

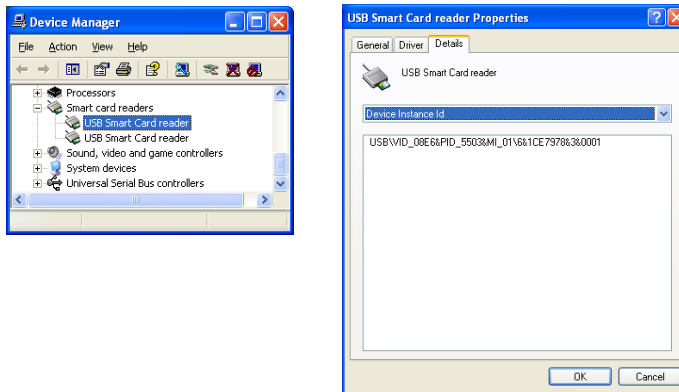
- Driver support for system wake from suspend/hibernate state on card insertion (even if the reader sets the remote wake bit). This feature may be made available as more smart-card readers provide remote wake functionality.
- Selective suspend support.
- Support for issuing multiple commands to a reader that implements queuing capabilities.
- Support for any of the mechanical driver features.

Enabling the CCID Escape Command feature into the Microsoft driver

To enable the CCID Escape Command feature with a Prox-DU or Prox-SU reader/writer the following operations should be performed to customize the Microsoft CCID driver:

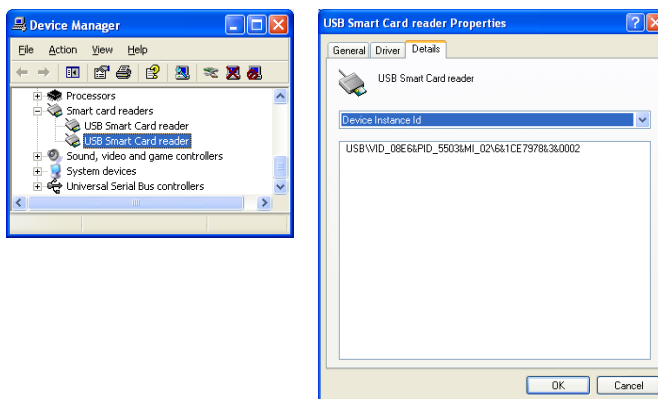
1- First find in your computer the USB information related to the two USB Smart Card readers:

- Open the Device Manager window to display the two devices:
- Double click on the first icon to get the properties of the first device:



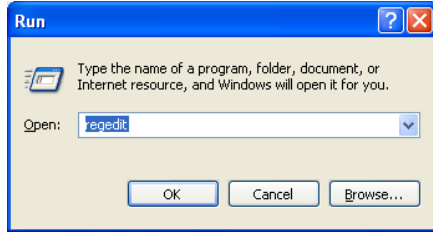
The USB information is “USB\VID_08E6&PID_5503&MI_01\6&1CE7978&3&0001” in the previous picture.

- Double click on the second icon to get the properties of the second device:

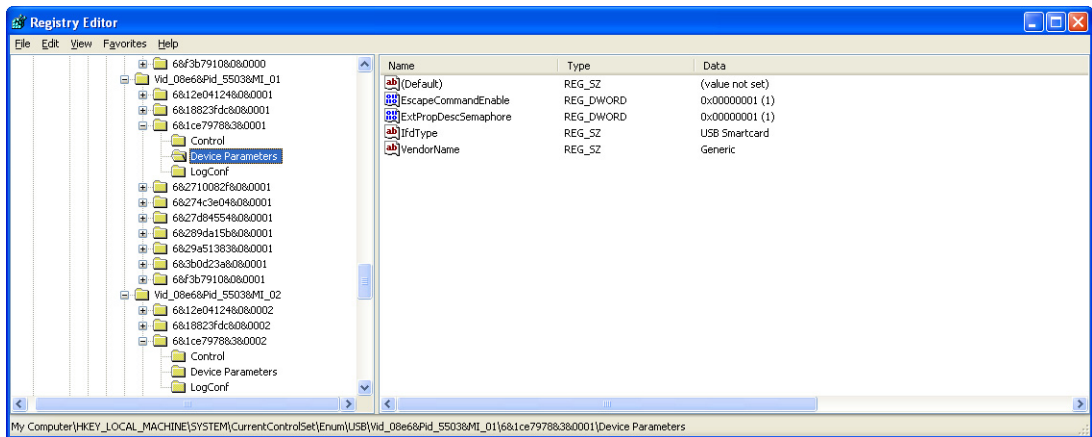


The USB information is “USB\VID_08E6&PID_5503&MI_02\6&1CE7978&3&0002” in the previous picture.

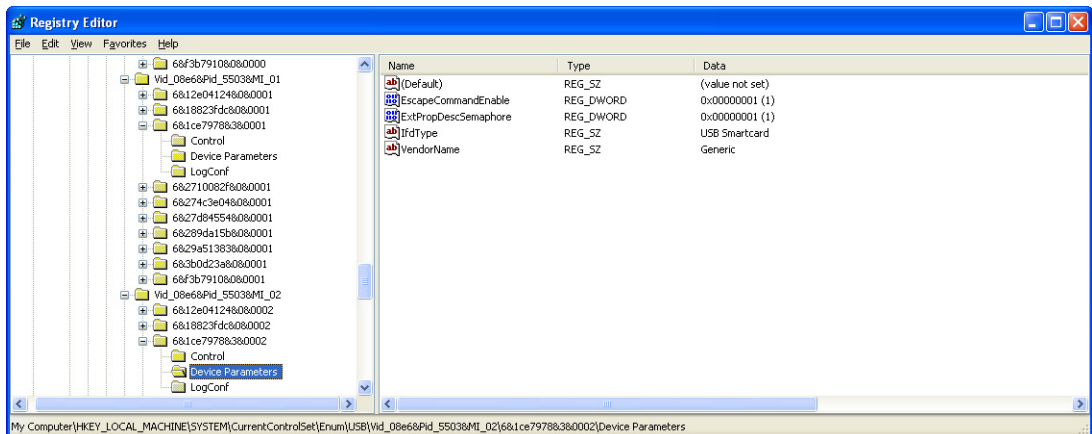
2- Then edit the computer registry using the “Run” program and the “regedit” command:



- Add the DWORD registry value “EscapeCommandEnable” and set to a non-zero value under the HKLM\SYSTEM\CCS\Enum\USB\Vid*Pid**\Device Parameters key using the USB information previously recovered:
 - Open the appropriate folder: “HKLM\SYSTEM\CCS\Enum\USB\Vid*Pid**\Device Parameters”
 - Click the “Edit\New\DWORD Value” menu
 - Rename the new key “EscapeCommandEnable”
 - Double click the new key to edit it and set the value to “1”
- Perform the operation for the first USB Smart Card readers: (USB\VID_08E6&PID_5503&MI_01\6&1CE7978&3&0001 in the example)



- Perform the operation with the second USB Smart Card readers: (USB\VID_08E6&PID_5503&MI_02\6&1CE7978&3&0002 in the example)



- Close the Registry Editor window

The registry modification will apply on the next connection of the Prox-DU or Prox-SU reader/writer.

CCID Escape Control Code for Microsoft Operating Systems

The application should use the following control code to send the escape command:

- `#define IOCTL_CCID_ESCAPE SCARD_CTL_CODE(3500)`

Defining the vendor IOCTL for the CCID Escape Commands supported by the Prox-DU or Prox-SU reader/writer:

- Firmware version
- Switch interface

Note: a smart card should be connected when the application want to use the Escape Command on the corresponding interface. Else an error will be returned.

Linux and Mac CCID class driver

The Linux and Mac CCID class driver is also compatible with the USB Chip/Smart Card Interface Devices (CCID) Specification (revision 1.0 or later).

The following web site <http://pcsclite.alioth.debian.org/ccid.html> is proposing a package that provides the source code for a generic USB CCID (Chip/Smart Card Interface Devices) driver and ICCD (Integrated Circuit(s) Card Devices).

The main CCID/ICCD features supported are the following:

- Exchange levels
 - short APDU
 - extended APDU (with some limitations)
 - TPDU
 - character
- Multi-slot readers
- PC/SC v2 part 10 features:
 - secure PIN verify
 - modify PIN entry
 - ifd PIN properties
 - MCT reader direct
- Data rates list
- LCD display (Gemalto GemPC PIN PAD)
- Extended APDU (for T=1 cards only and if your reader is in TPDU mode or extended APDU mode)
- SCardGetAttrib() attributes
- ICCD versions A and B

The next CCID features are not yet supported:

- Manage suspend/resume (for example in a laptop)
- Reader clock frequency change

The supported operating systems are the following:

- Linux
- Mac OS X
 - Snow Leopard (10.6)
 - Leopard (10.5)
 - Tiger (10.4)

CCID Escape Control Code for Linux and Mac Operating Systems

The application should use the following control code to send the escape command:

- `#define IOCTL_CCID_ESCAPE SCARD_CTL_CODE(1)`

Defining the vendor IOCTL for the CCID Escape Commands supported by the Prox-DU or Prox-SU reader/writer:

- Get firmware version
- Switch interface

HID Devices

The HID class consists primarily of devices that are used by humans to control the operation of computer systems as keyboard and standard mouse.

Other devices that may not require human interaction but provide data in a similar format can also be defined as HID class devices.

Therefore, the HID class definition includes support for various types of output directed to the end user.

The Prox-DU and Prox-SU reader/writers use a vendor defined HID interface for the device administration.

Gemalto proprietary commands

The HID interface is used for device administration using Gemalto proprietary commands.

The commands are coded using the following format:

Offset	Field	Size	Value	Description
0	bCommandFamily	1	XX	Code of the command family
1	bCommandType	1	XX	Code of the command type
2	bCommand	1	XX	Code of the command
3, 4	wLength	2	XXXX	Size of the Data field
5...		x	XX	Optional Data of the command

bCommandFamily:

- 52h = Reader command
- 53h = Bootloader command

bCommandType:

- F8h = Management command
- F9h = Download Management

bInstruction:

Code of the command to execute (detail in the following paragraphs)

wLength:

- This is the number of byte of the optional Data field.
- If no data are present, wLength = 0000h

The response of the commands uses the following format:

Offset	Field	Size	Value	Description
0,1	abStatus	2	XX XX	Command Status execution
2, 3	wLength	2	XXXX	Size of the Data field
4...		x		Optional Data of the response

abStatus:

- This is the command execution status:
- 0x00 0x00 = OK
- Other values report an error:
- The first byte is the origin of the error

Prox-DU & Prox-SU

0xFF: General common error

0xFB: Bootloader

The second byte is the error code

Refer to the “HID commands error codes” paragraph for more information about the error status values.

wLength:

This is the number of byte of the optional Data field.

If no data are present, wLength = 0000h

Convention:

Placeholder prefixes such as ‘b’ ‘w’ ‘ab’ are used to denote placeholder type

b : byte (1 bytes)

w : word (2 bytes)

dw : double word (4 bytes)

ab: array of bytes

The field that are larger than a byte are stored in little Indian (LSB first).

Proprietary commands

All the proprietary commands and response are sent using the HID interface.

The general proprietary commands are the followings:

- Firmware version request
- Read EEPROM parameters
- Write EEPROM parameters
- Switch interface
- Read switch interface state
- Reset reader
- Start download
- Download firmware file
- End download

Firmware version request command

This command enables the user to retrieve the reader firmware version.

The command format is the following:

Offset	Field	Size	Value	Description
0	bCommandFamily	1	52h	Reader command
1	bCommandType	1	F8h	Management
2	bCommand	1	02h	Firmware version request
3, 4	wLength	2	0000h	Size of the Data field

The response format is the following:

Offset	Field	Size	Value	Description
0, 1	abStatus	2	XX XX	Command Status execution
2, 3	wLength	2	XXXX	Size of the Data field
4	abData	n		firmware version string

Prox-DU & Prox-SU

The firmware version string is defined in the paragraph “Firmware versioning rules”.

The abStatus field can report a possible execution error.

Note: When the bootloader is running, this command enables the user to determine the Bootloader version. The response can be checked to confirm if the bootloader is running or if the reader firmware is running.

Read EEPROM parameters command

This command allows reading the EEPROM parameters contents.

The command format is the following:

Offset	Field	Size	Value	Description
0	bCommandFamily	1	52h	Reader command
1	bCommandType	1	F8h	Management
2	bCommand	1	00h	Read EEPROM
3, 4	wLength	2	0002h	Size of the Data field
5	boffset	1	00h-41h	EEPROM Offset (0 to 65)
6	bNbBytes	1	01h-42h	Number of byte to read (1 to 66)

Warning: Only 66 bytes from offset 00h to 41h can be read. Then (offset + Nb bytes) must be lower or equal to 42h.

The response format is the following:

Offset	Field	Size	Value	Description
0,1	abStatus	2	XX XX	Command Status execution
2, 3	wLength	2	00XXh	Size of the Data field (2 + bNbBytes)
4	bNbBytes	1	01h-42h	Number of byte read (1 to 66)
5	abDataEEPROM	n		Data read in the EEPROM

The abStatus field can report a possible execution error.

Write EEPROM parameters command

This command allows writing the EEPROM parameters contents.

The command format is the following:

Offset	Field	Size	Value	Description
0	bCommandFamily	1	52h	Reader command
1	bCommandType	1	F8h	Management
2	bCommand	1	01h	Write EEPROM
3, 4	wLength	2	XXXXh	Size of the Data field (2+n)
5	bOffset	1	00h-41h	EEPROM Offset (0 to 65)
6	bNbBytes	1	01h-42h	Number of byte to write (1 to 66)
7	abDataEEPROM	n		Data to write in the EEPROM

Warning: Only 66 bytes from offset 00h to 41h can be written. Then (offset + Nb bytes) must be lower or equal to 42h. Else no byte will be written.

The response format is the following:

Offset	Field	Size	Value	Description
0,1	abStatus	2	XX XX	Command Status execution
2, 3	wLength	2	0000h	Size of the Data field

The abStatus field can report a possible execution error.

Switch interface command

This command is used to switch to the contactless interface after a contact card is inserted and to switch back to the contact card interface.

The command format is the following:

Offset	Field	Size	Value	Description
0	bCommandFamily	1	52h	Reader command
1	bCommandType	1	F8h	Management
2	bCommand	1	04h	04h = Switch interface
3, 4	wLength	2	0001h	Size of the Data field
5	blnInterface	1	01h,02h	01h = Switch to contactless interface 02h = Switch to contact interface

The response format is the following:

Offset	Field	Size	Value	Description
0,1	abStatus	2	XX XX	Command Status execution
2, 3	wLength	2	0000h	Size of the Data field

The abStatus field can report a possible execution error.

Read switch interface state command

This command is used to read the current state of the interface switch.

The command format is the following:

Offset	Field	Size	Value	Description
0	bCommandFamily	1	52h	Reader command
1	bCommandType	1	F8h	Management
2	bCommand	1	03h,	03h = Read current interface switch state
3, 4	wLength	2	0000h	Size of the Data field

The response format is the following:

Offset	Field	Size	Value	Description
0,1	abStatus	2	XX XX	Command Status execution
2, 3	wLength	2	0001h	Size of the Data field
4	blnInterface	1	00h, 01h, 02h	Current interfaces switch state 00h = initial state 01h = contactless interface 02h = contact interface

The abStatus field can report a possible execution error.

Notes:

Current interfaces switch state = 00h when no contactless card is detected and no contact card is inserted in the reader slot. This corresponds to the state 0 of the dual power security manager.

Current interfaces switch state = 01h when a contactless card is detected and a contact card is inserted or not in the reader slot. This corresponds to the state 1, 3 and 4 of the dual power security manager.

Current interfaces switch state = 02h when no contactless card is detected and a contact card is inserted in the reader slot. This corresponds to the state 2 of the dual power security manager.

Reset reader command

This command is used to reset the Prox-DU.

The command format is the following:

Offset	Field	Size	Value	Description
0	bCommandFamily	1	52h	Reader command
1	bCommandType	1	F8h	Management
2	bCommand	1	05h	03h = Read current interface switch state 04h = Switch interface
3, 4	wLength	2	0000h	Size of the Data field

The response format is the following:

Offset	Field	Size	Value	Description
No answer for this command				

Start download command

This command is sent by the host to the reader.

The command operates in different ways depending on the current firmware mode.

If the reader runs in the kernel mode:

- When this command is received,
- All the reader operation are terminated,
- A removal card message is send for the two interface,
- An indicator (ApplicationValid byte) is cleared for the boot loader to stay in download operations.
- An acknowledge is sent to the Host using the HID interface.
- The reader is restarted, using the microcontroller watch dog.

If the reader already runs in the boot mode (that means that the user pushed the rescue button):

- The indicator (ApplicationValid byte) is NOT cleared. (it allows the user to reboot the reader & start the previous application without updating it)
- An acknowledge is sent to the Host using the HID interface.

The command format is the following:

Offset	Field	Size	Value	Description
0	bCommandFamily	1	53h	Bootloader command
1	bCommandType	1	F9h	Download Management
2	bCommand	1	00h	Start Download
3, 4	wLength	2	0000h	Size of the Data field

The response format is the following:

Offset	Field	Size	Value	Description
0, 1	abStatus	2	XX XX	Command Status execution
2, 3	wLength	2	0000h	Size of the Data field

The abStatus field can report a possible execution error.

Download firmware file command

This command is used to download the file that contains the reader firmware data.

Prox-DU & Prox-SU

This command is only available when the boot-loader is running.
Refer to the “Boot-Loader” paragraph for more information.

End download command

This command is the last command of the download process.
This command is only available when the boot-loader is running.
Refer to the “Boot-Loader” paragraph for more information.

HID Library

A library is available for the following operating systems:

- Windows
- Linux
- Mac OS X

This library supports all the commands listed in the previous paragraph.
The HID libraries are available in the following web link <http://support.gemalto.com>.

HID Commands Error Codes

In the following table:

- The MSB byte correspond to the first byte reported in the abStatus field of the HID response
- The LSB byte correspond to the second byte reported in the abStatus field of the HID response

Error label	Value	Meaning
ERR_OK	0000h	Execution OK
Wrong command or wrong data parameters (command rejected)		
ERR_BAD_COMMAND	FF82h	Wrong command bytes
ERR_BAD_PARAM	FF83h	Wrong data parameters
Other codes FFxxh not listed are RFU		

Table 26 – Common error codes

Firmware Versioning Rules

Reader firmware string version

The string of the reader firmware version is composed of several fields:

<Name> <separator> <Release version> <separator> <Customer> <separator>
<Casing/usage> <Order number>

Name:	"Gemalto_Pro_DU"	product name of Prox Dual reader
	"Gemalto_Prox_SU"	product name of Prox SU reader
Separator:	"-"	
Release version:	"Vx.yz"	release version number x.yz
Separator:	"-"	
Customer:	"G"	Gemalto
Casing/Usage:	"XD"	Official release
	"W"	working release
Order number:	"nn"	incremental number for each version. "00" to "99"
		It restarts to 00 when the release version number is incremented

Boot-loader string version

The string version of the boot-loader respects the same rules than the reader firmware version but the name is different.

Gemalto_Prox_BootU -Vx.yz-GXDnn

Name: "Gemalto_Prox_BootU"

USB Descriptors

This chapter provides information about the USB descriptors for the Prox-DU and Prox-SU devices.

Standard USB Descriptors

Device descriptor

The device is USB 2.0 compatible.

The reserved PID/VID of the devices is the following:

- VID = 08E6h
- PID = 5502h : Boot Loader Prox Dual
- PID = 5503h : Prox-DU
- PID = 5504h : Prox-SU
- PID = 5505h : reserved for future use

The device has one configuration.

Device Descriptor		
Offset	Value	Field
0	0x12	bLength (18 bytes)
1	0x01	bDescriptorType (Device)
2	0x00	bcdUSB release number (2.00)
	0x02	
4	0x00	bDeviceClass
5	0x00	bDeviceSubClass
6	0x00	bDeviceProtocol
7	0x20	bMaxPacketSize0 (Max packet size Endpoint 0 = 32 bytes)
8	0xE6	IdVendor = 0x08E6
	0x08	
10	XX	IdProduct = 0x5502 / 0x5503 / 0x5504
	0x55	
12	0x00	BcdDevice = 0x0100 (Device release number 1.00)
	0x01	
14	0x01	iManufacturer (Index of string descriptor manufacturer = 1)
15	0x02	iProduct (Index of string descriptor product = 2)
16	0x03	iSerialNumber ((Index of string descriptor device serial number = 3)
17	0x01	bNumConfigurations (Number of possible configurations)

Table 27 – USB Device Descriptor

Configuration descriptor

The device has three interfaces (one interface for the contact smart card, one interface for the contactless smart card and one interface for the administration of the device).

The device is “Bus Powered” (delivered from the USB cable, no external power).

The device doesn't support remote wake.

The maximum power current is 200 mA.

Refer to chapter for the specific configuration descriptor when the boot loader is running

Configuration Descriptor		
Offset	Value	Field
0	0x09	bLength (9 bytes)
1	0x02	bDescriptorType (configuration)
2	0xCA	wTotalLength of the Configuration Descriptor (configuration + interface + endpoint class specific) $9 + (9 + 9 + 7) + (9 + 54 + 7 * 3) * 2 = 202$
	0x00	
4	0x03	bNumInterfaces (number of interface = 3)
5	0x01	bConfigurationValue
6	0x00	iConfiguration (ignored)
7	0x80	bmAttributes (b6 = 0 Bus powered, b0 = 1 Remote Wake Up Not supported)
8	0x64	MaxPower (100*2=200 mA)

Table 28 – USB Configuration Descriptor

Interfaces descriptors

The interface for the device administration is HID.

The class for the contactless smart card and contact smart card interfaces is Smart Card CCID.

These two interfaces have three endpoints.

HID Interface Descriptor		
Offset	Value	Field
0	0x09	bLength (9 bytes)
1	0x04	bDescriptorType (Interface)
2	0x00	bInterfaceNumber (Interface 1)
3	0x00	bAlternateSetting
4	0x01	bNumEndpoints (1 Endpoints)
5	0x03	bInterfaceClass (HID class)
6	0x00	bInterfaceSubClass (No subclass)
7	0x00	bInterfaceProtocol (none)
8	0x04	iInterface (index to interface string descriptor = 4)

Table 29 – USB HID Interface Descriptor

Contactless Smart Card Interface Descriptor		
Offset	Value	Field
0	0x09	bLength (9 bytes)
1	0x04	bDescriptorType (Interface)
2	0x01	bInterfaceNumber (Interface 2)
3	0x00	bAlternateSetting
4	0x03	bNumEndpoints (3 Endpoints)
5	0x0B	bInterfaceClass (Smart Card device class)
6	0x00	bInterfaceSubClass (No subclass)
7	0x00	bInterfaceProtocol (none)
8	0x06	iInterface (index to interface string descriptor = 6)

Table 30 – USB Contactless Smart Card Interface Descriptor

The interface for the contact smart card has three endpoints.

The class for the contact smart card interface is Smart Card CCID.

Contact Smart Card Interface Descriptor		
Offset	Value	Field
0	0x09	bLength (9 bytes)
1	0x04	bDescriptorType (Interface)
2	0x02	bInterfaceNumber (Interface 3)
3	0x00	bAlternateSetting
4	0x03	bNumEndpoints (3 Endpoints)
5	0x0B	bInterfaceClass (Smart Card device class)
6	0x00	bInterfaceSubClass (No subclass)
7	0x00	bInterfaceProtocol (none)
8	0x05	iInterface (index to interface string descriptor = 6)

Table 31 – USB Contact Smart Card Interface Descriptor

Device Class Descriptors

HID class descriptor

HID Class Descriptor		
Offset	Value	Field
0	0x09	bLength (9 bytes)
1	0x21	bDescriptorType
2	0x00 0x01	bcdHID (1.00)
4	0x00	bCountryCode (not supported)
5	0x01	bNumDescriptors (1 report)
6	0x22	bDescriptorType
7	0x32 0x00	wDescriptorLength (50 bytes)

Table 32 – USB HID Class Descriptor

HID interface endpoint descriptor

HID Interface Endpoint Descriptor (endpoint 4 Interrupt In)		
Offset	Value	Field
0	0x07	bLength (7 bytes)
1	0x05	bDescriptorType (Endpoint)
2	0x83	bEndpointAddress (b7=1 IN, b3-b0 = address 3)
3	0x03	bmAttributes (03h =Interrupt endpoint)
4	0x08 0x00	wMaxPacketSize (8 bytes max)
6	0xFE	bInterval (254ms)

Table 33 – USB HID Interface Endpoint Descriptor

HID report descriptor

HID Report Descriptor		
Offset	Value	Field
0	06	Usage page (Vendor defined)
1	00	

Prox-DU & Prox-SU

2	FF		
3	09	Usage ID (Vendor defined 01)	
4	01		
5	A1	Collection (Application)	
6	01		
7	09	Input Report	
8	02	Usage ID (Vendor defined 02)	
9	15	Input Report	
10	00	Logical Minimum (0)	
11	26	Input Report	
12	FF		Logical Maximum (255)
13	00		
14	75	Input Report	
15	08	Report Size (8) 8 bits per data	
16	96	Input Report	
17	16		Report Count (0x116) 278 x8 bits
18	01		
19	81	Input (Data, Variable, Absolute)	
20	02		
21	09	Output Report	
22	03	Usage ID (Vendor defined 03)	
23	15	Output Report	
24	00	Logical Minimum (0)	
25	26	Output Report	
26	FF		Logical Maximum (255)
27	00		
28	75	Output Report	
29	08	Report Size (8) 8 bits per data	
30	96	Output Report	
31	16		Report Count (0x116) 278 x 8bits
32	01		
33	91	Output (Data, Variable, Absolute)	
34	02		
35	09	Feature Report	
36	04	Usage ID (Vendor defined 04)	
37	15	Feature Report	
38	00	Logical Minimum (0)	
39	26	Feature Report	
40	FF		Logical Maximum (255)
41	00		
42	75	Feature Report	
43	08	Report Size (8) 8bits per data	
44	96	Feature Report	
45	16		Report Count (0x116) 278 x 8 bits
46	01		
47	B1	Feature (Data, Variable, Absolute)	
48	02		
49	C0	End of Collection (Application)	

Table 34 – USB HID Report Descriptor

Contactless smart card device class descriptor

Contactless Smart Card Device Class Descriptor		
Offset	Value	Field
0	0x36	bLength (54 bytes)
1	0x21	bDescriptorType
2	0x10	bcdCCID CCID (1.10 Class release number)
	0x01	
4	0x00	bmaxSlotIndex (1 slots)
5	0x01	bVoltageSupport = 0x01 b0 = 1 : 5V only (its virtual for contactless interface Only one voltage to avoid multiple power on attempt)
6	0x03	dwProtocols (b1=1 supports T=1, b0=1 supports T=0)
	0x00	
	0x00	
	0x00	
10	0xA0	dwDefaultClock (4 MHz = 4000 KHz:= 0x00000FA0)
	0x0F	
	0x00	
	0x00	
14	0xA0	dwMaximumClock (4 MHz = 4000 KHz:= 0x00000FA0))
	0x0F	
	0x00	
	0x00	
18	0x00	bNumClockSupported (Manual setting not allowed)
19	0x00	dwDataRate (clock / 372 = 10752 bps = 0x00002A00)
	0x2A	
	0x00	
	0x00	
23	0x00	dwDataRate (clock / 372 = 10752 bps = 0x00002A00)
	0x2A	
	0x00	
	0x00	
27	0x00	bNumDataRatesSupported (manual setting not allowed)
28	0xFE	dwMaxIFSD (254 bytes) (Frame T=CL: 251 or 252 or 253 data bytes according to CID and NAD presence. Don't care in ADPU mode and automatic IFSD exchange)
	0x00	
	0x00	
	0x00	
32	0x00	dwSynchProtocols (no synchronous card)
	0x00	
	0x00	
	0x00	
36	0x00	dwMechanical (no special characteristics)
	0x00	
	0x00	
	0x00	
40	0x72	dwFeatures = 00040672h 00000002h: Automatic parameter configuration based on ATR data

Prox-DU & Prox-SU

	0x06	00000010h: Automatic ICC clock frequency change 00000020h: Automatic baud rate according to parameters 00000040h: Automatic parameters negotiation made by the CCID (use warm or cold reset or PPS)
	0x04	00000200h : NAD value other than 00 Accepted (T=1)
	0x00	00000400h: Automatic IFSD exchange as first exchange (T=1) 00040000h: Extended APDU level exchange with CCID
44	0x00	dwMaxCCIDMessageLength = 512 Bulk-Out message max length
	0x02	
	0x00	
	0x00	
48	0x00	bClassGetResponse
49	0x00	bClassEnvelope
50	0x00	wLcdLayout (0 line, 0 character per line)
	0x00	
52	0x00	bPINSupport (PIN verification not supported)
53	0x01	bMaxCCIDBusySlots (1 slot can be busy at the time)

Table 35 – USB Contactless Smart Card Device Class Descriptor

Contactless smart card interface endpoint descriptors

Contactless Smart Card Interface Endpoint Descriptor (endpoint 1 Bulk Out)		
Offset	Value	Field
0	0x07	bLength (7 bytes)
1	0x05	bDescriptorType (Endpoint)
2	0x01	bEndpointAddress (b7=0 OUT, b3-b0 = address 1)
3	0x02	bmAttributes (02h =Bulk endpoint)
4	0x40	wMaxPacketSize (64 bytes max)
	0x00	
6	0x00	bInterval (For full speed : Ignored)

Table 36 – USB Contactless Smart Card Interface Endpoint Descriptor (Bulk Out)

Contactless Smart Card Interface Endpoint Descriptor (endpoint 2 Bulk In)		
Offset	Value	Field
0	0x07	bLength (7 bytes)
1	0x05	bDescriptorType (Endpoint)
2	0x82	bEndpointAddress (b7=1 IN, b3-b0 = address 2)
3	0x02	bmAttributes (02h = Bulk endpoint)
4	0x40	wMaxPacketSize (64 bytes max)
	0x00	
6	0x00	bInterval (For full speed : Ignored)

Table 37 – USB Contactless Smart Card Interface Endpoint Descriptor (Bulk In)

Contactless Smart Card Interface Endpoint Descriptor (endpoint 6 Interrupt In)		
Offset	Value	Field
0	0x07	bLength (7 bytes)
1	0x05	bDescriptorType (Endpoint)
2	0x86	bEndpointAddress (b7=1 IN, b3-b0 address 6)
3	0x03	bmAttributes (Interrupt endpoint)
4	0x08	wMaxPacketSize (8 bytes max)
	0x00	
6	0x18	bInterval (Polling Interval = 24 ms)

Table 38 – USB Contactless Smart Card Interface Endpoint Descriptor (Interrupt In)

Contact smart card device class descriptor

Contact Smart Card Device Class Descriptor		
Offset	Value	Field
0	0x36	bLength (54)
1	0x21	bDescriptorType
2	0x10	bcdCCID CCID (1.10 Class release number)
	0x01	
4	0x00	bmaxSlotIndex (1 slots)
5	0x07	TPDU mode : bVoltageSupport = 0x07 (5V + 3V + 1,8V) APDU/EMV mode : bVoltageSupport = 0x01 (5V)
6	0x03	dwProtocols (b0=1 supports T=0 and b1=1 supports T=1)
	0x00	
	0x00	
	0x00	
10	0xA0	dwDefaultClock (4 MHz = 4000 KHz:= 0x00000FA0)
	0x0F	
	0x00	
	0x00	
14	0xA0	dwMaximumClock (4 MHz = 4000 KHz:= 0x00000FA0)
	0x0F	
	0x00	
	0x00	
18	0x00	bNumClockSupported (Manual setting not allowed)
19	0x00	dwDataRate (clock / 372 = 10752 bps = 0x00002A00)
	0x2A	
	0x00	
	0x00	
23	0x20	dwMaxDataRate (clock*D/F : 500000 bps = 0x0007A120 for TA1 = 97)
	0xA1	
	0x07	
	0x00	
27	0x00	bNumDataRatesSupported (manual setting not allowed)
28	0xFE	dwMaxIFSD (254 bytes)
	0x00	
	0x00	
	0x00	
32	0x00	dwSynchProtocols (no synchronous card)
	0x00	
	0x00	
	0x00	
36	0x00	dwMechanical (no special characteristics)
	0x00	
	0x00	
	0x00	
40	0x30	<u>TPDU/ISO mode</u>

Prox-DU & Prox-SU

	0x02	dwFeatures = 00010230h 00000010h: Automatic ICC clock according to parameters 00000020h: Automatic baud rate according to parameters
	0x01	00000200h: NAD value other than 00 accepted 00010000h: TPDU level exchanges with CCID
	0x00	<u>APDU/EMV mode</u> dwFeatures = 00020472h 00000002h: Automatic parameter configuration based on ATR data 00000010h: Automatic ICC clock frequency change 00000020h: Automatic baud rate according to parameters 00000040h: Automatic parameters negotiation made by the CCID 00000400h: Automatic IFSD exchange as first exchange (T=1) 00020000h: Short APDU level exchange with CCID
44	0x0F	dwMaxCCIDMessageLength (271 bytes)
	0x01	
	0x00	
	0x00	
48	0x00	bClassGetResponse
49	0x00	bClassEnvelope
50	0x00	wLcdLayout (0 line, 0 character per line)
	0x00	
52	0x00	bPINSupport (PIN verification not supported)
53	0x01	bMaxCCIDBusySlots (1 slot can be busy at the time)

Table 39 – USB Contact Smart Card Device Class Descriptor

Contact smart card interface endpoint descriptors

Contact Smart Card Interface Endpoint Descriptor (endpoint 4 Bulk Out)		
Offset	Value	Field
0	0x07	bLength (7 bytes)
1	0x05	bDescriptorType (Endpoint)
2	0x04	bEndpointAddress (b7=0 OUT, b3-b0 = address 4)
3	0x02	bmAttributes (02h =Bulk endpoint)
4	0x40	wMaxPacketSize (64 bytes max)
	0x00	
6	0x00	bInterval (For full speed : Ignored)

Table 40 – USB Contact Smart Card Interface Endpoint Descriptor (Bulk Out)

Contact Smart Card Interface Endpoint Descriptor (endpoint 5 Bulk In)		
Offset	Value	Field
0	0x07	bLength (7 bytes)
1	0x05	bDescriptorType (Endpoint)
2	0x85	bEndpointAddress (b7=1 IN, b3-b0 = address 5)
3	0x02	bmAttributes (02h = Bulk endpoint)
4	0x40	wMaxPacketSize (64 bytes max)
	0x00	
6	0x00	bInterval (For full speed : Ignored)

Table 41 – USB Contact Smart Card Interface Endpoint Descriptor (Bulk In)

Contact Smart Card Interface Endpoint Descriptor (endpoint 7 Interrupt IN)		
Offset	Value	Field
0	0x07	bLength (7 bytes)

Prox-DU & Prox-SU

1	0x05	bDescriptorType (Endpoint)
2	0x87	bEndpointAddress (b7=1 IN, b3-b0 address 7)
3	0x03	bmAttributes (Interrupt endpoint)
4	0x08	wMaxPacketSize (8 bytes max)
	0x00	
6	0x18	bInterval (Polling Interval =24 ms)

Table 42 – USB Contact Smart Card Interface Endpoint Descriptor (Interrupt In)

String Descriptors

LangID string descriptor

LangID String Descriptor		
Offset	Value	Field
0	0x04	bLength (12)
1	0x03	bDescriptorType (String)
2	0x09	wLangID[0] (U.S. English = 0409h)
	0x04	

Table 43 – USB LangID String Descriptor

Manufacturer string descriptor

String01 (index iManufacturer of device descriptor)		
Offset	Value	Field
0	0x10	bLength
1	0x03	bDescriptorType (String)
2	'G',0	bString = "Gemalto"
	'e',0	
	'm',0	
	'a',0	
	'l',0	
	't',0	
	'o',0	

Table 44 – USB Manufacturer String Descriptor

Product string descriptor

String02 (index iProduct of device descriptor)		
Offset	Value	Field
0	0x3A	bLength (58 bytes for Prox DU or 54 bytes for Prox SU)
1	0x03	bDescriptorType (String)
2	0x50,0x00	bString = "Prox Dual USB PC Link Reader" or "Prox SU USB PC Link Reader"
	0x72,0x00	
	0x6F,0x00,	
	0x78,0x00,	
	0x20,0x00,	

Prox-DU & Prox-SU

0x44,0x00,	(hexadecimal value are shown for Prox Dual)
0x75,0x00,	
0x61,0x00,	
0x6C,0x00,	
0x20,0x00,	
0x55,0x00,	
0x53,0x00,	
0x42,0x00,	
0x20,0x00,	
0x50,0x00,	
0x43,0x00,	
0x20,0x00,	
0x4C,0x00,	
0x69,0x00,	
0x6E,0x00,	
0x6B,0x00,	
0x20,0x00,	
0x52,0x00,	
0x65,0x00,	
0x61,0x00,	
0x64,0x00,	
0x65,0x00,	
0x72,0x00	

Table 45 – USB Product String Descriptor

Serial number string descriptor

String03 (index iSerialNumber of device descriptor)		
Offset	Value	Field
0	0x22	bLength (34 bytes)
1	0x03	bDescriptorType (String)
2	SN7,0x00,	bString = "SN7 SN6 SN5 SN4 SN3 SN2 SN1 SN0" The serial number value is the 8 ASCII characters string of the serial number printed on the reader label and bar code
	SN7,0x00,	
	SN6,0x00,	
	SN6,0x00,	
	SN5,0x00,	
	SN5,0x00,	
	SN4,0x00,	
	SN4,0x00,	
	SN3,0x00,	
	SN3,0x00,	
	SN2,0x00,	
	SN2,0x00,	
	SN1,0x00,	
	SN1,0x00,	
SN0,0x00,		
SN0,0x00,		

Table 46 – USB Serial Number String Descriptor

HID interface string descriptor

String04 (index iInterface of interface descriptor) (HID interface)		
Offset	Value	Field
0	0x2A	bLength (42 bytes)
1	0x03	bDescriptorType (String)
2	'P',0x00, 'r',0x00, 'o',0x00, 'x',0x00, '-',0x00, 'D',0x00, 'U',0x00, ' ',0x00, 'H',0x00, 'I',0x00, 'D',0x00, '_',0x00, 'x',0x00, 'x',0x00, 'x',0x00, 'x',0x00, 'x',0x00, 'x',0x00, 'x',0x00, 'x',0x00, 'x',0x00, 'x',0x00,	bString = "Prox-DU HID_XXXXXXXX" or " Prox-SU HID_XXXXXXXX " where XXXXXXXX is the reader serial number printed on the label

Table 47 – USB HID Interface String Descriptor

Contactless smart card interface string descriptor

String05 (index iInterface of interface descriptor) (Contactless smart card interface)		
Offset	Value	Field
0	0x3A	bLength (58 bytes)
1	0x03	bDescriptorType (String)
2	'P',0x00 'r',0x00 'o',0x00 'x',0x00 '-',0x00 'D',0x00 'U',0x00 ' ',0x00 'C',0x00, 'o',,0x00, 'n',0x00, 't',0x00, 'a',0x00, 'c',0x00,	bString = "Prox-DU Contactless_XXXXXXXX" or "Prox-SU Contactless_XXXXXXXX" where XXXXXXXX is the reader serial number printed on the label

't',0x00,
'l',0x00,
'e',0x00,
's',0x00,
's',0x00,
'_',0x00,
'x',0x00,
'x',0x00,
'x',0x00,
'x',0x00,
'x',0x00,
'x',0x00,
'x',0x00,
'x',0x00,
'x',0x00,
'x',0x00,

Table 48 – USB Contactless Smart Card Interface String Descriptor

Contact smart card interface string descriptor

String06 (index iInterface of interface descriptor) (Contact card interface)		
Offset	Value	Field
0	0x32	bLength (50 bytes)
1	0x03	bDescriptorType (String)
2	'P',0x00	bString = "Prox-DU Contact_xxxxxxx" or "Prox-SU Contact_xxxxxxx" where xxxxxxxx is the reader serial number printed on the label
	'r',0x00	
	'o',0x00	
	'x',0x00	
	'-',0x00	
	'D',0x00,	
	'U',0x00,	
	' ',0x00,	
	'C',0x00,	
	'o',0x00,	
	'n',0x00,	
	't',0x00,	
	'a',0x00,	
	'c',0x00,	
	't',0x00,	
	'_',0x00,	
	'x',0x00,	
	'x',0x00,	
	'x',0x00,	
	'x',0x00,	
'x',0x00,		
'x',0x00,		
'x',0x00,		
'x',0x00,		
'x',0x00,		
'x',0x00,		
'x',0x00,		

Table 49 – USB Contact Smart Card Interface String Descriptor

Boot-Loader

The boot-loader is used to update the device's firmware and to start the device if the download is successful.

The boot-loader is embedded into the device and is not normally running.

It will be launched using the dedicated HID command "Start Download".

Refer to the "General proprietary command" paragraph for more information.

Hardware requirement

If it is not possible to download a new firmware into the Prox-DU and Prox-SU device because of a failure, it is possible to force the device entering the boot-loader by pressing a push button (S1) located on the main printed circuit board of the device.

It should be used only in case of rescue because it is needed to open the casing of the device to have access to the push button.

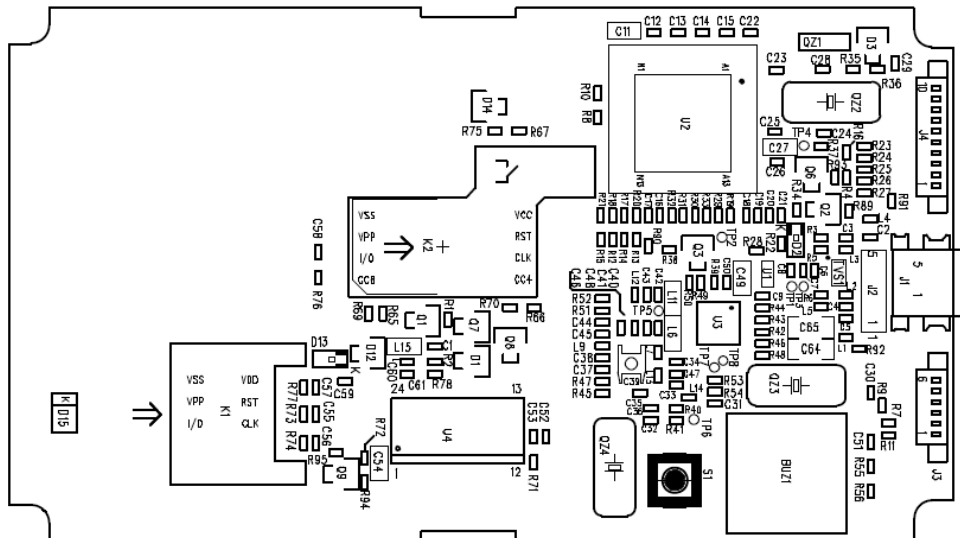


Figure 39 – Push button PCB location (S1)

Boot-loader start up operations

When the USB cable is plugged into the host or when the device is restarted, the boot-loader starts to run. Two cases are possible depending of the push button state:

1 - If the push button is not activated (normal start up),

The boot-loader checks the firmware checksum

- If the checksum is correct, the device is started.
- If the checksum is not correct, the device is not started. (If no firmware is loaded the checksum check will not be correct).

2 - If the push button is activated (rescue start up)

The device is not started and the boot-loader continues to run.

The boot-loader is enumerated by the USB bus and then the download operation can be executed.

At the end of download operation the device firmware is checked (signature):

- If the signature is correct, the boot-loader unplugs itself from the USB bus and the device is restarted. At this time, the push button is normally not activated and the new firmware can start to run like the case 1.
- If the signature is not correct, the device firmware is not started. The download operation can be performed again.

In normal operation, when the device firmware is running, the boot-loader can be started using a proprietary HID command.

This command saves information to tell to the boot-loader that it must not start the device firmware, even if the push button is not activated. The device firmware unplugs itself from the USB bus and the device is restarted. Then following start up operations are executed like the case 2.

The boot-loader is declared as a HID device with one interface. The two interfaces (smart card and contactless) cannot be used while the boot-loader is running.

Note: The boot-loader is write protected and cannot be updated.

Boot-loader download operations

The download operation requires 3 commands:

- A command to start the download
- A command to download the firmware file
- A command to end the process and restart the device

Note: The command to start the download is not used by the boot-loader itself but by the device firmware.

Additional commands are needed to control the download operations:

- Boot-loader version request
- Reset reader

All commands and responses are sent using the HID interface.

Start download command

This command is sent by the host to the reader when the device is operating normally.

Refer to the “General proprietary command” paragraph for more information.

Download firmware file command

This command is used to download the file that contains the reader firmware data.

The file is downloaded in successive packets.

The first packet contains the information needs to program the firmware into the microcontroller and to verify it at the end of downloading operation

The command format is the following:

Offset	Field	Size	Value	Description
0	bCommandFamily	1	53h	Bootloader command
1	bCommandType	1	F9h	Download Management
2	bCommand	1	01h	Load firmware file
3, 4	wLength	2	XXXX	Size of the Data field
5	dwPacketId	4		Packet identifier
9	abDataPacket	256		Firmware file packet (TDES

Prox-DU & Prox-SU

				ciphered) (256 bytes)
--	--	--	--	--------------------------

The response format is the following:

Offset	Field	Size	Value	Description
0,1	abStatus	2	XX XX	Command Status execution
2, 3	wLength	2	0004h	Size of the Data field
4	dwPacketId	4		Packet identifier

Possible Errors:

The returned ulPacketId should have the same value as the received packet.

If the value is different, the host must re-send the ulPacketId (the one which is requested in the response command).

Special case: for the first download packet, the response ulPacketId should be 0 if it succeed, but will be 0xFFFFFFFF (-1 in decimal notation) if it failed.

End download command

This command is the last command of the download process.

While this command is received, the firmware integrity is check using its signature.

If the signature is verified, the ApplicationValid indicator is set and the reader can be restarted.

If the signature is not verified, the reader is not restarted.

The command format is the following:

Offset	Field	Size	Value	Description
0	bCommandFamily	1	53h	Bootloader command
1	bCommandType	1	F9h	Download Management
2	bCommand	1	02h	End download
3, 4	wLength	2	0000h	Size of the Data field

The response format is the following:

Offset	Field	Size	Value	Description
0,1	abStatus	2	XXXX	Command Status execution
2, 3	wLength	2	0000h	Size of the Data field

Possible Errors:

If the download succeeds and the signature computed by the bootloader is the same as the one computed by the Host, abStatus will be set to 0x0000.

Other values (TBD) of abStatus indicate a bad image signature, a bad firmware CRC control or no firmware is present.

Note: If the host sent a "Start download" command followed by an "End download" command (without any download packet), the bootloader should compute the application CRC and compare it to the one fused in EEPROM.

If the two CRC are the same, the ApplicationValid byte is set, the bootloader answers with abStatus = 0x0000 and reboots.

If the two CRC are different, the bootloader answers with an error code and stay in the boot mode.

Boot-loader version request command

This command enables the user to determine the Boot-loader version.

It is the same than firmware version request for the reader. Then the response can be checked to confirm if the boot-loader is running or the device firmware is running.

Reset reader command

This command is used to reset the Prox-DU or Prox-SU device.

The command format is the following:

Offset	Field	Size	Value	Description
0	bCommandFamily	1	52h	Reader command
1	bCommandType	1	F8h	Management
2	bCommand	1	05h	03h = Read current interface switch state 04h = Switch interface
3, 4	wLength	2	0000h	Size of the Data field

The response format is the following:

Offset	Field	Size	Value	Description
No answer for this command				

Boot-loader error codes

In the following table:

- The MSB byte correspond to the first byte reported in the abStatus field of the HID response
- The LSB byte correspond to the second byte reported in the abStatus field of the HID response

Error label	Value	Meaning
ERR_OK	0000h	Execution OK
Wrong command or wrong data parameters (command rejected)		
NOT_READY_TO_UPDATE	FB40h	Wrong command bytes
BAD_FIRMWARE_SIGNATURE_ERROR	FB41h	Wrong data parameters
CRC_ERROR	FB9Bh	Wrong checksum
WRITE_ERROR	FB9Ah	Wrong EEPROM write
Other codes FFxxh not listed are RFU		

Table 50 – Boot-loader HID error codes

Typical Download Operations

The following steps give the list of the commands to be used for a standard download operation:

- The “**Start download**” command is used to initiate the download operation
- The “**Reset reader**” command is used to activate the boot-loader.
- The “**Download firmware file**” command is used several times to send the successive firmware binary packets to the device
- The “**End download**” command is used to complete the download operation
- The “**Reset reader**” command is used to activate the new firmware.

Downloaded File Format

The beginning of the file must contain the information needed to program the firmware and to verify it.

Firmware size (4 bytes): This is the total size of the data firmware that will be downloaded

SHA firmware signature (digest 20 bytes): This is the signature of the firmware to control its integrity after deciphering at the end of the download operation.

The firmware data are following the file header.

The whole file is ciphered using a 3DES key.

The keys to decipher the downloaded file are resident into the boot-loader.

Boot-loader USB descriptors

Device Descriptor

This is the same descriptor than the descriptor for the reader firmware, PID = 0x5502 except.

Configuration Descriptor

The device has only one interface for proprietary commands.

The device is Bus Powered (no external power).

The device doesn't support remote wake.

The maximum power current is 200 mA.

Configuration Descriptor		
Offset	Value	Field
0	0x09	bLength (9 bytes)
1	0x02	bDescriptorType (configuration)
2	0x22 0x00	wTotalLength of the Configuration Descriptor (configuration + interface + endpoint class specific + 1 endpoint) 9 + 9 + 9 + 7 = 34
4	0x01	bNumInterfaces (number of interface = 1)
5	0x01	bConfigurationValue
6	0x00	iConfiguration (ignored)
7	0x80	bmAttributes (b6 = 0 Bus powered, b0 = 1 Remote Wake Up Not supported)
8	0x64	MaxPower (100 x 2 = 200 mA)

Table 51 – USB Boot-loader Configuration Descriptor

Interface descriptor

The interface for the proprietary command is HID.

This is the same descriptor than the HID descriptor for the reader firmware.

Refer to the “USB Descriptors” paragraph for more information.

HID class descriptor

This is the same descriptor than the HID class descriptor for the reader firmware.

Refer to the “USB Descriptors” paragraph for more information.

HID endpoint descriptor

This is the same descriptor than the HID Endpoint descriptor for the reader firmware.

Refer to the “USB Descriptors” paragraph for more information.

HID report descriptor

This is the same descriptor than the HID report descriptor for the reader firmware. Refer to the “USB Descriptors” paragraph for more information.

String descriptors

LangID string descriptor

This is the same descriptor than LangID string descriptor for the reader firmware. Refer to the “USB Descriptors” paragraph for more information.

Manufacturer string descriptor

This is the same descriptor than Manufacturer string descriptor for the reader firmware. Refer to the “USB Descriptors” paragraph for more information.

Product string descriptor

This is the same descriptor than the product string descriptor for the reader firmware. Refer to the “USB Descriptors” paragraph for more information.

Serial number string descriptor

This is the same descriptor than the serial number string descriptor for the reader firmware. Refer to the “USB Descriptors” paragraph for more information.

HID interface string descriptor

String04 (index iInterface of interface descriptor) (HID interface)		
Offset	Value	Field
0	0x2E	bLength (46 bytes)
1	0x03	bDescriptorType (String)
2	'H',0x00,	bString = "HID Download Interface"
	'I',0x00,	
	'D',0x00,	
	' ',0x00,	
	'D',0x00,	
	'o',0x00,	
	'w',0x00,	
	'n',0x00,	
	'l',0x00,	
	'o',0x00,	
	'a',0x00,	
	'd',0x00,	
	' ',0x00,	
	'l',0x00,	
	'n',0x00,	
	't',0x00,	
'e',0x00,		
'r',0x00,		
'f',0x00,		
'a',0x00,		

	'c',0x00,	
	'e',0x00,	

Table 52 – USB Boot-loader Interface String Descriptor

LEDs states for the boot-loader

4 LEDs states are possible for the boot-loader according to the following table:

Yellow LED	Blue LED	State
Off	Off	The reader is not in operation (not powered, in suspend mode or failure)
Blink 1	Blink 1	The boot-loader is ready
Blink 2	Blink 2	The download operation is in progress
Blink 3	Blink 3	The download operation failed
Blink 4	Blink 4	The firmware check operation failed

Table 53 – LEDs states for the Boot-loader LEDs

Blink 1: 250 ms on every 500 ms. (When the yellow LED is on, the blue LED is off and vice-versa)

Blink 2: 100 ms on every 200 ms. (When the yellow LED is on, the blue LED is off and vice-versa)

Blink 3: 50 ms on every 100 ms. (When the yellow LED is on, the blue LED is on).

Blink 4: the yellow LED is on and the blue LED is off for 400 ms every 500 ms.

Downloading a firmware

The Gemalto downloader tool “**Gemalto_Download_Prox.exe**” must be used to download a new firmware into the Prox-DU or Prox-SU device.

The latest firmware binary file to download should be used (.bin extension file).

The two items are available in the following web link <http://support.gemalto.com>.

The Gemalto downloader tool performs the operations listed in the “Typical download operations” with additional commands to display the firmware and the boot-loader string version.

Note: Only **one** Prox-DU or Prox-SU device should be plugged to your computer.

This tool should be used with Windows based operating systems only.

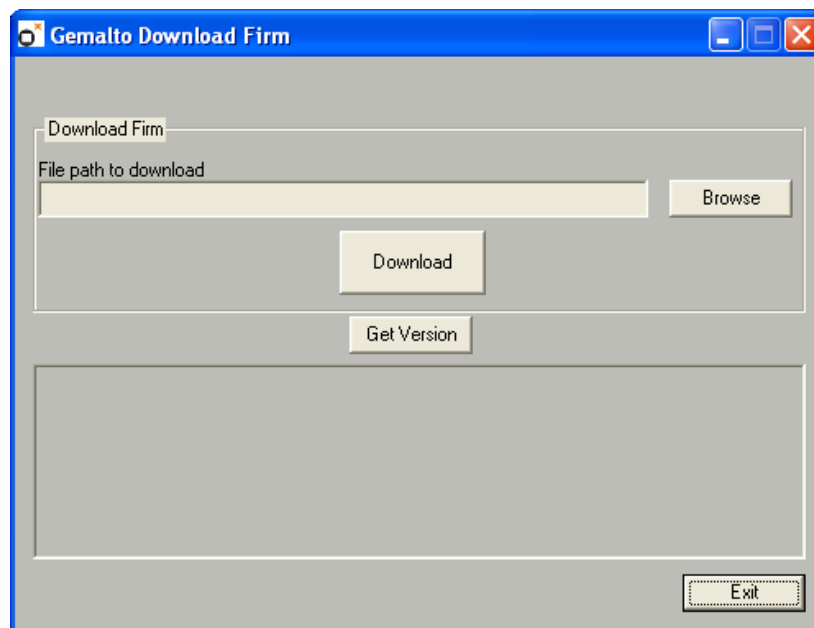
The HID library is not needed because the HID commands are integrated into the tool.

Download tool operations

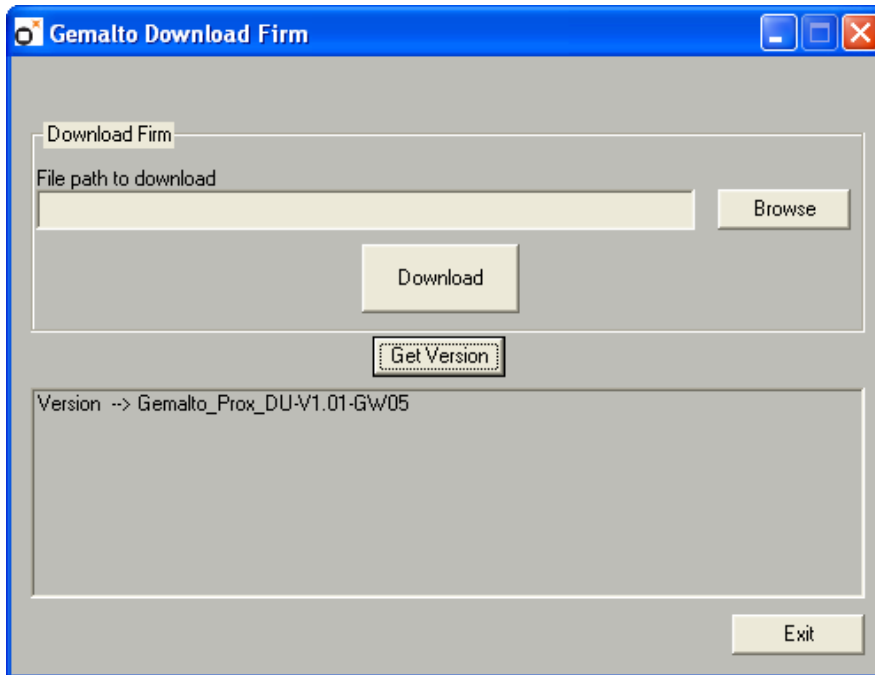
First copy the “Gemalto_Download_Prox.exe” file in a new directory of your computer and copy the last update of the firmware binary file in the same directory (.bin extension file).

The next steps should be performed to download the new firmware into the Prox-DU or Prox-SU device:

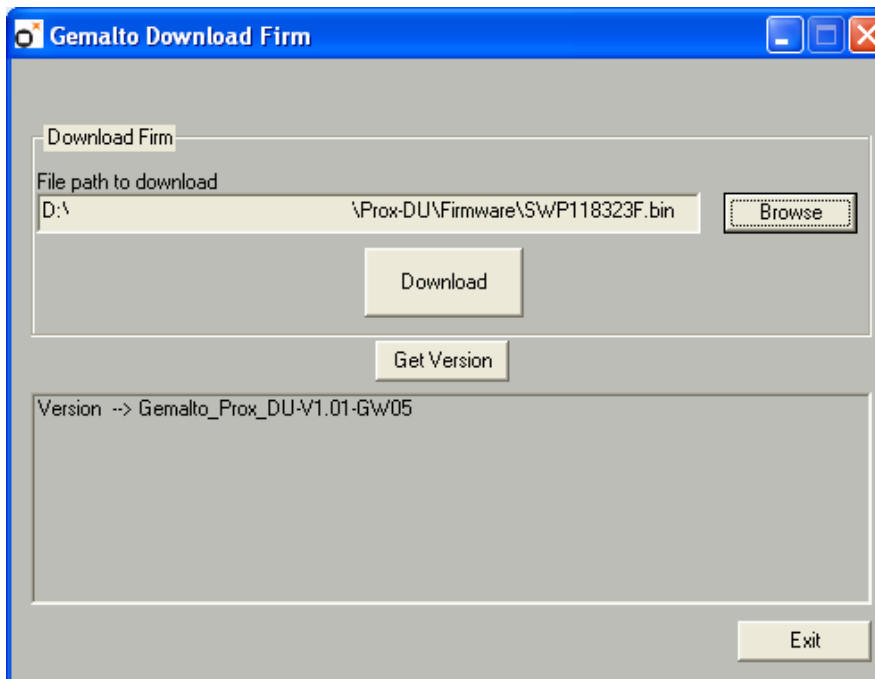
- Run the “**Gemalto_Download_Prox.exe**” file. The following window will be displayed:



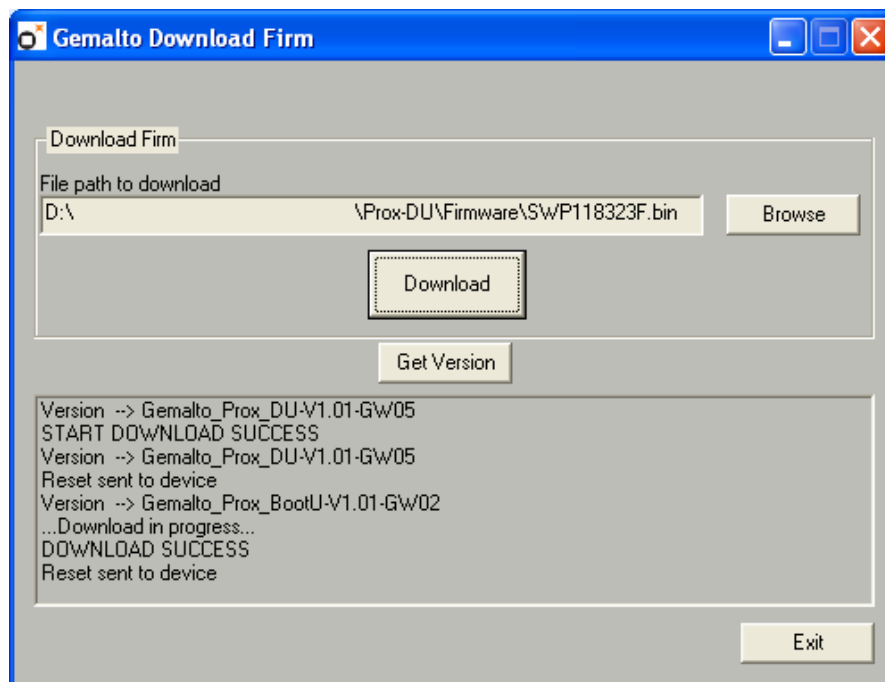
- Click the “**Get Version**” button to retrieve the current string version of the device. This operation will check the communication with the device. The next figure displays the string “Gemato_Prox_DU-V1.01-GW05”:



- Click the **“Browse”** button to indicate the directory where the binary file was previously stored and choose the firmware binary file to download (SWF118323F.bin in the example hereafter):



- Click the **“Download”** button. The download process is running until its termination. The next figure will be displayed:



The “Start Download Success” message is displayed

The current firmware string version of the device is displayed.

Then a reset of the device is performed to start the boot-loader.

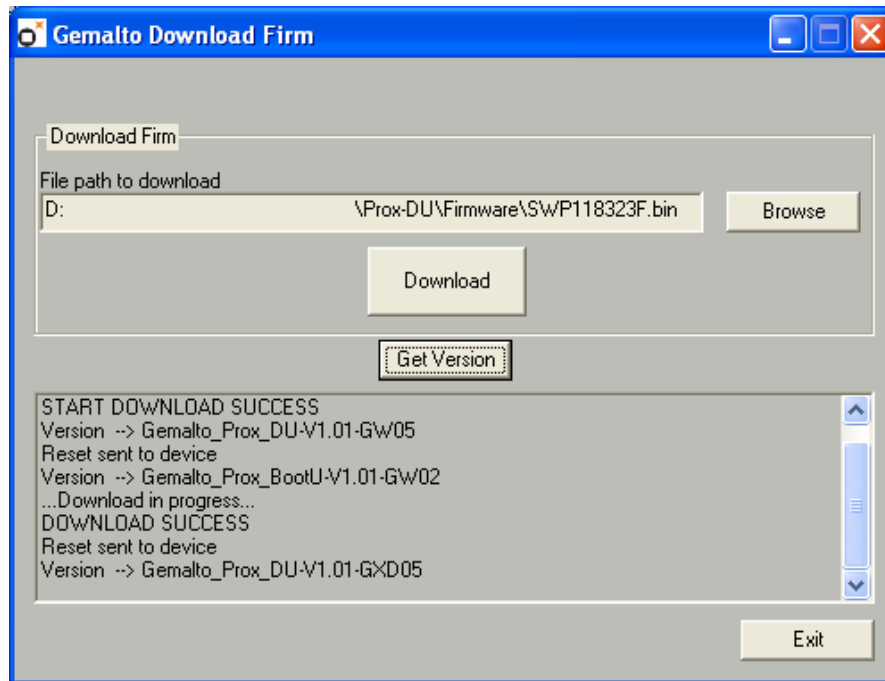
The current boot-loader string version is displayed.

The “...Download in progress...” message is displayed during the download operation.

The “Download Success” message is displayed at the end of the download

Then a reset of the device is performed to start the new firmware.

- Click the “**Get Version**” button to check the new string version of the device. The next figure displays the new string “Gemato_Prox_DU-V1.01-GXD05”:



The download operation is now completed.

Note: the download duration is about 15 seconds.

MIFARE[®] Cards Mapping

MIFARE[®] 1K Memory Mapping

This is a 8-Kbit (1 Kbyte) MIFARE[®] memory contactless smart card arranged as 16 four-block sectors as shown in the following table:

		Bytes																
Sector	Block	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Description
0	0																	Manufacturer Block
	1																	Data
	2																	Data
	3	Key A					Access Bits				Key B					Sector Trailer 0		
1	4																	Data
	5																	Data
	6																	Data
	7	Key A					Access Bits				Key B					Sector Trailer 1		
2	8																	Data
	9																	Data
	10																	Data
	11	Key A					Access Bits				Key B					Sector Trailer 2		
---	---	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	---	
15	60																	Data
	61																	Data
	62																	Data
	63	Key A					Access Bits				Key B					Sector Trailer 15		
Table 54 – Memory Sectors of MIFARE [®] 1K																		

Each contactless smart card consists of a 16-byte memory block assembled in sectors.

The first block of the first sector contains manufacturing information.

The last block of each sector is the sector trailer containing the keys and the access conditions of the blocks.

MIFARE[®] Mini Memory Mapping

This is a 2.5-Kbit (320 bytes) MIFARE[®] memory contactless smart card arranged as 5 four-block sectors as shown in the following table:

Sector	Block	Bytes														Description		
		0	1	2	3	4	5	6	7	8	9	10	11	12	13		14	15
0	0																	Manufacturer Block
	1																	Data
	2																	Data
	3	Key A					Access Bits				Key B					Sector Trailer 0		
1	4																	Data
	5																	Data
	6																	Data
	7	Key A					Access Bits				Key B					Sector Trailer 1		
2	8																	Data
	9																	Data
	10																	Data
	11	Key A					Access Bits				Key B					Sector Trailer 2		
---	---	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	---
4	16																	Data
	17																	Data
	18																	Data
	19	Key A					Access Bits				Key B					Sector Trailer 4		

Table 55 – Memory Sectors of MIFARE[®] Mini

Each contactless smart card consists of a 16-byte memory block assembled in sectors.

The first block of the first sector contains manufacturing information.

The last block of each sector is the sector trailer containing the keys and the access conditions of the blocks.

MIFARE[®] 4K Memory Mapping

This is a 32-Kbit (4 Kbytes) MIFARE[®] memory contactless smart card arranged as 32 four-block sectors and 8 sixteen-block sectors as shown in the following table:

Sector	Block	Bytes														Description		
		0	1	2	3	4	5	6	7	8	9	10	11	12	13		14	15
0	0																	Manufacturer Block
	1																	Data
	2																	Data
	3	Key A					Access Bits				Key B					Sector Trailer 0		
1	4																	Data
	5																	Data
	6																	Data
	7	Key A					Access Bits				Key B					Sector Trailer 1		
2	8																	Data
	9																	Data
	10																	Data
	11	Key A					Access Bits				Key B					Sector Trailer 2		
----	----	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	----	
31	124																	Data
	125																	Data
	126																	Data
	127	Key A					Access Bits				Key B					Sector Trailer 31		
32	128																	Data
	129																	Data
	130																	Data
	131																	Data
	132																	Data
	133																	Data
	134																	Data

Prox-DU & Prox-SU

	135																Data
	136																Data
	137																Data
	138																Data
	139																Data
	140																Data
	141																Data
	142																Data
	143	Key A						Access Bits				Key B				Sector Trailer 32	
----	----	-	-	-	-	-	-	-	-	-	-	-	-	-	-	----	
39	240																Data
	241																Data
	242																Data
	243																Data
	244																Data
	245																Data
	246																Data
	247																Data
	248																Data
	249																Data
	250																Data
	251																Data
	252																Data
253																Data	
254																Data	
255	Key A						Access Bits				Key B				Sector Trailer 39		
Table 56 – Memory Sectors of MIFARE® 4K																	

Each contactless smart card consists of a 16-byte memory block assembled in sectors.

The first block of the first sector contains manufacturing information.

The last block of each sector is the sector trailer containing the keys and the access

conditions of the blocks.

MIFARE[®] UL Memory Mapping

The MIFARE[®] Ultralight chip is a 512-bit EEPROM memory card.

The MIFARE[®] UL memory is organized in 16 pages with 4 bytes each as depicted in the following table:

Byte Number	0	1	2	3	Page
-------------	---	---	---	---	------

Serial Number	SN0	SN1	SN2	BCC0	0
Serial Number	SN3	SN4	SN5	SN6	1
Internal/Lock	BCC1	Internal	Lock0	Lock1	2
OTP	OTP0	OTP1	OTP2	OTP3	3
Data Read-Write	Data0	Data1	Data2	Data3	4
Data Read-Write	Data4	Data5	Data6	Data7	5
Data Read-Write	Data8	Data9	Data10	Data11	6
Data Read-Write	Data12	Data13	Data14	Data15	7
Data Read-Write	Data16	Data17	Data18	Data19	8
Data Read-Write	Data20	Data21	Data22	Data23	9
Data Read-Write	Data24	Data25	Data26	Data27	10
Data Read-Write	Data28	Data29	Data30	Data31	11
Data Read-Write	Data32	Data33	Data34	Data35	12
Data Read-Write	Data36	Data37	Data38	Data39	13
Data Read-Write	Data40	Data41	Data42	Data43	14
Data Read-Write	Data44	Data45	Data46	Data47	15

Bold frame indicates user area.

Table 57 – Memory mapping of MIFARE[®] UL

Serial Number Area

SN0-SN7 is the 7 bytes serial number according to ISO14443-3.

BCC0 and BCC1 are the check bytes according to ISO14443-3.

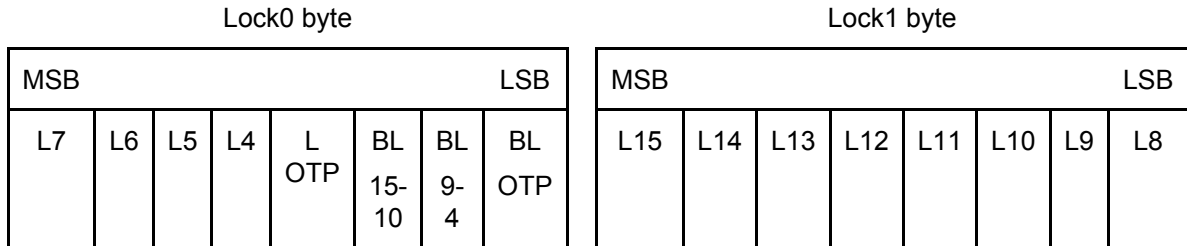
Internal byte is reserved for internal data.

These 10 bytes are write-protected after having been programmed by the chip manufacturer after production.

Lock Bytes Area

Lock0 and Lock1 represent the field-programmable read-only locking mechanism. Each Page x from 3 (OTP) to 15 may be locked individually to prevent further write access by setting the corresponding locking bit Lx to 1. After locking the page is read-only memory.

The 3 least significant bits of lock byte 0 are the block-locking bits. Bit 2 handles pages 15 to 10, bit 1 pages 9 to 4 and bit 0 page 3 (OTP). Once the blocking-locking bits are set the locking configuration for the corresponding memory area is frozen - for example if BL15-10 is set to "1", L15 to L10 (bit 7 to bit 2 of lock byte 2) can no longer be changed.



Lx locks Page x to read-only

BLx blocks further locking for the memory area x

The locking and block-locking bits are set via standard write command to Page 2.

Bytes 2 and 3 of the write command and the actual contents of the lock bytes are bite-wise "OR-ed" and the result then becomes the new contents of the lock bytes.

This process is irreversible. If a bit is set to "1", it cannot be changed back to "0" again.

Note: The content of bytes 0 and 1 of Page 2 is not affected by the corresponding data bytes of the write command.

Warning: To activate the new locking configuration after a write to the lock bit area, a new smart card selection has to be carried out.

OTP Bytes Area

Page 3 is the OTP page. It is pre-set to all "0" (zeros) after production. These bytes may be bit-wise modified by a write command.

The bytes of the write command of the current contents of the OTP bytes are bit-wise "OR-ed" and the result becomes the new contents of the OTP bytes.

This process is irreversible. If a bit is set to "1", it cannot be changed back to "0" again.

Note: This memory area may be used as a 32 ticks one-time counter.

Data Bytes Area

Pages 4 to 15 constitute the user read/write area. After production the data pages are initialized to all "0" (zeroes).

MIFARE[®] UL Read/Write Operation

The MIFARE[®] Ultralight chip does not embed the MIFARE[®] Classic security.

So no authentication operation is required before any read/write operation.

MIFARE[®] Memory Organization

Sector Trailer

The last block of every sector is the sector trailer. It contains the individual secret authentication Key A, optional Key B and the access condition bits for the blocks of the particular sector.

Sector Trailer	Byte	bit 8 MSB	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1 LSB
Secret Key	0	Authentication Key A							
	1								
	2								
	3								
	4								
	5								
Access Bits	6	$\overline{C2_3}$	$\overline{C2_2}$	$\overline{C2_1}$	$\overline{C2_0}$	$\overline{C1_3}$	$\overline{C1_2}$	$\overline{C1_1}$	$\overline{C1_0}$
	7	$C1_3$	$C1_2$	$C1_1$	$C1_0$	$\overline{C3_3}$	$\overline{C3_2}$	$\overline{C3_1}$	$\overline{C3_0}$
	8	$C3_3$	$C3_2$	$C3_1$	$C3_0$	$C2_3$	$C2_2$	$C2_1$	$C2_0$
Data	9								
Secret Key	10	Authentication Key B							
	11								
	12								
	13								
	14								
	15								
CXy: Access bit x for block y $\overline{\quad}$ CXy: Complement of CXy									

Authentication Keys

Each sector contains a six-byte authentication Key A and a six-byte optional Key B. All sectors are assigned to the different applications determined by different system providers.

The mutual authentication procedure is performed between the reader/writer and the contactless card and is driven by the reader/writer. Access to the data stored in a sector is only possible after a successful authentication.

The secret authentication keys are always read as logical "0". In applications using only one authentication key, Key A, user can set the access bits where the memory space of the optional authentication Key B can be used for data storage.

In this case when the authentication key, Key B can no longer be used for authentication,

the card will not allow memory access using Key B.

Access Bits

The access conditions for the specified operations are defined for each block. The sector trailer and the data blocks are controlled independently.

In sectors consisting of four blocks, the access conditions for each individual block are programmable.

In sectors consisting of sixteen blocks, the 15 data blocks are arranged into three groups of five blocks, with the access conditions are defined independently for each group.

Refer to “Access to Data Blocks” Table and “Access to Sector Trailer” Table for the values of these bytes.

The access bits determine the access rights to the memory using the authentication keys A and B. The access conditions may be altered, provided that the relevant key is known and the actual access condition allows this operation.

The following table describes only access bits in the non-inverted mode (although they can be stored in both non-inverted and inverted mode):

Access Bits			Valid Commands	Block	Description
C1 ₀	C2 ₀	C3 ₀	Read, Write, Increment, Decrement, Transfer, Restore	0	Data Block
C1 ₁	C2 ₁	C3 ₁	Read, Write, Increment, Decrement, Transfer, Restore	1	Data Block
C1 ₂	C2 ₂	C3 ₂	Read, Write, Increment, Decrement, Transfer, Restore	2	Data Block
C1 ₃	C2 ₃	C3 ₃	Read, Write	3	Sector Trailer

Table 19 – Access Bits and the Valid Commands

The internal logic of the MIFARE[®] circuit ensures that the commands are executed only after an authentication using either Key A or Key B has been successfully performed.

Note: the “Increment”, “Decrement”, “Transfer”, “Restore” commands are not available using the PC/SC V2 MIFARE[®] commands.

Data Block Access Conditions

The access bits for the data blocks are specified as Never, Key A or Key B.

The setting of the relevant access bits defines the application and the resulting applicable commands. Key A | B indicates that access is possible only after an authentication using Key A or Key B of this sector.

The access condition for every block is dependant on the sector number as explained in the following table:

Sector	Block	Description
N (0 – 31)	0	C3 ₀ - C2 ₀ - C1 ₀
	1	C3 ₁ - C2 ₁ - C1 ₁
	2	C3 ₂ - C2 ₂ - C1 ₂

Prox-DU & Prox-SU

	3	C3 ₃ - C2 ₃ - C1 ₃
N (32 - 39)	0	C3 ₀ - C2 ₀ - C1 ₀
	1	
	2	
	3	
	4	
	5	C3 ₁ - C2 ₁ - C1 ₁
	6	
	7	
	8	
	9	
	10	C3 ₂ - C2 ₂ - C1 ₂
	11	
	12	
	13	
	14	
	15	C3 ₃ - C2 ₃ - C1 ₃
Table 20 – Access Condition for Data Blocks		

The MIFARE[®] system regards authentication Key B as the primary key for access control to the data memory. Operations which are performed with authentication Key A can also be done with authentication Key B. But, only some sensitive operations can be performed with Key B.

The previous Table “Access Bits and the Valid Commands” shows the types of access conditions associated with their bit values and the access granted by authentication with Key A and Key B.

Access Bits			Access Condition Data Block or Superior Block Group b = 0, 1, 2				
C1 _b	C2 _b	C3 _b	Read	Write	Increment	Decremen t/ Transfer/ Restore	Comments
0	0	0	Key A B ¹	Key A B ¹	Key A B ¹	Key A B ¹	A or B All function memory block
0	0	1	Key A B ¹	Never	Never	Key A B ¹	A or B Read /Subtract Value block
0	1	0	Key A B ¹	Never	Never	Never	A or B Read only memory block
0	1	1	Key B ¹	Key B ¹	Never	Never	B Read /Write memory block

Prox-DU & Prox-SU

1	0	0	Key A B ¹	Key B ¹	Never	Never	A or B Read and B Write memory block
1	0	1	Key B ¹	Never	Never	Never	B Read only memory Block
1	1	0	Key A B ¹	Key B ¹	Key B ¹	Key A B ¹	A Read/Subtract B Write/Add Value block
1	1	1	Never	Never	Never	Never	Locked block, Access never allowed
<p>Transport Configuration: When the card is delivered, the access conditions for the sector trailer and the authentication Keys A and B are already containing a particular transport configuration.</p> <p>¹ When Key B can be read in the corresponding Sector trailer, it cannot be used for authentication. If the reader/writer tries to authenticate any block of a sector with Key B using the shaded access conditions, the card will reject subsequent memory access after authentication.</p>							
<p>Table 58 – Access to Data Blocks</p>							

Note: the “Increment”, “Decrement”, “Transfer”, “Restore” commands are not available using the PC/SC V2 MIFARE® commands.

The following describes the functions of the blocks in previous Table “Access Condition for Data Blocks”:

Read/Write Block	The operation read and write are allowed,
Value Block	Allows the additional value operations such as Increment, Decrement, Transfer and Restore. In the case ('001') only Read and Decrement are possible for a non-rechargeable card. In the other case ('110') recharging is possible using Key B.
Manufacturer Block	The read-only condition is not affected by the setting of the access bits.
Key Management	In transport configuration, the use of Key A for authentication is mandatory.

Sector Trailer Access Conditions

The access bits for the sector trailer shown in the following table determine the access condition to either of the authentication keys or the access bits themselves to be Never, Key B, or Key A | B.

Key A | B indicates the access for this sector is only possible after an authentication using either Key A or Key B.

Access Bits			Access Condition			Comments
			Authentication Key A	Access Bits	Authentication Key B	

C1 _b	C2 _b	C3 _b	Read	Write	Read	Write	Read	Write	
0	0	0	Never	Key A	Key A	Never	Key A	Key A	Key B may be read
0	0	1	Never	Key A	Key A	Key A	Key A	Key A	Key B may be read. (Transport configuration)
0	1	0	Never	Never	Key A	Never	Key A	Never	Key B may be read
0	1	1	Never	Key B	Key A B	Key B	Never	Key B	
1	0	0	Never	Key B	Key A B	Never	Never	Key B	
1	0	1	Never	Never	Key A B	Key B	Never	Never	
1	1	0	Never	Never	Key A B	Never	Never	Never	
1	1	1	Never	Never	Key A B	Never	Never	Never	
The shaded areas are access conditions where Key B is readable and may be used for data.									
Table 59 – Access to Sector Trailer									

The access conditions for the sector trailer and Key A are predefined as transport configuration upon card delivery.

As Key B is read in transport configuration, new cards are authenticated with Key A.

Note:

The access bits can also be blocked by the user to prohibit any further changes to the access conditions.

As the access bits can be altered by the user, special care should be taken during personalization phase.

For More Information

Standards and Specifications

- ISO/IEC 14443-1 Identification cards - Contactless ICC- Proximity cards Part 1: Physical characteristics
- ISO/IEC 14443-2 Identification cards - Contactless ICC- Proximity cards Part 2: Radio frequency power and signal interface
- ISO/IEC 14443-2 AMD1 Identification cards - Contactless ICC- Proximity cards Part 2: Radio frequency power and signal interface - Amendment 1: Bit rates of fc/64, fc/32 and fc/16
- ISO/IEC 14443-3 Identification cards - Contactless ICC- Proximity cards Part 3: Initialization and anti collision
- ISO/IEC 14443-3 AMD1 Identification cards - Contactless ICC- Proximity cards Part 3: Initialization and anti collision - Amendment 1: Bit rates of fc/64, fc/32 and fc/16
- ISO/IEC 14443-3 AMD1 COR1 Identification cards - Contactless ICC- Proximity cards Part 3: Initialization and anti collision - Amendment 1: Bit rates of fc/64, fc/32 and fc/16 – Technical Corrigendum 1
- ISO/IEC 14443-3 AMD3 Identification cards - Contactless ICC - Proximity cards Part 3: initialization and anti collision - Amendment 3: Handling of reserved fields and values
- ISO/IEC 14443-4 Identification cards - Contactless ICC- Proximity cards Part 4: Transmission protocol
- ISO/IEC 14443-4 AMD1 Identification cards - Contactless ICC- Proximity cards Part 4: Transmission protocol - Amendment 1: Handling of reserved fields and values
- ISO/IEC 7816-1 Identification cards - Integrated circuits cards with contacts Part 1: Physical characteristics
- ISO/IEC 7816-2 Identification cards - Integrated circuits cards with contacts Part 2: Dimensions and location of the contacts
- ISO/IEC 7816-3 Identification cards - Integrated circuits cards with contacts Part 3: Electronics signals and transmission protocols
- ISO/IEC 7816-4 Identification cards - Integrated circuits cards with contacts Part 4: Organization, security and command for interchange
- Universal Serial Bus - Device Class: Smart Card CCID - Specification for Integrated Circuit(s) Cards Interface Devices - Revision 1.1
- PC/SC V2 specifications: Part 3. Requirements for PC-Connected Interface Devices - Revision 2.01.09

End of Document