



PMA13205-BD
DRAFT - PUBLIC

SPRINGCARD SMART READERS & RFID SCANNERS

Template System Reference Manual

DOCUMENT IDENTIFICATION

Category	Configuration and Software Guide		
Family/Customer	Smart Readers & RFID Scanners		
Reference	PMA13205	Version	BD
Status	draft	Classification	Public
Keywords	RDR, Prox'N'Roll RFID Scanner, FunkyGate, Prox'N'Drive RFID Scanner		
Abstract			

File name	[PMA13205-BD] Smart Readers and RFID Scanners Template System.odt		
Date saved	09/09/16	Date printed	03/06/15

REVISION HISTORY

Ver.	Date	Author	Valid. by		Approv. by	Details
			Tech.	Qual.		
AA	27/09/13	JDA				Created from PMA8P3P
AB	03/04/14	JDA				Documented new template: NDEF, ISO 15693 Added support for Innovision Jewel/Topaz (NFC Forum type 1 Tag) and Kovio RF Barcode in ID Only templated
AC	31/10/14	JDA				Added support for EM4134 Fixed a few typos
BA	21/05/15	JDA				Added support for SpringBlue HCE and Orange NFC Retail
BB	29/04/16	JDA				Title changed for "Smart Readers" Finalisation of SpringBlue HCE Added ASK CTS support for 663-based products, starting in v1.68 Added Desfire EV1 support (to be implemented in v1.69)
BC	14/06/16	JDA				New numbering of chapters for improved readability Added documentation of Orange NFC Office
BD	08/08/16	JDA				Added documentation of SpringBlue BLE

CONTENTS

1. INTRODUCTION.....	6	8.4. LOCATION OF DATA (LOC REGISTER).....	33
1.1. ABSTRACT.....	6	8.5. AUTHENTICATION KEY (AUT REGISTER).....	33
1.2. SUPPORTED PRODUCTS.....	6	9. MIFARE ULTRALIGHT TEMPLATE.....	34
1.3. AUDIENCE.....	7	9.1. LOOKUP LIST (LKL REGISTER).....	34
1.4. SUPPORT AND UPDATES.....	7	9.2. SIZE AND FORMAT OF OUTPUT (TOF REGISTER).....	34
2. PRINCIPLES.....	8	9.3. PREFIX (PFX REGISTER).....	34
2.1. THE TEMPLATE SYSTEM.....	8	9.4. LOCATION OF DATA (LOC REGISTER).....	35
2.2. CONFIGURATION REGISTERS.....	8	10. DESFIRE TEMPLATE.....	36
2.3. EDITING THE CONFIGURATION.....	9	10.1. LOOKUP LIST (LKL REGISTER).....	36
2.4. REGISTERS USED BY THE TEMPLATES.....	9	10.2. SIZE AND FORMAT OF OUTPUT (TOF REGISTER).....	36
3. REFERENCE TABLES.....	10	10.3. PREFIX (PFX REGISTER).....	36
3.1. LIST OF TEMPLATES BY LKL VALUES.....	10	10.4. LOCATION OF DATA (LOC REGISTER).....	37
3.2. LIST OF VALUES FOR LKL BY TEMPLATE.....	12	10.4.1. Reading a StdDataFile (or a BackupDataFile).....	37
3.3. SUMMARY OF CONFIGURATION REGISTERS.....	14	10.4.2. Reading a RecordFile (or a CyclicRecordFile).....	38
4. IMPLEMENTATION MATRIX.....	15	10.5. AUTHENTICATION KEY (AUT REGISTER).....	38
4.1. READERS BASED ON THE CSB6 CORE.....	16	10.5.1. No authentication.....	38
4.2. READERS BASED ON THE H663, K663, E663, S663 CORES.....	17	10.5.2. Authentication EV0 DES/3DES2K.....	39
4.3. READERS BASED ON THE K632 CORE.....	18	10.5.3. Authentication EV1 DES/3DES2K.....	41
5. FEATURES SHARED AMONG TEMPLATES.....	19	10.5.4. Authentication EV1 3DES3K.....	42
5.1. SIZE AND FORMAT OF OUTPUT (TOF REGISTER).....	19	10.5.5. Authentication EV1 AES.....	43
5.1.1. RAW mode.....	20	11. ISO 7816-4 TEMPLATE.....	44
5.1.2. Decimal mode.....	21	11.1. LOOKUP LIST (LKL REGISTER).....	44
5.1.3. Short String mode (up to 16 bytes).....	22	11.2. SIZE AND FORMAT OF OUTPUT (TOF REGISTER).....	45
5.1.4. Long String mode.....	23	11.3. PREFIX (PFX REGISTER).....	45
5.2. PREFIX (PFX REGISTER).....	24	11.4. LOCATION OF DATA (LOC REGISTER).....	45
6. ID-ONLY TEMPLATE.....	25	11.5. ISO 7816-4 APDU 1 (AU1 REGISTER).....	45
6.1. LOOKUP LIST (LKL REGISTER).....	25	11.6. ISO 7816-4 APDU 2 (AU2 REGISTER).....	46
6.2. OUTPUT FORMAT (TOF REGISTER).....	26	11.7. ISO 7816-4 APDU 3 (AU3 REGISTER).....	46
6.3. PREFIX (PFX REGISTER).....	26	12. ISO 15693 MEMORY TEMPLATE.....	47
6.4. LOCATION (LOC REGISTER).....	27	12.1. LOOKUP LIST (LKL REGISTER).....	47
6.5. MISCELLANEOUS OPTIONS (OPT REGISTER).....	27	12.2. SIZE AND FORMAT OF OUTPUT (TOF REGISTER).....	47
6.6. USING THE ID-ONLY TEMPLATE WITH NON-ISO PICCs.....	28	12.3. PREFIX (PFX REGISTER).....	47
7. MIFARE CLASSIC TEMPLATE.....	29	12.4. LOCATION OF DATA (LOC REGISTER).....	48
7.1. LOOKUP LIST (LKL REGISTER).....	29	13. EM4134 TEMPLATE.....	49
7.2. SIZE AND FORMAT OF OUTPUT (TOF REGISTER).....	29	13.1. LOOKUP LIST (LKL REGISTER).....	49
7.3. PREFIX (PFX REGISTER).....	29	13.2. SIZE AND FORMAT OF OUTPUT (TOF REGISTER).....	49
7.4. LOCATION OF DATA (LOC REGISTER).....	30	13.3. PREFIX (PFX REGISTER).....	49
7.4.1. Using an AID in the MAD.....	30	13.4. LOCATION OF DATA (LOC REGISTER).....	50
7.4.2. Using an absolute block address.....	30	14. NDEF DATA.....	51
7.5. AUTHENTICATION KEY (AUT REGISTER).....	31	14.1. LOOKUP LIST (LKL REGISTER).....	51
8. MIFARE PLUS SL3 TEMPLATE.....	32	14.2. SIZE AND FORMAT OF OUTPUT (TOF REGISTER).....	52
8.1. LOOKUP LIST (LKL REGISTER).....	32	14.3. PREFIX (PFX REGISTER).....	52
8.2. SIZE AND FORMAT OF OUTPUT (TOF REGISTER).....	32	14.4. TYPE NAME AND FORMAT (TNF REGISTER).....	53
8.3. PREFIX (PFX REGISTER).....	32	14.5. TYPE (TYP REGISTER).....	53

14.6.USING TNF AND TYP REGISTERS TO SELECT THE DATA FROM AN NDEF RECORD.....	53
14.6.1.Reading a URI.....	53
14.6.2.Reading a Text.....	54
14.6.3.Reading a SpringCard data entry.....	54
14.6.4.Reading a custom data entry.....	54
15.SPRINGBLUE HCE.....	55
15.1.LOOKUP LIST (LKL REGISTER).....	55
15.2.SIZE AND FORMAT OF OUTPUT (TOF REGISTER).....	55
15.3.PREFIX (PFX REGISTER).....	55
15.4.LOCATION OF DATA (LOC REGISTER).....	56
15.5.AUTHENTICATION KEY (AUT REGISTER).....	56
16.ORANGE NFC APIS (RETAIL).....	57
16.1.LOOKUP LIST (LKL REGISTER).....	57
16.2.SIZE AND FORMAT OF OUTPUT (TOF REGISTER).....	57
16.3.PREFIX (PFX REGISTER).....	57
16.4.FIELD SELECT AND OFFSETS (LOC REGISTER).....	58
17.ORANGE NFC API (OFFICE).....	59
17.1.LOOKUP LIST (LKL REGISTER).....	59
17.2.SIZE AND FORMAT OF OUTPUT (TOF REGISTER).....	59
17.3.PREFIX (PFX REGISTER).....	59
17.4.FIELD SELECT AND OFFSETS (LOC REGISTER).....	60
17.5.APPLICATION ID.....	60
17.6.ZONE ID.....	60
17.7.MASTER KEY.....	60
18.SPRINGBLUE BLE.....	61
18.1.ENABLING THE SPRINGBLUE BLE APPLICATION.....	61
18.2.SIZE AND FORMAT OF OUTPUT (TOF REGISTER).....	61
18.3.PREFIX (PFX REGISTER).....	61
18.4.LOCATION OF DATA (LOC REGISTER).....	62
18.5.AUTHENTICATION KEY (AUT REGISTER).....	62

1. INTRODUCTION

1.1. ABSTRACT

SpringCard offers Access Control Readers, OEM Readers and PC-connected Readers dedicated to the automated processing of contactless smart cards and RFID labels or tags:

- The **FunkyGate** family: wall-mounted access control readers, available with either Data+Clock, Wiegand, serial RS-232, serial RS-485, serial emulation on top of USB, and TCP over Ethernet communication options,
- The **RFID Scanner** family: USB products for the desktop operating in keyboard emulation mode (“wedge”)
- The **RDR** family: a wide range of OEM Readers featuring various communication options (RS-232, RS-TTL, RS-485, serial emulation on top of USB).

All these families share a large part of their feature, the core being their exclusive **Template System**, which allows them to accept mixed types of cards or tags, and to fetch virtually any kind of data from anywhere on the card or tag.

This document provides all necessary information to perform a low-level configuration of the **Templates** into a **SpringCard FunkyGate, RFID Scanner, or RDR Reader**.

1.2. SUPPORTED PRODUCTS

At the time of writing, this document refers to:

- **Prox'N'Roll RFID Scanner**: desktop USB reader, operating in keyboard emulation mode,
- **K632/RDR, K632/RDR-TTL, K632/RDR-232**: a standalone OEM Reader based on the SpringCard **K632** hardware,
- **K663/RDR, K663/RDR-TTL, K663/RDR-232**: a standalone OEM Reader based on the SpringCard **K663** hardware,
- **Prox'N'Drive/RDR**: a Reader for automotive applications, based on the SpringCard **K663** core,
- **FunkyGate-DW, FunkyGate-DW NFC, FunkyGate-DW NFC+BLE**: a wall-mounted Access Control Reader with selectable Data+Clock, Wiegand or RS-485 interface, the latter featuring a BLE interface too,
- **FunkyGate-IP NFC, FunkyGate-IP+POE NFC**: a wall-mounted Access Control Reader with TCP over Ethernet interface,
- **TwistyWrite-IP/RDR**: an OEM Access Control Reader with TCP over Ethernet interface and a remote antenna,

1.3. AUDIENCE

This manual is designed for use by application developers and system integrators. It assumes that the reader has a good knowledge of computer development and a good knowledge of the RFID/NFC technologies.

1.4. SUPPORT AND UPDATES

Useful related materials (product datasheets, application notes, sample software, HOWTOs and FAQs...) are available at SpringCard's web site:

www.springcard.com

Updated versions of this document and others are posted on this web site as soon as they are available.

For technical support enquiries, please refer to SpringCard support page, on the web at

www.springcard.com/support

2. PRINCIPLES

2.1. THE TEMPLATE SYSTEM

SpringCard Smart Readers & RFID/NFC Scanners are able to “read” different types of cards, and to access different sources of data of each cards.

A **Template** tells the reader

- Which type of PICC/VICC it shall look for,
- What and where is the data to fetch: protocol-defined “serial number” or data stored in PICC/VICC's memory, what is the authentication key is the data is protected...
- How to format the data when sending it to the target system (is it an ASCII string, a decimal number, or raw data that must be transmitted in hexadecimal for readability?)

Most readers are able to run up to **4 Templates**. When a PICC/VICC is presented in front of the reader, the reader tries its **Template** one after the other, until it succeeds getting some data from the PICC/VICC.

This means a single reader could return data from 1 to 4 different types of PICC/VICC, yet silently ignoring the ones that don't match any of the **Templates**.

The **Templates** are stored among other runtime parameters in the **reader's configuration registers** introduced below.

2.2. CONFIGURATION REGISTERS

The configuration is stored in a set of non-volatile¹ Configuration Registers, numbered $_{h}01$ to $_{h}FE$. There are two groups of Registers:

- The Registers that control the global behaviour of the Reader are fully documented in the product's technical manual itself.
- The Registers that control the Template System are shared among all SpringCard Smart Readers, and are documented in this manual.

When the reader starts, it loads its configuration from the registers present in its configuration memory. If a register is not present (never defined, or erased by a configuration tool), the reader uses the factory default value assigned to this register.

Changing the configuration of a reader means writing (or sometimes erasing) configuration registers into the reader's memory.

¹ The physical storage in either an E2PROM or a DATA FLASH. Please refer to the product's manual to know the write endurance of its configuration memory. Rewriting the configuration more times than specified may permanently damage the product.

2.3. EDITING THE CONFIGURATION

There are many ways to edit the Reader's Configuration Registers, depending on the Reader's hardware and specification:

1. Using the Console, either through a serial link for Readers featuring a serial communication port (RS232, RS484, serial-over-USB), or through the network for TCP/IP Readers with a Telnet server onboard,
2. Using the USB stream for RFID Scanners,
3. Using a Master Card,
4. Using a NFC mobile phone.

Please refer to the actual product's manual to know which method(s) your reader supports. If a dedicated configuration software is available for your reader, using this software is the preferred method to change the reader's configuration.

The new **SpringCard Configuration Tool** software (**ScMultiConf.exe, ref # SN14007**) for Windows makes it easy to edit the configuration of all the Readers.

2.4. REGISTERS USED BY THE TEMPLATES

Templates are numbered 1, 2, 3, 4.

Every Template uses 1 to 15 configuration registers, detailed in the next chapters.

The configuration registers belonging to a given Template are numbered

(template number << 4) + (index of configuration register within the template)

Therefore the configuration registers are

- h_{10} to h_{1F} for Template 1 (we call h_{10} the *base address* for Template 1)
- h_{20} to h_{2F} for Template 2 (we call h_{20} the base address for Template 2)
- h_{30} to h_{3F} for Template 3 (we call h_{30} the base address for Template 3)
- h_{40} to h_{4F} for Template 4 (we call h_{40} the base address for Template 4)
- (and so on if a reader has more than 4 Templates)

3. REFERENCE TABLES

3.1. LIST OF TEMPLATES BY LKL VALUES

LKL	Supported PICCs/VICCs	Template(s)	Chapter
h01	ISO 14443 type A (up to layer 3)	ID Only	6
h02	ISO 14443 type B (up to layer 3)		
h03	ISO 14443 (A & B) (up to layer 3)		
h04	ISO 15693		
h07	ISO 14443 (A & B) and ISO 15693		
h08	NXP ICODE1		
h0C	ISO 15693 and NXP ICODE1		
h0F	All of the above		
h11	ISO 14443 type A (up to layer 4 "T=CL")	ISO 7816-4	11
h12	ISO 14443 type B (up to layer 4 "T=CL")		
h13	ISO 14443 (A & B) (up to layer 4 "T=CL")		
h20	Kovio RF Barcode	ID Only	6
h21	Innovision Topaz/Jewel		
h22	ST MicroElectronics SR family		
h23	ASK CTS256B and CTS512B		
h24	Inside Secure PicoTag (including HID iClass)		
h28	Felica		
h40	Receive NDEF by SNEP (peer-to-peer)	NFC Forum NDEF Data	14
h41	Read NDEF from NFC Forum type 1 Tags		
h42	Read NDEF from NFC Forum type 2 Tags		
h43	Read NDEF from NFC Forum type 3 Tags		
h44	Read NDEF from NFC Forum type 4 Tags (A & B)		
h4A	Read NDEF from NFC Forum type 4A Tags		
h4B	Read NDEF from NFC Forum type 4B Tags		
h4E	Read NDEF from any NFC Forum Tag		
h4F	Read NDEF from any NFC Forum Tag and Receive NDEF by SNEP (peer-to-peer)		
h54	ISO 15693	ISO 15693 Memory	12
h56	EM4134	EM4134 Memory	13

h61	NXP Mifare Classic 1K & 4K	Mifare Classic	7
h62	NXP Mifare UltraLight	Mifare UltraLight	9
h63	NXP Mifare Plus 2K & 4K, S & X, in SL3	Mifare Plus SL3	8
h71	NXP Desfire	Desfire	10
h72	Innovatron Radio Protocol (deprecated Calypso cards)	ID Only	6
		ISO 7816-4	11
hB0	SpringBlue HCE (card emulation)	SpringBlue HCE	15
hC0	NFC phone with Orange Retail applet in the SIM	Orange NFC Retail	16
hC1	NFC phone with Orange Office application (HCE)	Orange NFC Office	17
hFF	Accept all supported PICCs/VICCs	ID Only	6

3.2. LIST OF VALUES FOR LKL BY TEMPLATE

Template	Chapter	LKL	Supported PICCs/VICCs
ID Only	6	h01	ISO 14443 type A (up to layer 3)
		h02	ISO 14443 type B (up to layer 3)
		h03	ISO 14443 (A & B) (up to layer 3)
		h04	ISO 15693
		h07	ISO 14443 (A & B) and ISO 15693
		h08	NXP ICODE1
		h0C	ISO 15693 and NXP ICODE1
		h0F	All of the above
		h20	Kovio RF Barcode
		h21	Innovision Topaz/Jewel
		h22	ST MicroElectronics SR family
		h23	ASK CTS256B and CTS512B
		h24	Inside Secure PicoTag (including HID iClass)
		h28	Felica
		h72	Innovatron Radio Protocol (deprecated Calypso cards)
hFF	Accept all supported PICCs/VICCs		
NDEF data	14	h40	Receive NDEF by SNEP (peer-to-peer)
		h41	Read NDEF from NFC Forum type 1 Tags
		h42	Read NDEF from NFC Forum type 2 Tags
		h43	Read NDEF from NFC Forum type 3 Tags
		h44	Read NDEF from NFC Forum type 4 Tags
		h4A	Read NDEF from NFC Forum type 4A Tags
		h4B	Read NDEF from NFC Forum type 4B Tags
		h4E	Read NDEF from any NFC Forum Tag and Receive NDEF by SNEP (peer-to-peer)
ISO 15693 Memory	12	h54	ISO 15693
EM4134 Memory	13	h56	EM4134
Mifare Classic	7	h61	NXP Mifare Classic 1K & 4K
Mifare UltraLight	9	h62	NXP Mifare UltraLight
Mifare Plus SL3	8	h63	NXP Mifare Plus 2K & 4K, S & X, in SL3
Desfire	10	h71	NXP Desfire

ISO 7816-4	11	h11	ISO 14443 type A (up to layer 4 "T=CL")
		h12	ISO 14443 type B (up to layer 4 "T=CL")
		h13	ISO 14443 (A & B) (up to layer 4 "T=CL")
		h72	Innovatron Radio Protocol (deprecated Calypso cards)
SpringBlue HCE	15	hB0	Support SpringBlue HCE
Orange NFC Retail	16	hC0	Support Orange NFC Retail
Orange NFC Office	17	hC1	Support Orange NFC Office

3.3. SUMMARY OF CONFIGURATION REGISTERS

Addr. base +	Template				
	ID Only	Mifare UL ISO 15693 mem EM4134 mem	Mifare Classic Mifare Plus Desfire	ISO 7816-4	NDEF data
h00	Lookup List (LKL)				
h01	Size and format of output (TOF)				
h02	Output prefix (PFX)				
h03	Offset (LOC)	Location of data (LOC)		Offset (LOC)	Offset (LOC)
h04	Options (OPT)				
h05			Auth. (AUT)	APDU 1 (AU1)	TNF (TNF)
h06				APDU 2 (AU2)	Type (TYP)
h07				APDU 3 (AU3)	

The **base** address is:

- h10 for the 1st Template
- h20 for the 2nd Template
- h30 for the 3rd Template
- h40 for the 4th Template
- (and so on for readers having more than 4 Templates)

Example: suppose you want to read a Mifare Classic PICC using the 1st Template. You'll put the LKL value specified for Mifare Classic cards (h61) into register h10, the location of data (LOC) into register h13, and so on.

4. IMPLEMENTATION MATRIX

Pay attention that some hardware doesn't support all the protocols or access modes depicted in this document. On the other hand, new features are introduced regularly in the embedded software (firmware); therefore, old versions of the firmware may lack some features.

The **implementation matrix** below gives an overview of what is supported (or not) by every reader family.

4.1. READERS BASED ON THE CSB6 CORE

There's only one Reader in this family:

- **Prox'N'Roll RFID Scanner**

Feature	See §	Supported?
NXP ICODE1 in ID-only template	6.1	Yes
Sony Felica in ID-only template		No
Kovio RF Barcode in ID-only template	6.1	Firmware ≥ 1.56
Innovision Topaz/Jewel, NFC Forum type 1 tag in ID-only template	6.1	Firmware ≥ 1.56
ASK CTS 256B and CTS512B in ID-only template	6.1	Yes
Mifare Plus in SL3		No
Desfire EVO authentication	10.5	Yes
Desfire EV1 authentication		No
Long string in Templates: Mifare Classic, Mifare UltraLight, Mifare Plus, Desfire EVO and Desfire EV1	5.1.4	Firmware ≥ 1.40
Not-byte-aligned data (shift bits left feature) in Templates:		
- Mifare Classic and Mifare Plus	7.4	Firmware ≥ 1.45
- Desfire	10.4	Firmware ≥ 1.45
- ISO 7816-4	11.4	Firmware ≥ 1.44
Data from ISO 15693 tags	12	Firmware ≥ 1.56
Data from EM4134 tags	13	Firmware ≥ 1.60
NDEF data from NFC Forum Tags	14	Firmware ≥ 1.56 types 1, 2 & 4 only
NDEF data in NFC peer-to-peer mode (SNEP over LLCP)		No
SpringBlue HCE	15	Firmware ≥ 1.63
Orange NFC APIs (Retail)		No
Orange NFC APIs (Office)		No

4.2. READERS BASED ON THE H663, K663, E663, S663 CORES

Readers in this family are:

- K663/RDR, K663/RDR-232, K663/RDR-TTL, Prox'N'Drive/RDR
- H663/RDR, H663/RDR-USB
- E663/RDR, FunkyGate-IP NFC, FunkyGate-IP+POE NFC, TwistyWriter-IP/RDR
- S663/RDR, FunkyGate-DW NFC, FunkyGate-DW NFC+BLE

Feature	See §	Supported?
NXP ICODE1 in ID-only template		No
Sony Felica in ID-only template	6.1	Yes
Kovio RF Barcode in ID-only template	6.1	Yes
InnoVision Topaz/Jewel, NFC Forum type 1 tag in ID-only template	6.1	Firmware ≥ 1.57
ASK CTS 256B and CTS512B in ID-only template	6.1	Firmware ≥ 1.68
Mifare Plus in SL3	8	Firmware ≥ 1.57
Desfire EVO authentication	10.5	Yes
Desfire EV1 authentication	10.5	Firmware ≥ 1.69
Long string in Templates: Mifare Classic, Mifare UltraLight, Mifare Plus, Desfire EVO and Desfire EV1	5.1.4	Yes
Not-byte-aligned data (shift bits left feature) in Templates:		
- Mifare Classic and Mifare Plus	7.4	Yes
- Desfire	10.4	Yes
- ISO 7816-4 templates	11.4	Yes
Data from ISO 15693 tags	12	Firmware ≥ 1.57
Data from EM4134 tags	13	Firmware ≥ 1.60
NDEF data from NFC Forum Tags	14	Firmware ≥ 1.57
NDEF data in NFC peer-to-peer mode (SNEP over LLCP)	14	Custom firmware only
SpringBlue HCE	15	Firmware ≥ 1.63
Orange NFC APIs (Retail)	16	Firmware ≥ 1.63
Orange NFC APIs (Office)	17	Firmware ≥ 1.65

4.3. READERS BASED ON THE K632 CORE

Readers in this family are:

- K632/RDR, K632/RDR-232, K632/RDR-TTL
- FunkyGate-DW, FunkyGate-SU

Feature	See §	Supported?
NXP ICODE1 in ID-only template	6.1	Yes
Sony Felica in ID-only template		No
Kovio RF Barcode in ID-only template		No
Innovision Topaz/Jewel in ID-only template		No
ASK CTS 256B and CTS512B in ID-only template	6.1	Yes
Mifare Plus in SL3		No
Desfire EVO authentication	10.5	Yes
Desfire EV1 authentication	10.5	No
Long string in Templates: Mifare Classic, Mifare UltraLight, Mifare Plus, Desfire EVO and Desfire EV1	5.1.4	Firmware ≥ 1.40
Not-byte-aligned data (shift bits left feature) in Templates:		
- Mifare Classic and Mifare Plus		No
- Desfire		No
- ISO 7816-4 templates		No
Data from ISO 15693 tags		No
Data from EM4134 tags		No
NDEF data from NFC Forum Tags		No
NDEF data in NFC peer-to-peer mode (SNEP over LLCP)		No
SpringBlue HCE		No
Orange NFC APIs (Retail)		No
Orange NFC APIs (Office)		No

5. FEATURES SHARED AMONG TEMPLATES

Some configuration registers are common to most or all templates. They are listed in this chapter.

5.1. SIZE AND FORMAT OF OUTPUT (TOF REGISTER)

The **TOF** register defines the size and the format of the output. It allows to choose between three primary modes

- **RAW**: for readers that are able to transmit digits and letters only (Serial MK1 protocol, RFID Scanner running in keyboard emulation...), the data is expressed in hexadecimal. For other readers (Serial MK2, Wiegand...), the data is transmitted exactly 'as is'.
- **Decimal**: the data is understood as a number. This number is expressed in decimal, and then transmitted. This mode is suitable for all protocols, including DataClock.
- **String**: the data is transmitted exactly 'as is', and is assumed to be made of ASCII letters and digits only. This implies that the actual data stored onto the card must obey to the reader's output rules -otherwise the reader will discard them.

For instance, if the string holds only ASCII-expressed digits, it could be transmitted through DataClock, but any non-digit value will be removed from the output. On a RFID Scanner running in keyboard emulation, non-ASCII characters will have unpredictable behaviour, varying with the OS receiving the key-stroke.

The **String** mode has two minor sub-modes:

- **Short String**: this mode is suitable for fixed-length messages, with a length under 16 characters. It is supported by all readers.
- **Long String**: this mode is suitable for variable-length messages, up to a few hundreds of characters (vary with the product and firmware version). The messages shall be null-terminated ($\text{h}00$ value). *This mode is not supported by all readers, and can't be used with Wiegand, DataClock and Serial MK2 output.*

The definition of the **TOF** register is shared among all templates but **ID-only** and **NDEF data**.

5.1.1. RAW mode

In **RAW** mode, the data could take any value, and no interpretation is performed.

For readers able to transmit arbitrary data, the data is transmitted exactly 'as it'. For readers that are able to transmit valid ASCII characters only, the data is transmitted in hexadecimal format².

Tip: this format is also suitable for numbers stored in BCD

Common TOF register, RAW mode – Address: base + _n01, size: 1 byte

Bits	Value	Meaning
Byte swapping		
7	b0	Direct order (use the data bytes as they are returned by the card)
	b1	Reverse order (swap the data bytes before sending)
Mode (raw/decimal or string)		
6	b0	Raw mode → must be b0
Padding (applicable if read length < specified output length)		
5	b0	Padd using '0' char on the left
	b1	Padd using 'F' char on the right
Strip leading zeroes		
4	b0	Do not remove leading '0' chars
	b1	Remove leading '0' chars
Size of output		
3-0	b0001	4 bytes
	b0002	8 bytes
	b0011	5 bytes
	b0100	10 bytes
	b0101	7 bytes
	b0110	11 bytes
	b1000	16 bytes
	b1001	20 bytes
	b1010	24 bytes
	b1011	32 bytes

² The reader uses uppercase for digits A to F.

5.1.2. Decimal mode

In **decimal** mode, the data could take any value. The value is understood as a number and expressed in decimal.

Common TOF register, decimal mode – Address: base + $_{h}01$, size: 1 byte

Bits	Value	Meaning
Byte swapping		
7	$_{b}0$	Direct order (use the data bytes as they are returned by the card)
	$_{b}1$	Reverse order (swap the data bytes before processing)
Mode (raw/decimal or string)		
6	$_{b}0$	Raw mode → must be $_{b}0$
Padding		
5	$_{b}0$	must be $_{b}0$
Strip leading zeroes		
4	$_{b}0$	Do not remove leading '0' digits
	$_{b}1$	Remove leading '0' digits
Output length		
3-0	$_{b}0000$	10 digits <i>4 bytes are transmitted in decimal</i>
	$_{b}1100$	12 digits <i>5 bytes are transmitted in decimal</i>
	$_{b}1101$	13 digits <i>- " -</i>
	$_{b}1110$	Unlimited <i>16 bytes are transmitted in decimal</i>

5.1.3. Short String mode (up to 16 bytes)

In **Short String** mode, only valid ASCII characters shall be stored on the card. The reader transmits the letters until either a zero value is read ($_{h}00$, i.e. '\0' i.e. the “end of string” char) or the specified length is reached.

Common TOF register, Short String mode – Address: base + $_{h}01$, size: 1 byte

Bits	Value	Meaning
Byte swapping		
7	$_{b}0$	Direct order (use the data bytes as they are returned by the card)
	$_{b}1$	Reverse order (swap the data bytes before sending)
Mode (raw/decimal or string)		
6	$_{b}1$	String mode → must be $_{b}1$
Fixed length / Variable length with padding		
5	$_{b}0$	Variable length (no padding)
	$_{b}1$	Padd with ' ' char (SPACE) on the right until the max output length is reached
Short/long string mode		
4	$_{b}0$	Short string mode → must be $_{b}0$
Output length (max)		
3-0	$_{b}0000$	16 characters
	$_{b}0001$	1 character
	$_{b}0010$	2 characters
	$_{b}0011$	3 characters
	$_{b}0100$	4 characters
	$_{b}0101$	5 characters
	$_{b}0110$	6 characters
	$_{b}0111$	7 characters
	$_{b}1000$	8 characters
	$_{b}1001$	9 characters
	$_{b}1010$	10 characters
	$_{b}1011$	11 characters
	$_{b}1100$	12 characters
	$_{b}1101$	13 characters
	$_{b}1110$	14 characters
	$_{b}1111$	15 characters

5.1.4. Long String mode

In short string mode, the block must store only valid ASCII bytes. The reader transmits the letters until either a zero value is read ($_{h}00$, i.e. '\0' i.e. the “end of string” char) or the specified length is reached.

Not all readers support this mode, please check the Implementation Matrix (page 15). This mode can't be used with Wiegand, DataClock and Serial MK2 output modes.

Common TOF register, Long String mode – Address: base + $_{h}01$, size: 2 bytes

Bits	Value	Meaning
Byte 0		
Byte swapping		
7	$_{b}0$ $_{b}1$	Direct order (use the data bytes as they are returned by the card) Reverse order (swap the data bytes before sending)
Mode (raw/decimal or string)		
6	$_{b}1$	String mode → must be $_{b}1$
Fixed length / Variable length with padding		
5	$_{b}0$ $_{b}1$	Variable length (no padding) Padd with ' ' char (SPACE) on the right until the max output length is reached
Short/long string mode		
4	$_{b}1$	Long string mode → must be $_{b}1$
(unused)		
3-0	$_{b}0000$	Must be $_{b}0000$
Byte 1		
Max output length		
7-0	$_{h}01$ to $_{h}FF$	From 1 to 255 chars (Most readers are actually limited to 240 chars, some to 64, please check with the reader's datasheet)

5.2. PREFIX (PFX REGISTER)

The PFX register stores a constant value that the reader will use to prefix the data. This is typically used to discriminate among templates (template 1 may use '1:' as prefix, template 2 '2:' and so on...)?

PFX for ID-only – base + $\text{h}02$, size: 0 to 8 bytes

Uses the PFX register to transmit an arbitrary (constant) string before the data returned by this Template.

Make sure the constant is suitable to be transmitted in the reader's output mode:

- For a reader using either Wiegand or Serial MK2 protocol, this field must be a valid hexadecimal string,
- For a reader using the DataClock protocol, this field must be a valid decimal string,
- For a reader using the Serial MK1 protocol, this field must hold only printable ASCII chars,
- For a RFID Scanner running in keyboard emulation mode, this field must hold only printable ASCII chars or chars that map to special keys on the keyboard (for instance value $\text{h}09$ e.g. '\t' for TAB key).

6. ID-ONLY TEMPLATE

Use the **ID-Only Template** to fetch the serial number and some of the protocol-related constants from PICCs/VICCs.

Depending on the setting you define in the Template's LKL register, the reader may either

- Process any supported PICC or VICC,
- Process only a specific family of PICC or VICC.

6.1. LOOKUP LIST (LKL REGISTER)

LKL for ID-only – address: base + h00, size: 1 byte

Value	Meaning	Notes
h01	Accept ISO 14443 type A PICCs	
h02	Accept ISO 14443 type B PICCs	
h03	Accept ISO 14443 type A and type B PICCs	
h04	Accept ISO 15693 VICCs	
h07	Accept ISO 14443 type A and type B PICCs and ISO 15693	
h08	Accept NXP ICODE1 VICCs	A
h0C	Accept ISO 15693 and NXP ICODE1 VICCs	A
h0F	Accept all of the above	
h20	Accept Kovio RF Barcode family	
h21	Accept Innovision Topaz/Jewel family	
h22	Accept ST MicroElectronics SR family	
h23	Accept ASK CTS256B and CTS512B	B
h24	Accept Inside Secure PicoTag (including HID iClass)	
h28	Accept Sony Felica Family	C
h72	Accept Innovatron Radio Protocol (deprecated Calypso cards)	
hFF	Accept all supported PICCs/VICCs	

Notes (see the Implementation Matrix starting page 15 for details):

- A NXP ICODE1 is not supported by the hardware based on the RC663 chip
- B ASK CTS256B and CTS512B are not supported by the hardware based on the RC663 chip
- C SONY Felica is not supported by the hardware base on the RC632 chip

6.2. OUTPUT FORMAT (TOF REGISTER)

TOF for ID-only – abase + $\text{h}01$, size: 1 byte

Bits	Value	Meaning	
Byte swapping			
7-6	$\text{b}00$	Never swap ID bytes (the ID is transmitted “as is”)	
	$\text{b}01$	RFU	
	$\text{b}10$	Swap ID bytes for single-size (4 bytes) ISO 14443 type A UID ³ only	
	$\text{b}11$	Swap ID bytes for all kind of PICCs/VICCs	
Padding (if data is shorter than specified output length)			
5	$\text{b}0$	Padd with $\text{h}00$ on the left	
	$\text{b}1$	Padd with $\text{h}FF$ on the right	
ISO 14443 type B protocol			
4	$\text{b}0$	Uses the PUPI (4 bytes) as the ID	
	$\text{b}1$	Uses the whole ATQB (11 bytes) as the ID	
		Output format	
		Output length	
3-0	$\text{b}0000$	Decimal	10 digits (after truncation to 4 bytes if needed)
	$\text{b}0001$	Raw (hex)	Fixed, 4 bytes
	$7\text{b}0010$	Raw (hex)	Fixed, 8 bytes
	$\text{b}0011$	Raw (hex)	Fixed, 5 bytes
	$\text{b}0100$	Raw (hex)	Fixed, 10 bytes
	$\text{b}0101$	Raw (hex)	Fixed, 7 bytes
	$\text{b}0110$	Raw (hex)	Fixed, 11 bytes
	$\text{b}0111$	RFU	RFU
	$\text{b}1000$	Raw (hex)	Fixed, 16 bytes
	$\text{b}1001$	Raw (hex)	Fixed, 20 bytes
	$\text{b}1010$	Raw (hex)	Fixed, 24 bytes
	$\text{b}1011$	Raw (hex)	Fixed, 32bytes
	$\text{b}1100$	Decimal	12 digits (after truncation to 5 bytes if needed)
	$\text{b}1101$	Decimal	13 digits (after truncation to 5 bytes if needed)
$\text{b}1110$	Decimal	Variable number of digits	
$\text{b}1111$	Raw (hex)	Variable length	

6.3. PREFIX (PFX REGISTER)

Please refer to § 5.2.

³ Some old readers based on NXP documentations (not on ISO standards) uses this order by default for the Mifare short UIDs.

6.4. LOCATION (LOC REGISTER)

LOC for ID-only – base + $\text{h}03$, size: 0 or 1 byte

Uses the LOC register to specify an offset in a fixed-length output. This make it possible to select some bytes in the ID, not only the first ones.

See the examples related to non-ISO PICCs in the paragraph 6.6 for details.

6.5. MISCELLANEOUS OPTIONS (OPT REGISTER)

If this register is set, the reader adds a token to its output to tell the receiver what kind of PICC/VICC has been read.

OPT for ID-only – Address: base + $\text{h}04$, size: 1 byte

Bits	Value	Meaning
7-4		<i>RFU</i>
Position of the “card type” token in the output		
3-2	$\text{b}00$	Before the PFX constant (§ 5.2)
	$\text{b}01$	After the PFX constant, but before the actual data
	$\text{b}10$	After the actual data
	$\text{b}11$	<i>RFU</i>
Add a “card type” token to the output		
0-1	$\text{b}00$	Do not add the “card type” token
	$\text{b}01$	Add a numerical value as “card type” token, see table below
	$\text{b}10$	Add a char as “card type” token, see table below
	$\text{b}11$	<i>RFU</i>

Values for the “card type” token (if OPT is present and non-zero)

Recognized card type	Numerical value	Char
ISO 14443 type A (at least level 3)	h01	A
ISO 14443 type B (at least level 3)	h02	B
Felica	h03	F
ISO 15693	h04	V
NXP ICODE1	h08	I (<i>upper case i</i>)
Inside Secure PicoTag (including HID iClass)	h10	i
Innovision Topaz/Jewel	h11	z
Kovio RF Barcode	h18	k
ST MicroElectronics SR family	h20	s
ASK CTS256B or CTS512B	h40	a
Innovatron Radio Protocol (deprecated Calypso card)	h80	C

6.6. USING THE ID-ONLY TEMPLATE WITH NON-ISO PICCs

A few manufacturers still offer non standard cards, using either a proprietary frame format (protocol) or a proprietary command set, or both.

As those cards don’t answer to ISO 14443 / ISO 15693 standard detection commands, a specific LKL value must be chosen to process them.

7. MIFARE CLASSIC TEMPLATE

Use the **Mifare Classic Template** to read data from a NXP Mifare Classic PICC (Mifare Classic 1K, Mifare Classic 4K) or from any compliant PICCs (including Mifare Plus running in Security Level 1⁴).

The reader may either

- **Read arbitrary data**; the data will be transmitted in **hexadecimal** format. To do so, select the **RAW** mode in **TOF** register.
- Read a **number (decimal output)**. To do so, select the decimal mode in **TOF** register.
- Read a **string** (ASCII-encoded data). To do so, select either the **Short String** or the **Long String** mode in **TOF** register.

The target data is pointed to by either:

- A sector **AID in the MAD** of the card (Mifare Access Directory⁵) plus an optional offset within the sector to select the block (or a given start byte in one the of the sector's block)
- An **absolute block address** plus an optional offset to select a given start byte in the block.

The AID or the block address is specified in the **LOC** register.

7.1. LOOKUP LIST (LKL REGISTER)

LKL for Mifare Classic – Address: base + _h00, size: 1 byte

Value	Meaning	Notes
_h 61	Accept Mifare Classic PICCs (and compliant)	

7.2. SIZE AND FORMAT OF OUTPUT (TOF REGISTER)

Please refer to § 5.1 on page 19.

7.3. PREFIX (PFX REGISTER)

Please refer to § 5.2 on page 24.

⁴ The reader doesn't support the optional SL1 AES authentication of Mifare Plus PICCs.

⁵ Visit www.mifare.net for details. The reader supports both MAD1 (Mifare Classic 1K, up to 16 sectors) and MAD2 (Mifare Classic 4K, up to 40 sectors). As specified by NXP, the CRYPTO1 key to read the MAD1 or MAD2 sectors is _hA0A1A2A3A4A5.

7.4. LOCATION OF DATA (LOC REGISTER)

7.4.1. Using an AID in the MAD

LOC for Mifare Classic, AID mode – Address: base + $_{h}03$, size: 3, 4 or 5 bytes

Byte	Meaning	Notes / Valid range
Mandatory bytes		
0	AID of the sector, high-order byte	
1	AID of the sector, low-order byte	
2	Must be $_{h}00$	
Optional bytes		
3	Byte offset within the sector	$_{h}00$ to $_{h}EF$ ($_{h}00$ for block 0, $_{h}10$ for block 1...)
4	Shift bits to the left	$_{h}00$ to $_{h}07$

7.4.2. Using an absolute block address

LOC for Mifare Classic, absolute mode – Address: base + $_{h}03$, size: 3, 4 or 5 bytes

Byte	Meaning	Notes / Valid range
Mandatory bytes		
0	Must be $_{h}0000$	
1		
2	Address of <u>block</u>	
Optional bytes		
3	Byte offset within the block	$_{h}00$ to $_{h}0E$
4	Shift bits to the left	$_{h}00$ to $_{h}07$

⁶ It is technically possible to read the sector trailers, but the value is worthless since the keys are masked by zeros

7.5. AUTHENTICATION KEY (AUT REGISTER)

Reading data from a Mifare Classic involves a mandatory CRYPTO1 authentication. The CRYPTO1 algorithm uses 6-byte-long keys. Every sector is protected by two different keys, named 'key A' and 'key B'.

Use the AUT register to tell the reader

- The value of the CRYPTO1 key to access the sector holding the data
- Whether this key is the sector's 'key A' or 'key B'.

AUT for Mifare Classic – Address: base + _h05, size: 7 bytes

Bits	Value	Meaning
Byte 0		
Which key is it?		
7	_b 0	Key A
	_b 1	Key B
(unused)		
6-0	_h 00	Must be _b 0000000
Bytes 1 to 6		
Value of the CRYPTO 1 key (6 bytes)		

8. MIFARE PLUS SL3 TEMPLATE

Use the **Mifare Plus SL3 Template** to read data from a NXP Mifare Plus PICC (Mifare Plus 2K, Mifare Plus 4K) running in Security Level 3⁷.

Not all readers support this Template, please check the Implementation Matrix.

The reader may either

- **Read arbitrary data**; the data will be transmitted in **hexadecimal** format. To do so, select the **RAW** mode in **TOF** register.
- Read a **number (decimal output)**. To do so, select the decimal mode in **TOF** register.
- Read a **string** (ASCII-encoded data). To do so, select either the **Short String** or the **Long String** mode in **TOF** register.

The target data is pointed to by either:

- A sector **AID in the MAD** of the card (Mifare Access Directory⁸) plus an optional offset within the sector to select the block (or a given start byte in one the of the sector's block)
- An **absolute block address** plus an optional offset to select a given start byte in the block.

The AID or the block address is specified in the **LOC** register.

8.1. LOOKUP LIST (LKL REGISTER)

LKL for Mifare Plus – Address: base + $h00$, size: 1 byte

Value	Meaning	Notes
$h63$	Accept Mifare Plus PICCs (and compliant)	

8.2. SIZE AND FORMAT OF OUTPUT (TOF REGISTER)

Please refer to § 5.1 on page 19.

8.3. PREFIX (PFX REGISTER)

Please refer to § 5.2 on page 24.

⁷ For Security Level 1, use the Mifare Classic Template. The reader does not support the Security Level 2.

⁸ Visit www.mifare.net for details. The reader supports both MAD1 (Mifare Plus 2K, up to 16 sectors in the MAD) and MAD2 (Mifare Plus 4K, up to 40 sectors). As specified by NXP, the AES key to read the MAD1 or MAD2 sectors is $hA0A1A2A3A4A5A6A7A0A1A2A3A4A5A6A7$.

8.4. LOCATION OF DATA (LOC REGISTER)

This register has the same definition as for a Mifare Classic card. Please refer to the LOC register for the Mifare Classic Template, § 7.4 on page 30.

8.5. AUTHENTICATION KEY (AUT REGISTER)

Reading data from a Mifare Plus involves a mandatory AES authentication. The AES algorithm uses 16-byte-long keys. Every sector is protected by two different keys, named 'key A' and 'key B'.

Use the AUT register to tell the reader

- The value of the AES key to access the sector holding the data,
- Whether this key is the sector's 'key A' or 'key B',
- The secure-communication scheme to read the sector's blocks (this must match the options specified on the card, in the sector's access control bits).

AUT for Mifare Plus – Address: base + _h05, size: 17 bytes

Bits	Value	Meaning
Byte 0		
Which key is it?		
7	b0	Key A
	b1	Key B
(unused)		
6-3	b0000	Must be b0000
Read mode		
2-0	b000	Reading encrypted, MAC on command, no MAC on response
	b001	Reading encrypted, MAC on command, MAC on response ⁹
	b010	Reading in plain, MAC on command, no MAC on response
	b011	Reading in plain, MAC on command, MAC on response ¹⁰
	b100	Reading encrypted, no MAC on command, no MAC on response
	b101	Reading encrypted, no MAC on command, MAC on response
	b110	Reading in plain, no MAC on command, no MAC on response
	b111	Reading in plain, no MAC on command, MAC on response
Bytes 1 to 16		
Value of the AES key (16 bytes)		

⁹ This is the mode providing the best security level with a Mifare Plus X card

¹⁰ This is the only mode available for a Mifare Plus S card

9. MIFARE ULTRALIGHT TEMPLATE

Use the **Mifare UltraLight Template** to read data from a PICC within the NXP Mifare UltraLight family (including Mifare UltraLight C and NTAG203). Any PICC compliant with the **NFC Forum Type 2 Tag** specification could also be read using this template (but the reader is unable to decode the NFC Forum NDEF data).

The reader may either

- **Read arbitrary data**; the data will be transmitted in **hexadecimal** format. To do so, select the **RAW** mode in **TOF** register.
- Read a **number (decimal output)**. To do so, select the decimal mode in **TOF** register.
- Read a **string** (ASCII-encoded data). To do so, select either the **Short String** or the **Long String** mode in **TOF** register.

The target data is pointed to by an absolute page number (yet the data may occupy more than one page). The page number is specified in the **LOC** register.

9.1. LOOKUP LIST (LKL REGISTER)

LKL for Mifare UltraLight – Address: base + $_{h}00$, size: 1 byte

Value	Meaning	Notes
$_{h}62$	Accept Mifare UltraLight PICCs (and compliant)	

9.2. SIZE AND FORMAT OF OUTPUT (TOF REGISTER)

Please refer to § 5.1 on page 19.

9.3. PREFIX (PFX REGISTER)

Please refer to § 5.2 on page 24.

9.4. LOCATION OF DATA (LOC REGISTER)

LOC for Mifare UltraLight – Address: base + $h03$, size: 1, 2 or 2 bytes

Byte	Meaning	Notes / Valid range
Byte 0 (Mandatory)		
0	Address of 1 st page	Depends on the PICC's actual memory size
Optional bytes		
1	Offset within the 1 st page	$h00$ to $h03$
2	Shift bits to the left	$h00$ to $h07$

10. DESFIRE TEMPLATE

Use the **Desfire Template** to read data from a NXP Desfire PICC. The PICC could be either a Desfire EVO or a Desfire EV1.

The reader may either

- **Read arbitrary data**; the data will be transmitted in **hexadecimal** format. To do so, select the **RAW** mode in **TOF** register.
- Read a **number (decimal output)**. To do so, select the decimal mode in **TOF** register.
- Read a **string** (ASCII-encoded data). To do so, select either the **Short String** or the **Long String** mode in **TOF** register.

The target data is pointed to by an **Application Identifier (AID)** and a **File Identifier**. An offset within the file could be specified. The AID and the File Identifier are specified in the **LOC** register.

Access to the file could involve a **Mutual Authentication** with the card, and the data could be transmitted either in plain, MACed/CMACed or ciphered modes. The key number, key value and communication mode are specified in the **AUT** register.

The EV1 authentication and communication modes are available only in version 1.69 and onwards.

10.1. LOOKUP LIST (LKL REGISTER)

LKL for Desfire – Address: base + $_{h}00$, size: 1 byte

Value	Meaning	Notes
$_{h}71$	Accept Desfire PICCs (and compliant)	

10.2. SIZE AND FORMAT OF OUTPUT (TOF REGISTER)

Please refer to § 5.1 on page 19.

10.3. PREFIX (PFX REGISTER)

Please refer to § 5.2 on page 24.

10.4. LOCATION OF DATA (LOC REGISTER)

10.4.1. Reading a StdDataFile (or a BackupDataFile)

LOC for Desfire, ReadData command – Address: base + $_{\text{h}}03$, size: 4, 7, 8 or 9 bytes

Byte	Bits	Meaning	Notes / Valid range
Mandatory bytes			
0		Application ID, byte 0 (MSB)	The reader stores the AID in MSB-first format
1		Application ID, byte 1	
2		Application ID, byte 2 (LSB)	
3	7	$_{\text{b}}0$: StdData or BackupData File	The reader uses the Desfire ReadData command
	6-0	File ID within the application	$_{\text{h}}00$ to $_{\text{h}}7\text{F}$
Optional bytes			
4		Offset within the file, byte 0 (MSB)	The reader stores the offset in MSB-first format
5		Offset within the file, byte 1	
6		Offset within the file, byte 2 (LSB)	
7		Read length	Could be $_{\text{h}}00$ if the file is shorter than 64 bytes. If the file is longer than 64 bytes, provide the desired length. <i>The reader is limited to 240 bytes.</i>
8	7-3	Shift bytes to the left	$_{\text{d}}0$ to $_{\text{d}}32$
	2-0	Shift bits to the left	$_{\text{d}}0$ to $_{\text{d}}7$

10.4.2. Reading a RecordFile (or a CyclicRecordFile)

LOC for Desfire, ReadRecord command – Address: base + $_{h}03$, size: 4, 7, 8 or 9 bytes

Byte	Bits	Meaning	Notes / Valid range
Mandatory bytes			
0		Application ID, byte 0 (MSB)	The reader stores the AID in MSB-first format ¹¹
1		Application ID, byte 1	
2		Application ID, byte 2 (LSB)	
3	7	$_{b}1$: Record or CyclicRecord File	The reader uses the Desfire ReadRecord command
	6-0	File ID within the application	$_{h}00$ to $_{h}7F$
Optional bytes			
4		Record number, byte 0 (MSB)	The reader stores the record number in MSB-first format. Only one record is read.
5		Record number, byte 1	
6		Record number, byte 2 (LSB)	
7		Record size	Provide the exact length of the record. <i>The reader is limited to 240 bytes.</i>
8	7-3	Shift bytes to the left	$_{d}0$ to $_{d}32$
	2-0	Shift bits to the left	$_{d}0$ to $_{d}7$

10.5. AUTHENTICATION KEY (AUT REGISTER)

Reading data from a Desfire involves an optional authentication; the authentication mode could be either

- DES or 3-DES for Desfire EV0 – or Desfire EV1 operated in legacy mode,
- AES, DES, 3-DES 2K, 3-DES 3K using ISO cipher mode for Desfire EV1.

NB: The EV1 authentication modes are available only in version 1.69 and onwards.

10.5.1. No authentication

AUT for Desfire, no authentication – Address: base + $_{h}05$, size: 0 bytes

Leave this register blank to disable the authentication.

¹¹ In the Desfire command set the AID is LSB-first

10.5.2. Authentication EV0 DES/3DES2K

In this mode, a DES or 3-DES authentication is performed on the Desfire application, before reading the data from the file, and a session key is generated.

The DES or 3-DES algorithm uses 16-byte-long keys.

Use the AUT register to tell the reader

- The number of the key within the application,
- The value of the DES or 3-DES key,
- The secure-communication scheme to read the file's data (this must match the allowed modes specified on the card, in the file's access control bits).

a. All versions

AUT for Desfire, authentication EV0 – Address: base + 05, size: 17 bytes

Bits	Value	Meaning
Byte 0		
Communication mode		
7-6	b00	Plain
	b01	MACed with using the session key
	b10	RFU
	b11	Encrypted using the session key
(unused)		
5-4	b00	Must be b00
Key number within the Desfire application		
3-0	b0000 to b1110	(Value b1111 is not allowed by the Desfire card)
Bytes 1 to 16		
Value of the DES or 3-DES key (16 bytes) For a DES key, both halves of the key are equal.		

b. Starting with version 1.69

AUT for Desfire, authentication EV0 – Address: base + _n05, size: 18 bytes

Bits	Value	Meaning
Byte 0		
Communication mode		
7-6	_b 00	Plain
	_b 01	MACed with using the session key
	_b 10	<i>RFU</i>
	_b 11	Encrypted using the session key
(unused)		
5-4	_b 00	Must be _b 00
Key number within the Desfire application		
3-0	_b 0000 to _b 1110	(Value _b 1111 is not allowed by the Desfire card)
Byte 1		
7-0	_n 01	_n 01 selects "AUTHENTICATE" command of Desfire EV0
Bytes 2 to 17		
Value of the DES or 3-DES key (16 bytes) For a DES key, both halves of the key are equal.		

10.5.3. Authentication EV1 DES/3DES2K

In this mode, the Desfire EV1's "AUTHENTICATE ISO" command is used to perform the authentication (where in § 10.5.2 the EV0's "AUTHENTICATE" command is used).

The DES or 3-DES algorithm uses 16-byte-long keys (3DES2K).

Use the AUT register to tell the reader

- The number of the key within the application,
- The value of the DES or 3-DES key,
- The secure-communication scheme to read the file's data (this must match the allowed modes specified on the card, in the file's access control bits).

AUT for Desfire, authentication EV1 DES/3DES2K– Address: base + _h05, size: 18 bytes

Bits	Value	Meaning
Byte 0		
Communication mode		
7-6	_b 00	Plain
	_b 01	CMACed with using the session key
	_b 10	<i>RFU</i>
	_b 11	Encrypted using the session key
(unused)		
5-4	_b 00	Must be _b 00
Key number within the Desfire application		
3-0	_b 0000 to _b 1110	(Value _b 1111 is not allowed by the Desfire card)
Byte 1		
7-0	_h 02	_h 02 selects "AUTHENTICATE ISO" command of Desfire EV1
Bytes 2 to 17		
Value of the DES or 3-DES key (16 bytes) For a DES key, both halves of the key are equal.		

10.5.4. Authentication EV1 3DES3K

In this mode, the Desfire EV1's "AUTHENTICATE ISO" command is used to perform the authentication.

The 3-DES algorithm uses 24-byte-long keys (3DES3K).

Use the AUT register to tell the reader

- The number of the key within the application,
- The value of the 3-DES key,
- The secure-communication scheme to read the file's data (this must match the allowed modes specified on the card, in the file's access control bits).

AUT for Desfire, authentication EV1 3DES3K– Address: base + $h05$, size: 26 bytes

Bits	Value	Meaning
Byte 0		
Communication mode		
7-6	$b00$	Plain
	$b01$	CMACed with using the session key
	$b10$	RFU
	$b11$	Encrypted using the session key
(unused)		
5-4	$b00$	Must be $b00$
Key number within the Desfire application		
3-0	$b0000$ to $b1110$	(Value $b1111$ is not allowed by the Desfire card)
Byte 1		
7-0	$h02$	$h02$ selects "AUTHENTICATE ISO" command of Desfire EV1
Bytes 2 to 25		
Value of the 3-DES key (24 bytes)		

10.5.5. Authentication EV1 AES

In this mode, the Desfire EV1's "AUTHENTICATE AES" command is used to perform the authentication.

The AES algorithm uses 16-byte-long keys.

Use the AUT register to tell the reader

- The number of the key within the application,
- The value of the AES key,
- The secure-communication scheme to read the file's data (this must match the allowed modes specified on the card, in the file's access control bits).

AUT for Desfire, authentication EV1 AES– Address: base + $h05$, size: 18 bytes

Bits	Value	Meaning
Byte 0		
Communication mode		
7-6	$b00$	Plain
	$b01$	CMACed with using the session key
	$b10$	<i>RFU</i>
	$b11$	Encrypted using the session key
(unused)		
5-4	$b00$	Must be $b00$
Key number within the Desfire application		
3-0	$b0000$ to $b1110$	(Value $b1111$ is not allowed by the Desfire card)
Byte 1		
7-0	$h03$	$h03$ selects "AUTHENTICATE AES" command of Desfire EV1
Bytes 2 to 17		
Value of the AES key (16 bytes)		

11. ISO 7816-4 TEMPLATE

ISO 7816-4 is the standard for smart card commands. According to this standard, the card is structured as a (lightweight) file-system, providing Directory Files and Elementary Files. Functions are defined to select the files and read the data from them. Every function call is called an “APDU”. The card's response is always terminated by a 2-B status word which denotes the success (value $_{h}9xxx$, typically $_{h}9000$) or the failure (value $_{h}6xxx$).

Using the **ISO 7816-4 Template**, the reader is able to send 1, 2 or 3 APDUs and to find the data in the last card's response. The communication with the card is performed using either the **ISO 14443-4 protocol** (“T=CL”) or the Innovatron Radio Protocol (deprecated Calypso cards).

11.1. LOOKUP LIST (LKL REGISTER)

Address: base + $_{h}00$, size: 1 byte

Value	Meaning	Notes
$_{h}11$	Read data from ISO 14443-4 type A PICCs	
$_{h}12$	Read data from ISO 14443-4 type B PICCs	
$_{h}13$	Read data from both ISO 14443-4 type A and type B PICCs	
$_{h}72$	Read data from Calypso cards using the Innovatron Radio Protocol	

11.2. SIZE AND FORMAT OF OUTPUT (TOF REGISTER)

Please refer to § 5.1 on page 19.

11.3. PREFIX (PFX REGISTER)

Please refer to § 5.2 on page 24.

11.4. LOCATION OF DATA (LOC REGISTER)

LOC for ISO 7816-4 – Address: base + $h03$, size: 0, 1 or 2 bytes

Byte	Meaning	Notes / Valid range
Optional bytes		
0	Offset within the response <u>to the last APDU</u>	
1	Shift bits to the left	$h00$ to $h07$

11.5. ISO 7816-4 APDU 1 (AU1 REGISTER)

Typically, this is a SELECT instruction (SELECT APPLICATION or SELECT DIRECTORY FILE).

AU1 for ISO 7816-4 – Address: base + $h05$, size: 4 to 32 bytes

Notes:

- *This 1st APDU can't be left empty,*
- *The reader's receive buffer is limited to 128 bytes. Specify a L_E below $h80$ to make sure the card's response will not overflow this buffer,*
- *The reader does check the status word. The card must return a "success" SW ($h9xxx$). Otherwise, the reader stops the transactions.*

11.6. ISO 7816-4 APDU 2 (AU2 REGISTER)

Typically, this is another SELECT instruction (SELECT ELEMENTARY FILE), unless the file could be implicitly selected by a SFI (Short File Identifier) within a READ instruction.

AU2 for ISO 7816-4 – Address: base + $_{h}06$, size: 4 to 32 bytes

Notes:

- *This 2nd APDU could be left empty (in this case the data is taken from the response to the 1st APDU),*
- *The reader's receive buffer is limited to 128 bytes. Specify a L_E below $_{h}80$ to make sure the card's response will not overflow this buffer,*
- *The reader does check the status word. The card must return a "success" SW ($_{h}9xxx$). Otherwise, the reader stops the transactions.*

11.7. ISO 7816-4 APDU 3 (AU3 REGISTER)

Typically, this is a READ instruction (READ BINARY or READ RECORD).

AU3 for ISO 7816-4 – Address: base + $_{h}07$, size: 4 to 32 bytes

Notes:

- *This 3rd APDU could be left empty (in this case the data is taken from the response to the 2nd APDU),*
- *The reader's receive buffer is limited to 128 bytes. Specify a L_E below $_{h}80$ to make sure the card's response will not overflow this buffer,*
- *The reader does check the status word. The card must return a "success" SW ($_{h}9xxx$). Otherwise, the reader stops the transactions.*

12. ISO 15693 MEMORY TEMPLATE

Use the **ISO 15693 Memory Template** to read data from a VICC compliant with ISO 15693-3.

The reader may either

- **Read arbitrary data**; the data will be transmitted in **hexadecimal** format. To do so, select the **RAW** mode in **TOF** register.
- Read a **number (decimal output)**. To do so, select the decimal mode in **TOF** register.
- Read a **string** (ASCII-encoded data). To do so, select either the **Short String** or the **Long String** mode in **TOF** register.

The target data is pointed to by an absolute block number (yet the data may occupy more than one block). According to the standard, the size of every block depends of the card (its a manufacturer's choice). Typical block sizes are 1, 2, 4 or 8 bytes. The reader accepts block sizes up to 64 bytes. The block number is specified in the **LOC** register.

Not all readers support this Template, please check the Implementation Matrix.

12.1. LOOKUP LIST (LKL REGISTER)

LKL for ISO 15693 Memory – Address: base + $h00$, size: 1 byte

Value	Meaning	Notes
$h54$	Accept ISO 15693-3 VICCs	

12.2. SIZE AND FORMAT OF OUTPUT (TOF REGISTER)

Please refer to § 5.1 on page 19.

12.3. PREFIX (PFX REGISTER)

Please refer to § 5.2 on page 24.

12.4. LOCATION OF DATA (LOC REGISTER)

LOC for ISO 15693 Memory – Address: base + $_{h}03$, size: 1, 2 or 3 bytes

Byte	Meaning	Notes / Valid range
Byte 0 (Mandatory)		
0	Address of 1 st block	Depends on the PICC's actual memory size
Optional bytes		
1	Offset within the 1 st page	$_{h}00$ to $_{h}03$
2	Shift bits to the left	$_{h}00$ to $_{h}07$

13. EM4134 TEMPLATE

Use the **EM4134 Memory Template** to read data from the EM MicroElectronic Marin EM4134 chip.

The reader may either

- **Read arbitrary data**; the data will be transmitted in **hexadecimal** format. To do so, select the **RAW** mode in **TOF** register.
- Read a **number (decimal output)**. To do so, select the decimal mode in **TOF** register.
- Read a **string** (ASCII-encoded data). To do so, select either the **Short String** or the **Long String** mode in **TOF** register.

The target data is pointed to by an absolute word number; a word is 4-byte long (yet the data may occupy more than one word). The word number is specified in the **LOC** register.

Not all readers support this Template, please check the Implementation Matrix.

13.1. LOOKUP LIST (LKL REGISTER)

LKL for EM4134 Memory – Address: base + $_{h}00$, size: 1 byte

Value	Meaning	Notes
$_{h}56$	Accept EM4134 chip	

13.2. SIZE AND FORMAT OF OUTPUT (TOF REGISTER)

Please refer to § 5.1 on page 19.

13.3. PREFIX (PFX REGISTER)

Please refer to § 5.2 on page 24.

13.4. LOCATION OF DATA (LOC REGISTER)

LOC for EM4134 Memory – Address: base + $h03$, size: 1, 2 or 3 bytes

Byte	Meaning	Notes / Valid range
Byte 0 (Mandatory)		
0	Address of 1 st word	1 to 15
Optional bytes		
1	Offset within the 1 st word	$h00$ to $h03$
2	Shift bits to the left	$h00$ to $h07$

14. NDEF DATA

Use the **NDEF Data Template** to read the content of either a card formatted according to the **NFC Forum Tag specification**, or to receive a message through **NFC beam (SNEP over LLCP)**.

Not all readers support these Templates, please check the Implementation Matrix.

14.1. LOOKUP LIST (LKL REGISTER)

LKL for NDEF Data – Address: base + $h00$, size: 1 byte

Value	Meaning	Notes
$h40$	Receive NDEF by SNEP (peer-to-peer)	
$h41$	Read NDEF from NFC Forum type 1 Tags	
$h42$	Read NDEF from NFC Forum type 2 Tags	
$h43$	Read NDEF from NFC Forum type 3 Tags	
$h44$	Read NDEF from NFC Forum type 4 Tags (A & B)	
$h4A$	Read NDEF from NFC Forum type 4A Tags	
$h4B$	Read NDEF from NFC Forum type 4B Tags	
$h4E$	Read NDEF from any NFC Forum Tag	
$h4F$	Read NDEF from any NFC Forum Tag and Receive NDEF by SNEP (peer-to-peer)	

14.2. SIZE AND FORMAT OF OUTPUT (TOF REGISTER)

When the NDEF Data template is selected, only “Long String” mode is supported.

The NDEF Record's Payload must contain only valid ASCII bytes. The reader transmits the letters until either a zero value is read ($_{h}00$, i.e. '\0' i.e. the “end of string” char) or the specified length is reached.

TOF for NDEF Data, long string mode – Address: base + $_{h}01$, size: 2 bytes

Bits	Value	Meaning
Byte 0		
Byte swapping		
7	$_{b}0$	Send the data “as is”
	$_{b}1$	Reverse the data before sending (last char first)
Mode (raw/decimal or string)		
6	$_{b}1$	String mode → must be $_{b}1$
Fixed length		
5	$_{b}0$	Variable length (no padding)
	$_{b}1$	Padd with ' ' chars (SPACE) on the right until the max output length is reached
Short/long string mode		
4	$_{b}1$	Long string mode → must be $_{b}1$
(unused)		
3-0	$_{b}0000$	Must be $_{b}0000$
Byte 1		
(unused)		
7	$_{b}0$	RFU (must be $_{b}0$)
Output length (max)		
6-0	$_{h}01$ to $_{h}FF$	From 1 to 255 chars 0 means “no limit”

14.3. PREFIX (PFX REGISTER)

Please refer to § 5.2 on page 24.

14.4. TYPE NAME AND FORMAT (TNF REGISTER)

TNF for NDEF Data – Address: base + _h05, size: 1 byte

Value	Meaning	Notes
_h 01	Read a NFC Forum well-known record (NFC RTD)	TYP register contains the actual type
_h 02	Read a Media record (RFC 2046)	
_h 03	Read an absolute URI (RFC 3986)	
_h 04	Read a NFC Forum external record (NFC RTD)	
_h 53	Read a SpringCard data entry (see § 14.6.3)	TYP register must remain empty
_h 54	Read a Text (see § 14.6.2)	
_h 55	Read a URI (see § 14.6.1)	
_h FF	Read either a SpringCard data entry, a URI or a Text	

14.5. TYPE (TYP REGISTER)

TYP for NDEF Data – Address: base + _h06, size: 0 to 16 bytes

Use the TYP register to select the NDEF Record you want to read. Please refer to the specifications of NFC Forum NDEF Data and Records, and to the specifications of the MIME Media, to provide a valid TNF,TYP combination.

For instance, two valid TNF,TYP combinations are:

- *TNF=_h01, TYP=_h55 (“U”): this is a URI as specified in NFC Forum’s “RTD URI” specification. Note that in this case the raw URI is returned, which is not the case when TNF=_h55 (see § 14.6.1)*
- *TNF=_h02, TYP=_h74 65 78 74 2F 70 6C 61 69 6E (“text/plain”): this is a Media record, the MIME type is plaintext*

14.6. USING TNF AND TYP REGISTERS TO SELECT THE DATA FROM AN NDEF RECORD

14.6.1. Reading a URI

If TNF=_h55 (and TYP is empty), the Reader tries to read either

- An absolute URI record (RFC 3986),
- A URI record according to the NFC Forum “RTD URI” specification,
- A URI record within a smartposter record according to the NFC Forum “RTD SmartPoster” specification,

In the first case, the record's content is returned “as is”. In the two later cases, the record's content is expanded into a real URI string according to the NFC Forum “RTD URI” specification.

The Reader supports the ASCII character set only. Valid characters are _h20 to _h7F only.

Note that the Reader stops when the first valid URI is found in the NDEF message. It is not possible to read a later record.

14.6.2. Reading a Text

If TNF=_h54 (and TYP is empty), the Reader tries to read either

- A text media (RFC2046 with MIME type set to “text/plain”),
- A Text record according to the NFC Forum “RTD Text” specification,
- A Text record within a smartposter record according to the NFC Forum “RTD SmartPoster” specification,

In the first case, the record's content is returned “as is”. In the two later cases, the “lang” part of the Text content is removed.

The Reader supports the ASCII character set only. Valid characters are _h20 to _h7F only.

Note that the Reader stops when the first valid Text is found in the NDEF message. It is not possible to read a later record. It is not possible to select a particular “lang” value.

14.6.3. Reading a SpringCard data entry

If TNF=_h53 (and TYP is empty), the Reader tries to read a Media record having type “text/x-springcard-scan-data”.

14.6.4. Reading a custom data entry

Set TNF=_h02 and use a custom MIME Media type (in TYP) to access your custom data.

In this mode, the Reader supports only reading ASCII string. Therefore, your custom data shall be in the “text/” family.

15. SPRINGBLUE HCE

SpringBlue is a turnkey identification solution developed by **SpringCard** to send a credential to a reader from virtually any recent mobile phone:

- Android smartphones featuring HCE (host card emulation) are handled as contactless cards thanks to this **SpringBlue HCE** template,
- For all other smartphones (running either Android, iOS or Windows), the credential may also be transmitted over BLE (Bluetooth v4 Low Energy). To date, only the **FunkyGate-DW NFC+BLE** supports SpringBlue over BLE. This feature uses a different set of configuration registers, and is documented in chapter 18.

Merchants and service providers are identified by a unique *InstallationId* (on 4 bytes). Every user “belonging” to this merchant or service provider is identified by a unique *UserId* (on 8 bytes).

The **SpringBlue HCE** template retrieves the *UserId* for a given *InstallationId*.

In HCE (and in BLE mode), a security scheme (based on a secret key and HMAC-SHA1 computation) allows to verify that the credential comes from a trusted source.

Not all readers support this Template, please check the Implementation Matrix.

15.1. LOOKUP LIST (LKL REGISTER)

LKL for SpringBlue HCE – Address: base + $h00$, size: 1 byte

Value	Meaning	Notes
$hB0$	SpringBlue HCE	

15.2. SIZE AND FORMAT OF OUTPUT (TOF REGISTER)

Please refer to § 5.1 on page 19.

15.3. PREFIX (PFX REGISTER)

Please refer to § 5.2 on page 24.

15.4. LOCATION OF DATA (LOC REGISTER)

LOC for SpringBlue HCE – Address: base + $_{h}03$, size: 4 bytes

Byte	Meaning	Notes / Valid range
Byte 0-4		
0	InstallationId	MSB
1		
2		LSB
3		

15.5. AUTHENTICATION KEY (AUT REGISTER)

AUT for SpringBlue HCE – Address: base + $_{h}05$, size: 16 bytes

Bits	Value	Meaning
Bytes 0 to 15		
Value of the authentication key (16 bytes)		

16. ORANGE NFC APIs (RETAIL)

Orange NFC APIs for Retail is a framework for NFC services, offered by Mobile Network Operator Orange. It targets loyalty, couponing and alike applications.

The services rely on a general-purpose cardlet, loaded into a NFC-enabled smartphone's SIM card.

Merchants and service providers are identified by a unique *RetailerId* (on 4 bytes). Every user “belonging” to this merchant or service provider is identified by either

- A unique *HolderId* (on 20 bytes), assigned by Orange to this very user,
- A *CustomerId*, defined freely by the merchant or service provider in association with a customer account,
- A *LoyaltyId*, defined freely by the merchant or service provider in association with a loyalty card or coupons.

*Note that the optional security scheme is not handled by the reader.
Not all readers support this Template, please check the Implementation Matrix.*

16.1. LOOKUP LIST (LKL REGISTER)

LKL for Orange NFC APIs (Retail) – Address: base + _h00, size: 1 byte

Value	Meaning	Notes
_h CO	Orange NFC APIs (retail)	

16.2. SIZE AND FORMAT OF OUTPUT (TOF REGISTER)

Please refer to § 5.1 on page 19.

16.3. PREFIX (PFX REGISTER)

Please refer to § 5.2 on page 24.

16.4. FIELD SELECT AND OFFSETS (LOC REGISTER)

LOC for Orange NFC APIs (Retail) – Address: base + $h03$, size: 5 to 7 bytes

Byte	Meaning	Notes / Valid range
Bytes 0 to 5 (Mandatory)		
0	RetailerId	MSB
1		
2		LSB
3		
4	Options & field select	See below
Optional bytes		
5	Offset within the field	$d0$ to field's size
6	Shift bits to the left	$d0$ to $d7$

Detail of 'Options & field select' (byte 4 in LOC)

Bits	Value	Meaning
Options		
4-7		Value that the reader will “push” to the mobile phone after processing. Allowed values are $h0$, $h1$ and hF
3	$b0$	Don't suppress trailing zeros
	$b1$	Suppress trailing zeros (valid for field select = $h2$ or $h4$ only)
Field select		
2-0	$h01$	Read the MeHolderId field
	$h02$	Read the CustomerId field
	$h04$	Read the LoyaltyCardId field
		Other values are RFU and <u>shall not be used</u>

17. ORANGE NFC API (OFFICE)

Orange NFC APIs for Office is a framework for NFC services, offered by Mobile Network Operator Orange. It targets user identification applications (access control, private e-purse, car park & card sharing, and alike).

The services are provided by the smartphone either in host-based card emulation mode (HCE) or through a SIM card.

Organisations are identified by an *ApplicationId* (variable length) and their organisation units by a *Zoneld* (on 6 bytes). A diversified AES key protects the data. Every user “belonging” to the organisation is identified by a *Badge* number on 10 bytes.

Not all readers support this Template, please check the Implementation Matrix.

17.1. LOOKUP LIST (LKL REGISTER)

LKL for Orange NFC APIs (retail) – Address: base + _h00, size: 1 byte

Value	Meaning	Notes
_h C0	Orange NFC APIs (retail)	

17.2. SIZE AND FORMAT OF OUTPUT (TOF REGISTER)

Please refer to § 5.1 on page 19.

17.3. PREFIX (PFX REGISTER)

Please refer to § 5.2 on page 24.

17.4. FIELD SELECT AND OFFSETS (LOC REGISTER)

LOC for Orange NFC APIs (Office) – Address: base + $_{h}03$, size: 1 to 3 bytes

Byte	Meaning	Notes / Valid range
Byte 0 (Mandatory)		
0	$_{h}00$: no authentication $_{h}01$: use AES authentication	See below
Optional bytes		
1	Offset within the field	$_{d}0$ to $_{d}10$
2	Shift bits to the left	$_{d}0$ to $_{d}7$

17.5. APPLICATION ID

Application ID for Orange NFC APIs (Office) – Address: base + $_{h}05$, size: 1 to 32 bytes

Bits	Value	Meaning
Bytes 0 to ...		
Application ID. This value is transmitted by the reader to the mobile phone within a SELECT APPLICATION command APDU (00 A4 04 00 ... <Application ID>)		

17.6. ZONE ID

Zone ID for Orange NFC APIs (Office) – Address: base + $_{h}06$, size: 6 bytes

Bits	Value	Meaning
Bytes 0 to 5		
Zone ID. This value is transmitted by the reader to the mobile phone within a PUT DATA command APDU (80 B0 80 00 06 <Zone ID>)		

17.7. MASTER KEY

Master Key for Orange NFC APIs (Office) – Address: base + $_{h}07$, size: 16 bytes

Bits	Value	Meaning
Bytes 0 to 15		
Master Key. This value is diversified with the data returned by the SELECT APPLICATION and used to decrypt the <i>Badge</i> number is the authentication is enabled in LOC register.		

18. SPRINGBLUE BLE

SpringBlue is a turnkey identification solution developed by **SpringCard** to send a credential to a reader from virtually any recent mobile phone:

- Android smartphones featuring HCE (host card emulation) are handled as contactless cards thanks to the **SpringBlue HCE** template which is documented in chapter 15,
- For all other smartphones (running either Android, iOS or Windows), the credential could be pushed over BLE (Bluetooth v4 Low Energy). To date, only the **FunkyGate-DW NFC+BLE** supports the **SpringBlue BLE** feature documented here.

Merchants and service providers are identified by a unique *InstallationId* (on 4 bytes). Every user “belonging” to this merchant or service provider is identified by a unique *UserId* (on 8 bytes).

The **SpringBlue BLE** template retrieves the *UserId* for a given *InstallationId*.

A security scheme (based on a secret key and HMAC-SHA1 computation) allows to verify that the credential comes from a trusted source.

| *Only the FunkyGate-DW NFC+BLE supports this feature.*

18.1. ENABLING THE SPRINGBLUE BLE APPLICATION

18.2. SIZE AND FORMAT OF OUTPUT (TOF REGISTER)

Please refer to § 5.1 on page 19.

18.3. PREFIX (PFX REGISTER)

Please refer to § 5.2 on page 24.

18.4. LOCATION OF DATA (LOC REGISTER)

LOC for SpringBlue BLE – Address: $_{h}03$, size: 4 bytes

Byte	Meaning	Notes / Valid range
Byte 0-4		
0	InstallationId	MSB
1		
2		LSB
3		

18.5. AUTHENTICATION KEY (AUT REGISTER)

AUT for SpringBlue BLE – Address: $_{h}05$, size: 16 bytes

Bits	Value	Meaning
Bytes 0 to 15		
Value of the authentication key (16 bytes)		

DISCLAIMER

This document is provided for informational purposes only and shall not be construed as a commercial offer, a license, an advisory, fiduciary or professional relationship between PRO ACTIVE and you. No information provided in this document shall be considered a substitute for your independent investigation.

The information provided in document may be related to products or services that are not available in your country.

This document is provided "as is" and without warranty of any kind to the extent allowed by the applicable law. While PRO ACTIVE will use reasonable efforts to provide reliable information, we don't warrant that this document is free of inaccuracies, errors and/or omissions, or that its content is appropriate for your particular use or up to date. PRO ACTIVE reserves the right to change the information at any time without notice.

PRO ACTIVE doesn't warrant any results derived from the use of the products described in this document. PRO ACTIVE will not be liable for any indirect, consequential or incidental damages, including but not limited to lost profits or revenues, business interruption, loss of data arising out of or in connection with the use, inability to use or reliance on any product (either hardware or software) described in this document.

These products are not designed for use in life support appliances, devices, or systems where malfunction of these product may result in personal injury. PRO ACTIVE customers using or selling these products for use in such applications do so on their own risk and agree to fully indemnify PRO ACTIVE for any damages resulting from such improper use or sale.

COPYRIGHT NOTICE

All information in this document is either public information or is the intellectual property of PRO ACTIVE and/or its suppliers or partners.

You are free to view and print this document for your own use only. Those rights granted to you constitute a license and not a transfer of title : you may not remove this copyright notice nor the proprietary notices contained in this documents, and you are not allowed to publish or reproduce this document, either on the web or by any mean, without written permission of PRO ACTIVE.

Copyright © PRO ACTIVE SAS 2016, all rights reserved.

EDITOR'S INFORMATION

PRO ACTIVE SAS company with a capital of 227 000 €

RCS EVRY B 429 665 482

Parc Gutenberg, 13 voie La Cardon

91120 Palaiseau – FRANCE

CONTACT INFORMATION

For more information and to locate our sales office or distributor in your country or area, please visit

www.springcard.com