PMA13205-CA
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	CA
Status	draft	Classification	Public
Keywords	RDR, Prox'N'Roll RFID Scanner, FunkyGate, Prox'N'Drive RFID Scanner		
Abstract			

File name	[PMA13205-CA] Smart Readers and RFID Scanners Template System.odt		
Date saved	06/11/17	Date printed	06/11/17

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 ThinFilm 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
BE	10/03/17	MBA LCO				Added the “tag type” prefix in OPT register of all templates Kovio RF Barcode → ThinFilm
CA	06/01/17	JDA				Added the NXP AN10922 key diversification algorithm, for Desfire EV1 cards, in AES mode (FW v1.79) Added support for NFC Forum type 5 tags (FW 1.70)

Contents

1. INTRODUCTION....6
1.1. Abstract....6
1.2. Supported products....6
1.3. Audience....7
1.4. Support and updates....7
2. PRINCIPLES....8
2.1. The Template system....8
2.2. Configuration registers....8
2.3. Editing the configuration....9
2.4. Registers used by the Templates....9
3. REFERENCE TABLES....10
3.1. List of Templates by LKL values....10
3.2. List of values for LKL by Template....12
3.3. Summary of configuration registers....14
4. IMPLEMENTATION MATRIX....15
4.1. Readers based on the CSB6 core....16
4.2. Readers based on the H663, K663, E663, S663 cores....17
4.3. Readers based on the K632 core....18
5. FEATURES SHARED AMONG TEMPLATES....19
5.1. Size and format of output (TOF register)....19
5.1.1. RAW mode....20
5.1.2. Decimal mode....21
5.1.3. Short String mode (up to 16 bytes)....22
5.1.4. Long String mode....23
5.2. Prefix (PFX register)....24
6. ID-ONLY TEMPLATE....25
6.1. Lookup List (LKL register)....25
6.2. Output format (TOF register)....26
6.3. Prefix (PFX register)....26
6.4. Location (LOC register)....27
6.5. Miscellaneous options (OPT register)....27
6.6. Using the ID-only Template with non-ISO PICCs....28
7. MIFARE CLASSIC TEMPLATE....29
7.1. Lookup List (LKL register)....29
7.2. Size and format of output (TOF register)....29
7.3. Prefix (PFX register)....29
7.4. Location of data (LOC register)....30
7.4.1. Using an AID in the MAD....30
7.4.2. Using an absolute block address....30
7.5. Miscellaneous options (OPT register)....31
7.6. Authentication key (AUT register)....31
8. MIFARE PLUS SL3 TEMPLATE....32
8.1. Lookup List (LKL register)....32
8.2. Size and format of output (TOF register)....32
8.3. Prefix (PFX register)....32
8.4. Location of data (LOC register)....33
8.5. Miscellaneous options (OPT register)....33
8.6. Authentication key (AUT register)....34
9. MIFARE ULTRALIGHT TEMPLATE....35
9.1. Lookup List (LKL register)....35
9.2. Size and format of output (TOF register)....35
9.3. Prefix (PFX register)....35
9.4. Location of data (LOC register)....36
9.5. Miscellaneous options (OPT register)....36
10. DESFIRE TEMPLATE....37
10.1. Lookup List (LKL register)....37
10.2. Size and format of output (TOF register)....37
10.3. Prefix (PFX register)....37
10.4. Location of data (LOC register)....38
10.4.1. Reading a StdDataFile (or a BackupDataFile)....38
10.4.2. Reading a RecordFile (or a CyclicRecordFile)....39
10.5. Miscellaneous options (OPT register)....40
10.6. Authentication key (AUT register)....40
10.6.1. No authentication....40
10.6.2. Authentication EV0 DES/3DES2K....41
10.6.3. Authentication EV1 DES/3DES2K....43
10.6.4. Authentication EV1 3DES3K....44
10.6.5. Authentication EV1 AES....45
10.6.6. AES key diversification....46
11. ISO 7816-4 TEMPLATE....47
11.1. Lookup List (LKL register)....47
11.2. Size and format of output (TOF register)....48
11.3. Prefix (PFX register)....48
11.4. Location of data (LOC register)....48
11.5. Miscellaneous options (OPT register)....48
11.6. ISO 7816-4 APDU 1 (AU1 register)....49
11.7. ISO 7816-4 APDU 2 (AU2 register)....49
11.8. ISO 7816-4 APDU 3 (AU3 register)....49
12. ISO 15693 MEMORY TEMPLATE....50
12.1. Lookup List (LKL register)....50
12.2. Size and format of output (TOF register)....50
12.3. Prefix (PFX register)....50
12.4. Location of data (LOC register)....51
12.5. Miscellaneous options (OPT register)....51
13. EM4134 TEMPLATE....52
13.1. Lookup List (LKL register)....52
13.2. Size and format of output (TOF register)....52
13.3. Prefix (PFX register)....52
13.4. Location of data (LOC register)....53
13.5. Miscellaneous options (OPT register)....53

14. NDEF DATA....54
14.1. Lookup List (LKL register)....54
14.2. Size and format of output (TOF register)....55
14.3. Prefix (PFX register)....55
14.4. Miscellaneous options (OPT register)....56
14.5. Type Name and Format (TNF register)....56
14.6. Type (TYP register)....56
14.7. Using TNF and TYP registers to select the data from an NDEF Record....57
14.7.1. Reading a URI....57
14.7.2. Reading a Text....57
14.7.3. Reading a SpringCard data entry....58
14.7.4. Reading a custom data entry....58
15. SPRINGBLUE HCE....59
15.1. Lookup List (LKL register)....59
15.2. Size and format of output (TOF register)....59
15.3. Prefix (PFX register)....59
15.4. Miscellaneous options (OPT register)....60
15.5. Location of data (LOC register)....60
15.6. Authentication key (AUT register)....60
16. ORANGE NFC APIS (RETAIL)....61
16.1. Lookup List (LKL register)....61
16.2. Size and format of output (TOF register)....61
16.3. Prefix (PFX register)....61
16.4. Miscellaneous options (OPT register)....62
16.5. Field select and offsets (LOC register)....62
17. ORANGE NFC API (OFFICE)....64
17.1. Lookup List (LKL register)....64
17.2. Size and format of output (TOF register)....64
17.3. Prefix (PFX register)....64
17.4. Field select and offsets (LOC register)....65
17.5. Miscellaneous options (OPT register)....65
17.6. Application ID....65
17.7. Zone ID....66
17.8. Master Key....66
18. SPRINGBLUE BLE....67
18.1. Enabling the SpringBlue BLE application....67
18.2. Size and format of output (TOF register)....67
18.3. Prefix (PFX register)....67
18.4. Location of data (LOC register)....68
18.5. Miscellaneous options (OPT register)....68
18.6. Authentication key (AUT register)....68

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-volatile[1] 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	ThinFilm	**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)		
h45	Read NDEF from NFC Forum type 5 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		
h4F	Read NDEF from any NFC Forum Tag and Receive NDEF by SNEP (peer-to-peer)		
h54	ISO 15693	**ISO 15693 Memory**	12

$_h$56	EM4134	**EM4134 Memory**	13
$_h$61	NXP Mifare Classic 1K & 4K	**Mifare Classic**	7
$_h$62	NXP Mifare UltraLight	**Mifare UltraLight**	9
$_h$63	NXP Mifare Plus 2K & 4K, S & X, in SL3	**Mifare Plus SL3**	8
$_h$71	NXP Desfire	**Desfire**	10
$_h$72	Innovatron Radio Protocol (deprecated Calypso cards)	**ID Only**	6
		ISO 7816-4	11
$_h$B0	SpringBlue HCE (card emulation)	**SpringBlue HCE**	15
$_h$C0	NFC phone with Orange Retail applet in the SIM	**Orange NFC Retail**	16
$_h$C1	NFC phone with Orange Office application (HCE)	**Orange NFC Office**	17
$_h$FF	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	ThinFilm
		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
		h45	Read NDEF from NFC Forum type 5 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
		h4F	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	$_h$71	NXP Desfire
ISO 7816-4	11	$_h$11	ISO 14443 type A (up to layer 4 “T=CL”)
		$_h$12	ISO 14443 type B (up to layer 4 “T=CL”)
		$_h$13	ISO 14443 (A & B) (up to layer 4 “T=CL”)
		$_h$72	Innovatron Radio Protocol (deprecated Calypso cards)
SpringBlue HCE	15	$_h$B0	Support SpringBlue HCE
Orange NFC Retail	16	$_h$C0	Support Orange NFC Retail
Orange NFC Office	17	$_h$C1	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			Div. const. (DIV)	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**
ThinFilm 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 EV0 authentication	10.6	**Yes**
Desfire EV1 authentication		**No**
NXP AN10922 AES key diversification for Desfire EV1		**No**
Long string in Templates: Mifare Classic, Mifare UltraLight, Mifare Plus, Desfire EV0 and Desfire EV1	5.1.4	Firmware ≥ 1.40
Not-byte-aligned data (shift bits left feature) in Templates: - Mifare Classic and Mifare Plus - Desfire - ISO 7816-4	 7.4 10.4 11.4	 Firmware ≥ 1.45 Firmware ≥ 1.45 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, types 1, 2 & 4	14	Firmware ≥ 1.56
NDEF data from NFC Forum Tags, type 3		**No**
NDEF data from NFC Forum Tags, type 5		**No**
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**
ThinFilm 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 EV0 authentication	10.6	**Yes**
Desfire EV1 authentication	10.6	Firmware ≥ 1.69
NXP AN10922 AES key diversification for Desfire EV1	10.6	Firmware ≥ 1.79
Long string in Templates: Mifare Classic, Mifare UltraLight, Mifare Plus, Desfire EV0 and Desfire EV1	5.1.4	**Yes**
Not-byte-aligned data (shift bits left feature) in Templates: - Mifare Classic and Mifare Plus - Desfire - ISO 7816-4 templates	 7.4 10.4 11.4	 **Yes** **Yes** **Yes**
Data from ISO 15693 tags	12	Firmware ≥ 1.57
Data from EM4134 tags	13	Firmware ≥ 1.60
NDEF data from NFC Forum Tags, types 1, 2 & 4	14	Firmware ≥ 1.57
NDEF data from NFC Forum Tags, type 3	14	Firmware ≥ 1.58
NDEF data from NFC Forum Tags, type 5	14	Firmware ≥ 1.80
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**
ThinFilm 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 EV0 authentication	10.6	**Yes**
Desfire EV1 authentication		**No**
NXP AN10922 AES key diversification for Desfire EV1		**No**
Long string in Templates: Mifare Classic, Mifare UltraLight, Mifare Plus, Desfire EV0 and Desfire EV1	5.1.4	Firmware ≥ 1.40
Not-byte-aligned data (shift bits left feature) in Templates: - Mifare Classic and Mifare Plus - Desfire - ISO 7816-4 templates		 **No** **No** **No**
Data from ISO 15693 tags		**No**
Data from EM4134 tags		**No**
NDEF data from NFC Forum Tags		**No**
NDEF data from NFC Forum Tags, type 3		**No**
NDEF data from NFC Forum Tags, type 5		**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 ($_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 format[2].

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

Common TOF register, RAW 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$0	Raw mode → must be $_b$0
Padding (applicable if read length < specified output length)		
5	$_b$0	Padd using '0' char on the left
	$_b$1	Padd using 'F' char on the right
Strip leading zeroes		
4	$_b$0	Do not remove leading '0' chars
	$_b$1	Remove leading '0' chars
Size of output		
3-0	$_b$0001	4 bytes
	$_b$0002	8 bytes
	$_b$0011	5 bytes
	$_b$0100	10 bytes
	$_b$0101	7 bytes
	$_b$0110	11 bytes
	$_b$1000	16 bytes
	$_b$1001	20 bytes
	$_b$1010	24 bytes
	$_b$1011	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 *4 bytes are transmitted in decimal*
	$_b$1100	12 digits *5 bytes are transmitted in decimal*
	$_b$1101	13 digits - " -
	$_b$1110	Unlimited *16 bytes are transmitted in decimal*

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 + $_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 $_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 + $_h$00, size: 1 byte

Value	Meaning	Notes
$_h$01	Accept **ISO 14443 type A** PICCs	
$_h$02	Accept **ISO 14443 type B** PICCs	
$_h$03	Accept **ISO 14443 type A** and **type B** PICCs	
$_h$04	Accept **ISO 15693** VICCs	
$_h$07	Accept **ISO 14443 type A** and **type B** PICCs and **ISO 15693**	
$_h$08	Accept **NXP ICODE1** VICCs	A
$_h$0C	Accept **ISO 15693** and **NXP ICODE1** VICCs	A
$_h$0F	Accept all of the above	
$_h$20	Accept **ThinFilm** family	
$_h$21	Accept **Innovision Topaz/Jewel** family	
$_h$22	Accept **ST MicroElectronics SR** family	
$_h$23	Accept **ASK CTS256B and CTS512B**	B
$_h$24	Accept **Inside Secure PicoTag** (including HID iClass)	
$_h$28	Accept **Sony Felica** Family	C
$_h$72	Accept **Innovatron Radio Protocol** (deprecated Calypso cards)	
$_h$FF	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 + $_h$01, size: 1 byte

Bits	Value	Meaning	
Byte swapping			
7-6	$_b$00	Never swap ID bytes (the ID is transmitted "as is")	
	$_b$01	RFU	
	$_b$10	Swap ID bytes for single-size (4 bytes) ISO 14443 type A UIDs only[3]	
	$_b$11	Swap ID bytes for all kind of PICCs/VICCs	
Padding (if data is shorter than specified output length)			
5	$_b$0	Padd with $_h$00 on the left	
	$_b$1	Padd with $_h$FF on the right	
ISO 14443 type B protocol			
4	$_b$0	Uses the PUPI (4 bytes) as the ID	
	$_b$1	Uses the whole ATQB (11 bytes) as the ID	
		Output format	**Output length**
3-0	$_b$0000	Decimal	10 digits (after truncation to 4 bytes if needed)
	$_b$0001	Raw (hex)	Fixed, 4 bytes
	$_b$0010	Raw (hex)	Fixed, 8 bytes
	$_b$0011	Raw (hex)	Fixed, 5 bytes
	$_b$0100	Raw (hex)	Fixed, 10 bytes
	$_b$0101	Raw (hex)	Fixed, 7 bytes
	$_b$0110	Raw (hex)	Fixed, 11 bytes
	$_b$0111	*RFU*	*RFU*
	$_b$1000	Raw (hex)	Fixed, 16 bytes
	$_b$1001	Raw (hex)	Fixed, 20 bytes
	$_b$1010	Raw (hex)	Fixed, 24 bytes
	$_b$1011	Raw (hex)	Fixed, 32bytes
	$_b$1100	Decimal	12 digits (after truncation to 5 bytes if needed)
	$_b$1101	Decimal	13 digits (after truncation to 5 bytes if needed)
	$_b$1110	Decimal	Variable number of digits
	$_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 + $_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 + $_h$04, size: 1 byte

Bits	Value	Meaning
7-4		*RFU*
Position of the "card type" token in the output		
3-2	$_b$00	Before the PFX constant (§ 5.2)
	$_b$01	After the PFX constant, but before the actual data
	$_b$10	After the actual data
	$_b$11	*RFU*
Add a "card type" token to the output		
0-1	$_b$00	Do not add the "card type" token
	$_b$01	Add a ***numerical value*** as "card type" token, see table below
	$_b$10	Add a ***char*** as "card type" token, see table below
	$_b$11	*RFU*

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)	$_h$01	A
ISO 14443 type B (at least level 3)	$_h$02	B
Felica	$_h$03	F
ISO 15693	$_h$04	V
NXP ICODE1	$_h$08	I *(upper case i)*
Inside Secure PicoTag (including HID iClass)	$_h$10	i
Innovision Topaz/Jewel	$_h$11	z
ThinFilm	$_h$18	k
ST MicroElectronics SR family	$_h$20	s
ASK CTS256B or CTS512B	$_h$40	a
Innovatron Radio Protocol (deprecated Calypso card)	$_h$80	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 1[4]).

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 Directory[5]) 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 + $_h$00, 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 $_h$A0A1A2A3A4A5.

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 block	$_h$00 to $_h$FF [6]
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. Miscellaneous options (OPT register)

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

OPT for Mifare Classic – Address: base + $_h$04, size: 1 byte

Bits	Value	Meaning
7-4		*RFU*
Position of the "card type" token in the output		
3-2	$_b$00	Before the PFX constant (§ 5.2)
	$_b$01	After the PFX constant, but before the actual data
	$_b$10	After the actual data
	$_b$11	*RFU*
Add a "card type" token to the output		
0-1	$_b$00	Do not add the "card type" token
	$_b$01	Add a $_h$**61** as "card type" token
	$_b$10	Add a **'M'** as "card type" token
	$_b$11	Add $_h$**00**

7.6. 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 + $_h$05, 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 3[7].

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 Directory[8]) 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 + $_h$00, size: 1 byte

Value	Meaning	Notes
$_h$63	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 $_h$A0A1A2A3A4A5A6A7A0A1A2A3A4A5A6A7.

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. Miscellaneous options (OPT register)

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

OPT for Mifare Plus – Address: base + $_h$04, size: 1 byte

Bits	Value	Meaning
7-4		*RFU*
Position of the "card type" token in the output		
3-2	$_b$00	Before the PFX constant (§ 5.2)
	$_b$01	After the PFX constant, but before the actual data
	$_b$10	After the actual data
	$_b$11	*RFU*
Add a "card type" token to the output		
0-1	$_b$00	Do not add the "card type" token
	$_b$01	Add a $_h$**63** as "card type" token
	$_b$10	Add a **'M'** as "card type" token
	$_b$11	Add $_h$**00**

8.6. 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 + $_h$05, size: 17 bytes

Bits	Value	Meaning
Byte 0		
Which key is it?		
7	$_b$0	Key A
	$_b$1	Key B
(unused)		
6-3	$_b$0000	Must be $_b$0000
Read mode		
2-0	$_b$000	Reading encrypted, MAC on command, no MAC on response
	$_b$001	Reading encrypted, MAC on command, MAC on response [9]
	$_b$010	Reading in plain, MAC on command, no MAC on response
	$_b$011	Reading in plain, MAC on command, MAC on response[10]
	$_b$100	Reading encrypted, no MAC on command, no MAC on response
	$_b$101	Reading encrypted, no MAC on command, MAC on response
	$_b$110	Reading in plain, no MAC on command, no MAC on response
	$_b$111	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 + $_h$03, size: 1, 2 or 2 bytes

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

9.5. Miscellaneous options (OPT register)

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

OPT for Mifare UltraLight – Address: base + $_h$04, size: 1 byte

Bits	Value	Meaning
7-4		*RFU*
Position of the "card type" token in the output		
3-2	$_b$00	Before the PFX constant (§ 5.2)
	$_b$01	After the PFX constant, but before the actual data
	$_b$10	After the actual data
	$_b$11	*RFU*
Add a "card type" token to the output		
0-1	$_b$00	Do not add the "card type" token
	$_b$01	Add a $_h$**62** as "card type" token
	$_b$10	Add a **'U'** as "card type" token
	$_b$11	Add $_h$**00**

10. Desfire Template

Use the **Desfire Template** to read data from a NXP Desfire PICC. The PICC could be either a Desfire EV0 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 + $_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	$_B$0 : StdData or BackupData File	The reader uses the Desfire ReadData command
	6-0	File ID within the application	$_h$00 to $_h$7F
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 $_h$00 if the file is shorter than 64 bytes. If the file is longer than 64 bytes, provide the desired length. *The reader is limited to 240 bytes.*
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

10.4.2. 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. *The reader is limited to 240 bytes.*
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

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

10.5. Miscellaneous options (OPT register)

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

OPT for Desfire – Address: base + $_h$04, size: 1 byte

Bits	Value	Meaning
7-5		*RFU*
4	$_b$0	Do not send SELECT APPLICATION (DESFIRE AID)
	$_b$1	Send SELECT APPLICATION (DESFIRE AID) before processing
Position of the "card type" token in the output		
3-2	$_b$00	Before the PFX constant (§ 5.2)
	$_b$01	After the PFX constant, but before the actual data
	$_b$10	After the actual data
	$_b$11	*RFU*
Add a "card type" token to the output		
0-1	$_b$00	Do not add the "card type" token
	$_b$01	Add a $_h$**71** as "card type" token
	$_b$10	Add a **'D'** as "card type" token
	$_b$11	Add $_h$**00**

10.6. 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.

10.6.1. No authentication

AUT for Desfire, no authentication – Address: base + $_h$05, size: 0 bytes

Leave this register blank to disable the authentication.

10.6.2. 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 + $_h$05, size: 17 bytes

Bits	Value	Meaning
Byte 0		
Communication mode		
7-6	$_b$00 $_b$01 $_b$10 $_b$11	Plain MACed with using the session key *RFU* 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)
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 + $_h$05, size: 18 bytes

Bits	Value	Meaning
Byte 0		
Communication mode		
7-6	$_b$00 $_b$01 $_b$10 $_b$11	Plain MACed with using the session key *RFU* 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$01	$_h$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.		

10.6.3. Authentication EV1 DES/3DES2K

In this mode, the Desfire EV1's “AUTHENTICATE ISO” command is used to perform the authentication (where in § 10.6.2 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 + $_h$05, size: 18 bytes

Bits	Value	Meaning
Byte 0		
Communication mode		
7-6	$_b$00 $_b$01 $_b$10 $_b$11	Plain CMACed with using the session key *RFU* 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.		

10.6.4. 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 + $_h$05, size: 26 bytes

Bits	Value	Meaning
Byte 0		
Communication mode		
7-6	$_b$00 $_b$01 $_b$10 $_b$11	Plain CMACed with using the session key *RFU* 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 25		
Value of the 3-DES key (24 bytes)		

10.6.5. 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 + $_h$05, 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	*RFU*
	$_b$11	Encrypted using the session key
Key diversification[12]		
5-4	$_b$00	No key diversification
	$_b$01	NXP AN10922 key diversification, see § 10.6.6 for details
	$_b$10	*RFU*
	$_b$11	*RFU*
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$03	$_h$03 selects "AUTHENTICATE AES" command of Desfire EV1
Bytes 2 to 17		
Value of the AES key (16 bytes)		

[12] This feature has been introduced in FW v1.79. With earlier versions, bits 5-4 of this byte shall be set to $_b$00.

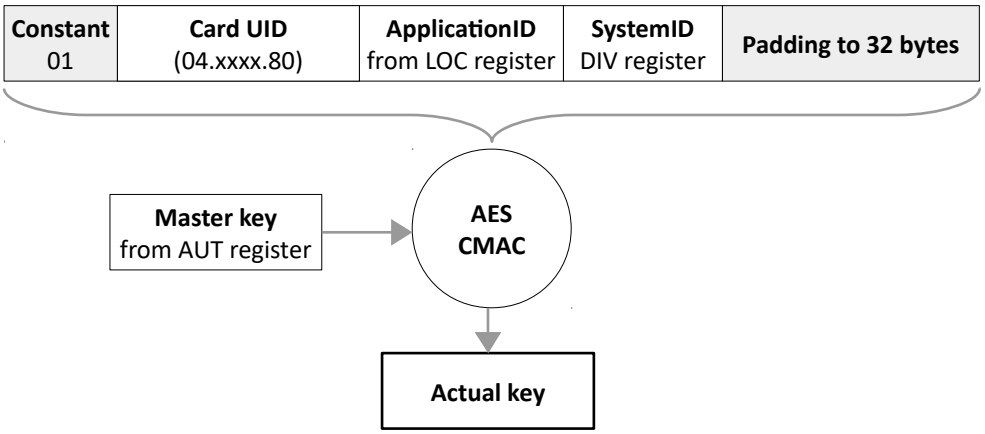
10.6.6. AES key diversification

Key diversification is a process of deriving the authentication key from a master (base) key using some unique input. The unique input is typically constructed other the card's UID, so every card is getting a different key value, while only the readers have to know the master key.

NXP application note AN10922 "Symmetric key diversification" recommends the use of an algorithm based on a AES128-CMAC (cryptographic message authentication code)[13]. This algorithm is implemented in SpringCard Smart Readers & RFID Scanners, starting with firmware version 1.79, and is selected by setting bits 5-4 of AUT register's byte 0 are to $_b$01.

The AN10922 uses a constant value named 'System Identifier', which has to be provided in the DIV register (# 6) or could be left empty.

The key diversification algorithm is implemented as follow:



DIV for Desfire, System ID constant for AN10922 diversification – Address: base + $_h$06, size: var.

Bits	Value	Meaning
Bytes 0 to Length-1		
Value of the System ID constant (variable length, 0 to 16 bytes)		

[13] Issuing systems are likely to use a NXP SAM AV2 to store the master key and write the diversified key into every card.

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 + $_h$03, size: 0, 1 or 2 bytes

Byte	Meaning	Notes / Valid range
Optional bytes		
0	Offset within the response to the last APDU	
1	Shift bits to the left	$_h$00 to $_h$07

11.5. Miscellaneous options (OPT register)

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

OPT for ISO 7816-4 – Address: base + $_h$04, size: 1 byte

Bits	Value	Meaning
7-4		*RFU*
Position of the "card type" token in the output		
3-2	$_b$00 $_b$01 $_b$10 $_b$11	Before the PFX constant (§ 5.2) After the PFX constant, but before the actual data After the actual data *RFU*
Add a "card type" token to the output		
0-1	$_b$00 $_b$01 $_b$10 $_b$11	Do not add the "card type" token Add a $_h$**1F** as "card type" token Add a **'T'** as "card type" token Add $_h$**00**

11.6. 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 + $_h$05, 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 $_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 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.*

11.8. 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 + $_h$00, size: 1 byte

Value	Meaning	Notes
$_h$54	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 1st block	Depends on the PICC's actual memory size
Optional bytes		
1	Offset within the 1st page	$_h$00 to $_h$03
2	Shift bits to the left	$_h$00 to $_h$07

12.5. Miscellaneous options (OPT register)

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

OPT for ISO 15693 Memory – Address: base + $_h$04, size: 1 byte

Bits	Value	Meaning
7-4		*RFU*
Position of the "card type" token in the output		
3-2	$_b$00	Before the PFX constant (§ 5.2)
	$_b$01	After the PFX constant, but before the actual data
	$_b$10	After the actual data
	$_b$11	*RFU*
Add a "card type" token to the output		
0-1	$_b$00	Do not add the "card type" token
	$_b$01	Add a $_h$**54** as "card type" token
	$_b$10	Add a **'V'** as "card type" token
	$_b$11	Add $_h$**00**

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 + $_h$03, size: 1, 2 or 3 bytes

Byte	Meaning	Notes / Valid range
Byte 0 (Mandatory)		
0	Address of 1st word	1 to 15
Optional bytes		
1	Offset within the 1st word	$_h$00 to $_h$03
2	Shift bits to the left	$_h$00 to $_h$07

13.5. Miscellaneous options (OPT register)

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

OPT for EM4134 Memory – Address: base + $_h$04, size: 1 byte

Bits	Value	Meaning
7-4		*RFU*
Position of the "card type" token in the output		
3-2	$_b$00	Before the PFX constant (§ 5.2)
	$_b$01	After the PFX constant, but before the actual data
	$_b$10	After the actual data
	$_b$11	*RFU*
Add a "card type" token to the output		
0-1	$_b$00	Do not add the "card type" token
	$_b$01	Add a $_h$**56** as "card type" token
	$_b$10	Add a **'V'** as "card type" token
	$_b$11	Add $_h$**00**

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 + $_h$00, size: 1 byte

Value	Meaning	Notes
$_h$40	Receive NDEF by SNEP (**peer-to-peer**)	
$_h$41	Read NDEF from NFC Forum **type 1** Tags	
$_h$42	Read NDEF from NFC Forum **type 2** Tags	
$_h$43	Read NDEF from NFC Forum **type 3** Tags	
$_h$44	Read NDEF from NFC Forum **type 4** Tags (A & B)	
$_h$45	Read NDEF from NFC Forum **type 5** Tags (aka “Type V”)	
$_h$4A	Read NDEF from NFC Forum **type 4A** Tags	
$_h$4B	Read NDEF from NFC Forum **type 4B** Tags	
$_h$4E	Read NDEF from any NFC Forum Tag	
$_h$4F	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 $_b$1	Send the data “as is”) 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 $_b$1	Variable length (no padding) 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. Miscellaneous options (OPT register)

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

OPT for NDEF Data – Address: base + $_h$04, size: 1 byte

Bits	Value	Meaning
7-4		*RFU*
Position of the "card type" token in the output		
3-2	$_b$00	Before the PFX constant (§ 5.2)
	$_b$01	After the PFX constant, but before the actual data
	$_b$10	After the actual data
	$_b$11	*RFU*
Add a "card type" token to the output		
0-1	$_b$00	Do not add the "card type" token
	$_b$01	Add a $_h$**4F** as "card type" token
	$_b$10	Add a **'N'** as "card type" token
	$_b$11	Add $_h$**00**

14.5. Type Name and Format (TNF register)

TNF for NDEF Data – Address: base + $_h$05, 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 § 14.7.3)	TYP register must remain empty
$_h$54	Read a Text (see § 14.7.2)	
$_h$55	Read a URI (see § 14.7.1)	
$_h$FF	Read either a SpringCard data entry, a URI or a Text	

14.6. Type (TYP register)

TYP for NDEF Data – Address: base + $_h$06, 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=$_h$01, TYP=$_h$55 ("U"): this is a URI as specified in NFC Forum's "RTD URI" specification. Note that in this case the raw URI is returned, which is not the case when TNF=$_h$55 (see § 14.7.1)
- TNF=$_h$02, TYP=$_h$74 65 78 74 2F 70 6C 61 69 6E ("text/plain"): this is a Media record, the MIME type is plaintext

14.7. Using TNF and TYP registers to select the data from an NDEF Record

14.7.1. Reading a URI

If TNF=$_h$55 (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 $_h$20 to $_h$7F 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.

14.7.2. Reading a Text

If TNF=$_h$54 (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 $_h$20 to $_h$7F 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.

14.7.3. Reading a SpringCard data entry

If TNF=$_h$53 (and TYP is empty), the Reader tries to read a Media record having type “text/x-springcard-scan-data”.

14.7.4. Reading a custom data entry

Set TNF=$_h$02 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 + $_h$00, size: 1 byte

Value	Meaning	Notes
$_h$B0	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. Miscellaneous options (OPT register)

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

OPT for SpringBlue HCE – Address: base + $_h$04, size: 1 byte

Bits	Value	Meaning
7-4		*RFU*
Position of the "card type" token in the output		
3-2	$_b$00	Before the PFX constant (§ 5.2)
	$_b$01	After the PFX constant, but before the actual data
	$_b$10	After the actual data
	$_b$11	*RFU*
Add a "card type" token to the output		
0-1	$_b$00	Do not add the "card type" token
	$_b$01	Add a $_h$**B0** as "card type" token
	$_b$10	Add a **'S'** as "card type" token
	$_b$11	Add $_h$**00**

15.5. 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		
3		LSB

15.6. 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 + $_h$00, size: 1 byte

Value	Meaning	Notes
$_h$C0	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. Miscellaneous options (OPT register)

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

OPT for Orange NFC APIs (Retail) – Address: base + $_h$04, size: 1 byte

Bits	Value	Meaning
7-4		*RFU*
Position of the "card type" token in the output		
3-2	$_b$00	Before the PFX constant (§ 5.2)
	$_b$01	After the PFX constant, but before the actual data
	$_b$10	After the actual data
	$_b$11	*RFU*
Add a "card type" token to the output		
0-1	$_b$00	Do not add the "card type" token
	$_b$01	Add a $_h$**C0** as "card type" token
	$_b$10	Add a **'O'** as "card type" token
	$_b$11	Add $_h$**00**

16.5. Field select and offsets (LOC register)

LOC for Orange NFC APIs (Retail) – Address: base + $_h$03, size: 5 to 7 bytes

Byte	Meaning	Notes / Valid range
Bytes 0 to 5 (Mandatory)		
0	***RetailerId***	MSB
1		
2		
3		LSB
4	**Options & field select**	See below
Optional bytes		
5	Offset within the field	$_d$0 to field's size
6	Shift bits to the left	$_d$0 to $_d$7

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 $_h$0, $_h$1 and $_h$F
3	$_b$0 $_b$1	Don't suppress trailing zeros Suppress trailing zeros (valid for field select = $_h$2 or $_h$4 only
		Field select
2-0	$_h$01 $_h$02 $_h$04	Read the ***MeHolderId*** field Read the ***CustomerId*** field Read the ***LoyaltyCardId*** field Other values are RFU and shall not be used

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 alikes).

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 *ZoneId* (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 (Office) – Address: base + $_h$00, size: 1 byte

Value	Meaning	Notes
$_h$C1	Orange NFC APIs (Office)	

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. Miscellaneous options (OPT register)

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

OPT for Orange NFC APIs (Office) – Address: base + $_h$04, size: 1 byte

Bits	Value	Meaning
7-4		*RFU*
Position of the "card type" token in the output		
3-2	$_b$00 $_b$01 $_b$10 $_b$11	Before the PFX constant (§ 5.2) After the PFX constant, but before the actual data After the actual data *RFU*
Add a "card type" token to the output		
0-1	$_b$00 $_b$01 $_b$10 $_b$11	Do not add the "card type" token Add a $_h$**C1** as "card type" token Add a **'O'** as "card type" token Add $_h$**00**

17.6. 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>)		

17.7. 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>		

17.8. 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 *Badge* 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

Please refer to the product's documentation.

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

18.5. Miscellaneous options (OPT register)

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

OPT for SpringBlue BLE – Address: base + $_h$04, size: 1 byte

Bits	Value	Meaning
7-4		*RFU*
Position of the "card type" token in the output		
3-2	$_b$00	Before the PFX constant (§ 5.2)
	$_b$01	After the PFX constant, but before the actual data
	$_b$10	After the actual data
	$_b$11	*RFU*
Add a "card type" token to the output		
0-1	$_b$00	Do not add the "card type" token
	$_b$01	Add a $_h$**B0** as "card type" token
	$_b$10	Add a **'S'** as "card type" token
	$_b$11	Add $_h$**00**

18.6. 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 SPRINGCARD 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 SPRINGCARD 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. SPRINGCARD reserves the right to change the information at any time without notice.

SPRINGCARD doesn't warrant any results derived from the use of the products described in this document. SPRINGCARD 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. SPRINGCARD customers using or selling these products for use in such applications do so on their own risk and agree to fully indemnify SPRINGCARD for any damages resulting from such improper use or sale.

Copyright notice

Editor's information

SPRINGCARD SAS company with a capital of 227 000 €

RCS EVRY B 429 665 482

Parc Gutenberg, 2 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