PMA13257-BB
FINAL - PUBLIC

SpringCard FunkyGate-IP NFC

Integration and Configuration Guide

www.springcard.com

Document Identification

Category	Owner's Manual		
Family/Customer	FunkyGate NFC Series		
Reference	PMA13257	Version	BB
Status	Final	Classification	Public
Keywords			
Abstract			

File name	V:\Dossiers\SpringCard\A-Notices\RFID scanners et lecteurs\FunkyGate-IP\[PMA13257-BB] FunkyGate-IP NFC Integration and Configuration Guide.odt		
Date saved	29/07/15	Date printed	03/06/14

Revision History

Ver.	Date	Author	Valid. by		Approv. by	Details
			Tech.	Qual.		
AA	29/09/13	JDA				Created from PMA959P
AB	24/04/14	JDA				Rewritten the authentication scheme
AC	28/05/14	JDA				Added the HTTP server and the REST API
BA	30/07/14	JDA			JDA	Network part moved to PMA14166
BB	29/07/15	JIZ				Changed reset command: §6.2.1 and §6.4.3

Contents

1. Introduction

1.1. Abstract

SpringCard FunkyGate-IP NFC is a RFID (13.56MHz) and NFC wall-mount Reader, for access control applications. **SpringCard FunkyGate-IP NFC** features an exclusive TCP/IP over Ethernet interface.

The attractive styling and the efficiency of the Ethernet interface make it the preferred choice for corporate environments. Advanced support of the widest range of technologies and exclusive security features allow high-end access control schemes to be deployed seamlessly.

Thanks to a **versatile Template System** (shared with all other **SpringCard** Readers and RFID/NFC Scanners), **SpringCard FunkyGate-IP NFC** is able to read either a serial number or virtually any data coming from standard ISO/IEC 14443 proximity cards, ISO/IEC 15693 vicinity labels or tags. It is also able to fetch NDEF data from RFID chips formatted according to one the NFC Forum Tag specifications, and to receive NDEF data from a NFC Forum “peer-to-peer” (SNEP server on top of LLCP).

The **SpringCard FunkyGate-IP+POE NFC** version provides the “powered by the network” (POE) feature.

This document provides all necessary information to configure both the **FunkyGate-IP NFC** and **FunkyGateIP-POE + NFC** Readers, and to develop a software that will handle data coming from the Reader, and to drive or re-configure the Reader when needed.

1.2. Supported product

Order code	Product
SC14002	**FunkyGate-IP NFC:** new generation wall-mount RFID/NFC/contactless card Reader, with Ethernet interface (10 or 100 Mbit/s)
SC14003	**FunkyGate-IP+POE NFC**: new generation wall-mount RFID/NFC/contactless card Reader, with Ethernet interface (10 or 100 Mbit/s), powered by the network

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, TCP/IP networks, 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

1.5. Related documents

1.5.1. Products' specifications

You'll find the feature-list and the technical characteristics of every product in the corresponding leaflet.

Document ref.	Content
PFL13276	**FunkyGate NFC** family leaflet

1.5.2. Common documentations

a. Network integration and configuration

SpringCard FunkyGate-IP NFC Reader shares the same network communication protocol (on top of TCP/IP) and the same way of configuring the network as **SpringCard HandyDrummer-IP** I/O Module. A document share among both products covers this part.

Document ref.	Content
PMA14166	**FunkyGate-IP NFC, HandyDrummer-IP** Network Integration and Configuration

b. Card reading & processing

All SpringCard Readers and RFID Scanners products share the same “card processing” system through 1 to 4 processing templates. How the Reader process the card is therefore detailed in a document shared among all products in the family.

Document ref.	Content
PMA13205	**Readers / RFID Scanners Template System**

2. Define the Reader's IP Address

The Reader comes out of factory without an IP address. This means that you must assign it an IP address before being able to access it either through Telnet link (chapter 3) or using the TCP client/server protocol depicted in chapter 5.

Using **SpringCard Network Device Discovery Utility (NDDU)** is the preferred method to assign an IP address to the Reader.

> *This Reader does not support the Dynamic Host Configuration Protocol (DHCP). Only fixed IPv4 addresses are supported.*

2.1. Assign an IP address using NDDU software

SpringCard Network Device Discovery Utility (NDDU) is a Windows-based software that discovers and configures SpringCard Device connected on same the Local Area Network (LAN) as the computer it is running on.

> *Please use a wired network connection, and make sure the Reader(s) you want to configure are on the same LAN as your computer. NDDU makes use of broadcast UDP frames to discover and configure the Readers; therefore, it won't work behind a router or gateway.*

2.1.1. Download and install the NDDU software

Make sure your Windows account has administrative privileges.

Download the installer from URL

www.springcard.com/download/find/file/sn13210

Install the software.

> *This software relies on the .NET framework version 4. Please download and install this framework from Microsoft's in case it hasn't already been deployed onto your computer.*

2.1.2. Run the NDDU software

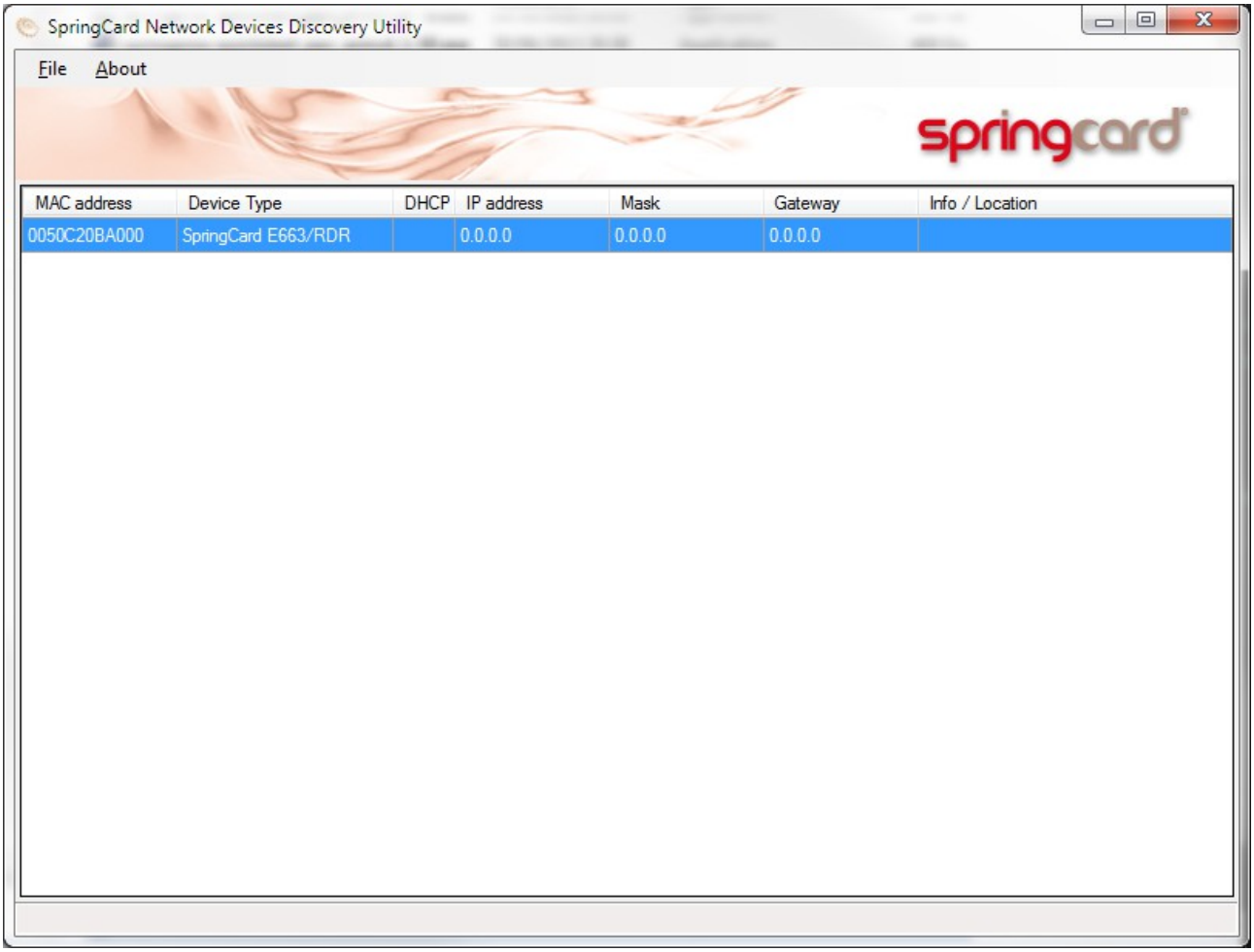
Make sure your Windows account has administrative privileges.

Launch the software: Start Menu → SpringCard → Network Discovery → Network Device Discovery Utility.

On first startup, you should be prompted by Windows Firewall whether you want to allow NDDU to access the network. Please confirm.

2.1.3. Discovered devices

After a few seconds, NDDU displays the list of devices it has found on the LAN.



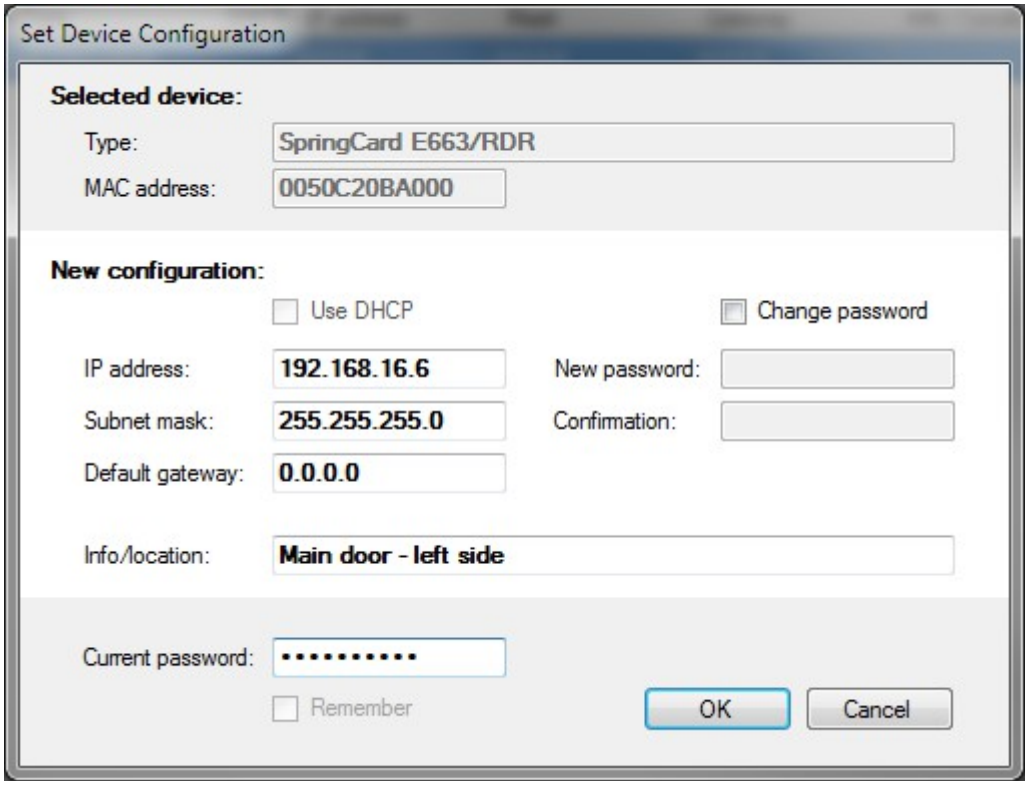
The software's main screen shows 7 columns:

- The MAC address (Ethernet address and also serial number) of every SpringCard Device found on the LAN,
- The device type (code name **SpringCard E663/RDR** for **SpringCard FunkyGate-IP NFC** and related products),
- Whether DHCP is enabled or not (DHCP is not supported by **SpringCard FunkyGate-IP NFC**),
- The device's current IP address, local network mask, and default gateway. Until the device has been properly configured, those entries show has “0.0.0.0”,

- A user-defined string named “Info / location”, which will be used as an hint to identify the device in your own system.

2.1.4. Configure a Reader

Double-click one of the devices in the list. The configuration form appears:



The form shows the device's current configuration. Enter the new configuration. IP address and subnet mask are mandatory data and couldn't be left empty. The default gateway is optional; if the devices won't need to use a gateway, leave this field to “0.0.0.0”.

In the “info/location” field, enter a short string (less than 32 characters) as a reminder of the device's location or role.

Terminate by entering the device's current password to confirm your allowed to change this device's configuration.

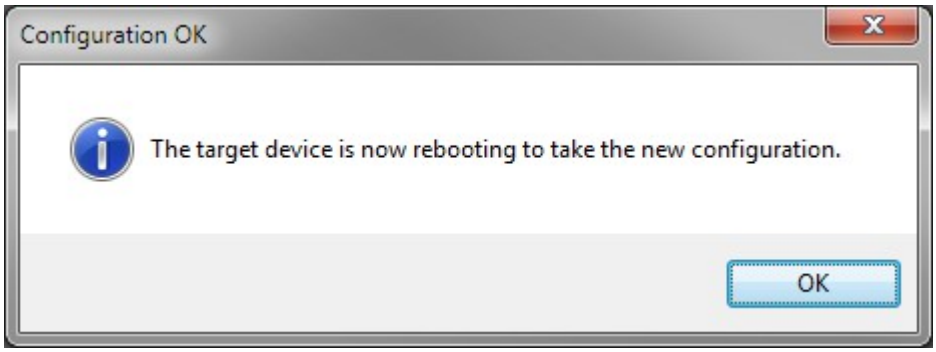
> *The default password for all devices is **springcard**.*

Check the box “change password” and enter a new password twice if you want to change it.

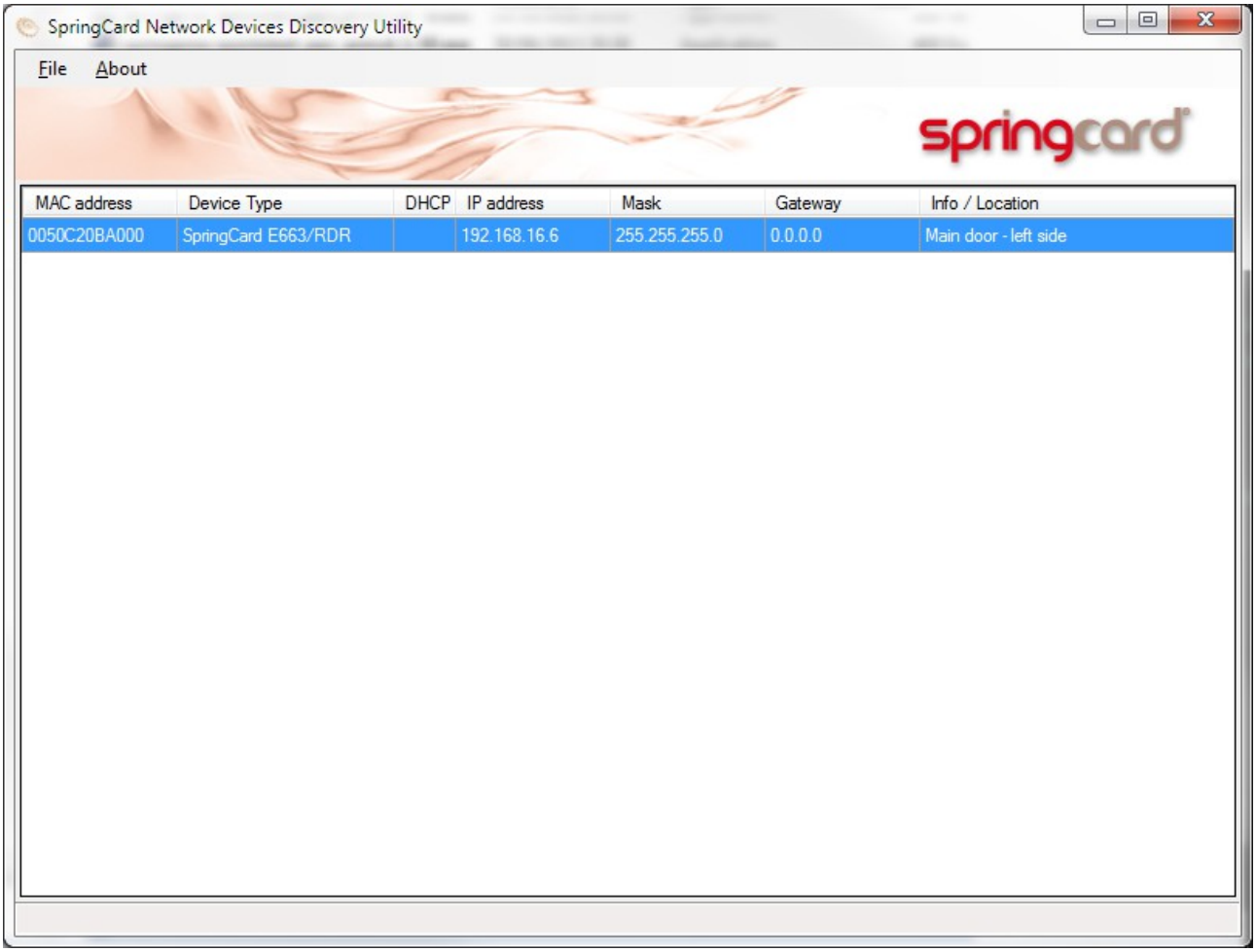
When ready, click “OK”.

2.1.5. Verify the new configuration

If everything is OK, including the current password, the NDDU software is able to configure the device. The following message confirms that the new configuration has been accepted:



After a few seconds, the list of devices is refreshed and shows the new configuration:



2.2. Assign an IP address using a Master Card

The Master Cards are NXP Desfire cards formatted and programmed by **SpringCard Configuration Tool (ScMultiConf.exe, ref # SN14007)** for Windows.

Please refer to this software's documentation for details.

3. Telnet access to the Reader

3.1. Reader's Console

The Reader features a “human” command processor (shell or console). This feature accessible through the Telnet protocol. It is primarily made for testing and demonstration purpose. Only the few commands depicted in this chapter could safely be used for configuration and diagnostic.

Note that the SEC Configuration Register (h6E, § 8.4) may be used to disable the Console.

3.1.1. Open a Telnet session to the reader

On most operating systems you could find a Telnet client in the default system tools. Open a console and enter

```
telnet xxx.xxx.xxx.xxx
```

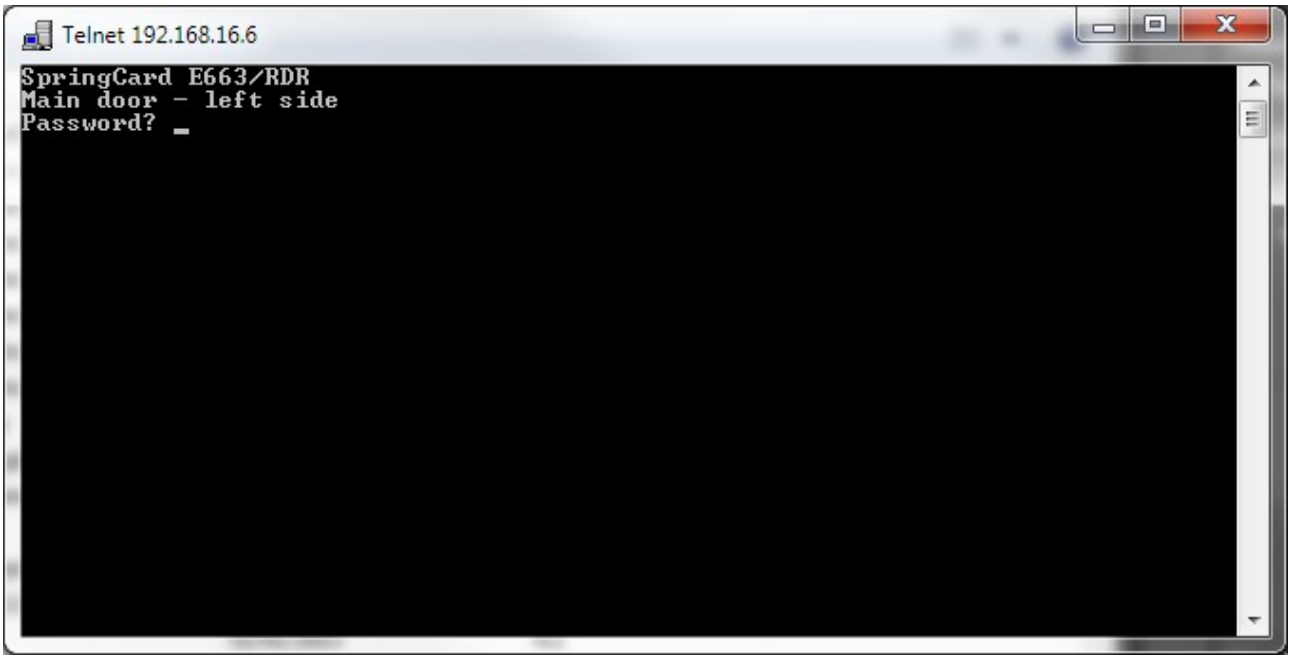
where *xxx.xxx.xxx.xxx* is the Reader's IP Address as defined in chapter 2.

*Windows Vista / 7 / 8 : the Telnet client may be missing from you OS default install. Go to **Control Panel**, **Programs and Features** section, and then enable **Telnet client** in the **Turn Windows features on or off** tab.*
*Alternatively, you may download a free terminal client such as **Putty**, that is also a Telnet client.*

The Reader's Telnet shell says “SpringCard E663/RDR”, then the Info / Location string that has been entered in chapter 2, and finally prompts for a password.

Enter the Reader's password that you've defined in chapter 2.

*If you haven't changed the password, the default password is **springcard**.*



3.1.2. Sending a command to the Reader

Write the command line as documented below, and terminate by hitting the ENTER key.

Note that the Reader echoes the entered characters.

3.1.3. List of Console commands

Command	Meaning
version	Show the firmware version
info	Show the firmware information data
show	Show the current configuration
cfg	Dump all Configuration Registers written into persistent memory
cfgXX=YY...YY	Write value $_h$YY...YY to Configuration Register $_h$XX
cfgXX=!!	Erase Configuration Register $_h$XX
cfgXX	Read Configuration Register $_h$XX
exit	Terminate the Telnet session

4. Using the reader in HTTP mode – REST API

4.1. Abstract

The Reader runs a tiny embedded HTTP (web) server that listens on TCP port 80 and provides a basic REST API. The API provides its results as JSON structures, or as a script, containing a single JavaScript function call (the parameter to the function call being the same JSON structure).

To get started with REST and JSON, please read

- *http://en.wikipedia.org/wiki/Representational_State_Transfer*
- *http://en.wikipedia.org/wiki/JSON*

This feature makes it possible to prototype some applications using the reader in HTTP-client mode.

The HTTP mode is intrisically unsecure. Real-word access control applications shall be built on top of the Authenticated protocol depicted in chapter Erreur : source de la référence non trouvée, and not on top of HTTP.

4.2. Enabling the HTTP server

The HTTP (web) server is disabled by default for security reason.

To enable it, set bit 6 to 1 in configuration register SEC ($_h$6E, § 8.4).

When the HTTP server is enabled, it remains possible to connect the Reader using the specific TCP protocol. Nevertheless, it is not adviced to access the Reader from both interfaces at the same time.

4.3. Limitations

Do not query the Reader's HTTP (web) server more than 4 times per second (i.e. once every 250ms) to let the Reader perform its core job of being a Reader.

4.4. REST API – Function list

Resource	Description	See §
Functions that returns a JSON response		
GET iwm2/read	Return the Card Identifier of the last Card that has been read. The Card Identifier is cleared afterwards.	4.5.1
GET iwm2/read/keep	Return the Card Identifier of the last Card that has been read. The Card Identifier is not cleared afterwards.	4.5.2
GET iwm2/buzz/{time_ms}	Drive the buzzer	4.6.1
GET iwm2/led/{color}/{mode}	Drive one LED (permanent)	4.6.2
GET iwm2/led/{color}/{mode}/{time_ms}	Drive one LED (temporary)	4.6.2
GET iwm2/leds/auto	Go back to default mode for both LEDs	4.6.3
Functions that returns a script (containing a JavaScript function call)		
GET iwm2_cb/read	Return the Card Identifier of the last Card that has been read. The Card Identifier is cleared afterwards.	4.5.1
GET iwm2_cb/read/keep	Return the Card Identifier of the last Card that has been read. The Card Identifier is not cleared afterwards.	4.5.2
GET iwm2_cb/buzz/{time_ms}	Drive the buzzer	4.6.1
GET iwm2_cb/led/{color}/{mode}	Drive one LED (permanent)	4.6.2
GET iwm2_cb/led/{color}/{mode}/{time_ms}	Drive one LED (temporary)	4.6.2
GET iwm2_cb/leds/auto	Go back to default mode for both LEDs	4.6.3

4.5. REST API – Polling loop

4.5.1. iwm2/read and iwm2_cb/read

This is the Reader's main entry point. Invoke this function to get the identifier of the last card that has been read by the Reader.

The function must be invoked at least every 60 seconds, otherwise the Reader signals the "disconnected" state on its LEDs.

> *The last identifier is cleared when this function returns. Use iwm2/read/keep if the identifier must remain visibible for another client request.*

a. iwm2/read

Invoke **iwm2/read**

The Reader returns a JSON structure. 3 cases are possible:

No card has been read since last invokation – last identifier is empty

The Reader returns:

```
{ "iwm2": {
      "success": "true",
      "id": ""
}}
```

A card has been read less than 1 second ago

The Reader returns:

```
{ "iwm2": {
      "success": "true",
      "id": "The card's identifier"
}}
```

A card has been read more than 1 second ago (and less than 60 seconds ago)

The Reader returns:

```
{ "iwm2": {
      "success": "true",
      "id": "The card's identifier",
      "seconds_ago": "Number of seconds elapsed since the card has been read"
}}
```

b. iwm2_cb/read

Invoke **iwm2_cb/read**

The Reader returns a script. 3 cases are possible:

No card has been read since last invokation – last identifier is empty

The Reader returns:

```
iwm2_cb( { "iwm2": {
      "success": "true",
      "id": ""
}});
```

A card has been read less than 1 second ago

The Reader returns:

```
iwm2_cb( { "iwm2": {
      "success": "true",
      "id": "The card's identifier"
}});
```

A card has been read more than 1 second ago (and less than 60 seconds ago)

The Reader returns:

```
iwm2_cb( { "iwm2": {
      "success": "true",
      "id": "The card's identifier",
      "seconds_ago": "Number of seconds elapsed since the card has been read"
}});
```

4.5.2. iwm2/read/keep and iwm2_cb/read/keep

Same as **iwm2/read** and **iwm2_cb/read/keep** but the last identifier remains alive.

4.6. REST API – Sending commands to the Reader

4.6.1. iwm2/buzz and iwm2_cb/buzz

Turns the buzzer ON/OFF.

- To turn the buzzer ON, invoke **iwm2/buzz/{time_ms}** where **{time_ms}** is the decimal value of the sound's duration, expressed in milliseconds (1 to 65534).
- To turn the buzzer OFF, invoke **iwm2/buzz/0**.

a. Iwm2/buzz

Invoke **iwm2/buzz/{time_ms}**

The Reader returns:

```
{ "iwm2": {
      "success": "true"
}}
```

b. iwm2_cb/buzz

Invoke **iwm2_cb/buzz/{time_ms}**

The Reader returns:

```
iwm2_cb( { "iwm2": {
      "success": "true"
}});
```

4.6.2. iwm2/led and iwm2_cb/led

This command makes it possible to drive the Reader's LEDs.

Complete syntax is **iwm2/led/{color}/{mode}/{time_ms}**

Allowed values for the **{color}** parameter are

- **red**
- **green**

The Blue LED can't be driven explicitely.

Allowed values for the **{mode}** parameter are

- **on**
- **fast**
- **slow**
- **heart**

The **{time_ms}** parameter is the decimal value of the command's duration, expressed in milliseconds (1 to 65534). Suppress the **{time_ms}** parameter (or set it to 0) to make it permanent.

a. Iwm2/led

Invoke **iwm2/led/{color}/{mode}/{time_ms}**

The Reader returns:

```
{ "iwm2": {
      "success": "true"
}}
```

b. iwm2_cb/led

Invoke **iwm2_cb/led/{color}/{mode}/{time_ms}**

The Reader returns:

```
iwm2_cb( { "iwm2": {
      "success": "true"
}});
```

4.6.3. iwm2/leds and iwm2_cb/leds

Invoke command **iwm2/leds/auto** to cancel a pending LED command (§ 4.6.3).

a. Iwm2/leds

Invoke **iwm2/leds/auto**

The Reader returns:

```
{ "iwm2": {
      "success": "true"
}}
```

b. iwm2_cb/leds

Invoke **iwm2_cb/leds/auto**

The Reader returns:

```
iwm2_cb( { "iwm2": {
      "success": "true"
}});
```

4.7. REST API – Errors

a. iwm2 namespace

Upon any error, the Reader returns:

```
{ "iwm2": {
      "success": "false"
}}
```

b. iwm2_cb namespace

Upon any error, the Reader returns:

```
iwm2_cb( { "iwm2": {
      "success": "false"
}});
```

5. SpringCard Network Device C/S Protocol

5.1. Abstract

SpringCard Network Device C/S Protocol is a light-weight, bandwidth-efficient network protocol. The Reader is a TCP Server, and the Host (access control unit or computer in charge) is the Client.

Note that the Reader is not able to accept more than one Client at the time. Trying to connect to the same Reader from two different Host is not supported, and shall not be tried. An undefined behaviour may occur.

There are two communication modes:

- **Plain mode**, with no security involved
- **Secure mode**, based on AES cryptography.

In Secure mode, there are two authentication keys, leading to two authentication levels:

- The **Operation key** gives access to the basic operations of the Reader (§ 6.3). This is the key an access control unit would use to operate the Reader.
- The **Administration Key** makes it possible to change the Readerss configuration (§ 6.4). This key would typically used by a configuration software, when installing the Reader.

The Protocol is fully documented in [PMA14166], chapters 5 and 6.

6. SpringCard Network Device – Reader Application Layer

6.1. Principles

This chapter describes the **Reader Application Layer**, whose Application-Level Datagrams are transmitted on top of the Protocol described in [PMA14166], chapters 5 and 6.

The Application Layer is fully documented in [PMA14166], chapter 7. This chapter is only an extract that summarizes only the Reader's features.

> *The **SpringCard Network Device C/S Protocol** is not a Request/Response Protocol; since a TCP channel is truly full-duplex, both the Host and the Reader may talk at any time. Therefore, the Host must be ready to process (or at least to queue) an Application-Level Datagram coming from the Reader at any time.*

6.2. List of operation-codes and data-field identifiers

6.2.1. Operation-codes (Host → Reader)

T (Tag)	Operation	See §
h00	Get Global Status	6.3.1
h0A	Start / Stop Reader	6.3.2
hD000	Clear LEDs Set LEDs Start LEDs	6.3.3 6.3.4 6.3.5
hD100	Buzzer	6.3.6
	Restricted operations (available only after authentication using Administration Key)	
h0C	Write Configuration Erase Configuration	6.4.1 6.4.2
h0B	Reset the Reader (to apply the Configuration)	6.4.3

6.2.2. Data-field identifiers (Reader → Host)

T (Tag)	Operation	See §
h8100	Reader Identifier	6.5.1
h2F	Tamper Status	6.5.2
hB000	Card Read	6.5.3
hB100	Card Inserted Card Removed	6.5.4 6.5.5

6.3. Host → Reader, basic operations

The operations listed in this chapter are available **whatever the mode**:

- Plain (no authentication),
- Secure, after authentication using the Operation Key,
- Secure, after authentication using the Administration Key.

6.3.1. Get Global Status

T	L
$_h$00	$_h$00

The Reader answers by 2 frames:

1. Reader Identifier
2. Tamper Status

6.3.2. Start/Stop Reader

T	L	V
$_h$0A	$_h$01	**mode**

- **mode:** start/stop command
  - $_h$00 Reader goes OFF (RF field OFF, no activity on RF)
  - $_h$01 Reader goes ON

6.3.3. Clear LEDs command

Both LEDs go OFF.

T	L
$_h$D000	$_h$00

6.3.4. Set LEDs command

Both LEDs are driven – until a Clear LEDs command is received.

T	L	V	
$_h$D000	$_h$02	red	green

- **red:** command for red LED
  - $_h$00 OFF
  - $_h$01 ON
  - $_h$02 blinks slowly
  - $_h$03 blinks quickly
- **green:** command for green LED
  - $_h$00 OFF
  - $_h$01 ON
  - $_h$02 blinks slowly
  - $_h$03 blinks quickly

6.3.5. Start LED sequence command

Both LEDs are driven – until a Clear LEDs command is received or a timeout occurs.

T	L	V		
$_h$D000	$_h$04	red	green	time (sec)

- **red:** same as above,
- **green:** same as above,
- **time:** time (in seconds, MSB-first) before returning to all-LED-OFF state.

6.3.6. Buzzer command

T	L	V
$_h$D100	$_h$01	seq.

- **seq:**
  - $_h$00 buzzer OFF,
  - $_h$01 buzzer ON,
  - $_h$02 buzzer short sequence,
  - $_h$03 buzzer long sequence.

6.4. Host → Reader, restricted operations

The operations listed in this chapter are available only in **Secure mode, after authentication using the Administration Key**.

6.4.1. Write Configuration Register

The Reader's behaviour is defined by Configuration Registers documented in chapters 8 and 9. The Write Configuration Register command allows to write into any Configuration Register given its address.

<addr> is the Register number on one byte (valid values are $_h$00 to $_h$FE).

T	L	V	
$_h$0C	<var.>	<addr>	<value>

6.4.2. Erase Configuration Register

The Reader's behaviour is defined by Configuration Registers documented in chapters 8 and 9. The Erase Configuration Register command allows to delete any Configuration Register given its address. Once a Register is deleted, the default value for this Register is used.

<addr> is the Register number on one byte (valid values are $_h$00 to $_h$FE).

T	L	V
$_h$0C	$_h$01	<addr>

6.4.3. Reset the Reader

The Reader must be re-setted in order for the new configuration to take effect. When receiving this command, the Reader drops the connection and resets.

T	L	V	
$_h$0B	$_h$02	$_h$DE	$_h$AD

6.5. Reader → Host

6.5.1. Reader Identifier

This T,L,V is transmitted in response to the **Get Global Status** command.

T	L	V
$_h$8100	$_h$1C	SpringCard E663/RDR x.xx

6.5.2. Tamper Status

This T,L,V is transmitted in response to the **Get Global Status** command or when one of the tampers is broken/restored.

T	L	V
$_h$2F	$_h$01	Bit field, the broken tampers are denoted by the corresponding bit set to 1. V = $_h$00 when all tampers are OK.

6.5.3. Card Read

This T,L,V is transmitted when the Reader has read a card, if the Insert/Remove mode is disabled.

T	L	V
$_h$B000	<var.>	Card Identifier

6.5.4. Card Inserted

This T,L,V is transmitted when the Reader has read a card, if the Insert/Remove mode is enabled.

T	L	V
$_h$B100	<var.>	Card Identifier

6.5.5. Card Removed

This T,L,V is transmitted when the card is removed, if the Insert/Remove mode is enabled.

T	**L**
$_{h}$B100	$_{h}$00

7. Editing Reader's configuration

The Reader's configuration is stored in a set of non-volatile Configuration Registers. There are two groups of Registers:

- The Registers that control the behaviour of the Reader are fully documented in chapter 8. Some of them are common to various SpringCard Readers, but some of them are very specific to the **SpringCard FunkyGate-IP NFC**.
- The Registers that control the Template System are shared among all SpringCard Readers. Chapter 9 is therefore a place-holder that redirects to the document describing this Template System precisely.

But this subtle distinction between these two groups is only there to keep the documents short, and to ease switching from one Reader to the other. Technically speaking, all Registers are defined (and accessed) the same way.

There are four ways to edit the Reader's Configuration Registers:

1. Through the Telnet link
2. Using Master Cards
3. Using the SpringCard Network Device C/S Protocol, after authentication with Administration Key.

> *Note that the SEC Configuration Register ($_h$6E, § 8.4) may be used to disable either way to access the Configuration Registers.*
> *Administration Key is defined in the IPS Configuration Register ($_h$83, § 8.5.3)*

7.1. Through the Telnet link

Open a Telnet session to the Reader as instructed in § 3.1.

7.1.1. Reading Configuration Registers

Enter “cfg” to list all Configuration registers currently defined (registers that are not explicitely defined keep their default value).

Enter “cfgXX” to read the value of the Configuration register $_h$XX.

Note that Configuration registers $_h$55, $_h$56, $_h$6E and $_h$6F that hold sensitive data (the keys used by Master Cards and the Reader's secret keys and password) are masked.

7.1.2. Writing Configuration Registers

Enter “cfgXX=YYYY” to update Configuration Register $_h$XX with value $_h$YYYY. YYYY can me any length between 1 and 32 bytes.

Enter “cfgXX=!!” to erase Configuration Register $_h$XX.

7.2. Using Master Cards

The Master Cards are NXP Desfire cards formatted and programmed by **SpringCard Configuration Tool (ScMultiConf.exe, ref # SN14007)** for Windows.

Please refer to this software's documentation for details.

7.3. Through the SpringCard Network Device C/S protocol

Please refer to [PMA14166] § 7.5 .

8. Global configuration of the Reader

8.1. General options

Name	Tag	Description	Size
OPT	$_h$60	General option, see table below	1 or 2

General options bits

Bits	Value	Meaning
		Byte 0
7	0	Normal mode
	1	Power saving mode (the Reader is slower)
6	0	Track the cards by their ID only
	1	Keep the RF field active to track the cards (works with Random IDs)
5 - 4		***Anti-collision mode***
	00	Read every card one after the other
	01	*RFU*
	10	Read only one card at a time (ignore the other ones)
	11	Prevent reading when there's more than one card in the field
3 - 2		***Master Card and NFC configuration***
	00	Disable configuration by Master Card or NFC
	01	Allow configuration by Master Card or NFC at power up only
	10	*RFU*
	11	Allow configuration by Master Card or NFC all the time
1	0	*RFU (set to 0)*
0	0	*RFU (set to 0)*
		Byte 1 (optional)
7	0	*RFU (set to 0)*
6	0	*RFU (set to 0)*
5	0	*RFU (set to 0)*
4	0	Insert/Remove mode is disabled (§ 6.5.3)
	1	Insert/Remove mode is enabled (§ 6.5.4 and § 6.5.5)
3	0	*RFU (set to 0)*
2	0	*RFU (set to 0)*
1	0	*RFU (set to 0)*
0	0	Reader is active on startup
	1	Reader is not active on startup (Host must send an activation command)

Default value: $_b$00001100 00000000

8.2. Delays and repeat

Name	Tag	Description	Min	Max
ODL	$_h$61	Min. delay between 2 consecutive outputs (0.1s)	0	100
RDL	$_h$62	Min. delay between 2 consecutive identical outputs (0.1s) A value of 255 means that the card must be removed from the field –and re-inserted into– before being read again	0	100

Default value: ODL = 5 (1ms) RDL = 20 (2s)

8.3. LEDs and Buzzer

Name	Tag	Description	Size
CLD	$_h$63	LEDs control, see table below	1
CBZ	$_h$64	Buzzer control, see table below	1

LEDs control bits

Bits	Value	Meaning
7	0	Short LED sequences (3 seconds)
	1	Long LED sequences (10 seconds)
6 - 5	00	When idle, blue LED blinks slowly (“heart beat” sequence)
	01	When idle, blue LED is always on
	10	When idle, blue LED is always off
	11	RFU
4	0	Green LED stays OFF
	1	Green LED blinks when a valid card has been processed
3	0	Red LED stays OFF
	1	Red LED blinks when an unsupported card has been processed
2	0	Green LED stays OFF
	1	Green LED blinks as soon as a card is seen in the field
1 - 0	00	*RFU, do not use*
	01	LED driven by Host commands only
	10	*RFU, do not use*
	11	LED driven by internal state machine and Host commands

Default value: $_b$00001111

Buzzer control bits

Bits	Value	Meaning
7	0	Buzzer short pulse = 0,2 sec
	1	Buzzer short pulse = 0,5 sec
6	0	Buzzer long pulse = 0,7 sec
	1	Buzzer long pulse = 1,5 sec
5		*RFU*
4	0	No action on buzzer before specified by host controller
	1	Short pulse when a valid card has been processed
3	0	No action on buzzer for unsupported cards
	1	Long pulse when an unsupported card has been processed
2	0	No action on buzzer before processing is achieved
	1	Short pulse as soon as a card is seen in the field
1 - 0	00	Buzzer is disabled, other settings are ignored
	01	Buzzer controlled by serial commands, other settings are ignored
	10	Buzzer controlled by internal software, serial commands are ignored
	11	Buzzer controlled by both internal software and serial commands

Default value : $_b$00010010

8.4. Security options

Name	Tag	Description	Size
SEC	$_h$6E	Security option bits. See table a below	1

Security option bits

Bits	Value	Meaning
		Standard network servers
7	0	Telnet server is disabled
	1	Telnet server is enabled
6	0	HTTP (web) server is disabled
	1	HTTP (web) server is enabled
5	0	*RFU (set to 0)*
4	0	*RFU (set to 0)*
3	0	*RFU (set to 0)*
		Tampers
2	0	Do not signal tamper alarms on buzzer
	1	Signal tamper alarms on buzzer
1	0	Reader keeps on reading even if a tamper is broken
	1	Reader stops reading when a tamper is broken
0	0	Do not raise alarm if a tamper is broken at power up
	1	Raise alarm on tamper broken even at power up

Default value: $_b$10000100

8.5. TCP CONFIGURATION

8.5.1. IPv4 address, mask, and gateway

Name	Tag	Description	Size
IPA	$_h$80	IPv4 configuration bytes, see table below	4, 8 or 12

IPv4 configuration bytes

Bytes	Contains	Remark
0	Address, 1st byte	Reader's IPv4 Address.
1	Address, 2nd byte	
2	Address, 3rd byte	If these bytes are missing, the default IP Address $_h$C0 A8 00 FA (192.168.0.250) is taken.
3	Address, 4th byte	
4	Mask, 1st byte	Network Mask.
5	Mask, 2nd byte	
6	Mask, 3rd byte	If these bytes are missing, the default Mask $_h$FF FF FF FF (255.255.255.0) is taken.
7	Mask, 4th byte	
8	Gateway, 1st byte	Default Gateway.
9	Gateway, 2nd byte	
10	Gateway, 3rd byte	If these bytes are missing, the value $_h$00 00 00 00 (0.0.0.0) is taken, meaning that there's no Gateway.
11	Gateway, 4th byte	

Default value: $_h$C0 A8 00 FA FF FF FF 00 00 00 00 00

(address = 192.168.0.250, mask = 255.255.255.0, no gateway)

8.5.2. SpringCard Network Device C/S Protocol – Server port

Name	Tag	Description	Size
IPP	$_h$81	Listen TCP port for the server (2 bytes, MSB-first)	2

Default value: $_h$0F 9F (server TCP port = 3999)

8.5.3. SpringCard Network Device C/S Protocol – Security settings and authentication keys

Name	Tag	Description	Size
IPS	$_h$84	Server security settings bits, see table below	1

Security settings bits

Bits	Value	Meaning
7	0	*RFU (set to 0)*
6	0	*RFU (set to 0)*
5	0	*RFU (set to 0)*
4	0	*RFU (set to 0)*
3	0	*RFU (set to 0)*
2	0	The Administration Key is enabled
	1	The Administration Key is disabled
1	0	The Operation Key is enabled
	1	The Operation Key is disabled
0	0	Plain communication is allowed
	1	Secure communication is mandatory

Default value: $_b$00000100

(only Operation Key is enabled, plain communication is allowed)

8.5.4. SpringCard Network Device C/S Protocol – Operation Key

Name	Tag	Description	Size
IPK.OPE	$_h$85	C/S Protocol Operation Key	16

Default value: $_h$00 … $_h$00

8.5.5. SpringCard Network Device C/S Protocol – Administration Key

Name	Tag	Description	Size
IPK.ADM	$_h$86	C/S Protocol Administation Key	16

Default value: $_h$00 … $_h$00

8.5.6. Ethernet configuration

Name	Tag	Description	Size
ETC	$_h$8D	Ethernet configuration bits. See table a below	1

Ethernet configuration bits

Bits	Value	Meaning
7	0	*RFU (set to 0)*
6	0	*RFU (set to 0)*
5	0	*RFU (set to 0)*
4	0	*RFU (set to 0)*
3	0	*RFU (set to 0)*
2	0	*RFU (set to 0)*
1	0	*RFU (set to 0)*
0	0	Use auto-configuration (10/100Mbps, half or full-duplex)
	1	Force bitrate = 10Mbps, half-duplex

Default value: $_b$00000000

8.5.7. Info / Location

Name	Tag	Description	Size
ILI	$_h$8E	Info / Location string	Var. 0-30

Default value: empty

The **Info / Location** string is a text value (ASCII) that appears

- When someone tries to connect on Telnet,
- In the NDDU software (§ 2.1.3).

Use this string as a reminder of where your Reader is installed, or what is its role in your access-control system.

8.5.8. Password for Telnet access

Name	Tag	Description	Size
ITP	$_h$8F	Password for Telnet access string	Var. 0-16

Default value: “springcard”

The **Password for Telnet access** string is a text value (ASCII) that protects the access to the Reader using Telnet protocol.

The password is mandatory. If this registry is not set, default value “springcard” is used.

9. The Template System

SpringCard FunkyGate-IP NFC provides 4 “Card Processing Templates” that defines how the Reader which fetch data from various cards/tags, and how the Card Identifier will be constructed from these data before being sent to the Host.

The template system is fully described in document **[PMA13205] “RFID/NFC Scanners Template System**”.

Please use this document as reference to configure the “Reader part” of your **SpringCard FunkyGate-IP NFC.**

10. 3rd-party licenses

SpringCard FunkyGate-IP NFC has been developed using open-source software components.

10.1. FreeRTOS

FreeRTOS is a market leading real time operating system (or RTOS) from Real Time Engineers Ltd. **SpringCard FunkyGate-IP NFC** runs on FreeRTOS v7.5.2.

FreeRTOS is distributed under a modified GNU General Public License (GPL) that allows to use it in commercial, closed-source products.

For more information, or to download the source code of FreeRTOS, please visit

www.freertos.org

10.2. uIP

µIP is an open-source TCP/IP stack initially developed by Adam Dunkels and licensed under a BSD style license.

SpringCard FunkyGate-IP NFC uses FreeTCPIP, a modified version of µIP that comes with FreeRTOS. To comply with the original license of µIP, we have to copy the full text here:

> Copyright (c) 2001-2003, Adam Dunkels.
>
> All rights reserved.
>
> Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
>
> 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
>
> 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
>
> 3. The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

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

Editor's information

PRO ACTIVE 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