



PMA14166-BA
FINAL - PUBLIC

SPRINGCARD E663/RDR, FUNKYGATE-IP NFC, E663/MIO, HANDYDRUMMER-IP

Network Integration and Configuration

DOCUMENT IDENTIFICATION

Category	Admin/Config Manual		
Family/Customer	Network Devices		
Reference	PMA14166	Version	BA
Status	Final	Classification	Public
Keywords			
Abstract			

File name	V:\Dossiers\SpringCard\A-Notices\RFID scanners et lecteurs\IWM2-Commun\[PMA14166-BA] E663-RDR, FunkyGate-IP, E663-MIO, HandyDrummer-IP Network Integration and Configuration.odt		
Date saved	11/12/18	Date printed	

REVISION HISTORY

Ver.	Date	Author	Valid. by		Approv. by	Details
			Tech.	Qual.		
AA	30/07/14	JDA				Created from PMA13257-AC
AB	12/08/14	JDA				Defined the HTTP REST API for I/O Modules Improved the explanation in 8.3.6 and 8.6.1.b Fixed the definition of p in 8.6.2.b
AC	09/09/16	JDA				Added support for DNS client (resolv) in version 1.69 of firmware Removed HTTP REST API (moved to product's specific documentation)
BA	11/12/18	JDA				New implementation of the SpringCard MK2 protocol on top of UDP (previously on top of TCP only)

CONTENTS

1. INTRODUCTION.....	7	7.1. PRESENTATION LAYER.....	28
1.1. ABSTRACT.....	7	7.1.1. Block format.....	28
1.2. DOCUMENT ORGANISATION.....	7	7.1.2. Description of the fields.....	28
1.3. AUDIENCE.....	8	7.1.3. Size of the blocks.....	29
1.4. SUPPORT AND UPDATES.....	8	7.1.4. Format of the TYPE byte.....	29
1.5. RELATED DOCUMENTS.....	8	7.2. GENERAL COMMUNICATION FLOW.....	31
1.5.1. Products' specifications.....	8	7.2.1. Session establishment.....	31
1.5.2. Products' specific technical documentations.....	8	7.2.2. Nominal dialog.....	31
2. DEFINE THE DEVICE'S IP ADDRESS.....	9	7.2.3. Timings.....	32
2.1. ASSIGN AN IP ADDRESS USING NDDU SOFTWARE.....	9	7.2.4. Chaining.....	32
2.1.1. Download and install the NDDU software.....	9	7.3. ERROR HANDLING AND RECOVERY.....	33
2.1.2. Run the NDDU software.....	9	7.3.1. For the Device.....	33
2.1.3. Discovered devices.....	10	7.3.2. For the Host.....	33
2.1.4. Configure a Device.....	11	7.3.3. Recovery.....	33
2.1.5. Verify the new configuration.....	13	7.4. APPL LEVEL.....	33
2.2. ASSIGN AN IP ADDRESS TO A READER USING A MASTER CARD.....	14	8. SPRINGCARD "MK2" NETWORK PROTOCOL – TCP SECURE MODE.....	34
3. TELNET ACCESS TO THE DEVICE.....	15	8.1. ABSTRACT.....	34
3.1. THE DEVICE'S CONSOLE.....	15	8.2. CRYPTOGRAPHIC BACKGROUND.....	35
3.1.1. Open a Telnet session to the Device.....	15	8.3. 3-PASS AUTHENTICATION.....	35
3.1.2. Sending a command to the Device.....	16	8.3.1. Device's HELO.....	35
3.1.3. List of Console commands.....	17	8.3.2. Host's HELO-Auth.....	36
4. EDITING DEVICE'S CONFIGURATION.....	18	8.3.3. Authentication, step 1.....	36
4.1. THROUGH THE TELNET LINK.....	18	8.3.4. Authentication, step 2.....	37
4.1.1. Reading Configuration Registers.....	18	8.3.5. Authentication, step 3.....	37
4.1.2. Writing Configuration Registers.....	19	8.3.6. Conclusion of the Authentication sequence – Host's HELO-OK.....	38
4.2. USING MASTER CARDS (ONLY AVAILABLE ON A READER).....	19	8.4. PRESENTATION LAYER AFTER AUTHENTICATION.....	38
4.3. THROUGH THE SPRINGCARD "MK2" NETWORK PROTOCOL.....	19	8.4.1. Block format.....	38
5. SPRINGCARD "MK2" NETWORK PROTOCOL – OVERVIEW.....	20	8.4.2. Description of the fields.....	39
5.1. OPERATION OVER UDP.....	20	8.4.3. Size of the blocks.....	39
5.1.1. Standard operation (Unicast).....	20	8.4.4. Format of the TYPE byte.....	39
5.1.2. Operation using UDP broadcasts.....	21	8.5. SESSION KEYS.....	40
5.2. OPERATION OVER TCP.....	22	8.5.1. Session Encryption Key.....	40
5.3. KNOWN LIMITATIONS OF THE DEVICE AND PRECAUTIONS TO BE TAKEN ON THE HOST SIDE.....	23	8.5.2. Session CMAC Key.....	41
6. SPRINGCARD "MK2" NETWORK PROTOCOL – UDP MODE 24		8.6. SECURE COMMUNICATION CHANNEL.....	41
6.1. PRESENTATION LAYER.....	24	8.6.1. CMAC.....	42
6.1.1. Block format.....	24	8.6.2. AES-CBC encryption.....	43
6.1.2. Description of the fields.....	25	8.6.3. Receiving.....	45
6.1.3. Size of the blocks.....	25	8.7. NEW AUTHENTICATION – GENERATION OF A NEW SESSION KEY.....	45
6.1.4. Format of the TYPE byte.....	26	8.8. GENERAL COMMUNICATION FLOW.....	45
6.2. GENERAL COMMUNICATION FLOW.....	26	8.8.1. Nominal dialog.....	45
6.2.1. Nominal dialog.....	26	8.8.2. Timings.....	45
6.2.2. Timings.....	27	8.8.3. Chaining.....	46
6.2.3. Chaining.....	27	8.9. ERROR HANDLING AND RECOVERY.....	46
6.3. APPL LEVEL.....	27	8.9.1. For the Device.....	46
7. SPRINGCARD "MK2" NETWORK PROTOCOL – TCP PLAIN MODE.....	28	8.9.2. For the Host.....	46
		8.9.3. Recovery.....	46
		8.10. APPLICATION LAYER.....	47
		9. SPRINGCARD "MK2" NETWORK PROTOCOL – APPLICATION LAYER.....	48

9.1. PRINCIPLES.....	48	11. 3RD-PARTY LICENSES.....	67
9.2. FORMAT OF THE APPLICATION LEVEL DATAGRAM UNITS.....	48	11.1. FREERTOS.....	67
9.3. LIST OF COMMANDS AND DATA-FIELD IDENTIFIERS.....	49	11.2. UIP.....	68
9.3.1. Command codes (Host → Device).....	49		
9.3.2. Event codes and response codes (Device → Host).....	50		
9.4. HOST → DEVICE, BASIC OPERATIONS.....	51		
9.4.1. Get Global Status.....	51		
9.4.2. Get Device Name.....	51		
9.4.3. Get Device Capabilities.....	51		
9.4.4. Get Device Serial Number.....	52		
9.4.5. Read Inputs (<i>MIO only</i>).....	52		
9.4.6. Start/Stop Reader (<i>RDR only</i>).....	52		
9.4.7. Set Output command (<i>MIO only</i>).....	53		
9.4.8. Clear Output command (<i>MIO only</i>).....	53		
9.4.9. Clear LEDs command (<i>RDR only</i>).....	54		
9.4.10. Set LEDs command (<i>RDR only</i>).....	54		
9.4.11. Start LED sequence command (<i>RDR only</i>).....	55		
9.4.12. Buzzer command (<i>RDR only</i>).....	55		
9.5. HOST → DEVICE, RESTRICTED OPERATIONS.....	56		
9.5.1. Write Configuration Register.....	56		
9.5.2. Erase Configuration Register.....	56		
9.5.3. Reset the Device.....	56		
9.6. DEVICE → HOST.....	57		
9.6.1. Device Name.....	57		
9.6.2. Device Capabilities.....	57		
9.6.3. Device Serial Number.....	57		
9.6.4. Reading Head Identifier.....	58		
9.6.5. Tamper Status.....	58		
9.6.6. Card Read (<i>RDR only</i>).....	59		
9.6.7. Card Inserted (<i>RDR only</i>).....	59		
9.6.8. Card Removed (<i>RDR only</i>).....	59		
9.6.9. Reader sequence terminated.....	59		
9.6.10. Input Changed (<i>MIO only</i>).....	60		
10. COMMON CONFIGURATION REGISTERS FOR SPRINGCARD NETWORK DEVICES.....	61		
10.1. GENERAL OPTIONS.....	61		
10.2. SECURITY OPTIONS.....	62		
10.3. TCP CONFIGURATION.....	63		
10.3.1. IPv4 address, mask, and gateway.....	63		
10.3.2. SpringCard “MK2” Network Protocol – Device’s TCP listen port.....	64		
10.3.3. SpringCard “MK2” Network Protocol – Device’s UDP listen port.....	64		
10.3.4. SpringCard “MK2” Network Protocol – Host address and UDP listen port.....	64		
10.3.5. SpringCard “MK2” Network Protocol – Security settings and authentication keys.....	64		
10.3.6. SpringCard “MK2” Network Protocol – Operation Key.....	65		
10.3.7. SpringCard “MK2” Network Protocol – Administration Key.....	65		
10.3.8. Ethernet configuration.....	65		
10.3.9. Info / Location.....	65		
10.3.10. Password for Telnet access.....	66		

1. INTRODUCTION

1.1. ABSTRACT

SpringCard E663/RDR is SpringCard's network-attached, smart Reader core. It provides in the same device a versatile RFID (13.56MHz) and NFC interface, and the logic to fetch data from virtually any compliant card or tag, including secure authentication on popular NXP MIFARE Classic, Plus and DESFire chips.

The **SpringCard E663/RDR** core is notably present in these products:

- **SpringCard FunkyGate-IP NFC** is a wall-mount network-attached Reader, for access control applications,
- **SpringCard FunkyGate-IP+POE NFC** adds the “powered by the network” (POE) feature,
- **SpringCard TwistyWriter-IP/RDR** is an OEM Reader, suitable for integration in kiosks, gate controllers, etc.

SpringCard E663/MIO is SpringCard's network-attached, I/O controller core. It is present in one product:

- **SpringCard HandyDrummer-IP** is a I/O Module, featuring 8 ON/OFF input and 8 output (relays), and highly expandable. The **SpringCard HandyDrummer-IP+POE** version adds the “powered by the network” (POE) feature.

Both **E663/RDR** and **E663/MIO** share the same TCP/IP (IPv4) on top of Ethernet implementation. This document provides all necessary information to perform the network configuration of all the products based on these two cores, and to develop a software that will exchange data with them.

1.2. DOCUMENT ORGANISATION

This document is divided into 3 parts:

- **Chapters 2, 3 and 4** are a kind of **Getting Started guide**: they show how to configure the device onto your network and manage its settings afterwards.
- **Chapters 5, 7, 8, 6 and 9 detail the protocol** between the host and the device. There are three low-level communication protocols (UDP, TCP plain, TCP secure) sharing a single high-level application protocol. Chapter 5 will guide you in selecting the appropriate protocol given the constraints of your system.
- **Chapter 10 lists the Configuration Registers** that control the device.

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 TCP/IP networks.

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
PFL14164	HandyDrummer family leaflet

1.5.2. Products' specific technical documentations

Every product has its own Integration and Configuration Guide that details the parts not covered in this document.

Document ref.	Content
PMA13257	FunkyGate-IP NFC Integration and Configuration Guide
PMA14165	HandyDrummer-IP Integration and Configuration Guide

2. DEFINE THE DEVICE'S IP ADDRESS

The **Device** 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 chapters 5 and 8.

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

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 Device(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 Devices; therefore, it can'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

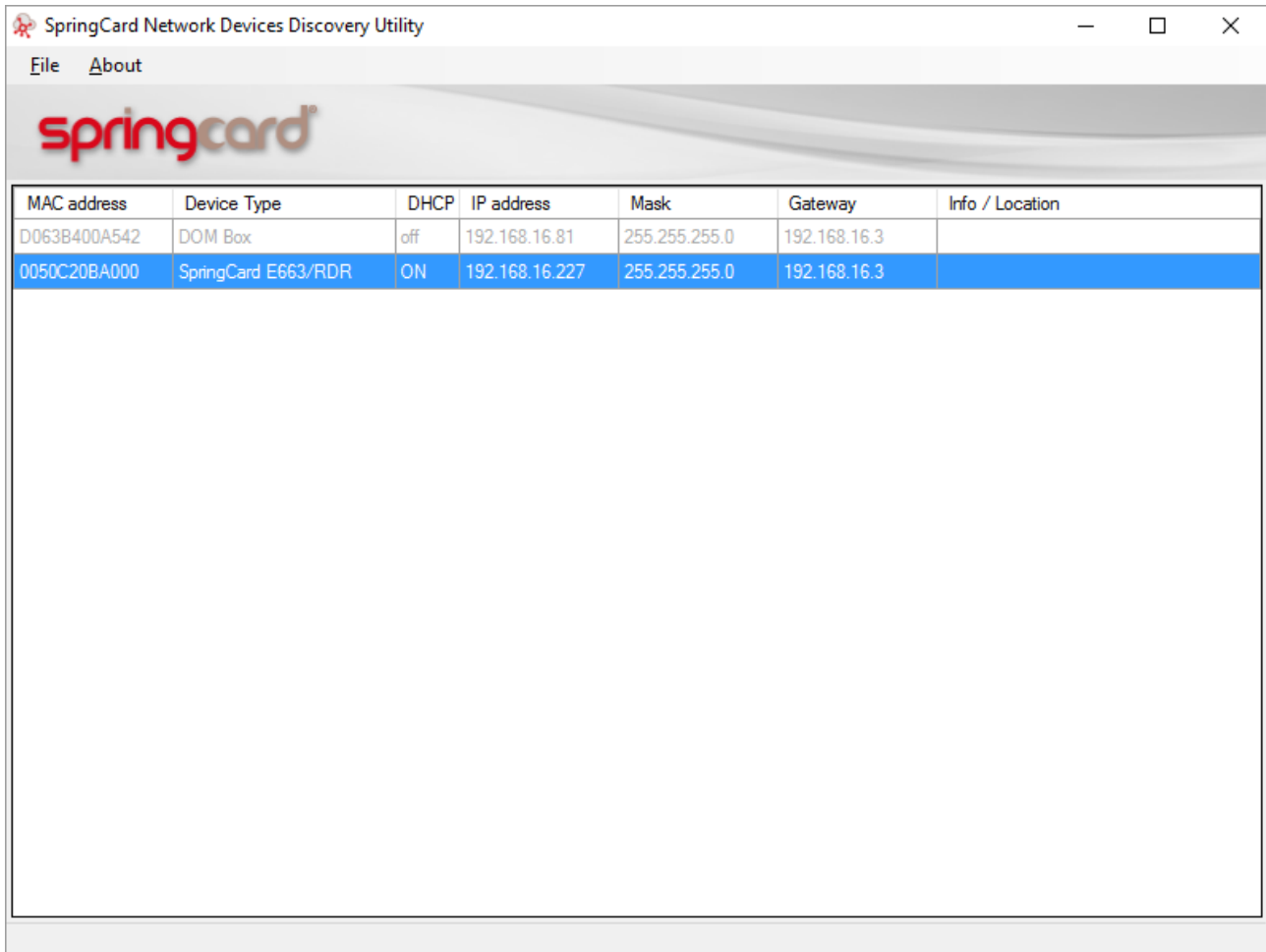
Make sure your Windows account has administrative privileges.

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 screenshot shows a window titled "SpringCard Network Devices Discovery Utility". It has a menu bar with "File" and "About". Below the menu bar is the SpringCard logo. The main area contains a table with the following data:

MAC address	Device Type	DHCP	IP address	Mask	Gateway	Info / Location
D063B400A542	DOM Box	off	192.168.16.81	255.255.255.0	192.168.16.3	
0050C20BA000	SpringCard E663/RDR	ON	192.168.16.227	255.255.255.0	192.168.16.3	

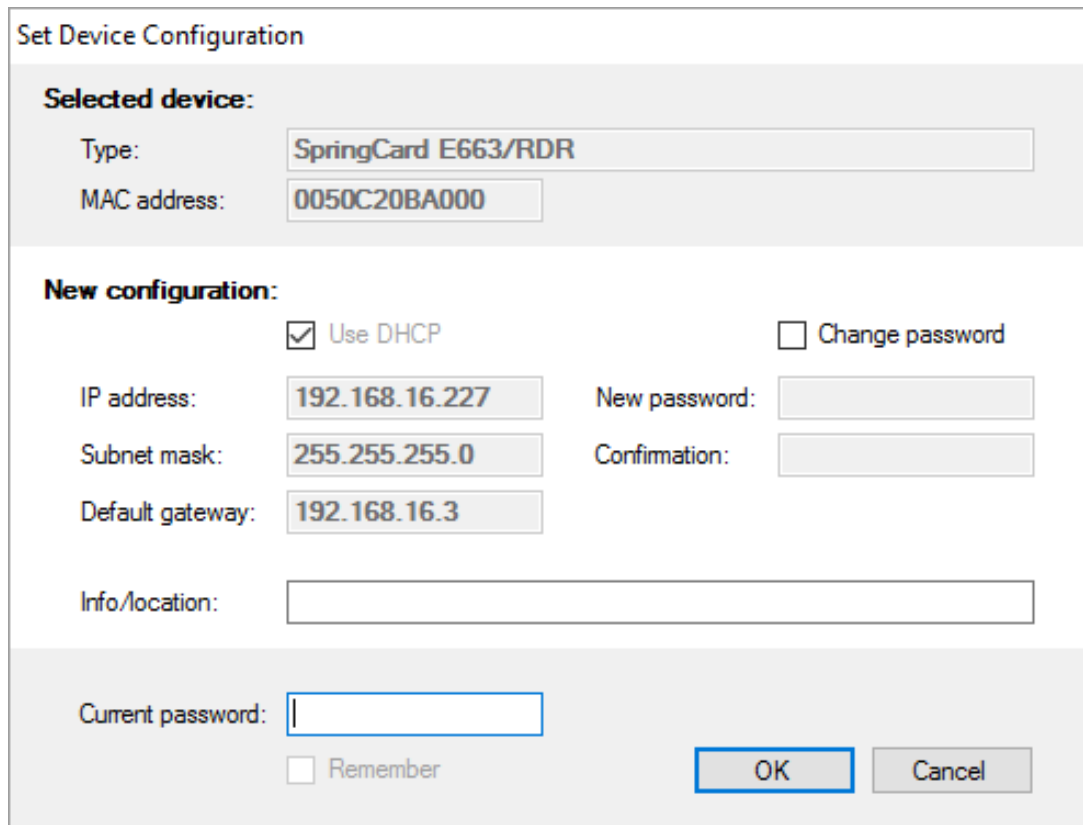
The software's main screen shows 7 columns:

- 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 FunkyGate-IP NFC, FunkyGate-IP+POE NFC, TwistyWriter-IP/RDR, and any future product based on E663/RDR core,
 - code name **SpringCard E663/MIO** for HandyDrummer-IP, HandyDrummer-IP+POE, and any future product based on E663/MIO core,
- Whether DHCP is enabled or not (DHCP is not supported on early firmware versions),
- The device's current IP address, local network mask, and default gateway. Until the device has been properly configured, those entries show as "0.0.0.0",

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

2.1.4. Configure a Device

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



The dialog box titled "Set Device Configuration" is divided into three main sections. The top section, "Selected device:", shows the device type as "SpringCard E663/RDR" and the MAC address as "0050C20BA000". The middle section, "New configuration:", contains several fields: "IP address" (192.168.16.227), "Subnet mask" (255.255.255.0), "Default gateway" (192.168.16.3), "New password" (empty), and "Confirmation" (empty). There are checkboxes for "Use DHCP" (checked) and "Change password" (unchecked). Below these is an "Info/location:" field. The bottom section contains a "Current password:" field (empty) and a "Remember" checkbox (unchecked). At the bottom right are "OK" and "Cancel" buttons.

The form shows the device's current configuration. Enter the new configuration.

a. Use DHCP?

DHCP stands for Dynamic Host Configuration Protocol. Enable DHCP on the device only if there's a DHCP server running on the network.

Note: in most situations, software that use the device will connect as a client to a server service running in the device. If the device uses DHCP, its address is likely to change frequently, and so the client software must be reconfigured accordingly to know the actual address of its server. It is recommended to reserve a permanent lease on the DHCP server for the device to suppress this annoyance.

b. Static configuration

IPv4 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".

c. Info/Location

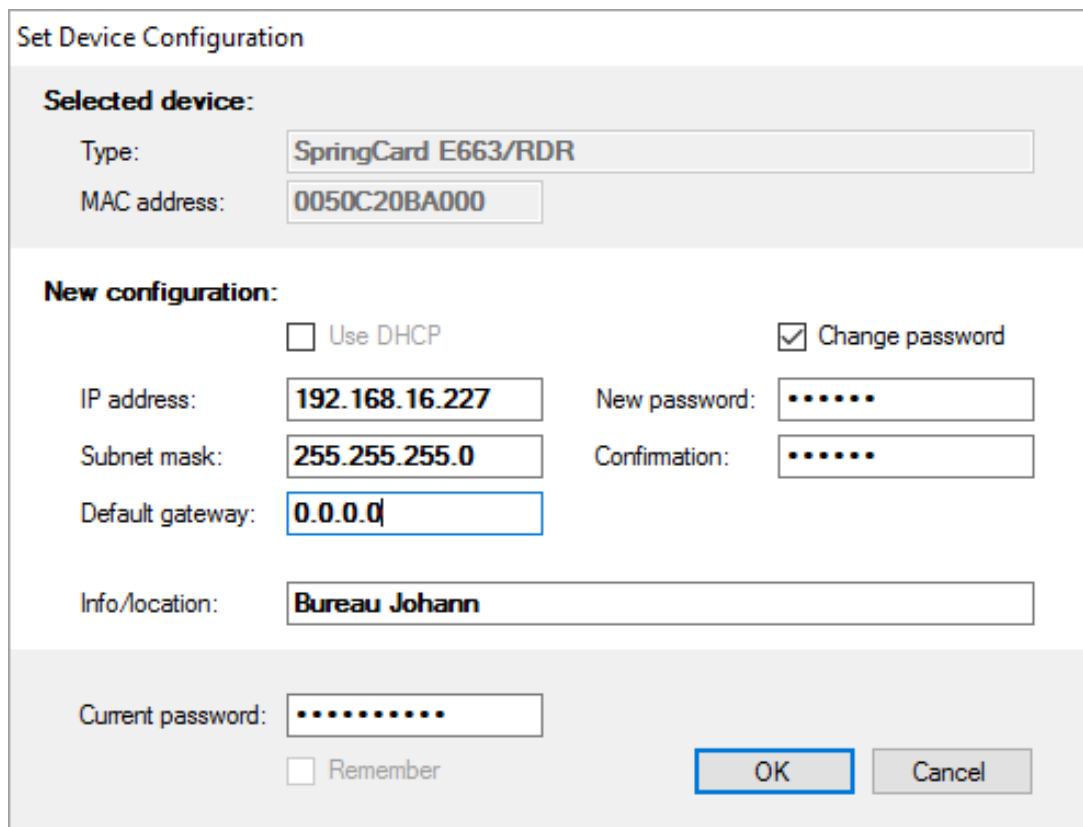
In the "info/location" field, enter a short string (less than 32 characters) as a reminder of the device's location or role.

d. Password

Check the box "change password" and enter a new password twice if you want to change the device's password.

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

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



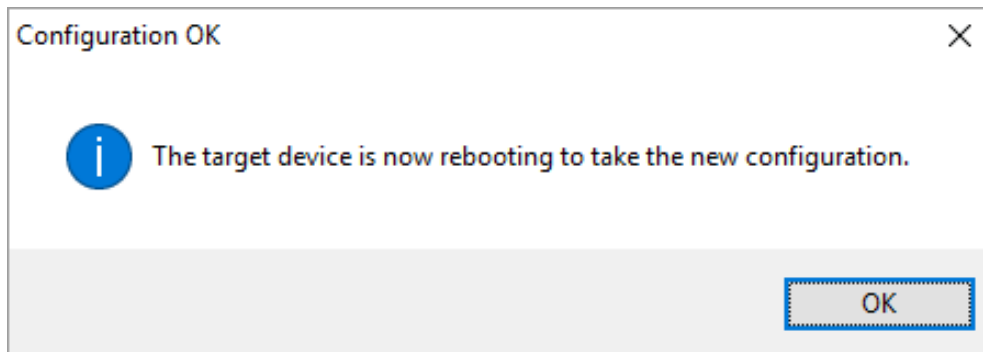
The image shows a 'Set Device Configuration' dialog box. It is divided into several sections. The 'Selected device:' section shows 'Type: SpringCard E663/RDR' and 'MAC address: 0050C20BA000'. The 'New configuration:' section contains a 'Use DHCP' checkbox (unchecked) and a 'Change password' checkbox (checked). Below these are fields for 'IP address: 192.168.16.227', 'Subnet mask: 255.255.255.0', 'Default gateway: 0.0.0.0', 'New password: [masked]', and 'Confirmation: [masked]'. There is also an 'Info/location:' field with the value 'Bureau Johann'. At the bottom, there is a 'Current password:' field with masked characters and a 'Remember' checkbox (unchecked). 'OK' and 'Cancel' buttons are at the bottom right.

Set Device Configuration	
Selected device:	
Type:	SpringCard E663/RDR
MAC address:	0050C20BA000
New configuration:	
☐ Use DHCP	☒ Change password
IP address:	192.168.16.227
Subnet mask:	255.255.255.0
Default gateway:	0.0.0.0
New password:	
Confirmation:	
Info/location:	Bureau Johann
Current password:	
☐ Remember	
OK Cancel	

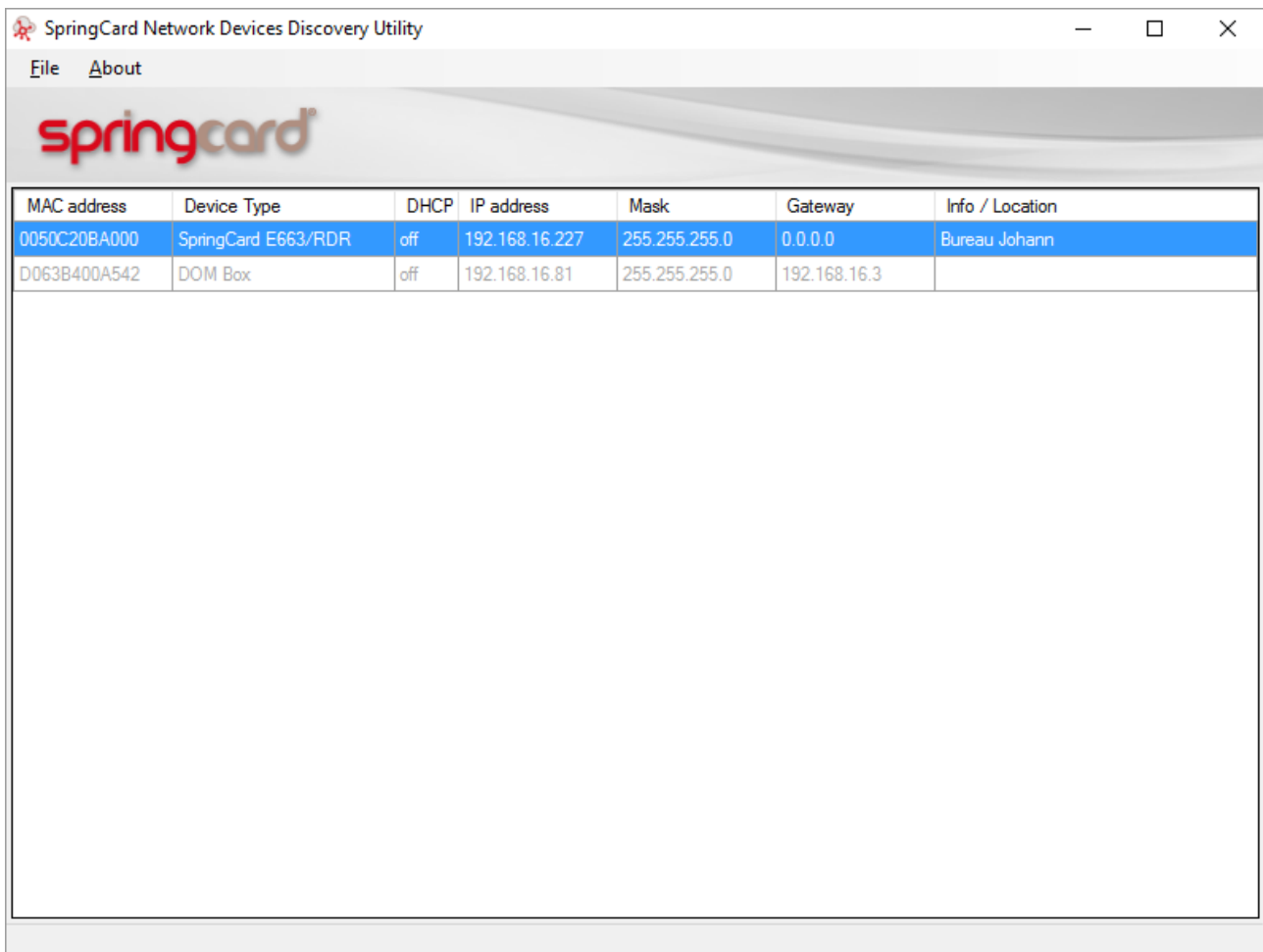
When ready, click “OK”.

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 TO A READER USING A MASTER CARD

SpringCard Readers in the **FunkyGate** family could be configured by the mean of a contactless Master Card.

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

Please refer to this software's documentation for details.

3. TELNET ACCESS TO THE DEVICE

3.1. THE DEVICE'S CONSOLE

The Device 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, § 10.2) may be used to disable the Console.

3.1.1. Open a Telnet session to the Device

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 Device's IP Address as defined in chapter 2.

*Windows Vista / 7 / 8 / 10 : 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.*

a. Device's connection prompt

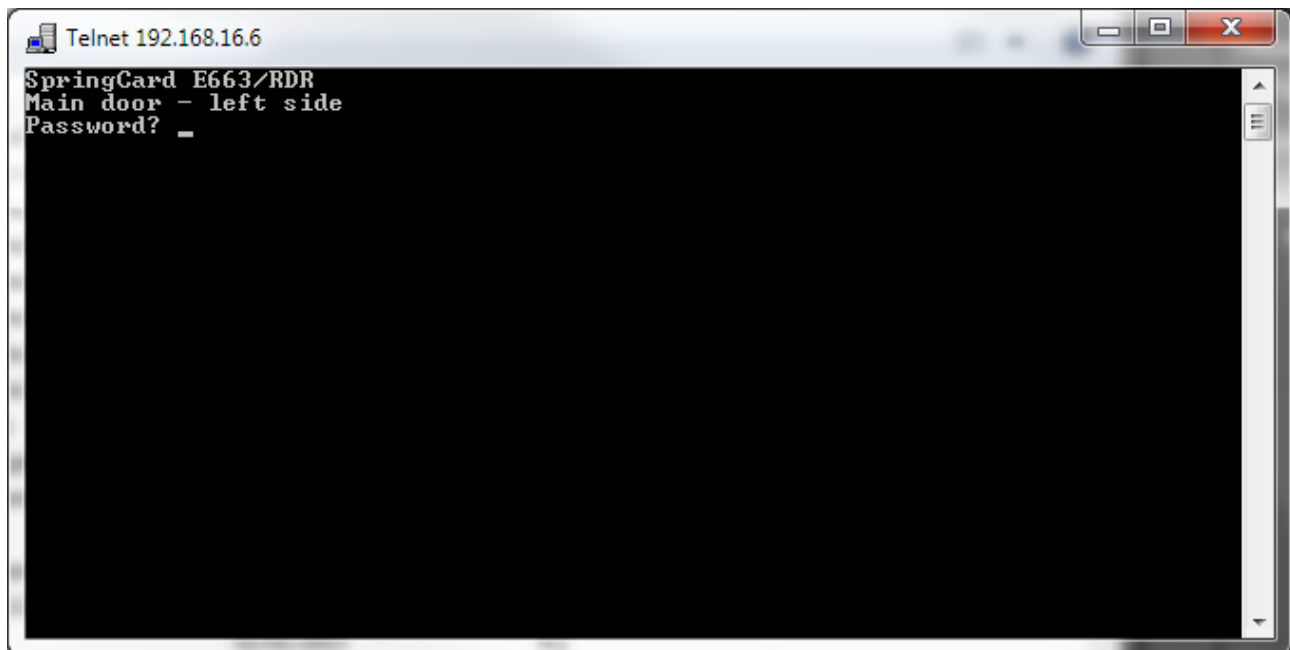
The Device sends its connection prompt. **FunkyGate** Readers say “SpringCard E663/RDR”, **HandyDrummer** Modules say “SpringCard E663/MIO”.

The second line shows the Info / Location string that has been entered in chapter 2 (if some).

On the third line the Device prompts for a password.

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

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



3.1.2. Sending a command to the Device

Write the command line as documented below, and terminate by hitting the ENTER key.
Note that the Device 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 $_hYY...YY$ to Configuration Register $_hXX$
cfgXX=!!	Erase Configuration Register $_hXX$
cfgXX	Read Configuration Register $_hXX$
exit	Terminate the Telnet session

4. EDITING DEVICE'S CONFIGURATION

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

- The Registers that control the IP configuration and the operation on the network are fully documented in this document,
- The Registers that are specific to the Device family (either **SpringCard FunkyGate** Reader or **SpringCard HandyDrummer** I/O Module) are documented only in the Integration and Configuration Guide of the family.

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

1. Through the Telnet link
2. Using Master Cards (only available if the Device is a Reader)
3. Using the SpringCard "MK2" Network Protocol, after authentication with the Administration Key.

Note that the SEC Configuration Register ($_{h6E}$, § 10.2) may be used to disable either way to access the Configuration Registers.

Administration Key is enabled in the IPS Configuration Register ($_{h83}$, § 10.3.5) and defined in the IPK.ADM Configuration Register ($_{h85}$, § 10.3.7).

4.1. THROUGH THE TELNET LINK

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

4.1.1. Reading Configuration Registers

Enter "cfg" to list all Configuration registers currently defined (registers that are not explicitly defined keep their default value).

Enter "cfgXX" to read the value of the Configuration register $_{hXX}$.

Note that Configuration registers $_{h55}$, $_{h56}$, $_{h6E}$ and $_{h6F}$ that hold sensitive data (the keys used by Master Cards and the Device's secret keys and password) are masked.

4.1.2. Writing Configuration Registers

Enter “cfgXX=YYYY” to update Configuration Register $_{h}XX$ with value $_{h}YYYY$. YYYY can be any length between 1 and 32 bytes.

Enter “cfgXX=!!” to erase Configuration Register $_{h}XX$.

4.2. USING MASTER CARDS (ONLY AVAILABLE ON A READER)

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

Please refer to this software's documentation for details.

4.3. THROUGH THE SPRINGCARD “MK2” NETWORK PROTOCOL

Please refer to § 9.5.

5. SPRINGCARD “MK2” NETWORK PROTOCOL – OVERVIEW

SpringCard “MK2” Network Protocol is a light-weight, bandwidth-efficient network protocol.

The MK2 Protocol supports either TCP or UDP for communication.

5.1. OPERATION OVER UDP

5.1.1. Standard operation (Unicast)

Both the Device and the Host are a UDP server.

- The Host listens on UDP port 3997;
- The Device listens on UDP port 3998.

These default port numbers could be overwritten using the IPU and IPH configuration registers ($_{h82}$, § 10.3.3 and $_{h83}$, § 10.3.4).

The UDP communication use only the **UDP plain mode** described in chapter 6. Secure communication is not possible over UDP. Only I-Blocks (§ 7.1.4.a) are possible over UDP and convey the **Application Level Datagrams** defined in chapter 9.

The communication may be unidirectional, in the Device to Host direction only, if the Host does not need to control the Device. Only event notifications are sent in this case to the Host’s UDP server port.

If the Host needs to control the Device, the communication is bidirectional; the Host sends its commands to the Device’s UDP server port, and receives the responses over its own UDP server port.

Care must be taken when developing the Host-side application(s) that

1. The protocol is in fully asynchronous. Event notifications and responses arrives in no specific order;
2. Using UDP, there is no 1-to-1 channel between the Host and a Device. All Devices on the network are likely to send their event notification and responses to the very same Host’s UDP server port;
3. UDP does not provide any guarantee of delivery. Packets may be lost. In the case of a command/response, the Host must be ready to repeat its command if it does not see a response after a timeout. **In the case of an event notification (tag read, input change etc), the information is lost, period.**

Applications that expects reliability or security shall use the TCP protocol, not UDP.

5.1.2. Operation using UDP broadcasts

To ease the deployment of many Devices on a closed-loop local area network (no routing), the Device may be operated using UDP broadcasts instead of unicast frames.

The overall behaviour is the same as in § 5.1.1, with only the following differences:

- All Devices may have the same IP address, since the IP address is never used for communication. Anyway, the Devices shall be configured with a static IP address if there is no DHCP server on the LAN (otherwise, they will wait numerous minutes for the DHCP server before starting with the fall-back address);
- The communication in the Host to Devices direction uses either the general broadcast address (255.255.255.255) or the broadcast address of the LAN (typically x.x.x.255);
- The communication in the Devices to Host direction may user either the same broadcast address or an explicit address for the Host. If the 2 UDP ports (Host side / Device side) are set to the same value, only the second option is allowed¹;

¹ If the Host and the Device are listening on the same UDP port and the Devices to Host communication uses a broadcast address, all the Devices will be receiving and processing all messages, which induces a strong CPU load in all Devices. This is not efficient in terms of energy, and likely to slow down all the operations of the Devices.

5.2. OPERATION OVER TCP

The Device is a TCP Server, and the Host (access control unit or computer in charge) is the Client.

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

The Device listens on TCP port 3999. This default port number could be overwritten using the IPP configuration register ($_{h81}$, § 10.3.2).

The Host initiates the communication by connecting to this TCP port and is the master of the communication. If the host closes the channel, the Device enters the reader-off state.

The TCP communication may use either the **TCP plain mode** described in chapter 7 or the **TCP secure mode** described in chapter 8.

To enforce security, the plain communication mode may be disabled by setting bit 0 of byte 0 to 1 in the IPS configuration register ($_{h82}$, § 10.3.5). In this case, the Device will reject any Host that doesn't switch to the secure communication mode immediately after having connected.

Whatever the communication mode, after the initial session-establishment, the Host (Client) and the Device (Server) exchange only I-Blocks (§ 7.1.4.a) or I_s -Blocks (§ 8.4.4.c). Both block types indistinctly convey the **Application Level Datagrams** defined in chapter 9.

Although the protocol may appear at first sight as a command/response protocol, the activity of the Device may insert event notifications (tag read, input state change) anywhere before a response, or between two responses that were expected for the same command.

Care must be taken when developing the Host-side application(s) that **the protocol is in fact fully asynchronous**.

5.3. KNOWN LIMITATIONS OF THE DEVICE AND PRECAUTIONS TO BE TAKEN ON THE HOST SIDE

The Device runs a lightweight network stack, and due to strong constraints over both the hardware and the memory footprint and execution speed of the embedded software, does not behave exactly as the network card of a computer would.

The table below summarizes the specificities of the Device's network stack, and suggests the appropriate behaviour for the Host-side application(s).

Limitation/Special behaviour	Impact / Countermeasure
MTU limited to 1500B (no jumbo frames)	None – the upper layer protocols are in fact limited to 256B
The Device sends all Ethernet frames twice (legacy of μ P stack)	TCP: no impact – the duplicated frames will be ignored by the host's TCP/IP stack UDP: every message will be received twice. The application shall process the message and track the sequence number to eliminate duplicates A network packet spy (such as WireShark) will see the duplicates. This is not an issue.
The Device is only able to process one command at once	The Host shall wait for the response to the current command before sending the next command

6. SPRINGCARD “MK2” NETWORK PROTOCOL – UDP MODE

This chapter describes the protocol’s **Plain Transport Layer for UDP communications**. There is no session-establishment and no possibility to switch to secure communication.

6.1. PRESENTATION LAYER

6.1.1. Block format

Every block transmitted in the channel is formatted as follow:

LENGTH	TYPE	MAC	SEQUENCE	PAYLOAD	CRC
1 byte	1 byte	6 bytes	8 bytes	Variable length	4 bytes
<i>mandatory</i>	<i>mandatory</i>	<i>optional</i>	<i>optional</i>	<i>may be empty</i>	<i>optional</i>

6.1.2. Description of the fields

Field	Description
LENGTH	The LENGTH byte is the total length of the block, this byte included.
TYPE	The TYPE byte is used to convey the information required to control the data transmission. There is only one type of block: I-block: convey information for use by the upper layers (Application Level Datagrams, see chapter 9).
MAC	The 6-byte MAC Address of the target device, when the host is sending, or of the source device, when the device is sending. Specifying the MAC Address in the block makes it possible to use UDP communication over dynamic or shared IP addresses (broadcast mode). If only fixed addresses and unicast modes are used, the field can be suppressed from all the blocks.
SEQUENCE	A 8-byte unique sequence number, defined by the sender. This allows the receiver to discard duplicated blocks, and to detect lost blocks. This field can be suppressed if duplicated blocks are acceptable for the application.
PAYLOAD	The PAYLOAD field is optional. When present, the PAYLOAD field conveys application data.
CRC	The CRC32 of all the preceding bytes.

6.1.3. Size of the blocks

The size of the PAYLOAD is between 0 and 64 bytes.

This leads to a block size up to 84 bytes.

6.1.4. Format of the TYPE byte

a. I-Block

Bit	Description
7 (msb)	Direction 0 for Host → Device 1 for Device → Host
6	Shall be set to 0
5	Shall be set to 0
4	Chaining 0: no chaining – this block is the only one, or the last one in a sequence 1: chaining enabled – more block(s) to come
3	Shall be set to 0
2	1 if the CRC field is present, 0 otherwise
1	1 if the SEQUENCE field is present, 0 otherwise
0 (lsb)	1 if the MAC field is present, 0 otherwise

6.2. GENERAL COMMUNICATION FLOW

6.2.1. Nominal dialog

There is no UDP channel but 2 independent, unreliable links (Device to Host, Host to Device); both the Device and the Host may send at any time, and therefore must be ready to receive at any time.

The Host sends I-Blocks to transmit its commands or to query the Device.

The Device sends I-Block to transmit its event notifications or its responses.

The Devices sends at least one message every 10 seconds. If the Device has no information to send, the Keep Alive message is an empty I-Block.

The Host shall not send Keep Alive requests.

6.2.2. Timings

a. Command/response

The Host should allow a 3s-timeout between a command and its response before reporting an error.

b. Channel monitor – UDP

The Device sends a Keep Alive message every 10 seconds.

The Device does not expect anything from the Host.

6.2.3. Chaining

If the application data buffer is longer than the max size for the PAYLOAD field, the data shall be divided onto multiple I-Blocks. In this case, the Chaining bit is set to 1 for every I-Block but the last one.

Chaining is not implemented in the current version of the Device's firmware. The Host shall not use this feature (and the Device will not use it).

6.3. APPL LEVEL

Chapter 9 contains the Application Level Protocol. The application-level messages are conveyed within I-Blocks.

7. SPRINGCARD “MK2” NETWORK PROTOCOL – TCP PLAIN MODE

This chapter describes the protocol’s **Plain Transport Layer for TCP connections**.

This Transport Layer is designed to support the transmission of variable-length blocks.

The session starts with a session-establishment handshake, that makes it possible for both partners to check they are running the same protocol. The Host (Client) may also decide to switch to the optional Secure Transport Layer (chapter 8).

The Plain Transport Layer over UDP is described in chapter 6.

7.1. PRESENTATION LAYER

7.1.1. Block format

Every block transmitted in the channel is formatted as follow:

LENGTH	TYPE	PAYLOAD
1 byte	1 byte	Variable length

7.1.2. Description of the fields

Field	Description
LENGTH	The LENGTH byte is the total length of the block, this byte included.
TYPE	The TYPE byte is used to convey the information required to control the data transmission. There are two fundamental types of blocks: • I-block: convey information for use by the upper layers (Application Level Datagrams, see chapter 9). • H-block: convey handshaking and connection-control data
PAYLOAD	The PAYLOAD field is optional. When present, the PAYLOAD field conveys application data.

7.1.3. Size of the blocks

The size of every block must be less or equal to 66 bytes.

This lead to a PAYLOAD between 0 and 64 bytes.

If the application layer needs to transmit more than 64 bytes, chaining shall be used.

7.1.4. Format of the TYPE byte

a. I-Block

Bit	Description
7 (msb)	Direction 0 for Host → Device 1 for Device → Host
6	Shall be set to 0
5	Shall be set to 0
4	Chaining 0: no chaining – this block is the only one, or the last one in a sequence 1: chaining enabled – more block(s) to come
3	Shall be set to 0000
2	
1	
0 (lsb)	

b. H-Block

Bit	Description
7 (msb)	Direction 0 for Host → Device 1 for Device → Host
6	Shall be set to 1
5	Block type: - b00: HELO (Device's "hello" block) - b01: HELO-OK (Host's "hello" acknowledge) - b10: RFU, do not use - b11: HELO-AUTH (see § 8.3)
4	
3	Protocol Version for HELO block Key Number for the first HELO-AUTH block 0000 for the other blocks.
2	
1	
0 (lsb)	

c. Protocol Version

The Device sets this field to b0000. Any other value shall be interpreted by the Host as an error.

7.2. GENERAL COMMUNICATION FLOW

7.2.1. Session establishment

The Host tries to connect to one (or many) Device.

When a connection is established on the Device, the Device sends a HELO block. The payload of the HELO block is the Device's MAC address on 6 bytes.

HELO block (Device → Host)

LENGTH	TYPE	PAYLOAD
_h 08	_h C0	Device's MAC address on 6 bytes

The Host may check that the claimed MAC address is coherent with its records.

The Host may check that the Device's Protocol Version is acceptable.

If everything is OK, the Host sends a HELO-OK block. The payload of the HELO-OK block is empty.

HELO-OK block (Host → Device)

LENGTH	TYPE	PAYLOAD
_h 02	_h 50	empty

7.2.2. Nominal dialog

The TCP channel is full-duplex; both the Device and the Host may send at any time, and therefore must be ready to receive at any time.

The Host sends I-Blocks to transmit its commands or to query the Device.

The Device sends I-Block to transmit its event notifications or its responses.

The Host may use send an empty I-Block as a Keep Alive request; the Device answers with an I-Block that is empty when no other data is available, or populated if the Device has an event notification in its queue.

7.2.3. Timings

a. Command/response

The Host should allow a 3s-timeout between a command and its response.

This is also applicable to the HELO blocks and to the Keep Alive request/response scheme.

b. Channel monitor

The Device expects to receive a block from the Host at least every 60s, otherwise it assumes the channel has been broken and closes it on its side.

7.2.4. Chaining

If the application data buffer is longer than the max size for the PAYLOAD field, the data shall be divided onto multiple I-Blocks. In this case, the Chaining bit is set to 1 for every I-Block but the last one.

Chaining is not implemented in the current version of the Device's firmware. The Host shall not use this feature (and the Device will not use it).

7.3. ERROR HANDLING AND RECOVERY

7.3.1. For the Device

- **Bad sequence during session establishment:** is the Device receives a block before having transmitted its HELO, the Device drops the connection,
- **Protocol error:** if the Device receives an invalid block from the Host (LENGTH not coherent with actual length, or unallowed value for TYPE), the Device drops the connection,
- **No more activity error:** if the Host remains silent for 60s, the Device drops the connection.

7.3.2. For the Host

- **Bad sequence during session establishment:** is the first block received by Host is not a valid HELO, or the Host receives another block before having transmitted its HELO-OK, the Host shall drop the connection,
- **Protocol error:** if the Host receives an invalid block from a Device (LENGTH not coherent with actual length, or unallowed value for TYPE), the Host shall drop the connection,
- **Timeout error:** if the Device doesn't answer within 3s, the Host shall drop the connection.

7.3.3. Recovery

If the connection is dropped for any reason, the Host shall wait at least 5s before trying to connect again to the same Device.

7.4. APPL LEVEL

Chapter 9 contains the Application Level Protocol. The application-level messages are conveyed within I-Blocks.

8. SPRINGCARD “MK2” NETWORK PROTOCOL – TCP SECURE MODE

8.1. ABSTRACT

This chapter describes the protocol's **Secure Transport Layer**. In this mode,

- The Device and the Host perform a **3-pass mutual authentication** to prove each other that they share the very same authentication key (one of the 2 Device's secret key). In the same time, the 3-pass authentication establishes a one-time, random session key – that remains also a secret only shared by both peers,
- The blocks conveyed between the two partners are **ciphered and authenticated**, i.e. their content remains undisclosed, and a defrauder could not insert its own packets into the sequence without being noticed.

The Device has 2 secret keys. Both keys are enabled in the IPS Configuration Register ($_{h84}$, § 10.3.5) and their values are defined in the IPK.OPE ($_{h85}$, § 10.3.6) and IPK.ADM ($_{h86}$, § 10.3.7) Configuration Registers, respectively.

The Host choose either key when asking for authentication, depending on the actions it wants to perform onto the Device:

- The **Operation Key** gives access to the basic operations of the Device (§ 9.4). This is the key an access control unit would use to operate the Device,
- The **Administration Key** makes it possible to change the Device's configuration (§ 9.5). This key would typically be used by a configuration software, when installing the Device.

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

After the initial session-establishment, the Host (Client) and the Device (Server) exchange only I_S -Blocks (§ 8.4.4.c). **The I_S -Blocks convey the Application Level Datagrams defined in chapter 9.**

The Secure Transport Layer supports TCP only, and is not available over UDP.

8.2. CRYPTOGRAPHIC BACKGROUND

The Device uses the **AES block cipher** (Rijndael) . AES has a fixed 128-bit (16 bytes) block size. The Device supports **128-bit keys** (16 bytes) only.

In the following paragraphs,

- $E(K, P)$ means “AES encrypt operation (cipher) over block P using the key K ”,
- $D(K, C)$ means “AES decrypt operation (decipher) over block C using the key K ”.

Note that the size of blocks P and C must exactly 16 bytes.

When more than one block are involved, the encrypt and decrypt operations are performed in **CBC (cipher block chaining) mode**.

In the following paragraphs,

- $E_{CBC}(K, V, P)$ means “AES encrypt operation (cipher) over buffer P using the key K and the Init Vector V ”,
- $D_{CBC}(K, V, C)$ means “AES decrypt operation (decipher) over buffer C using the key K and the Init Vector V ”.

Note that the size of buffers P and C must be a multiple of 16 bytes. As a consequence, a padding is generally involved.

8.3. 3-PASS AUTHENTICATION

The 3-pass authentication is initiated by the Host after receiving the HELO block from the Device (§ 7.2.1) and follows the specification of the H-Block introduced in § 7.1.4.b.

8.3.1. Device's HELO

HELO block (Device → Host)

LENGTH	TYPE	PAYLOAD
$_{h}08$	$_{h}C0$	Device's MAC address on 6 bytes

The HELO block contains the Device's MAC address. This makes it possible for the Host

1. To check this Device is the expected one (table IP address $\leftrightarrow$ MAC address)
2. To select this Device's secret key.

8.3.2. Host's HELO-Auth

The Host asks the Device to open a secure session by sending an HELO-Auth block. The payload of the HELO-Auth block is empty. The low-order bit of the TYPE byte selects the key

HELO-Auth block (Host → Device) selecting Operation Key

LENGTH	TYPE	PAYLOAD
$_{h}02$	$_{h}71$	empty

HELO-Auth block (Host → Device) selecting Administration Key

LENGTH	TYPE	PAYLOAD
$_{h}02$	$_{h}72$	empty

8.3.3. Authentication, step 1

After receiving the HELO-Auth block from the Host,

- The Device activates the selected **secret key** K_{AUTH} ,
- The Device generate a random challenge (C_R) on 16 bytes,
- The Device sends to the Host a block containing $E (K_{AUTH}, C_R)$.

Authentication, step 1: block Device → Host

LENGTH	TYPE	PAYLOAD
$_{h}12$	$_{h}F0$	$E (K_{AUTH}, C_R)$ on 16 bytes

*The Init Vector of the AES cipher is cleared to (00..00) before computing $E (K_{AUTH}, C_R)$.
1 block is crypted, no padding is applied.*

8.3.4. Authentication, step 2

- The Host activates the secret key K_{AUTH} ,
- The Host decrypts the payload received from the Device, and retrieves C_R ,
- The Host computes $C_R' = C_R \ll 1 \parallel C_R \gg 127$ (shift left with carry),
- The Host generate a random challenge (C_H) on 16 bytes,
- The Host sends to the Device a block containing $E (K_{AUTH}, C_H \parallel C_R')$,

Authentication, step 2: block Host → Device

LENGTH	TYPE	PAYLOAD
$_{h}22$	$_{h}70$	$E (K_{AUTH}, C_H \parallel C_R')$ on 32 bytes

*The Init Vector of the AES cipher is cleared to (00..00) before computing $E (K_{AUTH}, C_H \parallel C_R')$.
2 block are crypted in CBC mode, no padding is applied.*

8.3.5. Authentication, step 3

- The Device decrypts the payload received from the Host, and retrieves C_H and C_R' ,
- The Device checks that C_R' is valid. This is the proof that the Host knows the secret key,
- The Device computes $C_H' = C_H \ll 1 \parallel C_H \gg 127$ (shift left with carry),
- The Device sends to the Host a block containing $E (K_s, C_H')$,

Authentication, step 3: block Device → Host

LENGTH	TYPE	PAYLOAD
$_{h}12$	$_{h}F0$	$E (K_{AUTH}, C_H')$ on 16 bytes

*The Init Vector of the AES cipher is cleared to (00..00) before computing $E (K_{AUTH}, C_H')$.
1 block is crypted, no padding is applied.*

8.3.6. Conclusion of the Authentication sequence – Host's HELO-OK

- The Host decipheres the payload received from the Device, and retrieves C_H' ,
- The Host checks that C_H' is valid. This is the proof that the Device knows the secret key,
- The Host generates a random nounce (N_H) on 16 bytes
- The Host sends to the Device a HELO-OK block containing $E (K_{SESS}, N_H)$

HELO-OK block (Host → Device)

LENGTH	TYPE	PAYLOAD
$_{h22}$	$_{h50}$	$E (K_{SESS}, N_H CMAC PADD)$ on 32 bytes

The CMAC is computed as specified in 8.6.1 . The sequence number of the CMAC is cleared to 0 before computing the CMAC. The initial size of the payload (before CMAC and Padding) is $_{h10}$ (size of N_H).

After CMAC, the size of the payload is $_{h18}$ (size of CMAC is 8 bytes).

The Padding is applied as specified in 8.6.2.b to reach 32 bytes of payload. The final Length of the packet is therefore $_{h22}$ (2-byte header + 32-byte payload).

The Init Vector of the AES cipher is cleared to (00..00) before computing $E (K_{SESS}, ...)$. 2 blocks are crypted in CBC mode as specified in 8.6.2.c.

8.4. PRESENTATION LAYER AFTER AUTHENTICATION

8.4.1. Block format

Every block transmitted in the channel is formatted as follow:

LENGTH	TYPE	CIPHERED PAYLOAD		
		PAYLOAD	CMAC	PADDING
1 byte	1 byte	Variable length	8 bytes	Variable length

8.4.2. Description of the fields

Field	Description
LENGTH	The LENGTH byte is the total length of the block, this byte included.
TYPE	The TYPE byte is used to convey the information required to control the data transmission. After authentication, only I _S -Blocks could be transmitted
PAYLOAD	The PAYLOAD field is optional. When present, the PAYLOAD field conveys application data.
CMAC	The CMAC field is computed over the initial PAYLOAD field and the TYPE and SEQUENCE fields, as specified in 8.6.1 .
PADDING	The cipher algorithm uses fixed-size blocks. Therefore a PADDING shall be applied to ensure that the size of content to be ciphered is a multiple of the cipher's block size. The PADDING is specified in 8.6.2 .
CIPHERED PAYLOAD	After addition of the CMAC and PADDING field, the whole PAYLOAD is ciphered (encrypted) as specified in 8.6.2 .

8.4.3. Size of the blocks

If the application layer needs to transmit more than 64 bytes, chaining shall be used.

With a PAYLOAD between 0 and 64 bytes, the total size of every block is between 18 and 82.

8.4.4. Format of the TYPE byte

a. I-Block

Same as § 7.1.4.a.

b. H-Block

Same as § 7.1.4.b.

c. *I_S*-Block

Bit	Description
7 (msb)	Direction 0 for Host → Device 1 for Device → Host
6	Shall be set to 0
5	Shall be set to 1
4	Chaining 0: no chaining – this block is the only one, or the last one in a sequence 1: chaining enabled – more block(s) to come
3	Shall be set to 0000
2	
1	
0 (lsb)	

8.5. SESSION KEYS

In order to secure the communication, two session keys are derived from the two random challenges exchanged during the authentication:

- **K_{SESS}** is the session **Encryption Key**, used to ensure confidentiality over the TCP channel,
- **K_{CMAC}** is the session **CMAC Key**, used to ensure confidence over the TCP channel.

8.5.1. Session Encryption Key

Let C_R be the Device's random challenge (§ 8.3.3). C_R is a 16-byte value ($C_R[0] \dots C_R[15]$).

Let C_H be the Host's random challenge (§ 8.3.4). C_H is a 16-byte value ($C_H[0] \dots C_H[15]$).

Let K_{AUTH} be the key used for authentication.

Construct T , a 16-byte buffer, as follow:

- | | |
|--------------------|--------------------|
| ■ $T[0] = C_H[11]$ | ■ $T[5] = C_R[11]$ |
| ■ $T[1] = C_H[12]$ | ■ $T[6] = C_R[12]$ |
| ■ $T[2] = C_H[13]$ | ■ $T[7] = C_R[13]$ |
| ■ $T[3] = C_H[14]$ | ■ $T[8] = C_R[14]$ |
| ■ $T[4] = C_H[15]$ | ■ $T[9] = C_R[15]$ |

- $T[10] = C_H[4] \oplus C_R[4]$
- $T[11] = C_H[5] \oplus C_R[5]$
- $T[12] = C_H[6] \oplus C_R[6]$
- $T[13] = C_H[7] \oplus C_R[7]$
- $T[14] = C_H[8] \oplus C_R[8]$
- $T[15] = h11$

Compute $K_{\text{SESS}} = E (K_{\text{AUTH}}, T)$.

8.5.2. Session CMAC Key

Let C_R be the Device's random challenge (§ 8.3.3). C_R is a 16-byte value ($C_R[0] \dots C_R[15]$).

Let C_H be the Host's random challenge (§ 8.3.4). C_H is a 16-byte value ($C_H[0] \dots C_H[15]$).

Let K_{AUTH} be the key used for authentication.

Construct T , a 16-byte buffer, as follow:

- $T[0] = C_H[7]$
- $T[1] = C_H[8]$
- $T[2] = C_H[9]$
- $T[3] = C_H[10]$
- $T[4] = C_H[11]$
- $T[5] = C_R[7]$
- $T[6] = C_R[8]$
- $T[7] = C_R[9]$
- $T[8] = C_R[10]$
- $T[9] = C_R[11]$
- $T[10] = C_H[0] \oplus C_R[0]$
- $T[11] = C_H[1] \oplus C_R[1]$
- $T[12] = C_H[2] \oplus C_R[2]$
- $T[13] = C_H[3] \oplus C_R[3]$
- $T[14] = C_H[4] \oplus C_R[4]$
- $T[15] = h22$

Compute $K_{\text{CMAC}} = E (K_{\text{AUTH}}, T)$.

8.6. SECURE COMMUNICATION CHANNEL

After the authentication, the communication is secured by the combination of:

- A **8-byte CMAC** computed over the plain-text using K_{CMAC} ,
- The **AES-CBC encryption** of the plain-text plus the CMAC using K_{SESS} ,
- The synchronisation of a **Sequence number** and the synchronisation of the **Init Vectors (IV)** between sender and receiver, to prevent any kind of injection.

8.6.1. CMAC

a. Sequence numbers

The calculation of the CMAC includes a Sequence number, to protect against the injection or the removal of blocks. Both the Device and the Host shall maintain 2 Sequence numbers for the CMAC:

- **SEQ_H is used by the Host to compute its outgoing CMAC**, and by the Device to verify the its incoming CMAC. SEQ_H is incremented every time the Hosts sends a block.
- **SEQ_R is used by the Device to compute its outgoing CMAC**, and by the Host to verify the its incoming CMAC. SEQ_R is incremented every time the Hosts sends a block.

Both SEQ_H and SEQ_R are cleared at the end of the authentication (§ 8.3.6) and evolve independently afterwards.

b. Computing the CMAC

Let P be the (plain) payload of the packet. P is an arbitrary-length buffer.

Construct H, an 8-byte buffer, as follow:

- H[0] to H[3] = SEQ_H or SEQ_R (sequence number of the sender), expressed in MSB-first format
- H[4] = TYPE of the packet (see § 8.4.2)
- H[5] = size of P (see § 8.4.2)
- H[6] = $\text{hFF} \oplus \text{TYPE}$
- H[7] = $\text{hFF} \oplus \text{LENGTH}$

Construct T = H || P

If size of T is not a multiple of 16 bytes, padd T as follow:

- $T = T || \text{h80}$
- While size of T is not a multiple of 16 bytes, $T = T || \text{h00}$

Compute $C = E_{\text{CBC}} (K_{\text{CMAC}}, \text{h00} .. \text{h00}, T)$

(T encrypted in CBC mode, using K_{CMAC} and an all-zero Init Vector)

Keep C_{LAST}, the last 16 bytes of C.

Extract CMAC, a 8-byte buffer, as follow:

- | | |
|---|--|
| ■ $\text{CMAC}[0] = C_{\text{LAST}}[0]$ | ■ $\text{CMAC}[4] = C_{\text{LAST}}[8]$ |
| ■ $\text{CMAC}[1] = C_{\text{LAST}}[2]$ | ■ $\text{CMAC}[5] = C_{\text{LAST}}[10]$ |
| ■ $\text{CMAC}[2] = C_{\text{LAST}}[4]$ | ■ $\text{CMAC}[6] = C_{\text{LAST}}[12]$ |
| ■ $\text{CMAC}[3] = C_{\text{LAST}}[6]$ | ■ $\text{CMAC}[7] = C_{\text{LAST}}[14]$ |

c. *New payload*

The AES-CBC encryption will now be applied over $P' = P \parallel \text{CMAC}$

8.6.2. AES-CBC encryption

a. *Init Vectors*

All operation are performed in CBC mode. The Init Vector is preserved among all operations on both sides. Both the Device and the Host shall maintain 2 Init Vectors for the encryption/decryption:

- **IV_H is used by the Host to send** (encrypt), and by the Device to receive (decrypt),
- **IV_R is used by the Device to send** (encrypt), and by the Device to receive (decrypt).

When receiving the HELO-OK block (§ 8.3.6), the Device decrypts the received cryptogram after starting from a clear IV (00..00). As a consequence, the Device's IV_H becomes synchronised with the Host's IV_H. At this step, both **the Device and the Host copy IV_H into IV_R**.

Afterwards, IV_H into IV_R will evolve independently.

b. *Padding*

The AES-CBC encryption could be performed only if the size of the text is a multiple of 16 bytes. A padding is always applied.

Let P' be the packet's payload ($P' = P \parallel \text{CMAC}$ as per § 8.6.1.c).

Compute $p = 16 - (\text{sizeof}(P') \bmod 16)$

(p is 16 minus the reminder of the size of P' in the division by 16)

If $p = 0$, then set $p = 16$.

Construct T, a p-byte buffer, whose every byte value is b:

- $T[0] = p$
- $T[1] = p$
- ...

- $T[p-1] = p$

Set $P'' = P' || T$

Size of P'' is now a multiple of 16 bytes.

c. Encryption

Compute $C = E_{CBC} (K_{SESS}, IV_H \text{ or } IV_R, P'')$

(P'' encrypted in CBC mode, using K_{SESS} and using either the Init Vector of the sender)

The final IS-Block packet (§ 8.4.4.c) is then

- $LENGTH = 2 + \text{sizeof} (C)$
- $TYPE = I_S\text{-Block}$
- Actual payload = C

8.6.3. Receiving

When receiving a I_S -Block packet, the receiver must follow the reverse path:

1. Check that LENGTH and TYPE are valid
2. Check that the size of the packet's content (C) is multiple of 16
3. Retrieve $P'' = D_{CBC} (K_{SESS}, IV, C)$
(remember to use the Init Vector of the sender)
4. Check the padding, suppress the padding to recover P' from P''
5. Extract P and CMAC from P' , verify the CMAC using K_{CMAC} and the sequence number of the sender

8.7. NEW AUTHENTICATION — GENERATION OF A NEW SESSION KEY

The Host may require a new authentication at any time, by sending a new HELO-Auth block as specified in § 8.3.2 .

8.8. GENERAL COMMUNICATION FLOW

8.8.1. Nominal dialog

The TCP channel is full-duplex; both the Device and the Host may send at any time, and therefore must be ready to receive at any time.

The Host sends I_S -Blocks to transmit its commands or to query the Device. An empty I_S -Block denotes a Keep Alive request.

The Device sends I-Block to transmit its notifications or its answers. An empty I_S -Block denotes a Keep Alive response (when no other data is available).

8.8.2. Timings

The Device ensures that it answer to every block coming from the Host by a response block within 2.5s. The Host may use a 3s-timeout to watch-out the Device. This is also applicable to the HELO block that is sent by the Device immediately when the connection is opened.

The Device expects to receive a block from the Host at least every 60s.

8.8.3. Chaining

If the application data buffer is longer than the max size for the PAYLOAD field, the data shall be divided onto multiple I_S-Blocks.

In this case,

- The Chaining bit is set to 1 for every I_S-Block but the last one,
- Only the first I_S-Block contains the SEQUENCE field,
- Only the last I_S-Block contains the CMAC and PADDING fields.

8.9. ERROR HANDLING AND RECOVERY

8.9.1. For the Device

- **Bad sequence during session establishment:** is the Device receives a block before having transmitted its HELO, the Device drops the connection,
- **Protocol error:** if the Device receives an invalid block from the Host (LENGTH not coherent with actual length, or unallowed value for TYPE), the Device drops the connection,
- **No more activity error:** if the Host remains silent for 60s, the Device drops the connection.

8.9.2. For the Host

- **Bad sequence during session establishment:** is the first block received by Host is not a valid HELO, or the Host receives another block before having transmitted its HELO-OK, the Host shall drop the connection,
- **Protocol error:** if the Host receives an invalid block from a Device (LENGTH not coherent with actual length, or unallowed value for TYPE), the Host shall drop the connection,
- **Timeout error:** if the Device doesn't answer within 3s, the Host shall drop the connection.

8.9.3. Recovery

If the connection is dropped for any reason, the Host shall wait at least 5s before trying to connect again to the same Device.

8.10. APPLICATION LAYER

Chapter 9 contains the application layer protocol. The application-level messages are conveyed within I_S-Blocks.

9. SPRINGCARD “MK2” NETWORK PROTOCOL – APPLICATION LAYER

9.1. PRINCIPLES

This chapter describes the **Application Layer**, whose Application-Level Datagrams are transmitted in either **Plain** Transport Datagrams over TCP or UDP (chapter 5) or **Secure** Transport Datagrams over TCP (chapter 8).

*The **SpringCard MK2 Protocol** is not a Request/Response Protocol; both the Host and the Device 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 Device at any time.*

9.2. FORMAT OF THE APPLICATION LEVEL DATAGRAM UNITS

The **Application Level Datagrams** are obeys to a T,L,V scheme:

- **T (Tag):** this is the operation-code of a command, or the identifier of a data field. The Tag is on either 1 or 2 bytes,
- **L (Length):** this is the length of the following Value, on 1 byte. Allowed values are $_{h}00$ to $_{h}7F$,
- **V (Value):** the parameters to the command, or the data field itself. The length is specified by L, from 0 to 127 bytes.

9.3. LIST OF COMMANDS AND DATA-FIELD IDENTIFIERS

9.3.1. Command codes (Host → Device)

T (Tag)	Meaning	See §
h00	Get Global Status	9.4.1
h01	Get Device Name	9.4.2
h02	Get Device Capabilities	9.4.3
h03	Get Device Serial Number	9.4.4
h04	Read Inputs	9.4.5
h0A	Start / Stop Reader	9.4.6
h90xx	Set Output	9.4.7
hA0xx	Clear Output	9.4.8
hD000	Clear LEDs	9.4.9
	Set LEDs	9.4.10
	Start LEDs	9.4.11
hD100	Buzzer	9.4.12
Restricted operations		
(available only in a secure session, after authentication with the Administration Key)		
h0C	Write Configuration	9.5.1
	Erase Configuration	9.5.2
	Reset the Device (to apply the Configuration)	9.5.3

9.3.2. Event codes and response codes (Device → Host)

T (Tag)	Meaning	Response Event	See §
h01	Device Name		9.6.1
h02	Device Capabilities		9.6.2
h03	Device Serial Number		9.6.3
h0A	Reader sequence terminated		9.6.9
h2F	Tamper Status		9.6.5
h8100	Reader Name		9.6.4
hB000	Card Read		9.6.6
hB100	Card Inserted		9.6.7
	Card Removed		9.6.8
hC0xx	Input Changed		9.6.10

9.4. HOST → DEVICE, 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.

9.4.1. Get Global Status

T	L
h00	h00

The Device answers by a sequence of messages:

1. **Reading Head Identifier** (§ 9.6.4) if the Device is a Reader (all RDR),
2. **Tamper Status** (§ 9.6.5) if the Device has Tamperers (FunkyGate only).

9.4.2. Get Device Name

T	L
h01	h00

The Device answers by sending the **Device Name** message (§ 9.6.1).

9.4.3. Get Device Capabilities

T	L
h02	h00

The Device answers by sending the **Device Capabilities** message (§ 9.6.2).

9.4.4. Get Device Serial Number

T	L
h03	h00

The Device answers by sending the **Device Serial Number** message (§ 9.6.3).

9.4.5. Read Inputs (*MIO only*)

T	L
h0C	h00

The MIO answers by sending one **Input Changed** message (§ 9.6.10) for every input line it has.

9.4.6. Start/Stop Reader (*RDR only*)

a. Permanent mode

T	L	V
h0A	h01	mode

- **mode:** start/stop command
 - **h00** Reader goes OFF (RF field OFF, no activity on RF)
 - **h01** Reader goes ON

b. Single-shot mode

T	L	V
h0A	h02	h01 time

- the 1st byte (mode) is set to **h01** (start),
- the 2nd byte (time) defines the duration of the scan, in tenth of second (0.1 to 2.55s). Set to **h00** for a single iteration of the polling loop.

When single-shot mode is used and no card is found in the specified interval, the Device sends a **Reader sequence terminated** message (§ 9.6.9) upon completion.

9.4.7. Set Output command (*MIO only*)

a. Permanent

The MIO asserts the Output until the **Clear Output** (§ 9.4.8) command is received.

T	L
_h 90xx	_h 00

The 'xx' part in the Tag is the number of the Output.

b. Temporary

The MIO asserts the Output for the specified time (in seconds). If the time is 0s, the Output is asserted for about 100ms.

T	L	V
_h 90xx	_h 02	Time-out (s)

The 'xx' part in the Tag is the number of the Output.

9.4.8. Clear Output command (*MIO only*)

The MIO de-assert the specified Output.

T	L
_h A0xx	_h 00

The 'xx' part in the Tag is the number of the Output.

9.4.9. Clear LEDs command (*RDR only*)

Both LEDs go OFF.

T	L
_h D000	_h 00

9.4.10. Set LEDs command (*RDR only*)

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
 - _h00 OFF
 - _h01 ON
 - _h02 blinks slowly
 - _h03 blinks quickly
- **green:** command for green LED
 - _h00 OFF
 - _h01 ON
 - _h02 blinks slowly
 - _h03 blinks quickly

9.4.11. Start LED sequence command (*RDR only*)

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

T	L	V		
<code>hD000</code>	<code>h04</code>	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.

9.4.12. Buzzer command (*RDR only*)

T	L	V
<code>hD100</code>	<code>h01</code>	seq.

- **seq:**
 - `h00` buzzer OFF,
 - `h01` buzzer ON,
 - `h02` buzzer short sequence,
 - `h03` buzzer long sequence.

9.5. HOST → DEVICE, RESTRICTED OPERATIONS

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

9.5.1. Write Configuration Register

The Device's behaviour is defined by Configuration Registers. 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>

9.5.2. Erase Configuration Register

The Device's behaviour is defined by Configuration Registers. 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>

9.5.3. Reset the Device

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

T	L
$_{h}0C$	$_{h}00$

9.6. DEVICE → HOST

9.6.1. Device Name

This T,L,V is transmitted in response to the **Get Device Name** command (§ 9.4.2).

a. For a RDR

T	L	V
h01	h1C	SpringCard E663/RDR x.xx

b. For a MIO

T	L	V
h01	h1C	SpringCard E663/MIO x.xx

9.6.2. Device Capabilities

This T,L,V is transmitted in response to the **Get Device Capabilities** command (§ 9.4.3).

T	L	V			
h02	h03	<table> <tr> <td>Number of Reading Heads</td><td>Number of Inputs</td><td>Number of Outputs</td></tr> </table>	Number of Reading Heads	Number of Inputs	Number of Outputs
Number of Reading Heads	Number of Inputs	Number of Outputs			

A RDR would return V = h01, h00, h00.

A MIO with 8 input lines and 8 output lines would return V = h00, h08, h08.

9.6.3. Device Serial Number

This T,L,V is transmitted in response to the **Get Device Serial Number** command (§ 9.4.4).

T	L	V
h03	h06	Serial number (MAC address)

9.6.4. Reading Head Identifier

This T,L,V is transmitted in response to the **Get Global Status** command, for every Reading Head that is available on the Device.

(Basically, there's only one Reading Head on a RDR device, and none on a MIO device).

a. For a RFID/NFC Reading Head

T	L	V
<code>h8100</code>	<code>h1C</code>	SpringCard E663/RDR x.xx

b. For a RFID/NFC Reading Head with a pinpad

T	L	V
<code>h8100</code>	<code>h1C</code>	SpringCard E663/RDR+PIN x.xx

c. For a RFID/NFC Reading Head with a BLE interface

T	L	V
<code>h8100</code>	<code>h1C</code>	SpringCard E663/RDR+BLE x.xx

9.6.5. 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
<code>h2F</code>	<code>h01</code>	Bit field, the broken tampers are denoted by the corresponding bit set to 1. V = <code>h00</code> when all tampers are OK.

9.6.6. Card Read (RDR only)

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

9.6.7. Card Inserted (RDR only)

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

9.6.8. Card Removed (RDR only)

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

T	L
_h B100	_h 00

9.6.9. Reader sequence terminated

This T,L,V is transmitted upon completion of a single-shot read sequence (§ 9.4.6.b) when no Card has been read.

T	L
_h 0A	_h 00

9.6.10. Input Changed (*MIO only*)

This T,L,V is transmitted when an Input changes.

T	L	V
^h C0xx	^h 01	^h 00 : Input not asserted ^h 01 : Input asserted

The 'xx' part in the Tag is the number of the Input.

When the Host sends the **Read Inputs** command, the MIO sends one **Input Changed** messages for every Input it has.

10. COMMON CONFIGURATION REGISTERS FOR SPRINGCARD NETWORK DEVICES

10.1. GENERAL OPTIONS

The General Options register is completely defined in every product's detailed documentation. Only the common part is described here.

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

General options bits

Bits	Value	Meaning
Byte 0		
7 - 2		See the detailed documentation of the actual product you are using
1 - 0	00 SpringCard "MK2" Network Protocol, TCP mode 01 The device runs in HTTP server mode 10 The device runs in HTTP client mode 11 SpringCard "MK2" Network Protocol, UDP mode HTTP server and HTTP client modes are detailed in the product's documentation	
Byte 1 (optional)		
7 - 0		See the detailed documentation of the actual product you are using

Default value: _bXXXXXX00 XXXXXXXX

10.2. SECURITY OPTIONS

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

Security option bits

Bits	Value	Meaning	
Standard network servers			
7	0	Telnet server is disabled	
	1	Telnet server is enabled	
6	0	NDDU server is disabled	
	1	NDDU server is enabled	
5	0	RFU (set to 0)	
4	0	SpringField Colorado notifier is disabled	RDR only ²
	1	SpringField Colorado notifier is enabled	
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: _b10010100

² SpringField Colorado is a NFC-enabled application running on Android, or embedded in a specific NFC Tag, that retrieves and shows the Reader's data: firmware name and version, serial number, IP address etc.

10.3. TCP CONFIGURATION

10.3.1. IPv4 address, mask, and gateway

Name	Tag	Description	Size
IPA	$\text{h}80$	IPv4 configuration bytes, see table below	4 to 20

IPv4 configuration bytes

Bytes	Contains	Remark
0	Address, 1 st byte	Device's IPv4 Address. If these bytes are missing, the default IP Address hC0 A8 00 FA (192.168.0.250) is taken.
1	Address, 2 nd byte	
2	Address, 3 rd byte	
3	Address, 4 th byte	
4	Mask, 1 st byte	Network Mask. If these bytes are missing, the default Mask hFF FF FF FF (255.255.255.0) is taken.
5	Mask, 2 nd byte	
6	Mask, 3 rd byte	
7	Mask, 4 th byte	
8	Gateway, 1 st byte	Default Gateway. If these bytes are missing, the value h00 00 00 00 (0.0.0.0) is taken, meaning that there's no Gateway.
9	Gateway, 2 nd byte	
10	Gateway, 3 rd byte	
11	Gateway, 4 th byte	
12	DNS server 1, 1 st byte	Address of 1 st DNS server. If these bytes are missing, the value h00 00 00 00 (0.0.0.0) is taken, meaning that there's no DNS server.
13	DNS server 1, 2 nd byte	
14	DNS server 1, 3 rd byte	
15	DNS server 1, 4 th byte	
16	DNS server 2, 1 st byte	Address of 2 nd DNS server. If these bytes are missing, the value h00 00 00 00 (0.0.0.0) is taken, meaning that there's no DNS server.
17	DNS server 2, 2 nd byte	
18	DNS server 2, 3 rd byte	
19	DNS server 2, 4 th byte	

Default value: $\text{hC0 A8 00 FA FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00 00}$

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

10.3.2. SpringCard “MK2” Network Protocol – Device’s TCP listen port

Name	Tag	Description	Size
IPP	_h 81	TCP server port on the Device (2 bytes, MSB-first)	2

Default value: _h0F 9F (server TCP port = 3999)

10.3.3. SpringCard “MK2” Network Protocol – Device’s UDP listen port

Name	Tag	Description	Size
IPU	_h 82	UDP server port on the Device (2 bytes, MSB-first)	2

Default value: _h0F 9E (UDP port = 3998)

10.3.4. SpringCard “MK2” Network Protocol – Host address and UDP listen port

Name	Tag	Description	Size
IPH	_h 83	UDP server port on the Host (2 bytes, MSB-first)	2
		Host IPv4 address	4

Default value: _h0F 9D FF FF FF FF (UDP port = 3997, address = 255.255.255.255)

10.3.5. SpringCard “MK2” Network 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: _b00000100

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

10.3.6. SpringCard “MK2” Network Protocol – Operation Key

Name	Tag	Description	Size
IPK.OPE	_h 85	“MK2” Operation Key (AES 128)	16

Default value: _h00 ... _h00

10.3.7. SpringCard “MK2” Network Protocol – Administration Key

Name	Tag	Description	Size
IPK.ADM	_h 86	“MK2” Administration Key (AES 128)	16

Default value: _h00 ... _h00

10.3.8. Ethernet configuration

Name	Tag	Description	Size
ETC	_h 8D	Ethernet configuration bits. See table 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: _b00000000

10.3.9. 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 Device is installed, or what is its role in your access-control system.

10.3.10. 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 Device using Telnet protocol.

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

11. 3RD-PARTY LICENSES

SpringCard has been developed using open-source software components.

11.1. FREERTOS



FreeRTOS is an open source real time kernel, owned and managed by Amazon Web Services and provided under the following MIT open source licence:

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

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

www.freertos.org

11.2. μ IP

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

SpringCard uses FreeTCPIP, a modified version of μ IP that comes with FreeRTOS. To comply with the original license of μ IP, 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.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

DISCLAIMER

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

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

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

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

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

COPYRIGHT NOTICE

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

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

Copyright © PRO ACTIVE SAS 2018, all rights reserved.

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