

CARD READER CUSTOM CONFIGURATION (CONFIDENTIAL)

CONTENT

1	Disclaimer	5
2	Overview	6
3	Supported products	8
4	YSoft Configuration Cards	9
4.1	Configuration cards supported products	9
4.2	Uploading to Y Soft products	10
4.3	Removing custom card reader configuration from Y Soft products	11
4.4	Checking custom card reader configuration	12
5	YSoft Card Reader Tool	15
5.1	About	15
5.2	Command Line Parameters Overview	16
5.3	Configuration Page Overview	17
5.3.1	1. Connected Card Readers	17
5.3.2	2. Reload readers (F5)	17
5.3.3	3. Configure Selected Card Readers	17
5.3.4	4. Update Firmware on Selected Card Readers	17
5.3.5	5. Update Service Firmware on Selected Card Readers	18
5.3.6	6. Card Reader Detail	19
5.4	Card Reader Configuration	21
5.4.1	Configuration Options	21
5.4.2	USB Mode	22
5.4.3	Enable sound	24
5.4.4	Debug mode	24
5.4.5	Disable further usage of Configuration Cards	24
5.4.6	Reset to defaults	25
5.5	Reading Test	25
5.5.1	Reload readers (F5)	25
5.5.2	Card number and type	25
5.5.3	Save Read Logs	26
5.6	YSoft Card Reader Tool Advanced Configuration	26
5.6.1	A step-by-step guide for advanced configuration	27
5.6.2	Step-by-step configuration cards	31
5.6.3	Step-by-step custom card reader configuration	33
5.6.4	Using configuration card to configure Y Soft USB Card Reader	36
5.7	Card testing	38
5.7.1	A step-by-step guide for card testing	39
5.7.2	Card testing results	45

5.7.3	Sending card testing consultations	47
5.7.4	Quick tips: How to get the best card testing results	47
6	Card Reader Configuration Parameters	49
6.1	Common parameters	50
6.2	HF reader parameters	51
6.3	Mifare Classic (1k/4k) parameters	53
6.3.1	Mifare cards structure	53
6.4	Mifare DESFire (0.6, EV1, EV2) parameters	55
6.4.1	Mifare DESFire internal diagram	57
6.4.2	Mifare DESFire file structure	57
6.5	Mifare SAM AV2 parameters	58
6.6	Legic parameters	60
6.6.1	Legic File Structure	63
6.7	Legic Connect parameters	63
6.8	HID iClass parameters	67
6.9	HF ISO14443A/B parameters	68
6.10	LF reader parameters	69
6.11	HID Prox parameters	72
6.12	HID Indala parameters	73
6.13	Hitag 1/Hitag S256/Hitag S2048 parameters	74
6.14	Hitag 2 parameters	74
6.15	EM4050/V4050/P4150/P4350/EM4150/EM4350/EM4450/EM4550 A6 (opt64) parameters	75
6.16	ATMEL Q5 (T5555)/Q5B (T5555B)/T5557/T5567/T5577 M1/T5577 M2/T5577 M3 parameters	76
6.17	Combining technologies together	78
7	Card Manager Conversion Rules for Card Readers	83
7.1	Conversion Function	83
7.2	Conditions	84
7.3	Description of Rules	85
7.4	The Most Common Card Number Conversions	106
8	Reader Configurations	112
8.1	B-015 - Reader 3 LF	112
8.2	B-016 - 125kHz ASK FSK	116
8.3	B-017 - Multireader LF	119
8.4	B-018 - Multireader LF + HF v2	125
8.5	B-019 - Reader 3 MF	134
8.6	B-027 - Indala UIN	141
8.7	B-028 - Reader 3 Indala	144
8.8	B-037 - Legic Advant + HID Prox	148
8.9	B-038 - HID Prox v3	155

8.10	B-039 - Reader 3 LF+	158
8.11	B-041 - MultiReader LF + HF	164
8.12	B-045 - MULTI ISO	171
8.13	B-047 - Reader 3 Multi ISO	173
8.14	B-050 - Unique	175
8.15	B-051 - Reader 3 Unique	177
8.16	B-052 - TIRIS 134kHz	178
8.17	B-054 - Reader 3 TIRIS	179
8.18	B-060 - COTAG	180
8.19	B-061 - Reader 3 Cotag	181
8.20	B-063 - Reader 3 Inside Contactless	182
8.21	B-065 - Reader 3 MF+	183
8.22	B-067 - Inside Contactless	194
8.23	B-073 - Reader 3 MF SAM	195
8.24	B-074 - Reader 3 MFX	203
8.25	B-076 - MFX Mobile Reader	220
8.26	B-084 - Magnetic cards v2	241
8.27	B-086 - Reader 3 Magstripe	242
8.28	B-087 - MultiReader HF	244
8.29	B-088 - Reader 3 MF and Legic	247
8.30	B-089 - Legic Advant v3	260
8.31	B-090 - iButton (Dallas)	266
8.32	B-091 - SmartCard v2	267
8.33	B-093 - Reader 3 Smart Card	268
8.34	B-095 - Reader 3 iButton	269

1 DISCLAIMER

This document is work in progress. Y Soft Corporation a.s. is not liable for any errors or omissions that may appear in this document, and disclaims responsibility for any consequences resulting from the use of this information. Please note that this information will be subject to further changes of specifications and product descriptions.

Before you make any purchasing decision based on this document, please kindly consult your friendly Y Soft representative to confirm that the products in question comply with your requirements.

All parts of this document are protected by copyright law and all rights are reserved. This documentation may not, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior consent, in writing, from Y Soft Corporation a.s. The information set forth in this document is considered to be "Proprietary" and "Confidential" property owned by Y Soft and is subject to Y Soft's EULA available here: <https://www.ysoft.com/en/support-services/eula>

2 OVERVIEW

It is possible to create customized card reader configuration for Y Soft products.

The configuration may change the following card reader behavior:

- Card number conversions that is done in the card reader before emitting the card number (for example hex2dec conversions, binary shifts, conditions etc.)
- Reading of encrypted data (for example encrypted data from Mifare Classic, Mifare DESFire or Legic cards)
- Enabling or disabling reading of specific RFID technology or their combinations
- Error reporting in case any kind of card read error occurs
- Advanced handling of cards (special commands etc. depending on card type)

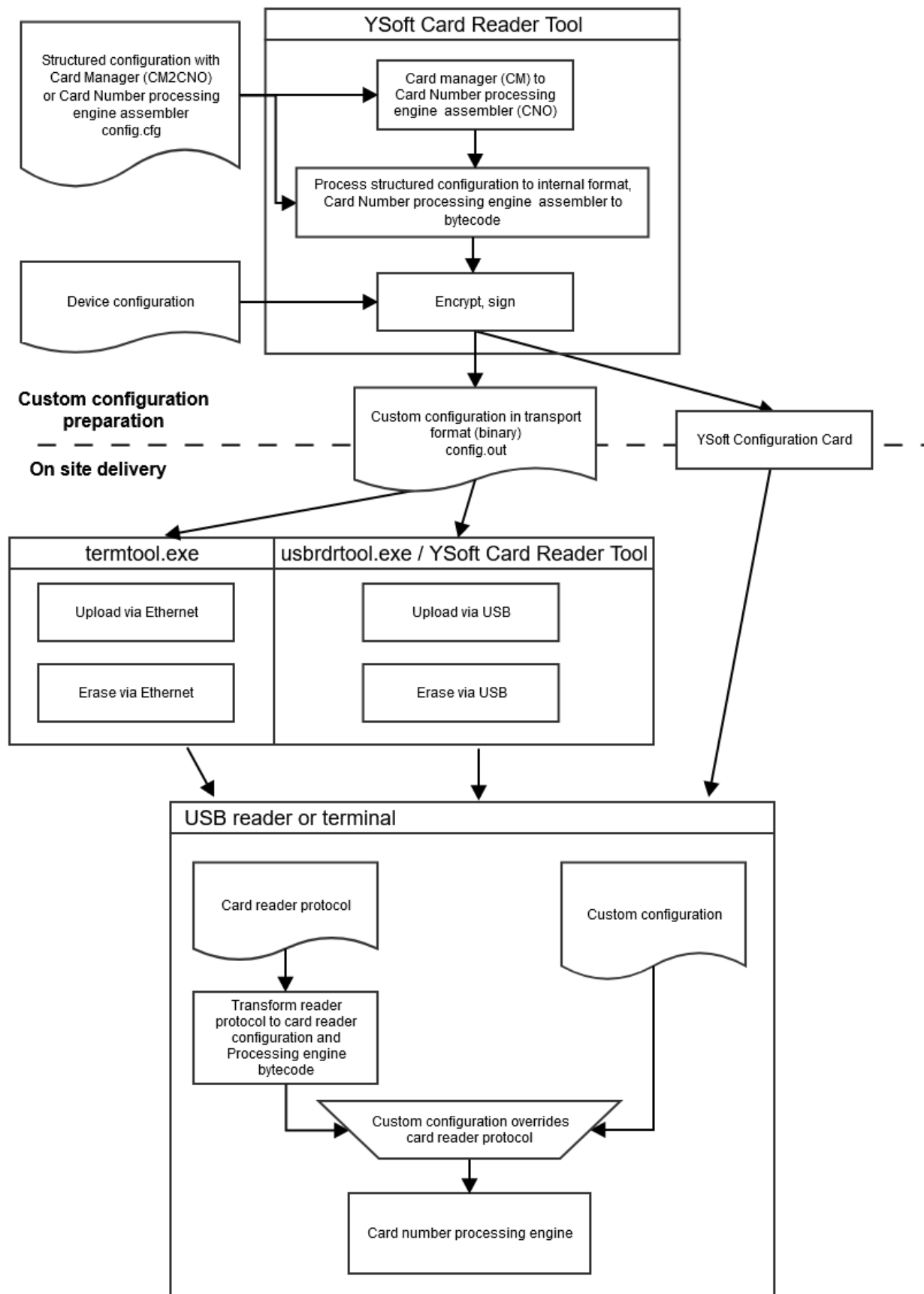
On some products (i.e. USB v3) it is also possible to store the device configuration (USB operating mode, debug, sound etc.) inside the custom card reader configuration for easy on-site deployment.

It is possible to create the custom card reader configuration in the YSoft Card Reader Tool starting from version 1.2 on Advanced configuration tab. See [YSoft Card Reader Tool Advanced Configuration](#) for more details.


It is also possible to create a configuration card that configures a card reader / terminal with predefined custom card reader configuration after the card is presented to the reader / terminal.

The configuration is encrypted and converted to a binary form. It is uploaded to supported Y Soft products using standard usbrdrtool.exe, termtool.exe and YSoft Card Reader Tool applications.

The card reader configuration's checksum is reported as version so customers can see that the proper custom configuration is in place. For security reasons it is not possible to read the configuration back.



3 SUPPORTED PRODUCTS

 Please note that it is highly recommended to update to the latest firmware version as it may contain important functional fixes and improvements.

Product	Custom configuration v1 (legacy rdrcfgtool.exe)		Custom configuration v2 (YSoft Card Reader Tool)	
SafeQ Recharging Station v2	✓	From firmware 3.15.0	✓	From firmware 3.15.12 Card reader custom config only, no device config
SafeQ Payment Machine	✓	From firmware 3.15.0	✓	From firmware 3.15.12 Custom config only, no device config
Terminal Professional v3 monochrome	✓	From firmware 3.15.0	✓	From firmware 3.15.12 Custom config only, no device config
Terminal Professional v3 color	✓	From firmware 3.15.0	✓	From firmware 3.15.12 Custom config only, no device config
USB Card Reader v2	✓	From firmware 2.2.0	✓	From firmware 2.4.0
FX EPA Reader	✓	From firmware 2.2.0	✓	From firmware 2.4.0
USB Card Reader v3	✓	From firmware 2.2.4	✓	From firmware 2.4.0
Terminal Ultralight	✓	From firmware 1.3.0	✓	From firmware 1.3.1
Network Card Reader	✓	From firmware 1.3.0	✓	From firmware 1.3.1
Terminal Professional v4	✗	Support planned	✗	Support planned

4 YSOFT CONFIGURATION CARDS

Starting from version of 1.2 of the YSoft Card Reader Tool it is possible to create a configuration card that configures a card reader with a predefined custom card reader configuration. See [YSoft Card Reader Tool Step-by-step configuration cards](#) for more details how to create configuration cards.

On some products (i.e. USB v3) it is also possible to store the device configuration (operating mode, debug, sound etc.) on the card or inside the custom card reader configuration.

Configuration cards work only if all the the following points are fulfilled:

1. They work only on selected card readers (see table below)
2. ISO14443-A (HF technology) must be enabled either by protocol selection or in custom card reader configuration. This is not required if USB firmware 2.5.1 or newer is used.
3. Configuration cards processing is enabled in device configuration
4. Configuration cards work only within 2 minutes from the device power-up or restart
5. It is necessary to keep the card placed for at least 10-15 seconds at the reader. If it is removed earlier then it will not be processed.
6. Configuration cards are already disabled on custom protocols.

In all other cases they will not work at all, or they will register as a standard ISO14443-A cards.

Card readers from production have configuration cards enabled and protocol set to default value.

Correct signalization of start of processing of configuration cards is by blue LED on USB v3.

Correct signalization of card processed is by Green LED on USB v3.

4.1 CONFIGURATION CARDS SUPPORTED PRODUCTS

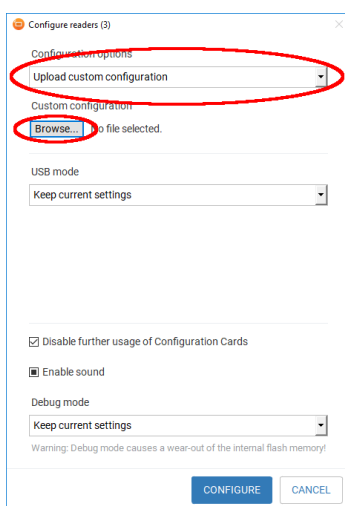
Product	Configuration cards supported	From Firmware	Note
SafeQ Recharging Station v2	✗		Not supported
SafeQ Payment Machine	✗		Not supported
Terminal Professional v3 monochrome	✗		Not supported
Terminal Professional v3 color	✗		Not supported
USB Card Reader v2	✗		No supported card readers

Product	Configuration cards supported	From Firmware	Note
FX EPA Reader	✓	2.4.0	Reader 3 MF+ Only
USB Card Reader v3	✓	2.4.0	Reader 3 MF Reader 3 MF+ Reader 3 MF SAM Reader 3 MF&Legic
		2.5.0	Reader 3 MFX
		2.6.0	MFX Mobile Reader
Terminal Ultralight	✓	1.3.1	Reader 3 MF+ Only
Network Card Reader	✗		No supported card readers
Terminal Professional v4	✗		Support planned

4.2 UPLOADING TO Y SOFT PRODUCTS

Using YSoft Card Reader tool for USB v2/v3 card readers:

Configuration → Configure → Upload custom configuration → Browse



Using usbrdrtool.exe for USB v2/v3 card readers:

Usbrdrtool version at least 1.14 is needed

```
usbrdrtool.exe -i config.out -m
```

Using termtool on Terminal Professional v3 or SPM:

Termtool version at least 0.12 needed. You will be prompted for service menu level 2 pin

```
termtool.exe -I <terminal_ip_address> -i config.out -CR
```

Using termtool on Terminal Ultralight / Network Card Reader:

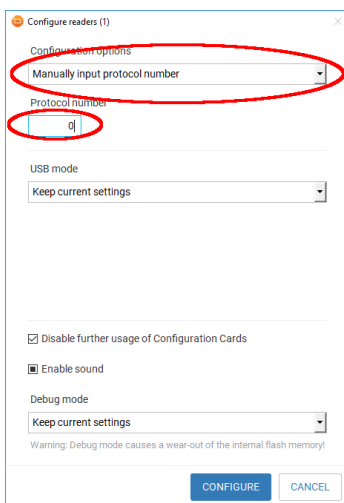
Termtool version at least 0.12 needed.

```
termtool.exe -I <terminal_ip_address> -i config.out -Cr
```

4.3 REMOVING CUSTOM CARD READER CONFIGURATION FROM Y SOFT PRODUCTS

Using YSoft Card Reader tool for USB v2/v3 card readers:

Remove configuration by setting card reader protocol to default (protocol 0) or any other value.



Using usbrdrtool on USBv2/v3 card readers:

Remove configuration by setting card reader protocol to default or any other valid value.

Starting from usbrdrtool 1.17 you can configure/reset the protocol from command line:

```
usbrdrtool.exe -p 0 -k
```

Using termtool on Terminal Professional v3 or SPM:

Remove configuration by setting reader protocol to default or any other valid value

Termtool version at least 0.12 needed. You will need service menu level 2 pin

```
echo READERPROTOCOL=0 | termtool.exe -I <terminal_ip_address> -C2 -p  
<service_menu_level2_pin>
```

Using termtool on Terminal Ultralight / Network Card Reader

Remove configuration by setting reader protocol to a valid value.

Termtool version at least 0.12 needed.

```
echo READERPROTOCOL=1220 | termtool.exe -I <terminal_ip_address> -c
```

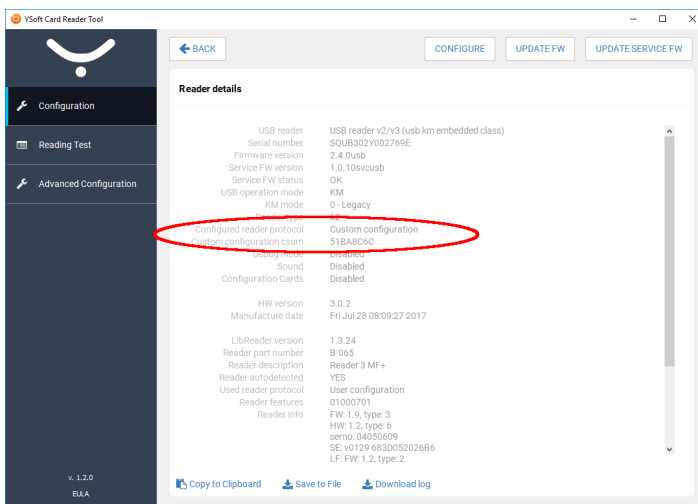
4.4 CHECKING CUSTOM CARD READER CONFIGURATION

Checksum (version) of the card reader configuration can be obtained as a first line from the resulting .out file:

```
# Structured configuration checksum: EF08C60F  
AgIDAgAAAAGDvCMYPAAAA+MiPmIC7vMZ7LS80wxUOTAR+uZKqLa60mlkjtRXWZo1vazA8l30YebiG  
ehuaNPpLPqEZx9Fsa437asga/hwetwDYi1+7kQxxEfq0ILC/4/atKY0oZxPse7cqGtBoFsv1zw==
```

Using YSoft Card Reader tool for USB v2/v3 card readers:

Configuration → Detail on a reader



Using usbrdrtool on usbv2/v3 card readers:

Start the application without any parameter and open the reader's information screen

```
C:\Temp\rdrctgtool\usbrdrtool.exe

USB Reader information:
Usb reader      : USB reader v2/v3 (usb keyboard class)
Serial no.      : SQUB3020000007E
Firmware ver.   : 2.2.8usb
Servicefw ver.  : 1.0.8svcusb
Servicefw stat. : OK
USB op. mode    : USB keyboard
USB kbd params  : 0 ms press, 0 ms release
Reader time     : 60
Reader proto    : User configuration
Usb cfg csum    : EF08C60F
Debug mode      : no log
Sound           : Enabled

Hardware information:
HW version      : 3.0.2
Serial number   : SQUB3020000007E
Manuf. date     : Thu Aug 25 13:22:10 2016

Card reader information:
LibReader ver.  : 1.3.13
Reader count    : 1
Reader partno   : B-065
Reader descr.   : Reader 3 MF+
Reader auto     : YES
```

Using termtool.exe on Terminal Professional v3, SPM, Terminal Ultralight / Network Card Reader:

Start the application without any parameter and open the terminal's information screen

```
C:\Temp\rdrctgtool\termtool.exe

Advanced information:
Auth. type      : 3 = Card or PIN
Mode            : Normal
Joblist mode    : Queue/printed/favorites
PIN dlg. text   : Without characters count
P/C alert msg.  : Enabled
Sum. after P/C : Pages and price
Menu timeout    : 30
Info timeout    : 20
Sound           : Disabled
Debug mode      : No log

Card reader and IO module:
Reader type     : 44 = B-087 Multireader W1
Reader proto    : User defined configuration
IOmodule type   : 1 = N/A Generic IOmodules
IOmodule mode   : 9 = Autodetect smartcable

Language information:
Default lang.   : 2 = English
Other langs.    : 1 = Czech;2 = English;
Lang. selector : Direct for 2 lang mode

-----
Terminal:
```

Configuration checksum using termtool.exe on Terminal Professional v3, SPM, Terminal Ultralight / Network Card Reader:

Termtool version at least 0.12 needed. You will need service menu level 1 pin

```
termtool.exe -I <terminal_ip_address> -GR -p <service_menu_level1_pin>
```

TPv3 / SPM service menu information

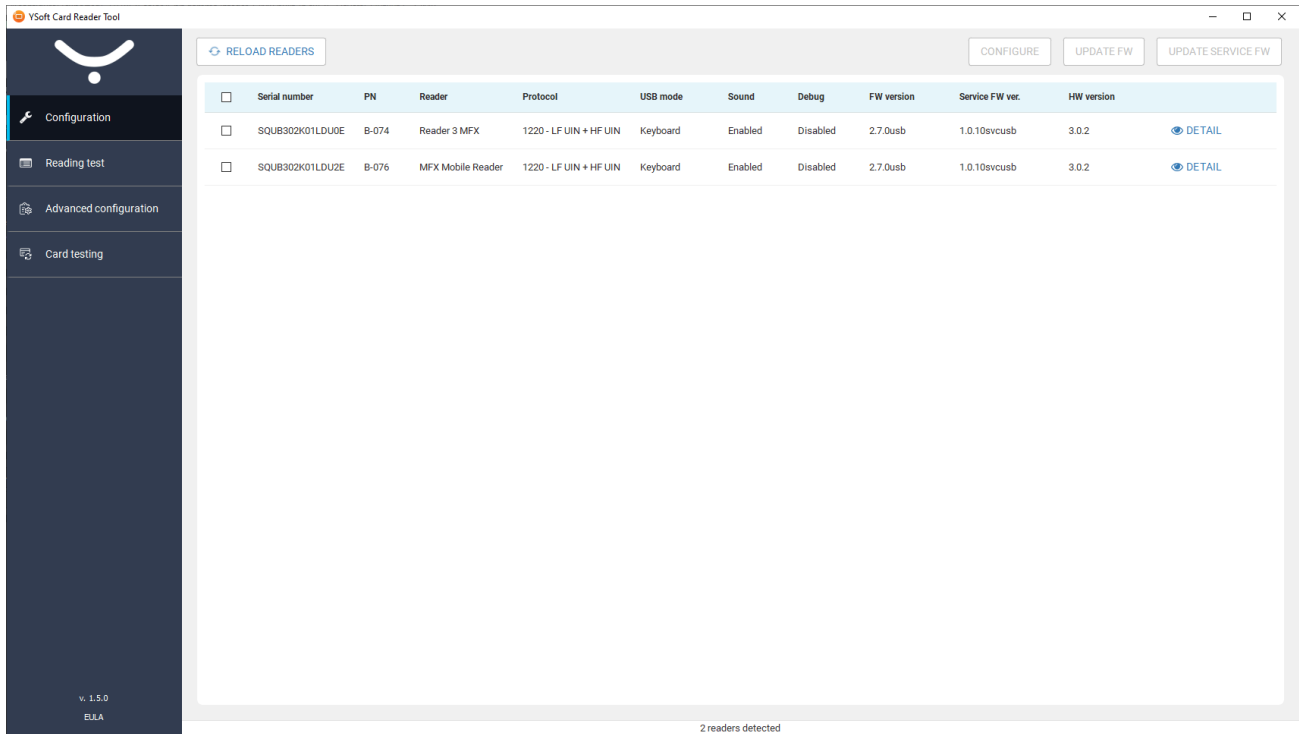
Go to service menu -> Card Reader Settings -> Reader Drivers Info

Part number:	B-087
Reader type:	MultiReader HF
Autodetected:	YES
Protocol (csum):	User configuration (EF08C60F)
Features:	00000401
Additional info:	FW: 1.6, type: 3
	HW: 1.1, type: 1
	serno: 00000AF7
Touch the screen to continue	1/1

5 YSOFT CARD READER TOOL

5.1 ABOUT

This document describes the configuration tool version 1.5.0 for YSoft USB Card Readers v2/v3



The tool consists of the following parts:

Configuration

- A list of connected readers with basic details
- Operations on one or multiple selected readers
 - Configure readers (Custom configuration, Reader protocol, USB mode, Debug setting, etc.)
 - Update main firmware
 - Update service firmware
- Show full reader details
- Download reader log


Reading test

- A list of connected readers with the last read card number and card type
- Save read logs as a text file

Advanced Configuration

- A list of connected readers with the last read of YSoft Configuration Card label

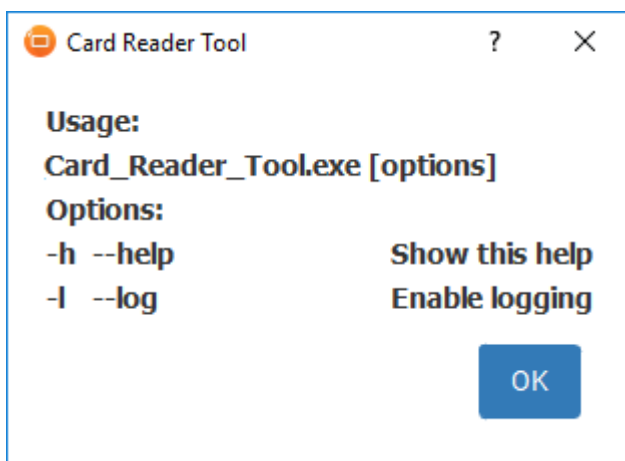
- Format compatible cards as YSoft Configuration Card
- Define the configuration content for Configuration Card (import existing configuration, define custom configuration, reader protocol, device configuration, card label, etc.)
- Custom configuration can be:
 - Loaded from a text file
 - Manually defined
 - Reused from predefined protocols
 - Repeatedly evaluated and tested on connected readers
 - Stored to connected readers
- Write the configuration to Configuration Card or export to disk

 Advanced configuration is intended for advanced users/experts only and is described in a separate document.

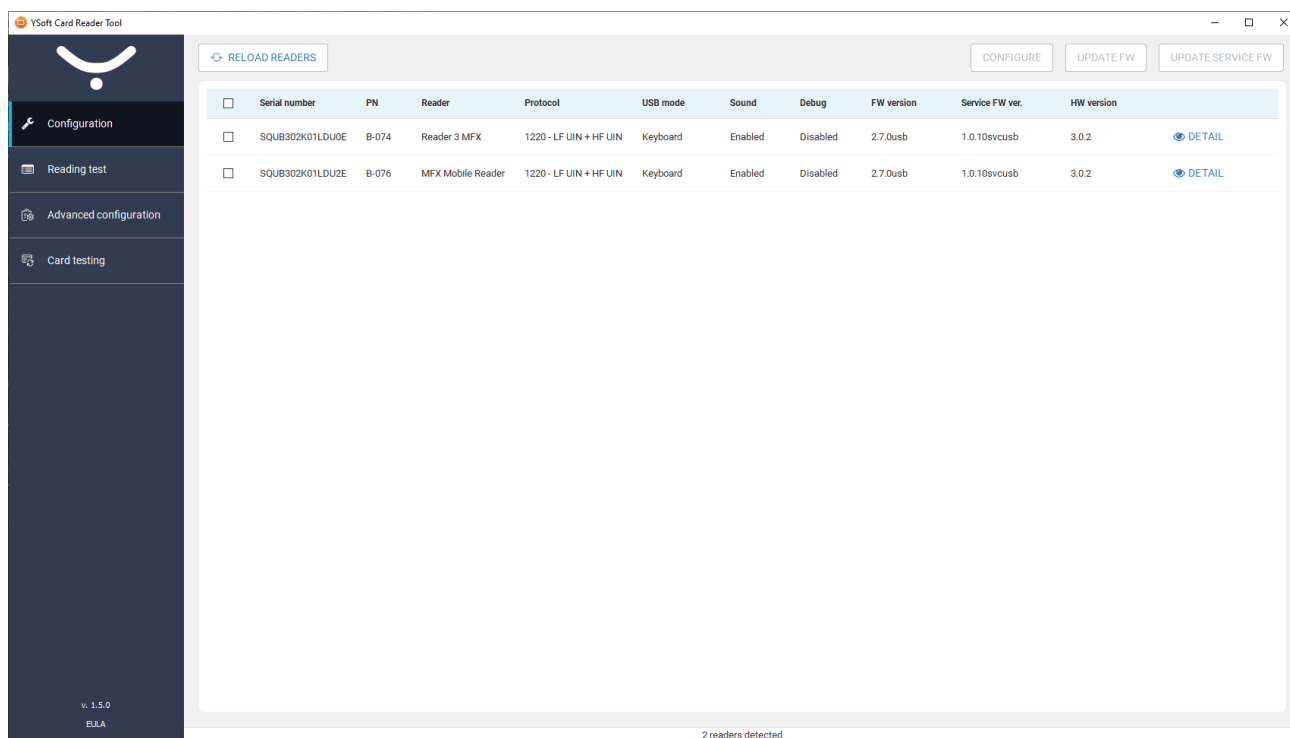
Card testing

- Card testing wizard for card type identification

5.2 COMMAND LINE PARAMETERS OVERVIEW



5.3 CONFIGURATION PAGE OVERVIEW



5.3.1 1. CONNECTED CARD READERS

<input type="checkbox"/>	Serial number	PN	Reader	Protocol	USB mode	Sound	Debug	FW version	Service FW ver.	HW version	
<input type="checkbox"/>	SQUB302Y002769E	B-065	Reader 3 MF+	1220 - LF UIN + HF UIN	KM	Disabled	Disabled	2.2.12usb	1.0.9svcusb	3.0.2	DETAIL
<input type="checkbox"/>	SQUB302Y00A7FBE	B-047	Reader 3 Multi ISO	44 - Multi ISO UIN	Keyboard	Enabled	Disabled	2.2.12usb	1.0.8svcusb	3.0.2	DETAIL

The table is sortable. Sort by the desired column by clicking its header.

Operations are performed on selected readers. Select them by clicking the select checkbox, or anywhere on the line (except the detail button).

The checkbox at the upper left-hand side acts as Select/Deselect all

5.3.2 2. RELOAD READERS (F5)

Refresh the list of connected readers using the **Reload readers** button.

5.3.3 3. CONFIGURE SELECTED CARD READERS

Open the configuration dialog for the selected card readers using the **Configure** button. [Card Reader Configuration](#).

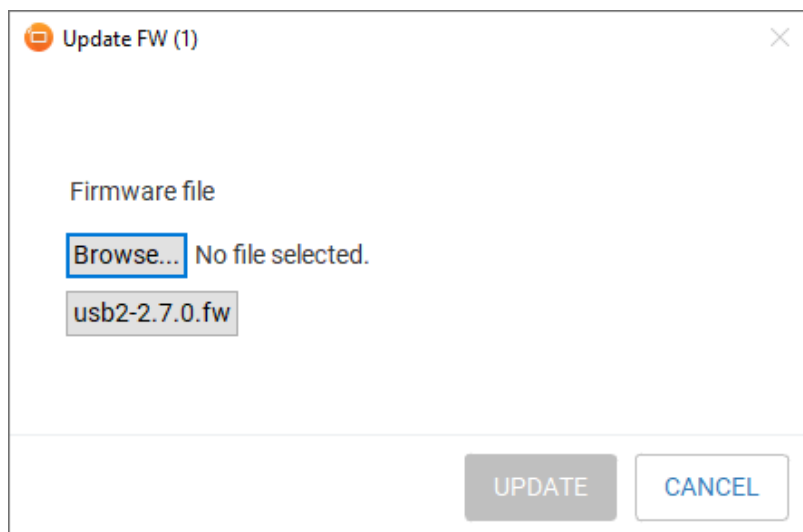
5.3.4 4. UPDATE FIRMWARE ON SELECTED CARD READERS

Open the update firmware dialog for the selected card readers using the **Update FW** button.

Firmware update

Click **Browse**, then select the latest firmware package and click **Update**.

Another possibility is to click the button with the name of the embedded firmware. The embedded firmware file name appears as the selected file.



Starting with firmware 2.3.0, it is necessary to manually update the service firmware of the USB reader at least to version 1.0.9

With any older versions, 2.3.0 and newer firmware will not be recognized as valid firmware.

5.3.5 5. UPDATE SERVICE FIRMWARE ON SELECTED CARD READERS

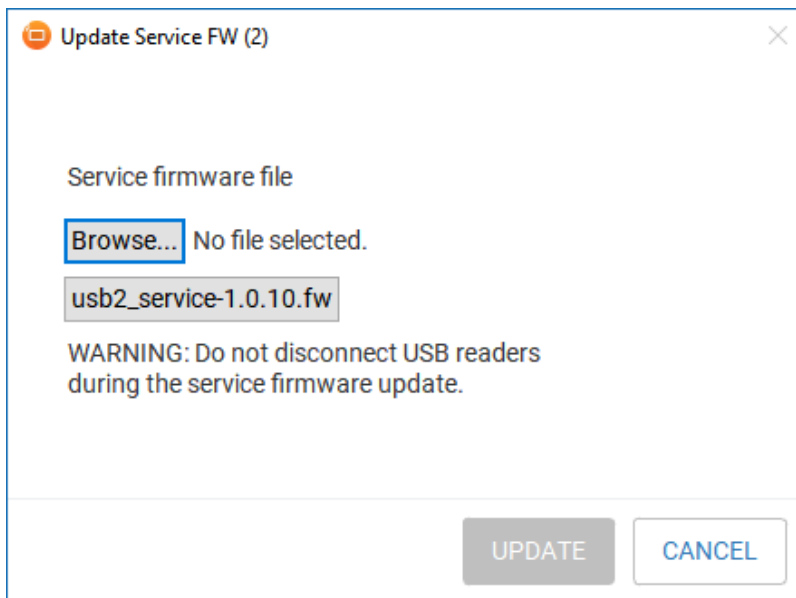
Open the service update firmware dialog for the selected card readers using the **Update Service FW** button.

Service firmware update

Click **Browse**, then select the latest service firmware package.

Another possibility is to click the button with the name of the embedded service firmware. The embedded firmware file name appears as the selected file.

Then click **Update**. Do not disconnect USB card readers during the update procedure!



5.3.6 6. CARD READER DETAIL

Show the card reader detail page using the **Detail** button:

Reader details	
USB reader	USB reader v2/v3 (usb keyboard class)
Serial number	SQUB3020000123E
Firmware version	2.5.0usb
Service FW version	1.0.10svcusb
Service FW status	OK
USB operation mode	Keyboard
USB keyboard parameters	1 ms press, 1 ms release
USB keyboard layout	0 - Numpad
Reader type	76
Configured reader protocol	0
Debug mode	Disabled
Sound	Disabled
Configuration Cards	Enabled
HW version	3.0.2
Manufacture date	Wed Mar 22 15:44:16 2017
LibReader version	1.3.32
Reader part number	B-074
Reader description	Reader 3 MF X
Reader autotected	YES
Used reader protocol	1220 - LF UIN + HF UIN
Reader features	01000701
Reader info	FW: 2.1, type: 5, HW: 3.1, type: 7 SE: v0129 683D052026B6 LF:f:126003Hz,Q:8.8,Ampl:36.2V,caps:9,6,5. HF:AP:10,err:33,Tun:4:18.8V,Mod:C0,28,E8.
Reader status	Reader operating ok
Status message	
Module PCB version (type)	3.1.2 (27)
Module features	00000000
Module serial number	SQRDJ33312BB0005E
Module part number	B-074
Module manufacture date	Fri Aug 23 11:49:03 2019
Module SW version	Not applicable
Module requires LibReader version	1.3.32
Copy to Clipboard Save to File Download log	

Copy to Clipboard

Copy the reader details to the clipboard or to a text file.

Save to File

Save reader details to a text file.

Download log

Download logs from the card reader and save to a text file. If the log file is empty, then check if debug mode is correctly set in the configuration.

5.4 CARD READER CONFIGURATION

Configure readers (2)

Configuration options

Keep current protocol

USB mode

Keep current settings

☒ Enable sound

Debug mode

Keep current settings

Warning: Debug mode causes a wear-out of the internal flash memory!

☒ Disable further usage of Configuration Cards

☐ Reset to defaults

CONFIGURE **CANCEL**

5.4.1 CONFIGURATION OPTIONS

Configuration options

Keep current protocol

Keep current protocol

Select available protocol

Manually input protocol number

Upload custom configuration

Keep current protocol

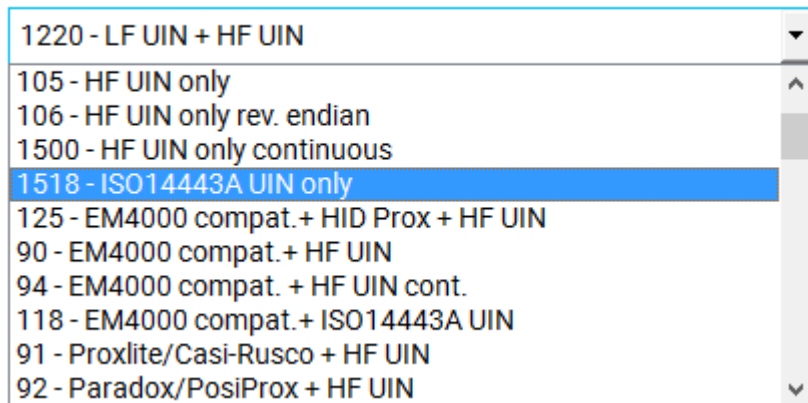
Do not change the configured protocol or custom configuration.

Select available protocol

Select a protocol from the list of available protocols.

When you configure more than one type of card reader at once, this mode is not available. You can set the protocol number manually.

Available protocol



Manually input protocol number

Manually set the protocol number.

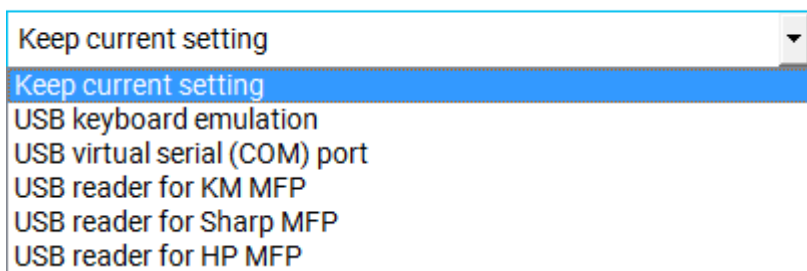
Upload custom configuration

Upload a binary file with the custom configuration.

5.4.2 USB MODE

Change the mode of the USB interface.

USB mode



Keep current setting

Do not change the USB mode.

USB keyboard emulation

In this mode, the USB reader works as a standard USB keyboard.

The typing of keys is emulated as a numeric keypad for 0-9 and A-F keys in standard EN/US keyboard layout. If the card number contains any other characters, they are ignored. Caps-lock and

Num-lock are enabled and disabled if necessary and returned to their previous state after the card number is read. The "Shift", "Ctrl", and "Alt" keys must not be held during card placement, or it will collide with Caps-lock and Num-lock settings, and an incorrect card number will be read. If an international keyboard layout is used (such as French, Russian, Chinese, or another), a wrong card number will be entered. In such cases, switching to US/EN keyboard layout is necessary before placing the card.

USB mode

USB keyboard emulation

Keyboard parameters

0

msec key pressed

0

msec key released

Parameters:

- msec key pressed
- msec key released

USB virtual serial (COM) port

This mode emulates the operation of a COM port.

On a Windows platform, installation of the USB reader in this mode requires a `usb2-reader.inf` file. This file is distributed along with the USB reader firmware and should be selected in the "new hardware" wizard after USB reader plugging/reconfiguration.

On a Linux platform, the driver for the serial port is installed automatically.

USB reader for Konica Minolta MFD

This mode is for Konica Minolta (KM) MFDs.

USB mode

USB reader for KM MFP

KM mode

0 - Legacy

0 - Legacy

1 - AU201

KM mode

- Legacy – ASCII mode
- AU201 – binary mode

USB reader for Sharp MFD

This mode is for Sharp MFDs.

USB reader for HP MFD

This mode is for HP MFDs. It is not supported on USB v2 card readers.

USB reader in HID raw mode

This mode is intended for special cases.

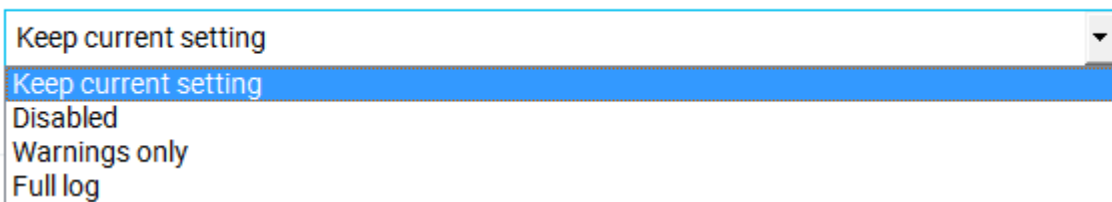
5.4.3 ENABLE SOUND

Values:

- ☒ **Enable sound** - Enable Sound
- ☐ **Enable sound** - Disable sound
- ☐ **Enable sound** - Do not change sound configuration

5.4.4 DEBUG MODE

Debug mode



Values:

- Keep current setting – do not change the logging configuration
- Disabled – logging is disabled
- Warnings only – only log warning messages
- Full log – debug logging.

⚠ Debug logging causes a wear-out of the internal flash memory. For short-term use only.
The log is a 32Kb circular buffer with 10k erase cycles. Card reader operation with worn-out flash memory has not been verified.

5.4.5 DISABLE FURTHER USAGE OF CONFIGURATION CARDS

Values:

- ☒ **Disable further usage of Configuration Cards** - Disable further usage of Configuration Cards
- ☐ **Disable further usage of Configuration Cards** - Enable further usage of Configuration Cards
- ☐ **Disable further usage of Configuration Cards** - Do not change the configuration setting

5.4.6 RESET TO DEFAULTS

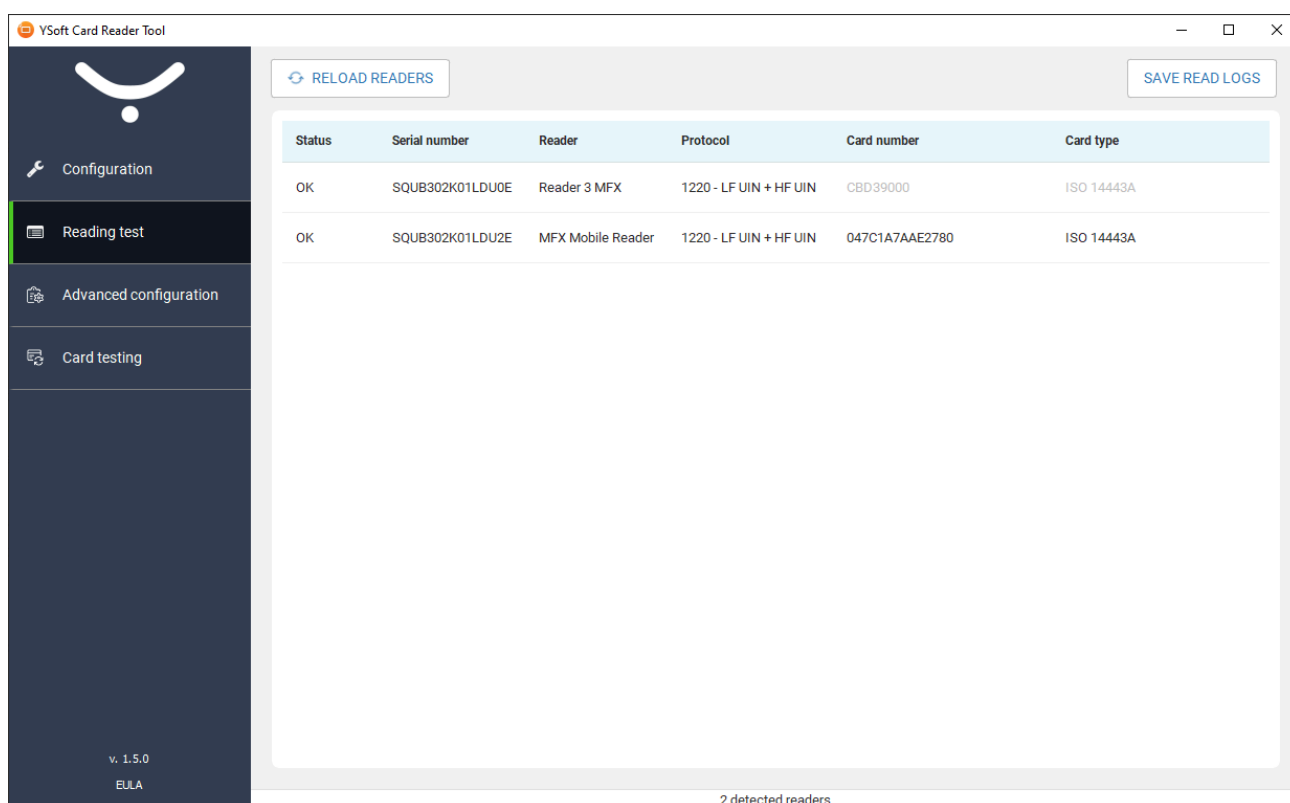
Values:

- ☐ **Reset to defaults** - Do not perform configuration reset
- ☒ **Reset to defaults** - Perform configuration reset to manufacturing defaults (Could not be combined with other options)

5.5 READING TEST

In this section, you can test the reading of your card readers and save testing results to a file. Multiple card readers can be tested at once.

When the status of the card reader is OK, you can start placing cards on the card reader.



5.5.1 RELOAD READERS (F5)

Refresh the list of connected readers and clean the card reading log.

5.5.2 CARD NUMBER AND TYPE

You can find the number and type of the last placed card.

When using in KM mode, the card number also includes MFP data emitted from a loadable driver (in parenthesis).

Card number	Card type
6B000A940E	EM4000 and compatible

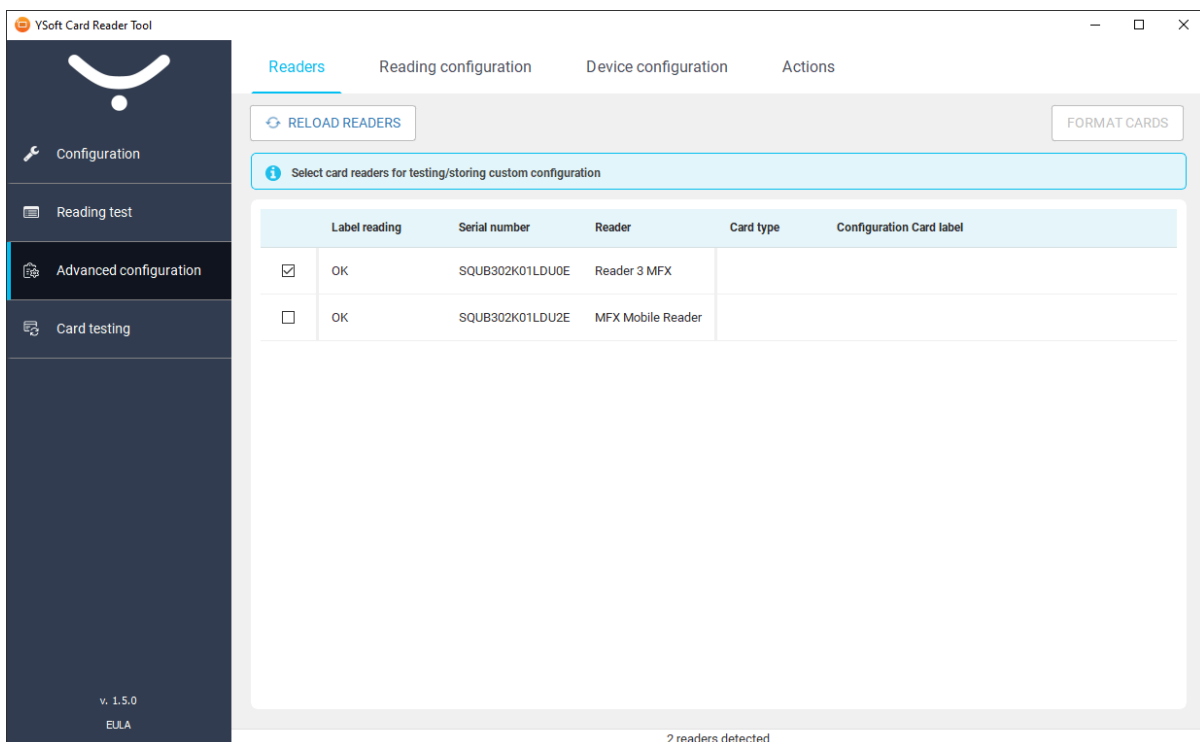
5.5.3 SAVE READ LOGS

Download a log of all cards numbers read by the card readers using the **Save Read Logs** button.

Log example

```
-----
USB reader v2/v3 (usb keyboard class) (SQUB302Y002769E)
B-065 Reader 3 MF+ (1220 - LF UIN + HF UIN)
No.          Card number    Card type
1.          6B000A8B80    EM4000 and compatible
2.          6B000A4928    EM4000 and compatible
3.          6B000A940E    EM4000 and compatible
-----
```

5.6 YSOFT CARD READER TOOL ADVANCED CONFIGURATION



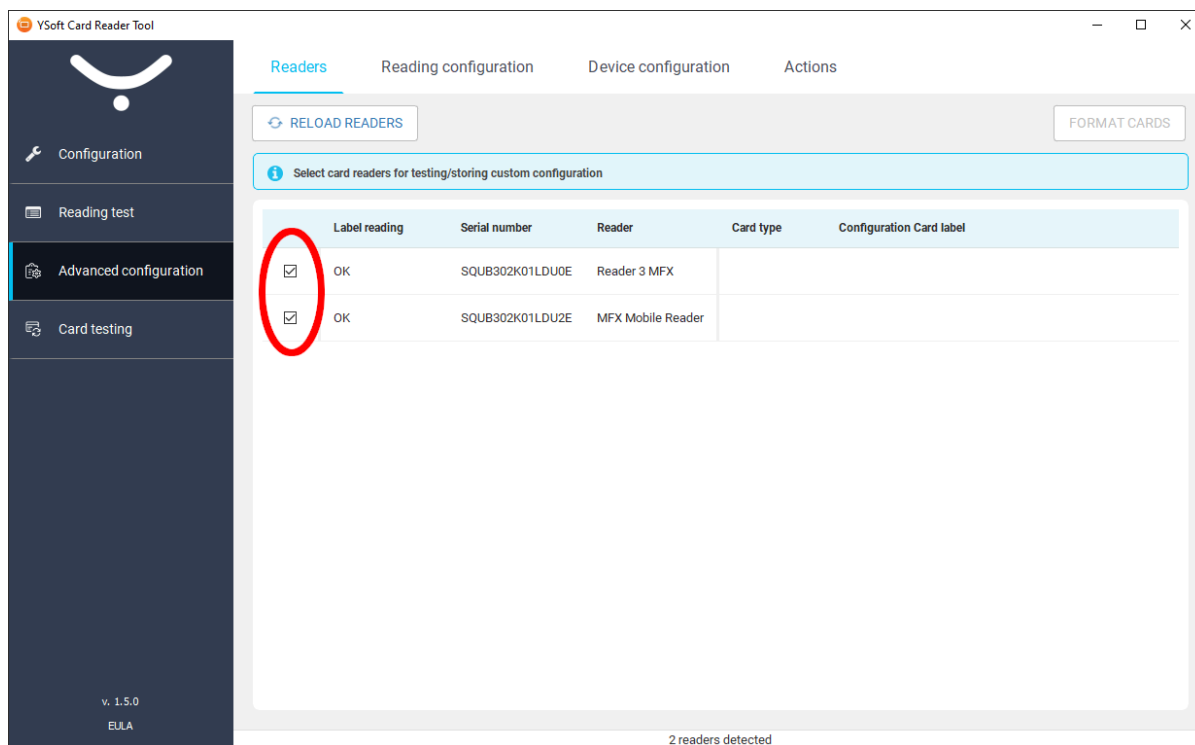
Starting with version 1.2 of YSoft Card Reader Tool, it is possible to create or upload an advanced configuration for Y Soft card readers. The advanced configuration includes:

1. Customized card reading
2. Device operational parameters
3. Creation of custom configuration cards

5.6.1 A STEP-BY-STEP GUIDE FOR ADVANCED CONFIGURATION

Step 1:

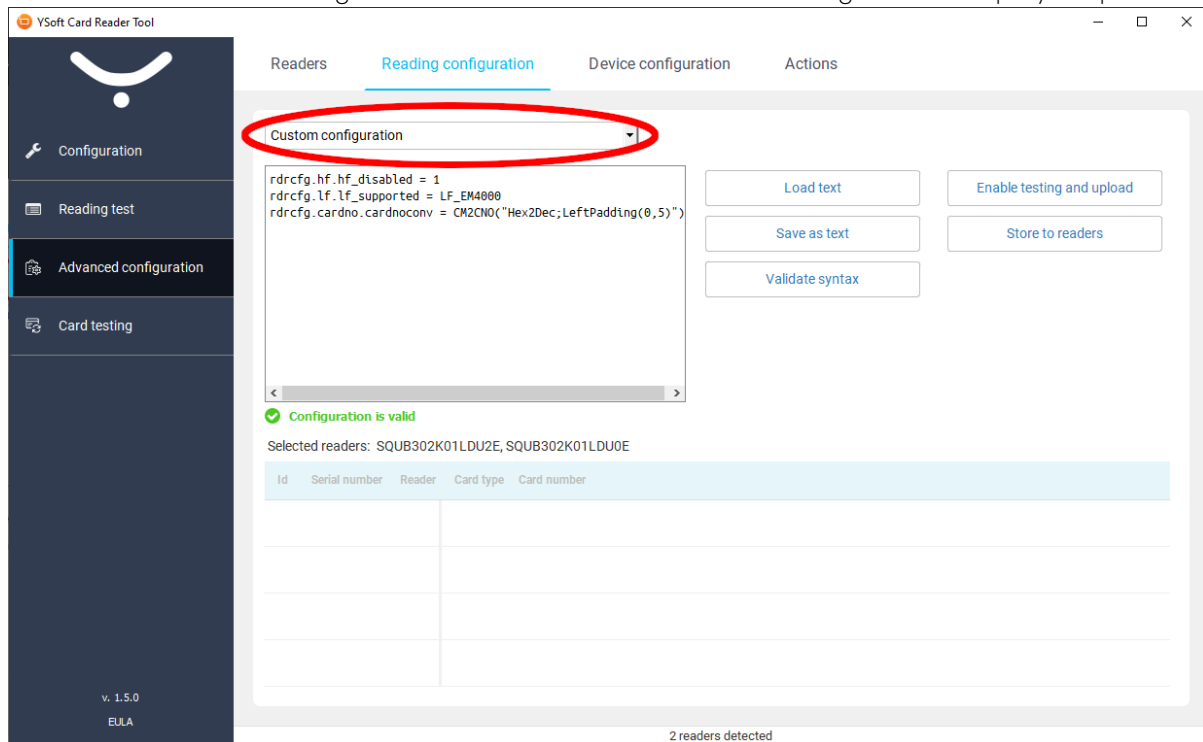
Select which reader you want the advanced configuration tested on in the next reading configuration dialog. This step can be omitted if no testing is required.



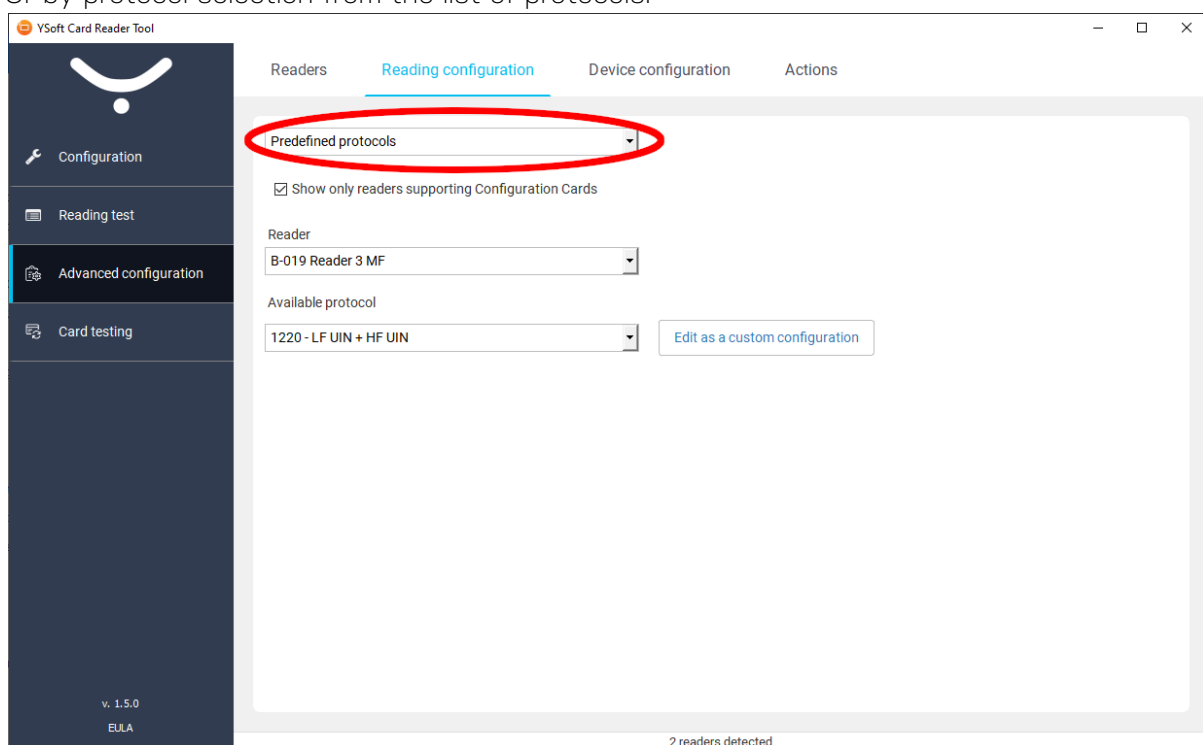
Step 2:

Select the card reader configuration either by:

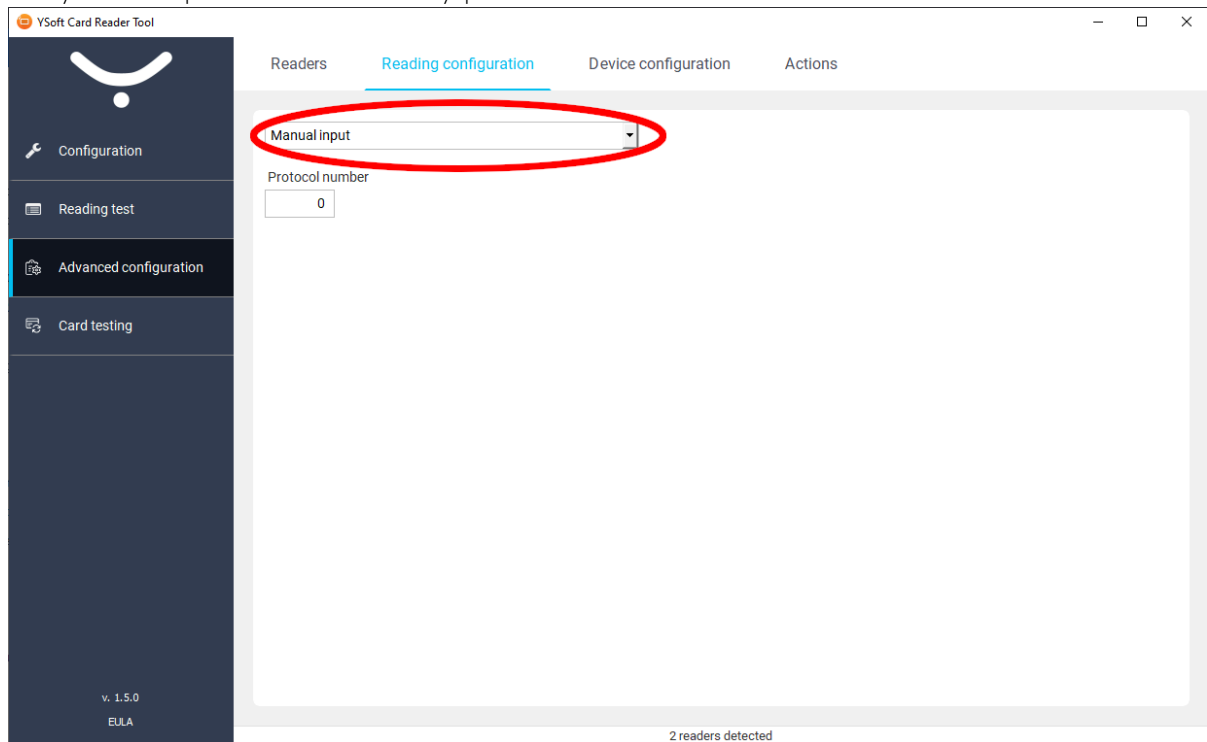
1. Custom card reader configuration. See below for a custom configuration step-by-step.



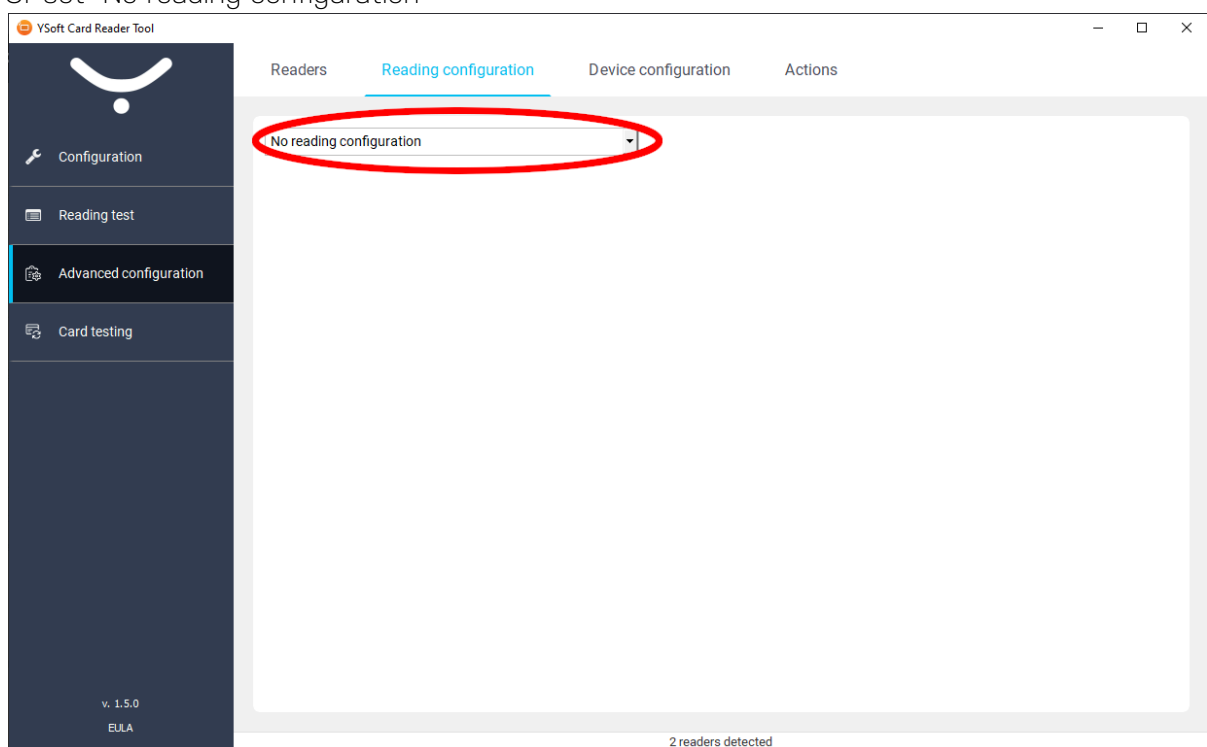
2. Or by protocol selection from the list of protocols.



3. Or by manual protocol selection by protocol number.



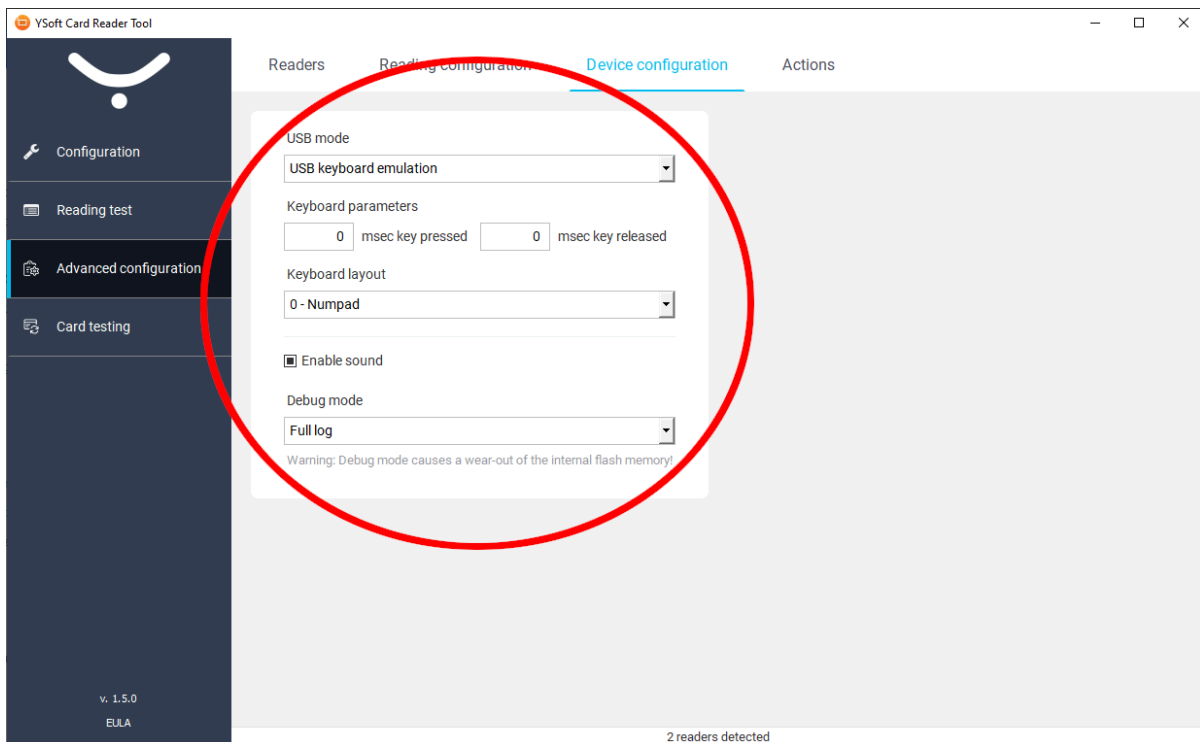
4. Or set "No reading configuration"



Step 3:

Select device configuration.

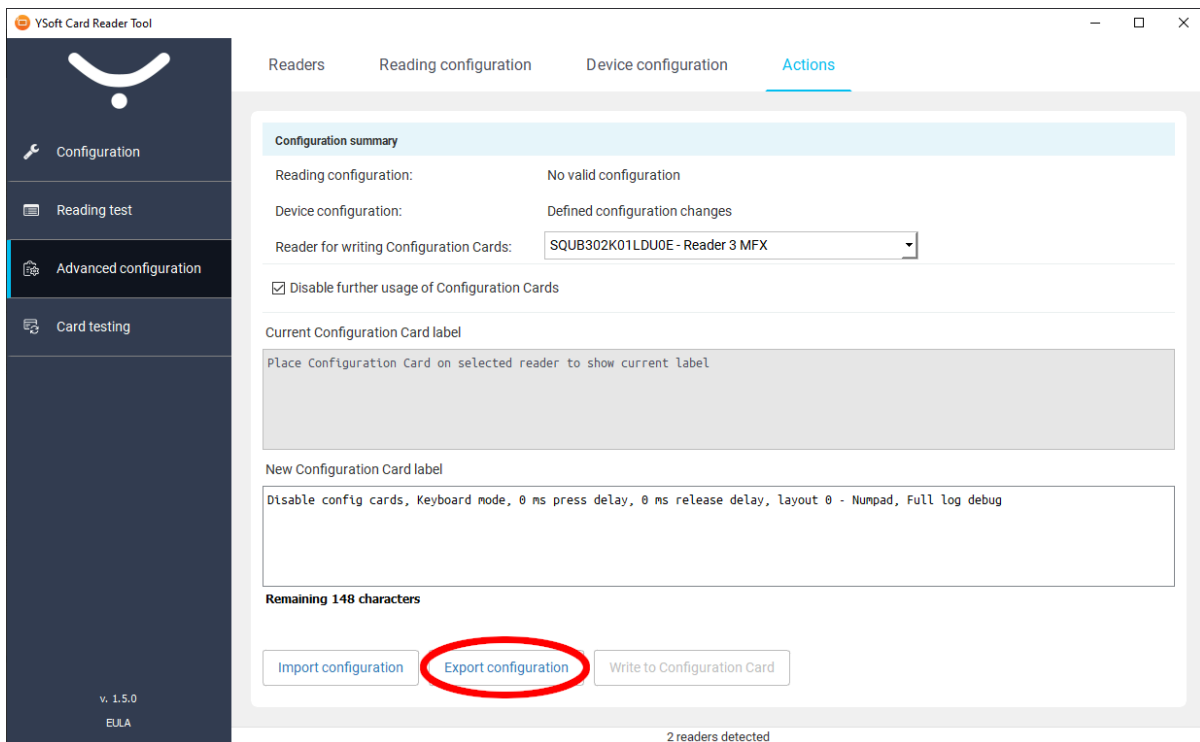
May be left without any change if no changes are required. Items correspond to the options in the configuration dialog.



Step 4:

Save the configuration

- Save the configuration to a file by clicking export configuration. The file can be later used in the **Configuration > Configure dialog > Upload custom configuration**.
- Or the config can be written on a configuration card, see below for step-by-step configuration cards.



⚠ Users may start with an already existing custom configuration. Then it is possible to import it and define only the overriding changes. Redefining custom configuration is not supported.

5.6.2 STEP-BY-STEP CONFIGURATION CARDS

See the "Card Reader Custom Configuration" document for details and limitations about the configuration cards usage. Please ask your sales representative for the document.

⚠ Configuration Cards are supported from firmware version 2.4.0 on these readers:

- USB Reader 3 MF
- USB Reader 3 MF+
- USB Reader 3 MF&Logic
- USB Reader 3 MF SAM

From firmware version 2.5.0 on this reader:

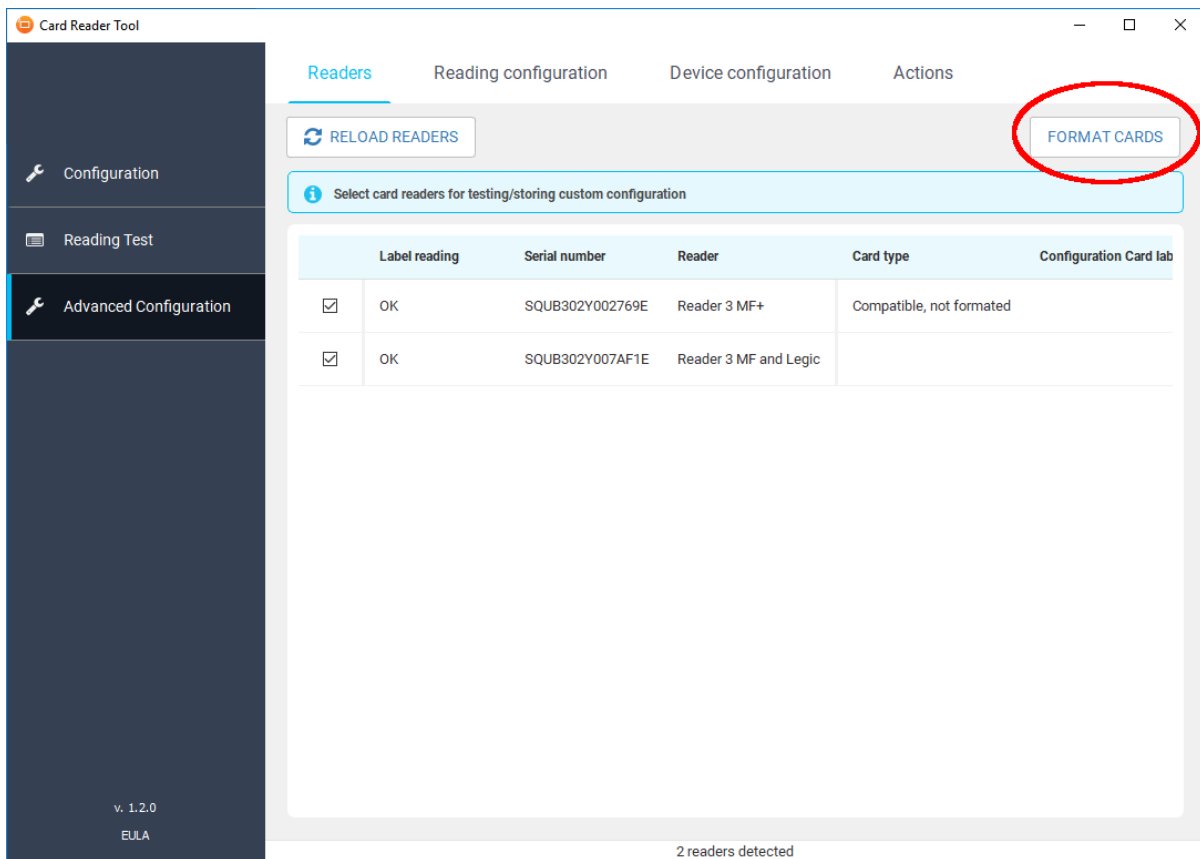
- USB Reader 3 MFX

From firmware version 2.6.0 on this reader:

- MFX Mobile Reader

Step 1: Format empty card.

Place the compatible card and click FORMAT CARDS. This step is skipped if the card is already formatted.



Compatible cards are:

1. Mifare DESFire EV1 8k
2. Mifare DESFire EV2 8k

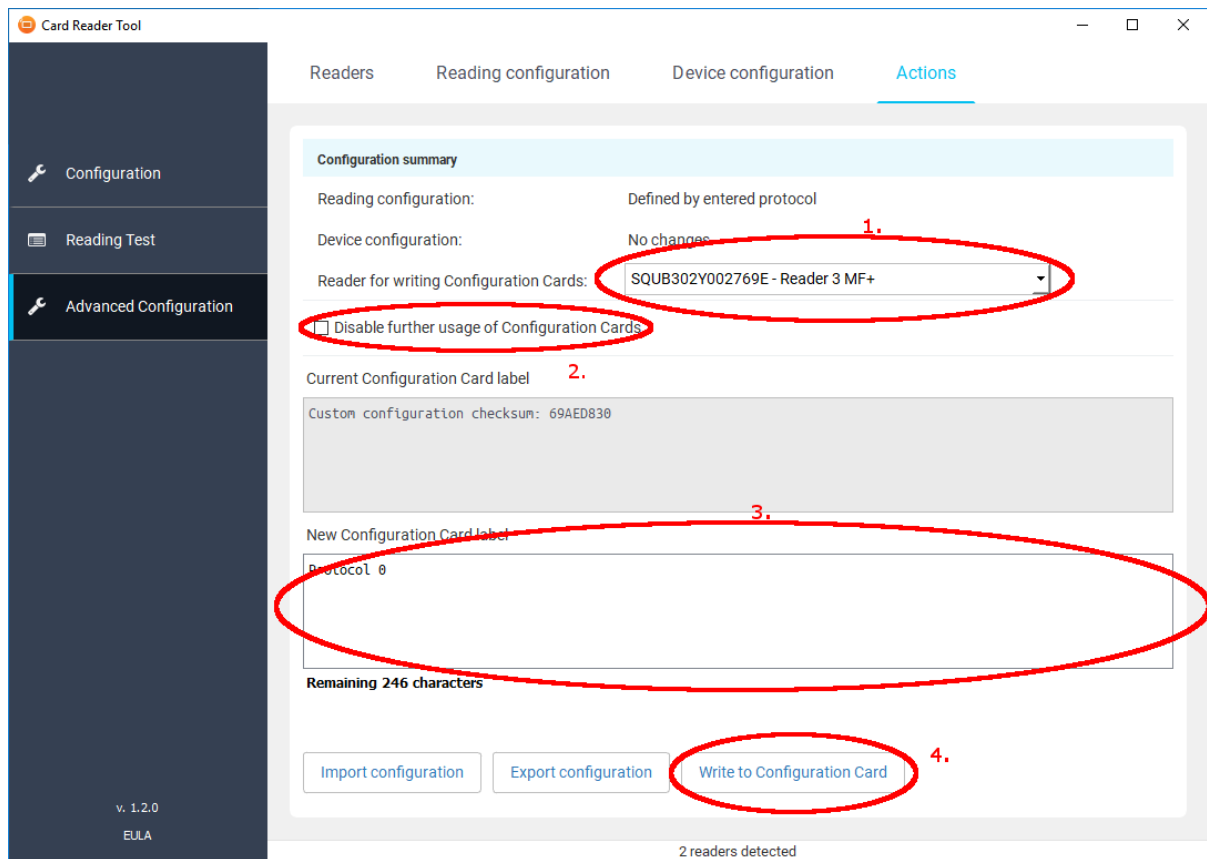


The card must have a default master key (like a blank card) before formatting. Formatting will erase all data stored on the card.

Step 2: Prepare reading configuration and device configuration (see above)

Step 3: Write the configuration onto a card

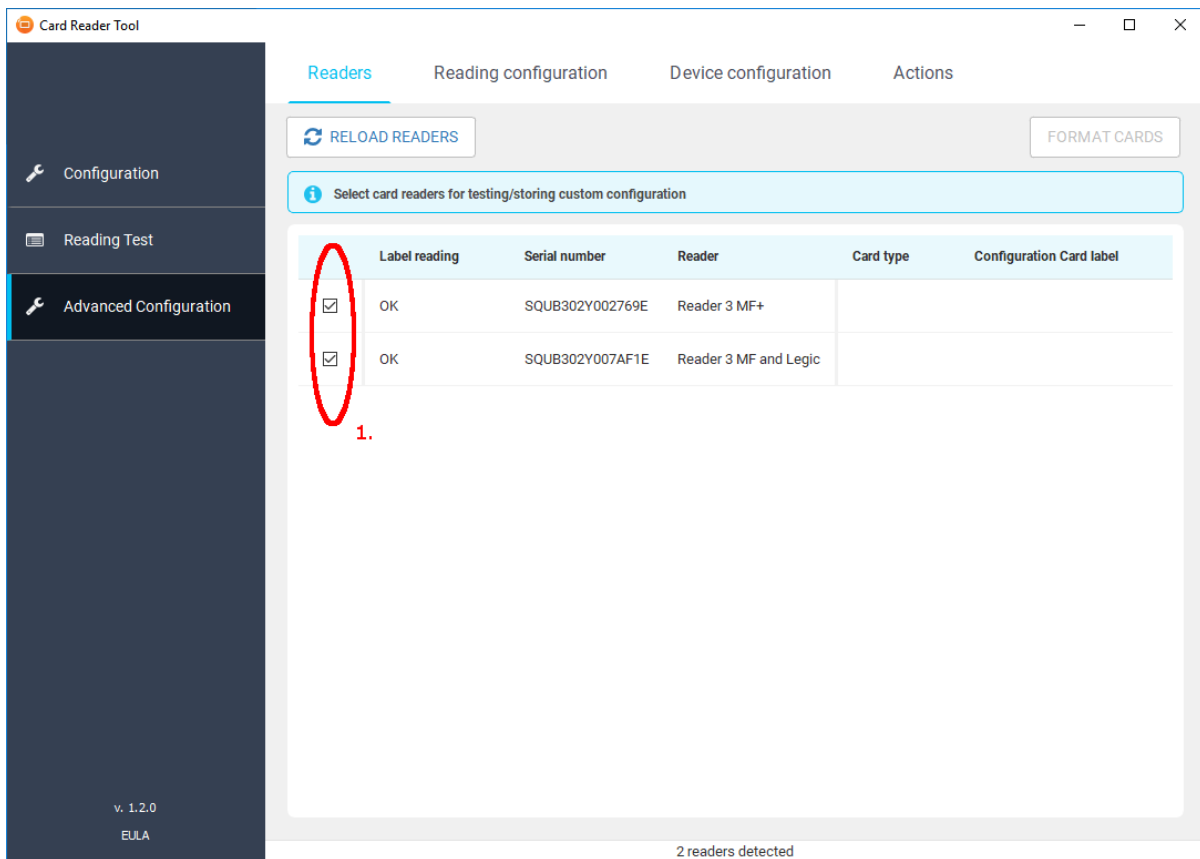
1. Select the reader you wish to write the config card on and place the formatted config card on it.
2. Select if you want to disable the further processing of the configuration cards. Generally, it is not safe to leave it on. However, for debug and testing purposes, it is better to leave it enabled.
3. Edit the card label. The label is initially generated based on the configuration settings, however, it can be changed to anything to suit customer needs. The label can be read back by an NFC-enabled mobile phone or tablet.
4. Write the configuration card



5.6.3 STEP-BY-STEP CUSTOM CARD READER CONFIGURATION

Step 1:

Select the card readers you want the custom configuration tested on:

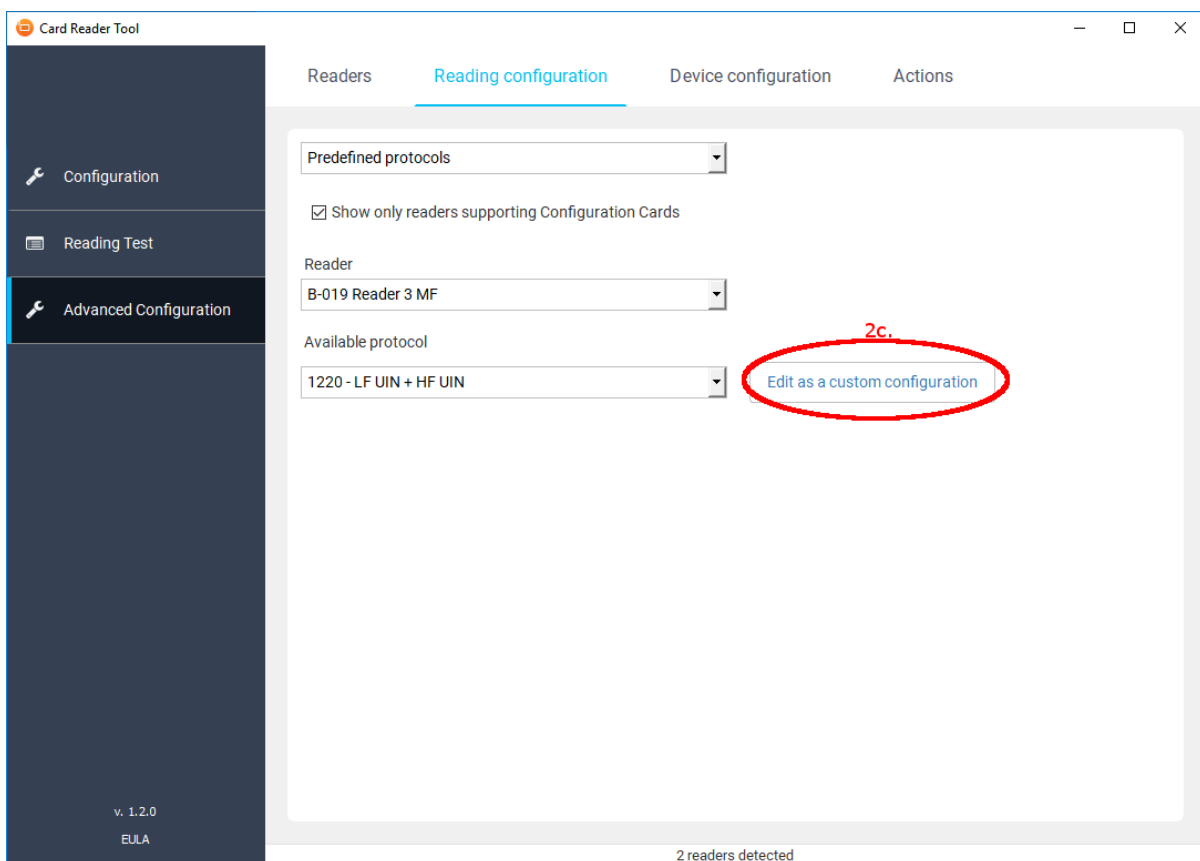
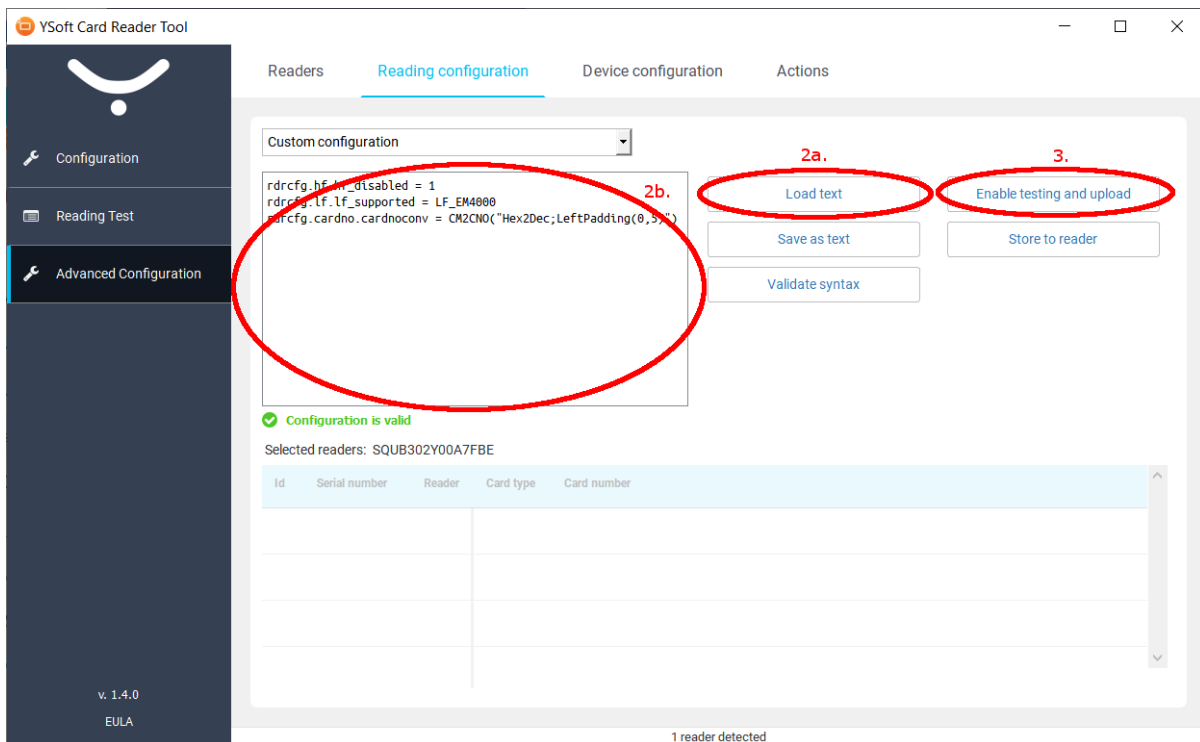


Step 2:

Make sure you have card readers selected.

- a) Load a custom configuration text file, or
- b) Edit the custom configuration in the text box, or
- c) You can use the "Edit as custom configuration" button on the selection of protocols.

See the "Card Reader Custom Configuration" document for details about what can be used as a custom configuration. Please ask your sales representative for the document.



Step 3: Click Enable testing and upload

Place the card and check the result on the list below. If you are not satisfied with the result and want to modify the custom configuration, just edit it and repeat step 3 – click the upload button.

5.6.4 USING CONFIGURATION CARD TO CONFIGURE Y SOFT USB CARD READER

Supported Card Readers

- Y Soft USB Card Reader 3 MF
- Y Soft USB Card Reader 3 MF+
- Y Soft USB Card Reader 3 MF&Legic
- Y Soft USB Card Reader 3 MF SAM
- Y Soft USB Card Reader 3 MFX
- Y Soft MFX Mobile Reader

Requirements

- Firmware version 2.4.0 or higher
- HF technology must be enabled either by protocol selection or in custom card reader configuration (enabled by default, not needed for firmware 2.5.1 or newer)
- Configuration cards processing is enabled in device configuration (enabled by default)
- Configuration cards work only within 2 minutes from the device power-up or restart

Put configuration card on a card reader



Wait 15 seconds

Card reader programming is indicated by blue LED color.



Successful programming is indicated by a green LED color and beep sound.

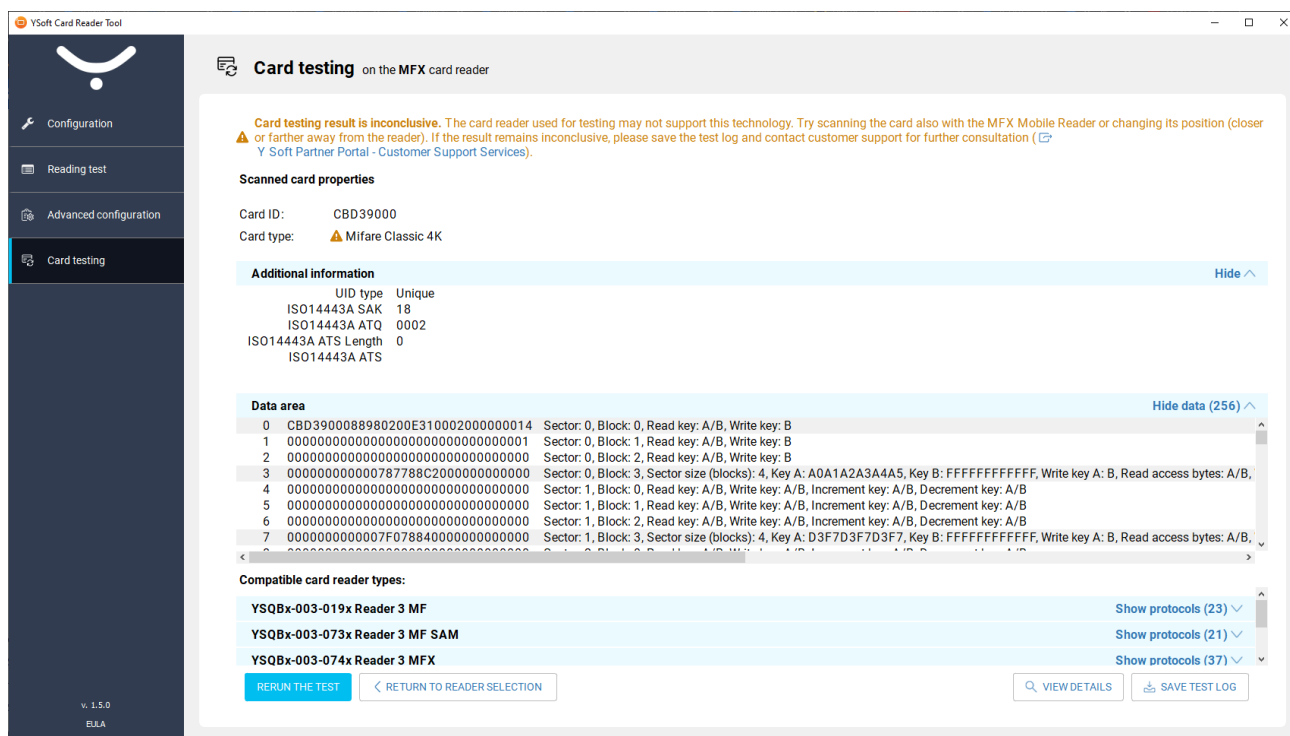
Remove configuration card

USB reader will reboot and use a new configuration.

Check that MFD and card reader are working properly. Some MFDs might need restart after card reader programming.



5.7 CARD TESTING



Starting with version 1.5.0 of YSoft Card Reader Tool, it is possible to perform detail card scanning to identify card type.

Card testing is supported on the following readers:

- Y Soft USB Card Reader 3 MFX
- Y Soft MFX Mobile Reader

Card testing results will provide this information:

- Card Unique ID
- Card type as reported on YSoft card readers
- Additional information about the chip type (when applicable)
- Content of freely readable data areas or information that the data area is not readable (depending on supported technology and permissions configured on the card)
- Compatible YSoft card readers and their protocols which are suitable to read identified card ID

Card testing supports multiple technologies detection. Multiple results can be also shown if the single chip can be read in different ways.



RESULTS PRODUCED BY THIS APPLICATION ARE INDICATIVE ONLY. Before placing an order, it is necessary to test that the recommended card reader and its configuration reads the data required by the customer. If you need help with card identification, please use free-of-charge Y Soft Card Testing Service. ([Y Soft Partner Portal](#) → [Customer Support Services](#) → [Card Testing](#))

5.7.1 A STEP-BY-STEP GUIDE FOR CARD TESTING

Testing cards checklist:

1. Get at least two card samples of each card type or each series of cards used by the customer. Two samples are recommended to make sure the card is not broken also it may be helpful in determining the card number conversion.
2. Verify that the cards are working with customer readers in a customer environment
3. If possible get card numbers as recorded in the customer's access or attendance system. This may be helpful in determining the proper card reader
4. If possible get the expected chip type from the customer

Testing equipment checklist:

1. A PC with Windows 7 and newer
2. The latest version of the Card Reader Tool application (Available at Y Soft Partner Portal)
3. Reader 3 MFX reader
4. MFX Mobile reader
5. Plastic spacer 20mm (3D printed) or non-metallic object (post-it notes etc.)

Having the cards tested on both card readers is necessary in order to check whether the technology is not directly supported on the other reader.

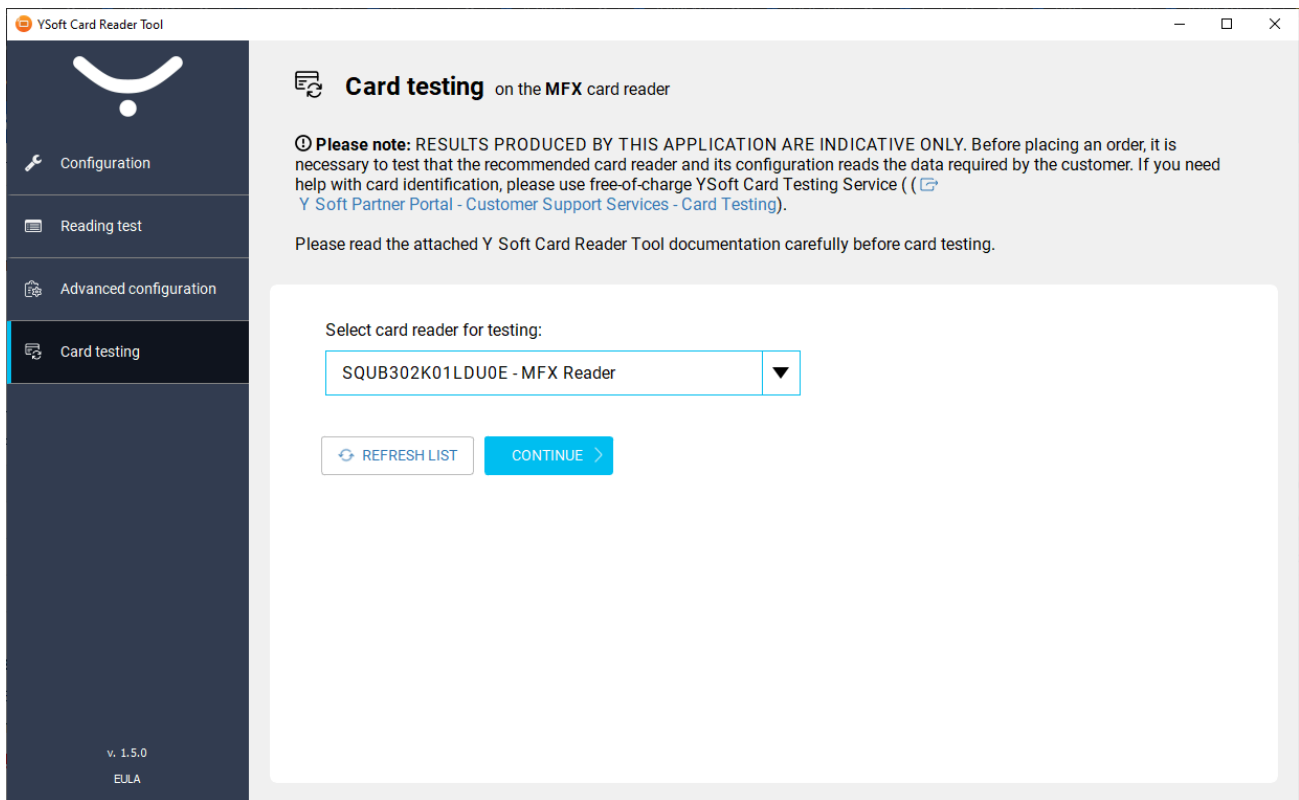
Reader 3 MFX card reader and MFX mobile have got the identical LF part (125kHz, 132kHz, 134.2KHz). The main difference is in the NFC/HF part (13.56MHz) supported technologies. See the following table:

	Reader 3 MFX	MFX Mobile reader
PACS data from HID iClass / SEOS credentials (HID SE Processor)	✓	✗ (SAM only)
Legic Prime / Legic Advant (Legic data reading)	✗	✓ (obsoleted Legic chips not supported)
Less common HF cards (Sielox, Calypso, etc.)	✓	✗

Step 1:

Select which reader you want to use as a testing reader. Only one reader could be used at a time.

1. Select reader



- Select the required reader from the drop-down menu and click the **Continue** button. When there is only one compatible reader then the reader will be already selected.
- You may refresh the list of connected readers using the **Refresh list** button on reader connect. List refresh may take a longer time.

2. No compatible reader detected

Select card reader for testing:

Please select...

No card readers compatible with card testing tool were found.
Please connect MFX or MFX Mobile reader first and try again.

↺
REFRESH LIST

CONTINUE >

Card testing is available only on supported readers. Please connect one of them. If there is already a reader present try to disconnect it, reconnect it, and refresh the reader list.

3. Incompatible firmware

Select card reader for testing:

SQUB302K01LDU0E - MFX Reader ▼

New firmware version is required for selected card reader.
Please update the firmware and try again.

REFRESH LIST

UPDATE FIRMWARE

Firmware in the selected reader is not compatible with the used version of YSoft Card Reader Tool. You can update it directly by clicking on the **Update firmware** button.

4. Incompatible tool

Select card reader for testing:

SQUB302K01LDU0E - MFX Reader ▼

Selected reader's firmware is not compatible with current version of this application.
Please **download and install** compatible version of Card Reader Tool and try again.

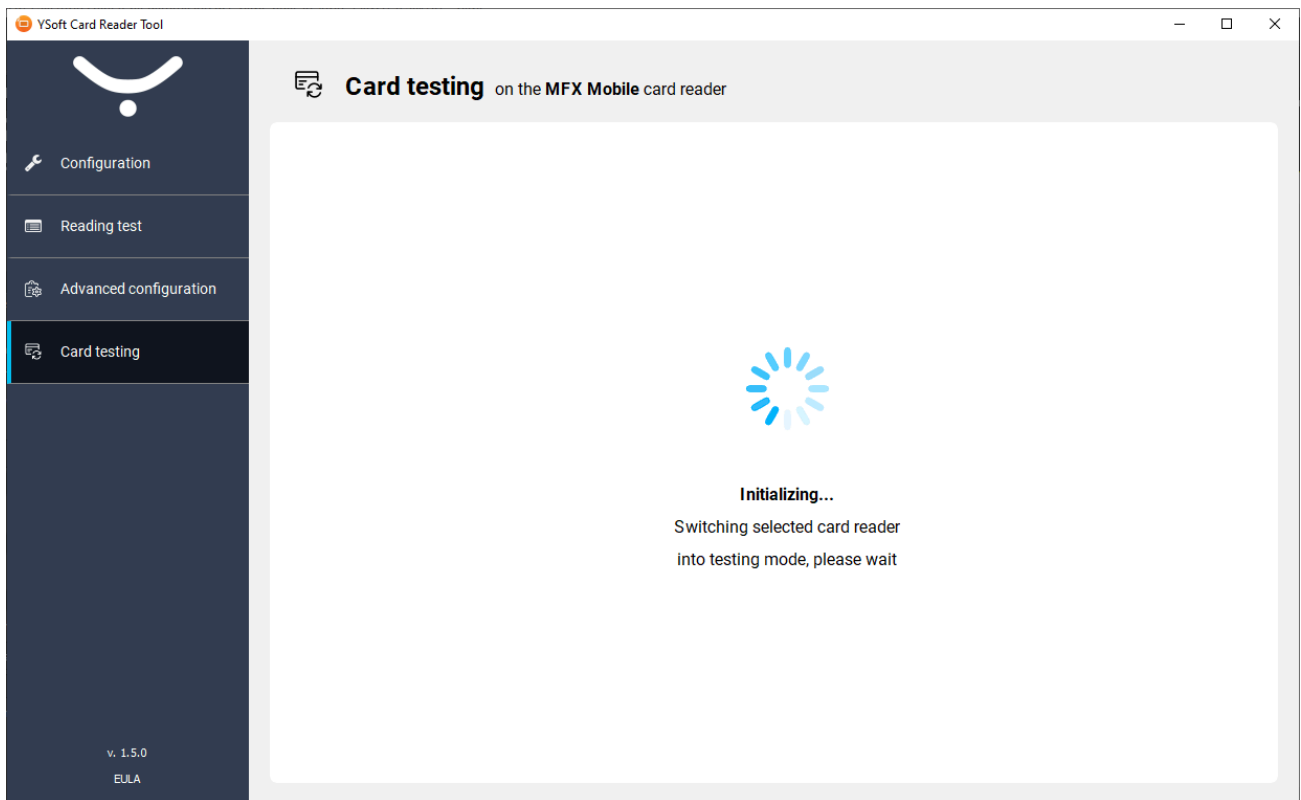
REFRESH LIST

CONTINUE >

Firmware in the selected reader is not compatible with the used version of YSoft Card Reader Tool. Firmware is newer so updating YSoft Card Reader Tool is needed. The download is available on Y Soft Partner Portal.

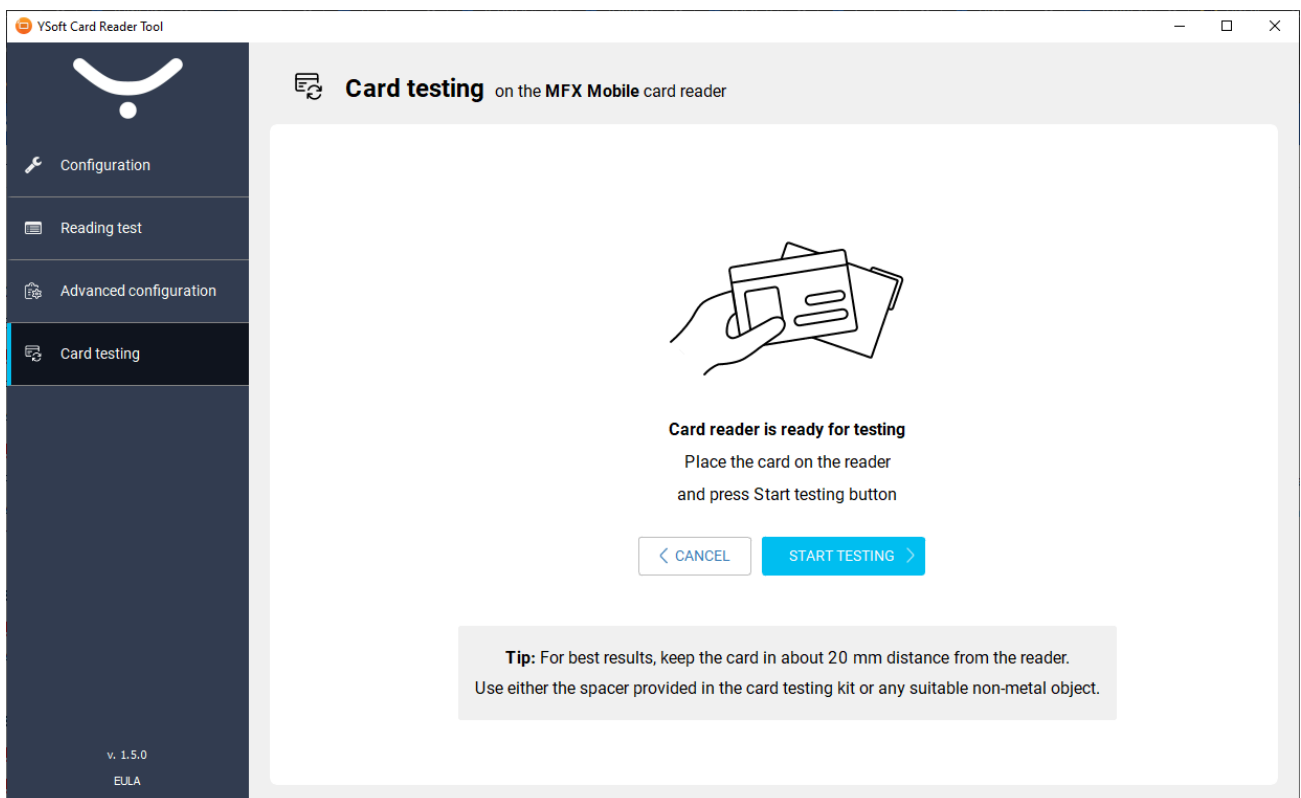
Another way how to solve version incompatibility is to downgrade reader firmware. This is however not recommended as the newer firmware generally contains bug fixes and new features. It may also not be available in some situations when the reader cannot be downgraded under its minimal firmware version.

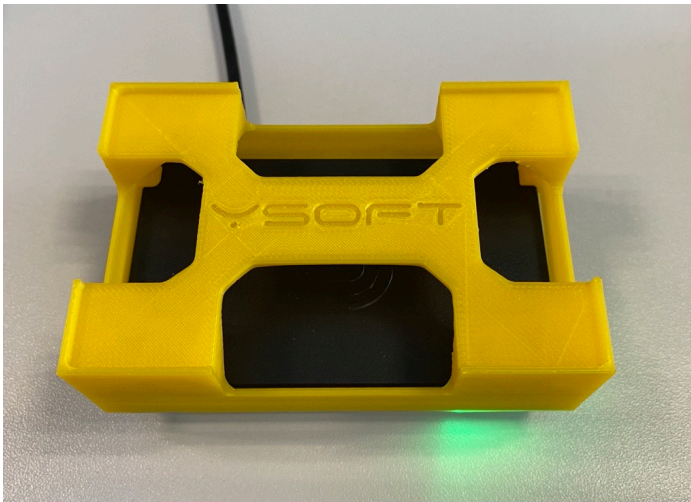
Step 2:



After selecting reader is initialized to testing mode, please wait until initialization is complete.

Step 3:





Now is time to place the card on the reader. For the best results, keep the card approximately 20mm from the card reader. You may use the 3D printed spacer provided with the card testing kit or any other non-metallic object (post-it notes, paper box, etc.).

Reasons for placing the card at a distance are the following:

1. While it is ok for just simple authentication to place a card directly at a reader, it may not provide stable results for a complete card scan at some card technologies.
2. To verify that the card will be read without any problems at this distance.

20mm seems to be ok for the vast majority of cards however in some cases, namely, with some contact less Smart Cards or special key fobs, it may be necessary to place the card closer as those require more energy.

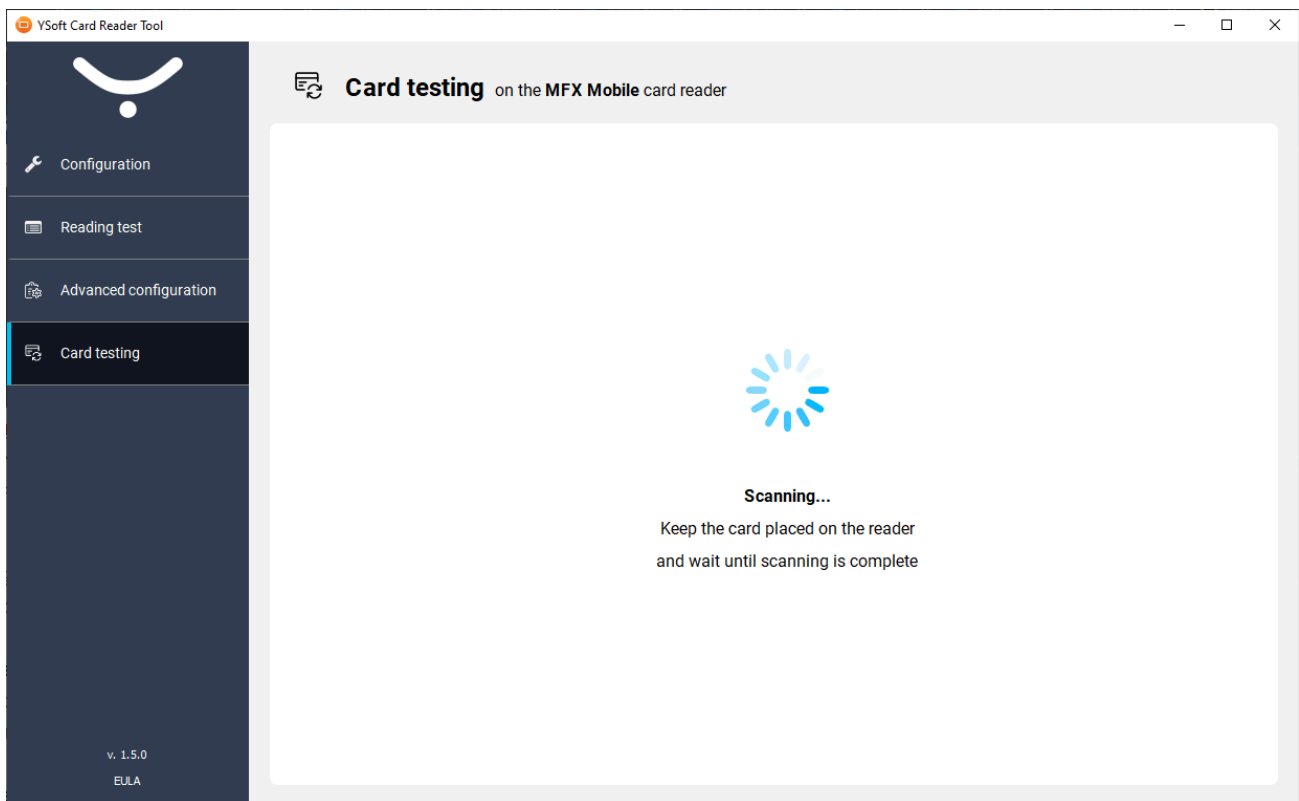
On the other hand, some cards (such as Cotag Active or Tiris) may require to be placed more than 20mm away

Step 4:

Click on **Start testing** to perform testing.

To return to reader selection and deactivating testing mode, click on the **Cancel** button.

Step 5:



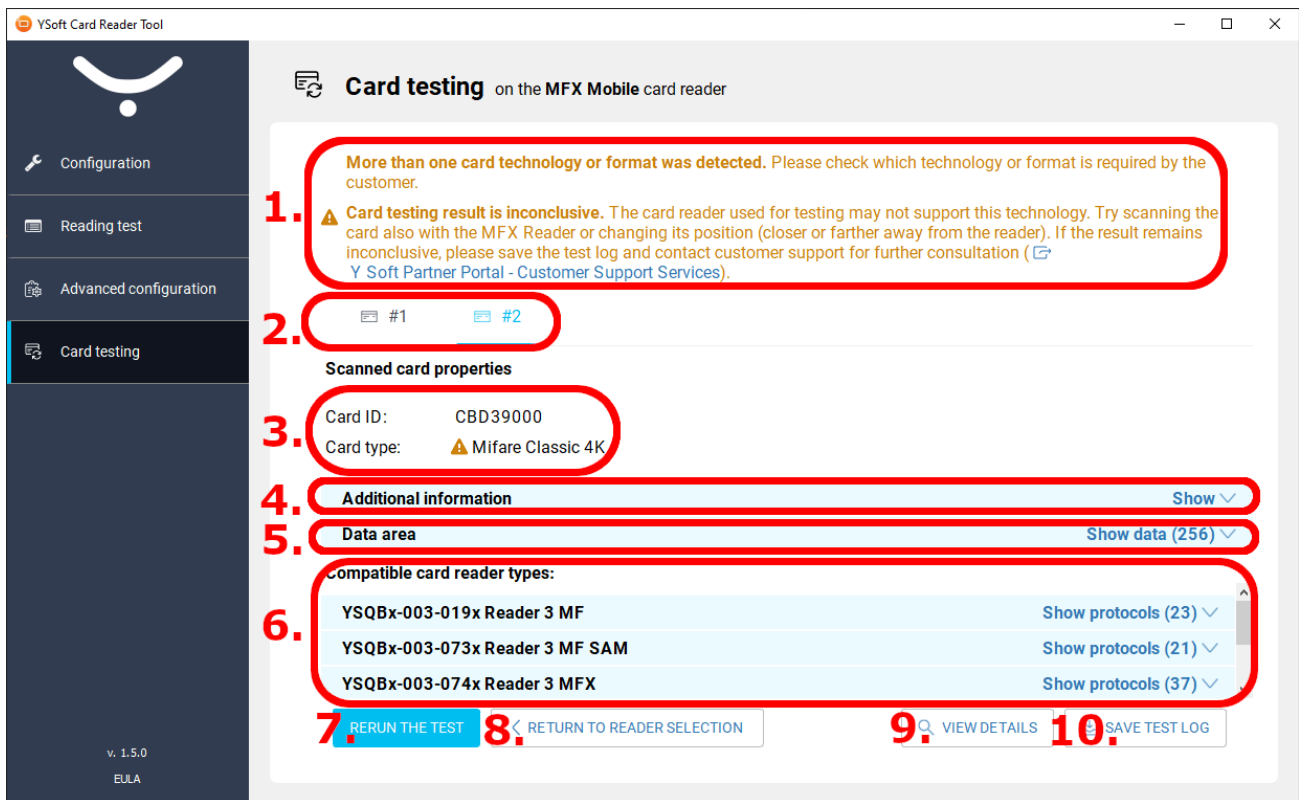
Keep the card on the reader without moving with it.

Scanning of a card may take a longer time depending on the card type and data area presence and size. Scanning is performed multiple times to make sure the results are stable.

Once card types are known or could not be determined, the screen moves to the results screen.

Step 6:

Result screen description



1. Testing status
2. Multiple technologies tabs – If multiple chips or multiple ways of reading the card have been detected
3. Scanned card properties – Card UIN and Card type
4. Additional information – Additional information about the card such as underlying standard information, bit lengths where applicable, hardware/software versions, etc.
5. Data areas – List of data, keys or applications detected, or other information that can be used to create a card reading customization
6. Compatible card readers and protocols – The protocols are listed in the order of preference. Single technology protocols are listed first, universal UIN protocols are listed next and remaining protocols that may also combine other technologies are after that.
7. Rerun the test – Run the same test again or test another card.
8. Return to reader selection – Select another reader, for example in case the result is inconclusive
9. View Details – View complete card scan JSON file, it may contain more information than displayed on the result screen.
10. Save test Log – Save the JSON file for further analysis by customer support.

5.7.2 CARD TESTING RESULTS

1. Unique card reader technology detected

✔ Scanning completed, 1 unique card technology was detected.

The technology has been uniquely identified and it cannot be anything else. However, validation with numbers requested by the customer is highly recommended as they might require some sort of card number conversion.

If unsure then save the test log and contact customer support for further consultation. Please see sending card testing consultations guide below.

2. Inconclusive results

⚠ **Card testing result is inconclusive.** The card reader used for testing may not support this technology. Try scanning the card also with the MFX Reader or changing its position (closer or farther away from the reader). If the result remains inconclusive, please save the test log and contact customer support for further consultation ([☞ Y Soft Partner Portal - Customer Support Services](#)).

The detected technology is known to be used as an underlying technology for other technologies. Usually, the card number may be stored on the card's memory and special settings or security keys are required.

1. Try another reader. If you tested the card on MFX then do the testing again on MFX Mobile and vice versa. This step is necessary is to check whether the technology is not directly supported by the other reader.
2. If the results are still inconclusive then validation with numbers requested by the customer is required.

If the results are still inconclusive then save the test log from both readers and contact customer support for further consultation. Please see sending card testing consultations guide below.

3. Multiple cards or formats detected

⚠ **More than one card technology or format was detected.** Please check which technology or format is required by the customer.

Multiple chips or ways of reading the chip are detected.

1. The card may be a composite card (also called a hybrid card), which is made up of a high-frequency card and a low-frequency card composite.
2. Card transmits UIN in multiple formats.

In this case, after consultation with the customer, it is necessary to select the card with the expected result.

4. Unknown card technology or unstable results detected

⚠ **Unknown card technology or unstable results detected.** Try scanning the card also with the MFX Mobile Reader or changing its position (closer or farther away from the reader). If you need more accurate results, please save the test log and contact customer support for further consultation ([☞ Y Soft Partner Portal - Customer Support Services](#)).

Something has been detected however it cannot be safely or directly used for user identification. The technology may use random UIN, the card cannot be properly handled by the reader or the data read is unreliable.

Try the other reader. If you tested the card on MFX then do the testing again on MFX Mobile and vice versa. You may also try repositioning the card - closer or farther from the reader.

If the results are still unknown then save the test log from both readers and contact customer support for further consultation. Please see sending card testing consultations guide below.

5. No card detected

▲ No card detected. Try scanning the card also with the MFX Reader or changing its position (closer to or farther from the reader). If it does not help, please use free-of-charge YSoft Card Testing Service ([Y Soft Partner Portal - Customer Support Services - Card Testing](#)).

No card has been detected.

Try the other reader. If you tested the card on MFX then do the testing again on MFX Mobile and vice versa. You may also try repositioning the card - closer or farther from the reader.

If it does not help then please use the card testing service ([Y Soft Partner Portal → Customer Support Services → Card Testing](#))

6. Unable to communicate with the card reader

▲ Unable to communicate with the card reader. Please make sure the card reader is connected properly and try again.

There has been an error in communicating with the reader. Please try reconnecting the reader and restart testing again. It may be also necessary to restart the Card Reader Tool.

If the problem persists then please contact customer support.

5.7.3 SENDING CARD TESTING CONSULTATIONS

You may enter the consultation request at [Partner Portal → Customer Support Services](#)

Consultation checklist:


1. For each card include testing logs from both MFX and MFX Mobile card readers
2. For each card include hi-resolution photos from both sides so that any technology or card number marking is visible. You may blur other personal information if needed.
3. For each card include information about the expected type and a card number expected by the customer.


5.7.4 QUICK TIPS: HOW TO GET THE BEST CARD TESTING RESULTS


- For the most accurate results, we strongly recommend using both MXF and MXF Mobile readers. Some card technologies are only partially supported by one reader and the other has direct full support.

- If the card test results are unexpected or unstable, change the reading distance - some cards require a specific field strength to fully read all features. For the most common cards, we recommend a 20mm 3d printed spacer.
- Compare result card number with customer expectations. Some cards require conversions or custom configuration to provide the expected results.

6 CARD READER CONFIGURATION PARAMETERS

 Not every configuration option is supported at every reader. Please see [Reader Configurations](#) for a list of currently supported card readers and their supported configurations.

 If the given configuration item is not specified then a default value is used. It is usually 0 but it may be also reader-specific, please see default values in [Reader Configurations](#).

 Some of the configuration items are mandatory depending on the others, check will be performed by the YSoft Card Reader Tool.

Formatting rules:

Configuration value formatting	Description	Usage example
(VALUE_A, VALUE_B)	One of values, values are enum fields	<code>rdrcfg.wiegand.wiegand_format = W27;</code>
(-4, 0, 4)	One of values, values are integers	<code>rdrcfg.ble.tx_power = 0;</code>
<0,127>	One value from min - max range (integers)	<code>rdrcfg.hf.legic_read.from_seg = 2;</code>
{VALUE_A, VALUE_B}	Values which could be combined into one value by binary OR	<code>rdrcfg.hf.desfire_read.desfire_fopts = DESFIRE_FILE_MACED DESFIRE_FILE_ENCRYPTED;</code>
[VALUE_A, VALUE_B]	Field which allows to assign multiple values	<code>rdrcfg.hf.hf_supported_multiple.extend ([HF_ICLASS, HF_ISO14443A]);</code>
u32	Any value from unsigned 32-bit integers	<code>rdrcfg.lf.lf_custom.start_cond = 0x000001FF;</code>
s32	Any value from signed 32-bit integers	<code>^</code>
uint	Any value from unsigned integers, width not important	<code>rdrcfg.hf.config_card_allow_time = 200;</code>
int	Any value from signed integers, width not important	<code>^</code>
data	Array of bytes, length not defined	<code>rdrcfg.sam.auth_key.key_data = STR([0x01, 0x02, 0x03, 0x04]);</code>
data[n]	Array of bytes, length of array is required to be n	<code>^</code>
data[n,m]	Array of bytes, array length must be from n to m bytes	<code>^</code>

6.1 COMMON PARAMETERS

Examples:

Read ISO14443A UIN and convert it to dec

```
rdrcfg.lf.lf_disabled = 1
rdrcfg.hf.hf_supported = HF_ISO14443A
rdrcfg.cardno.cardnoconv = CM2CNO("Hex2Dec")
```

```
rdrcfg.lf.lf_supported = LF_EM4000;
rdrcfg.common.cardno_conv = CARDCONV_HEX2DEC;
```

Supported parameters:

Supported cfg items	Supported values	Note
rdrcfg.common.cardno_conv	(CARDCONV_APPENDLRC, CARDCONV_HEX2DEC, CARDCONV_HEX2DEC10, CARDCONV_HEX2DECLEFT16, CARDCONV_INTERFLEX, CARDCONV_ISO14443ATRUNCREVENDIAN, CARDCONV_REVENDIAN, CARDCONV_REVENDIAN2DEC, CARDCONV_REVENDIANHEX2DEC10, CARDCONV_REVENDIANWOHIDPROX)	Builtin card number conversions. Will be overwritten if rdrcfg.cardno.cardnoconv specified
rdrcfg.cardno.cardnoconv		Custom specified card number conversion. CM2CNO("card manager") or CNOASM(FILE("filename")).
rdrcfg.common.continuous	(0, 1)	Special setting that enables card requested for login and card remove notification on some products.
rdrcfg.common.testing_mode	(0, 1)	Special testing setting to scan all supported technologies. Only for debug or testing purposes, DO NOT USE IN PRODUCTION ENVIRONMENT, some limitations may apply.
rdrcfg.common.exact_cardtype	(0, 1)	Enable returning exact card type at readers or settings where applicable.
rdrcfg.common.cardno_conv		Description
CARDCONV_APPENDLRC		Append one byte checksum (Xor) over all bytes of card number.
CARDCONV_HEX2DEC		Convert from Hex to Dec

rdrcfg.common.cardno_conv	Description
CARDCONV_HEX2DEC10	Convert from Hex to Dec, expanded to 10 decimal places, prepended by 0.
CARDCONV_HEX2DECLEFT16	Convert from Hex to Dec, left 16 characters only.
CARDCONV_INTERFLEX	Read data segment from Legic Interflex cards
CARDCONV_ISO14443ATRUNCREVENDIAN	On ISO14443A cards, reverse byte order and return only cascade level 1 UIN
CARDCONV_REVENDIAN	Reverse byte order
CARDCONV_REVENDIAN2DEC	Reverse byte order and convert to decimal
CARDCONV_REVENDIANHEX2DEC10	Reverse byte order, convert to decimal, expand to 10 decimal places
CARDCONV_REVENDIANWOHIDPROX	Reverse byte order, except HID Prox cards
rdrcfg.cardno.cardnoconv	Description
CM2CNO("card manager")	Card manager as specified in Card Manager Conversion Rules for Card Readers
CNOASM(FILE("filename"))	Card number conversion file as specified in LibReader cardno processing unit (legacy)

6.2 HF READER PARAMETERS

Examples

ISO14443A only reading

```
rdrcfg.hf.hf_supported = HF_ISO14443A;
```

ISO14443A and HID iClass UIN reading

```
rdrcfg.hf.hf_supported_multiple.extend([ HF_ICLASS, HF_ISO14443A ]);
```

Supported parameters

Supported cfg items	Supported values	Note
rdrcfg.hf.hf_disabled	(0, 1)	Completely disable processing of HF technologies
rdrcfg.common.no_uin_check	(0, 1)	Do not check if the card UIN is still the same. Required for cards with random UIN

Supported cfg items	Supported values	Note
rdrcfg.hf.hf_supported	(HF_DEFAULT, HF_LEGIC_PRIME, HF_LEGIC_ADVANT, HF_LEGIC_ADVANT_ISO14443A, HF_LEGIC_ADVANT_ISO1693, HF_ISO14443A, HF_ISO14443B, HF_ISO15693, HF_INSIDE, HF_FELICA, HF_SIELOX)	
rdrcfg.hf.hf_supported_multiple	[HF_DEFAULT, HF_LEGIC_PRIME, HF_LEGIC_ADVANT, HF_LEGIC_ADVANT_ISO14443A, HF_LEGIC_ADVANT_ISO1693, HF_ISO14443A, HF_ISO14443B, HF_ISO15693, HF_INSIDE, HF_FELICA, HF_SIELOX]	Multiple values supported
rdrcfg.hf.hf_uin_revendian	(0, 1)	Reverse endianness of UIN
rdrcfg.hf.report_read_error	(REP_NONE, REP_AUTO, REP_ERROR, REP_REFUSED)	Change default behaviour of error reporting if the card is not read correctly. By default the card is ignored but the reader may be configured to issue card read error or card refused messages.
rdrcfg.hf.hf_iso_select	(0, 1)	ISO14443A/B-4 ISO select. Required for DESFire/ISO14443A/B-4 commands.
rdrcfg.hf.hf_uin_mode	(HF_UIN_AUTO, HF_UIN_NORMAL, HF_UIN_LEGIC)	Configuration of how UIN should be reported. Some readers may have a different endianness on some technologies.
rdrcfg.hf.read_error_return_uin	(0, 1)	Any error in reading data areas will result in returning the card UIN
rdrcfg.hf.config_card_disabled	(0, 1)	Disable processing of further YSoft configuration cards
rdrcfg.hf.config_card_process_time	int	Time in msec for which the config card will be checked for presence before processing, default 8000
rdrcfg.hf.config_card_allow_time	int	Time in sec after startup in which configuration card is allowed, -1 means always enabled, default 120
rdrcfg.other.piv_mode	(PIV_MODE_AUTO, PIV_MODE_FASCN, PIV_MODE_GUID)	Specify mode of reading of PIV-II cards

6.3 MIFARE CLASSIC (1K/4K) PARAMETERS

⚠ Always use `rdrcfg.lf.lf_disabled = 1`; for disabling LF technologies on combined multireaders. LF technologies are not secure and leaving it enabled presents a security risk.

ℹ If Mifare Classic reading is used in custom card reader configuration then every detected RFID card is scanned and searched for specified parameters. This is especially useful if the card used contains dual chip or the card is carried along with other cards (for example with traffic cards or bank cards in a purse).

Example

Mifare classic 1k/4k reading

```
rdrcfg.lf.lf_disabled = 1;
rdrcfg.hf.hf_supported = HF_ISO14443A;
rdrcfg.hf.mifare_read.mode = MIFARE_CLASSIC;
rdrcfg.hf.mifare_read.mifare_block = 0;
rdrcfg.hf.mifare_read.mifare_key.key_type = KEY_MIFARE_A;
rdrcfg.hf.mifare_read.mifare_key.key_data = STR([ 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF ]);
```

Supported configurations

Supported cfg items	Supported values	Note
<code>rdrcfg.hf.mifare_read.mode</code>	(MIFARE_CLASSIC)	
<code>rdrcfg.hf.mifare_read.block</code>	<0,240>	0-63 (Mifare 1k), 0-240 (Mifare 4k)
<code>rdrcfg.hf.mifare_read.mifare_key.key_type</code>	(KEY_MIFARE_A, KEY_MIFARE_B)	
<code>rdrcfg.hf.mifare_read.mifare_key.key_data</code>	data[6]	key data

6.3.1 MIFARE CARDS STRUCTURE

Mifare 1k internal structure:

Sector	Block	Byte Number within a Block																Description
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
15	3	Key A				Access Bits				Key B								Sector Trailer 15
	2																	Data
	1																	Data
	0																	Data
14	3	Key A				Access Bits				Key B								Sector Trailer 14
	2																	Data
	1																	Data
	0																	Data
:	:																	
:	:																	
:	:																	
1	3	Key A				Access Bits				Key B								Sector Trailer 1
	2																	Data
	1																	Data
	0																	Data
0	3	Key A				Access Bits				Key B								Sector Trailer 0
	2																	Data
	1																	Data
	0	Manufacturer Data																Manufacturer Block

Mifare 4k internal structure:

Sector	Block	Byte Number within a Block																Description
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
39	15	Key A				Access Bits				Key B								Sector Trailer 39
	14																	Data
	13																	Data
	:																	:
	:																	:
	2																	Data
	1																	Data
	0																	Data
:	:																	:
:	:																	:
:	:																	:
32	15	Key A				Access Bits				Key B								Sector Trailer 32
	14																	Data
	13																	Data
	:																	:
	:																	:
	2																	Data
	1																	Data
	0																	Data
31	3	Key A				Access Bits				Key B								Sector Trailer 31
	2																	Data
	1																	Data
	0																	Data
:	:																	:
:	:																	:
:	:																	:
0	3	Key A				Access Bits				Key B								Sector Trailer 0
	2																	Data
	1																	Data
	0	Manufacturer Data																Manufacturer Block

Sector Trailer organization:

Byte Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Description	Key A						Access Bits				Key B (optional)					

Data access bits description:

Data block access bit rules (blocks 0-2)

Access bits			Access condition				Application
C1	C2	C3	Read	Write	Increment	Dec/Trans/Rest	
0	0	0	key A B	key A B	key A B	key A B	transport
0	1	0	key A B	never	never	never	read/write block
1	0	0	key A B	key B	never	never	read/write block
1	1	0	key A B	key B	key B	key A B	value block
0	0	1	key A B	never	never	key A B	value block
0	1	1	key B	key B	never	never	read/write block
1	0	1	key B	never	never	never	read/write block
1	1	1	never	never	never	never	read/write block

Sector trailer access bit rules (block 3)

Access bits			Access condition for						Remarks
			KEY A		Access bits		Key B		
C1	C2	C3	Read	Write	Read	Write	Read	Write	
0	0	0	never	key A	key A	never	key A	key A	Key A is able to read key B Key A is able to read key B
0	1	0	never	never	key A	never	key A	never	
1	0	0	never	key B	key A B	never	never	key B	
1	1	0	never	never	key A B	never	never	never	
0	0	1	never	key A	key A	key A	key A	key A	Key A is able to read key B
0	1	1	never	key B	key A B	key B	never	key B	
1	0	1	never	never	key A B	key B	never	never	
1	1	1	never	never	key A B	never	never	never	

f

6.4 MIFARE DESFIRE (0.6, EV1, EV2) PARAMETERS



Always use `rdrcfg.If.If_disabled = 1`; for disabling LF technologies on combined multireaders. LF technologies are not secure and leaving it enabled presents a security risk.



If Mifare DESFire reading is used in custom card reader configuration then every detected RFID card is scanned and searched for specified parameters. This is especially useful if the card used contains dual chip or the card is carried along with other cards (for example with traffic cards or bank cards in a purse).

Example

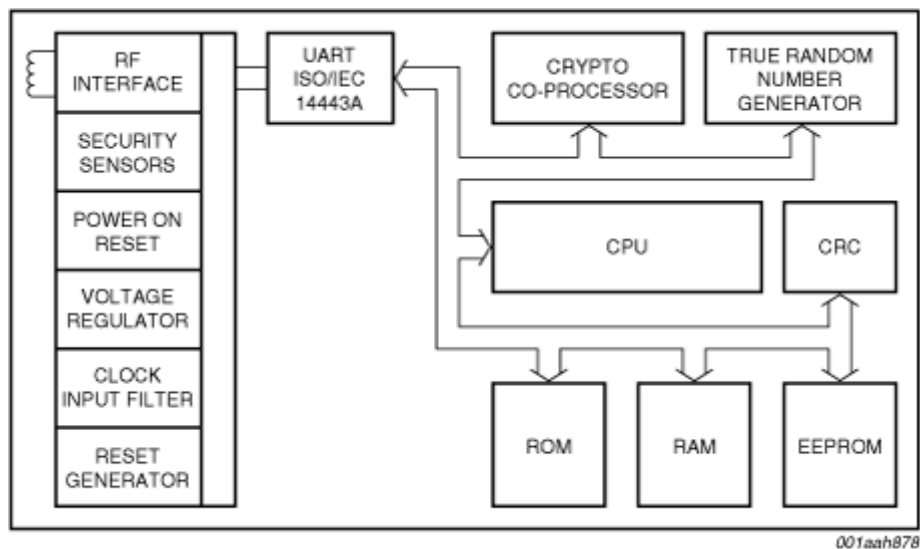
Mifare DESFire / Mifare DESFire EV1 reading

```
rdrcfg.lf.lf_disabled = 1;
rdrcfg.hf.hf_supported = HF_ISO14443A;
rdrcfg.hf.hf_iso_select = 1;
rdrcfg.hf.desfire_read.desfire_aid = STR ([ 0x00, 0x00, 0x01 ]);
rdrcfg.hf.desfire_read.desfire_fid = 0x00;
rdrcfg.hf.desfire_read.desfire_ffset = 0;
rdrcfg.hf.desfire_read.desfire_flen = 0;
rdrcfg.hf.desfire_read.desfire_fopts = DESFIRE_FILE_AUTO;
rdrcfg.hf.desfire_read.desfire_key_no = 1;
rdrcfg.hf.desfire_read.desfire_key.key_type = KEY_2K3DES;
rdrcfg.hf.desfire_read.desfire_key.key_data = STR([ 0x01, 0x23, 0x45, 0x67, 0x89, 0xAB, 0xCD, 0xEF, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00 ]);
rdrcfg.cardno.cardnoconv = CM2CNO("Substring(6)")
```

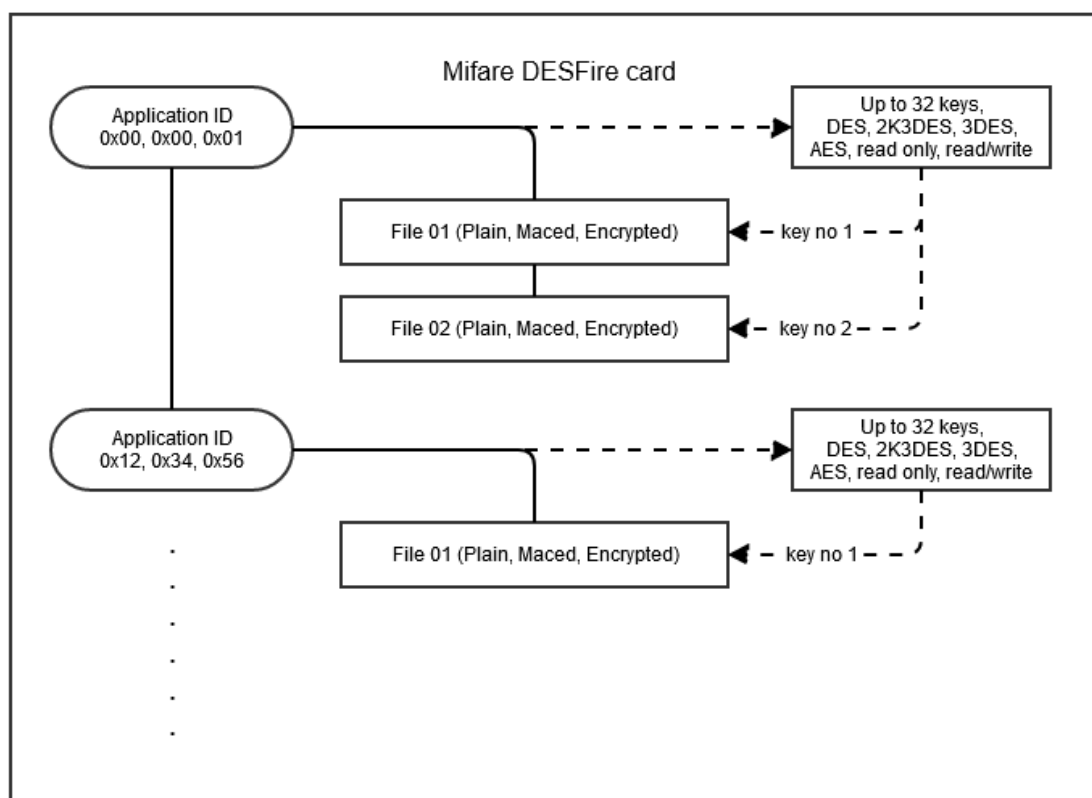
Supported parameters

Supported cfg items	Supported values	Note
rdrcfg.hf.hf_iso_select	(0, 1)	ISO14443A/B-4 ISO select. Required for DESFire/ISO14443A/ B-4 commands.
rdrcfg.hf.desfire_read.desfire_aid	data[3]	3 byte DESFire Application ID in endiannes reported from card
rdrcfg.hf.desfire_read.desfire_fid	<0,15>	file id
rdrcfg.hf.desfire_read.desfire_ffset	<0,65535>	offset in file
rdrcfg.hf.desfire_read.desfire_flen	<0,230>	0 returns complete file up to 230 bytes
rdrcfg.hf.desfire_read.desfire_fopts	{DESFIRE_FILE_AUTO, DESFIRE_FILE_PLAIN, DESFIRE_FILE_MACED, DESFIRE_FILE_ENCRYPTED, DESFIRE_FILE_STD, DESFIRE_FILE_VALUE, DESFIRE_FILE_RECORD}	
rdrcfg.hf.desfire_read.desfire_key_no	<0,15>	
rdrcfg.hf.desfire_read.desfire_key.key_type	(KEY_DES, KEY_2K3DES, KEY_3K3DES, KEY_AES128)	
rdrcfg.hf.desfire_read.desfire_key.key_data	data[8,24]	key data

6.4.1 MIFARE DESFIRE INTERNAL DIAGRAM



6.4.2 MIFARE DESFIRE FILE STRUCTURE



6.5 MIFARE SAM AV2 PARAMETERS



Always use `rdrcfg.lf.lf_disabled = 1`; for disabling LF technologies on combined multireaders. LF technologies are not secure and leaving it enabled presents a security risk.



If Mifare Classic/DESFire reading is used in custom card reader configuration together with SAM AV2 then every detected RFID card is scanned and searched for specified parameters. This is especially useful if the card used contains dual chip or the card is carried along with other cards (for example with traffic cards or bank cards in a purse).

Examples

Reading of Mifare classic card using SAM

```
rdrcfg.lf.lf_disabled = 1;
rdrcfg.hf.hf_supported = HF_ISO14443A;
rdrcfg.hf.mifare_read.mode = MIFARE_CLASSIC;
rdrcfg.hf.mifare_read.mifare_block = 0;
rdrcfg.hf.mifare_read.mifare_key.key_type = KEY_MIFARE_A;
rdrcfg.hf.mifare_read.mifare_key.sam_key_no = 0x1B;
rdrcfg.hf.mifare_read.mifare_key.key_version = 0x00;
rdrcfg.sam.sam_mode = SAM_MIFAREAV2;
# The following is required only if the SAM card needs to be authenticated before use
rdrcfg.sam.sam_auth = SAM_AV2_AUTH_PLAIN;
rdrcfg.sam.auth_key_num = 0;
rdrcfg.sam.auth_key.key_type = KEY_AES192;
rdrcfg.sam.auth_key.key_version = 0;
rdrcfg.sam.auth_key.key_data = STR([0x01, 0x23, 0x45, 0x67, 0x89, 0xAB, 0xCD, 0xEF, 0x01, 0x23, 0x45, 0x67,
0x89, 0xAB, 0xCD, 0xEF, 0x01, 0x23, 0x45, 0x67, 0x89, 0xAB, 0xCD, 0xEF ])
```

Reading of Mifare DESFire using SAM


```
rdrcfg.lf.lf_disabled = 1;
rdrcfg.hf.hf_supported = HF_ISO14443A;
rdrcfg.hf.hf_iso_select = 1;
rdrcfg.hf.desfire_read.desfire_aid = STR ([ 0x00, 0x00, 0x04 ]);
rdrcfg.hf.desfire_read.desfire_fid = 0x02;
rdrcfg.hf.desfire_read.desfire_foffset = 0;
rdrcfg.hf.desfire_read.desfire_flen = 0;
rdrcfg.hf.desfire_read.desfire_fopts = DESFIRE_FILE_AUTO;
rdrcfg.hf.desfire_read.desfire_key_no = 1;
rdrcfg.hf.desfire_read.desfire_key.key_type = KEY_AES128;
rdrcfg.hf.desfire_read.desfire_key.sam_key_no = 0x17;
rdrcfg.hf.desfire_read.desfire_key.key_version = 0x00;
rdrcfg.hf.desfire_read.desfire_key.key_div = DIV_NONE;
rdrcfg.sam.sam_mode = SAM_MIFAREAV2;
# The following is required only if the SAM card needs to be authenticated before use
rdrcfg.sam.sam_auth = SAM_AV2_AUTH_PLAIN;
rdrcfg.sam.auth_key_num = 0;
rdrcfg.sam.auth_key.key_type = KEY_AES192;
rdrcfg.sam.auth_key.key_version = 0;
rdrcfg.sam.auth_key.key_data = STR([0x01, 0x23, 0x45, 0x67, 0x89, 0xAB, 0xCD, 0xEF, 0x01, 0x23, 0x45, 0x67,
0x89, 0xAB, 0xCD, 0xEF, 0x01, 0x23, 0x45, 0x67, 0x89, 0xAB, 0xCD, 0xEF ])
```


Supported parameters

Supported cfg items	Supported values	Note
rdrcfg.hf.mifare_read.mifare_key.sam_key_no	<0,127>	Authentication key number
rdrcfg.hf.mifare_read.mifare_key.key_version	<0,255>	Version of authentication key
rdrcfg.hf.mifare_read.mifare_key.key_div	(DIV_NONE, DIV_UIN)	Key diversification method
rdrcfg.hf.desfire_read.desfire_key.sam_key_no	<0,127>	Authentication key number
rdrcfg.hf.desfire_read.desfire_key.key_version	<0,255>	Version of authentication key
rdrcfg.hf.desfire_read.desfire_key.key_div	(DIV_NONE, DIV_UIN)	Key diversification method
rdrcfg.sam.sam_mode	(SAM_NONE, SAM_MIFAREAV2, SAM_HIDSEPROC, SAM_AUTO)	Type of SAM
rdrcfg.sam.sam_auth	(SAM_NO_AUTH, SAM_AV2_UNLOCK, SAM_AV2_AUTH_PLAIN)	Authentication mode of SAM
rdrcfg.sam.auth_key_num	<0,127>	Authentication key number
rdrcfg.sam.auth_key.key_type	(KEY_AES128, KEY_AES192)	Authentication key type
rdrcfg.sam.auth_key.key_version	<0,255>	Version of authentication key

Supported cfg items	Supported values	Note
rdrcfg.sam.auth_key.key_data	data[16,24]	key data

6.6 LEGIC PARAMETERS

 Always use `rdrcfg.lf.lf_disabled = 1`; for disabling LF technologies on combined multireaders. LF technologies are not secure and leaving it enabled presents a security risk.

 If Legic reading is used in custom card reader configuration then every detected RFID card is scanned and searched for specified parameters. This is especially useful if the card used contains dual chip or the card is carried along with other cards (for example with traffic cards or bank cards in a purse).

Examples

Legic data segment reading with stamp search on Legic Advant and Legic Prime cards

```
rdrcfg.lf.lf_disabled = 1
rdrcfg.hf.hf_supported_multiple.extend([ HF_LEGIC_PRIME, HF_LEGIC_ADVANT ])
rdrcfg.hf.legic_read.mode = LEGIC_DATA_SEG
rdrcfg.hf.legic_read.flags = LEGIC_FLAG_NONE
rdrcfg.hf.legic_read.search_string = STR([ 0x00, 0xAA ])
rdrcfg.hf.legic_read.read_len = 3
rdrcfg.hf.legic_read.read_offset = 0
rdrcfg.hf.legic_read.from_seg = 0
```

Legic KGH segment 1 reading

```
rdrcfg.lf.lf_disabled = 1;
rdrcfg.hf.legic_read.mode = LEGIC_KGH;
rdrcfg.hf.legic_read.flags = LEGIC_KGH_SHORT;
rdrcfg.hf.legic_read.from_seg = 1;
```

Legic KGH segment reading with stamp search

```
rdrcfg.lf.lf_disabled = 1;
rdrcfg.hf.legic_read.mode = LEGIC_KGH;
rdrcfg.hf.legic_read.flags = LEGIC_KGH_SHORT;
rdrcfg.hf.legic_read.search_string = STR([ 0x00, 0xAA, 0x11, 0x22 ]);
rdrcfg.hf.legic_read.from_seg = 0;
```

Legic Advant Access segment reading with stamp search

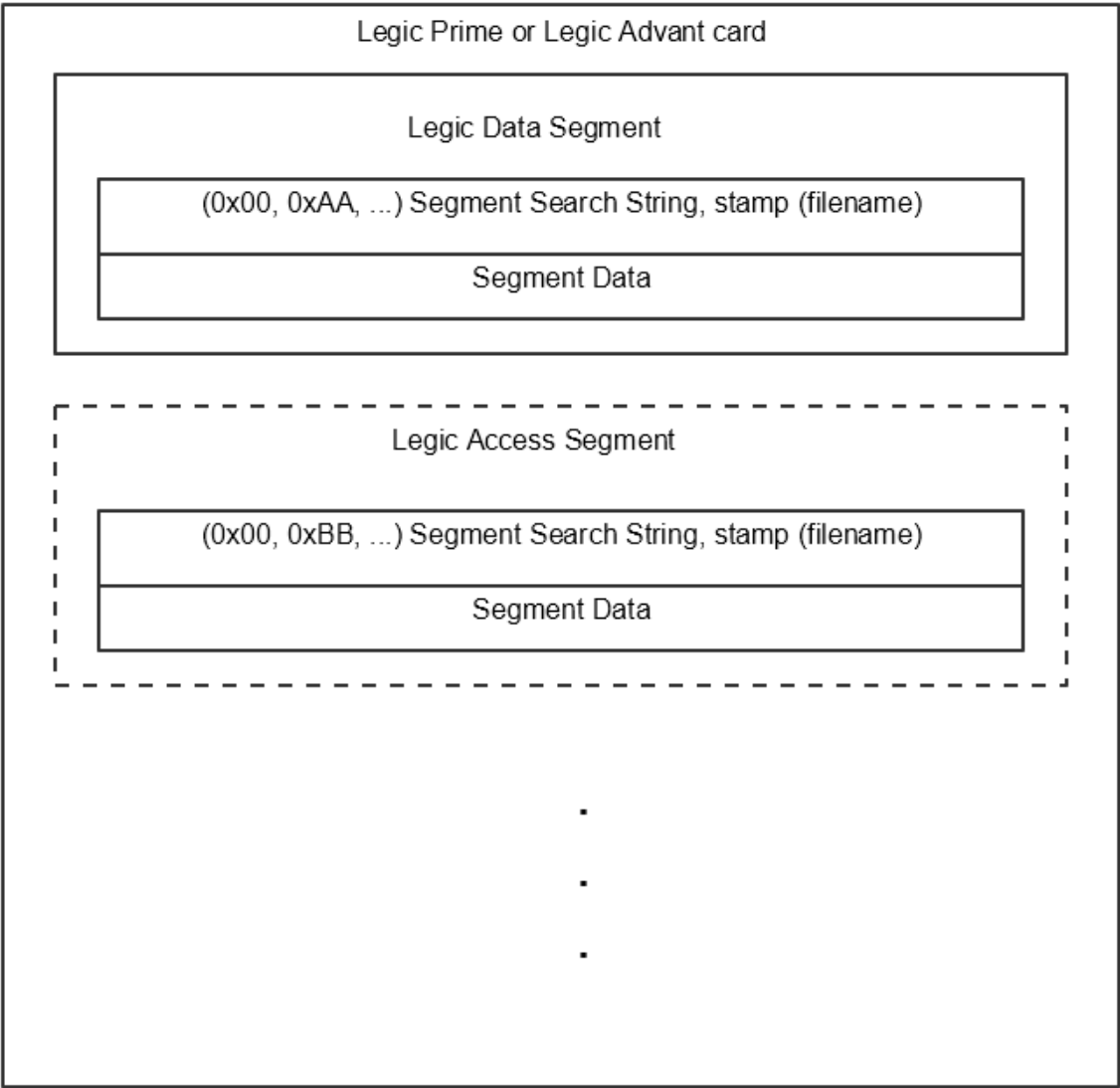
```
rdrcfg.lf.disabled = 1;
rdrcfg.hf.hf_supported = HF_LEGIC_ADVANT;
rdrcfg.hf.legic_read.mode = LEGIC_ACCESS;
rdrcfg.hf.legic_read.flags = LEGIC_ACCESS_ID;
rdrcfg.hf.legic_read.search_string = STR([ 0x00, 0x11, 0x22 ]);
rdrcfg.hf.legic_read.from_seg = 0;
```

Supported parameters

Supported cfg items	Supported values	Note
rdrcfg.hf.legic_read.mode	(LEGIC_DATA_SEG, LEGIC_KGH, LEGIC_ACCESS)	
rdrcfg.hf.legic_read.flags	{LEGIC_FLAG_NONE, LEGIC_KGH_SHORT, LEGIC_KGH_OLD, LEGIC_KGH_COMPATV2, LEGIC_ACCESS_STAMP, LEGIC_ACCESS_ID, LEGIC_ACCESS_USER}	OR combinations possible
rdrcfg.hf.legic_read.from_seg	<1,128>	
rdrcfg.hf.legic_read.read_len	<1,199>	
rdrcfg.hf.legic_read.read_offset	<0,65535>	
rdrcfg.hf.legic_read.search_string	data[1,12]	1 to 12 bytes of segment stamp
rdrcfg.hf.legic.legic_prime_disabled	(0, 1)	Disable processing of Legic Prime cards
rdrcfg.hf.legic.legic_advant_disabled	(0, 1)	Disable processing of Legic Advant cards
rdrcfg.hf.legic.enabled_cards_multiple	[CARD_LEGICMIM22, CARD_LEGICMIM256, CARD_LEGICMIM1024, CARD_LEGICATC128_MV210, CARD_LEGICATC256_MV210, CARD_LEGICATC512_MP110, CARD_LEGICATC1024_MV110, CARD_LEGICATC2048_MP110, CARD_LEGICATC4096_MP310, CARD_LEGICATC4096_MP311, CARD_LEGICATC4096_MP312, CARD_LEGICAFS4096_JP1X, CARD_LEGICATC1024_MV010, CARD_LEGICATC256_MV410, CARD_LEGICCTC4096_MP410, CARD_LEGICCTC4096_MM410]	List of Legic card types that are allowed

Supported cfg items	Supported values	Note
rdrcfg.hf.legic.disabled_cards_multiple	[CARD_LEGICMIM22, CARD_LEGICMIM256, CARD_LEGICMIM1024, CARD_LEGICATC128_MV210, CARD_LEGICATC256_MV210, CARD_LEGICATC512_MP110, CARD_LEGICATC1024_MV110, CARD_LEGICATC2048_MP110, CARD_LEGICATC4096_MP310, CARD_LEGICATC4096_MP311, CARD_LEGICATC4096_MP312, CARD_LEGICAFS4096_JP1X, CARD_LEGICATC1024_MV010, CARD_LEGICATC256_MV410, CARD_LEGICCTC4096_MP410, CARD_LEGICCTC4096_MM410]	List of Legic card types that should be rejected
rdrcfg.hf.legic.clear_launch_records	(0, 1)	Clear all launch records
rdrcfg.hf.legic.clear_user_keys	(0, 1)	Clear all user specified keys. Will not clear factory keys
rdrcfg.hf.legic.launch_media_disabled	(0, 1)	Disable launch/delaunch media processing

6.6.1 LEGIC FILE STRUCTURE



6.7 LEGIC CONNECT PARAMETERS



Always use `rdrcfg.If.If_disabled = 1`; for disabling LF technologies on combined multireaders. LF technologies are not secure and leaving it enabled presents a security risk.



Legic Connect is secure cloud based ecosystem with secure delivery to mobile applications based on Legic Connect SDKs (LC app). Single mobile device may hold multiple LC apps (each identified by it's application ID). Each LC mobile app may hold files from multiple projects.

❗ Legic Neon file stands for file in Mobile applications (based on Legic Connect SDKs). Application may hold multiple Neon files (each identified by project ID and file ID). Each Neon file may hold some data like credential ID (Virtual card number), may differ in length, requires authentication and encryption key for access.

❗ Legic Orbit is secure cloud based delivery of keys (for neon files) to Legic based readers via mobile applications based on Legic Connect SDKs. Configuration is performed by delivering Versatile Configuration Package (VCP).

❗ Keys for Neon files may be defined by user or generated by Legic Trusted Service. When keys are defined by customer then keys may be delivered to reader by VCP or reader configuration. When keys are generated by Legic, then the only way to deliver them to reader is by VCP.

Examples

Reading Printer Neon file from Demo EKA application (over BLE or NFC)

```
rdrcfg.lf.lf_disabled = 1;
rdrcfg.hf.hf_supported = HF_ISO14443A;
rdrcfg.ble.min_rssi = -80;
rdrcfg.ble.tx_power = -40;
rdrcfg.ble.central_role_enabled = 1;
rdrcfg.ble.peripheral_role_enabled = 1;
rdrcfg.legic_connect.process_over_ble_central_enabled = 1;
rdrcfg.legic_connect.process_over_ble_peripheral_enabled = 1;
rdrcfg.legic_connect.process_over_hce_enabled = 1;
rdrcfg.legic_connect.project_id = STR([0x01, 0xD9, 0x56, 0x4D]);
rdrcfg.legic_connect.application_id = STR([0x01, 0xD9, 0x56, 0x67]);
rdrcfg.legic_connect.neon.file_id = STR([0x70, 0x72, 0x69, 0x6E, 0x74, 0x65, 0x72, 0x00, 0x00, 0x00, 0x00, 0x00]);
rdrcfg.legic_connect.neon.read_offset = 0;
rdrcfg.legic_connect.neon.read_len = 32;
rdrcfg.legic_connect.neon.auth_key_type = LEGIC_NEON_KEY_TYPE_RO;
rdrcfg.legic_connect.neon.auth_key = STR([0x81, 0x6C, 0x18, 0x1F, 0xA1, 0xEA, 0xDA, 0x63, 0x66, 0x27, 0x06, 0x39, 0x00, 0x89, 0xF5, 0xA3]);
rdrcfg.legic_connect.neon.enc_key = STR([0x15, 0x4B, 0x46, 0x18, 0x5D, 0xAA, 0x99, 0x70, 0x53, 0xD9, 0xD1, 0xCA, 0x54, 0x4B, 0xB1, 0xE6]);
```


Reading Printer Neon file from Demo EKA application (over BLE central role only)(Keys already delivered by VCP)

```
rdrcfg.lf.lf_disabled = 1;
rdrcfg.hf.hf_disabled = 1;
rdrcfg.ble.min_rssi = -80;
rdrcfg.ble.tx_power = -40;
rdrcfg.ble.central_role_enabled = 1;
rdrcfg.legic_connect.process_over_ble_central_enabled = 1;
rdrcfg.legic_connect.project_id = STR([0x01, 0xD9, 0x56, 0x4D]);
rdrcfg.legic_connect.application_id = STR([0x01, 0xD9, 0x56, 0x67]);
rdrcfg.legic_connect.neon.file_id = STR([0x70, 0x72, 0x69, 0x6E, 0x74, 0x65, 0x72, 0x00, 0x00, 0x00, 0x00, 0x00]);
rdrcfg.legic_connect.neon.read_offset = 0;
rdrcfg.legic_connect.neon.read_len = 32;
rdrcfg.legic_connect.neon.auth_key_type = LEGIC_NEON_KEY_TYPE_RO;
```

Process Legic Orbit VCP file for key delivery (over BLE or NFC)

```
rdrcfg.lf.lf_disabled = 1;
rdrcfg.hf.hf_supported = HF_ISO14443A;
rdrcfg.ble.min_rssi = -80;
rdrcfg.ble.tx_power = -40;
rdrcfg.ble.central_role_enabled = 1;
rdrcfg.ble.peripheral_role_enabled = 1;
rdrcfg.legic_connect.process_over_ble_central_enabled = 1;
rdrcfg.legic_connect.process_over_ble_peripheral_enabled = 1;
rdrcfg.legic_connect.process_over_hce_enabled = 1;
rdrcfg.legic_connect.project_id = STR([0x01, 0xD9, 0x56, 0x4D]);
rdrcfg.legic_connect.application_id = STR([0x01, 0xD9, 0x56, 0x67]);
rdrcfg.legic_connect.vcp.file_id = STR([0x59, 0x53, 0x6f, 0x66, 0x74, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00]);
rdrcfg.legic_connect.vcp.label_id = STR([0x12, 0x34]);
rdrcfg.legic_connect.vcp.ask_for_password.message = STR(" Enter VCP password ");
rdrcfg.legic_connect.vcp.ask_for_password.timeout = 30000
```


Supported parameters


Supported cfg items	Supported values	Note
rdrcfg.ble.min_rssi	<-128,127>	Minimum RSSI [dBm] for BLE device filtering
rdrcfg.ble.tx_power	(-40, -20, -16, -12, -8, -4, 0, 4)	Transmit power level [dBm]
rdrcfg.ble.flags	{BLE_NONE , BLE_SEARCH_IOS , BLE_SEARCH_LC , BLE_SEARCH_LC_ALL, BLE_SEARCH_ACTIVE}	
rdrcfg.ble.central_role_enabled	(0, 1)	Enable central role search


Supported cfg items	Supported values	Note
rdrcfg.ble.central_scan_duration	<50,2000>	Scan duration for central role
rdrcfg.ble.peripheral_role_enabled	(0, 1)	Enable peripheral role search
rdrcfg.ble.peripheral_scan_duration	<50,2000>	Scan duration for peripheral role
rdrcfg.legic_connect.process_over_ble_central_enabled	(0, 1)	Process Legic Connect over devices found in central role search
rdrcfg.legic_connect.process_over_ble_peripheral_enabled	(0, 1)	Process Legic Connect over devices found in peripheral role search
rdrcfg.legic_connect.process_over_hce_enabled	(0, 1)	Process Legic Connect over devices found in HF search
rdrcfg.legic_connect.project_id	data[4]	Project id of credential neon file and Orbit configuration message (VCP)
rdrcfg.legic_connect.application_id	data[4]	Provide application ID of credential neon file and Orbit configuration message (VCP) or application must support project addressing
rdrcfg.legic_connect.vcp.file_id	data[12]	File ID of Orbit configuration message (VCP)
rdrcfg.legic_connect.vcp.label_id	data[2]	Label ID of key set in Orbit configuration message (VCP)
rdrcfg.legic_connect.vcp.password	data[0,32]	Password for Orbit configuration message (VCP) (UTF-8)
rdrcfg.legic_connect.vcp.ask_for_password.message	data[0,32]	Message shown on phone (UTF-8)
rdrcfg.legic_connect.vcp.ask_for_password.timeout	uint	Timeout [ms]
rdrcfg.legic_connect.neon.file_id	data[12]	File ID of credential neon file
rdrcfg.legic_connect.neon.read_offset	<0, 65535>	Read from offset in file
rdrcfg.legic_connect.neon.read_len	<1, 200>	Read this many bytes
rdrcfg.legic_connect.neon.auth_key_type	(LEGIC_NEON_KEY_TYPE_RO, LEGIC_NEON_KEY_TYPE_RW)	Key type must be provided despite of defining or delivering keys
rdrcfg.legic_connect.neon.auth_key	data[16]	Keys for credential neon file (if available, else deliver by VCP and left keys empty)
rdrcfg.legic_connect.neon.enc_key	data[16]	Keys for credential neon file (if available, else deliver by VCP and left keys empty)

Supported cfg items	Supported values	Note
rdrcfg.legic_connect.neon.use_ysoft_mobile_id	(0, 1)	Use keys for YSoft Connect ID credentials

6.8 HID ICLASS PARAMETERS

 PACS bits stands for Physical Access Control System - number that is stored in and encrypted area and it is returned by standard door card readers. It is not the same as HF card UIN which is a manufacturing serial number and is not encrypted and could be read by any UIN reader.

 If HID iClass reading is used in custom card reader configuration then every detected RFID card is scanned and searched for specified parameters. This is especially useful if the card used contains dual chip or the card is carried along with other cards (for example with traffic cards or bank cards in a purse).

 Always use rdrcfg.lf.lf_disabled = 1; for disabling LF technologies on combined multireaders. LF technologies are not secure and leaving it enabled presents a security risk. Unless you need to enable HID Prox technology.

Examples

Read only HID iClass SE/SEOS credentials

```
rdrcfg.lf.lf_disabled = 1;
rdrcfg.hf.hf_supported_multiple.extend([ HF_ICLASS, HF_ISO14443A ]);
rdrcfg.hf.iclass.process_pac_bits = 1;
rdrcfg.hf.iclass.allowed_data_models_multiple.extend([ ICLASS_PICO15693_SIO, ICLASS_ISO14443A_SIO,
ICLASS_ISO14443A_HID_MIFARE ]);
rdrcfg.hf.iclass.allowed_card_types_multiple.extend([ CARD_HID_DESFIRE_SE, CARD_HID_MIFARE_SE,
CARD_ICLASS_SE, CARD_HID_MIFARE_PLUS_SE, CARD_HID_SEOS ]);
```

Supported parameters

Supported cfg items	Supported values	Note
rdrcfg.hf.iclass.allowed_data_models_multiple	[ICLASS_PICO15693_ICLASS, ICLASS_PICO15693_SIO, ICLASS_ISO14443A_SIO, ICLASS_ISO14443A_HID_MIFARE]	iClass data models, appropriate HF technologies must be enabled separately

Supported cfg items	Supported values	Note
rdrcfg.hf.iclass.allowed_card_types_multiple	[CARD_HID_DESFIRE_SE, CARD_HID_DESFIRE, CARD_HID_MIFARE_SE, CARD_HID_MIFARE, CARD_ICCLASS_SE, CARD_ICCLASS, CARD_HID_MIFARE_PLUS_SE, CARD_HID_MIFARE_PLUS, CARD_HID_SEOS]	If present allow only these card types, else any. Appropriate HF technologies must be enabled separately.
rdrcfg.hf.iclass.process_pac_bits	(0, 1)	Enable processing of data from HID iClass cards
rdrcfg.hf.iclass.no_pac_bits_return_uin	(0, 1)	Card with not valid data (or incompatible keys) will return UIN
rdrcfg.hf.iclass.not_allowed_cards_return_uin	(0, 1)	cards that are not allowed by allowed_card_types_multiple will return UIN
rdrcfg.hf.iclass.report_card_read_error	(0, 1)	If there is a problem with reading the card, return card read error otherwise ignore the card
rdrcfg.hf.iclass.report_card_refused	(0, 1)	If there is a problem with reading the card, return card refused otherwise ignore the card
rdrcfg.hf.iclass.config_card_allow_time	<1, 65535>	Time in seconds after reader startup in which configuration cards are allowed, -1 means always enabled

6.9 HF ISO14443A/B PARAMETERS



Always use `rdrcfg.If.If_disabled = 1`; for disabling LF technologies on combined multireaders. LF technologies are not secure and leaving it enabled presents a security risk.

Examples

Cepas example

```
rdrcfg.If.If_disabled = 1;
rdrcfg.common.no_uin_check = 1;
rdrcfg.hf.hf_supported = HF_ISO14443B;
rdrcfg.hf.hf_iso_select = 1;
rdrcfg.hf.iso_read.read = ISO_CEPAS;
```

PIV CHUID example

```
rdrcfg.lf.lf_disabled = 1;  
rdrcfg.hf.hf_supported_multiple.extend([ HF_ISO14443A, HF_ISO14443B ]);  
rdrcfg.hf.hf_iso_select = 1;  
rdrcfg.hf.iso_read.read = ISO_CHUID;
```

Supported parameters

Supported cfg items	Supported values	Note
rdrcfg.common.no_uin_check	(0, 1)	Do not check if the card UIN is still the same. Required for cards with random UIN
rdrcfg.hf.hf_supported	(HF_DEFAULT, HF_ISO14443A, HF_ISO14443B, HF_ISO15693, HF_INSIDE, HF_FELICA, HF_SIELOX)	
rdrcfg.hf.hf_supported_multiple	[HF_ISO14443A, HF_ISO14443B, HF_ISO15693, HF_INSIDE, HF_FELICA, HF_SIELOX]	Multiple values supported
rdrcfg.hf.hf_iso_select	(0, 1)	ISO14443A/B-4 ISO select. Required for DESFire/ISO14443A/B-4 commands.
rdrcfg.hf.report_read_error	(REP_NONE, REP_AUTO, REP_ERROR, REP_REFUSED)	
rdrcfg.hf.iso_read.read	(ISO_CHUID, ISO_CEPAS)	

6.10 LF READER PARAMETERS

Examples

Enable both Paradox and ioProx technologies at the same time

```
rdrcfg.lf.lf_supported_multiple.extend([ LF_PARADOX, LF_IOPROX ]);
```

Enable reading of Proxlite technology

```
rdrcfg.lf.lf_supported = LF_PROXLITE;
```

Read custom passive LF card

```
rdrcfg.lf.lf_supported = LF_CUSTOM
rdrcfg.lf.lf_custom.modulation = LF_ASK
rdrcfg.lf.lf_custom.bitrate = LF_F32
rdrcfg.lf.lf_custom.start_decoding = LF_DECODE_MANCHESTER
rdrcfg.lf.lf_custom.data_decoding = LF_DECODE_MANCHESTER
rdrcfg.lf.lf_custom.start_decode_invert = 1
rdrcfg.lf.lf_custom.data_decode_invert = 1
rdrcfg.lf.lf_custom.start_cond = 0x000001FF
rdrcfg.lf.lf_custom.start_mask = 0x000001FF
rdrcfg.lf.lf_custom.padding_bits = 0
rdrcfg.lf.lf_custom.count_bits = 55
rdrcfg.lf.lf_custom.read_count = 3
rdrcfg.lf.lf_custom.remove_timeout = 600
rdrcfg.lf.lf_custom.decode = LF_DECODE_EM4K
rdrcfg.lf.lf_custom.card_type = CARD_UIN
```

Configuration parameters



Please note that not all LF technologies are supported on every card reader. Please see [Reader Configurations](#) for a list of currently supported card readers and their supported card reader configurations.

Supported cfg items	Supported values	Note
rdrcfg.lf.lf_disabled	(0, 1)	Completely disable processing of LF technologies
rdrcfg.lf.lf_supported	(LF_AUTO, LF_EM4000, LF_PROXLITE, LF_PARADOX, LF_NEDAP, LF_AWID, LF_PYRAMID, LF_DEISTER, LF_DATASEC, LF_GPROX, LF_IOPROX, LF_EM4050RO, LF_PAC, LF_URMET, LF_CARDAX, LF_JABLOTRON, LF_INDALA, LF_NEXKEY, LF_COTAG, LF_TIRIS, LF_KEYPAC, LF_IDTECK, LF_NORALSY, LF_KERI, LF_HITAG1S, LF_HITAG2, LF_EM4550, LF_HIDPROX, LF_CUSTOM)	
rdrcfg.lf.lf_supported_multiple	[LF_AUTO, LF_EM4000, LF_PROXLITE, LF_PARADOX, LF_NEDAP, LF_AWID, LF_PYRAMID, LF_DEISTER, LF_DATASEC, LF_GPROX, LF_IOPROX, LF_EM4050RO, LF_PAC, LF_URMET, LF_CARDAX, LF_JABLOTRON, LF_INDALA, LF_NEXKEY, LF_COTAG, LF_TIRIS, LF_KEYPAC, LF_IDTECK, LF_NORALSY, LF_KERI, LF_HITAG1S, LF_HITAG2, LF_EM4550, LF_HIDPROX]	Combining other technologies with LF_CUSTOM is limited.
rdrcfg.lf.lf_custom.modulation	(LF_ASK, LF_FSK2, LF_PSK1)	Modulation type

Supported cfg items	Supported values	Note
rdrcfg.lf.lf_custom.bitrate	(LF_F16, LF_F32, LF_F50, LF_F64, LF_F8, LF_F20, LF_F40)	Card bitrate
rdrcfg.lf.lf_custom.start_decoding	(LF_DECODE_NRZ, LF_DECODE_MANCHESTER, LF_DECODE_DIFFBIPHASE)	Decoding of start condition
rdrcfg.lf.lf_custom.data_decoding	(LF_DECODE_NRZ, LF_DECODE_MANCHESTER, LF_DECODE_DIFFBIPHASE)	Decoding of data
rdrcfg.lf.lf_custom.start_condition	u64	64 bit start condition. Must be only 32bit if compatibility with ASKFSK frontend is required.
rdrcfg.lf.lf_custom.start_mask	u64	64 bit start condition mask. Must be only continuous sequence of bits and 32bit if compatibility with ASKFSK frontend is required.
rdrcfg.lf.lf_custom.count_bits	int	Data bits count without start condition. If you do not specify complete bith length, then adjust ignore_bits for complete data reading
rdrcfg.lf.lf_custom.start_decode_invert	(0, 1)	Invert start condition data. Must be set to 1 if LF_DECODE_DIFFBIPHASE is used and compatibility with ASKFSK is required.
rdrcfg.lf.lf_custom.data_decode_invert	(0, 1)	Invert data. Must be set to 1 if DIFFBIPHASE decoding is used and compatibility with ASKFSK frontend is required.
rdrcfg.lf.lf_custom.input_invert	(0, 1)	Invert data incoming to decodes
rdrcfg.lf.lf_custom.read_count	int	How many times the data should be read and checked and compared as identical
rdrcfg.lf.lf_custom.padding_bits	int	Padding 0 bits before actual data output
rdrcfg.lf.lf_custom.ignore_bits	int	Bits at the end of the data stream that should be ignored (i.e. void bits), used for proper timing calculation
rdrcfg.lf.lf_custom.card_bootup_time	int	Value to add to card timeout - card bootup overhead in us
rdrcfg.lf.lf_custom.decode	(LF_DECODE_RAW, LF_DECODE_EM4K)	Use particular decode function
rdrcfg.lf.lf_custom.remove_timeout	uint	Timeout for card remove

Supported cfg items	Supported values	Note
rdrcfg.lf.lf_custom.card_type	uint	Card type to report

6.11 HID PROX PARAMETERS



Please note that LF_HIDPROX is supported only on some card readers. Please see [Reader Configurations](#) for a list of currently supported card readers and their supported card reader configurations.

Examples:

Read card UIN from HID Prox UDF cards

```
rdrcfg.lf.lf_supported = LF_HIDPROX;
rdrcfg.lf.hidprox.wiegand_format = WAUTO;
rdrcfg.lf.hidprox.hid_udf_enable = 1;
```

Read card number as printed on the card

```
rdrcfg.lf.lf_supported = LF_HIDPROX;
rdrcfg.lf.hidprox.wiegand_format = WAUTO;
rdrcfg.cardno.cardnoconv = CM2CNO("RightHexShift(1);HexAnd(FFFFF);Hex2Dec");
```

Configuration items:

Supported cfg items	Supported values	Note
rdrcfg.lf.hidprox.wiegand_format	(WAUTO, W26, W27, W32, W34, W37, W40, W40_PAR, W44, WABA2, WMAGSTRIPE)	
rdrcfg.lf.hidprox.hid_udf_enable	(0, 1)	Enable processing of HID Prox User Defined Formats - cards with non-standard customer code
rdrcfg.lf.hidprox.no_wiegand_parity_check	(0, 1)	Do not check parity on wiegand formats that contain parity
rdrcfg.lf.hidprox.no_error_report	(0, 1)	Do not report parity or other errors

6.12 HID INDALA PARAMETERS



Please note that LF_INDALA is supported only on some card readers. Please see [Reader Configurations](#) for a list of currently supported card readers and their supported card reader configurations.

Examples:

Read card UIN from HID Indala cards

```
rdrcfg.lf.lf_supported = LF_INDALA;  
rdrcfg.lf.indala.format = INDALA_UIN1_LONG;  
rdrcfg.hf.hf_disabled = 1;
```


Read card number as printed on the card


```
rdrcfg.lf.lf_supported = LF_INDALA;  
rdrcfg.lf.indala.format = INDALA_10022;  
rdrcfg.lf.indala.wiegand_format = W26;  
rdrcfg.hf.hf_disabled = 1;  
rdrcfg.cardno.cardnoconv = CM2CNO("RightHexShift(1);HexAnd(FFFFF);Hex2Dec");
```

Supported cfg items	Supported values	Note
rdrcfg.lf.indala.format	(INDALA_UIN1_SHORT, INDALA_UIN2_SHORT, INDALA_UIN1_LONG, INDALA_UIN2_LONG, INDALA_10022, INDALA_10251, INDALA_11037, INDALA_15024, INDALA_17531, INDALA_1771X, INDALA_10324, INDALA_11045, INDALA_DEFWIE, INDALA_DEFABA, INDALA_DEFSER, INDALA_CUSTOM1, INDALA_CUSTOM2, INDALA_CUSTOM3)	
rdrcfg.lf.indala.wiegand_format	(WAUTO, W26, W27, W32, W34, W37, W40, W40_PAR, W44, WABA2, WMAGSTRIPE)	
rdrcfg.lf.indala.no_wiegand_parity_check	(0, 1)	Do not check parity on wiegand formats that contain parity
rdrcfg.lf.indala.magstripe_format	(MAG_FMT_DEFAULT, MAG_ABA, MAG_IATA, MAG_NTT, MAG_CADMV1, MAG_CADMV3, MAG_SECURITAS, MAG_BINARY)	
rdrcfg.lf.indala.serial_mode	(SER_INDALA_SER, SER_INDALA_UIN, SER_INDALA_UINLONG, SER_INDALA_HEX)	
rdrcfg.lf.indala.custom_format_data	data[256]	256 bytes of reader configuration from NHX file (Indala ProxSmith)

Supported cfg items	Supported values	Note
rdrcfg.lf.indala.asp_disabled	(0, 1)	Do not process ASP technology on LF_UIN mode
rdrcfg.lf.indala.asp_plus_disabled	(0, 1)	Do not process ASP+ technology on LF_UIN mode
rdrcfg.lf.indala.ecr_disabled	(0, 1)	Do not process ECR technology on LF_UIN mode

6.13 HITAG 1/HITAG S256/HITAG S2048 PARAMETERS


 Please note that LF_HITAG1S is supported only on some card readers. Please see [Reader Configurations](#) for a list of currently supported card readers and their supported card reader configurations.


 Plain mode only. Crypto mode not supported.

Examples:

Read two data words from 6th word		
<pre>rdrcfg.lf.lf_supported = LF_HITAG1S; rdrcfg.lf.hitag1s_read.address = 6; rdrcfg.lf.hitag1s_read.read_count = 2; rdrcfg.hf.hf_disabled = 1;</pre>		
Supported cfg items	Supported values	Note
rdrcfg.lf.hitag1s_read.address	Hitag 1: <0,63>; Hitag S 256: <0,7>; Hitag S 2048: <0,63>	Read from n-th word (1 word = 32 bits)
rdrcfg.lf.hitag1s_read.read_count	<1, 64>	Read this many words (each has 32-bits)

6.14 HITAG 2 PARAMETERS

 Please note that LF_HITAG2 is supported only on some card readers. Please see [Reader Configurations](#) for a list of currently supported card readers and their supported card reader configurations.

 Password mode only. Crypto mode not supported.

Examples:

Read two data words from 4th word of Hitag2 card with default passwords

```
rdrcfg.lf.lf_supported = LF_HITAG2;  
rdrcfg.lf.hitag2_read.flags = LF_HITAG2_NONE;  
rdrcfg.lf.hitag2_read.address = 4;  
rdrcfg.lf.hitag2_read.read_count = 2;  
rdrcfg.hf.hf_disabled = 1;
```

Read data word from Hitag2 card with custom passwords

```
rdrcfg.lf.lf_supported = LF_HITAG2;  
rdrcfg.lf.hitag2_read.flags = LF_HITAG2_USE_PASSWORD | LF_HITAG2_CHECK_TAG_PASSWORD;  
rdrcfg.lf.hitag2_read.password = STR([0x01, 0x23, 0x45, 0x67]);  
rdrcfg.lf.hitag2_read.tag_password = STR([0xAB, 0xCD, 0xEF]);  
rdrcfg.lf.hitag2_read.address = 4;  
rdrcfg.lf.hitag2_read.read_count = 1;  
rdrcfg.hf.hf_disabled = 1;
```

Supported cfg items	Supported values	Note
rdrcfg.lf.hitag2_read.flags	{LF_HITAG2_NONE, LF_HITAG2_USE_PASSWORD, LF_HITAG2_CHECK_TAG_PASSWORD}	OR combinations possible. Use provided password. Check for provided tag password.
rdrcfg.lf.hitag2_read.password	data[4]	RWD password
rdrcfg.lf.hitag2_read.tag_password	data[3]	TAG password
rdrcfg.lf.hitag2_read.address	<0,7>	Read from n-th word (1 word = 32 bits)
rdrcfg.lf.hitag2_read.read_count	<1,8>	Read this many words (each has 32-bits)

6.15 EM4050/V4050/P4150/P4350/EM4150/EM4350/EM4450/EM4550 A6 (OPT64) PARAMETERS



Please note that LF_EM4550 is supported only on some card readers. Please see [Reader Configurations](#) for a list of currently supported card readers and their supported card reader configurations.

Examples:

Read two data words from 8th word


```
rdrcfg.lf.lf_supported = LF_EM4550;  
rdrcfg.lf.em4050_read.flags = LF_EM4050_NONE;  
rdrcfg.lf.em4050_read.address = 8;  
rdrcfg.lf.em4050_read.read_count = 2;  
rdrcfg.hf.hf_disabled = 1;
```

Read data word from card with custom passwords

```
rdrcfg.lf.lf_supported = LF_EM4550;  
rdrcfg.lf.em4050_read.flags = LF_EM4050_USE_PASSWORD;  
rdrcfg.lf.em4050_read.password = STR([0x01, 0x23, 0x45, 0x67]);  
rdrcfg.lf.em4050_read.address = 9;  
rdrcfg.lf.em4050_read.read_count = 1;  
rdrcfg.hf.hf_disabled = 1;
```

Supported cfg items	Supported values	Note
rdrcfg.lf.em4050_read.flags	{LF_EM4050_NONE, LF_EM4050_USE_PASSWORD}	OR combinations possible. Login to card with provided password.
rdrcfg.lf.em4050_read.password	data[4]	Password for read protected area
rdrcfg.lf.em4050_read.address	<1,33>	Read from n-th word (1 word = 32 bits)
rdrcfg.lf.em4050_read.read_count	<1,33>	Read this many words (each has 32-bits)

6.16 ATMEL Q5 (T5555)/Q5B (T5555B)/T5557/T5567/T5577 M1/T5577 M2/T5577 M3 PARAMETERS

 Please note that ATMEL Q5 family is supported only on some card readers. Please see [Reader Configurations](#) for a list of currently supported card readers and their supported card reader configurations.

 Supported only selected configurations of emulated cards - Manchester f/64

Examples:

Read two data words from 4th word - from any Manchester f/64 modulated card

```
rdrcfg.lf.lf_supported = LF_Q5_MANCH_64_DATA;  
rdrcfg.lf.q5_read.flags = LF_Q5_NONE;  
rdrcfg.lf.q5_read.address = 4;  
rdrcfg.lf.q5_read.read_count = 2;  
rdrcfg.lf.q5_read.card_detect = LF_Q5_MANCH_64_DATA;  
rdrcfg.hf.hf_disabled = 1;
```

Read two data words from 4th word of Q5 emulating EM4000

```
rdrcfg.lf.lf_supported = LF_EM4000;  
rdrcfg.lf.q5_read.flags = LF_Q5_NONE;  
rdrcfg.lf.q5_read.address = 4;  
rdrcfg.lf.q5_read.read_count = 2;  
rdrcfg.lf.q5_read.card_detect = LF_EM4000;  
rdrcfg.hf.hf_disabled = 1;
```

Read data word from card with custom passwords

```
rdrcfg.lf.lf_supported = LF_Q5_MANCH_64_DATA;  
rdrcfg.lf.q5_read.flags = LF_Q5_USE_PASSWORD;  
rdrcfg.lf.q5_read.password = STR([0x01, 0x23, 0x45, 0x67]);  
rdrcfg.lf.q5_read.address = 4;  
rdrcfg.lf.q5_read.read_count = 1;  
rdrcfg.lf.q5_read.card_detect = LF_Q5_MANCH_64_DATA;  
rdrcfg.hf.hf_disabled = 1;
```

Supported cfg items	Supported values	Note
rdrcfg.lf.q5_read.flags	{LF_Q5_NONE, LF_Q5_USE_PASSWORD}	OR combinations possible. Use provided password.
rdrcfg.lf.q5_read.password	data[4]	Password for tags with read protection
rdrcfg.lf.q5_read.address	Q5, Q5B: <0,7>; T5557: <0,10>; T5577: <0,11>	Read from n-th word (1 word = 32 bits)
rdrcfg.lf.q5_read.read_count	<1,12>	Read this many words (each has 32-bits)
rdrcfg.lf.q5_read.card_detect	(LF_Q5_MANCH_64_DATA, LF_EM4000, LF_KEYPAC)	Process only cards detected as some other simulated LF technology

6.17 COMBINING TECHNOLOGIES TOGETHER



Not every configuration option is supported at every reader. Please see [Reader Configurations](#) for a list of currently supported card readers and their supported configurations.

Examples:

Combining Indala W26 and ISO14443-A (Mifare) UIN

```
rdrcfg.hf.hf_supported = HF_ISO14443A;  
rdrcfg.lf.lf_supported = LF_INDALA;  
rdrcfg.lf.indala.format = INDALA_10022;  
rdrcfg.cardno.cardnoconv = CM2CNO("LeftPadding(0,8)")
```

HID Prox and Indala W26

```
rdrcfg.lf.lf_supported_multiple.extend([ LF_HIDPROX, LF_INDALA ]);  
rdrcfg.lf.hidprox.wiegand_format = WAUTO;  
rdrcfg.lf.indala.format = INDALA_10022;
```



Please note that combining multiple HF data area reading is supported only from the newest 2.6.0 USB card reader firmware and only on the following card readers:

- Multireader HF, LF+HF, LF+HF v2
- Legic Advant v3, Legic Advant + HID Prox
- Reader 3 MF, MF+, MF SAM, MF&Legic,
- Reader 3 MFX, MFX Mobile Reader

Mifare classic 1k/4k and Mifare DESFire reading

```
rdrcfg.lf.lf_disabled = 1;
rdrcfg.hf.hf_supported = HF_ISO14443A;

# Mifare classic parameters
rdrcfg.hf.mifare_read.mode = MIFARE_CLASSIC;
rdrcfg.hf.mifare_read.mifare_block = 0;
rdrcfg.hf.mifare_read.mifare_key.key_type = KEY_MIFARE_A;
rdrcfg.hf.mifare_read.mifare_key.key_data = STR([ 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF ]);

# Mifare DESFire parameters
rdrcfg.hf.desfire_read.desfire_aid = STR([ 0x00, 0x00, 0x01 ]);
rdrcfg.hf.desfire_read.desfire_fid = 0x00;
rdrcfg.hf.desfire_read.desfire_foffset = 0;
rdrcfg.hf.desfire_read.desfire_flen = 0;
rdrcfg.hf.desfire_read.desfire_fopts = DESFIRE_FILE_AUTO;
rdrcfg.hf.desfire_read.desfire_key_no = 1;
rdrcfg.hf.desfire_read.desfire_key.key_type = KEY_2K3DES;
rdrcfg.hf.desfire_read.desfire_key.key_data = STR([ 0x01, 0x23, 0x45, 0x67, 0x89, 0xAB, 0xCD, 0xEF, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00 ]);
```

YSoft Connect ID (over BLE or NFC) and Legic data segment

```
rdrcfg.lf.lf_disabled = 1;
rdrcfg.hf.hf_supported_multiple.extend([ HF_ISO14443A, HF_LEGIC_PRIME, HF_LEGIC_ADVANT ])

#
# Legic connect parameters
#
# Limit BLE receive signal
rdrcfg.ble.min_rssi = -60;
# Limit BLE transmit power
rdrcfg.ble.tx_power = -40;

# BLE Central role enabled
rdrcfg.ble.central_role_enabled = 1;
rdrcfg.legic_connect.process_over_ble_central_enabled = 1;

# Authentication time is longer if the reader operates in peripheral role
# and mobile phone in central role (older Android phones, or some new phones
# with poor BLE implementation).
# When this scenario not needed then commenting the following two lines will
# improve reader overall response time.
rdrcfg.ble.peripheral_role_enabled = 1;
rdrcfg.legic_connect.process_over_ble_peripheral_enabled = 1;

# NFC/HCE processing enabled
rdrcfg.legic_connect.process_over_hce_enabled = 1;

# Support from production environment
rdrcfg.legic_connect.project_id = STR([0x04, 0x61, 0xBA, 0xF7]);
rdrcfg.legic_connect.application_id = STR([0x04, 0x61, 0xBA, 0xF5]);
rdrcfg.legic_connect.neon.file_id = STR([0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00]);
rdrcfg.legic_connect.neon.read_offset = 0;
rdrcfg.legic_connect.neon.read_len = 8;
rdrcfg.legic_connect.neon.auth_key_type = LEGIC_NEON_KEY_TYPE_RO;
rdrcfg.legic_connect.neon.use_ysoft_mobile_id = 1;

#
# Legic card parameters
#
rdrcfg.hf.legic_read.mode = LEGIC_DATA_SEG
rdrcfg.hf.legic_read.flags = LEGIC_FLAG_NONE
rdrcfg.hf.legic_read.search_string = STR([ 0x00, 0xAA ])
rdrcfg.hf.legic_read.read_len = 3
rdrcfg.hf.legic_read.read_offset = 0
rdrcfg.hf.legic_read.from_seg = 0
```


YSoft Connect ID (over BLE or NFC) and HID iClass SE credentials (MFX Mobile reader with SE processor in SAM slot only)

```
rdrcfg.lf.lf_disabled = 1;
rdrcfg.hf.hf_supported_multiple.extend([ HF_ISO14443A, HF_ICLASS ])

#
# Legic connect parameters
#
# Limit BLE receive signal
rdrcfg.ble.min_rssi = -60;
# Limit BLE transmit power
rdrcfg.ble.tx_power = -40;

# BLE Central role enabled
rdrcfg.ble.central_role_enabled = 1;
rdrcfg.legic_connect.process_over_ble_central_enabled = 1;

# Authentication time is longer if the reader operates in peripheral role
# and mobile phone in central role (older Android phones, or some new phones
# with poor BLE implementation).
# When this scenario not needed then commenting the following two lines will
# improve reader overall response time.
rdrcfg.ble.peripheral_role_enabled = 1;
rdrcfg.legic_connect.process_over_ble_peripheral_enabled = 1;

# NFC/HCE processing enabled
rdrcfg.legic_connect.process_over_hce_enabled = 1;

# Support from production environment
rdrcfg.legic_connect.project_id = STR([0x04, 0x61, 0xBA, 0xF7]);
rdrcfg.legic_connect.application_id = STR([0x04, 0x61, 0xBA, 0xF5]);
rdrcfg.legic_connect.neon.file_id = STR([0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00]);
rdrcfg.legic_connect.neon.read_offset = 0;
rdrcfg.legic_connect.neon.read_len = 8;
rdrcfg.legic_connect.neon.auth_key_type = LEGIC_NEON_KEY_TYPE_RO;
rdrcfg.legic_connect.neon.use_ysoft_mobile_id = 1;

#
# iClass SE credentials
#
rdrcfg.sam.sam_mode = SAM_HIDSEPROC;
rdrcfg.hf.iclass.process_pac_bits = 1;
rdrcfg.hf.iclass.allowed_data_models_multiple.extend([ ICLASS_PICO15693_SIO, ICLASS_ISO14443A_SIO,
ICLASS_ISO14443A_HID_MIFARE ]);
rdrcfg.hf.iclass.allowed_card_types_multiple.extend([ CARD_HID_DESFIRE_SE, CARD_HID_MIFARE_SE,
CARD_ICLASS_SE, CARD_HID_MIFARE_PLUS_SE, CARD_HID_SEOS ]);
```

YSoft Connect ID (over BLE or NFC) and HID Prox

```
# Enable HID Prox reading
rdrcfg.lf.lf_supported = LF_HIDPROX;
rdrcfg.lf.hidprox.wiegand_format = WAUTO;

# ISO14443A needed for NFC/HCE below
rdrcfg.hf.hf_supported_multiple.extend([ HF_ISO14443A ])

# Limit BLE receive signal
rdrcfg.ble.min_rssi = -60;
# Limit BLE transmit power
rdrcfg.ble.tx_power = -40;

# BLE Central role enabled
rdrcfg.ble.central_role_enabled = 1;
rdrcfg.legic_connect.process_over_ble_central_enabled = 1;

# Authentication time is longer if the reader operates in peripheral role
# and mobile phone in central role (older Android phones, or some new phones
# with poor BLE implementation).
# When this scenario not needed then commenting the following two lines will
# improve reader overall response time.
rdrcfg.ble.peripheral_role_enabled = 1;
rdrcfg.legic_connect.process_over_ble_peripheral_enabled = 1;

# NFC/HCE processing enabled
rdrcfg.legic_connect.process_over_hce_enabled = 1;

# Support from production environment
rdrcfg.legic_connect.project_id = STR([0x04, 0x61, 0xBA, 0xF7]);
rdrcfg.legic_connect.application_id = STR([0x04, 0x61, 0xBA, 0xF5]);
rdrcfg.legic_connect.neon.file_id = STR([0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00]);
rdrcfg.legic_connect.neon.read_offset = 0;
rdrcfg.legic_connect.neon.read_len = 8;
rdrcfg.legic_connect.neon.auth_key_type = LEGIC_NEON_KEY_TYPE_RO;
rdrcfg.legic_connect.neon.use_ysoft_mobile_id = 1;
```

7 CARD MANAGER CONVERSION RULES FOR CARD READERS

Card conversion can be used for **translating of card reader output** (the data reader gets after user swipes a card) to **card number stored in user database** (LDAP, AD, YSoft SafeQ, etc.) if these two numbers are different.

7.1 CONVERSION FUNCTION

SafeQ Server or Card Readers support conversions of card numbers as read by Card Reader at Terminal. If conversion function is defined, card numbers are automatically transformed prior matching with (or storing to) SafeQ Identity Database.

Typical conversion configuration looks as follows and is represented by `conversion` attribute in SafeQ configuration:

```
ASCII2Hex;Hex2Dec;Substring(-8)
```

Each rule is represented by its name (see description of rules) and separated by semicolon. Some rules have one or two parameters which are in parentheses and separated by comma.

```
Substring(2);Hex2Dec;LeftPadding(0,3) + Substring(2,6);Hex2Dec
```

Some conversions may contain two (or more) independent processing parts that are concentrated by operator `+`.

The conversion is **executed** from the left one rule after another. Where operator `+` appears it concentrates results of the last conversion in each parts. Each part is processed in parallel with original number as input.

If the rule expect some limitations of input which are not met (for example Dec2Hex expect decimal representation of number but input string contain hexadecimal values), then input string is leaved unconverted in most cases.

The conversion string on server is interpreted by an internal Java Class, where on card readers it is compiled into internal card number processing engine byte code. Implementation of each rule on SafeQ Server and in Card Reader therefore may slightly differ in some cases. These differences are noted in each rules as **Server notes** and **Reader notes**.


Example of use in custom configuration


```
rdrcfg.cardno.cardnoconv = CM2CNO("Hex2Dec;LeftPadding(0,5)")
```

If the resulting conversion is empty then no card number is returned (valid from YSoft Card Reader Tool v. 1.4.0)

7.2 CONDITIONS

It is possible to specify conditions inside the card manager conversion rules so that the card number conversion applies if the condition is met.

 Please note that some of the conditions are supported from YSoft Card Reader Tool v. 1.4.0 only. Also "+" operator can be used to form more complex if - else constructs there.

 Please note that some conditions are currently **not** supported on SafeQ server. Card Type conditions even technically cannot be implemented on SafeQ as the server does not have the information about card type available.

Condition list:

Condition	Description
IsCardTypeEE	Allow processing of next rules only if placed card type masked by <i>mask</i> (binary AND) is equal to <i>type</i> else empty current output.
IsEmbed	Allow processing of next rules (until operator "+") only if card number is from embedded reader.
IsEven	Allow processing of next rules (until operator "+") only if card number length is even.
IsEvenEE	Allow processing of next rules only if card number length is even else empty current output.
IsLength	Allow processing of next rules (until operator "+") only if card number length is equal to specified value.
IsLengthEE	Allow processing of next rules only if card number length is equal to specified value else empty current output.
IsLengthGreater	Allow processing of next rules (until operator "+") only if card number length is greater than specified value.
IsLengthGreaterEE	Allow processing of next rules only if card number length is greater than specified value else empty current output.
IsLengthNot	Allow processing of next rules (until operator "+") only if card number length is different from value.
IsLengthNotEE	Allow processing of next rules only if card number length is different from value else empty current output.
IsLengthSmaller	Allow processing of next rules (until operator "+") only if card number length is smaller than specified value.
IsLengthSmallerEE	Allow processing of next rules only if card number length is smaller than specified value else empty current output.
IsNotCardTypeEE	Allow processing of next rules only if placed card type masked by <i>mask</i> (binary AND) is different to <i>type</i> else empty current output.

Condition	Description
IsNotStartWith	Allow processing of next rules (until operator "+") only if card number doesn't start with specified string.
IsNotStartWithEE	Allow processing of next rules only if card number doesn't start with specified string else empty current output.
IsOdd	Allow processing of next rules (until operator "+") only if card number length is odd.
IsOddEE	Allow processing of next rules only if card number length is odd else empty current output.
IsStartWith	Allow processing of next rules (until operator "+") only if card number starts with specified string.
IsStartWithEE	Allow processing of next rules only if card number starts with specified string else empty current output.

Examples:

Example of a simple condition. If the card number length is 10 characters then do the specified conversion otherwise return the original number:

```
IsLength(10);SwapPair
```

Example of if-else condition. If the card type is HID Prox then do RightHexShift(1);HexAnd(FFFF);Hex2Dec conversion, in other cases do just Hex2Dec :

```
IsCardTypeEE(CARD_HIDPROX, CARD_TYPE_MASK);RightHexShift(1);HexAnd(FFFF);Hex2Dec + IsNotCardTypeEE(CARD_HIDPROX, CARD_TYPE_MASK);Hex2Dec
```

7.3 DESCRIPTION OF RULES

Server notes:	⚠ Please note that rule names are CASE SENSITIVE
Reader notes:	⚠ Please note that rule names are case insensitive
	⚠ These rules are NOT IMPLEMENTED: DESDecrypt, DESEncrypt, MD5

Following rules are sorted alphabetically.

ASCII2Hex	<p>This rule converts string in ASCII format (typically from KM reader) into hex form. Other input is not changed. ASCII format is <code>"^([34][0-9])+([fF]{2})*\$"</code></p> <p>Syntax:</p> <ul style="list-style-type: none"> • ASCII2Hex – convert only string with length 32 signs • ASCII2Hex(length) – convert first length characters of input - length have to be even <p>Examples:</p> <ul style="list-style-type: none"> • ASCII2Hex(16) • 33303730314446383030FFFFFFFFFFFF => 30701DF800 • 30701DF800 => 30701DF800 <p>Server notes:</p> <ul style="list-style-type: none"> • Accepts also ASCII representation of @, G, H, I • Rule with length: length could not exceed length of input string • Rule with length: input string could contain FF at the end, which are converted to 'ÿ' instead of stripping them <p>Reader notes:</p> <ul style="list-style-type: none"> • Accepts only ASCII representation of 0-9, A-F (30 - 39, 41 - 46) and multiple trailing FF • Rule with length: Strip trailing FFs
Bin2Dec	<p>Converts number from binary format into decimal format.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • Bin2Dec <p>Example:</p> <ul style="list-style-type: none"> • Bin2Dec • 101011 => 43 <p>Server notes:</p> <ul style="list-style-type: none"> • Resulting number must fit to 64-bit signed integer (Input should not have more then 63 characters) <p>Reader notes:</p> <ul style="list-style-type: none"> • Input string may contain spaces
CardNo	<p>Returns original card number. Could be used with operator +.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • CardNo <p>Examples:</p> <ul style="list-style-type: none"> • CardNo + Const(@ysoft.com) • 12345 => 12345@ysoft.com <p>Server notes:</p> <ul style="list-style-type: none"> • NOT IMPLEMENTED

	<p>Reader notes:</p> <ul style="list-style-type: none"> Supported from YSoft Card Reader Tool v. 1.4.0
CardType	<p>Returns card type internal id (hexadecimal number, not padded, without hex notation - "0x"). Could be used with operator +. Card type is basically 16-bit unsigned integer, where higher 8 bits define card family and lower 8 bits define exact type in family (Exact type detection has limited support).</p> <p>Syntax:</p> <ul style="list-style-type: none"> CardType <p>Examples:</p> <ul style="list-style-type: none"> CardType; LeftPadding(0,4) + Const(-) + LeftPadding(0,6) iClass 16k, (16K/16): 12345 => 0A03-012345 CardType + Const(-) + LeftPadding(0,6) iClass 16k, (16K/16): 12345 => A03-012345 <p>Server notes:</p> <ul style="list-style-type: none"> NOT IMPLEMENTED <p>Reader notes:</p> <ul style="list-style-type: none"> Supported from YSoft Card Reader Tool v. 1.4.0
Const	<p>Returns specified string. Could be used with operator +. Similar functionality provides LeftAppend and RightAppend.</p> <p>Syntax:</p> <ul style="list-style-type: none"> Const(12345) <p>Examples:</p> <ul style="list-style-type: none"> LeftPadding(0,6) + Const(@ysoft.com) 12345 => 012345@ysoft.com 123 => 000123@ysoft.com <p>Server notes:</p> <ul style="list-style-type: none"> No functional limitations <p>Reader notes:</p> <ul style="list-style-type: none"> No functional limitations
Dec2Bin	<p>Converts number from decimal format into binary format.</p> <p>Syntax:</p> <ul style="list-style-type: none"> Dec2Bin <p>Example:</p> <ul style="list-style-type: none"> Dec2Bin 43 => 101011 <p>Server notes:</p> <ul style="list-style-type: none"> Input number must fit to 64-bit signed integer (Output should not have more then 63 characters) <p>Reader notes:</p> <ul style="list-style-type: none"> No functional limitations

Dec2Hex	<p>Converts number in decimal format into hexadecimal format. Output is in lowercase.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • Dec2Hex <p>Examples:</p> <ul style="list-style-type: none"> • Dec2Hex • 12345 => 3039 • 123 => 7b <p>Server notes:</p> <ul style="list-style-type: none"> • Input number must fit to 64-bit signed integer (Output should not have more then 16 characters) <p>Reader notes:</p> <ul style="list-style-type: none"> • No functional limitations
DecimalAdd	<p>Adds value in decimal format to current value in decimal format</p> <p>Syntax:</p> <ul style="list-style-type: none"> • DecimalAdd(value) <p>Examples:</p> <ul style="list-style-type: none"> • DecimalAdd(1) • 12345 => 12346 • 123 => 124 <p>Server notes:</p> <ul style="list-style-type: none"> • Input and output number must fit to 64-bit signed integer (Should not have more then 19 characters) <p>Reader notes:</p> <ul style="list-style-type: none"> • No functional limitations
DecimalAnd	<p>Make binary AND. Mask is in decimal format.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • DecimalAnd(mask) <p>Example:</p> <ul style="list-style-type: none"> • DecimalAnd(15) • 7 => 7 • 467825 => 1 <p>Server notes:</p> <ul style="list-style-type: none"> • Input number and mask must fit to 64-bit signed integer (Should not have more then 19 characters) <p>Reader notes:</p> <ul style="list-style-type: none"> • No functional limitations

DecValue2Hex	<p>Inversion function to Hex2DecValue. Converts each pair of decimal number to hexadecimal digit. (08 -> 8, 11 -> b). Input must have even length. Output is in lowercase.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • DecValue2Hex <p>Examples:</p> <ul style="list-style-type: none"> • 09101112 => 9abc • 0DD => 0DD <p>Server notes:</p> <ul style="list-style-type: none"> • Decimal values greater then 15 (f) are supported and return two hexadecimal characters (1045 => a2d) <p>Reader notes:</p> <ul style="list-style-type: none"> • Decimal values greater then 15 (f) are supported and return two hexadecimal characters (1045 => a2d)
DESDecrypt	<p>Decodes value encrypted by DES in Base64 format.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • DESDecrypt <p>Example:</p> <ul style="list-style-type: none"> • AzRapSymPps= => 1234 <p>Server notes:</p> <ul style="list-style-type: none"> • No functional limitations <p>Reader notes:</p> <ul style="list-style-type: none"> • NOT IMPLEMENTED
DESEncrypt	<p>Encodes value into DES and Base64 format.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • DESEncrypt <p>Example:</p> <ul style="list-style-type: none"> • 1234 => AzRapSymPps= <p>Server notes:</p> <ul style="list-style-type: none"> • No functional limitations <p>Reader notes:</p> <ul style="list-style-type: none"> • NOT IMPLEMENTED

Hex2ASCII	<p>This is inverse function to ASCII2Hex. Converts hexadecimal string to its ASCII representation. Input string could have even length and maximum of 16 signs. Otherwise original input is returned.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • Hex2ASCII <p>Examples:</p> <ul style="list-style-type: none"> • Hex2ASCII • 12AB => 31324142 • JEDNA => JEDNA • 30701DF800 => 33303730314446383030 <p>Server notes:</p> <ul style="list-style-type: none"> • Non convertible input string is changed to uppercase <p>Reader notes:</p> <ul style="list-style-type: none"> • Non convertible input string is changed to uppercase
Hex2Dec	<p>Converts number from hexadecimal format into decimal format. Input string could have even length.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • Hex2Dec <p>Example:</p> <ul style="list-style-type: none"> • Hex2Dec • 12AB => 4779 <p>Server notes:</p> <ul style="list-style-type: none"> • Input number must fit to 64-bit signed integer (Should not have more then 16 characters) <p>Reader notes:</p> <ul style="list-style-type: none"> • No functional limitations
Hex2DecValue	<p>Converts each hexadecimal digit into decimal representation (8 – 08, A – 10, B – 11, etc).</p> <p>Syntax:</p> <ul style="list-style-type: none"> • Hex2DecValue <p>Example:</p> <ul style="list-style-type: none"> • Hex2DecValue • 12AB => 01021011 <p>Server notes:</p> <ul style="list-style-type: none"> • No functional limitations <p>Reader notes:</p> <ul style="list-style-type: none"> • No functional limitations

Hex2Oct	<p>Converts number from hexadecimal format into octal format.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • Hex2Oct <p>Example:</p> <ul style="list-style-type: none"> • Hex2Oct • 12AB => 11253 <p>Server notes:</p> <ul style="list-style-type: none"> • Input number must fit to 64-bit signed integer (Should not have more then 16 characters) <p>Reader notes:</p> <ul style="list-style-type: none"> • No functional limitations
HexASCII2Char	<p>This rule converts pair of hexadecimal signs into string where each pair represents value in ASCII table of resulting character. Allowed format is "[^](3[0-9] 4[1-9a-fA-F] 5[0-9aA] 6[1-9a-fA-F] 7[0-9aA])+\$. Even length is necessary. Input with odd length is left unmodified.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • HexASCII2Char – convert whole string • HexASCII2Char(length) – convert first length characters of input - length have to be even <p>Examples:</p> <ul style="list-style-type: none"> • HexASCII2Char(8) • 33303730314446383030 => 3070 • 416A373330 => Aj73 <p>Server notes:</p> <ul style="list-style-type: none"> • Rule with length: input string after length must also full fill format requirements <p>Reader notes:</p> <ul style="list-style-type: none"> • Accepts all characters with hexadecimal representation from 30 to 7A
HexAnd	<p>Make binary AND. Mask is in hexadecimal format. Similar functionality contains DecimalAnd.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • HexAnd(mask) <p>Examples:</p> <ul style="list-style-type: none"> • HexAnd(FF) • 7 => 7 • 7237B => 7B <p>Server notes:</p> <ul style="list-style-type: none"> • Input number must fit to 64-bit signed integer (Should not have more then 16 characters) <p>Reader notes:</p> <ul style="list-style-type: none"> • No functional limitations

IsCardTypeEE	<p><i>Mask</i> and <i>type</i> are numbers or internal defines CARD_*. To use value from CardType format as hex - prepend "0x". Allow processing of next rules only if placed card type masked by <i>mask</i> (binary AND) is equal to <i>type</i> else empty current output and skip to next "+" operator.</p> <p>Syntax:</p> <ul style="list-style-type: none"> IsCardTypeEE(type, mask) <p>Predefined masks:</p> <ul style="list-style-type: none"> CARD_TYPE_MASK - (0xFF00) - Use only card family type part. For example iClass 2K, iClass 16k, etc. have family type iClass. CARD_FULL_MASK - (0xFFFF) - Use exact match with concrete card type. For example iClass 2K, iClass 16k has different card type. <p>Examples:</p> <ul style="list-style-type: none"> IsCardTypeEE(0xA00, 0xFF00);LeftAppend(iClass-) + IsCardTypeEE(0xD00, 0xFF00);LeftAppend(Inside-) IsCardTypeEE(CARD_ICLASS, CARD_TYPE_MASK);LeftAppend(iClass-) + IsCardTypeEE(CARD_INSIDE, CARD_TYPE_MASK);LeftAppend(Inside-) iClass 32k: 123456 => iClass-123456 Inside Contactless 16k: 123456 => Inside-123456 IsCardTypeEE(CARD_EM4000, CARD_FULL_MASK);Hex2Dec EM4K: 12AB => 4779 others: Ignored IsCardTypeEE(CARD_EM4000, CARD_FULL_MASK);Hex2Dec + IsNotCardTypeEE(CARD_EM4000, CARD_FULL_MASK);CardNo EM4K: 12AB => 4779 others: leave without change <p>Server notes:</p> <ul style="list-style-type: none"> NOT IMPLEMENTED <p>Reader notes:</p> <ul style="list-style-type: none"> Supported from YSoft Card Reader Tool v. 1.4.0
IsEmbed	<p>Allow processing of next rules (until operator "+") only if card number is from embedded reader.</p> <p>Syntax:</p> <ul style="list-style-type: none"> IsEmbed <p>Examples:</p> <ul style="list-style-type: none"> IsEmbed;RightStrip(F) 12345FFFFFFFFF (from embedded terminal) => 12345 123F (from profi terminal) => 123F <p>Server notes:</p> <ul style="list-style-type: none"> No functional limitations

	<p>Reader notes:</p> <ul style="list-style-type: none"> • Always FALSE, so rest of rules until operator "+" are not processed
IsEven	<p>Allow processing of next rules (until operator "+") only if card number length is even.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • IsEven <p>Examples:</p> <ul style="list-style-type: none"> • IsEven;LeftAppend(0) • 12AB => 012AB • 12A => 12A <p>Server notes:</p> <ul style="list-style-type: none"> • No functional limitations <p>Reader notes:</p> <ul style="list-style-type: none"> • No functional limitations
IsEvenEE	<p>Allow processing of next rules only if card number length is even else empty current output and skip to next "+" operator.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • IsEvenEE <p>Examples:</p> <ul style="list-style-type: none"> • IsEvenEE;LeftAppend(0) • 12AB => 012AB • 12A => "" • IsEvenEE;LeftAppend(0) + IsOddEE;LeftAppend(99) • 12AB => 012AB • 12A => 9912A <p>Server notes:</p> <ul style="list-style-type: none"> • NOT IMPLEMENTED <p>Reader notes:</p> <ul style="list-style-type: none"> • Supported from YSoft Card Reader Tool v. 1.4.0
IsLength	<p>Allow processing of next rules (until operator "+") only if card number length is equal to specified value.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • IsLength(value) <p>Example:</p> <ul style="list-style-type: none"> • IsLength(10);SwapPair • 1234567890 => 2143658709 <p>Server notes:</p> <ul style="list-style-type: none"> • No functional limitations

	<p>Reader notes:</p> <ul style="list-style-type: none"> • No functional limitations
IsLengthEE	<p>Allow processing of next rules only if card number length is equal to specified value else empty current output and skip to next "+" operator.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • IsLengthEE(value) <p>Example:</p> <ul style="list-style-type: none"> • IsLengthEE(10);SwapPair • 1234567890 => 2143658709 • 123456789 => "" <p>Server notes:</p> <ul style="list-style-type: none"> • NOT IMPLEMENTED <p>Reader notes:</p> <ul style="list-style-type: none"> • Supported from YSoft Card Reader Tool v. 1.4.0
IsLengthGreater	<p>Allow processing of next rules (until operator "+") only if card number length is greater than specified value.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • IsLengthGreater(value) <p>Examples:</p> <ul style="list-style-type: none"> • IsLengthGreater(5);Substring(5) • 12AB => 12AB • 12345678 => 12345 <p>Server notes:</p> <ul style="list-style-type: none"> • No functional limitations <p>Reader notes:</p> <ul style="list-style-type: none"> • No functional limitations
IsLengthGreaterEE	<p>Allow processing of next rules only if card number length is greater than specified value else empty current output and skip to next "+" operator.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • IsLengthGreaterEE(value) <p>Examples:</p> <ul style="list-style-type: none"> • IsLengthGreaterEE(5);Substring(5) • 12AB => "" • 12345678 => 12345 <p>Server notes:</p> <ul style="list-style-type: none"> • NOT IMPLEMENTED <p>Reader notes:</p> <ul style="list-style-type: none"> • Supported from YSoft Card Reader Tool v. 1.4.0

IsLengthNot	<p>Allow processing of next rules (until operator "+") only if card number length is different from value.</p> <p>Syntax:</p> <ul style="list-style-type: none"> IsLengthNot(value) <p>Examples:</p> <ul style="list-style-type: none"> IsLengthNot(9);LeftAppend(0) 12456789 => 123456789 12345 => 012345 <p>Server notes:</p> <ul style="list-style-type: none"> No functional limitations <p>Reader notes:</p> <ul style="list-style-type: none"> No functional limitations
IsLengthNotEE	<p>Allow processing of next rules only if card number length is different from value else empty current output and skip to next "+" operator.</p> <p>Syntax:</p> <ul style="list-style-type: none"> IsLengthNotEE(value) <p>Examples:</p> <ul style="list-style-type: none"> IsLengthNotEE(9);LeftAppend(0) 12456789 => "" 12345 => 012345 <p>Server notes:</p> <ul style="list-style-type: none"> NOT IMPLEMENTED <p>Reader notes:</p> <ul style="list-style-type: none"> Supported from YSoft Card Reader Tool v. 1.4.0
IsLengthSmaller	<p>Allow processing of next rules (until operator "+") only if card number length is smaller than specified value.</p> <p>Syntax:</p> <ul style="list-style-type: none"> IsLengthSmaller(value) <p>Examples:</p> <ul style="list-style-type: none"> IsLengthSmaller(5);LeftAppend(0) 12AB => 012AB 12345678 => "" <p>Server notes:</p> <ul style="list-style-type: none"> NOT IMPLEMENTED <p>Reader notes:</p> <ul style="list-style-type: none"> Supported from YSoft Card Reader Tool v. 1.4.0

IsLengthSmallerEE	<p>Allow processing of next rules only if card number length is smaller than specified value else empty current output and skip to next "+" operator.</p> <p>Syntax:</p> <ul style="list-style-type: none"> IsLengthSmallerEE(value) <p>Examples:</p> <ul style="list-style-type: none"> IsLengthSmallerEE(5);LeftAppend(0) 12AB => 012AB 12345678 => "" <p>Server notes:</p> <ul style="list-style-type: none"> NOT IMPLEMENTED <p>Reader notes:</p> <ul style="list-style-type: none"> Supported from YSoft Card Reader Tool v. 1.4.0
IsNotCardTypeEE	<p><i>Mask</i> and <i>type</i> are numbers or internal defines CARD_*. To use value from CardType format as hex - prepend "0x". Allow processing of next rules only if placed card type masked by <i>mask</i> (binary AND) is different to <i>type</i> else empty current output and skip to next "+" operator.</p> <p>Syntax:</p> <ul style="list-style-type: none"> IsNotCardTypeEE(type, mask) <p>Examples:</p> <ul style="list-style-type: none"> IsCardTypeEE(0xA00, 0xFF00);LeftAppend(iClass-) + IsNotCardTypeEE(0xA00, 0xFF00);LeftAppend(others-) IsCardTypeEE(CARD_ICLASS, CARD_TYPE_MASK);LeftAppend(iClass-) + IsNotCardTypeEE(CARD_ICLASS, CARD_TYPE_MASK);LeftAppend(others-) iClass 2k: 123456 => iClass-123456 iClass 32k: 123456 => iClass-123456 All others then IClass: 123456 => others-123456 <p>Server notes:</p> <ul style="list-style-type: none"> NOT IMPLEMENTED <p>Reader notes:</p> <ul style="list-style-type: none"> Supported from YSoft Card Reader Tool v. 1.4.0
IsNotStartWith	<p>Allow processing of next rules (until operator "+") only if card number doesn't start with specified string.</p> <p>Syntax:</p> <ul style="list-style-type: none"> IsNotStartWith(string) <p>Examples:</p> <ul style="list-style-type: none"> IsNotStartWith(~);LeftPadding(0,8) ~1234 => ~1234 1234 => 00001234

	<p>Server notes:</p> <ul style="list-style-type: none"> • No functional limitations <p>Reader notes:</p> <ul style="list-style-type: none"> • No functional limitations
IsNotStartWithEE	<p>Allow processing of next rules only if card number doesn't start with specified string else empty current output and skip to next "+" operator.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • IsNotStartWithEE(string) <p>Examples:</p> <ul style="list-style-type: none"> • IsNotStartWithEE(~);LeftPadding(0,8) • ~1234 => "" • 1234 => 00001234 • IsNotStartWithEE(~);LeftPadding(0,8) + IsStartWithEE(~);LeftCut(~);LeftPadding(0,5) • ~1234 => 01234 • 1234 => 00001234 <p>Server notes:</p> <ul style="list-style-type: none"> • NOT IMPLEMENTED <p>Reader notes:</p> <ul style="list-style-type: none"> • Supported from YSoft Card Reader Tool v. 1.4.0
IsOdd	<p>Allow processing of next rules (until operator "+") only if card number length is odd.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • IsOdd <p>Examples:</p> <ul style="list-style-type: none"> • IsOdd;LeftAppend(0) • 12AB => 12AB • 12A => 012A <p>Server notes:</p> <ul style="list-style-type: none"> • NOT IMPLEMENTED <p>Reader notes:</p> <ul style="list-style-type: none"> • Supported from YSoft Card Reader Tool v. 1.4.0

IsOddEE	<p>Allow processing of next rules only if card number length is odd else empty current output and skip to next "+" operator.</p> <p>Syntax:</p> <ul style="list-style-type: none"> IsOddEE <p>Examples:</p> <ul style="list-style-type: none"> IsOddEE;LeftAppend(0) 12AB => "" 12A => 012A IsOddEE;LeftAppend(0) + IsEvenEE;LeftAppend(99) 12AB => 9912AB 12A => 012A <p>Server notes:</p> <ul style="list-style-type: none"> NOT IMPLEMENTED <p>Reader notes:</p> <ul style="list-style-type: none"> Supported from YSoft Card Reader Tool v. 1.4.0
IsStartWith	<p>Allow processing of next rules (until operator "+") only if card number starts with specified string.</p> <p>Syntax:</p> <ul style="list-style-type: none"> IsStartWith(PIN) <p>Examples:</p> <ul style="list-style-type: none"> IsStartWith(PIN);Substring(3,0) PIN1234 => 1234 12345 => 12345 <p>Server notes:</p> <ul style="list-style-type: none"> No functional limitations <p>Reader notes:</p> <ul style="list-style-type: none"> No functional limitations
IsStartWithEE	<p>Allow processing of next rules only if card number starts with specified string else empty current output and skip to next "+" operator.</p> <p>Syntax:</p> <ul style="list-style-type: none"> IsStartWithEE(PIN) <p>Examples:</p> <ul style="list-style-type: none"> IsStartWithEE(PIN);Substring(3,0) PIN1234 => 1234 12345 => "" IsStartWithEE(PIN);Substring(3,0) + IsNotStartWithEE(PIN);Const(CARD) + IsNotStartWithEE(PIN) PIN1234 => 1234 12345 => CARD12345

	Server notes: <ul style="list-style-type: none"> • NOT IMPLEMENTED
	Reader notes: <ul style="list-style-type: none"> • Supported from YSoft Card Reader Tool v. 1.4.0
LeftAppend	Append specified string from left side. Similar functionality has RightAppend. Syntax: <ul style="list-style-type: none"> • LeftAppend(prefix) Example: <ul style="list-style-type: none"> • LeftAppend(YSOFT-) • 12AB => YSOFT-12AB
	Server notes: <ul style="list-style-type: none"> • No functional limitations
	Reader notes: <ul style="list-style-type: none"> • No functional limitations
LeftCut	Cut specified prefix from left. If prefix doesn't match than do nothing. Syntax: <ul style="list-style-type: none"> • LeftCut(prefix) Examples: <ul style="list-style-type: none"> • LeftCut(~1) • ~12AB => 2AB • 12A => 12A
	Server notes: <ul style="list-style-type: none"> • No functional limitations
	Reader notes: <ul style="list-style-type: none"> • No functional limitations
LeftHexShift	Unary bit operation LEFT SHIFT for specified count of bits. Input and output are in hexadecimal format. Count is decimal number. This operation is equivalent to multiplying by 2^{count} Syntax: <ul style="list-style-type: none"> • LeftHexShift(count) Examples: <ul style="list-style-type: none"> • LeftHexShift(1) • 12AB => 2556 • 254 => 4A8
	Server notes: <ul style="list-style-type: none"> • Input and output number must fit to 64-bit signed integer (Should not have more then 16 characters)
	Reader notes: <ul style="list-style-type: none"> • No functional limitations

LeftPadding	<p>Pads with specified sign from left to specified length.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • LeftPadding(sign,length) <p>Example:</p> <ul style="list-style-type: none"> • LeftPadding(0,10) • 1234ABCD => 001234ABCD <p>Server notes:</p> <ul style="list-style-type: none"> • Result will contains integer multiply of sign, so if sign is longer then one character, then result should be longer then specified length <p>Reader notes:</p> <ul style="list-style-type: none"> • Result will contains integer multiply of sign, so if sign is longer then one character, then result should be longer then specified length
LeftShift	<p>Unary bit operation LEFT SHIFT for specified count of bits. Input and output are in decimal format. Count is decimal number. Similar behavior has LeftHexShift. This operation is equivalent to multiplying by 2^{count}</p> <p>Syntax:</p> <ul style="list-style-type: none"> • LeftShift(count) <p>Example:</p> <ul style="list-style-type: none"> • LeftShift(1) • 128 => 256 <p>Server notes:</p> <ul style="list-style-type: none"> • Input and output number must fit to 64-bit signed integer (Should not have more then 19 characters) <p>Reader notes:</p> <ul style="list-style-type: none"> • No functional limitations
LeftStrip	<p>Strips specified sign from left.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • LeftStrip(sign) <p>Examples:</p> <ul style="list-style-type: none"> • LeftStrip(0) • 000012AB => 12AB • 00000254 => 254 <p>Server notes:</p> <ul style="list-style-type: none"> • No functional limitations <p>Reader notes:</p> <ul style="list-style-type: none"> • No functional limitations

LowerCase	<p>Convert alphabetical sign to its lowercase representation.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • LowerCase <p>Example:</p> <ul style="list-style-type: none"> • LowerCase • CARD123 => card123 <p>Server notes:</p> <ul style="list-style-type: none"> • No functional limitations <p>Reader notes:</p> <ul style="list-style-type: none"> • No functional limitations
LRC	<p>Computes Longitudinal Redundancy Check http://en.wikipedia.org/wiki/Longitudinal_redundancy_check and adds it to the end. Input must be in hexadecimal format and have even length. LRC is in lowercase.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • LRC <p>Example:</p> <ul style="list-style-type: none"> • LRC • 01044F24CC => 01044F24CCa2 <p>Server notes:</p> <ul style="list-style-type: none"> • No functional limitations <p>Reader notes:</p> <ul style="list-style-type: none"> • No functional limitations
MD5	<p>Computes MD5 hash of input.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • MD5 <p>Example:</p> <ul style="list-style-type: none"> • MD5 • 1234 => 81dc9bdb52d04dc20036dbd8313ed055 <p>Server notes:</p> <ul style="list-style-type: none"> • No functional limitations <p>Reader notes:</p> <ul style="list-style-type: none"> • NOT IMPLEMENTED

Replace	<p>Replaces all occurrences of one sequence in input string with another one.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • Replace(source) – only removes specified source (replace with empty string) • Replace(source,dest) – replace specified source with dest <p>Examples:</p> <ul style="list-style-type: none"> • Replace(~,0) • ~1234~ => 012340 • 12~34 => 12034 <p>Server notes:</p> <ul style="list-style-type: none"> • No functional limitations <p>Reader notes:</p> <ul style="list-style-type: none"> • No functional limitations
Reverse2	<p>Byte reverse - it is useful for hexadecimal input because 2 signs represent one byte, but other character are allowed.. Therefore this operation makes reverse string by pair. Even length is necessary. Input with odd length is left unmodified.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • Reverse2 <p>Example:</p> <ul style="list-style-type: none"> • Reverse2 • 12345678 => 78563412 <p>Server notes:</p> <ul style="list-style-type: none"> • No functional limitations <p>Reader notes:</p> <ul style="list-style-type: none"> • No functional limitations
Reverse	<p>Reverse of string.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • Reverse <p>Example:</p> <ul style="list-style-type: none"> • Reverse • 12345678 => 87654321 <p>Server notes:</p> <ul style="list-style-type: none"> • No functional limitations <p>Reader notes:</p> <ul style="list-style-type: none"> • No functional limitations

RightAppend	<p>This function is similar to LeftAppend. Append specified string from right side.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • RightAppend(suffix) <p>Example:</p> <ul style="list-style-type: none"> • RightAppend(-YSOFT) • 12AB => 12AB-YSOFT <p>Server notes:</p> <ul style="list-style-type: none"> • No functional limitations <p>Reader notes:</p> <ul style="list-style-type: none"> • No functional limitations
RightHexShift	<p>Unary bit operation RIGHT SHIFT for specified count of bits. Input and output are in hexadecimal format. Count is decimal number. This operation is equivalent to dividing by 2^{count}</p> <p>Syntax:</p> <ul style="list-style-type: none"> • RightHexShift(count) <p>Examples:</p> <ul style="list-style-type: none"> • RightHexShift(1) • 12AB => 955 • 254 => 12A <p>Server notes:</p> <ul style="list-style-type: none"> • Input and output number must fit to 64-bit signed integer (Should not have more then 16 characters) <p>Reader notes:</p> <ul style="list-style-type: none"> • No functional limitations
RightPadding	<p>Pads with specified sign from right to specified length.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • RightPadding(sign,length) <p>Example:</p> <ul style="list-style-type: none"> • RightPadding(F,10) • 1234ABCD => 1234ABCDFF <p>Server notes:</p> <ul style="list-style-type: none"> • Result will contains integer multiply of sign, so if sign is longer then one character, then result should be longer then specified length <p>Reader notes:</p> <ul style="list-style-type: none"> • Result will contains integer multiply of sign, so if sign is longer then one character, then result should be longer then specified length

RightShift	<p>Unary bit operation RIGHT SHIFT for specified count of bits. Input and output are in decimal format. Count is decimal number. Similar behavior has RightHexShift. This operation is equivalent to dividing by 2^{count}</p> <p>Syntax:</p> <ul style="list-style-type: none"> • RightShift(count) <p>Example:</p> <ul style="list-style-type: none"> • RightShift(1) • 256 => 128 <p>Server notes:</p> <ul style="list-style-type: none"> • Input and output number must fit to 64-bit signed integer (Should not have more then 19 characters) <p>Reader notes:</p> <ul style="list-style-type: none"> • No functional limitations
RightStrip	<p>Strips specified sign from right.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • RightStrip(sign) <p>Example:</p> <ul style="list-style-type: none"> • RightStrip(F) • 30344142FFFFFFFFFFFFFFFF => 30344142 <p>Server notes:</p> <ul style="list-style-type: none"> • No functional limitations <p>Reader notes:</p> <ul style="list-style-type: none"> • No functional limitations
SignReverse	<p>This conversion takes every string in hexadecimal format and makes its binary reverse. For example (5 is represented in binary as 0101, reverse transfer it into 1010 that is A)</p> <p>Syntax:</p> <ul style="list-style-type: none"> • SignReverse <p>Example:</p> <ul style="list-style-type: none"> • SignReverse • 0123456789ABCDEF => 084C2A6E195D3B7F <p>Server notes:</p> <ul style="list-style-type: none"> • Input string must be in upper case • All non hexadecimal characters are translated as "null" <p>Reader notes:</p> <ul style="list-style-type: none"> • No functional limitations

Substring	<p>Selects substring of input. If any argument is negative then it is used from right side (from end).</p> <p>Syntax:</p> <ul style="list-style-type: none"> • Substring(n) • Substring(start,end) <p>Examples:</p> <ul style="list-style-type: none"> • Substring(5) <ul style="list-style-type: none"> • 1234567890 => 12345 • Substring(-5) <ul style="list-style-type: none"> • 1234567890 => 67890 • Substring(3,5) <ul style="list-style-type: none"> • 1234567890 => 456 • 123ABCDE => ABC • Substring(3,0) <ul style="list-style-type: none"> • 1234567890 => 4567890 • 123ABCDE => ABCDE • Substring(2,-2) <ul style="list-style-type: none"> • 1234567890 => 345678 • 123ABCDE => 3ABC • Substring(-7,-2) <ul style="list-style-type: none"> • 1234567890 => 45678 • 123ABCDE => 23ABC <p>Server notes:</p> <ul style="list-style-type: none"> • No functional limitations <p>Reader notes:</p> <ul style="list-style-type: none"> • No functional limitations
Swap12785634	<p>Swap 4th byte with 2nd. It is useful only for hexadecimal format. Input string could have minimally 8 signs. Rest after first 8 signs is discarded.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • Swap12785634 <p>Example:</p> <ul style="list-style-type: none"> • Swap12785634 • 12345678 => 12785634 <p>Server notes:</p> <ul style="list-style-type: none"> • No functional limitations <p>Reader notes:</p> <ul style="list-style-type: none"> • No functional limitations

SwapPair	Swaps even and odd signs. Input string could have even length. Input with odd length is left unconverted. Syntax: <ul style="list-style-type: none">• SwapPair Example: <ul style="list-style-type: none">• SwapPair• 123456 => 214365
	Server notes: <ul style="list-style-type: none">• No functional limitations
	Reader notes: <ul style="list-style-type: none">• No functional limitations
UpperCase	Convert alphabetical sign to its uppercase representation. Syntax: <ul style="list-style-type: none">• UpperCase Example: <ul style="list-style-type: none">• UpperCase• card123 => CARD123
	Server notes: <ul style="list-style-type: none">• No functional limitations
	Reader notes: <ul style="list-style-type: none">• No functional limitations

7.4 THE MOST COMMON CARD NUMBER CONVERSIONS

Examples of the most commonly used conversions for different technologies:

Technology	Protocol setting	Example of card number read	Expected Card Number / Number On Card	Card manager
LF:				
EM4k and compatible	LF UIN Protocol 90 - EM4000 compat.	0F0244B41B	064462566427	Hex2Dec;LeftPadding(0,12)
			0F0422D28D	SignReverse
			64493900429	SignReverse;Hex2Dec;
			0069390989	Substring(2,0);SignReverse;Hex2Dec;LeftPadding(0,10)
			1075981784	Substring(2,0);SwapPair;SignReverse;Hex2Dec
			30463034323244323844	SignReverse;Hex2ASCII
			15000400020202131308	SignReverse;SwapPair;Hex2DecValue

			69390989	Substring(3,0);SignReverse;Hex2Dec
			0004502555	Substring(4,10);Hex2Dec;LeftPadding(0, 10)
			06846107	Substring(4,6);Hex2Dec;LeftPadding(0, 3) + Substring(6,10);Hex2Dec;LeftPadding(0,5)
			0038056987	Substring(2,0);Hex2Dec;LeftPadding(0, 10)
			0058046107	Substring(2,6);Hex2Dec;LeftPadding(0, 5) + Substring(6,10);Hex2Dec;LeftPadding(0,5)
Proxlite	LF UIN Protocol 91 - Proxlite/Casi- Rusco	12D1009BC6		Not known/Not enough data
Paradox	LF UIN Protocol 92 - Paradox/ PosiProx	0060FB47	64327	HexAnd(FFFF);Hex2Dec;LeftPadding(0, 5)
Nedap	LF UIN Protocol 89 - Nedap	125290407100185 7	1857	Substring(-6);LeftStrip(0)
Awid	LF UIN Protocol 87 - Awid	22EE000034F50	108456	HexAnd(FFFFFFF);RightHexShift(1);Hex2Dec
Pyramid	LF UIN Protocol 86 - Pyramid	2123AC3	7521	RightHexShift(1);HexAnd(FFFF);Hex2Dec
Pyramid MAX	LF UIN Protocol 86 - Pyramid	2B44CDA	9837	RightHexShift(1);HexAnd(FFFF);Hex2Dec
Desiter	LF UIN Protocol 85 - Deister	C8DC5FFB84		Not known/Not enough data
Datasec	LF UIN Protocol 84 - Datasec	03C86800B12B		Not known/Not enough data

G-Prox	LF UIN Protocol 83 - G-Prox	214361E		Not known/Not enough data
Ioprox	LF UIN Protocol 82 - ioProx label	014E48850	48850	Substring(4,0)
PAC	LF UIN Protocol 96 - PAC/KeyPAC	08948E53D		Not known/Not enough data
KeyPAC	Protocol 96 - PAC/KeyPAC	59465E8B		Not known/Not enough data
URMET	LF UIN Protocol 120 - URMET/FDI	03044233	03044233	No conversion needed
Cardax	LF UIN Protocol 1601 - Cardax	0D8F148D0D0D0D 0C		Not known/Not enough data
Jablotron	LF UIN Protocol 1602 - Jablotron	0000108975	108975	Substring(4,0)
Noralsy	MFX Only LF UIN Protocol 1606 - Noralsy	9320282	9320282	No conversion needed
Keri	MFX Only LF UIN Protocol 1603 - Keri	330E7A	3346042	Hex2Dec
IDTECK	MFX Only LF UIN Protocol 1604 - IDTECK	80FA74B1		No conversion possible

Nexkey	MFX Only LF UIN Protocol 1605 - NexKey	053ED5DC	88004060	Hex2Dec
	MFX Only Protocol 1607 - NexKey compat. only	88004060	88004060	No conversion needed
ISO FDX-B	MFX Only LF UIN	01009F4B2C6A1E5 9		Not known/Not enough data
HID Prox	only readers with HID Prox support LF UIN Protocol 35 - HID Prox	01AFC1E	32271	Hex2Dec;RightShift(1);HexAnd(FFFF);LeftPadding(0,5)
		00001A08421086 01087F5	001000	RightHexShift(36);HexAnd(F);UpperCase;SignReverse + RightHexShift(31);HexAnd(F);UpperCase;SignReverse + RightHexShift(26);HexAnd(F);UpperCase;SignReverse + RightHexShift(21);HexAnd(F);UpperCase;SignReverse + RightHexShift(16);HexAnd(F);UpperCase;SignReverse + RightHexShift(11);HexAnd(F);UpperCase;SignReverse
		007E608402	16897	HexAnd(3FFFF);RightHexShift(1);Hex2Dec
Indala ASP	MFX, Indala card readers Indala UIN	32 bit		There is no simple conversion between UIN and printed number. Conversion includes bit swapping.
	Protocol 1105 - Indala W26 - format 10022	BC6A33	27187	HexAnd(FFFF);Hex2Dec;LeftPadding(0,5)
	Protocol 1106 - Indala W26 Wallb.- format 10022	378D467	27187	RightHexShift(1);HexAnd(FFFF);Hex2Dec;LeftPadding(0,5)
	Protocol 1108 - Indala W27 Wallb.- format 10251	07D48CC 07D48CE	52252 52254	

	Protocol 1109 - Indala ABA2 - format 11037	0010304543	04543	Substring(5,0)
	Protocol 1110 - Indala Lite - format 17531	B4041B	01051	HexAnd(FFFF);Hex2Dec;LeftPadding(0,5)
Indala ASP+	MFX, Indala card readers Indala UIN	128 bit		Conversion depends on the card format.
	Protocol 1130 - Indala ASP+ default key Wiegand	050083C	0004001054	RightHexShift(17);HexAnd(FF);Hex2Dec;LeftPadding(0,5) + RightHexShift(1);HexAnd(FFFF);Hex2Dec;LeftPadding(0,5)
Hitag 1 / 2 / S EM4050	LF UIN (MFX) Hitag 1, 2, S protocols (MFX, Unique)	005500D896	5500D896	Substring(2,0)
			1426118806	Hex2Dec
Tiris	LF UIN (MFX Only) Protocol 22 - Tiris	00000000085B9BD5		Not known/Not enough data
Cotag	LF UIN (MFX Only) Protocol 23 - Cotag	81AE04DC00000008		Not known/Not enough data
HF:				
ISO14443A / ISO14443B / ISO15693 / Generic HF UIN	HF UIN	9EB37AA3	2742727582	Reverse2;Hex2Dec
			2662562467	Hex2Dec
			A37AB39E	Reverse2

HID iClass / HID iClass SE / HID iClass SEOS	HF UIN	No relation between UIN and printed number. SEOS cards have random UIN.		
	Protocol 1210 - HID	23E082B	01045	Hex2Dec;RightShift(1);DecimalAnd(65535);LeftPadding(0,5)
	iClass + HID SIO (Only readers with SE processor)	2F64E27	10003	RightHexShift(1);HexAnd(FFFF);Hex2Dec;LeftPadding(0,5)

8 READER CONFIGURATIONS

8.1 B-015 - READER 3 LF

Reader description	Universal LF reader supporting ASK and FSK modulation	
Reader in production	Yes	
Reader technology	Contactless	
Reader frequency	125kHz	
Notes		
On LF UIN mode it may take longer to read the card for the first time. Protocol 117 should not be used for user identification, it is a special protocol intended as an aid for card testing and some limitations may apply.		
Supported cfg items	Supported values	Note
rdrcfg.common.cardno_conv	(CARDCONV_APPENDLRC, CARDCONV_HEX2DEC, CARDCONV_HEX2DEC10, CARDCONV_HEX2DECLEFT16, CARDCONV_INTERFLEX, CARDCONV_ISO14443ATRUNCREVENDIAN, CARDCONV_JAVACARDSERIAL, CARDCONV_REVBITSINBYTE, CARDCONV_REVENDIAN, CARDCONV_REVENDIAN2DEC, CARDCONV_REVENDIANHEX2DEC10, CARDCONV_REVENDIANWOHIDPROX)	Builtin card number conversions. Will be overwritten if rdrcfg.cardno.cardnoconv specified
rdrcfg.cardno.cardnoconv		Custom specified card number conversion. CM2CNO("text") or CNOASM(FILE("filename"))).
rdrcfg.common.continuous	(0, 1)	Special setting that enables card requested for login and card remove notification on some products.

Supported cfg items	Supported values	Note
rdrcfg.common.testing_mode	(0, 1)	Special testing setting to scan all supported technologies. Only for debug or testing purposes, DO NOT USE IN PRODUCTION ENVIRONMENT, some limitations may apply.
rdrcfg.lf.lf_supported	(LF_AUTO, LF_EM4000, LF_PROXLITE, LF_PARADOX, LF_NEDAP, LF_AWID, LF_PYRAMID, LF_DEISTER, LF_DATASEC, LF_GPROX, LF_IOPROX, LF_EM4050RO, LF_PAC, LF_URMET, LF_CARDAX, LF_JABLOTRON, LF_CUSTOM)	
rdrcfg.lf.lf_supported_multiple	[LF_AUTO, LF_EM4000, LF_PROXLITE, LF_PARADOX, LF_NEDAP, LF_AWID, LF_PYRAMID, LF_DEISTER, LF_DATASEC, LF_GPROX, LF_IOPROX, LF_EM4050RO, LF_PAC, LF_URMET, LF_CARDAX, LF_JABLOTRON]	Combining other technologies with LF_CUSTOM is limited.
rdrcfg.lf.lf_custom.modulation	(LF_ASK, LF_FSK2)	Modulation type
rdrcfg.lf.lf_custom.bitrate	(LF_F16, LF_F32, LF_F50, LF_F64)	Card bitrate
rdrcfg.lf.lf_custom.start_decoding	(LF_DECODE_NRZ, LF_DECODE_MANCHESTER, LF_DECODE_DIFFBIPHASE)	Decoding of start condition
rdrcfg.lf.lf_custom.data_decoding	(LF_DECODE_NRZ, LF_DECODE_MANCHESTER, LF_DECODE_DIFFBIPHASE)	Decoding of data
rdrcfg.lf.lf_custom.start_cond	u32	32 bit start condition
rdrcfg.lf.lf_custom.start_mask	u32	32 bit Start condition mask. Must be continuous sequence of bits on ASKFSK frontend
rdrcfg.lf.lf_custom.count_bits	int	Data bits count without start condition. If you do not specify complete bith length, then adjust ignore_bits for complete data reading

Supported cfg items		Supported values	Note
rdrcfg.lf.lf_custom.start_decode_invert		(0, 1)	Invert start condition data. Not applicable if LF_DECODE_DIFFBI PHASE is used
rdrcfg.lf.lf_custom.data_decode_invert		(0, 1)	Invert data. Not applicable if LF_DECODE_DIFFBI PHASE is used
rdrcfg.lf.lf_custom.input_invert		(0, 1)	Invert data incoming to decodes
rdrcfg.lf.lf_custom.read_count		int	How many times the data should be read and checked and compared as identical
rdrcfg.lf.lf_custom.padding_bits		int	Padding 0 bits before actual data output
rdrcfg.lf.lf_custom.ignore_bits		int	Bits at the end of the data stream that should be ignored (i.e. void bits), used for proper timing calculation
rdrcfg.lf.lf_custom.card_bootup_time		int	Value to add to card timeout - card bootup overhead in us
rdrcfg.lf.lf_custom.decode		(LF_DECODE_RAW, LF_DECODE_EM4K)	Use particular decode function
rdrcfg.lf.lf_custom.remove_timeout		uint	Timeout for card remove
rdrcfg.lf.lf_custom.card_type		uint	Card type to report
Default configuration equivalent			
rdrcfg.lf.lf_supported = LF_AUTO;			
Protocol ID and name	Configuration used		
Protocol 1220 - LF UIN	rdrcfg.lf.lf_supported = LF_AUTO		
Protocol 90 - EM4000 compatible	rdrcfg.lf.lf_supported = LF_EM4000;		

Protocol ID and name	Configuration used
Protocol 94 - EM4000 compatible cont.	<code>rdrcfg.common.continuous = 1;</code> <code>rdrcfg.lf.lf_supported = LF_EM4000;</code>
Protocol 88 - EM4000 compatible w/LRC (96008N1 compatible)	<code>rdrcfg.common.cardno_conv = CARDCONV_APPENDLRC;</code> <code>rdrcfg.lf.lf_supported = LF_EM4000;</code>
Protocol 91 - Proxlite/Casi-Rusco	<code>rdrcfg.lf.lf_supported = LF_PROXLITE;</code>
Protocol 92 - Paradox/PosiProx	<code>rdrcfg.lf.lf_supported = LF_PARADOX;</code>
Protocol 89 - Nedap	<code>rdrcfg.lf.lf_supported = LF_NEDAP;</code>
Protocol 87 - Awid	<code>rdrcfg.lf.lf_supported = LF_AWID;</code>
Protocol 86 - Pyramid	<code>rdrcfg.lf.lf_supported = LF_PYRAMID;</code>
Protocol 85 - Deister	<code>rdrcfg.lf.lf_supported = LF_DEISTER;</code>
Protocol 84 - Datasec	<code>rdrcfg.lf.lf_supported = LF_DATASEC;</code>
Protocol 83 - G-Prox	<code>rdrcfg.lf.lf_supported = LF_GPROX;</code>
Protocol 82 - ioProx label	<code>rdrcfg.lf.lf_supported = LF_IOPROX;</code>
Protocol 121 - EM4050 f/64 std. read mode	<code>rdrcfg.lf.lf_supported = LF_EM4050R0;</code>
Protocol 96 - PAC/KeyPAC	<code>rdrcfg.lf.lf_supported_multiple.extend([LF_PAC, LF_KEYPAC]);</code>
Protocol 120 - URMET/FDI	<code>rdrcfg.lf.lf_supported = LF_URMET;</code>
Protocol 1601 - Cardax/Gallagher	<code>rdrcfg.lf.lf_supported = LF_CARDAX;</code>
Protocol 1602 - Jablotron	<code>rdrcfg.lf.lf_supported = LF_JABLOTRON;</code>
<i>Protocol 117 - ASK/FSK testing</i>	<code>rdrcfg.common.testing_mode = 1;</code> <code>rdrcfg.common.exact_cardtype = 1;</code> <code>rdrcfg.sam.sam_mode = SAM_AUTO;</code>

8.2 B-016 - 125KHZ ASK FSK

Reader description	Universal LF reader supporting ASK and FSK modulation	
Reader in production	Yes	
Reader technology	Contactless	
Reader frequency	125kHz	
Notes		
On LF UIN mode it may take longer to read the card for the first time. Protocol 117 should not be used for user identification, it is a special protocol intended as an aid for card testing and some limitations may apply.		
Supported cfg items	Supported values	Note
rdrcfg.common.cardno_conv	(CARDCONV_APPENDLRC, CARDCONV_HEX2DEC, CARDCONV_HEX2DEC10, CARDCONV_HEX2DECLEFT16, CARDCONV_INTERFLEX, CARDCONV_ISO14443ATRUNCREVENDIAN, CARDCONV_JAVACARDSERIAL, CARDCONV_REVBITSINBYTE, CARDCONV_REVENDIAN, CARDCONV_REVENDIAN2DEC, CARDCONV_REVENDIANHEX2DEC10, CARDCONV_REVENDIANWOHIDPROX)	Builtin card number conversions. Will be overwritten if rdrcfg.cardno.cardnoconv specified
rdrcfg.cardno.cardnoconv		Custom specified card number conversion. CM2CNO("text") or CNOASM(FILE("filename")).
rdrcfg.common.continuous	(0, 1)	Special setting that enables card requested for login and card remove notification on some products.
rdrcfg.common.testing_mode	(0, 1)	Special testing setting to scan all supported technologies. Only for debug or testing purposes, DO NOT USE IN PRODUCTION ENVIRONMENT, some limitaions may apply.

Supported cfg items	Supported values	Note
rdrcfg.lf.lf_supported	(LF_AUTO, LF_EM4000, LF_PROXLITE, LF_PARADOX, LF_NEDAP, LF_AWID, LF_PYRAMID, LF_DEISTER, LF_DATASEC, LF_GPROX, LF_IOPROX, LF_EM4050RO, LF_PAC, LF_URMET, LF_CARDAX, LF_JABLOTRON, LF_CUSTOM)	
rdrcfg.lf.lf_supported_multiple	[LF_AUTO, LF_EM4000, LF_PROXLITE, LF_PARADOX, LF_NEDAP, LF_AWID, LF_PYRAMID, LF_DEISTER, LF_DATASEC, LF_GPROX, LF_IOPROX, LF_EM4050RO, LF_PAC, LF_URMET, LF_CARDAX, LF_JABLOTRON]	Combining other technologies with LF_CUSTOM is limited.
rdrcfg.lf.lf_custom.modulation	(LF_ASK, LF_FSK2)	Modulation type
rdrcfg.lf.lf_custom.bitrate	(LF_F16, LF_F32, LF_F50, LF_F64)	Card bitrate
rdrcfg.lf.lf_custom.start_decoding	(LF_DECODE_NRZ, LF_DECODE_MANCHESTER, LF_DECODE_DIFFBIPHASE)	Decoding of start condition
rdrcfg.lf.lf_custom.data_decoding	(LF_DECODE_NRZ, LF_DECODE_MANCHESTER, LF_DECODE_DIFFBIPHASE)	Decoding of data
rdrcfg.lf.lf_custom.start_cond	u32	32 bit start condition
rdrcfg.lf.lf_custom.start_mask	u32	32 bit Start condition mask. Must be continuous sequence of bits on ASKFSK frontend
rdrcfg.lf.lf_custom.count_bits	int	Data bits count without start condition. If you do not specify complete bith length, then adjust ignore_bits for complete data reading
rdrcfg.lf.lf_custom.start_decode_invert	(0, 1)	Invert start condition data. Not applicable if LF_DECODE_DIFFBI PHASE is used
rdrcfg.lf.lf_custom.data_decode_invert	(0, 1)	Invert data. Not applicable if LF_DECODE_DIFFBI PHASE is used

Supported cfg items	Supported values	Note
rdrcfg.lf.lf_custom.input_invert	(0, 1)	Invert data incoming to decodes
rdrcfg.lf.lf_custom.read_count	int	How many times the data should be read and checked and compared as identical
rdrcfg.lf.lf_custom.padding_bits	int	Padding 0 bits before actual data output
rdrcfg.lf.lf_custom.ignore_bits	int	Bits at the end of the data stream that should be ignored (i.e. void bits), used for proper timing calculation
rdrcfg.lf.lf_custom.card_bootup_time	int	Value to add to card timeout - card bootup overhead in us
rdrcfg.lf.lf_custom.decode	(LF_DECODE_RAW, LF_DECODE_EM4K)	Use particular decode function
rdrcfg.lf.lf_custom.remove_timeout	uint	Timeout for card remove
rdrcfg.lf.lf_custom.card_type	uint	Card type to report
Default configuration equivalent		
rdrcfg.lf.lf_supported = LF_AUTO;		
Protocol ID and name	Configuration used	
Protocol 1220 - LF UIN	rdrcfg.lf.lf_supported = LF_AUTO	
Protocol 90 - EM4000 compatible	rdrcfg.lf.lf_supported = LF_EM4000;	
Protocol 94 - EM4000 compatible cont.	rdrcfg.common.continuous = 1; rdrcfg.lf.lf_supported = LF_EM4000;	
Protocol 88 - EM4000 compatible w/LRC (96008N1 compatible)	rdrcfg.common.cardno_conv = CARDCONV_APPENDLRC; rdrcfg.lf.lf_supported = LF_EM4000;	
Protocol 91 - Proxlite/Casi-Rusco	rdrcfg.lf.lf_supported = LF_PROXLITE;	

Protocol ID and name	Configuration used
Protocol 92 - Paradox/PosiProx	<code>rdrcfg.lf.lf_supported = LF_PARADOX;</code>
Protocol 89 - Nedap	<code>rdrcfg.lf.lf_supported = LF_NEDAP;</code>
Protocol 87 - Awid	<code>rdrcfg.lf.lf_supported = LF_AWID;</code>
Protocol 86 - Pyramid	<code>rdrcfg.lf.lf_supported = LF_PYRAMID;</code>
Protocol 85 - Deister	<code>rdrcfg.lf.lf_supported = LF_DEISTER;</code>
Protocol 84 - Datasec	<code>rdrcfg.lf.lf_supported = LF_DATASEC;</code>
Protocol 83 - G-Prox	<code>rdrcfg.lf.lf_supported = LF_GPROX;</code>
Protocol 82 - ioProx label	<code>rdrcfg.lf.lf_supported = LF_IOPROX;</code>
Protocol 121 - EM4050 f/64 std. read mode	<code>rdrcfg.lf.lf_supported = LF_EM4050R0;</code>
Protocol 96 - PAC/KeyPAC	<code>rdrcfg.lf.lf_supported_multiple.extend([LF_PAC, LF_KEYPAC]);</code>
Protocol 120 - URMET/FDI	<code>rdrcfg.lf.lf_supported = LF_URMET;</code>
Protocol 1601 - Cardax/Gallagher	<code>rdrcfg.lf.lf_supported = LF_CARDAX;</code>
Protocol 1602 - Jablotron	<code>rdrcfg.lf.lf_supported = LF_JABLOTRON;</code>
<i>Protocol 117 - ASK/FSK testing</i>	<code>rdrcfg.common.testing_mode = 1;</code> <code>rdrcfg.common.exact_cardtype = 1;</code> <code>rdrcfg.sam.sam_mode = SAM_AUTO;</code>

8.3 B-017 - MULTIREADER LF

Reader description	Universal 125kHz reader supporting ASK and FSK modulation, includes HID Prox decoding
Reader in production	Yes
Reader technology	Contactless
Reader frequency	125kHz

Notes

On LF UIN mode it may take longer to read the card for the first time.
Protocol 117 should not be used for user identification, it is a special protocol intended as an aid for card testing and some limitations may apply.

Supported cfg items	Supported values	Note
rdrcfg.common.cardno_conv	(CARDCONV_APPENDLRC, CARDCONV_HEX2DEC, CARDCONV_HEX2DEC10, CARDCONV_HEX2DECLEFT16, CARDCONV_INTERFLEX, CARDCONV_ISO14443ATRUNCREVENDIAN, CARDCONV_JAVACARDSERIAL, CARDCONV_REVBITSINBYTE, CARDCONV_REVENDIAN, CARDCONV_REVENDIAN2DEC, CARDCONV_REVENDIANHEX2DEC10, CARDCONV_REVENDIANWOHIDPROX)	Builtin card number conversions. Will be overwritten if rdrcfg.cardno.cardnoconv specified
rdrcfg.cardno.cardnoconv		Custom specified card number conversion. CM2CNO("text") or CNOASM(FILE("filename"))).
rdrcfg.common.continuous	(0, 1)	Special setting that enables card requested for login and card remove notification on some products.
rdrcfg.common.testing_mode	(0, 1)	Special testing setting to scan all supported technologies. Only for debug or testing purposes, DO NOT USE IN PRODUCTION ENVIRONMENT, some limitations may apply.
rdrcfg.lf.lf_supported	(LF_AUTO, LF_EM4000, LF_PROXLITE, LF_PARADOX, LF_NEDAP, LF_AWID, LF_PYRAMID, LF_DEISTER, LF_DATASEC, LF_GPROX, LF_IOPROX, LF_EM4050RO, LF_PAC, LF_URMET, LF_CARDAX, LF_JABLOTRON, LF_HIDPROX, LF_CUSTOM)	

Supported cfg items	Supported values	Note
rdrcfg.lf.lf_supported_multiple	[LF_AUTO, LF_EM4000, LF_PROXLITE, LF_PARADOX, LF_NEDAP, LF_AWID, LF_PYRAMID, LF_DEISTER, LF_DATASEC, LF_GPROX, LF_IOPROX, LF_EM4050R0, LF_PAC, LF_URMET, LF_CARDAX, LF_JABLOTRON, LF_HIDPROX]	Combining other technologies with LF_CUSTOM is limited.
rdrcfg.lf.hidprox.wiegand_format	(WAUTO, W26, W27, W32, W34, W37, W40, W40_PAR, W44, WABA2, WMAGSTRIPE)	
rdrcfg.lf.hidprox.hid_udf_enable	(0, 1)	Enable processing of HID Prox User Defined Formats - cards with non-standard customer code
rdrcfg.lf.hidprox.no_wiegand_parity_check	(0, 1)	Do not check parity on wiegand formats that contain parity
rdrcfg.lf.hidprox.no_error_report	(0, 1)	Do not report parity or other errors
rdrcfg.lf.lf_custom.modulation	(LF_ASK, LF_FSK2)	Modulation type
rdrcfg.lf.lf_custom.bitrate	(LF_F16, LF_F32, LF_F50, LF_F64)	Card bitrate
rdrcfg.lf.lf_custom.start_decoding	(LF_DECODE_NRZ, LF_DECODE_MANCHESTER, LF_DECODE_DIFFBIPHASE)	Decoding of start condition
rdrcfg.lf.lf_custom.data_decoding	(LF_DECODE_NRZ, LF_DECODE_MANCHESTER, LF_DECODE_DIFFBIPHASE)	Decoding of data
rdrcfg.lf.lf_custom.start_cond	u32	32 bit start condition
rdrcfg.lf.lf_custom.start_mask	u32	32 bit Start condition mask. Must be continuous sequence of bits on ASKFSK frontend
rdrcfg.lf.lf_custom.count_bits	int	Data bits count without start condition. If you do not specify complete bith length, then adjust ignore_bits for complete data reading

Supported cfg items		Supported values	Note
rdrcfg.lf.lf_custom.start_decode_invert		(0, 1)	Invert start condition data. Not applicable if LF_DECODE_DIFFBI PHASE is used
rdrcfg.lf.lf_custom.data_decode_invert		(0, 1)	Invert data. Not applicable if LF_DECODE_DIFFBI PHASE is used
rdrcfg.lf.lf_custom.input_invert		(0, 1)	Invert data incoming to decodes
rdrcfg.lf.lf_custom.read_count		int	How many times the data should be read and checked and compared as identical
rdrcfg.lf.lf_custom.padding_bits		int	Padding 0 bits before actual data output
rdrcfg.lf.lf_custom.ignore_bits		int	Bits at the end of the data stream that should be ignored (i.e. void bits), used for proper timing calculation
rdrcfg.lf.lf_custom.card_bootup_time		int	Value to add to card timeout - card bootup overhead in us
rdrcfg.lf.lf_custom.decode		(LF_DECODE_RAW, LF_DECODE_EM4K)	Use particular decode function
rdrcfg.lf.lf_custom.remove_timeout		uint	Timeout for card remove
rdrcfg.lf.lf_custom.card_type		uint	Card type to report
Default configuration equivalent			
rdrcfg.lf.lf_supported = LF_AUTO;			
Protocol ID and name	Configuration used		
Protocol 1220 - LF UIN	rdrcfg.lf.lf_supported = LF_AUTO		

Protocol ID and name	Configuration used
Protocol 125 - EM4000 compat. + HID Prox Wallb.	<code>rdrcfg.lf.lf_supported_multiple.extend([LF_EM4000, LF_HIDPROX]);</code>
Protocol 90 - EM4000 compatible	<code>rdrcfg.lf.lf_supported = LF_EM4000;</code>
Protocol 94 - EM4000 compatible cont.	<code>rdrcfg.common.continuous = 1;</code> <code>rdrcfg.lf.lf_supported = LF_EM4000;</code>
Protocol 88 - EM4000 compatible w/LRC (96008N1 compatible)	<code>rdrcfg.common.cardno_conv = CARDCONV_APPENDLRC;</code> <code>rdrcfg.lf.lf_supported = LF_EM4000;</code>
Protocol 91 - Proxlite/Casi-Rusco	<code>rdrcfg.lf.lf_supported = LF_PROXLITE;</code>
Protocol 92 - Paradox/PosiProx	<code>rdrcfg.lf.lf_supported = LF_PARADOX;</code>
Protocol 89 - Nedap	<code>rdrcfg.lf.lf_supported = LF_NEDAP;</code>
Protocol 87 - Awid	<code>rdrcfg.lf.lf_supported = LF_AWID;</code>
Protocol 86 - Pyramid	<code>rdrcfg.lf.lf_supported = LF_PYRAMID;</code>
Protocol 85 - Deister	<code>rdrcfg.lf.lf_supported = LF_DEISTER;</code>
Protocol 84 - Datasec	<code>rdrcfg.lf.lf_supported = LF_DATASEC;</code>
Protocol 83 - G-Prox	<code>rdrcfg.lf.lf_supported = LF_GPROX;</code>
Protocol 82 - ioProx label	<code>rdrcfg.lf.lf_supported = LF_IOPROX;</code>
Protocol 121 - EM4050 f/64 std. read mode	<code>rdrcfg.lf.lf_supported = LF_EM4050R0;</code>
Protocol 96 - PAC/KeyPAC	<code>rdrcfg.lf.lf_supported_multiple.extend([LF_PAC, LF_KEYPAC]);</code>
Protocol 120 - URMET/FDI	<code>rdrcfg.lf.lf_supported = LF_URMET;</code>
Protocol 1601 - Cardax/Gallagher	<code>rdrcfg.lf.lf_supported = LF_CARDAX;</code>
Protocol 1602 - Jablotron	<code>rdrcfg.lf.lf_supported = LF_JABLOTRON;</code>
Protocol 35 - HID Prox Wallb.	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = WAUTO;</code>

Protocol ID and name	Configuration used
Protocol 5 - HID Prox W26	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W26;</code>
Protocol 6 - HID Prox W27	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W27;</code>
Protocol 7 - HID Prox W32	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W32;</code>
Protocol 8 - HID Prox W34	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W34;</code>
Protocol 9 - HID Prox W37	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W37;</code>
Protocol 27 - HID Prox W40	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W40;</code>
Protocol 46 - HID Prox W40 w/o par.	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W40_PAR;</code>
Protocol 10 - HID Prox W44	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W44;</code>
Protocol 56 - HID Prox Wallb. continuous	<code>rdrcfg.common.continuous = 1;</code> <code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = WAUTO;</code>
Protocol 1005 - HID Prox UDF W26	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W26;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1006 - HID Prox UDF W27	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W27;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1007 - HID Prox UDF W32	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W32;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1008 - HID Prox UDF W34	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W34;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1009 - HID Prox UDF W37	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W37;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1027 - HID Prox UDF W40	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W40;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>

Protocol ID and name	Configuration used
Protocol 1046 - HID Prox UDF W40 w/o par.	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W40_PAR;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1010 - HID Prox UDF W44	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W44;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1035 - HID Prox UDF Wallb.	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = WAUTO;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1056 - HID Prox UDF Wallb. cont.	<code>rdrcfg.common.continuous = 1;</code> <code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = WAUTO;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
<i>Protocol 117 - ASK/FSK testing</i>	<code>rdrcfg.common.testing_mode = 1;</code> <code>rdrcfg.common.exact_cardtype = 1;</code> <code>rdrcfg.sam.sam_mode = SAM_AUTO;</code>

8.4 B-018 - MULTIREADER LF + HF V2

Reader description	Universal 13.56 MHz and 125kHz reader supporting ASK and FSK modulation on 125kHz (includes HID Prox decoding) and various 13.56MHz standards
Reader in production	Yes
Reader technology	Contactless
Reader frequency	13.56MHz / 125kHz
Notes	
<p>Reading of Mifare Classic sectors supported.</p> <p>Reading of Mifare Classic sectors from 7 byte UIN or random UIN cards supported.</p> <p>Reading of Mifare DESFire data files supported.</p> <p>Reading of Mifare Plus sectors not supported at the moment (customization).</p> <p>Reading of Inside Contactless PicoPass blocks supported.</p> <p>ISO14443A/B-4 APDU commands supported.</p> <p>Processing of multiple HF cards at the same time supported.</p> <p>Only one ISO14443-B card supported in the field.</p> <p>On LF UIN mode it may take longer to read the card for the first time.</p> <p>Protocol 117 should not be used for user identification, it is a special protocol intended as an aid for card testing and some limitations may apply.</p>	

Supported cfg items	Supported values	Note
rdrcfg.common.cardno_conv	(CARDCONV_APPENDLRC, CARDCONV_HEX2DEC, CARDCONV_HEX2DEC10, CARDCONV_HEX2DECLEFT16, CARDCONV_INTERFLEX, CARDCONV_ISO14443ATRUNCREVENDIAN, CARDCONV_JAVACARDSERIAL, CARDCONV_REVBITSINBYTE, CARDCONV_REVENDIAN, CARDCONV_REVENDIAN2DEC, CARDCONV_REVENDIANHEX2DEC10, CARDCONV_REVENDIANWOHIDPROX)	Builtin card number conversions. Will be overwritten if rdrcfg.cardno.cardnoconv specified
rdrcfg.cardno.cardnoconv		Custom specified card number conversion. CM2CNO("text") or CNOASM(FILE("filename"))).
rdrcfg.common.continuous	(0, 1)	Special setting that enables card requested for login and card remove notification on some products.
rdrcfg.common.testing_mode	(0, 1)	Special testing setting to scan all supported technologies. Only for debug or testing purposes, DO NOT USE IN PRODUCTION ENVIRONMENT, some limitations may apply.
rdrcfg.lf.lf_disabled	(0, 1)	Completely disable processing of LF technologies
rdrcfg.lf.lf_supported	(LF_AUTO, LF_EM4000, LF_PROXLITE, LF_PARADOX, LF_NEDAP, LF_AWID, LF_PYRAMID, LF_DEISTER, LF_DATASEC, LF_GPROX, LF_IOPROX, LF_EM4050RO, LF_PAC, LF_URMET, LF_CARDAX, LF_JABLOTRON, LF_HIDPROX, LF_CUSTOM)	

Supported cfg items	Supported values	Note
rdrcfg.lf.lf_supported_multiple	[LF_AUTO, LF_EM4000, LF_PROXLITE, LF_PARADOX, LF_NEDAP, LF_AWID, LF_PYRAMID, LF_DEISTER, LF_DATASEC, LF_GPROX, LF_IOPROX, LF_EM4050R0, LF_PAC, LF_URMET, LF_CARDAX, LF_JABLOTRON, LF_HIDPROX]	Combining other technologies with LF_CUSTOM is limited.
rdrcfg.lf.hidprox.wiegand_format	(WAUTO, W26, W27, W32, W34, W37, W40, W40_PAR, W44, WABA2, WMAGSTRIPE)	
rdrcfg.lf.hidprox.hid_udf_enable	(0, 1)	Enable processing of HID Prox User Defined Formats - cards with non-standard customer code
rdrcfg.lf.hidprox.no_wiegand_parity_check	(0, 1)	Do not check parity on wiegand formats that contain parity
rdrcfg.lf.hidprox.no_error_report	(0, 1)	Do not report parity or other errors
rdrcfg.lf.lf_custom.modulation	(LF_ASK, LF_FSK2)	Modulation type
rdrcfg.lf.lf_custom.bitrate	(LF_F16, LF_F32, LF_F50, LF_F64)	Card bitrate
rdrcfg.lf.lf_custom.start_decoding	(LF_DECODE_NRZ, LF_DECODE_MANCHESTER, LF_DECODE_DIFFBIPHASE)	Decoding of start condition
rdrcfg.lf.lf_custom.data_decoding	(LF_DECODE_NRZ, LF_DECODE_MANCHESTER, LF_DECODE_DIFFBIPHASE)	Decoding of data
rdrcfg.lf.lf_custom.start_cond	u32	32 bit start condition
rdrcfg.lf.lf_custom.start_mask	u32	32 bit Start condition mask. Must be continuous sequence of bits on ASKFSK frontend
rdrcfg.lf.lf_custom.count_bits	int	Data bits count without start condition. If you do not specify complete bith length, then adjust ignore_bits for complete data reading

Supported cfg items	Supported values	Note
rdrcfg.lf.lf_custom.start_decode_invert	(0, 1)	Invert start condition data. Not applicable if LF_DECODE_DIFFBI PHASE is used
rdrcfg.lf.lf_custom.data_decode_invert	(0, 1)	Invert data. Not applicable if LF_DECODE_DIFFBI PHASE is used
rdrcfg.lf.lf_custom.input_invert	(0, 1)	Invert data incoming to decodes
rdrcfg.lf.lf_custom.read_count	int	How many times the data should be read and checked and compared as identical
rdrcfg.lf.lf_custom.padding_bits	int	Padding 0 bits before actual data output
rdrcfg.lf.lf_custom.ignore_bits	int	Bits at the end of the data stream that should be ignored (i.e. void bits), used for proper timing calculation
rdrcfg.lf.lf_custom.card_bootup_time	int	Value to add to card timeout - card bootup overhead in us
rdrcfg.lf.lf_custom.decode	(LF_DECODE_RAW, LF_DECODE_EM4K)	Use particular decode function
rdrcfg.lf.lf_custom.remove_timeout	uint	Timeout for card remove
rdrcfg.lf.lf_custom.card_type	uint	Card type to report
rdrcfg.hf.hf_disabled	(0, 1)	Completely disable processing of HF technologies
rdrcfg.common.no_uin_check	(0, 1)	Do not check if the card UIN is still the same. Required for cards with random UIN
rdrcfg.hf.hf_supported	(HF_DEFAULT, HF_ISO14443A, HF_ISO14443B, HF_ISO15693, HF_INSIDE, HF_FELICA, HF_SIELOX)	

Supported cfg items	Supported values	Note
rdrcfg.hf.hf_supported_multiple	[HF_ISO14443A, HF_ISO14443B, HF_ISO15693, HF_INSIDE, HF_FELICA, HF_SIELOX]	Multiple values supported
rdrcfg.hf.hf_uin_revendian	(0, 1)	Reverse endianness of UIN
rdrcfg.hf.report_read_error	(REP_NONE, REP_AUTO, REP_ERROR, REP_REFUSED)	Change default behaviour of error reporting if the card is not read correctly. By default the card is ignored but the reader may be configured to issue card read error or card refused messages.
rdrcfg.hf.hf_iso_select	(0, 1)	ISO14443A/B-4 ISO select. Required for DESFire/ISO14443A/B-4 commands.
rdrcfg.hf.hf_uin_mode	(HF_UIN_AUTO, HF_UIN_NORMAL, HF_UIN_LEGIC)	Configuration of how UIN should be reported. Some readers may have a different endianness on some technologies.
rdrcfg.hf.read_error_return_uin	(0, 1)	Any error in reading data areas will result in returning the card UIN. Use with caution as this lowers overall security.
rdrcfg.other.piv_mode	(PIV_MODE_AUTO, PIV_MODE_FASCN, PIV_MODE_GUID)	Specify mode of reading of PIV-II cards
rdrcfg.hf.inside_read.key_type	(KEY_KD, KEY_KC)	Type of used key
rdrcfg.hf.inside_read.key	data[8]	key for protected area

Supported cfg items	Supported values	Note
rdrcfg.hf.inside_read.page	uint	Book and page in the book, lower nibble (4 bits) are page, bit 4 (LSB of higher nibble) is book. Works only on paged media. Example: book 1, page 2 = 0x12
rdrcfg.hf.inside_read.address	<0,255>	Read from n-th block (1 block = 8 bytes), 2K page: <0,31>; 16K page: <0,255>
rdrcfg.hf.inside_read.read_count	<1,128>	Read this many blocks (each has 8 bytes)
rdrcfg.hf.mifare_read.mode	(MIFARE_CLASSIC)	
rdrcfg.hf.mifare_read.block	<0,240>	0-63 (Mifare 1k), 0-240 (Mifare 4k)
rdrcfg.hf.mifare_read.mifare_key.key_type	(KEY_MIFARE_A, KEY_MIFARE_B)	
rdrcfg.hf.mifare_read.mifare_key.key_data	data[6]	key data
rdrcfg.hf.mifare_read.card_type	uint	Card type to report instead of the default ISO standard
rdrcfg.hf.desfire_read.desfire_aid	data[3]	3 byte DESFire Application ID in endiannes reported from card
rdrcfg.hf.desfire_read.desfire_fid	<0,15>	file id
rdrcfg.hf.desfire_read.desfire_foffset	<0,65535>	offset in file
rdrcfg.hf.desfire_read.desfire_flen	<0,230>	0 returns complete file up to 230 bytes
rdrcfg.hf.desfire_read.desfire_fopts	{DESFIRE_FILE_AUTO, DESFIRE_FILE_PLAIN, DESFIRE_FILE_MACED, DESFIRE_FILE_ENCRYPTED, DESFIRE_FILE_STD, DESFIRE_FILE_VALUE, DESFIRE_FILE_RECORD}	file encryption options for read
rdrcfg.hf.desfire_read.desfire_key_no	<0,15>	
rdrcfg.hf.desfire_read.desfire_key.key_type	(KEY_DES, KEY_2K3DES, KEY_3K3DES, KEY_AES128, KEY_NONE)	
rdrcfg.hf.desfire_read.desfire_key.key_data	data[8,24]	key data

Supported cfg items	Supported values	Note
rdrcfg.hf.desfire_read.card_type	uint	Card type to report instead of the default ISO standard
Default configuration equivalent		
rdrcfg.lf.lf_supported = LF_AUTO; rdrcfg.hf.hf_supported = HF_DEFAULT;		
Protocol ID and name	Configuration used	
Protocol 1220 - LF UIN + HF UIN	rdrcfg.lf.lf_supported = LF_AUTO	
Protocol 105 - HF UIN only	rdrcfg.lf.lf_disabled = 1;	
Protocol 1221 - LF UIN only	rdrcfg.hf.hf_disabled = 1; rdrcfg.lf.lf_supported = LF_AUTO	
Protocol 90 - EM4000 compat.+ HF UIN	rdrcfg.lf.lf_supported = LF_EM4000;	
Protocol 94 - EM4000 compat. + HF UIN cont.	rdrcfg.common.continuous = 1; rdrcfg.lf.lf_supported = LF_EM4000;	
Protocol 118 - EM4000 compat.+ ISO14443A UIN	rdrcfg.lf.lf_supported = LF_EM4000; rdrcfg.hf.hf_supported = HF_ISO14443A;	
Protocol 125 - EM4000 compat.+ HID Prox + HF UIN	rdrcfg.lf.lf_supported_multiple.extend([LF_EM4000, LF_HIDPROX]);	
Protocol 91 - Prox-lite/Casi-Rusco + HF UIN	rdrcfg.lf.lf_supported = LF_PROXLITE;	
Protocol 92 - Paradox/PosiProx + HF UIN	rdrcfg.lf.lf_supported = LF_PARADOX;	
Protocol 89 - Nedap + HF UIN	rdrcfg.lf.lf_supported = LF_NEDAP;	
Protocol 87 - Awid + HF UIN	rdrcfg.lf.lf_supported = LF_AWID;	
Protocol 86 - Pyramid + HF UIN	rdrcfg.lf.lf_supported = LF_PYRAMID;	
Protocol 85 - Deister + HF UIN	rdrcfg.lf.lf_supported = LF_DEISTER;	

Protocol ID and name	Configuration used
Protocol 84 - Datasec + HF UIN	<code>rdrcfg.lf.lf_supported = LF_DATASEC;</code>
Protocol 83 - G-Prox + HF UIN	<code>rdrcfg.lf.lf_supported = LF_GPROX;</code>
Protocol 82 - ioProx label + HF UIN	<code>rdrcfg.lf.lf_supported = LF_IOPROX;</code>
Protocol 121 - EM4050 f/64 std. mode + HF UIN	<code>rdrcfg.lf.lf_supported = LF_EM4050R0;</code>
Protocol 96 - PAC/KeyPAC + HF UIN	<code>rdrcfg.lf.lf_supported_multiple.extend([LF_PAC, LF_KEYPAC]);</code>
Protocol 120 - URMET/FDI + HF UIN	<code>rdrcfg.lf.lf_supported = LF_URMET;</code>
Protocol 1601 - Cardax/Gallagher + HF UIN	<code>rdrcfg.lf.lf_supported = LF_CARDAX;</code>
Protocol 1602 - Jablotron + HF UIN	<code>rdrcfg.lf.lf_supported = LF_JABLOTRON;</code>
Protocol 35 - HID Prox Wallb. + HF UIN	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = WAUTO;</code>
Protocol 124 - HID Prox Wallb. + HF UIN rev. endian	<code>rdrcfg.hf.hf_uin_revendian = 1;</code> <code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = WAUTO;</code>
Protocol 5 - HID Prox W26 + HF UIN	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W26;</code>
Protocol 6 - HID Prox W27 + HF UIN	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W27;</code>
Protocol 7 - HID Prox W32 + HF UIN	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W32;</code>
Protocol 8 - HID Prox W34 + HF UIN	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W34;</code>
Protocol 9 - HID Prox W37 + HF UIN	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W37;</code>
Protocol 27 - HID Prox W40 + HF UIN	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W40;</code>
Protocol 46 - HID Prox W40 w/o par. + HF UIN	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W40_PAR;</code>

Protocol ID and name	Configuration used
Protocol 10 - HID Prox W44 + HF UIN	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W44;</code>
Protocol 1005 - HID Prox UDF W26 + HF UIN	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W26;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1006 - HID Prox UDF W27 + HF UIN	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W27;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1007 - HID Prox UDF W32 + HF UIN	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W32;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1008 - HID Prox UDF W34 + HF UIN	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W34;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1009 - HID Prox UDF W37 + HF UIN	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W37;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1027 - HID Prox UDF W40 + HF UIN	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W40;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1046 - HID Prox UDF W40 w/o par. + HF UIN	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W40_PAR;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1010 - HID Prox UDF W44 + HF UIN	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W44;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1035 - HID Prox UDF Wallb. + HF UIN	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = WAUTO;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
<i>Protocol 117 - ASK/ FSK testing + HF UIN</i>	<code>rdrcfg.common.testing_mode = 1;</code> <code>rdrcfg.common.exact_cardtype = 1;</code> <code>rdrcfg.sam.sam_mode = SAM_AUTO;</code>

8.5 B-019 - READER 3 MF

Reader description	Universal HF and LF reader supporting ASK and FSK modulation on LF and various HF standards
Reader in production	Yes
Reader technology	Contactless
Reader frequency	13.56MHz / 125kHz

Notes

YSoft Configuration Cards supported within 2 minutes after startup and if processing of configuration cards is enabled.

Procedure for programming using YSoft Configuration cards: place the card for approximately 10s on the reader, wait for successful programming indication.

Reading of Mifare Classic sectors supported.

Reading of Mifare Classic sectors from 7 byte UIN or random UIN cards supported.

Reading of Mifare DESFire data files supported.

Reading of Mifare Plus sectors not supported at the moment (customization).

Reading of Inside Contactless PicoPass blocks supported.

ISO14443A/B-4 APDU commands supported.

Processing of multiple HF cards at the same time supported.

Only one ISO14443-B card supported in the field.

On LF UIN mode it may take longer to read the card for the first time.

Protocol 117 should not be used for user identification, it is a special protocol intended as an aid for card testing and some limitations may apply.

Supported cfg items	Supported values	Note
rdrcfg.common.cardno_conv	(CARDCONV_APPENDLRC, CARDCONV_HEX2DEC, CARDCONV_HEX2DEC10, CARDCONV_HEX2DECLEFT16, CARDCONV_INTERFLEX, CARDCONV_ISO14443ATRUNCREVENDIAN, CARDCONV_JAVACARDSERIAL, CARDCONV_REVBITSINBYTE, CARDCONV_REVENDIAN, CARDCONV_REVENDIAN2DEC, CARDCONV_REVENDIANHEX2DEC10, CARDCONV_REVENDIANWOHIDPROX)	Builtin card number conversions. Will be overwritten if rdrcfg.cardno.cardnoconv specified
rdrcfg.cardno.cardnoconv		Custom specified card number conversion. CM2CNO("text") or CNOASM(FILE("filename"))).
rdrcfg.common.continuous	(0, 1)	Special setting that enables card requested for login and card remove notification on some products.

Supported cfg items	Supported values	Note
rdrcfg.common.testing_mode	(0, 1)	Special testing setting to scan all supported technologies. Only for debug or testing purposes, DO NOT USE IN PRODUCTION ENVIRONMENT, some limitations may apply.
rdrcfg.lf.lf_disabled	(0, 1)	Completely disable processing of LF technologies
rdrcfg.lf.lf_supported	(LF_AUTO, LF_EM4000, LF_PROXLITE, LF_PARADOX, LF_NEDAP, LF_AWID, LF_PYRAMID, LF_DEISTER, LF_DATASEC, LF_GPROX, LF_IOPROX, LF_EM4050RO, LF_PAC, LF_URMET, LF_CARDAX, LF_JABLOTRON, LF_CUSTOM)	
rdrcfg.lf.lf_supported_multiple	[LF_AUTO, LF_EM4000, LF_PROXLITE, LF_PARADOX, LF_NEDAP, LF_AWID, LF_PYRAMID, LF_DEISTER, LF_DATASEC, LF_GPROX, LF_IOPROX, LF_EM4050RO, LF_PAC, LF_URMET, LF_CARDAX, LF_JABLOTRON]	Combining other technologies with LF_CUSTOM is limited.
rdrcfg.lf.lf_custom.modulation	(LF_ASK, LF_FSK2)	Modulation type
rdrcfg.lf.lf_custom.bitrate	(LF_F16, LF_F32, LF_F50, LF_F64)	Card bitrate
rdrcfg.lf.lf_custom.start_decoding	(LF_DECODE_NRZ, LF_DECODE_MANCHESTER, LF_DECODE_DIFFBIPHASE)	Decoding of start condition
rdrcfg.lf.lf_custom.data_decoding	(LF_DECODE_NRZ, LF_DECODE_MANCHESTER, LF_DECODE_DIFFBIPHASE)	Decoding of data
rdrcfg.lf.lf_custom.start_cond	u32	32 bit start condition
rdrcfg.lf.lf_custom.start_mask	u32	32 bit Start condition mask. Must be continuous sequence of bits on ASKFSK frontend

Supported cfg items	Supported values	Note
rdrcfg.lf.lf_custom.count_bits	int	Data bits count without start condition. If you do not specify complete bith length, then adjust ignore_bits for complete data reading
rdrcfg.lf.lf_custom.start_decode_invert	(0, 1)	Invert start condition data. Not applicable if LF_DECODE_DIFFBI PHASE is used
rdrcfg.lf.lf_custom.data_decode_invert	(0, 1)	Invert data. Not applicable if LF_DECODE_DIFFBI PHASE is used
rdrcfg.lf.lf_custom.input_invert	(0, 1)	Invert data incoming to decodes
rdrcfg.lf.lf_custom.read_count	int	How many times the data should be read and checked and compared as identical
rdrcfg.lf.lf_custom.padding_bits	int	Padding 0 bits before actual data output
rdrcfg.lf.lf_custom.ignore_bits	int	Bits at the end of the data stream that should be ignored (i.e. void bits), used for proper timing calculation
rdrcfg.lf.lf_custom.card_bootup_time	int	Value to add to card timeout - card bootup overhead in us
rdrcfg.lf.lf_custom.decode	(LF_DECODE_RAW, LF_DECODE_EM4K)	Use particular decode function
rdrcfg.lf.lf_custom.remove_timeout	uint	Timeout for card remove
rdrcfg.lf.lf_custom.card_type	uint	Card type to report

Supported cfg items	Supported values	Note
rdrcfg.hf.hf_disabled	(0, 1)	Completely disable processing of HF technologies
rdrcfg.common.no_uin_check	(0, 1)	Do not check if the card UIN is still the same. Required for cards with random UIN
rdrcfg.hf.hf_supported	(HF_DEFAULT, HF_ISO14443A, HF_ISO14443B, HF_ISO15693, HF_INSIDE, HF_FELICA, HF_SIELOX)	
rdrcfg.hf.hf_supported_multiple	[HF_ISO14443A, HF_ISO14443B, HF_ISO15693, HF_INSIDE, HF_FELICA, HF_SIELOX]	Multiple values supported
rdrcfg.hf.hf_uin_revendian	(0, 1)	Reverse endianness of UIN
rdrcfg.hf.report_read_error	(REP_NONE, REP_AUTO, REP_ERROR, REP_REFUSED)	Change default behaviour of error reporting if the card is not read correctly. By default the card is ignored but the reader may be configured to issue card read error or card refused messages.
rdrcfg.hf.hf_iso_select	(0, 1)	ISO14443A/B-4 ISO select. Required for DESFire/ISO14443A/B-4 commands.
rdrcfg.hf.hf_uin_mode	(HF_UIN_AUTO, HF_UIN_NORMAL, HF_UIN_LEGIC)	Configuration of how UIN should be reported. Some readers may have a different endianness on some technologies.
rdrcfg.hf.read_error_return_uin	(0, 1)	Any error in reading data areas will result in returning the card UIN. Use with caution as this lowers overall security.
rdrcfg.other.piv_mode	(PIV_MODE_AUTO, PIV_MODE_FASCN, PIV_MODE_GUID)	Specify mode of reading of PIV-II cards

Supported cfg items	Supported values	Note
rdrcfg.hf.config_card_disabled	(0, 1)	Disable processing of YSoft Configuration Cards
rdrcfg.hf.config_card_process_time	int	Time in msec for which the config card will be checked for presence before processing, default 8000
rdrcfg.hf.config_card_allow_time	int	Time in sec after startup in which configuration card is allowed, -1 means always enabled, default 120
rdrcfg.hf.inside_read.key_type	(KEY_KD, KEY_KC)	Type of used key
rdrcfg.hf.inside_read.key	data[8]	key for protected area
rdrcfg.hf.inside_read.page	uint	Book and page in the book, lower nibble (4 bits) are page, bit 4 (LSB of higher nibble) is book. Works only on paged media. Example: book 1, page 2 = 0x12
rdrcfg.hf.inside_read.address	<0,255>	Read from n-th block (1 block = 8 bytes), 2K page: <0,31>; 16K page: <0,255>
rdrcfg.hf.inside_read.read_count	<1,128>	Read this many blocks (each has 8 bytes)
rdrcfg.hf.mifare_read.mode	(MIFARE_CLASSIC)	
rdrcfg.hf.mifare_read.block	<0,240>	0-63 (Mifare 1k), 0-240 (Mifare 4k)
rdrcfg.hf.mifare_read.mifare_key.key_type	(KEY_MIFARE_A, KEY_MIFARE_B)	
rdrcfg.hf.mifare_read.mifare_key.key_data	data[6]	key data
rdrcfg.hf.mifare_read.card_type	uint	Card type to report instead of the default ISO standard

Supported cfg items	Supported values	Note
rdrcfg.hf.desfire_read.desfire_aid	data[3]	3 byte DESFire Application ID in endiannes reported from card
rdrcfg.hf.desfire_read.desfire_fid	<0,15>	file id
rdrcfg.hf.desfire_read.desfire_foffset	<0,65535>	offset in file
rdrcfg.hf.desfire_read.desfire_flen	<0,230>	0 returns complete file up to 230 bytes
rdrcfg.hf.desfire_read.desfire_fopts	{DESFIRE_FILE_AUTO, DESFIRE_FILE_PLAIN, DESFIRE_FILE_MACED, DESFIRE_FILE_ENCRYPTED, DESFIRE_FILE_STD, DESFIRE_FILE_VALUE, DESFIRE_FILE_RECORD}	file encryption options for read
rdrcfg.hf.desfire_read.desfire_key_no	<0,15>	
rdrcfg.hf.desfire_read.desfire_key.key_type	(KEY_DES, KEY_2K3DES, KEY_3K3DES, KEY_AES128, KEY_NONE)	
rdrcfg.hf.desfire_read.desfire_key.key_data	data[8,24]	key data
rdrcfg.hf.desfire_read.card_type	uint	Card type to report instead of the default ISO standard
rdrcfg.hf.iso_read.read	(ISO_CHUID, ISO_CEPAS)	reading of predefined HF technologies
rdrcfg.hf.iso_read.card_type	uint	Card type to report instead of the default ISO standard

Default configuration equivalent

```
rdrcfg.lf.lf_supported = LF_AUTO;
rdrcfg.hf.hf_supported = HF_DEFAULT;
```

Protocol ID and name	Configuration used
Protocol 1220 - LF UIN + HF UIN	rdrcfg.lf.lf_supported = LF_AUTO
Protocol 1222 - LF UIN + HF UIN rev. endian	rdrcfg.lf.lf_supported = LF_AUTO rdrcfg.hf.hf_supported = HF_DEFAULT rdrcfg.hf.hf_uin_revendian = 1
Protocol 1221 - LF UIN only	rdrcfg.hf.hf_disabled = 1; rdrcfg.lf.lf_supported = LF_AUTO

Protocol ID and name	Configuration used
Protocol 105 - HF UIN only	<code>rdrcfg.lf.lf_disabled = 1;</code>
Protocol 106 - HF UIN only rev. endian	<code>rdrcfg.hf.hf_uin_revendian = 1</code> <code>rdrcfg.lf.lf_disabled = 1;</code>
Protocol 1500 - HF UIN only continuous	<code>rdrcfg.lf.lf_disabled = 1;</code> <code>rdrcfg.common.continuous = 1;</code>
Protocol 1518 - ISO14443A UIN only	<code>rdrcfg.lf.lf_disabled = 1;</code> <code>rdrcfg.hf.hf_supported = HF_ISO14443A;</code>
Protocol 90 - EM4000 compat.+ HF UIN	<code>rdrcfg.lf.lf_supported = LF_EM4000;</code>
Protocol 94 - EM4000 compat. + HF UIN cont.	<code>rdrcfg.common.continuous = 1;</code> <code>rdrcfg.lf.lf_supported = LF_EM4000;</code>
Protocol 118 - EM4000 compat.+ ISO14443A UIN	<code>rdrcfg.lf.lf_supported = LF_EM4000;</code> <code>rdrcfg.hf.hf_supported = HF_ISO14443A;</code>
Protocol 91 - Prox-lite/Casi-Rusco + HF UIN	<code>rdrcfg.lf.lf_supported = LF_PROXLITE;</code>
Protocol 92 - Paradox/PosiProx + HF UIN	<code>rdrcfg.lf.lf_supported = LF_PARADOX;</code>
Protocol 89 - Nedap + HF UIN	<code>rdrcfg.lf.lf_supported = LF_NEDAP;</code>
Protocol 87 - Awid + HF UIN	<code>rdrcfg.lf.lf_supported = LF_AWID;</code>
Protocol 86 - Pyramid + HF UIN	<code>rdrcfg.lf.lf_supported = LF_PYRAMID;</code>
Protocol 85 - Deister + HF UIN	<code>rdrcfg.lf.lf_supported = LF_DEISTER;</code>
Protocol 84 - Datasec + HF UIN	<code>rdrcfg.lf.lf_supported = LF_DATASEC;</code>
Protocol 83 - G-Prox + HF UIN	<code>rdrcfg.lf.lf_supported = LF_GPROX;</code>
Protocol 82 - ioProx label + HF UIN	<code>rdrcfg.lf.lf_supported = LF_IOPROX;</code>
Protocol 121 - EM4050 f/64 std. mode + HF UIN	<code>rdrcfg.lf.lf_supported = LF_EM4050R0;</code>

Protocol ID and name	Configuration used
Protocol 96 - PAC/KeyPAC + HF UIN	<code>rdrcfg.lf.lf_supported_multiple.extend([LF_PAC, LF_KEYPAC]);</code>
Protocol 120 - URMET/FDI + HF UIN	<code>rdrcfg.lf.lf_supported = LF_URMET;</code>
Protocol 1601 - Cardax/Gallagher + HF UIN	<code>rdrcfg.lf.lf_supported = LF_CARDAX;</code>
Protocol 1602 - Jablotron + HF UIN	<code>rdrcfg.lf.lf_supported = LF_JABLOTRON;</code>
Protocol 117 - ASK/FSK testing + HF UIN	<code>rdrcfg.common.testing_mode = 1;</code> <code>rdrcfg.common.exact_cardtype = 1;</code> <code>rdrcfg.sam.sam_mode = SAM_AUTO;</code>

8.6 B-027 - INDALA UIN

Reader description	Card reader supporting Motorola/Indala LF technology
Reader in production	Yes
Reader technology	Contactless
Reader frequency	125kHz
Notes	
<p>It incorporates the original Indala OEM module.</p> <p>The reader is configurable and highly customizable. Support of other or custom Indala formats possible as a customization.</p> <p>Programming with Indala programming (option) cards is supported.</p> <p>Indala ASP and ASP+ cards supported.</p> <p>Indala programming (option) cards supported on default formats.</p> <p>It may be necessary to select default W26/W26 all bits/W27 all bits/ABA protocols before and after programming by option cards.</p> <p>Procedure for programming: Configure default W26/W27/ABA2 protocols, place each of the option card for approximately 10s on the reader, keep the sequence as labelled, restart reader.</p> <p>Procedure for deprogramming: Configure special protocol 27131, configure back to default UIN protocol.</p> <p>Please note that setting of this protocol will erase any custom programmed format.</p>	

Supported cfg items	Supported values	Note
rdrcfg.common.cardno_conv	(CARDCONV_APPENDLRC, CARDCONV_HEX2DEC, CARDCONV_HEX2DEC10, CARDCONV_HEX2DECLEFT16, CARDCONV_INTERFLEX, CARDCONV_ISO14443ATRUNCREVENDIAN, CARDCONV_JAVACARDSERIAL, CARDCONV_REVBITSINBYTE, CARDCONV_REVENDIAN, CARDCONV_REVENDIAN2DEC, CARDCONV_REVENDIANHEX2DEC10, CARDCONV_REVENDIANWOHIDPROX)	Builtin card number conversions. Will be overwritten if rdrcfg.cardno.cardnoconv specified
rdrcfg.cardno.cardnoconv		Custom specified card number conversion. CM2CNO("text") or CNOASM(FILE("filename"))).
rdrcfg.lf.lf_supported	(LF_INDALA)	
rdrcfg.lf.indala.format	(INDALA_UIN1_SHORT, INDALA_UIN2_SHORT, INDALA_UIN1_LONG, INDALA_UIN2_LONG, INDALA_10022, INDALA_10251, INDALA_11037, INDALA_15024, INDALA_17531, INDALA_1771X, INDALA_10324, INDALA_11045, INDALA_DEFWIE, INDALA_DEFABA, INDALA_DEFSER, INDALA_CUSTOM1, INDALA_CUSTOM2, INDALA_CUSTOM3)	
rdrcfg.lf.indala.wiegand_format	(WAUTO, W26, W27, W32, W34, W37, W40, W40_PAR, W44, WABA2, WMAGSTRIPE)	
rdrcfg.lf.indala.no_wiegand_parity_check	(0, 1)	Do not check parity on wiegand formats that contain parity
rdrcfg.lf.indala.magstripe_format	(MAG_FMT_DEFAULT, MAG_ABA, MAG_IATA, MAG_NTT, MAG_CADMV1, MAG_CADMV3, MAG_SECURITAS, MAG_BINARY)	
rdrcfg.lf.indala.serial_mode	(SER_INDALA_SER, SER_INDALA_UIN, SER_INDALA_UINLONG, SER_INDALA_HEX)	
rdrcfg.lf.indala.custom_format_data	data[256]	256 bytes of reader configuration from NHX file (Indala ProxSmith)

Default configuration equivalent	
<pre>rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_UIN1_SHORT;</pre>	
Protocol ID and name	Configuration used
Protocol 1101 - UIN1 short	<pre>rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_UIN1_SHORT; rdrcfg.hf.hf_disabled = 1;</pre>
Protocol 1102 - UIN2 short	<pre>rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_UIN2_SHORT; rdrcfg.hf.hf_disabled = 1;</pre>
Protocol 1103 - UIN1 long	<pre>rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_UIN1_LONG; rdrcfg.hf.hf_disabled = 1;</pre>
Protocol 1104 - UIN2 long	<pre>rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_UIN2_LONG; rdrcfg.hf.hf_disabled = 1;</pre>
Protocol 1105 - W26 - format 10022	<pre>rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_10022; rdrcfg.lf.indala.wiegand_format = W26; rdrcfg.hf.hf_disabled = 1;</pre>
Protocol 1106 - W26 Wallb.- format 10022	<pre>rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_10022; rdrcfg.lf.indala.wiegand_format = WAUTO; rdrcfg.hf.hf_disabled = 1;</pre>
Protocol 1107 - W27 - format 10251	<pre>rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_10251; rdrcfg.lf.indala.wiegand_format = W27; rdrcfg.hf.hf_disabled = 1;</pre>
Protocol 1108 - W27 Wallb.- format 10251	<pre>rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_10251; rdrcfg.lf.indala.wiegand_format = WAUTO; rdrcfg.hf.hf_disabled = 1;</pre>
Protocol 1109 - ABA2 - format 11037	<pre>rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_11037; rdrcfg.lf.indala.magstripe_format = MAG_ABA; rdrcfg.hf.hf_disabled = 1;</pre>
Protocol 1110 - Lite - format 17531	<pre>rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_17531; rdrcfg.lf.indala.wiegand_format = W26; rdrcfg.hf.hf_disabled = 1;</pre>

Protocol ID and name	Configuration used
Protocol 1111 - Lite Wallb.- format 17531	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_17531; rdrcfg.lf.indala.wiegand_format = WAUTO; rdrcfg.hf.hf_disabled = 1;
Protocol 1112 - Kantech - format 15024	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_15024; rdrcfg.lf.indala.wiegand_format = W32; rdrcfg.hf.hf_disabled = 1;
Protocol 1113 - ESMI - format 1771x	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_1771X; rdrcfg.lf.indala.wiegand_format = WAUTO; rdrcfg.hf.hf_disabled = 1;
Protocol 1114 - ECR W26 Wallb. - format 10324	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_10324; rdrcfg.lf.indala.wiegand_format = WAUTO; rdrcfg.hf.hf_disabled = 1;
Protocol 1115 - ECR ABA2 - format 11045	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_11045; rdrcfg.lf.indala.magstripe_format = MAG_ABA; rdrcfg.hf.hf_disabled = 1;
Protocol 1130 - ASP+ default key Wiegand	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_DEFWIE; rdrcfg.lf.indala.wiegand_format = WAUTO; rdrcfg.hf.hf_disabled = 1;
Protocol 1131 - ASP+ default key ABA2	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_DEFABA; rdrcfg.lf.indala.magstripe_format = MAG_ABA; rdrcfg.hf.hf_disabled = 1;
Protocol 1132 - ASP+ default key Serial	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_DEFSER; rdrcfg.hf.hf_disabled = 1;

8.7 B-028 - READER 3 INDALA

Reader description	Card reader supporting Motorola/Indala LF technology
Reader in production	EOL 6/2021
Reader technology	Contactless
Reader frequency	125kHz

Notes

It incorporates the original Indala OEM module.

The reader is configurable and highly customizable. Support of other or custom Indala formats possible as a customization.

Programming with Indala programming (option) cards is supported.

Indala ASP and ASP+ cards supported.

Indala programming (option) cards supported on default formats.

It may be necessary to select default W26/W26 all bits/W27 all bits/ABA protocols before and after programming by option cards.

Procedure for programming: Configure default W26/W27/ABA2 protocols, place each of the option card for approximately 10s on the reader, keep the sequence as labelled, restart reader.

Procedure for deprogramming: Configure special protocol 27131, configure back to default UIN protocol.

Please note that setting of this protocol will erase any custom programmed format.

Supported cfg items	Supported values	Note
rdrcfg.common.cardno_conv	(CARDCONV_APPENDLRC, CARDCONV_HEX2DEC, CARDCONV_HEX2DEC10, CARDCONV_HEX2DECLEFT16, CARDCONV_INTERFLEX, CARDCONV_ISO14443ATRUNCREVENDIAN, CARDCONV_JAVACARDSERIAL, CARDCONV_REVBITSINBYTE, CARDCONV_REVENDIAN, CARDCONV_REVENDIAN2DEC, CARDCONV_REVENDIANHEX2DEC10, CARDCONV_REVENDIANWOHIDPROX)	Builtin card number conversions. Will be overwritten if rdrcfg.cardno.cardnoconv specified
rdrcfg.cardno.cardnoconv		Custom specified card number conversion. CM2CNO("text") or CNOASM(FILE("filename")).
rdrcfg.lf.lf_supported	(LF_INDALA)	
rdrcfg.lf.indala.format	(INDALA_UIN1_SHORT, INDALA_UIN2_SHORT, INDALA_UIN1_LONG, INDALA_UIN2_LONG, INDALA_10022, INDALA_10251, INDALA_11037, INDALA_15024, INDALA_17531, INDALA_1771X, INDALA_10324, INDALA_11045, INDALA_DEFWIE, INDALA_DEFABA, INDALA_DEFSER, INDALA_CUSTOM1, INDALA_CUSTOM2, INDALA_CUSTOM3)	
rdrcfg.lf.indala.wiegand_format	(WAUTO, W26, W27, W32, W34, W37, W40, W40_PAR, W44, WABA2, WMAGSTRIPE)	
rdrcfg.lf.indala.no_wiegand_parity_check	(0, 1)	Do not check parity on wiegand formats that contain parity

Supported cfg items	Supported values	Note
rdrcfg.lf.indala.magstripe_format	(MAG_FMT_DEFAULT, MAG_ABA, MAG_IATA, MAG_NTT, MAG_CADMV1, MAG_CADMV3, MAG_SECURITAS, MAG_BINARY)	
rdrcfg.lf.indala.serial_mode	(SER_INDALA_SER, SER_INDALA_UIN, SER_INDALA_UINLONG, SER_INDALA_HEX)	
rdrcfg.lf.indala.custom_format_data	data[256]	256 bytes of reader configuration from NHX file (Indala ProxSmith)
Default configuration equivalent		
<pre>rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_UIN1_SHORT;</pre>		
Protocol ID and name	Configuration used	
Protocol 1101 - UIN1 short	<pre>rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_UIN1_SHORT; rdrcfg.hf.hf_disabled = 1;</pre>	
Protocol 1102 - UIN2 short	<pre>rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_UIN2_SHORT; rdrcfg.hf.hf_disabled = 1;</pre>	
Protocol 1103 - UIN1 long	<pre>rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_UIN1_LONG; rdrcfg.hf.hf_disabled = 1;</pre>	
Protocol 1104 - UIN2 long	<pre>rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_UIN2_LONG; rdrcfg.hf.hf_disabled = 1;</pre>	
Protocol 1105 - W26 - format 10022	<pre>rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_10022; rdrcfg.lf.indala.wiegand_format = W26; rdrcfg.hf.hf_disabled = 1;</pre>	
Protocol 1106 - W26 Wallb.- format 10022	<pre>rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_10022; rdrcfg.lf.indala.wiegand_format = WAUTO; rdrcfg.hf.hf_disabled = 1;</pre>	
Protocol 1107 - W27 - format 10251	<pre>rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_10251; rdrcfg.lf.indala.wiegand_format = W27; rdrcfg.hf.hf_disabled = 1;</pre>	

Protocol ID and name	Configuration used
Protocol 1108 - W27 Wallb.- format 10251	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_10251; rdrcfg.lf.indala.wiegand_format = WAUTO; rdrcfg.hf.hf_disabled = 1;
Protocol 1109 - ABA2 - format 11037	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_11037; rdrcfg.lf.indala.magstripe_format = MAG_ABA; rdrcfg.hf.hf_disabled = 1;
Protocol 1110 - Lite - format 17531	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_17531; rdrcfg.lf.indala.wiegand_format = W26; rdrcfg.hf.hf_disabled = 1;
Protocol 1111 - Lite Wallb.- format 17531	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_17531; rdrcfg.lf.indala.wiegand_format = WAUTO; rdrcfg.hf.hf_disabled = 1;
Protocol 1112 - Kantech - format 15024	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_15024; rdrcfg.lf.indala.wiegand_format = W32; rdrcfg.hf.hf_disabled = 1;
Protocol 1113 - ESMI - format 1771x	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_1771X; rdrcfg.lf.indala.wiegand_format = WAUTO; rdrcfg.hf.hf_disabled = 1;
Protocol 1114 - ECR W26 Wallb. - format 10324	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_10324; rdrcfg.lf.indala.wiegand_format = WAUTO; rdrcfg.hf.hf_disabled = 1;
Protocol 1115 - ECR ABA2 - format 11045	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_11045; rdrcfg.lf.indala.magstripe_format = MAG_ABA; rdrcfg.hf.hf_disabled = 1;
Protocol 1130 - ASP+ default key Wiegand	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_DEFWIE; rdrcfg.lf.indala.wiegand_format = WAUTO; rdrcfg.hf.hf_disabled = 1;
Protocol 1131 - ASP+ default key ABA2	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_DEFABA; rdrcfg.lf.indala.magstripe_format = MAG_ABA; rdrcfg.hf.hf_disabled = 1;

Protocol ID and name	Configuration used
Protocol 1132 - ASP+ default key Serial	<pre>rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_DEFSER; rdrcfg.hf.hf_disabled = 1;</pre>

8.8 B-037 - LEGIC ADVANT + HID PROX

Reader description	Legic Advant v3 card reader with HID Prox card reader
Reader in production	Yes
Reader technology	Contactless
Reader frequency	13.56MHz / 125kHz

Notes

Reading of Legic Data segments supported.
 Reading of Legic KGH segments supported.
 Limited support of KGH segments on Advant cards.
 Reading of Legic Access segments supported.
 Reading of Mifare DESFire data files supported.
 Reading of Mifare Classic sectors NOT supported.
 Reading of Mifare Plus sectors not supported at the moment (customization).
 ISO14443A/B-4 APDU commands supported.
 Processing of multiple HF cards at the same time supported.
 Legic Access segments on Prime cards not tested.
 Only one ISO14443-B card supported in the field.
 Usage of Legic Prime cards together with ISO14443-A/B cards with random UIN is limited.
 Legic SAM-63 (Launch) and SAM-64 (Delaunch) cards supported.
 Procedure for Launch record delete: Configure special protocol 27132, configure back to default UIN protocol.

Supported cfg items	Supported values	Note
rdrcfg.common.cardno_conv	(CARDCONV_APPENDLRC, CARDCONV_HEX2DEC, CARDCONV_HEX2DEC10, CARDCONV_HEX2DECLEFT16, CARDCONV_INTERFLEX, CARDCONV_ISO14443ATRUNCREVENDIAN, CARDCONV_JAVACARDSERIAL, CARDCONV_REVBITSINBYTE, CARDCONV_REVENDIAN, CARDCONV_REVENDIAN2DEC, CARDCONV_REVENDIANHEX2DEC10, CARDCONV_REVENDIANWOHIDPROX)	Builtin card number conversions. Will be overwritten if rdrcfg.cardno.cardnoconv specified
rdrcfg.cardno.cardnoconv		Custom specified card number conversion. CM2CNO("text") or CNOASM(FILE("filename"))).

Supported cfg items	Supported values	Note
rdrcfg.common.continuous	(0, 1)	Special setting that enables card requested for login and card remove notification on some products.
rdrcfg.lf.lf_disabled	(0, 1)	Completely disable processing of LF technologies
rdrcfg.lf.hidprox.wiegand_format	(WAUTO, W26, W27, W32, W34, W37, W40, W40_PAR, W44, WABA2, WMAGSTRIPE)	
rdrcfg.lf.hidprox.no_wiegand_parity_check	(0, 1)	Do not check parity on wiegand formats that contain parity
rdrcfg.lf.hidprox.no_error_report	(0, 1)	Do not report parity or other errors
rdrcfg.hf.hf_disabled	(0, 1)	Completely disable processing of HF technologies
rdrcfg.common.no_uin_check	(0, 1)	Do not check if the card UIN is still the same. Required for cards with random UIN
rdrcfg.hf.hf_supported	(HF_DEFAULT, HF_LEGIC_PRIME, HF_LEGIC_ADVANT, HF_LEGIC_ADVANT_ISO14443A, HF_LEGIC_ADVANT_ISO1693, HF_ISO14443A, HF_ISO14443B, HF_ISO15693, HF_INSIDE, HF_FELICA)	
rdrcfg.hf.hf_supported_multiple	[HF_DEFAULT, HF_LEGIC_PRIME, HF_LEGIC_ADVANT, HF_LEGIC_ADVANT_ISO14443A, HF_LEGIC_ADVANT_ISO1693, HF_ISO14443A, HF_ISO14443B, HF_ISO15693, HF_INSIDE, HF_FELICA]	Multiple values supported
rdrcfg.hf.hf_uin_revendian	(0, 1)	Reverse endianness of UIN

Supported cfg items	Supported values	Note
rdrcfg.hf.report_read_error	(REP_NONE, REP_AUTO, REP_ERROR, REP_REFUSED)	Change default behaviour of error reporting if the card is not read correctly. By default the card is ignored but the reader may be configured to issue card read error or card refused messages.
rdrcfg.hf.hf_iso_select	(0, 1)	ISO14443A/B-4 ISO select. Required for DESFire/ISO14443A/B-4 commands.
rdrcfg.hf.hf_uin_mode	(HF_UIN_AUTO, HF_UIN_NORMAL, HF_UIN_LEGIC)	Configuration of how UIN should be reported. Some readers may have a different endianness on some technologies.
rdrcfg.hf.read_error_return_uin	(0, 1)	Any error in reading data areas will result in returning the card UIN. Use with caution as this lowers overall security.
rdrcfg.other.piv_mode	(PIV_MODE_AUTO, PIV_MODE_FASCN, PIV_MODE_GUID)	Specify mode of reading of PIV-II cards
rdrcfg.hf.legic_read.mode	(LEGIC_DATA_SEG, LEGIC_KGH, LEGIC_ACCESS)	

Supported cfg items	Supported values	Note
rdrcfg.hf.legic_read.flags	{LEGIC_FLAG_NONE, LEGIC_KGH_SHORT, LEGIC_KGH_OLD, LEGIC_KGH_COMPATV2, LEGIC_ACCESS_STAMP, LEGIC_ACCESS_ID, LEGIC_ACCESS_USER}	OR combinations possible. LEGIC_KGH_OLD - compatibility with old Legic Prime/Legic Advant readers. LEGIC_KGH_COMPATV2 - compatibility with Legic Advant v2 card readers. LEGIC_KGH_SHORT - returns just stamp and card number, preferred option for KGH.
rdrcfg.hf.legic_read.from_seg	<1,128>	
rdrcfg.hf.legic_read.read_len	<1,199>	
rdrcfg.hf.legic_read.read_offset	<0,65535>	
rdrcfg.hf.legic_read.search_string	data[1,12]	1 to 12 bytes of segment stamp
rdrcfg.hf.legic_read.card_type	uint	Card type to report instead of the default Legic type
rdrcfg.hf.legic.legic_prime_disabled	(0, 1)	Disable processing of Legic Prime cards
rdrcfg.hf.legic.legic_advant_disabled	(0, 1)	Disable processing of Legic Advant cards
rdrcfg.hf.legic.enabled_cards_multiple	[CARD_LEGICMIM22, CARD_LEGICMIM256, CARD_LEGICMIM1024, CARD_LEGICATC128_MV210, CARD_LEGICATC256_MV210, CARD_LEGICATC512_MP110, CARD_LEGICATC1024_MV110, CARD_LEGICATC2048_MP110, CARD_LEGICATC4096_MP310, CARD_LEGICATC4096_MP311, CARD_LEGICATC4096_MP312, CARD_LEGICAFS4096_JP1X, CARD_LEGICATC1024_MV010, CARD_LEGICATC256_MV410, CARD_LEGICCTC4096_MP410, CARD_LEGICCTC4096_MM410]	List of Legic card types that are allowed

Supported cfg items	Supported values	Note
rdrcfg.hf.legic.disabled_cards_multiple	[CARD_LEGICMIM22, CARD_LEGICMIM256, CARD_LEGICMIM1024, CARD_LEGICATC128_MV210, CARD_LEGICATC256_MV210, CARD_LEGICATC512_MP110, CARD_LEGICATC1024_MV110, CARD_LEGICATC2048_MP110, CARD_LEGICATC4096_MP310, CARD_LEGICATC4096_MP311, CARD_LEGICATC4096_MP312, CARD_LEGICAFS4096_JP1X, CARD_LEGICATC1024_MV010, CARD_LEGICATC256_MV410, CARD_LEGICCTC4096_MP410, CARD_LEGICCTC4096_MM410]	List of Legic card types that should be rejected
rdrcfg.hf.legic.clear_launch_records	(0, 1)	Clear all launch records
rdrcfg.hf.legic.launch_media_disabled	(0, 1)	Disable launch/delaunch media processing
rdrcfg.hf.desfire_read.desfire_aid	data[3]	3 byte DESFire Application ID in endiannes reported from card
rdrcfg.hf.desfire_read.desfire_fid	<0,15>	file id
rdrcfg.hf.desfire_read.desfire_foffset	<0,65535>	offset in file
rdrcfg.hf.desfire_read.desfire_flen	<0,230>	0 returns complete file up to 230 bytes
rdrcfg.hf.desfire_read.desfire_fopts	{DESFIRE_FILE_AUTO, DESFIRE_FILE_PLAIN, DESFIRE_FILE_MACED, DESFIRE_FILE_ENCRYPTED, DESFIRE_FILE_STD, DESFIRE_FILE_VALUE, DESFIRE_FILE_RECORD}	file encryption options for read
rdrcfg.hf.desfire_read.desfire_key_no	<0,15>	
rdrcfg.hf.desfire_read.desfire_key.key_type	(KEY_DES, KEY_2K3DES, KEY_3K3DES, KEY_AES128, KEY_NONE)	
rdrcfg.hf.desfire_read.desfire_key.key_data	data[8,24]	key data
rdrcfg.hf.desfire_read.card_type	uint	Card type to report instead of the default ISO standard
rdrcfg.hf.iso_read.read	(ISO_CHUID, ISO_CEPAS)	reading of predefined HF technologies

Supported cfg items	Supported values	Note
rdrcfg.hf.iso_read.card_type	uint	Card type to report instead of the default ISO standard
Default configuration equivalent		
rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.hf.hf_supported = HF_DEFAULT;		
Protocol ID and name	Configuration used	
Protocol 49 - Legic UIN+HID Prox	rdrcfg.lf.lf_supported = LF_HIDPROX;	
Protocol 119 - Legic UIN only	rdrcfg.lf.lf_disabled = 1;	
Protocol 35 - HID Prox only	rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO; rdrcfg.hf.hf_disabled = 1;	
Protocol 113 - Legic UIN rev. endian+HID Prox	rdrcfg.hf.hf_uin_revendian = 1; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;	
Protocol 74 - Legic Advant UIN+HID Prox	rdrcfg.hf.hf_supported = HF_LEGIC_ADVANT; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;	
Protocol 107 - Legic Prime UIN+HID Prox	rdrcfg.hf.hf_supported = HF_LEGIC_PRIME; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;	
Protocol 80 - FIPS201 ID from PIV-II+HID Prox	rdrcfg.hf.hf_supported = HF_ISO14443A; rdrcfg.hf.hf_iso_select = 1; rdrcfg.hf.iso_read.read = ISO_CHUID; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;	
Protocol 108 - Legic short KGH seg. 1+HID Prox	rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_SHORT; rdrcfg.hf.legic_read.from_seg = 1; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;	
Protocol 109 - Legic short KGH seg. 2+HID Prox	rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_SHORT; rdrcfg.hf.legic_read.from_seg = 2; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;	

Protocol ID and name	Configuration used
Protocol 110 - Legic short KGH seg. 3+HID Prox	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_SHORT; rdrcfg.hf.legic_read.from_seg = 3; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;</pre>
Protocol 111 - Legic short KGH seg. 4+HID Prox	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_SHORT; rdrcfg.hf.legic_read.from_seg = 4; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;</pre>
Protocol 75 - Interflex card ID+HID Prox	<pre>rdrcfg.common.cardno_conv = CARDCONV_INTERFLEX; rdrcfg.common.allow_empty_uin = 1; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;</pre>
Protocol 64 - Legic KGH seg. 1+HID Prox	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_COMPATV2; rdrcfg.hf.legic_read.from_seg = 1; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;</pre>
Protocol 65 - Legic KGH seg. 2+HID Prox	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_COMPATV2; rdrcfg.hf.legic_read.from_seg = 2; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;</pre>
Protocol 66 - Legic KGH seg. 3+HID Prox	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_COMPATV2; rdrcfg.hf.legic_read.from_seg = 3; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;</pre>
Protocol 67 - Legic KGH seg. 4+HID Prox	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_COMPATV2; rdrcfg.hf.legic_read.from_seg = 4; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;</pre>
Protocol 29 - Legic KGH seg. 1 compat. +HID Prox	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_OLD; rdrcfg.hf.legic_read.from_seg = 1; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;</pre>
Protocol 30 - Legic KGH seg. 2 compat. +HID Prox	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_OLD; rdrcfg.hf.legic_read.from_seg = 2; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;</pre>

Protocol ID and name	Configuration used
Protocol 31 - Legic KGH seg. 3 compat. +HID Prox	<pre> rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_OLD; rdrcfg.hf.legic_read.from_seg = 3; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO; </pre>
Protocol 32 - Legic KGH seg. 4 compat. +HID Prox	<pre> rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_OLD; rdrcfg.hf.legic_read.from_seg = 4; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO; </pre>

8.9 B-038 - HID PROX V3

Reader description	Card reader supporting HID Prox and HID Prox User Defined Format (Customer Code)	
Reader in production	Yes	
Reader technology	Contactless	
Reader frequency	125kHz	
Notes		
None		
Supported cfg items	Supported values	Note
rdrcfg.common.cardno_conv	(CARDCONV_APPENDLRC, CARDCONV_HEX2DEC, CARDCONV_HEX2DEC10, CARDCONV_HEX2DECLEFT16, CARDCONV_INTERFLEX, CARDCONV_ISO14443ATRUNCREVENDIAN, CARDCONV_JAVACARDSERIAL, CARDCONV_REVBITSINBYTE, CARDCONV_REVENDIAN, CARDCONV_REVENDIAN2DEC, CARDCONV_REVENDIANHEX2DEC10, CARDCONV_REVENDIANWOHIDPROX)	Builtin card number conversions. Will be overwritten if rdrcfg.cardno.cardnoconv specified
rdrcfg.cardno.cardnoconv		Custom specified card number conversion. CM2CNO("text") or CNOASM(FILE("filename"))).

Supported cfg items	Supported values	Note
rdrcfg.common.continuous	(0, 1)	Special setting that enables card requested for login and card remove notification on some products.
rdrcfg.lf.lf_supported	(LF_AUTO, LF_EM4000, LF_PROXLITE, LF_PARADOX, LF_NEDAP, LF_AWID, LF_PYRAMID, LF_DEISTER, LF_DATASEC, LF_GPROX, LF_IOPROX, LF_EM4050RO, LF_PAC, LF_URMET, LF_CARDAX, LF_JABLOTRON, LF_HIDPROX, LF_CUSTOM)	
rdrcfg.lf.lf_supported_multiple	[LF_AUTO, LF_EM4000, LF_PROXLITE, LF_PARADOX, LF_NEDAP, LF_AWID, LF_PYRAMID, LF_DEISTER, LF_DATASEC, LF_GPROX, LF_IOPROX, LF_EM4050RO, LF_PAC, LF_URMET, LF_CARDAX, LF_JABLOTRON, LF_HIDPROX]	Combining other technologies with LF_CUSTOM is limited.
rdrcfg.lf.hidprox.wiegand_format	(WAUTO, W26, W27, W32, W34, W37, W40, W40_PAR, W44, WABA2, WMAGSTRIPE)	
rdrcfg.lf.hidprox.hid_udf_enable	(0, 1)	Enable processing of HID Prox User Defined Formats - cards with non-standard customer code
rdrcfg.lf.hidprox.no_wiegand_parity_check	(0, 1)	Do not check parity on wiegand formats that contain parity
rdrcfg.lf.hidprox.no_error_report	(0, 1)	Do not report parity or other errors

Default configuration equivalent

None

Protocol ID and name	Configuration used
Protocol 35 - Wiegand all bits auto	rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;
Protocol 5 - Wiegand 26	rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = W26;
Protocol 6 - Wiegand 27	rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = W27;

Protocol ID and name	Configuration used
Protocol 7 - Wiegand 32	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W32;</code>
Protocol 8 - Wiegand 34	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W34;</code>
Protocol 9 - Wiegand 37	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W37;</code>
Protocol 27 - Wiegand 40	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W40;</code>
Protocol 46 - Wiegand 40 w/o parity	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W40_PAR;</code>
Protocol 10 - Wiegand 44	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W44;</code>
Protocol 56 - Wiegand all bits continuous	<code>rdrcfg.common.continuous = 1;</code> <code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = WAUTO;</code>
Protocol 1005 - UDF Wiegand 26	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W26;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1006 - UDF Wiegand 27	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W27;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1007 - UDF Wiegand 32	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W32;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1008 - UDF Wiegand 34	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W34;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1009 - UDF Wiegand 37	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W37;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1027 - UDF Wiegand 40	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W40;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1046 - UDF Wiegand 40 w/o parity	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W40_PAR;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>

Protocol ID and name	Configuration used
Protocol 1010 - UDF Wiegand 44	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W44;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1035 - UDF Wiegand all bits auto	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = WAUTO;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1056 - UDF Wiegand all bits continuous	<code>rdrcfg.common.continuous = 1;</code> <code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = WAUTO;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>

8.10 B-039 - READER 3 LF+

Reader description	Universal LF reader supporting ASK and FSK modulation, includes license for HID Prox decoding
Reader in production	Yes
Reader technology	Contactless
Reader frequency	125kHz

Notes

On LF UIN mode it may take longer to read the card for the first time.
Protocol 117 should not be used for user identification, it is a special protocol intended as an aid for card testing and some limitations may apply.

Supported cfg items	Supported values	Note
<code>rdrcfg.common.cardno_conv</code>	(<code>CARDCONV_APPENDLRC</code> , <code>CARDCONV_HEX2DEC</code> , <code>CARDCONV_HEX2DEC10</code> , <code>CARDCONV_HEX2DECLEFT16</code> , <code>CARDCONV_INTERFLEX</code> , <code>CARDCONV_ISO14443ATRUNCREVENDIAN</code> , <code>CARDCONV_JAVACARDSERIAL</code> , <code>CARDCONV_REVBITSINBYTE</code> , <code>CARDCONV_REVENDIAN</code> , <code>CARDCONV_REVENDIAN2DEC</code> , <code>CARDCONV_REVENDIANHEX2DEC10</code> , <code>CARDCONV_REVENDIANWOHIDPROX</code>)	Builtin card number conversions. Will be overwritten if <code>rdrcfg.cardno.cardnoconv</code> specified
<code>rdrcfg.cardno.cardnoconv</code>		Custom specified card number conversion. <code>CM2CNO("text")</code> or <code>CNOASM(FILE("filename"))</code> .

Supported cfg items	Supported values	Note
rdrcfg.common.continuous	(0, 1)	Special setting that enables card requested for login and card remove notification on some products.
rdrcfg.common.testing_mode	(0, 1)	Special testing setting to scan all supported technologies. Only for debug or testing purposes, DO NOT USE IN PRODUCTION ENVIRONMENT, some limitations may apply.
rdrcfg.lf.lf_supported	(LF_AUTO, LF_EM4000, LF_PROXLITE, LF_PARADOX, LF_NEDAP, LF_AWID, LF_PYRAMID, LF_DEISTER, LF_DATASEC, LF_GPROX, LF_IOPROX, LF_EM4050RO, LF_PAC, LF_URMET, LF_CARDAX, LF_JABLOTRON, LF_HIDPROX, LF_CUSTOM)	
rdrcfg.lf.lf_supported_multiple	[LF_AUTO, LF_EM4000, LF_PROXLITE, LF_PARADOX, LF_NEDAP, LF_AWID, LF_PYRAMID, LF_DEISTER, LF_DATASEC, LF_GPROX, LF_IOPROX, LF_EM4050RO, LF_PAC, LF_URMET, LF_CARDAX, LF_JABLOTRON, LF_HIDPROX]	Combining other technologies with LF_CUSTOM is limited.
rdrcfg.lf.hidprox.wiegand_format	(WAUTO, W26, W27, W32, W34, W37, W40, W40_PAR, W44, WABA2, WMAGSTRIPE)	
rdrcfg.lf.hidprox.hid_udf_enable	(0, 1)	Enable processing of HID Prox User Defined Formats - cards with non-standard customer code
rdrcfg.lf.hidprox.no_wiegand_parity_check	(0, 1)	Do not check parity on wiegand formats that contain parity
rdrcfg.lf.hidprox.no_error_report	(0, 1)	Do not report parity or other errors
rdrcfg.lf.lf_custom.modulation	(LF_ASK, LF_FSK2)	Modulation type
rdrcfg.lf.lf_custom.bitrate	(LF_F16, LF_F32, LF_F50, LF_F64)	Card bitrate

Supported cfg items	Supported values	Note
rdrcfg.lf.lf_custom.start_decoding	(LF_DECODE_NRZ, LF_DECODE_MANCHESTER, LF_DECODE_DIFFBIPHASE)	Decoding of start condition
rdrcfg.lf.lf_custom.data_decoding	(LF_DECODE_NRZ, LF_DECODE_MANCHESTER, LF_DECODE_DIFFBIPHASE)	Decoding of data
rdrcfg.lf.lf_custom.start_cond	u32	32 bit start condition
rdrcfg.lf.lf_custom.start_mask	u32	32 bit Start condition mask. Must be continuous sequence of bits on ASKFSK frontend
rdrcfg.lf.lf_custom.count_bits	int	Data bits count without start condition. If you do not specify complete bith length, then adjust ignore_bits for complete data reading
rdrcfg.lf.lf_custom.start_decode_invert	(0, 1)	Invert start condition data. Not applicable if LF_DECODE_DIFFBI PHASE is used
rdrcfg.lf.lf_custom.data_decode_invert	(0, 1)	Invert data. Not applicable if LF_DECODE_DIFFBI PHASE is used
rdrcfg.lf.lf_custom.input_invert	(0, 1)	Invert data incoming to decodes
rdrcfg.lf.lf_custom.read_count	int	How many times the data should be read and checked and compared as identical
rdrcfg.lf.lf_custom.padding_bits	int	Padding 0 bits before actual data output

Supported cfg items	Supported values	Note
<code>rdrcfg.lf.lf_custom.ignore_bits</code>	int	Bits at the end of the data stream that should be ignored (i.e. void bits), used for proper timing calculation
<code>rdrcfg.lf.lf_custom.card_bootup_time</code>	int	Value to add to card timeout - card bootup overhead in us
<code>rdrcfg.lf.lf_custom.decode</code>	(<code>LF_DECODE_RAW</code> , <code>LF_DECODE_EM4K</code>)	Use particular decode function
<code>rdrcfg.lf.lf_custom.remove_timeout</code>	uint	Timeout for card remove
<code>rdrcfg.lf.lf_custom.card_type</code>	uint	Card type to report
Default configuration equivalent		
<code>rdrcfg.lf.lf_supported = LF_AUTO;</code>		

Protocol ID and name	Configuration used
Protocol 1220 - LF UIN	<code>rdrcfg.lf.lf_supported = LF_AUTO</code>
Protocol 125 - EM4000 compat. + HID Prox Wallb.	<code>rdrcfg.lf.lf_supported_multiple.extend([LF_EM4000, LF_HIDPROX]);</code>
Protocol 90 - EM4000 compatible	<code>rdrcfg.lf.lf_supported = LF_EM4000;</code>
Protocol 94 - EM4000 compatible cont.	<code>rdrcfg.common.continuous = 1;</code> <code>rdrcfg.lf.lf_supported = LF_EM4000;</code>
Protocol 88 - EM4000 compatible w/LRC (96008N1 compatible)	<code>rdrcfg.common.cardno_conv = CARDCONV_APPENDLRC;</code> <code>rdrcfg.lf.lf_supported = LF_EM4000;</code>
Protocol 91 - Proxlite/Casi-Rusco	<code>rdrcfg.lf.lf_supported = LF_PROXLITE;</code>
Protocol 92 - Paradox/PosiProx	<code>rdrcfg.lf.lf_supported = LF_PARADOX;</code>
Protocol 89 - Nedap	<code>rdrcfg.lf.lf_supported = LF_NEDAP;</code>
Protocol 87 - Awid	<code>rdrcfg.lf.lf_supported = LF_AWID;</code>
Protocol 86 - Pyramid	<code>rdrcfg.lf.lf_supported = LF_PYRAMID;</code>
Protocol 85 - Deister	<code>rdrcfg.lf.lf_supported = LF_DEISTER;</code>
Protocol 84 - Datasec	<code>rdrcfg.lf.lf_supported = LF_DATASEC;</code>
Protocol 83 - G-Prox	<code>rdrcfg.lf.lf_supported = LF_GPROX;</code>
Protocol 82 - ioProx label	<code>rdrcfg.lf.lf_supported = LF_IOPROX;</code>
Protocol 121 - EM4050 f/64 std. read mode	<code>rdrcfg.lf.lf_supported = LF_EM4050R0;</code>
Protocol 96 - PAC/KeyPAC	<code>rdrcfg.lf.lf_supported_multiple.extend([LF_PAC, LF_KEYPAC]);</code>
Protocol 120 - URMET/FDI	<code>rdrcfg.lf.lf_supported = LF_URMET;</code>
Protocol 1601 - Cardax/Gallagher	<code>rdrcfg.lf.lf_supported = LF_CARDAX;</code>
Protocol 1602 - Jablotron	<code>rdrcfg.lf.lf_supported = LF_JABLOTRON;</code>

Protocol ID and name	Configuration used
Protocol 35 - HID Prox Wallb.	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = WAUTO;</code>
Protocol 5 - HID Prox W26	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W26;</code>
Protocol 6 - HID Prox W27	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W27;</code>
Protocol 7 - HID Prox W32	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W32;</code>
Protocol 8 - HID Prox W34	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W34;</code>
Protocol 9 - HID Prox W37	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W37;</code>
Protocol 27 - HID Prox W40	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W40;</code>
Protocol 46 - HID Prox W40 w/o par.	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W40_PAR;</code>
Protocol 10 - HID Prox W44	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W44;</code>
Protocol 56 - HID Prox Wallb. continuous	<code>rdrcfg.common.continuous = 1;</code> <code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = WAUTO;</code>
Protocol 1005 - HID Prox UDF W26	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W26;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1006 - HID Prox UDF W27	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W27;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1007 - HID Prox UDF W32	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W32;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1008 - HID Prox UDF W34	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W34;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1009 - HID Prox UDF W37	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W37;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>

Protocol ID and name	Configuration used
Protocol 1027 - HID Prox UDF W40	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W40;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1046 - HID Prox UDF W40 w/o par.	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W40_PAR;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1010 - HID Prox UDF W44	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = W44;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1035 - HID Prox UDF Wallb.	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = WAUTO;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1056 - HID Prox UDF Wallb. cont.	<code>rdrcfg.common.continuous = 1;</code> <code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = WAUTO;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
<i>Protocol 117 - ASK/FSK testing</i>	<code>rdrcfg.common.testing_mode = 1;</code> <code>rdrcfg.common.exact_cardtype = 1;</code> <code>rdrcfg.sam.sam_mode = SAM_AUTO;</code>

8.11 B-041 - MULTIREADER LF + HF

Reader description	Universal 13.56 MHz and 125kHz reader supporting ASK and FSK modulation on 125kHz and various 13.56MHz standards
Reader in production	Yes
Reader technology	Contactless
Reader frequency	13.56MHz / 125kHz
Notes	
<p>Reading of Mifare Classic sectors supported.</p> <p>Reading of Mifare Classic sectors from 7 byte UIN or random UIN cards supported.</p> <p>Reading of Mifare DESFire data files supported.</p> <p>Reading of Mifare Plus sectors not supported at the moment (customization).</p> <p>Reading of Inside Contactless PicoPass blocks supported.</p> <p>ISO14443A/B-4 APDU commands supported.</p> <p>Processing of multiple HF cards at the same time supported.</p> <p>Only one ISO14443-B card supported in the field.</p> <p>On LF UIN mode it may take longer to read the card for the first time.</p> <p>Protocol 117 should not be used for user identification, it is a special protocol intended as an aid for card testing and some limitations may apply.</p>	

Supported cfg items	Supported values	Note
rdrcfg.common.cardno_conv	(CARDCONV_APPENDLRC, CARDCONV_HEX2DEC, CARDCONV_HEX2DEC10, CARDCONV_HEX2DECLEFT16, CARDCONV_INTERFLEX, CARDCONV_ISO14443ATRUNCREVENDIAN, CARDCONV_JAVACARDSERIAL, CARDCONV_REVBITSINBYTE, CARDCONV_REVENDIAN, CARDCONV_REVENDIAN2DEC, CARDCONV_REVENDIANHEX2DEC10, CARDCONV_REVENDIANWOHIDPROX)	Builtin card number conversions. Will be overwritten if rdrcfg.cardno.cardnoconv specified
rdrcfg.cardno.cardnoconv		Custom specified card number conversion. CM2CNO("text") or CNOASM(FILE("filename"))).
rdrcfg.common.continuous	(0, 1)	Special setting that enables card requested for login and card remove notification on some products.
rdrcfg.common.testing_mode	(0, 1)	Special testing setting to scan all supported technologies. Only for debug or testing purposes, DO NOT USE IN PRODUCTION ENVIRONMENT, some limitations may apply.
rdrcfg.lf.lf_disabled	(0, 1)	Completely disable processing of LF technologies
rdrcfg.lf.lf_supported	(LF_AUTO, LF_EM4000, LF_PROXLITE, LF_PARADOX, LF_NEDAP, LF_AWID, LF_PYRAMID, LF_DEISTER, LF_DATASEC, LF_GPROX, LF_IOPROX, LF_EM4050RO, LF_PAC, LF_URMET, LF_CARDAX, LF_JABLOTRON, LF_CUSTOM)	
rdrcfg.lf.lf_supported_multiple	[LF_AUTO, LF_EM4000, LF_PROXLITE, LF_PARADOX, LF_NEDAP, LF_AWID, LF_PYRAMID, LF_DEISTER, LF_DATASEC, LF_GPROX, LF_IOPROX, LF_EM4050RO, LF_PAC, LF_URMET, LF_CARDAX, LF_JABLOTRON]	Combining other technologies with LF_CUSTOM is limited.

Supported cfg items	Supported values	Note
rdrcfg.lf.lf_custom.modulation	(LF_ASK, LF_FSK2)	Modulation type
rdrcfg.lf.lf_custom.bitrate	(LF_F16, LF_F32, LF_F50, LF_F64)	Card bitrate
rdrcfg.lf.lf_custom.start_decoding	(LF_DECODE_NRZ, LF_DECODE_MANCHESTER, LF_DECODE_DIFFBIPHASE)	Decoding of start condition
rdrcfg.lf.lf_custom.data_decoding	(LF_DECODE_NRZ, LF_DECODE_MANCHESTER, LF_DECODE_DIFFBIPHASE)	Decoding of data
rdrcfg.lf.lf_custom.start_cond	u32	32 bit start condition
rdrcfg.lf.lf_custom.start_mask	u32	32 bit Start condition mask. Must be continuous sequence of bits on ASKFSK frontend
rdrcfg.lf.lf_custom.count_bits	int	Data bits count without start condition. If you do not specify complete bith length, then adjust ignore_bits for complete data reading
rdrcfg.lf.lf_custom.start_decode_invert	(0, 1)	Invert start condition data. Not applicable if LF_DECODE_DIFFBI PHASE is used
rdrcfg.lf.lf_custom.data_decode_invert	(0, 1)	Invert data. Not applicable if LF_DECODE_DIFFBI PHASE is used
rdrcfg.lf.lf_custom.input_invert	(0, 1)	Invert data incoming to decodes
rdrcfg.lf.lf_custom.read_count	int	How many times the data should be read and checked and compared as identical
rdrcfg.lf.lf_custom.padding_bits	int	Padding 0 bits before actual data output

Supported cfg items	Supported values	Note
rdrcfg.lf.lf_custom.ignore_bits	int	Bits at the end of the data stream that should be ignored (i.e. void bits), used for proper timing calculation
rdrcfg.lf.lf_custom.card_bootup_time	int	Value to add to card timeout - card bootup overhead in us
rdrcfg.lf.lf_custom.decode	(LF_DECODE_RAW, LF_DECODE_EM4K)	Use particular decode function
rdrcfg.lf.lf_custom.remove_timeout	uint	Timeout for card remove
rdrcfg.lf.lf_custom.card_type	uint	Card type to report
rdrcfg.hf.hf_disabled	(0, 1)	Completely disable processing of HF technologies
rdrcfg.common.no_uin_check	(0, 1)	Do not check if the card UIN is still the same. Required for cards with random UIN
rdrcfg.hf.hf_supported	(HF_DEFAULT, HF_ISO14443A, HF_ISO14443B, HF_ISO15693, HF_INSIDE, HF_FELICA, HF_SIELOX)	
rdrcfg.hf.hf_supported_multiple	[HF_ISO14443A, HF_ISO14443B, HF_ISO15693, HF_INSIDE, HF_FELICA, HF_SIELOX]	Multiple values supported
rdrcfg.hf.hf_uin_revendian	(0, 1)	Reverse endianness of UIN
rdrcfg.hf.report_read_error	(REP_NONE, REP_AUTO, REP_ERROR, REP_REFUSED)	Change default behaviour of error reporting if the card is not read correctly. By default the card is ignored but the reader may be configured to issue card read error or card refused messages.

Supported cfg items	Supported values	Note
rdrcfg.hf.hf_iso_select	(0, 1)	ISO14443A/B-4 ISO select. Required for DESFire/ISO14443A/B-4 commands.
rdrcfg.hf.hf_uin_mode	(HF_UIN_AUTO, HF_UIN_NORMAL, HF_UIN_LEGIC)	Configuration of how UIN should be reported. Some readers may have a different endianness on some technologies.
rdrcfg.hf.read_error_return_uin	(0, 1)	Any error in reading data areas will result in returning the card UIN. Use with caution as this lowers overall security.
rdrcfg.other.piv_mode	(PIV_MODE_AUTO, PIV_MODE_FASCN, PIV_MODE_GUID)	Specify mode of reading of PIV-II cards
rdrcfg.hf.inside_read.key_type	(KEY_KD, KEY_KC)	Type of used key
rdrcfg.hf.inside_read.key	data[8]	key for protected area
rdrcfg.hf.inside_read.page	uint	Book and page in the book, lower nibble (4 bits) are page, bit 4 (LSB of higher nibble) is book. Works only on paged media. Example: book 1, page 2 = 0x12
rdrcfg.hf.inside_read.address	<0,255>	Read from n-th block (1 block = 8 bytes), 2K page: <0,31>; 16K page: <0,255>
rdrcfg.hf.inside_read.read_count	<1,128>	Read this many blocks (each has 8 bytes)
rdrcfg.hf.mifare_read.mode	(MIFARE_CLASSIC)	
rdrcfg.hf.mifare_read.block	<0,240>	0-63 (Mifare 1k), 0-240 (Mifare 4k)
rdrcfg.hf.mifare_read.mifare_key.key_type	(KEY_MIFARE_A, KEY_MIFARE_B)	
rdrcfg.hf.mifare_read.mifare_key.key_data	data[6]	key data

Supported cfg items	Supported values	Note
rdrcfg.hf.mifare_read.card_type	uint	Card type to report instead of the default ISO standard
rdrcfg.hf.desfire_read.desfire_aid	data[3]	3 byte DESFire Application ID in endiannes reported from card
rdrcfg.hf.desfire_read.desfire_fid	<0,15>	file id
rdrcfg.hf.desfire_read.desfire_foffset	<0,65535>	offset in file
rdrcfg.hf.desfire_read.desfire_flen	<0,230>	0 returns complete file up to 230 bytes
rdrcfg.hf.desfire_read.desfire_fopts	{DESFIRE_FILE_AUTO, DESFIRE_FILE_PLAIN, DESFIRE_FILE_MACED, DESFIRE_FILE_ENCRYPTED, DESFIRE_FILE_STD, DESFIRE_FILE_VALUE, DESFIRE_FILE_RECORD}	file encryption options for read
rdrcfg.hf.desfire_read.desfire_key_no	<0,15>	
rdrcfg.hf.desfire_read.desfire_key.key_type	(KEY_DES, KEY_2K3DES, KEY_3K3DES, KEY_AES128, KEY_NONE)	
rdrcfg.hf.desfire_read.desfire_key.key_data	data[8,24]	key data
rdrcfg.hf.desfire_read.card_type	uint	Card type to report instead of the default ISO standard

Default configuration equivalent

```
rdrcfg.lf.lf_supported = LF_AUTO;
rdrcfg.hf.hf_supported = HF_DEFAULT;
```

Protocol ID and name	Configuration used
Protocol 1220 - LF UIN + HF UIN	rdrcfg.lf.lf_supported = LF_AUTO
Protocol 90 - EM4000 compat.+ HF UIN	rdrcfg.lf.lf_supported = LF_EM4000;
Protocol 94 - EM4000 compat. + HF UIN cont.	rdrcfg.common.continuous = 1; rdrcfg.lf.lf_supported = LF_EM4000;
Protocol 118 - EM4000 compat.+ ISO14443A UIN	rdrcfg.lf.lf_supported = LF_EM4000; rdrcfg.hf.hf_supported = HF_ISO14443A;

Protocol ID and name	Configuration used
Protocol 1221 - LF UIN only	<code>rdrcfg.hf.hf_disabled = 1;</code> <code>rdrcfg.lf.lf_supported = LF_AUTO</code>
Protocol 105 - HF UIN only	<code>rdrcfg.lf.lf_disabled = 1;</code>
Protocol 91 - ProxLite/Casi-Rusco + HF UIN	<code>rdrcfg.lf.lf_supported = LF_PROXLITE;</code>
Protocol 92 - Paradox/PosiProx + HF UIN	<code>rdrcfg.lf.lf_supported = LF_PARADOX;</code>
Protocol 89 - Nedap + HF UIN	<code>rdrcfg.lf.lf_supported = LF_NEDAP;</code>
Protocol 87 - Awid + HF UIN	<code>rdrcfg.lf.lf_supported = LF_AWID;</code>
Protocol 86 - Pyramid + HF UIN	<code>rdrcfg.lf.lf_supported = LF_PYRAMID;</code>
Protocol 85 - Deister + HF UIN	<code>rdrcfg.lf.lf_supported = LF_DEISTER;</code>
Protocol 84 - Datasec + HF UIN	<code>rdrcfg.lf.lf_supported = LF_DATASEC;</code>
Protocol 83 - G-Prox + HF UIN	<code>rdrcfg.lf.lf_supported = LF_GPROX;</code>
Protocol 82 - ioProx label + HF UIN	<code>rdrcfg.lf.lf_supported = LF_IOPROX;</code>
Protocol 121 - EM4050 f/64 std. mode + HF UIN	<code>rdrcfg.lf.lf_supported = LF_EM4050R0;</code>
Protocol 96 - PAC/KeyPAC + HF UIN	<code>rdrcfg.lf.lf_supported_multiple.extend([LF_PAC, LF_KEYPAC]);</code>
Protocol 120 - URMET/FDI + HF UIN	<code>rdrcfg.lf.lf_supported = LF_URMET;</code>
Protocol 1601 - Cardax/Gallagher + HF UIN	<code>rdrcfg.lf.lf_supported = LF_CARDAX;</code>
Protocol 1602 - Jablotron + HF UIN	<code>rdrcfg.lf.lf_supported = LF_JABLOTRON;</code>
Protocol 117 - ASK/FSK testing + HF UIN	<code>rdrcfg.common.testing_mode = 1;</code> <code>rdrcfg.common.exact_cardtype = 1;</code> <code>rdrcfg.sam.sam_mode = SAM_AUTO;</code>

8.12 B-045 - MULTI ISO

Reader description	Reader supporting various HF standards
Reader in production	Yes
Reader technology	Contactless
Reader frequency	13.56MHz
Notes	
<p>Reading of Mifare Classic sectors supported. Reading of Mifare DESFire data files supported. Mifare Classic cards with 7 byte UIN are not supported. Mifare Classic cards with random UIN not tested. SAM module supported. Implementation of SAM modules possible as a customization. Only all or just a single supported technology may be selected.</p>	

Supported cfg items	Supported values	Note
rdrcfg.common.cardno_conv	(CARDCONV_APPENDLRC, CARDCONV_HEX2DEC, CARDCONV_HEX2DEC10, CARDCONV_HEX2DECLEFT16, CARDCONV_INTERFLEX, CARDCONV_ISO14443ATRUNCREVENDIAN, CARDCONV_JAVACARDSERIAL, CARDCONV_REVBITSINBYTE, CARDCONV_REVENDIAN, CARDCONV_REVENDIAN2DEC, CARDCONV_REVENDIANHEX2DEC10, CARDCONV_REVENDIANWOHIDPROX)	Builtin card number conversions. Will be overwritten if rdrcfg.cardno.cardnoconv specified
rdrcfg.cardno.cardnoconv		Custom specified card number conversion. CM2CNO("text") or CNOASM(FILE("filename"))).
rdrcfg.common.continuous	1/0	
rdrcfg.common.no_uin_check	1/0	required for cards with random UIN
rdrcfg.hf.hf_supported	HF_DEFAULT, HF_ISO14443A, HF_ISO14443B, HF_ISO15693, HF_ICODE_UID, HF_ICODE_EPC, HF_SR176, HF_ASKRFID	
rdrcfg.hf.hf_iso_select	1/0	required for DESFire/ISO reading
rdrcfg.hf.report_read_error	REP_NONE, REP_AUTO, REP_ERROR, REP_REFUSED	
rdrcfg.hf.hf_uin_revendian	(0/1)	Reverse endianness of UIN
rdrcfg.hf.mifare_read.mode	(MIFARE_CLASSIC)	
rdrcfg.hf.mifare_read.block	<0,240>	0-63 (Mifare 1k), 0-240 (Mifare 4k)
rdrcfg.hf.mifare_read.mifare_key.key_type	(KEY_MIFARE_A, KEY_MIFARE_B)	
rdrcfg.hf.mifare_read.mifare_key.key_data	data[6]	key data
rdrcfg.hf.mifare_read.card_type	uint	Card type to report instead of the default ISO standard
rdrcfg.hf.desfire_read.desfire_aid	data[3]	3 byte DESFire Application ID in endiannes reported from card
rdrcfg.hf.desfire_read.desfire_fid	<0,15>	file id

Supported cfg items	Supported values	Note
rdrcfg.hf.desfire_read.desfire_offset	<0,65535>	offset in file
rdrcfg.hf.desfire_read.desfire_flen	<0,230>	0 returns complete file up to 230 bytes
rdrcfg.hf.desfire_read.desfire_fopts	{DESFIRE_FILE_AUTO, DESFIRE_FILE_PLAIN, DESFIRE_FILE_MACED, DESFIRE_FILE_ENCRYPTED, DESFIRE_FILE_STD, DESFIRE_FILE_VALUE, DESFIRE_FILE_RECORD}	file encryption options for read
rdrcfg.hf.desfire_read.desfire_key_no	<0,15>	
rdrcfg.hf.desfire_read.desfire_key.key_type	(KEY_DES, KEY_2K3DES, KEY_3K3DES, KEY_AES128, KEY_NONE)	
rdrcfg.hf.desfire_read.desfire_key.key_data	data[8,24]	key data
rdrcfg.hf.desfire_read.card_type	uint	Card type to report instead of the default ISO standard
Default configuration equivalent		
rdrcfg.hf.hf_supported = HF_DEFAULT;		
Protocol ID and name	Configuration used	
Protocol 44 - Multi ISO UIN	# Default reader settings	
Protocol 106 - UIN rev. endian	rdrcfg.hf.hf_uin_revendian = 1 rdrcfg.lf.lf_disabled = 1;	
Protocol 72 - ASK-RFID UIN	rdrcfg.lf.lf_disabled = 1; rdrcfg.hf.hf_supported = HF_ASKRFID;	

8.13 B-047 - READER 3 MULTI ISO

Reader description	Reader supporting various HF standards
Reader in production	EOL 6/2021
Reader technology	Contactless
Reader frequency	13.56MHz

Notes		
Reading of Mifare Classic sectors supported. Reading of Mifare DESFire data files supported. Mifare Classic cards with 7 byte UIN are not supported. Mifare Classic cards with random UIN not tested. SAM module supported. Implementation of SAM modules possible as a customization. Only all or just a single supported technology may be selected.		
Supported cfg items	Supported values	Note
rdrcfg.common.cardno_conv	(CARDCONV_APPENDLRC, CARDCONV_HEX2DEC, CARDCONV_HEX2DEC10, CARDCONV_HEX2DECLEFT16, CARDCONV_INTERFLEX, CARDCONV_ISO14443ATRUNCREVENDIAN, CARDCONV_JAVACARDSERIAL, CARDCONV_REVBITSINBYTE, CARDCONV_REVENDIAN, CARDCONV_REVENDIAN2DEC, CARDCONV_REVENDIANHEX2DEC10, CARDCONV_REVENDIANWOHIDPROX)	Builtin card number conversions. Will be overwritten if rdrcfg.cardno.cardnoconv specified
rdrcfg.cardno.cardnoconv		Custom specified card number conversion. CM2CNO("text") or CNOASM(FILE("filename"))).
rdrcfg.common.continuous	1/0	
rdrcfg.common.no_uin_check	1/0	required for cards with random UIN
rdrcfg.hf.hf_supported	HF_DEFAULT, HF_ISO14443A, HF_ISO14443B, HF_ISO15693, HF_ICODE_UID, HF_ICODE_EPC, HF_SR176, HF_ASKRFID	
rdrcfg.hf.hf_iso_select	1/0	required for DESFire/ISO reading
rdrcfg.hf.report_read_error	REP_NONE, REP_AUTO, REP_ERROR, REP_REFUSED	
rdrcfg.hf.hf_uin_revendian	(0/1)	Reverse endianness of UIN
rdrcfg.hf.mifare_read.mode	(MIFARE_CLASSIC)	
rdrcfg.hf.mifare_read.block	<0,240>	0-63 (Mifare 1k), 0-240 (Mifare 4k)
rdrcfg.hf.mifare_read.mifare_key.key_type	(KEY_MIFARE_A, KEY_MIFARE_B)	
rdrcfg.hf.mifare_read.mifare_key.key_data	data[6]	key data

Supported cfg items	Supported values	Note
rdrcfg.hf.mifare_read.card_type	uint	Card type to report instead of the default ISO standard
rdrcfg.hf.desfire_read.desfire_aid	data[3]	3 byte DESFire Application ID in endiannes reported from card
rdrcfg.hf.desfire_read.desfire_fid	<0,15>	file id
rdrcfg.hf.desfire_read.desfire_foffset	<0,65535>	offset in file
rdrcfg.hf.desfire_read.desfire_flen	<0,230>	0 returns complete file up to 230 bytes
rdrcfg.hf.desfire_read.desfire_fopts	{DESFIRE_FILE_AUTO, DESFIRE_FILE_PLAIN, DESFIRE_FILE_MACED, DESFIRE_FILE_ENCRYPTED, DESFIRE_FILE_STD, DESFIRE_FILE_VALUE, DESFIRE_FILE_RECORD}	file encryption options for read
rdrcfg.hf.desfire_read.desfire_key_no	<0,15>	
rdrcfg.hf.desfire_read.desfire_key.key_type	(KEY_DES, KEY_2K3DES, KEY_3K3DES, KEY_AES128, KEY_NONE)	
rdrcfg.hf.desfire_read.desfire_key.key_data	data[8,24]	key data
rdrcfg.hf.desfire_read.card_type	uint	Card type to report instead of the default ISO standard

Default configuration equivalent

```
rdrcfg.hf.hf_supported = HF_DEFAULT;
```

Protocol ID and name	Configuration used
Protocol 44 - Multi ISO UIN	# Default reader settings
Protocol 106 - UIN rev. endian	rdrcfg.hf.hf_uin_revendian = 1 rdrcfg.lf.lf_disabled = 1;
Protocol 72 - ASK-RFID UIN	rdrcfg.lf.lf_disabled = 1; rdrcfg.hf.hf_supported = HF_ASKRFID;

8.14 B-050 - UNIQUE

Reader description	Reader for LF cards with bidirectional communication
--------------------	--

Reader in production	Yes
Reader technology	Contactless
Reader frequency	125kHz

Notes

Supports only f/64 Manchester modulation on Hitag 1/S cards.
 Crypto authentication on Hitag1 and Hitag2 cards not supported. Only password mode supported.
 Do not use protocol 14, it may pick up some noise and interpret it as a card number on cards that require other card reader for proper operation.

Supported cfg items	Supported values	Note
rdrcfg.common.cardno_conv	(CARDCONV_APPENDLRC, CARDCONV_HEX2DEC, CARDCONV_HEX2DEC10, CARDCONV_HEX2DECLEFT16, CARDCONV_INTERFLEX, CARDCONV_ISO14443ATRUNCREVENTIAN, CARDCONV_JAVACARDSERIAL, CARDCONV_REVBITSINBYTE, CARDCONV_REVENTIAN, CARDCONV_REVENTIAN2DEC, CARDCONV_REVENTIANHEX2DEC10, CARDCONV_REVENTIANWOHIDPROX)	Builtin card number conversions. Will be overwritten if rdrcfg.cardno.cardnoconv specified
rdrcfg.cardno.cardnoconv		Custom specified card number conversion. CM2CNO("text") or CNOASM(FILE("filename"))).
rdrcfg.lf.lf_supported	LF_EM4000, LF_HITAG1S, LF_HITAG2, LF_EM4550, LF_IDROE, LF_HDX_IDROG	

Default configuration equivalent

```
rdrcfg.lf.lf_supported = LF_DEFAULT;
```

Protocol ID and name	Configuration used
Protocol 14 - All tags slow	# Default reader settings
Protocol 15 - IDROA	rdrcfg.lf.lf_supported = LF_EM4000;
Protocol 16 - IDROB	rdrcfg.lf.lf_supported = LF_HITAG1S; rdrcfg.hf.hf_disabled = 1;
Protocol 17 - IDROC	rdrcfg.lf.lf_supported = LF_HITAG2; rdrcfg.hf.hf_disabled = 1;
Protocol 18 - IDROD	rdrcfg.lf.lf_supported = LF_EM4550; rdrcfg.hf.hf_disabled = 1;

Protocol ID and name	Configuration used
Protocol 28 - IDROE	<code>rdrcfg.lf.lf_supported = LF_IDROE;</code>
Protocol 51 - IDROG	<code>rdrcfg.lf.lf_supported = LF_HDX_IDROG;</code>

8.15 B-051 - READER 3 UNIQUE

Reader description	Reader for LF cards with bidirectional communication
Reader in production	EOL 6/2021
Reader technology	Contactless
Reader frequency	125kHz

Notes

Supports only f/64 Manchester modulation on Hitag 1/S cards.
 Crypto authentication on Hitag1 and Hitag2 cards not supported. Only password mode supported.
 Do not use protocol 14, it may pick up some noise and interpret it as a card number on cards that require other card reader for proper operation.

Supported cfg items	Supported values	Note
<code>rdrcfg.common.cardno_conv</code>	(CARDCONV_APPENDLRC, CARDCONV_HEX2DEC, CARDCONV_HEX2DEC10, CARDCONV_HEX2DECLEFT16, CARDCONV_INTERFLEX, CARDCONV_ISO14443ATRUNCREVENDIAN, CARDCONV_JAVACARDSERIAL, CARDCONV_REVBITSINBYTE, CARDCONV_REVENDIAN, CARDCONV_REVENDIAN2DEC, CARDCONV_REVENDIANHEX2DEC10, CARDCONV_REVENDIANWOHIDPROX)	Builtin card number conversions. Will be overwritten if <code>rdrcfg.cardno.cardnoconv</code> specified
<code>rdrcfg.cardno.cardnoconv</code>		Custom specified card number conversion. <code>CM2CNO("text")</code> or <code>CNOASM(FILE("filename"))</code> .
<code>rdrcfg.lf.lf_supported</code>	LF_EM4000, LF_HITAG1S, LF_HITAG2, LF_EM4550, LF_IDROE, LF_HDX_IDROG	

Default configuration equivalent

`rdrcfg.lf.lf_supported = LF_DEFAULT;`

Protocol ID and name	Configuration used
Protocol 14 - All tags slow	# Default reader settings
Protocol 15 - IDROA	<code>rdrcfg.lf.lf_supported = LF_EM4000;</code>
Protocol 16 - IDROB	<code>rdrcfg.lf.lf_supported = LF_HITAG1S;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>
Protocol 17 - IDROC	<code>rdrcfg.lf.lf_supported = LF_HITAG2;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>
Protocol 18 - IDROD	<code>rdrcfg.lf.lf_supported = LF_EM4550;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>
Protocol 28 - IDROE	<code>rdrcfg.lf.lf_supported = LF_IDROE;</code>
Protocol 51 - IDROG	<code>rdrcfg.lf.lf_supported = LF_HDX_IDROG;</code>

8.16 B-052 - TIRIS 134KHZ

Reader description	Reader supporting Texas Instrument Tiris technology
Reader in production	Yes
Reader technology	Contactless
Reader frequency	134kHz

Notes

The card will be read from approximately 2cm from the card reader, not less. It is the technology feature. Only Tiris RO transponders supported

Supported cfg items	Supported values	Note
<code>rdrcfg.common.cardno_conv</code>	(<code>CARDCONV_APPENDLRC</code> , <code>CARDCONV_HEX2DEC</code> , <code>CARDCONV_HEX2DEC10</code> , <code>CARDCONV_HEX2DECLEFT16</code> , <code>CARDCONV_INTERFLEX</code> , <code>CARDCONV_ISO14443ATRUNCREVENDIAN</code> , <code>CARDCONV_JAVACARDSERIAL</code> , <code>CARDCONV_REVBITSINBYTE</code> , <code>CARDCONV_REVENDIAN</code> , <code>CARDCONV_REVENDIAN2DEC</code> , <code>CARDCONV_REVENDIANHEX2DEC10</code> , <code>CARDCONV_REVENDIANWOHIDPROX</code>)	Builtin card number conversions. Will be overwritten if <code>rdrcfg.cardno.cardnoconv</code> specified

Supported cfg items	Supported values	Note
rdrcfg.cardno.cardnoconv		Custom specified card number conversion. CM2CNO("text") or CNOASM(FILE("filename")).
rdrcfg.lf.lf_supported	LF_TIRIS	
Default configuration equivalent		
rdrcfg.lf.lf_supported = LF_TIRIS;		
Protocol ID and name	Configuration used	
Protocol 22 - Tiris	rdrcfg.lf.lf_supported = LF_TIRIS; rdrcfg.hf.hf_disabled = 1;	

8.17 B-054 - READER 3 TIRIS

Reader description	Reader supporting Texas Instrument Tiris technology	
Reader in production	EOL 6/2021	
Reader technology	Contactless	
Reader frequency	134kHz	
Notes		
The card will be read from approximately 2cm from the card reader, not less. It is the technology feature. Tiris RO and RW transponders supported		
Supported cfg items	Supported values	Note
rdrcfg.common.cardno_conv	(CARDCONV_APPENDLRC, CARDCONV_HEX2DEC, CARDCONV_HEX2DEC10, CARDCONV_HEX2DECLEFT16, CARDCONV_INTERFLEX, CARDCONV_ISO14443ATRUNCREVENDIAN, CARDCONV_JAVACARDSERIAL, CARDCONV_REVBITSINBYTE, CARDCONV_REVENDIAN, CARDCONV_REVENDIAN2DEC, CARDCONV_REVENDIANHEX2DEC10, CARDCONV_REVENDIANWOHIDPROX)	Builtin card number conversions. Will be overwritten if rdrcfg.cardno.cardnoconv specified
rdrcfg.cardno.cardnoconv		Custom specified card number conversion. CM2CNO("text") or CNOASM(FILE("filename"))).

Supported cfg items	Supported values	Note
rdrcfg.lf.lf_supported	LF_TIRIS	
Default configuration equivalent		
rdrcfg.lf.lf_supported = LF_TIRIS;		
Protocol ID and name	Configuration used	
Protocol 22 - Tiris	rdrcfg.lf.lf_supported = LF_TIRIS; rdrcfg.hf.hf_disabled = 1;	

8.18 B-060 - COTAG

Reader description	Reader supporting Cotag technology	
Reader in production	Yes	
Reader technology	Contactless	
Reader frequency	132 kHz	
Notes		
Cotag active and passive tags supported.		
Supported cfg items	Supported values	Note
rdrcfg.common.cardno_conv	(CARDCONV_APPENDLRC, CARDCONV_HEX2DEC, CARDCONV_HEX2DEC10, CARDCONV_HEX2DECLEFT16, CARDCONV_INTERFLEX, CARDCONV_ISO14443ATRUNCREVENDIAN, CARDCONV_JAVACARDSERIAL, CARDCONV_REVBITSINBYTE, CARDCONV_REVENDIAN, CARDCONV_REVENDIAN2DEC, CARDCONV_REVENDIANHEX2DEC10, CARDCONV_REVENDIANWOHIDPROX)	Builtin card number conversions. Will be overwritten if rdrcfg.cardno.cardnoconv specified
rdrcfg.cardno.cardnoconv		Custom specified card number conversion. CM2CNO("text") or CNOASM(FILE("filename")).
rdrcfg.lf.lf_supported	LF_COTAG	
Default configuration equivalent		
rdrcfg.lf.lf_supported = LF_COTAG;		

Protocol ID and name	Configuration used
Protocol 23 - Cotag	<code>rdrcfg.lf.lf_supported = LF_COTAG;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>
Protocol 97 - Cotag no batt. check	<code>rdrcfg.lf.lf_supported = LF_COTAG;</code> <code>rdrcfg.other.cotag_no_batt_check = 1;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>

8.19 B-061 - READER 3 COTAG

Reader description	Reader supporting Cotag technology
Reader in production	EOL 6/2021
Reader technology	Contactless
Reader frequency	132 kHz

Notes

Cotag active and passive tags supported.

Supported cfg items	Supported values	Note
<code>rdrcfg.common.cardno_conv</code>	(CARDCONV_APPENDLRC, CARDCONV_HEX2DEC, CARDCONV_HEX2DEC10, CARDCONV_HEX2DECLEFT16, CARDCONV_INTERFLEX, CARDCONV_ISO14443ATRUNCREVENDIAN, CARDCONV_JAVACARDSERIAL, CARDCONV_REVBITSINBYTE, CARDCONV_REVENDIAN, CARDCONV_REVENDIAN2DEC, CARDCONV_REVENDIANHEX2DEC10, CARDCONV_REVENDIANWOHIDPROX)	Builtin card number conversions. Will be overwritten if <code>rdrcfg.cardno.cardnoconv</code> specified
<code>rdrcfg.cardno.cardnoconv</code>		Custom specified card number conversion. <code>CM2CNO("text")</code> or <code>CNOASM(FILE("filename"))</code> .
<code>rdrcfg.lf.lf_supported</code>	LF_COTAG	

Default configuration equivalent

`rdrcfg.lf.lf_supported = LF_COTAG;`

Protocol ID and name	Configuration used
Protocol 23 - Cotag	<code>rdrcfg.lf.lf_supported = LF_COTAG;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>

Protocol ID and name	Configuration used
Protocol 97 - Cotag no batt. check	<code>rdrcfg.lf.lf_supported = LF_COTAG;</code> <code>rdrcfg.other.cotag_no_batt_check = 1;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>

8.20 B-063 - READER 3 INSIDE CONTACTLESS

Reader description	Reader supporting Inside Contactless HF technology and various HF standards.
Reader in production	EOL 6/2021
Reader technology	Contactless
Reader frequency	13.56MHz

Notes

Only 4 byte UIN on ISO14443A cards supported
FeliCa cards with shorter preamble supported

Supported cfg items	Supported values	Note
<code>rdrcfg.common.cardno_conv</code>	(CARDCONV_APPENDLRC, CARDCONV_HEX2DEC, CARDCONV_HEX2DEC10, CARDCONV_HEX2DECLEFT16, CARDCONV_INTERFLEX, CARDCONV_ISO14443ATRUNCREVENDIAN, CARDCONV_JAVACARDSERIAL, CARDCONV_REVBITSINBYTE, CARDCONV_REVENDIAN, CARDCONV_REVENDIAN2DEC, CARDCONV_REVENDIANHEX2DEC10, CARDCONV_REVENDIANWOHIDPROX)	Builtin card number conversions. Will be overwritten if <code>rdrcfg.cardno.cardnoconv</code> specified
<code>rdrcfg.cardno.cardnoconv</code>		Custom specified card number conversion. CM2CNO("text") or CNOASM(FILE("filename")).
<code>rdrcfg.hf.hf_supported</code>	HF_INSIDE, HF_ISO14443A, HF_ISO14443B, HF_ISO15693, HF_FELICA, HF_ASKRFID	

Default configuration equivalent

`rdrcfg.hf.hf_supported = HF_INSIDE;`

Protocol ID and name	Configuration used
Protocol 68 - Inside Contactless UIN	<code>rdrcfg.hf.hf_supported = HF_INSIDE;</code>

Protocol ID and name	Configuration used
Protocol 69 - ISO 14443A-3 UIN	<code>rdrcfg.hf.hf_supported = HF_ISO14443A;</code>
Protocol 70 - ISO 14443B-3 UIN	<code>rdrcfg.hf.hf_supported = HF_ISO14443B;</code>
Protocol 71 - ISO 15693-3 UIN	<code>rdrcfg.hf.hf_supported = HF_ISO15693;</code>
Protocol 72 - ASK-RFID UIN	<code>rdrcfg.lf.lf_disabled = 1;</code> <code>rdrcfg.hf.hf_supported = HF_ASKRFID;</code>
Protocol 73 - Felica UIN	<code>rdrcfg.hf.hf_supported = HF_FELICA;</code>

8.21 B-065 - READER 3 MF+

Reader description	Universal HF and LF reader supporting ASK and FSK modulation on LF and various HF standards. Includes support for HID Prox, HID iClass SE and HID iClass SEOS and HID iClass Elite technologies.
Reader in production	EOL 1/2022
Reader technology	Contactless
Reader frequency	13.56MHz / 125kHz

Notes

YSoft Configuration Cards supported within 2 minutes after startup and if processing of configuration cards is enabled.

Procedure for programming using YSoft Configuration cards: place the card for approximately 10s on the reader, wait for successful programming indication.

Reading of Mifare Classic sectors supported.

Reading of Mifare Classic sectors from 7 byte UIN or random UIN cards supported.

Reading of Mifare DESFire data files supported.

Reading of Mifare Plus sectors not supported at the moment (customization).

Reading of Inside Contactless PicoPass blocks supported.

ISO14443A/B-4 APDU commands supported.

Processing of multiple HF cards at the same time supported.

Only one ISO14443-B card supported in the field.

On LF UIN mode it may take longer to read the card for the first time.

Protocol 117 should not be used for user identification, it is a special protocol intended as an aid for card testing and some limitations may apply.

Uses HID iClass SE Processor.

HID iClass Elite key configuration cards are supported only on protocols with PACS data enabled (Protocols 1201, 1202, 1203, 1204, 1210, 1212, 1213).

HID iClass Elite key configuration cards are supported only 2 minutes after reader startup.

Once the SE Processor is programmed with iClass Elite keys then for reverting the reader into factory state the customer iClass Elite to standard keys programming cards must be used. There is no way how to revert it back to use default keys, even Y Soft manufacturing cannot do it.

Finished programming by a HID iClass Elite key configuration card is signaled by a special reader indication.

Supported cfg items	Supported values	Note
rdrcfg.common.cardno_conv	(CARDCONV_APPENDLRC, CARDCONV_HEX2DEC, CARDCONV_HEX2DEC10, CARDCONV_HEX2DECLEFT16, CARDCONV_INTERFLEX, CARDCONV_ISO14443ATRUNCREVENDIAN, CARDCONV_JAVACARDSERIAL, CARDCONV_REVBITSINBYTE, CARDCONV_REVENDIAN, CARDCONV_REVENDIAN2DEC, CARDCONV_REVENDIANHEX2DEC10, CARDCONV_REVENDIANWOHIDPROX)	Builtin card number conversions. Will be overwritten if rdrcfg.cardno.cardnoconv specified
rdrcfg.cardno.cardnoconv		Custom specified card number conversion. CM2CNO("text") or CNOASM(FILE("filename"))).
rdrcfg.common.continuous	(0, 1)	Special setting that enables card requested for login and card remove notification on some products.
rdrcfg.common.testing_mode	(0, 1)	Special testing setting to scan all supported technologies. Only for debug or testing purposes, DO NOT USE IN PRODUCTION ENVIRONMENT, some limitations may apply.
rdrcfg.lf.lf_disabled	(0, 1)	Completely disable processing of LF technologies
rdrcfg.lf.lf_supported	(LF_AUTO, LF_EM4000, LF_PROXLITE, LF_PARADOX, LF_NEDAP, LF_AWID, LF_PYRAMID, LF_DEISTER, LF_DATASEC, LF_GPROX, LF_IOPROX, LF_EM4050RO, LF_PAC, LF_URMET, LF_CARDAX, LF_JABLOTRON, LF_HIDPROX, LF_CUSTOM)	

Supported cfg items	Supported values	Note
rdrcfg.lf.lf_supported_multiple	[LF_AUTO, LF_EM4000, LF_PROXLITE, LF_PARADOX, LF_NEDAP, LF_AWID, LF_PYRAMID, LF_DEISTER, LF_DATASEC, LF_GPROX, LF_IOPROX, LF_EM4050R0, LF_PAC, LF_URMET, LF_CARDAX, LF_JABLOTRON, LF_HIDPROX]	Combining other technologies with LF_CUSTOM is limited.
rdrcfg.lf.hidprox.wiegand_format	(WAUTO, W26, W27, W32, W34, W37, W40, W40_PAR, W44, WABA2, WMAGSTRIPE)	
rdrcfg.lf.hidprox.hid_udf_enable	(0, 1)	Enable processing of HID Prox User Defined Formats - cards with non-standard customer code
rdrcfg.lf.hidprox.no_wiegand_parity_check	(0, 1)	Do not check parity on wiegand formats that contain parity
rdrcfg.lf.hidprox.no_error_report	(0, 1)	Do not report parity or other errors
rdrcfg.lf.lf_custom.modulation	(LF_ASK, LF_FSK2)	Modulation type
rdrcfg.lf.lf_custom.bitrate	(LF_F16, LF_F32, LF_F50, LF_F64)	Card bitrate
rdrcfg.lf.lf_custom.start_decoding	(LF_DECODE_NRZ, LF_DECODE_MANCHESTER, LF_DECODE_DIFFBIPHASE)	Decoding of start condition
rdrcfg.lf.lf_custom.data_decoding	(LF_DECODE_NRZ, LF_DECODE_MANCHESTER, LF_DECODE_DIFFBIPHASE)	Decoding of data
rdrcfg.lf.lf_custom.start_cond	u32	32 bit start condition
rdrcfg.lf.lf_custom.start_mask	u32	32 bit Start condition mask. Must be continuous sequence of bits on ASKFSK frontend
rdrcfg.lf.lf_custom.count_bits	int	Data bits count without start condition. If you do not specify complete bith length, then adjust ignore_bits for complete data reading

Supported cfg items	Supported values	Note
rdrcfg.lf.lf_custom.start_decode_invert	(0, 1)	Invert start condition data. Not applicable if LF_DECODE_DIFFBI PHASE is used
rdrcfg.lf.lf_custom.data_decode_invert	(0, 1)	Invert data. Not applicable if LF_DECODE_DIFFBI PHASE is used
rdrcfg.lf.lf_custom.input_invert	(0, 1)	Invert data incoming to decodes
rdrcfg.lf.lf_custom.read_count	int	How many times the data should be read and checked and compared as identical
rdrcfg.lf.lf_custom.padding_bits	int	Padding 0 bits before actual data output
rdrcfg.lf.lf_custom.ignore_bits	int	Bits at the end of the data stream that should be ignored (i.e. void bits), used for proper timing calculation
rdrcfg.lf.lf_custom.card_bootup_time	int	Value to add to card timeout - card bootup overhead in us
rdrcfg.lf.lf_custom.decode	(LF_DECODE_RAW, LF_DECODE_EM4K)	Use particular decode function
rdrcfg.lf.lf_custom.remove_timeout	uint	Timeout for card remove
rdrcfg.lf.lf_custom.card_type	uint	Card type to report
rdrcfg.hf.hf_disabled	(0, 1)	Completely disable processing of HF technologies
rdrcfg.common.no_uin_check	(0, 1)	Do not check if the card UIN is still the same. Required for cards with random UIN
rdrcfg.hf.hf_supported	(HF_DEFAULT, HF_ISO14443A, HF_ISO14443B, HF_ISO15693, HF_INSIDE, HF_FELICA, HF_SIELOX)	

Supported cfg items	Supported values	Note
rdrcfg.hf.hf_supported_multiple	[HF_ISO14443A, HF_ISO14443B, HF_ISO15693, HF_INSIDE, HF_FELICA, HF_SIELOX]	Multiple values supported
rdrcfg.hf.hf_uin_revendian	(0, 1)	Reverse endianness of UIN
rdrcfg.hf.report_read_error	(REP_NONE, REP_AUTO, REP_ERROR, REP_REFUSED)	Change default behaviour of error reporting if the card is not read correctly. By default the card is ignored but the reader may be configured to issue card read error or card refused messages.
rdrcfg.hf.hf_iso_select	(0, 1)	ISO14443A/B-4 ISO select. Required for DESFire/ISO14443A/B-4 commands.
rdrcfg.hf.hf_uin_mode	(HF_UIN_AUTO, HF_UIN_NORMAL, HF_UIN_LEGIC)	Configuration of how UIN should be reported. Some readers may have a different endianness on some technologies.
rdrcfg.hf.read_error_return_uin	(0, 1)	Any error in reading data areas will result in returning the card UIN. Use with caution as this lowers overall security.
rdrcfg.other.piv_mode	(PIV_MODE_AUTO, PIV_MODE_FASCN, PIV_MODE_GUID)	Specify mode of reading of PIV-II cards
rdrcfg.hf.config_card_disabled	(0, 1)	Disable processing of YSoft Configuration Cards
rdrcfg.hf.config_card_process_time	int	Time in msec for which the config card will be checked for presence before processing, default 8000

Supported cfg items	Supported values	Note
rdrcfg.hf.config_card_allow_time	int	Time in sec after startup in which configuration card is allowed, -1 means always enabled, default 120
rdrcfg.hf.iclass.allowed_data_models_multiple	[ICLASS_PICO15693_ICLASS, ICLASS_PICO15693_SIO, ICLASS_ISO14443A_SIO, ICLASS_ISO14443A_HID_MIFARE]	iClass data models, appropriate HF technologies must be enabled separately
rdrcfg.hf.iclass.allowed_card_types_multiple	[CARD_HID_DESFIRE_SE, CARD_HID_DESFIRE, CARD_HID_MIFARE_SE, CARD_HID_MIFARE, CARD_ICLASS_SE, CARD_ICLASS, CARD_HID_MIFARE_PLUS_SE, CARD_HID_MIFARE_PLUS, CARD_HID_SEOS]	If present allow only these card types, else any. Appropriate HF technologies must be enabled separately.
rdrcfg.hf.iclass.process_pac_bits	(0, 1)	Enable processing of data from HID iClass cards
rdrcfg.hf.iclass.no_pac_bits_return_uin	(0, 1)	Card with not valid data (or incompatible keys) will return UIN
rdrcfg.hf.iclass.not_allowed_cards_return_uin	(0, 1)	cards that are not allowed by allowed_card_types_multiple will return UIN
rdrcfg.hf.iclass.report_card_read_error	(0, 1)	If there is a problem with reading the card, return card read error otherwise ignore the card
rdrcfg.hf.iclass.report_card_refused	(0, 1)	If there is a problem with reading the card, return card refused otherwise ignore the card
rdrcfg.hf.iclass.config_card_allow_time	<1, 65535>	Time in seconds after reader startup in which configuration cards are allowed, -1 means always enabled

Supported cfg items	Supported values	Note
rdrcfg.hf.inside_read.key_type	(KEY_KD, KEY_KC)	Type of used key
rdrcfg.hf.inside_read.key	data[8]	key for protected area
rdrcfg.hf.inside_read.page	uint	Book and page in the book, lower nibble (4 bits) are page, bit 4 (LSB of higher nibble) is book. Works only on paged media. Example: book 1, page 2 = 0x12
rdrcfg.hf.inside_read.address	<0,255>	Read from n-th block (1 block = 8 bytes), 2K page: <0,31>; 16K page: <0,255>
rdrcfg.hf.inside_read.read_count	<1,128>	Read this many blocks (each has 8 bytes)
rdrcfg.hf.mifare_read.mode	(MIFARE_CLASSIC)	
rdrcfg.hf.mifare_read.block	<0,240>	0-63 (Mifare 1k), 0-240 (Mifare 4k)
rdrcfg.hf.mifare_read.mifare_key.key_type	(KEY_MIFARE_A, KEY_MIFARE_B)	
rdrcfg.hf.mifare_read.mifare_key.key_data	data[6]	key data
rdrcfg.hf.mifare_read.card_type	uint	Card type to report instead of the default ISO standard
rdrcfg.hf.desfire_read.desfire_aid	data[3]	3 byte DESFire Application ID in endiannes reported from card
rdrcfg.hf.desfire_read.desfire_fid	<0,15>	file id
rdrcfg.hf.desfire_read.desfire_foffset	<0,65535>	offset in file
rdrcfg.hf.desfire_read.desfire_flen	<0,230>	0 returns complete file up to 230 bytes
rdrcfg.hf.desfire_read.desfire_fopts	{DESFIRE_FILE_AUTO, DESFIRE_FILE_PLAIN, DESFIRE_FILE_MACED, DESFIRE_FILE_ENCRYPTED, DESFIRE_FILE_STD, DESFIRE_FILE_VALUE, DESFIRE_FILE_RECORD}	file encryption options for read
rdrcfg.hf.desfire_read.desfire_key_no	<0,15>	

Supported cfg items	Supported values	Note
rdrcfg.hf.desfire_read.desfire_key.key_type	(KEY_DES, KEY_2K3DES, KEY_3K3DES, KEY_AES128, KEY_NONE)	
rdrcfg.hf.desfire_read.desfire_key.key_data	data[8,24]	key data
rdrcfg.hf.desfire_read.card_type	uint	Card type to report instead of the default ISO standard
rdrcfg.hf.iso_read.read	(ISO_CHUID, ISO_CEPAS)	reading of predefined HF technologies
rdrcfg.hf.iso_read.card_type	uint	Card type to report instead of the default ISO standard

Default configuration equivalent

```
rdrcfg.lf.lf_supported = LF_AUTO;
rdrcfg.hf.hf_supported = HF_DEFAULT;
```

Protocol ID and name	Configuration used
Protocol 1220 - LF UIN + HF UIN	rdrcfg.lf.lf_supported = LF_AUTO
Protocol 1222 - LF UIN + HF UIN rev. endian	rdrcfg.lf.lf_supported = LF_AUTO rdrcfg.hf.hf_supported = HF_DEFAULT rdrcfg.hf.hf_uin_revendian = 1
Protocol 1221 - LF UIN only	rdrcfg.hf.hf_disabled = 1; rdrcfg.lf.lf_supported = LF_AUTO
Protocol 105 - HF UIN only	rdrcfg.lf.lf_disabled = 1;
Protocol 106 - HF UIN only rev. endian	rdrcfg.hf.hf_uin_revendian = 1 rdrcfg.lf.lf_disabled = 1;
Protocol 1500 - HF UIN only continuous	rdrcfg.lf.lf_disabled = 1; rdrcfg.common.continuous = 1;
Protocol 1518 - ISO14443A UIN only	rdrcfg.lf.lf_disabled = 1; rdrcfg.hf.hf_supported = HF_ISO14443A;
Protocol 125 - EM4000 compat.+ HID Prox + HF UIN	rdrcfg.lf.lf_supported_multiple.extend([LF_EM4000, LF_HIDPROX]);
Protocol 90 - EM4000 compat.+ HF UIN	rdrcfg.lf.lf_supported = LF_EM4000;

Protocol ID and name	Configuration used
Protocol 94 - EM4000 compat. + HF UIN cont.	<code>rdrcfg.common.continuous = 1;</code> <code>rdrcfg.lf.lf_supported = LF_EM4000;</code>
Protocol 118 - EM4000 compat.+ ISO14443A UIN	<code>rdrcfg.lf.lf_supported = LF_EM4000;</code> <code>rdrcfg.hf.hf_supported = HF_ISO14443A;</code>
Protocol 91 - Prox-lite/Casi-Rusco + HF UIN	<code>rdrcfg.lf.lf_supported = LF_PROXLITE;</code>
Protocol 92 - Paradox/PosiProx + HF UIN	<code>rdrcfg.lf.lf_supported = LF_PARADOX;</code>
Protocol 89 - Nedap + HF UIN	<code>rdrcfg.lf.lf_supported = LF_NEDAP;</code>
Protocol 87 - Awid + HF UIN	<code>rdrcfg.lf.lf_supported = LF_AWID;</code>
Protocol 86 - Pyramid + HF UIN	<code>rdrcfg.lf.lf_supported = LF_PYRAMID;</code>
Protocol 85 - Deister + HF UIN	<code>rdrcfg.lf.lf_supported = LF_DEISTER;</code>
Protocol 84 - Datasec + HF UIN	<code>rdrcfg.lf.lf_supported = LF_DATASEC;</code>
Protocol 83 - G-Prox + HF UIN	<code>rdrcfg.lf.lf_supported = LF_GPROX;</code>
Protocol 82 - ioProx label + HF UIN	<code>rdrcfg.lf.lf_supported = LF_IOPROX;</code>
Protocol 121 - EM4050 f/64 std. mode + HF UIN	<code>rdrcfg.lf.lf_supported = LF_EM4050R0;</code>
Protocol 96 - PAC/KeyPAC + HF UIN	<code>rdrcfg.lf.lf_supported_multiple.extend([LF_PAC, LF_KEYPAC]);</code>
Protocol 120 - URMET/FDI + HF UIN	<code>rdrcfg.lf.lf_supported = LF_URMET;</code>
Protocol 1601 - Cardax/Gallagher + HF UIN	<code>rdrcfg.lf.lf_supported = LF_CARDAX;</code>
Protocol 1602 - Jablotron + HF UIN	<code>rdrcfg.lf.lf_supported = LF_JABLOTRON;</code>
Protocol 35 - HID Prox + HF UIN	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = WAUTO;</code>

Protocol ID and name	Configuration used
Protocol 124 - HID Prox + HF UIN rev. endian	<pre>rdrcfg.hf.hf_uin_revendian = 1; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;</pre>
Protocol 1035 - HID Prox UDF + HF UIN	<pre>rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO; rdrcfg.lf.hidprox.hid_udf_enable = 1;</pre>
Protocol 1201 - HID iClass + ISO14443A UIN compat.	<pre>rdrcfg.common.cardno_conv = CARDCONV_ISO14443ATRUNCREVENDIAN; rdrcfg.hf.hf_supported_multiple.extend([HF_ICLASS, HF_ISO14443A]); rdrcfg.hf.iclass.process_pac_bits = 1; rdrcfg.hf.iclass.not_allowed_cards_return_uin = 1; rdrcfg.lf.lf_disabled = 1; rdrcfg.hf.iclass.allowed_data_models_multiple.extend([ICLASS_ PIC015693_ICLASS]);</pre>
Protocol 1202 - HID iClass + HID Prox + ISO14443A UIN compat.	<pre>rdrcfg.common.cardno_conv = CARDCONV_ISO14443ATRUNCREVENDIAN; rdrcfg.hf.hf_supported_multiple.extend([HF_ICLASS, HF_ISO14443A]); rdrcfg.hf.iclass.process_pac_bits = 1; rdrcfg.hf.iclass.not_allowed_cards_return_uin = 1; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO; rdrcfg.hf.iclass.allowed_data_models_multiple.extend([ICLASS_ PIC015693_ICLASS]);</pre>
Protocol 1203 - HID iClass + ISO14443A UIN	<pre>rdrcfg.hf.hf_supported_multiple.extend([HF_ICLASS, HF_ISO14443A]); rdrcfg.hf.iclass.process_pac_bits = 1; rdrcfg.hf.iclass.not_allowed_cards_return_uin = 1; rdrcfg.lf.lf_disabled = 1; rdrcfg.hf.iclass.allowed_data_models_multiple.extend([ICLASS_ PIC015693_ICLASS]);</pre>
Protocol 1204 - HID iClass + HID Prox + ISO14443A UIN	<pre>rdrcfg.hf.hf_supported_multiple.extend([HF_ICLASS, HF_ISO14443A]); rdrcfg.hf.iclass.process_pac_bits = 1; rdrcfg.hf.iclass.not_allowed_cards_return_uin = 1; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO; rdrcfg.hf.iclass.allowed_data_models_multiple.extend([ICLASS_ PIC015693_ICLASS]);</pre>
Protocol 1210 - HID iClass + HID SIO	<pre>rdrcfg.hf.hf_supported_multiple.extend([HF_ICLASS, HF_ISO14443A]); rdrcfg.hf.iclass.process_pac_bits = 1; rdrcfg.lf.lf_disabled = 1;</pre>

Protocol ID and name	Configuration used
Protocol 1211 - HID SIO only	<pre>rdrcfg.hf.hf_supported_multiple.extend([HF_ICLASS, HF_ISO14443A]); rdrcfg.hf.iclass.process_pac_bits = 1; rdrcfg.lf.lf_disabled = 1; rdrcfg.hf.iclass.allowed_data_models_multiple.extend([ICLASS_ PIC015693_SIO, ICLASS_ISO14443A_SIO, ICLASS_ISO14443A_HID_MIFARE]); rdrcfg.hf.iclass.allowed_card_types_multiple.extend([CARD_HID_ _DESFIRE_SE, CARD_HID_MIFARE_SE, CARD_ICLASS_SE, CARD_HID_MIFARE_PLUS_SE, CARD_HID_SEOS]);</pre>
Protocol 1212 - HID iClass + HID SIO + HID Prox	<pre>rdrcfg.hf.hf_supported_multiple.extend([HF_ICLASS, HF_ISO14443A]); rdrcfg.hf.iclass.process_pac_bits = 1; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;</pre>
Protocol 1213 - HID iClass + HID SIO + HID Prox + EM4000	<pre>rdrcfg.hf.hf_supported_multiple.extend([HF_ICLASS, HF_ISO14443A]); rdrcfg.hf.iclass.process_pac_bits = 1; rdrcfg.lf.lf_supported_multiple.extend([LF_EM4000, LF_HIDPROX]); rdrcfg.lf.hidprox.wiegand_format = WAUTO;</pre>
Protocol 37 - HID iClass UIN	<pre>rdrcfg.hf.hf_supported = HF_ICLASS; rdrcfg.lf.lf_disabled = 1;</pre>
Protocol 76 - HID iClass UIN + HID Prox	<pre>rdrcfg.hf.hf_supported = HF_ICLASS; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;</pre>
Protocol 93 - HID Prox only	<pre>rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO; rdrcfg.hf.hf_disabled = 1;</pre>
Protocol 114 - EM4000 compatible only	<pre>rdrcfg.lf.lf_supported = LF_EM4000; rdrcfg.hf.hf_disabled = 1;</pre>
Protocol 80 - FIPS201 ID from PIV-II	<pre>rdrcfg.hf.hf_iso_select = 1; rdrcfg.hf.iso_read.read = ISO_CHUID; rdrcfg.hf.hf_supported = HF_ISO14443A; rdrcfg.lf.lf_disabled = 1;</pre>
Protocol 1070 - CEPAS CAN	<pre>rdrcfg.hf.hf_iso_select = 1; rdrcfg.hf.hf_supported = HF_ISO14443B; rdrcfg.hf.iso_read.read = ISO_CEPAS; rdrcfg.common.no_uin_check = 1; rdrcfg.lf.lf_disabled = 1;</pre>

Protocol ID and name	Configuration used
<i>Protocol 117 - LF testing + HID SIO + HF UIN</i>	<code>rdrcfg.common.testing_mode = 1;</code> <code>rdrcfg.common.exact_cardtype = 1;</code> <code>rdrcfg.sam.sam_mode = SAM_AUTO;</code>

8.22 B-067 - INSIDE CONTACTLESS

Reader description	Reader supporting Inside Contactless HF technology and various HF standards.
Reader in production	Yes
Reader technology	Contactless
Reader frequency	13.56MHz

Notes

Only 4 byte UIN on ISO14443A cards supported
FeliCa cards with shorter preamble supported

Supported cfg items	Supported values	Note
<code>rdrcfg.common.cardno_conv</code>	(CARDCONV_APPENDLRC, CARDCONV_HEX2DEC, CARDCONV_HEX2DEC10, CARDCONV_HEX2DECLEFT16, CARDCONV_INTERFLEX, CARDCONV_ISO14443ATRUNCREVENDIAN, CARDCONV_JAVACARDSERIAL, CARDCONV_REVBITSINBYTE, CARDCONV_REVENDIAN, CARDCONV_REVENDIAN2DEC, CARDCONV_REVENDIANHEX2DEC10, CARDCONV_REVENDIANWOHIDPROX)	Builtin card number conversions. Will be overwritten if <code>rdrcfg.cardno.cardnoconv</code> specified
<code>rdrcfg.cardno.cardnoconv</code>		Custom specified card number conversion. CM2CNO("text") or CNOASM(FILE("filename")).
<code>rdrcfg.hf.hf_supported</code>	HF_INSIDE, HF_ISO14443A, HF_ISO14443B, HF_ISO15693, HF_FELICA, HF_ASKRFID	

Default configuration equivalent

`rdrcfg.hf.hf_supported = HF_INSIDE;`

Protocol ID and name	Configuration used
Protocol 68 - Inside Contactless UIN	<code>rdrcfg.hf.hf_supported = HF_INSIDE;</code>

Protocol ID and name	Configuration used
Protocol 69 - ISO 14443A-3 UIN	<code>rdrcfg.hf.hf_supported = HF_ISO14443A;</code>
Protocol 70 - ISO 14443B-3 UIN	<code>rdrcfg.hf.hf_supported = HF_ISO14443B;</code>
Protocol 71 - ISO 15693-3 UIN	<code>rdrcfg.hf.hf_supported = HF_ISO15693;</code>
Protocol 72 - ASK-RFID UIN	<code>rdrcfg.lf.lf_disabled = 1;</code> <code>rdrcfg.hf.hf_supported = HF_ASKRFID;</code>
Protocol 73 - Felica UIN	<code>rdrcfg.hf.hf_supported = HF_FELICA;</code>

8.23 B-073 - READER 3 MF SAM

Reader description	Universal HF and LF reader supporting ASK and FSK modulation on LF and various HF standards. In addition it contains MIFARE SAM AV2 security coprocessor that stores keys for communication with encrypted HF cards.
Reader in production	Yes
Reader technology	Contactless
Reader frequency	13.56MHz / 125kHz

Notes

YSoft Configuration Cards supported within 2 minutes after startup and if processing of configuration cards is enabled.

Procedure for programming using YSoft Configuration cards: place the card for approximately 10s on the reader, wait for successful programming indication.

Blank MIFARE SAM AV2 in SIM format supplied inserted in the reader.

Using of MIFARE SAM AV2 for card reading available as a customization.

External MIFARE SAM AV2 programming before manufacturing available as a customization.

Reading of Mifare Classic sectors supported.

Reading of Mifare Classic sectors from 7 byte UIN or random UIN cards supported.

Reading of Mifare DESFire data files supported.

Reading of Mifare Plus sectors not supported at the moment (customization).

Reading of Inside Contactless PicoPass blocks supported.

ISO14443A/B-4 APDU commands supported.

Processing of multiple HF cards at the same time supported.

Only one ISO14443-B card supported in the field.

On LF UIN mode it may take longer to read the card for the first time.

Protocol 117 should not be used for user identification, it is a special protocol intended as an aid for card testing and some limitations may apply.

Supported cfg items	Supported values	Note
rdrcfg.common.cardno_conv	(CARDCONV_APPENDLRC, CARDCONV_HEX2DEC, CARDCONV_HEX2DEC10, CARDCONV_HEX2DECLEFT16, CARDCONV_INTERFLEX, CARDCONV_ISO14443ATRUNCREVENDIAN, CARDCONV_JAVACARDSERIAL, CARDCONV_REVBITSINBYTE, CARDCONV_REVENDIAN, CARDCONV_REVENDIAN2DEC, CARDCONV_REVENDIANHEX2DEC10, CARDCONV_REVENDIANWOHIDPROX)	Builtin card number conversions. Will be overwritten if rdrcfg.cardno.cardnoconv specified
rdrcfg.cardno.cardnoconv		Custom specified card number conversion. CM2CNO("text") or CNOASM(FILE("filename"))).
rdrcfg.common.continuous	(0, 1)	Special setting that enables card requested for login and card remove notification on some products.
rdrcfg.common.testing_mode	(0, 1)	Special testing setting to scan all supported technologies. Only for debug or testing purposes, DO NOT USE IN PRODUCTION ENVIRONMENT, some limitations may apply.
rdrcfg.lf.lf_disabled	(0, 1)	Completely disable processing of LF technologies
rdrcfg.lf.lf_supported	(LF_AUTO, LF_EM4000, LF_PROXLITE, LF_PARADOX, LF_NEDAP, LF_AWID, LF_PYRAMID, LF_DEISTER, LF_DATASEC, LF_GPROX, LF_IOPROX, LF_EM4050RO, LF_PAC, LF_URMET, LF_CARDAX, LF_JABLOTRON, LF_CUSTOM)	
rdrcfg.lf.lf_supported_multiple	[LF_AUTO, LF_EM4000, LF_PROXLITE, LF_PARADOX, LF_NEDAP, LF_AWID, LF_PYRAMID, LF_DEISTER, LF_DATASEC, LF_GPROX, LF_IOPROX, LF_EM4050RO, LF_PAC, LF_URMET, LF_CARDAX, LF_JABLOTRON]	Combining other technologies with LF_CUSTOM is limited.

Supported cfg items	Supported values	Note
rdrcfg.lf.lf_custom.modulation	(LF_ASK, LF_FSK2)	Modulation type
rdrcfg.lf.lf_custom.bitrate	(LF_F16, LF_F32, LF_F50, LF_F64)	Card bitrate
rdrcfg.lf.lf_custom.start_decoding	(LF_DECODE_NRZ, LF_DECODE_MANCHESTER, LF_DECODE_DIFFBIPHASE)	Decoding of start condition
rdrcfg.lf.lf_custom.data_decoding	(LF_DECODE_NRZ, LF_DECODE_MANCHESTER, LF_DECODE_DIFFBIPHASE)	Decoding of data
rdrcfg.lf.lf_custom.start_cond	u32	32 bit start condition
rdrcfg.lf.lf_custom.start_mask	u32	32 bit Start condition mask. Must be continuous sequence of bits on ASKFSK frontend
rdrcfg.lf.lf_custom.count_bits	int	Data bits count without start condition. If you do not specify complete bith length, then adjust ignore_bits for complete data reading
rdrcfg.lf.lf_custom.start_decode_invert	(0, 1)	Invert start condition data. Not applicable if LF_DECODE_DIFFBI PHASE is used
rdrcfg.lf.lf_custom.data_decode_invert	(0, 1)	Invert data. Not applicable if LF_DECODE_DIFFBI PHASE is used
rdrcfg.lf.lf_custom.input_invert	(0, 1)	Invert data incoming to decodes
rdrcfg.lf.lf_custom.read_count	int	How many times the data should be read and checked and compared as identical
rdrcfg.lf.lf_custom.padding_bits	int	Padding 0 bits before actual data output

Supported cfg items	Supported values	Note
rdrcfg.lf.lf_custom.ignore_bits	int	Bits at the end of the data stream that should be ignored (i.e. void bits), used for proper timing calculation
rdrcfg.lf.lf_custom.card_bootup_time	int	Value to add to card timeout - card bootup overhead in us
rdrcfg.lf.lf_custom.decode	(LF_DECODE_RAW, LF_DECODE_EM4K)	Use particular decode function
rdrcfg.lf.lf_custom.remove_timeout	uint	Timeout for card remove
rdrcfg.lf.lf_custom.card_type	uint	Card type to report
rdrcfg.hf.hf_disabled	(0, 1)	Completely disable processing of HF technologies
rdrcfg.common.no_uin_check	(0, 1)	Do not check if the card UIN is still the same. Required for cards with random UIN
rdrcfg.hf.hf_supported	(HF_DEFAULT, HF_ISO14443A, HF_ISO14443B, HF_ISO15693, HF_INSIDE, HF_FELICA, HF_SIELOX)	
rdrcfg.hf.hf_supported_multiple	[HF_ISO14443A, HF_ISO14443B, HF_ISO15693, HF_INSIDE, HF_FELICA, HF_SIELOX]	Multiple values supported
rdrcfg.hf.hf_uin_revendian	(0, 1)	Reverse endianness of UIN
rdrcfg.hf.report_read_error	(REP_NONE, REP_AUTO, REP_ERROR, REP_REFUSED)	Change default behaviour of error reporting if the card is not read correctly. By default the card is ignored but the reader may be configured to issue card read error or card refused messages.

Supported cfg items	Supported values	Note
rdrcfg.hf.hf_iso_select	(0, 1)	ISO14443A/B-4 ISO select. Required for DESFire/ISO14443A/B-4 commands.
rdrcfg.hf.hf_uin_mode	(HF_UIN_AUTO, HF_UIN_NORMAL, HF_UIN_LEGIC)	Configuration of how UIN should be reported. Some readers may have a different endianness on some technologies.
rdrcfg.hf.read_error_return_uin	(0, 1)	Any error in reading data areas will result in returning the card UIN. Use with caution as this lowers overall security.
rdrcfg.other.piv_mode	(PIV_MODE_AUTO, PIV_MODE_FASCN, PIV_MODE_GUID)	Specify mode of reading of PIV-II cards
rdrcfg.hf.config_card_disabled	(0, 1)	Disable processing of YSoft Configuration Cards
rdrcfg.hf.config_card_process_time	int	Time in msec for which the config card will be checked for presence before processing, default 8000
rdrcfg.hf.config_card_allow_time	int	Time in sec after startup in which configuration card is allowed, -1 means always enabled, default 120
rdrcfg.hf.inside_read.key_type	(KEY_KD, KEY_KC)	Type of used key
rdrcfg.hf.inside_read.key	data[8]	key for protected area
rdrcfg.hf.inside_read.page	uint	Book and page in the book, lower nibble (4 bits) are page, bit 4 (LSB of higher nibble) is book. Works only on paged media. Example: book 1, page 2 = 0x12

Supported cfg items	Supported values	Note
rdrcfg.hf.inside_read.address	<0,255>	Read from n-th block (1 block = 8 bytes), 2K page: <0,31>; 16K page: <0,255>
rdrcfg.hf.inside_read.read_count	<1,128>	Read this many blocks (each has 8 bytes)
rdrcfg.hf.mifare_read.mode	(MIFARE_CLASSIC)	
rdrcfg.hf.mifare_read.block	<0,240>	0-63 (Mifare 1k), 0-240 (Mifare 4k)
rdrcfg.hf.mifare_read.mifare_key.key_type	(KEY_MIFARE_A, KEY_MIFARE_B)	
rdrcfg.hf.mifare_read.mifare_key.key_data	data[6]	key data
rdrcfg.hf.mifare_read.card_type	uint	Card type to report instead of the default ISO standard
rdrcfg.hf.mifare_read.mifare_key.sam_key_no	<0,127>	Authentication key number
rdrcfg.hf.mifare_read.mifare_key.key_version	<0,255>	Version of authentication key
rdrcfg.hf.mifare_read.mifare_key.key_div	(DIV_NONE, DIV_UIN)	Key diversification method
rdrcfg.hf.desfire_read.desfire_aid	data[3]	3 byte DESFire Application ID in endiannes reported from card
rdrcfg.hf.desfire_read.desfire_fid	<0,15>	file id
rdrcfg.hf.desfire_read.desfire_foffset	<0,65535>	offset in file
rdrcfg.hf.desfire_read.desfire_flen	<0,230>	0 returns complete file up to 230 bytes
rdrcfg.hf.desfire_read.desfire_fopts	{DESFIRE_FILE_AUTO, DESFIRE_FILE_PLAIN, DESFIRE_FILE_MACED, DESFIRE_FILE_ENCRYPTED, DESFIRE_FILE_STD, DESFIRE_FILE_VALUE, DESFIRE_FILE_RECORD}	file encryption options for read
rdrcfg.hf.desfire_read.desfire_key_no	<0,15>	
rdrcfg.hf.desfire_read.desfire_key.key_type	(KEY_DES, KEY_2K3DES, KEY_3K3DES, KEY_AES128, KEY_NONE)	
rdrcfg.hf.desfire_read.desfire_key.key_data	data[8,24]	key data

Supported cfg items	Supported values	Note
rdrcfg.hf.desfire_read.card_type	uint	Card type to report instead of the default ISO standard
rdrcfg.hf.desfire_read.desfire_key.sam_key_no	<0,127>	Authentication key number
rdrcfg.hf.desfire_read.desfire_key.key_version	<0,255>	Version of authentication key
rdrcfg.hf.desfire_read.desfire_key.key_div	(DIV_NONE, DIV_UIN)	Key diversification method
rdrcfg.sam.sam_mode	(SAM_NONE, SAM_MIFAREAV2, SAM_HIDSEPROC, SAM_AUTO)	Type of SAM
rdrcfg.sam.sam_auth	(SAM_NO_AUTH, SAM_AV2_UNLOCK, SAM_AV2_AUTH_PLAIN)	Authentication mode of SAM
rdrcfg.sam.auth_key_num	<0,127>	Authentication key number
rdrcfg.sam.auth_key.key_type	(KEY_AES128, KEY_AES192)	Authentication key type
rdrcfg.sam.auth_key.key_version	<0,255>	Version of authentication key
rdrcfg.sam.auth_key.key_data	data[16,24]	key data
rdrcfg.hf.iso_read.read	(ISO_CHUID, ISO_CEPAS)	reading of predefined HF technologies
rdrcfg.hf.iso_read.card_type	uint	Card type to report instead of the default ISO standard
Default configuration equivalent		
rdrcfg.lf.lf_supported = LF_AUTO; rdrcfg.hf.hf_supported = HF_DEFAULT;		
Protocol ID and name	Configuration used	
Protocol 1220 - LF UIN + HF UIN	rdrcfg.lf.lf_supported = LF_AUTO	
Protocol 1222 - LF UIN + HF UIN rev. endian	rdrcfg.lf.lf_supported = LF_AUTO rdrcfg.hf.hf_supported = HF_DEFAULT rdrcfg.hf.hf_uin_revendian = 1	
Protocol 1221 - LF UIN only	rdrcfg.hf.hf_disabled = 1; rdrcfg.lf.lf_supported = LF_AUTO	

Protocol ID and name	Configuration used
Protocol 105 - HF UIN only	<code>rdrcfg.lf.lf_disabled = 1;</code>
Protocol 106 - HF UIN only rev. endian	<code>rdrcfg.hf.hf_uin_revendian = 1</code> <code>rdrcfg.lf.lf_disabled = 1;</code>
Protocol 1500 - HF UIN only continuous	<code>rdrcfg.lf.lf_disabled = 1;</code> <code>rdrcfg.common.continuous = 1;</code>
Protocol 1518 - ISO14443A UIN only	<code>rdrcfg.lf.lf_disabled = 1;</code> <code>rdrcfg.hf.hf_supported = HF_ISO14443A;</code>
Protocol 90 - EM4000 compat.+ HF UIN	<code>rdrcfg.lf.lf_supported = LF_EM4000;</code>
Protocol 94 - EM4000 compat. + HF UIN cont.	<code>rdrcfg.common.continuous = 1;</code> <code>rdrcfg.lf.lf_supported = LF_EM4000;</code>
Protocol 118 - EM4000 compat.+ ISO14443A UIN	<code>rdrcfg.lf.lf_supported = LF_EM4000;</code> <code>rdrcfg.hf.hf_supported = HF_ISO14443A;</code>
Protocol 91 - Prox-lite/Casi-Rusco + HF UIN	<code>rdrcfg.lf.lf_supported = LF_PROXLITE;</code>
Protocol 92 - Paradox/PosiProx + HF UIN	<code>rdrcfg.lf.lf_supported = LF_PARADOX;</code>
Protocol 89 - Nedap + HF UIN	<code>rdrcfg.lf.lf_supported = LF_NEDAP;</code>
Protocol 87 - Awid + HF UIN	<code>rdrcfg.lf.lf_supported = LF_AWID;</code>
Protocol 86 - Pyramid + HF UIN	<code>rdrcfg.lf.lf_supported = LF_PYRAMID;</code>
Protocol 85 - Deister + HF UIN	<code>rdrcfg.lf.lf_supported = LF_DEISTER;</code>
Protocol 84 - Datasec + HF UIN	<code>rdrcfg.lf.lf_supported = LF_DATASEC;</code>
Protocol 83 - G-Prox + HF UIN	<code>rdrcfg.lf.lf_supported = LF_GPROX;</code>
Protocol 82 - ioProx label + HF UIN	<code>rdrcfg.lf.lf_supported = LF_IOPROX;</code>
Protocol 121 - EM4050 f/64 std. mode + HF UIN	<code>rdrcfg.lf.lf_supported = LF_EM4050R0;</code>

Protocol ID and name	Configuration used
Protocol 96 - PAC/KeyPAC + HF UIN	<code>rdrcfg.lf.lf_supported_multiple.extend([LF_PAC, LF_KEYPAC]);</code>
Protocol 120 - URMET/FDI + HF UIN	<code>rdrcfg.lf.lf_supported = LF_URMET;</code>
Protocol 1601 - Cardax/Gallagher + HF UIN	<code>rdrcfg.lf.lf_supported = LF_CARDAX;</code>
Protocol 1602 - Jablotron + HF UIN	<code>rdrcfg.lf.lf_supported = LF_JABLOTRON;</code>
Protocol 117 - ASK/FSK testing + HF UIN	<code>rdrcfg.common.testing_mode = 1;</code> <code>rdrcfg.common.exact_cardtype = 1;</code> <code>rdrcfg.sam.sam_mode = SAM_AUTO;</code>

8.24 B-074 - READER 3 MFX

Reader description	Universal HF and LF reader supporting ASK, FSK and PSK modulation on LF and various HF standards. Includes support for HID Prox, Indala, HID iClass SE and HID iClass SEOS and HID iClass Elite technologies.
Reader in production	Yes
Reader technology	Contactless
Reader frequency	13.56MHz / 134.2kHz / 132kHz / 125kHz

Notes

YSoft Configuration Cards supported within 2 minutes after startup and if processing of configuration cards is enabled.

Procedure for programming using YSoft Configuration cards: place the card for approximately 10s on the reader, wait for successful programming indication.

NFC functionality requires customization.

Reading of Mifare Classic sectors supported.

Reading of Mifare Classic sectors from 7 byte UIN or random UIN cards supported.

Reading of Mifare DESFire data files supported.

Reading of Mifare Plus sectors not supported at the moment (customization).

Reading of Inside Contactless PicoPass blocks supported.

ISO14443A/B-4 APDU commands supported.

Processing of multiple HF cards at the same time supported.

Only one ISO14443-B card supported in the field.

Tiris cards cannot be placed too close to the reader otherwise they are not read. It is a standard limitation of Tiris cards.

EM4050RO mode not supported in LF UIN as it interferes with EM4050 UIN mode.

Q5 / T5555 compatible cards may require customization depending on reading requirements.

Protocol 117 should not be used for user identification, it is a special protocol intended as an aid for card testing and some limitations may apply.

Reader uses Indala and HID Prox license from HID Corporation.

Indala ASP, ASP+ and ECR cards supported.

Indala option cards not supported.

Indala custom formats possible as a custom card reader configuration.

Uses HID iClass SE Processor.

HID iClass Elite key configuration cards are supported only on protocols with PACS data enabled (Protocols 1201, 1202, 1203, 1204, 1210, 1212, 1213).

HID iClass Elite key configuration cards are supported only 2 minutes after reader startup.

Once the SE Processor is programmed with iClass Elite keys then for reverting the reader into factory state the customer iClass Elite to standard keys programming cards must be used. There is no way how to revert it back to use default keys, even Y Soft manufacturing cannot do it.

Finished programming by a HID iClass Elite key configuration card is signaled by a special reader indication.

Secondary HID SE Processor supported in SAM slot.

Supported cfg items	Supported values	Note
rdrcfg.common.cardno_conv	(CARDCONV_APPENDLRC, CARDCONV_HEX2DEC, CARDCONV_HEX2DEC10, CARDCONV_HEX2DECLEFT16, CARDCONV_INTERFLEX, CARDCONV_ISO14443ATRUNCREVENDIAN, CARDCONV_JAVACARDSERIAL, CARDCONV_REVBITSINBYTE, CARDCONV_REVENDIAN, CARDCONV_REVENDIAN2DEC, CARDCONV_REVENDIANHEX2DEC10, CARDCONV_REVENDIANWOHIDPROX)	Builtin card number conversions. Will be overwritten if rdrcfg.cardno.cardnoconv specified
rdrcfg.cardno.cardnoconv		Custom specified card number conversion. CM2CNO("text") or CNOASM(FILE("filename"))).

Supported cfg items	Supported values	Note
rdrcfg.common.continuous	(0, 1)	Special setting that enables card requested for login and card remove notification on some products.
rdrcfg.common.testing_mode	(0, 1)	Special testing setting to scan all supported technologies. Only for debug or testing purposes, DO NOT USE IN PRODUCTION ENVIRONMENT, some limitations may apply.
rdrcfg.lf.lf_disabled	(0, 1)	Completely disable processing of LF technologies
rdrcfg.lf.lf_supported	(LF_AUTO, LF_EM4000, LF_PROXLITE, LF_PARADOX, LF_NEDAP, LF_AWID, LF_PYRAMID, LF_DEISTER, LF_DATASEC, LF_GPROX, LF_IOPROX, LF_EM4050RO, LF_PAC, LF_URMET, LF_CARDAX, LF_JABLOTRON, LF_INDALA, LF_NEXKEY, LF_COTAG, LF_TIRIS, LF_KEYPAC, LF_IDTECK, LF_NORALSY, LF_KERI, LF_HITAG1S, LF_HITAG2, LF_EM4550, LF_HIDPROX, LF_CUSTOM)	
rdrcfg.lf.lf_supported_multiple	[LF_AUTO, LF_EM4000, LF_PROXLITE, LF_PARADOX, LF_NEDAP, LF_AWID, LF_PYRAMID, LF_DEISTER, LF_DATASEC, LF_GPROX, LF_IOPROX, LF_EM4050RO, LF_PAC, LF_URMET, LF_CARDAX, LF_JABLOTRON, LF_INDALA, LF_NEXKEY, LF_COTAG, LF_TIRIS, LF_KEYPAC, LF_IDTECK, LF_NORALSY, LF_KERI, LF_HITAG1S, LF_HITAG2, LF_EM4550, LF_HIDPROX]	Combining other technologies with LF_CUSTOM is limited.
rdrcfg.lf.hidprox.wiegand_format	(WAUTO, W26, W27, W32, W34, W37, W40, W40_PAR, W44, WABA2, WMAGSTRIPE)	
rdrcfg.lf.hidprox.hid_udf_enable	(0, 1)	Enable processing of HID Prox User Defined Formats - cards with non-standard customer code

Supported cfg items	Supported values	Note
rdrcfg.lf.hidprox.no_wiegand_parity_check	(0, 1)	Do not check parity on wiegand formats that contain parity
rdrcfg.lf.hidprox.no_error_report	(0, 1)	Do not report parity or other errors
rdrcfg.lf.indala.format	(INDALA_UIN1_SHORT, INDALA_UIN2_SHORT, INDALA_UIN1_LONG, INDALA_UIN2_LONG, INDALA_10022, INDALA_10251, INDALA_11037, INDALA_15024, INDALA_17531, INDALA_1771X, INDALA_10324, INDALA_11045, INDALA_DEFWIE, INDALA_DEFABA, INDALA_DEFSEI, INDALA_CUSTOM1, INDALA_CUSTOM2, INDALA_CUSTOM3)	
rdrcfg.lf.indala.wiegand_format	(WAUTO, W26, W27, W32, W34, W37, W40, W40_PAR, W44, WABA2, WMAGSTRIPE)	
rdrcfg.lf.indala.no_wiegand_parity_check	(0, 1)	Do not check parity on wiegand formats that contain parity
rdrcfg.lf.indala.magstripe_format	(MAG_FMT_DEFAULT, MAG_ABA, MAG_IATA, MAG_NTT, MAG_CADMV1, MAG_CADMV3, MAG_SECURITAS, MAG_BINARY)	
rdrcfg.lf.indala.serial_mode	(SER_INDALA_SER, SER_INDALA_UIN, SER_INDALA_UINLONG, SER_INDALA_HEX)	
rdrcfg.lf.indala.custom_format_data	data[256]	256 bytes of reader configuration from NHX file (Indala ProxSmith)
rdrcfg.lf.indala.asp_disabled	(0, 1)	Do not process ASP technology on LF_UIN mode
rdrcfg.lf.indala.asp_plus_disabled	(0, 1)	Do not process ASP+ technology on LF_UIN mode
rdrcfg.lf.indala.ecr_disabled	(0, 1)	Do not process ECR technology on LF_UIN mode
rdrcfg.lf.lf_custom.modulation	(LF_ASK, LF_FSK2, LF_PSK1)	Modulation type
rdrcfg.lf.lf_custom.bitrate	(LF_F16, LF_F32, LF_F50, LF_F64, LF_F8, LF_F20, LF_F40)	Card bitrate

Supported cfg items	Supported values	Note
rdrcfg.lf.lf_custom.start_decoding	(LF_DECODE_NRZ, LF_DECODE_MANCHESTER, LF_DECODE_DIFFBIPHASE)	Decoding of start condition
rdrcfg.lf.lf_custom.data_decoding	(LF_DECODE_NRZ, LF_DECODE_MANCHESTER, LF_DECODE_DIFFBIPHASE)	Decoding of data
rdrcfg.lf.lf_custom.start_cond	u64	64 bit start condition. Must be only 32bit if compatibility with ASKFSK frontend is required.
rdrcfg.lf.lf_custom.start_mask	u64	64 bit start condition mask. Must be only continuous sequence of bits and 32bit if compatibility with ASKFSK frontend is required.
rdrcfg.lf.lf_custom.count_bits	int	Data bits count without start condition. If you do not specify complete bith length, then adjust ignore_bits for complete data reading
rdrcfg.lf.lf_custom.start_decode_invert	(0, 1)	Invert start condition data. Must be set to 1 if LF_DECODE_DIFFBI PHASE is used and compatibility with ASKFSK is required
rdrcfg.lf.lf_custom.data_decode_invert	(0, 1)	Invert data. Must be set to 1 if DIFFBIPHASE decoding is used and compatibility with ASKFSK frontend is required
rdrcfg.lf.lf_custom.input_invert	(0, 1)	Invert data incoming to decodes

Supported cfg items	Supported values	Note
rdrcfg.lf.lf_custom.read_count	int	How many times the data should be read and checked and compared as identical
rdrcfg.lf.lf_custom.padding_bits	int	Padding 0 bits before actual data output
rdrcfg.lf.lf_custom.ignore_bits	int	Bits at the end of the data stream that should be ignored (i.e. void bits), used for proper timing calculation
rdrcfg.lf.lf_custom.card_bootup_time	int	Value to add to card timeout - card bootup overhead in us
rdrcfg.lf.lf_custom.decode	(LF_DECODE_RAW, LF_DECODE_EM4K)	Use particular decode function
rdrcfg.lf.lf_custom.remove_timeout	uint	Timeout for card remove
rdrcfg.lf.lf_custom.card_type	uint	Card type to report
rdrcfg.hf.hf_disabled	(0, 1)	Completely disable processing of HF technologies
rdrcfg.common.no_uin_check	(0, 1)	Do not check if the card UIN is still the same. Required for cards with random UIN
rdrcfg.hf.hf_supported	(HF_DEFAULT, HF_ISO14443A, HF_ISO14443B, HF_ISO15693, HF_INSIDE, HF_FELICA, HF_SIELOX)	
rdrcfg.hf.hf_supported_multiple	[HF_ISO14443A, HF_ISO14443B, HF_ISO15693, HF_INSIDE, HF_FELICA, HF_SIELOX]	Multiple values supported
rdrcfg.hf.hf_uin_revendian	(0, 1)	Reverse endianness of UIN

Supported cfg items	Supported values	Note
rdrcfg.hf.report_read_error	(REP_NONE, REP_AUTO, REP_ERROR, REP_REFUSED)	Change default behaviour of error reporting if the card is not read correctly. By default the card is ignored but the reader may be configured to issue card read error or card refused messages.
rdrcfg.hf.hf_iso_select	(0, 1)	ISO14443A/B-4 ISO select. Required for DESFire/ISO14443A/B-4 commands.
rdrcfg.hf.hf_uin_mode	(HF_UIN_AUTO, HF_UIN_NORMAL, HF_UIN_LEGIC)	Configuration of how UIN should be reported. Some readers may have a different endianness on some technologies.
rdrcfg.hf.read_error_return_uin	(0, 1)	Any error in reading data areas will result in returning the card UIN. Use with caution as this lowers overall security.
rdrcfg.other.piv_mode	(PIV_MODE_AUTO, PIV_MODE_FASCN, PIV_MODE_GUID)	Specify mode of reading of PIV-II cards.
rdrcfg.common.exact_cardtype	(0, 1)	Enable returning exact card type at readers or settings where applicable.
rdrcfg.hf.config_card_disabled	(0, 1)	Disable processing of YSoft Configuration Cards
rdrcfg.hf.config_card_process_time	int	Time in msec for which the config card will be checked for presence before processing, default 8000

Supported cfg items	Supported values	Note
rdrcfg.hf.config_card_allow_time	int	Time in sec after startup in which configuration card is allowed, -1 means always enabled, default 120
rdrcfg.hf.iclass.allowed_data_models_multiple	[ICLASS_PICO15693_ICLASS, ICLASS_PICO15693_SIO, ICLASS_ISO14443A_SIO, ICLASS_ISO14443A_HID_MIFARE]	iClass data models, appropriate HF technologies must be enabled separately
rdrcfg.hf.iclass.allowed_card_types_multiple	[CARD_HID_DESFIRE_SE, CARD_HID_DESFIRE, CARD_HID_MIFARE_SE, CARD_HID_MIFARE, CARD_ICLASS_SE, CARD_ICLASS, CARD_HID_MIFARE_PLUS_SE, CARD_HID_MIFARE_PLUS, CARD_HID_SEOS]	If present allow only these card types, else any. Appropriate HF technologies must be enabled separately.
rdrcfg.hf.iclass.process_pac_bits	(0, 1)	Enable processing of data from HID iClass cards
rdrcfg.hf.iclass.no_pac_bits_return_uin	(0, 1)	Card with not valid data (or incompatible keys) will return UIN
rdrcfg.hf.iclass.not_allowed_cards_return_uin	(0, 1)	cards that are not allowed by allowed_card_types_multiple will return UIN
rdrcfg.hf.iclass.report_card_read_error	(0, 1)	If there is a problem with reading the card, return card read error otherwise ignore the card
rdrcfg.hf.iclass.report_card_refused	(0, 1)	If there is a problem with reading the card, return card refused otherwise ignore the card
rdrcfg.hf.iclass.config_card_allow_time	<1, 65535>	Time in seconds after reader startup in which configuration cards are allowed, -1 means always enabled

Supported cfg items	Supported values	Note
rdrcfg.hf.inside_read.key_type	(KEY_KD, KEY_KC)	Type of used key
rdrcfg.hf.inside_read.key	data[8]	key for protected area
rdrcfg.hf.inside_read.page	uint	Book and page in the book, lower nibble (4 bits) are page, bit 4 (LSB of higher nibble) is book. Works only on paged media. Example: book 1, page 2 = 0x12
rdrcfg.hf.inside_read.address	<0,255>	Read from n-th block (1 block = 8 bytes), 2K page: <0,31>; 16K page: <0,255>
rdrcfg.hf.inside_read.read_count	<1,128>	Read this many blocks (each has 8 bytes)
rdrcfg.hf.mifare_read.mode	(MIFARE_CLASSIC)	
rdrcfg.hf.mifare_read.block	<0,240>	0-63 (Mifare 1k), 0-240 (Mifare 4k)
rdrcfg.hf.mifare_read.mifare_key.key_type	(KEY_MIFARE_A, KEY_MIFARE_B)	
rdrcfg.hf.mifare_read.mifare_key.key_data	data[6]	key data
rdrcfg.hf.mifare_read.card_type	uint	Card type to report instead of the default ISO standard
rdrcfg.hf.mifare_read.mifare_key.sam_key_no	<0,127>	Authentication key number
rdrcfg.hf.mifare_read.mifare_key.key_version	<0,255>	Version of authentication key
rdrcfg.hf.mifare_read.mifare_key.key_div	(DIV_NONE, DIV_UIN)	Key diversification method
rdrcfg.hf.desfire_read.desfire_aid	data[3]	3 byte DESFire Application ID in endiannes reported from card
rdrcfg.hf.desfire_read.desfire_fid	<0,15>	file id
rdrcfg.hf.desfire_read.desfire_foffset	<0,65535>	offset in file
rdrcfg.hf.desfire_read.desfire_flen	<0,230>	0 returns complete file up to 230 bytes

Supported cfg items	Supported values	Note
rdrcfg.hf.desfire_read.desfire_fopts	{DESFIRE_FILE_AUTO, DESFIRE_FILE_PLAIN, DESFIRE_FILE_MACED, DESFIRE_FILE_ENCRYPTED, DESFIRE_FILE_STD, DESFIRE_FILE_VALUE, DESFIRE_FILE_RECORD}	file encryption options for read
rdrcfg.hf.desfire_read.desfire_key_no	<0,15>	
rdrcfg.hf.desfire_read.desfire_key.key_type	(KEY_DES, KEY_2K3DES, KEY_3K3DES, KEY_AES128, KEY_NONE)	
rdrcfg.hf.desfire_read.desfire_key.key_data	data[8,24]	key data
rdrcfg.hf.desfire_read.card_type	uint	Card type to report instead of the default ISO standard
rdrcfg.hf.desfire_read.desfire_key.sam_key_no	<0,127>	Authentication key number
rdrcfg.hf.desfire_read.desfire_key.key_version	<0,255>	Version of authentication key
rdrcfg.hf.desfire_read.desfire_key.key_div	(DIV_NONE, DIV_UIN)	Key diversification method
rdrcfg.sam.sam_mode	(SAM_NONE, SAM_MIFAREAV2, SAM_HIDSEPROC, SAM_AUTO)	Type of SAM
rdrcfg.sam.sam_auth	(SAM_NO_AUTH, SAM_AV2_UNLOCK, SAM_AV2_AUTH_PLAIN)	Authentication mode of SAM
rdrcfg.sam.auth_key_num	<0,127>	Authentication key number
rdrcfg.sam.auth_key.key_type	(KEY_AES128, KEY_AES192)	Authentication key type
rdrcfg.sam.auth_key.key_version	<0,255>	Version of authentication key
rdrcfg.sam.auth_key.key_data	data[16,24]	key data
rdrcfg.hf.iso_read.read	(ISO_CHUID, ISO_CEPAS)	reading of predefined HF technologies
rdrcfg.hf.iso_read.card_type	uint	Card type to report instead of the default ISO standard
rdrcfg.lf.hitag1s_read.address	Hitag 1: <0,63>; Hitag S 256: <0,7>; Hitag S 2048: <0,63>	Read from n-th word (1 word = 32 bits)

Supported cfg items	Supported values	Note
rdrcfg.lf.hitag1s_read.read_count	<1, 64>	Read this many words (each has 32-bits)
rdrcfg.lf.hitag2_read.flags	{LF_HITAG2_NONE, LF_HITAG2_USE_PASSWORD, LF_HITAG2_CHECK_TAG_PASSWORD}	OR combinations possible. Use provided password. Check for provided tag password.
rdrcfg.lf.hitag2_read.password	data[4]	RWD password
rdrcfg.lf.hitag2_read.tag_password	data[3]	TAG password
rdrcfg.lf.hitag2_read.address	<0,7>	Read from n-th word (1 word = 32 bits)
rdrcfg.lf.hitag2_read.read_count	<1,8>	Read this many words (each has 32-bits)
rdrcfg.lf.em4050_read.flags	{LF_EM4050_NONE, LF_EM4050_USE_PASSWORD}	OR combinations possible. Login to card with provided password.
rdrcfg.lf.em4050_read.password	data[4]	Password for read protected area
rdrcfg.lf.em4050_read.address	<1,33>	Read from n-th word (1 word = 32 bits)
rdrcfg.lf.em4050_read.read_count	<1,33>	Read this many words (each has 32-bits)
rdrcfg.lf.q5_read.flags	{LF_Q5_NONE, LF_Q5_USE_PASSWORD}	OR combinations possible. Use provided password.
rdrcfg.lf.q5_read.password	data[4]	Password for tags with read protection
rdrcfg.lf.q5_read.address	Q5, Q5B: <0,7>; T5557: <0,10>; T5577: <0,11>	Read from n-th word (1 word = 32 bits)
rdrcfg.lf.q5_read.read_count	<1,12>	Read this many words (each has 32-bits)
rdrcfg.lf.q5_read.card_detect	(LF_Q5_MANCH_64_DATA, LF_EM4000, LF_KEYPAC)	Process only cards detected as some other simulated LF technology

Default configuration equivalent	
<pre>rdrcfg.lf.lf_supported = LF_AUTO; rdrcfg.hf.hf_supported = HF_DEFAULT;</pre>	
Protocol ID and name	Configuration used
Protocol 1220 - LF UIN + HF UIN	<pre>rdrcfg.lf.lf_supported = LF_AUTO</pre>
Protocol 1223 - LF UIN + HF UIN high pwr	<pre>rdrcfg.lf.lf_supported = LF_AUTO rdrcfg.hf.high_power = 1</pre>
Protocol 1222 - LF UIN + HF UIN rev. endian	<pre>rdrcfg.lf.lf_supported = LF_AUTO rdrcfg.hf.hf_supported = HF_DEFAULT rdrcfg.hf.hf_uin_revendian = 1</pre>
Protocol 1221 - LF UIN only	<pre>rdrcfg.hf.hf_disabled = 1; rdrcfg.lf.lf_supported = LF_AUTO</pre>
Protocol 105 - HF UIN only	<pre>rdrcfg.lf.lf_disabled = 1;</pre>
Protocol 106 - HF UIN only rev. endian	<pre>rdrcfg.hf.hf_uin_revendian = 1 rdrcfg.lf.lf_disabled = 1;</pre>
Protocol 1500 - HF UIN only continuous	<pre>rdrcfg.lf.lf_disabled = 1; rdrcfg.common.continuous = 1;</pre>
Protocol 1518 - ISO14443A UIN only	<pre>rdrcfg.lf.lf_disabled = 1; rdrcfg.hf.hf_supported = HF_ISO14443A;</pre>
Protocol 125 - EM4000 compat.+ HID Prox + HF UIN	<pre>rdrcfg.lf.lf_supported_multiple.extend([LF_EM4000, LF_HIDPROX]);</pre>
Protocol 90 - EM4000 compat.+ HF UIN	<pre>rdrcfg.lf.lf_supported = LF_EM4000;</pre>
Protocol 94 - EM4000 compat. + HF UIN cont.	<pre>rdrcfg.common.continuous = 1; rdrcfg.lf.lf_supported = LF_EM4000;</pre>
Protocol 118 - EM4000 compat.+ ISO14443A UIN	<pre>rdrcfg.lf.lf_supported = LF_EM4000; rdrcfg.hf.hf_supported = HF_ISO14443A;</pre>
Protocol 114 - EM4000 compatible only	<pre>rdrcfg.lf.lf_supported = LF_EM4000; rdrcfg.hf.hf_disabled = 1;</pre>
Protocol 91 - Prox-lite/Casi-Rusco + HF UIN	<pre>rdrcfg.lf.lf_supported = LF_PROXLITE;</pre>

Protocol ID and name	Configuration used
Protocol 92 - Paradox/PosiProx + HF UIN	<code>rdrcfg.lf.lf_supported = LF_PARADOX;</code>
Protocol 89 - Nedap + HF UIN	<code>rdrcfg.lf.lf_supported = LF_NEDAP;</code>
Protocol 87 - Awid + HF UIN	<code>rdrcfg.lf.lf_supported = LF_AWID;</code>
Protocol 86 - Pyramid + HF UIN	<code>rdrcfg.lf.lf_supported = LF_PYRAMID;</code>
Protocol 85 - Deister + HF UIN	<code>rdrcfg.lf.lf_supported = LF_DEISTER;</code>
Protocol 84 - Datasec + HF UIN	<code>rdrcfg.lf.lf_supported = LF_DATASEC;</code>
Protocol 83 - G-Prox + HF UIN	<code>rdrcfg.lf.lf_supported = LF_GPROX;</code>
Protocol 82 - ioProx label + HF UIN	<code>rdrcfg.lf.lf_supported = LF_IOPROX;</code>
Protocol 121 - EM4050 f/64 std. mode + HF UIN	<code>rdrcfg.lf.lf_supported = LF_EM4050R0;</code>
Protocol 96 - PAC/KeyPAC + HF UIN	<code>rdrcfg.lf.lf_supported_multiple.extend([LF_PAC, LF_KEYPAC]);</code>
Protocol 120 - URMET/FDI + HF UIN	<code>rdrcfg.lf.lf_supported = LF_URMET;</code>
Protocol 1601 - Cardax + HF UIN	<code>rdrcfg.lf.lf_supported = LF_CARDAX;</code>
Protocol 1602 - Jablotron + HF UIN	<code>rdrcfg.lf.lf_supported = LF_JABLOTRON;</code>
Protocol 1603 - Keri + HF UIN	<code>rdrcfg.lf.lf_supported = LF_KERI;</code>
Protocol 1604 - IDTECK + HF UIN	<code>rdrcfg.lf.lf_supported = LF_IDTECK;</code>
Protocol 1605 - NexKey + HF UIN	<code>rdrcfg.lf.lf_supported = LF_NEXKEY;</code>
Protocol 1607 - NexKey compat. only	<code>rdrcfg.lf.lf_supported = LF_NEXKEY;</code> <code>rdrcfg.common.cardno_conv = CARDCONV_HEX2DEC;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>
Protocol 1606 - Noralsy + HF UIN	<code>rdrcfg.lf.lf_supported = LF_NORALSY;</code>
Protocol 51 - ISO FDX-B + HF UIN	<code>rdrcfg.lf.lf_supported = LF_ISOFDXB;</code> <code>#rdrcfg.common.cardno_conv = CARDCONV_REVBITSINBYTE;</code>

Protocol ID and name	Configuration used
Protocol 1140 - Indala UIN auto	<code>rdrcfg.lf.lf_supported = LF_INDALA;</code> <code>rdrcfg.lf.indala.format = INDALA_UIN_AUTO;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>
Protocol 1101 - Indala UIN1 short	<code>rdrcfg.lf.lf_supported = LF_INDALA;</code> <code>rdrcfg.lf.indala.format = INDALA_UIN1_SHORT;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>
Protocol 1102 - Indala UIN2 short	<code>rdrcfg.lf.lf_supported = LF_INDALA;</code> <code>rdrcfg.lf.indala.format = INDALA_UIN2_SHORT;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>
Protocol 1103 - Indala UIN1 long	<code>rdrcfg.lf.lf_supported = LF_INDALA;</code> <code>rdrcfg.lf.indala.format = INDALA_UIN1_LONG;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>
Protocol 1104 - Indala UIN2 long	<code>rdrcfg.lf.lf_supported = LF_INDALA;</code> <code>rdrcfg.lf.indala.format = INDALA_UIN2_LONG;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>
Protocol 1105 - Indala W26 - format 10022	<code>rdrcfg.lf.lf_supported = LF_INDALA;</code> <code>rdrcfg.lf.indala.format = INDALA_10022;</code> <code>rdrcfg.lf.indala.wiegand_format = W26;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>
Protocol 1106 - Indala W26 Wallb.- format 10022	<code>rdrcfg.lf.lf_supported = LF_INDALA;</code> <code>rdrcfg.lf.indala.format = INDALA_10022;</code> <code>rdrcfg.lf.indala.wiegand_format = WAUTO;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>
Protocol 1107 - Indala W27 - format 10251	<code>rdrcfg.lf.lf_supported = LF_INDALA;</code> <code>rdrcfg.lf.indala.format = INDALA_10251;</code> <code>rdrcfg.lf.indala.wiegand_format = W27;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>
Protocol 1108 - Indala W27 Wallb.- format 10251	<code>rdrcfg.lf.lf_supported = LF_INDALA;</code> <code>rdrcfg.lf.indala.format = INDALA_10251;</code> <code>rdrcfg.lf.indala.wiegand_format = WAUTO;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>
Protocol 1109 - Indala ABA2 - format 11037	<code>rdrcfg.lf.lf_supported = LF_INDALA;</code> <code>rdrcfg.lf.indala.format = INDALA_11037;</code> <code>rdrcfg.lf.indala.magstripe_format = MAG_ABA;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>
Protocol 1110 - Indala Lite - format 17531	<code>rdrcfg.lf.lf_supported = LF_INDALA;</code> <code>rdrcfg.lf.indala.format = INDALA_17531;</code> <code>rdrcfg.lf.indala.wiegand_format = W26;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>

Protocol ID and name	Configuration used
Protocol 1111 - Indala Lite Wallb.- format 17531	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_17531; rdrcfg.lf.indala.wiegand_format = WAUTO; rdrcfg.hf.hf_disabled = 1;
Protocol 1112 - Indala Kantech - format 15024	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_15024; rdrcfg.lf.indala.wiegand_format = W32; rdrcfg.hf.hf_disabled = 1;
Protocol 1113 - Indala ESMI - format 1771x	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_1771X; rdrcfg.lf.indala.wiegand_format = WAUTO; rdrcfg.hf.hf_disabled = 1;
Protocol 1114 - Indala ECR W26 Wallb. - format 10324	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_10324; rdrcfg.lf.indala.wiegand_format = WAUTO; rdrcfg.hf.hf_disabled = 1;
Protocol 1115 - Indala ECR ABA2 - format 11045	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_11045; rdrcfg.lf.indala.magstripe_format = MAG_ABA; rdrcfg.hf.hf_disabled = 1;
Protocol 1130 - Indala ASP+ default key Wiegand	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_DEFWIE; rdrcfg.lf.indala.wiegand_format = WAUTO; rdrcfg.hf.hf_disabled = 1;
Protocol 1131 - Indala ASP+ default key ABA2	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_DEFABA; rdrcfg.lf.indala.magstripe_format = MAG_ABA; rdrcfg.hf.hf_disabled = 1;
Protocol 1132 - Indala ASP+ default key Serial	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_DEFSER; rdrcfg.hf.hf_disabled = 1;
Protocol 16 - Hitag1/S	rdrcfg.lf.lf_supported = LF_HITAG1S; rdrcfg.hf.hf_disabled = 1;
Protocol 17 - Hitag2	rdrcfg.lf.lf_supported = LF_HITAG2; rdrcfg.hf.hf_disabled = 1;
Protocol 18 - EM4x50 UIN	rdrcfg.lf.lf_supported = LF_EM4550; rdrcfg.hf.hf_disabled = 1;
Protocol 23 - Cotag	rdrcfg.lf.lf_supported = LF_COTAG; rdrcfg.hf.hf_disabled = 1;

Protocol ID and name	Configuration used
Protocol 22 - Tiris	<code>rdrcfg.lf.lf_supported = LF_TIRIS;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>
Protocol 35 - HID Prox + HF UIN	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = WAUTO;</code>
Protocol 124 - HID Prox + HF UIN rev. endian	<code>rdrcfg.hf.hf_uin_revendian = 1;</code> <code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = WAUTO;</code>
Protocol 1035 - HID Prox UDF + HF UIN	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = WAUTO;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 1201 - HID iClass + ISO14443A UIN compat.	<code>rdrcfg.common.cardno_conv = CARDCONV_ISO14443ATRUNCREVENDIAN;</code> <code>rdrcfg.hf.hf_supported_multiple.extend([HF_ICLASS, HF_ISO14443A]);</code> <code>rdrcfg.hf.iclass.process_pac_bits = 1;</code> <code>rdrcfg.hf.iclass.not_allowed_cards_return_uin = 1;</code> <code>rdrcfg.lf.lf_disabled = 1;</code> <code>rdrcfg.hf.iclass.allowed_data_models_multiple.extend([ICLASS_PIC015693_ICLASS]);</code>
Protocol 1202 - HID iClass + HID Prox + ISO14443A UIN compat.	<code>rdrcfg.common.cardno_conv = CARDCONV_ISO14443ATRUNCREVENDIAN;</code> <code>rdrcfg.hf.hf_supported_multiple.extend([HF_ICLASS, HF_ISO14443A]);</code> <code>rdrcfg.hf.iclass.process_pac_bits = 1;</code> <code>rdrcfg.hf.iclass.not_allowed_cards_return_uin = 1;</code> <code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = WAUTO;</code> <code>rdrcfg.hf.iclass.allowed_data_models_multiple.extend([ICLASS_PIC015693_ICLASS]);</code>
Protocol 1203 - HID iClass + ISO14443A UIN	<code>rdrcfg.hf.hf_supported_multiple.extend([HF_ICLASS, HF_ISO14443A]);</code> <code>rdrcfg.hf.iclass.process_pac_bits = 1;</code> <code>rdrcfg.hf.iclass.not_allowed_cards_return_uin = 1;</code> <code>rdrcfg.lf.lf_disabled = 1;</code> <code>rdrcfg.hf.iclass.allowed_data_models_multiple.extend([ICLASS_PIC015693_ICLASS]);</code>
Protocol 1204 - HID iClass + HID Prox + ISO14443A UIN	<code>rdrcfg.hf.hf_supported_multiple.extend([HF_ICLASS, HF_ISO14443A]);</code> <code>rdrcfg.hf.iclass.process_pac_bits = 1;</code> <code>rdrcfg.hf.iclass.not_allowed_cards_return_uin = 1;</code> <code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = WAUTO;</code> <code>rdrcfg.hf.iclass.allowed_data_models_multiple.extend([ICLASS_PIC015693_ICLASS]);</code>

Protocol ID and name	Configuration used
Protocol 1210 - HID iClass + HID SIO	<pre>rdrcfg.hf.hf_supported_multiple.extend([HF_ICLASS, HF_ISO14443A]); rdrcfg.hf.iclass.process_pac_bits = 1; rdrcfg.lf.lf_disabled = 1;</pre>
Protocol 1211 - HID SIO only	<pre>rdrcfg.hf.hf_supported_multiple.extend([HF_ICLASS, HF_ISO14443A]); rdrcfg.hf.iclass.process_pac_bits = 1; rdrcfg.lf.lf_disabled = 1; rdrcfg.hf.iclass.allowed_data_models_multiple.extend([ICLASS_ PIC015693_SIO, ICLASS_ISO14443A_SIO, ICLASS_ISO14443A_HID_MIFARE]); rdrcfg.hf.iclass.allowed_card_types_multiple.extend([CARD_HID _DESFIRE_SE, CARD_HID_MIFARE_SE, CARD_ICLASS_SE, CARD_HID_MIFARE_PLUS_SE, CARD_HID_SEOS]);</pre>
Protocol 1212 - HID iClass + HID SIO + HID Prox	<pre>rdrcfg.hf.hf_supported_multiple.extend([HF_ICLASS, HF_ISO14443A]); rdrcfg.hf.iclass.process_pac_bits = 1; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;</pre>
Protocol 1213 - HID iClass + HID SIO + HID Prox + EM4000	<pre>rdrcfg.hf.hf_supported_multiple.extend([HF_ICLASS, HF_ISO14443A]); rdrcfg.hf.iclass.process_pac_bits = 1; rdrcfg.lf.lf_supported_multiple.extend([LF_EM4000, LF_HIDPROX]); rdrcfg.lf.hidprox.wiegand_format = WAUTO;</pre>
Protocol 37 - HID iClass UIN	<pre>rdrcfg.hf.hf_supported = HF_ICLASS; rdrcfg.lf.lf_disabled = 1;</pre>
Protocol 76 - HID iClass UIN + HID Prox	<pre>rdrcfg.hf.hf_supported = HF_ICLASS; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;</pre>
Protocol 93 - HID Prox only	<pre>rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO; rdrcfg.hf.hf_disabled = 1;</pre>
Protocol 80 - FIPS201 ID from PIV-II	<pre>rdrcfg.hf.hf_iso_select = 1; rdrcfg.hf.iso_read.read = ISO_CHUID; rdrcfg.hf.hf_supported = HF_ISO14443A; rdrcfg.lf.lf_disabled = 1;</pre>
Protocol 1070 - CEPAS CAN	<pre>rdrcfg.hf.hf_iso_select = 1; rdrcfg.hf.hf_supported = HF_ISO14443B; rdrcfg.hf.iso_read.read = ISO_CEPAS; rdrcfg.common.no_uin_check = 1; rdrcfg.lf.lf_disabled = 1;</pre>

Protocol ID and name	Configuration used
Protocol 72 - ASK-RFID UIN	<code>rdrcfg.lf.lf_disabled = 1;</code> <code>rdrcfg.hf.hf_supported = HF_ASKRFID;</code>
<i>Protocol 117 - LF testing + HF testing</i>	<code>rdrcfg.common.testing_mode = 1;</code> <code>rdrcfg.common.exact_cardtype = 1;</code> <code>rdrcfg.sam.sam_mode = SAM_AUTO;</code>

8.25 B-076 - MFX MOBILE READER

Reader description	Universal BLE, HF and LF reader supporting ASK, FSK and PSK modulation on LF and various HF standards. Includes support for Legic Prime, Legic Advant Cards, Legic Connect, HID Prox, Indala.
Reader in production	Yes
Reader technology	Contactless
Reader frequency	2.4GHz / 13.56MHz / 134.2kHz / 132kHz / 125kHz

Notes

YSoft Configuration Cards supported within 2 minutes after startup and if processing of configuration cards is enabled.

Procedure for programming using YSoft Configuration cards: place the card for approximately 10s on the reader, wait for successful programming indication.

Reading of data from LEGIC prime MIM22, LEGIC advant ATC128-MV210, ATC256-MV210, ATC512-MP110, ATC4096-MP310 cards not supported due to Legic chipset limitation.

Reading of Legic Data segments supported.

Reading of Legic KGH segments supported.

Limited support of KGH segments on Advant cards.

Reading of Legic Access segments supported.

Legic Access segments on Prime cards not tested.

Reading of data from Neon files in Legic Connect based Mobile applications supported.

Key delivery via Legic Orbic configuration files (VCP) supported.

Reading of Mifare Classic sectors supported.

Reading of Mifare Classic sectors from 7 byte UIN or random UIN cards supported.

Reading of Mifare DESFire data files supported.

Reading of Mifare Plus sectors not supported at the moment (customization).

Reading of Inside Contactless PicoPass blocks supported.

ISO14443A/B-4 APDU commands supported.

Processing of multiple HF cards at the same time supported.

Only one ISO14443-B card supported in the field.

Usage of Legic Prime cards together with ISO14443-A/B cards with random UIN is limited.

Legic SAM-63 (Launch) and SAM-64 (Delaunch) cards supported.

Procedure for Launch record and Custom Legic Connect keys delete: Configure special protocol 27132, configure back to default UIN protocol.

Tiris cards cannot be placed too close to the reader otherwise they are not read. It is a standard limitation of Tiris cards.

EM4050RO mode not supported in LF UIN as it interferes with EM4050 UIN mode.

Q5 / T5555 compatible cards may require customization depending on reading requirements.

Protocol 117 should not be used for user identification, it is a special protocol intended as an aid for card testing and some limitations may apply.

Reader uses Indala and HID Prox license from HID Corporation.

Indala ASP, ASP+ and ECR cards supported.

Indala option cards not supported.

Indala custom formats possible as a custom card reader configuration.

HID SE Processor supported in SAM slot.

HID iClass Elite key configuration cards are supported only on protocols with PACS data enabled (Protocols 1201, 1202, 1203, 1204, 1210, 1212, 1213).

HID iClass Elite key configuration cards are supported only 2 minutes after reader startup.

Once the SE Processor is programmed with iClass Elite keys then for reverting the reader into factory state the customer iClass Elite to standard keys programming cards must be used. There is no way how to revert it back to use default keys, even Y Soft manufacturing cannot do it.

Finished programming by a HID iClass Elite key configuration card is signaled by a special reader indication.

Supported cfg items	Supported values	Note
rdrcfg.common.cardno_conv	(CARDCONV_APPENDLRC, CARDCONV_HEX2DEC, CARDCONV_HEX2DEC10, CARDCONV_HEX2DECLEFT16, CARDCONV_INTERFLEX, CARDCONV_ISO14443ATRUNCREVENDIAN, CARDCONV_JAVACARDSERIAL, CARDCONV_REVBITSINBYTE, CARDCONV_REVENDIAN, CARDCONV_REVENDIAN2DEC, CARDCONV_REVENDIANHEX2DEC10, CARDCONV_REVENDIANWOHIDPROX)	Builtin card number conversions. Will be overwritten if rdrcfg.cardno.cardnoconv specified
rdrcfg.cardno.cardnoconv		Custom specified card number conversion. CM2CNO("text") or CNOASM(FILE("filename"))).
rdrcfg.common.continuous	(0, 1)	Special setting that enables card requested for login and card remove notification on some products.
rdrcfg.common.testing_mode	(0, 1)	Special testing setting to scan all supported technologies. Only for debug or testing purposes, DO NOT USE IN PRODUCTION ENVIRONMENT, some limitations may apply.
rdrcfg.lf.lf_disabled	(0, 1)	Completely disable processing of LF technologies
rdrcfg.lf.lf_supported	(LF_AUTO, LF_EM4000, LF_PROXLITE, LF_PARADOX, LF_NEDAP, LF_AWID, LF_PYRAMID, LF_DEISTER, LF_DATASEC, LF_GPROX, LF_IOPROX, LF_EM4050RO, LF_PAC, LF_URMET, LF_CARDAX, LF_JABLOTRON, LF_INDALA, LF_NEXKEY, LF_COTAG, LF_TIRIS, LF_KEYPAC, LF_IDTECK, LF_NORALSY, LF_KERI, LF_HITAG1S, LF_HITAG2, LF_EM4550, LF_HIDPROX, LF_CUSTOM)	

Supported cfg items	Supported values	Note
rdrcfg.lf.lf_supported_multiple	[LF_AUTO, LF_EM4000, LF_PROXLITE, LF_PARADOX, LF_NEDAP, LF_AWID, LF_PYRAMID, LF_DEISTER, LF_DATASEC, LF_GPROX, LF_IOPROX, LF_EM4050RO, LF_PAC, LF_URMET, LF_CARDAX, LF_JABLOTRON, LF_INDALA, LF_NEXKEY, LF_COTAG, LF_TIRIS, LF_KEYPAC, LF_IDTECK, LF_NORALSY, LF_KERI, LF_HITAG1S, LF_HITAG2, LF_EM4550, LF_HIDPROX]	Combining other technologies with LF_CUSTOM is limited.
rdrcfg.lf.hidprox.wiegand_format	(WAUTO, W26, W27, W32, W34, W37, W40, W40_PAR, W44, WABA2, WMAGSTRIPE)	
rdrcfg.lf.hidprox.hid_udf_enable	(0, 1)	Enable processing of HID Prox User Defined Formats - cards with non-standard customer code
rdrcfg.lf.hidprox.no_wiegand_parity_check	(0, 1)	Do not check parity on wiegand formats that contain parity
rdrcfg.lf.hidprox.no_error_report	(0, 1)	Do not report parity or other errors
rdrcfg.lf.indala.format	(INDALA_UIN1_SHORT, INDALA_UIN2_SHORT, INDALA_UIN1_LONG, INDALA_UIN2_LONG, INDALA_10022, INDALA_10251, INDALA_11037, INDALA_15024, INDALA_17531, INDALA_1771X, INDALA_10324, INDALA_11045, INDALA_DEFWIE, INDALA_DEFABA, INDALA_DEFSEI, INDALA_CUSTOM1, INDALA_CUSTOM2, INDALA_CUSTOM3)	
rdrcfg.lf.indala.wiegand_format	(WAUTO, W26, W27, W32, W34, W37, W40, W40_PAR, W44, WABA2, WMAGSTRIPE)	
rdrcfg.lf.indala.no_wiegand_parity_check	(0, 1)	Do not check parity on wiegand formats that contain parity
rdrcfg.lf.indala.magstripe_format	(MAG_FMT_DEFAULT, MAG_ABA, MAG_IATA, MAG_NTT, MAG_CADMV1, MAG_CADMV3, MAG_SECURITAS, MAG_BINARY)	
rdrcfg.lf.indala.serial_mode	(SER_INDALA_SER, SER_INDALA_UIN, SER_INDALA_UINLONG, SER_INDALA_HEX)	

Supported cfg items	Supported values	Note
rdrcfg.lf.indala.custom_format_data	data[256]	256 bytes of reader configuration from NHX file (Indala ProxSmith)
rdrcfg.lf.indala.asp_disabled	(0, 1)	Do not process ASP technology on LF_UIN mode
rdrcfg.lf.indala.asp_plus_disabled	(0, 1)	Do not process ASP+ technology on LF_UIN mode
rdrcfg.lf.indala.ecr_disabled	(0, 1)	Do not process ECR technology on LF_UIN mode
rdrcfg.lf.lf_custom.modulation	(LF_ASK, LF_FSK2, LF_PSK1)	Modulation type
rdrcfg.lf.lf_custom.bitrate	(LF_F16, LF_F32, LF_F50, LF_F64, LF_F8, LF_F20, LF_F40)	Card bitrate
rdrcfg.lf.lf_custom.start_decoding	(LF_DECODE_NRZ, LF_DECODE_MANCHESTER, LF_DECODE_DIFFBIPHASE)	Decoding of start condition
rdrcfg.lf.lf_custom.data_decoding	(LF_DECODE_NRZ, LF_DECODE_MANCHESTER, LF_DECODE_DIFFBIPHASE)	Decoding of data
rdrcfg.lf.lf_custom.start_cond	u64	64 bit start condition. Must be only 32bit if compatibility with ASKFSK frontend is required.
rdrcfg.lf.lf_custom.start_mask	u64	64 bit start condition mask. Must be only continuous sequence of bits and 32bit if compatibility with ASKFSK frontend is required.
rdrcfg.lf.lf_custom.count_bits	int	Data bits count without start condition. If you do not specify complete bith length, then adjust ignore_bits for complete data reading

Supported cfg items	Supported values	Note
rdrcfg.lf.lf_custom.start_decode_invert	(0, 1)	Invert start condition data. Must be set to 1 if LF_DECODE_DIFFBI PHASE is used and compatibility with ASKFSK is required
rdrcfg.lf.lf_custom.data_decode_invert	(0, 1)	Invert data. Must be set to 1 if DIFFBI PHASE decoding is used and compatibility with ASKFSK frontend is required
rdrcfg.lf.lf_custom.input_invert	(0, 1)	Invert data incoming to decodes
rdrcfg.lf.lf_custom.read_count	int	How many times the data should be read and checked and compared as identical
rdrcfg.lf.lf_custom.padding_bits	int	Padding 0 bits before actual data output
rdrcfg.lf.lf_custom.ignore_bits	int	Bits at the end of the data stream that should be ignored (i.e. void bits), used for proper timing calculation
rdrcfg.lf.lf_custom.card_bootup_time	int	Value to add to card timeout - card bootup overhead in us
rdrcfg.lf.lf_custom.decode	(LF_DECODE_RAW, LF_DECODE_EM4K)	Use particular decode function
rdrcfg.lf.lf_custom.remove_timeout	uint	Timeout for card remove
rdrcfg.lf.lf_custom.card_type	uint	Card type to report
rdrcfg.hf.hf_disabled	(0, 1)	Completely disable processing of HF technologies

Supported cfg items	Supported values	Note
rdrcfg.common.no_uin_check	(0, 1)	Do not check if the card UIN is still the same. Required for cards with random UIN
rdrcfg.hf.hf_supported	(HF_DEFAULT, HF_LEGIC_PRIME, HF_LEGIC_ADVANT, HF_LEGIC_ADVANT_ISO14443A, HF_LEGIC_ADVANT_ISO1693, HF_ISO14443A, HF_ISO14443B, HF_ISO15693, HF_INSIDE, HF_FELICA)	
rdrcfg.hf.hf_supported_multiple	[HF_DEFAULT, HF_LEGIC_PRIME, HF_LEGIC_ADVANT, HF_LEGIC_ADVANT_ISO14443A, HF_LEGIC_ADVANT_ISO1693, HF_ISO14443A, HF_ISO14443B, HF_ISO15693, HF_INSIDE, HF_FELICA]	Multiple values supported
rdrcfg.hf.hf_uin_revendian	(0, 1)	Reverse endianness of UIN
rdrcfg.hf.report_read_error	(REP_NONE, REP_AUTO, REP_ERROR, REP_REFUSED)	Change default behaviour of error reporting if the card is not read correctly. By default the card is ignored but the reader may be configured to issue card read error or card refused messages.
rdrcfg.hf.hf_iso_select	(0, 1)	ISO14443A/B-4 ISO select. Required for DESFire/ISO14443A/B-4 commands.
rdrcfg.hf.hf_uin_mode	(HF_UIN_AUTO, HF_UIN_NORMAL, HF_UIN_LEGIC)	Configuration of how UIN should be reported. Some readers may have a different endianness on some technologies.
rdrcfg.hf.read_error_return_uin	(0, 1)	Any error in reading data areas will result in returning the card UIN. Use with caution as this lowers overall security.

Supported cfg items	Supported values	Note
rdrcfg.other.piv_mode	(PIV_MODE_AUTO, PIV_MODE_FASCN, PIV_MODE_GUID)	Specify mode of reading of PIV-II cards
rdrcfg.common.exact_cardtype	(0, 1)	Enable returning exact card type at readers or settings where applicable.
rdrcfg.hf.config_card_disabled	(0, 1)	Disable processing of YSoft Configuration Cards
rdrcfg.hf.config_card_process_time	int	Time in msec for which the config card will be checked for presence before processing, default 8000
rdrcfg.hf.config_card_allow_time	int	Time in sec after startup in which configuration card is allowed, -1 means always enabled, default 120
rdrcfg.hf.legic_read.mode	(LEGIC_DATA_SEG, LEGIC_KGH, LEGIC_ACCESS)	
rdrcfg.hf.legic_read.flags	{LEGIC_FLAG_NONE, LEGIC_KGH_SHORT, LEGIC_KGH_OLD, LEGIC_KGH_COMPATV2, LEGIC_ACCESS_STAMP, LEGIC_ACCESS_ID, LEGIC_ACCESS_USER}	OR combinations possible. LEGIC_KGH_OLD - compatibility with old Legic Prime/ Legic Advant readers. LEGIC_KGH_COMPATV2 - compatibility with Legic Advant v2 card readers. LEGIC_KGH_SHORT - returns just stamp and card number, preferred option for KGH.
rdrcfg.hf.legic_read.from_seg	<1,128>	
rdrcfg.hf.legic_read.read_len	<1,199>	
rdrcfg.hf.legic_read.read_offset	<0,65535>	
rdrcfg.hf.legic_read.search_string	data[1,12]	1 to 12 bytes of segment stamp

Supported cfg items	Supported values	Note
rdrcfg.hf.legic_read.card_type	uint	Card type to report instead of the default Legic type
rdrcfg.hf.legic.legic_prime_disabled	(0, 1)	Disable processing of Legic Prime cards
rdrcfg.hf.legic.legic_advant_disabled	(0, 1)	Disable processing of Legic Advant cards
rdrcfg.hf.legic.enabled_cards_multiple	[CARD_LEGICMIM256, CARD_LEGICMIM1024, CARD_LEGICATC1024_MV110, CARD_LEGICATC2048_MP110, CARD_LEGICATC4096_MP311, CARD_LEGICATC4096_MP312, CARD_LEGICAFS4096_JP1X, CARD_LEGICATC1024_MV010, CARD_LEGICATC256_MV410, CARD_LEGICCTC4096_MP410, CARD_LEGICCTC4096_MM410]	List of Legic card types that are allowed
rdrcfg.hf.legic.disabled_cards_multiple	[CARD_LEGICMIM256, CARD_LEGICMIM1024, CARD_LEGICATC1024_MV110, CARD_LEGICATC2048_MP110, CARD_LEGICATC4096_MP311, CARD_LEGICATC4096_MP312, CARD_LEGICAFS4096_JP1X, CARD_LEGICATC1024_MV010, CARD_LEGICATC256_MV410, CARD_LEGICCTC4096_MP410, CARD_LEGICCTC4096_MM410]	List of Legic card types that should be rejected
rdrcfg.hf.legic.clear_launch_records	(0, 1)	Clear all launch records
rdrcfg.hf.legic.clear_user_keys	(0, 1)	Clear all user specified keys. Will not clear factory keys
rdrcfg.hf.legic.launch_media_disabled	(0, 1)	Disable launch/delaunch media processing
rdrcfg.hf.inside_read.key_type	(KEY_KD, KEY_KC)	Type of used key
rdrcfg.hf.inside_read.key	data[8]	key for protected area

Supported cfg items	Supported values	Note
rdrcfg.hf.inside_read.page	uint	Book and page in the book, lower nibble (4 bits) are page, bit 4 (LSB of higher nibble) is book. Works only on paged media. Example: book 1, page 2 = 0x12
rdrcfg.hf.inside_read.address	<0,255>	Read from n-th block (1 block = 8 bytes), 2K page: <0,31>; 16K page: <0,255>
rdrcfg.hf.inside_read.read_count	<1,128>	Read this many blocks (each has 8 bytes)
rdrcfg.hf.mifare_read.mode	(MIFARE_CLASSIC)	
rdrcfg.hf.mifare_read.block	<0,240>	0-63 (Mifare 1k), 0-240 (Mifare 4k)
rdrcfg.hf.mifare_read.mifare_key.key_type	(KEY_MIFARE_A, KEY_MIFARE_B)	
rdrcfg.hf.mifare_read.mifare_key.key_data	data[6]	key data
rdrcfg.hf.mifare_read.card_type	uint	Card type to report instead of the default ISO standard
rdrcfg.hf.mifare_read.mifare_key.sam_key_no	<0,127>	Authentication key number
rdrcfg.hf.mifare_read.mifare_key.key_version	<0,255>	Version of authentication key
rdrcfg.hf.mifare_read.mifare_key.key_div	(DIV_NONE, DIV_UIN)	Key diversification method
rdrcfg.hf.desfire_read.desfire_aid	data[3]	3 byte DESFire Application ID in endiannes reported from card
rdrcfg.hf.desfire_read.desfire_fid	<0,15>	file id
rdrcfg.hf.desfire_read.desfire_foffset	<0,65535>	offset in file
rdrcfg.hf.desfire_read.desfire_flen	<0,230>	0 returns complete file up to 230 bytes

Supported cfg items	Supported values	Note
rdrcfg.hf.desfire_read.desfire_fopts	{DESFIRE_FILE_AUTO, DESFIRE_FILE_PLAIN, DESFIRE_FILE_MACED, DESFIRE_FILE_ENCRYPTED, DESFIRE_FILE_STD, DESFIRE_FILE_VALUE, DESFIRE_FILE_RECORD}	file encryption options for read
rdrcfg.hf.desfire_read.desfire_key_no	<0,15>	
rdrcfg.hf.desfire_read.desfire_key.key_type	(KEY_DES, KEY_2K3DES, KEY_3K3DES, KEY_AES128, KEY_NONE)	
rdrcfg.hf.desfire_read.desfire_key.key_data	data[8,24]	key data
rdrcfg.hf.desfire_read.card_type	uint	Card type to report instead of the default ISO standard
rdrcfg.hf.desfire_read.desfire_key.sam_key_no	<0,127>	Authentication key number
rdrcfg.hf.desfire_read.desfire_key.key_version	<0,255>	Version of authentication key
rdrcfg.hf.desfire_read.desfire_key.key_div	(DIV_NONE, DIV_UIN)	Key diversification method
rdrcfg.sam.sam_mode	(SAM_NONE, SAM_MIFAREAV2, SAM_HIDSEPROC, SAM_AUTO)	Type of SAM
rdrcfg.sam.sam_auth	(SAM_NO_AUTH, SAM_AV2_UNLOCK, SAM_AV2_AUTH_PLAIN)	Authentication mode of SAM
rdrcfg.sam.auth_key_num	<0,127>	Authentication key number
rdrcfg.sam.auth_key.key_type	(KEY_AES128, KEY_AES192)	Authentication key type
rdrcfg.sam.auth_key.key_version	<0,255>	Version of authentication key
rdrcfg.sam.auth_key.key_data	data[16,24]	key data
rdrcfg.hf.iso_read.read	(ISO_CHUID, ISO_CEPAS)	reading of predefined HF technologies
rdrcfg.hf.iso_read.card_type	uint	Card type to report instead of the default ISO standard
rdrcfg.ble.min_rssi	<-128,127>	Minimum RSSI [dBm] for BLE device filtering
rdrcfg.ble.tx_power	(-40, -20, -16, -12, -8, -4, 0, 4)	Transmit power level [dBm]

Supported cfg items	Supported values	Note
rdrcfg.ble.flags	{BLE_NONE , BLE_SEARCH_IOS , BLE_SEARCH_LC , BLE_SEARCH_LC_ALL, BLE_SEARCH_ACTIVE, BLE_SEARCH_FILTER_YSOFT}	
rdrcfg.ble.central_role_enabled	(0, 1)	Enable central role search
rdrcfg.ble.central_scan_duration	<50,2000>	Scan duration for central role
rdrcfg.ble.peripheral_role_enabled	(0, 1)	Enable peripheral role search
rdrcfg.ble.peripheral_scan_duration	<50,2000>	Scan duration for peripheral role
rdrcfg.legic_connect.process_over_ble_central_enabled	(0, 1)	Process Legic Connect over devices found in central role search
rdrcfg.legic_connect.process_over_ble_peripheral_enabled	(0, 1)	Process Legic Connect over devices found in peripheral role search
rdrcfg.legic_connect.process_over_hce_enabled	(0, 1)	Process Legic Connect over devices found in HF search
rdrcfg.legic_connect.project_id	data[4]	Project id of credential neon file and Orbit configuration message (VCP)
rdrcfg.legic_connect.application_id	data[4]	Provide application ID of credential neon file and Orbit configuration message (VCP) or application must support project addressing
rdrcfg.legic_connect.vcp.file_id	data[12]	File ID of Orbit configuration message (VCP)
rdrcfg.legic_connect.vcp.label_id	data[2]	Label ID of key set in Orbit configuration message (VCP)

Supported cfg items	Supported values	Note
rdrcfg.legic_connect.vcp.password	data[0,32]	Password for Orbit configuration message (VCP) (UTF-8)
rdrcfg.legic_connect.vcp.ask_for_password.message	data[0,32]	Message shown on phone (UTF-8)
rdrcfg.legic_connect.vcp.ask_for_password.timeout	uint	Timeout [ms]
rdrcfg.legic_connect.neon.file_id	data[12]	File ID of credential neon file
rdrcfg.legic_connect.neon.read_offset	<0, 65535>	Read from offset in file
rdrcfg.legic_connect.neon.read_len	<1, 200>	Read this many bytes
rdrcfg.legic_connect.neon.auth_key_type	(LEGIC_NEON_KEY_TYPE_RO, LEGIC_NEON_KEY_TYPE_RW)	Key type must be provided despite of defining or delivering keys
rdrcfg.legic_connect.neon.auth_key	data[16]	Keys for credential neon file (if available, else deliver by VCP and left keys empty)
rdrcfg.legic_connect.neon.enc_key	data[16]	Keys for credential neon file (if available, else deliver by VCP and left keys empty)
rdrcfg.legic_connect.neon.use_ysoft_mobile_id	(0, 1)	Use keys for YSoft Connect ID credentials
rdrcfg.lf.hitag1s_read.address	Hitag 1: <0,63>; Hitag S 256: <0,7>; Hitag S 2048: <0,63>	Read from n-th word (1 word = 32 bits)
rdrcfg.lf.hitag1s_read.read_count	<1, 64>	Read this many words (each has 32-bits)
rdrcfg.lf.hitag2_read.flags	{LF_HITAG2_NONE, LF_HITAG2_USE_PASSWORD, LF_HITAG2_CHECK_TAG_PASSWORD}	OR combinations possible. Use provided password. Check for provided tag password.
rdrcfg.lf.hitag2_read.password	data[4]	RWD password
rdrcfg.lf.hitag2_read.tag_password	data[3]	TAG password

Supported cfg items	Supported values	Note
rdrcfg.lf.hitag2_read.address	<0,7>	Read from n-th word (1 word = 32 bits)
rdrcfg.lf.hitag2_read.read_count	<1,8>	Read this many words (each has 32-bits)
rdrcfg.lf.em4050_read.flags	{LF_EM4050_NONE, LF_EM4050_USE_PASSWORD}	OR combinations possible. Login to card with provided password.
rdrcfg.lf.em4050_read.password	data[4]	Password for read protected area
rdrcfg.lf.em4050_read.address	<1,33>	Read from n-th word (1 word = 32 bits)
rdrcfg.lf.em4050_read.read_count	<1,33>	Read this many words (each has 32-bits)
rdrcfg.lf.q5_read.flags	{LF_Q5_NONE, LF_Q5_USE_PASSWORD}	OR combinations possible. Use provided password.
rdrcfg.lf.q5_read.password	data[4]	Password for tags with read protection
rdrcfg.lf.q5_read.address	Q5, Q5B: <0,7>; T5557: <0,10>; T5577: <0,11>	Read from n-th word (1 word = 32 bits)
rdrcfg.lf.q5_read.read_count	<1,12>	Read this many words (each has 32-bits)
rdrcfg.lf.q5_read.card_detect	(LF_Q5_MANCH_64_DATA, LF_EM4000, LF_KEYPAC)	Process only cards detected as some other simulated LF technology
Default configuration equivalent		
rdrcfg.lf.lf_supported = LF_AUTO; rdrcfg.hf.hf_supported = HF_DEFAULT;		

Protocol ID and name	Configuration used
Protocol 1800 - YSoft Connect ID	<pre> rdrcfg.lf.lf_disabled = 1 # ISO14443A needed for NFC/HCE below rdrcfg.hf.hf_supported_multiple.extend([HF_ISO14443A]) # Limit BLE receive signal rdrcfg.ble.min_rssi = -60; # Limit BLE transmit power rdrcfg.ble.tx_power = -40; # BLE Central role enabled rdrcfg.ble.central_role_enabled = 1; rdrcfg.legic_connect.process_over_ble_central_enabled = 1; # Authentication time is longer if the reader operates in peripheral role # and mobile phone in central role (older Android phones, or some new phones # with poor BLE implementation). # When this scenario not needed then commenting the following two lines will # improve reader overall response time. rdrcfg.ble.peripheral_role_enabled = 1; rdrcfg.legic_connect.process_over_ble_peripheral_enabled = 1; # NFC/HCE processing enabled rdrcfg.legic_connect.process_over_hce_enabled = 1; # Support from production environment rdrcfg.legic_connect.project_id = STR([0x04, 0x61, 0xBA, 0xF7]); rdrcfg.legic_connect.application_id = STR([0x04, 0x61, 0xBA, 0xF5]); rdrcfg.legic_connect.neon.file_id = STR([0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00]); rdrcfg.legic_connect.neon.read_offset = 0; rdrcfg.legic_connect.neon.read_len = 8; rdrcfg.legic_connect.neon.auth_key_type = LEGIC_NEON_KEY_TYPE_R0; rdrcfg.legic_connect.neon.use_ysoft_mobile_id = 1; </pre>
Protocol 1220 - LF UIN + HF UIN	<pre> rdrcfg.lf.lf_supported = LF_AUTO </pre>
Protocol 1222 - LF UIN + HF UIN rev. endian	<pre> rdrcfg.lf.lf_supported = LF_AUTO rdrcfg.hf.hf_supported = HF_DEFAULT rdrcfg.hf.hf_uin_revendian = 1 </pre>
Protocol 1322 - LF UIN + Legic UIN	<pre> rdrcfg.lf.lf_supported = LF_AUTO; rdrcfg.hf.hf_supported = HF_DEFAULT; rdrcfg.hf.hf_uin_mode = HF_UIN_LEGIC; </pre>

Protocol ID and name	Configuration used
Protocol 1320 - LF UIN + Legic UIN rev. endian	<code>rdrcfg.lf.lf_supported = LF_AUTO;</code> <code>rdrcfg.hf.hf_supported = HF_DEFAULT;</code> <code>rdrcfg.hf.hf_uin_revendian = 1;</code> <code>rdrcfg.hf.hf_uin_mode = HF_UIN_LEGIC;</code>
Protocol 1221 - LF UIN only	<code>rdrcfg.hf.hf_disabled = 1;</code> <code>rdrcfg.lf.lf_supported = LF_AUTO</code>
Protocol 105 - HF UIN only	<code>rdrcfg.lf.lf_disabled = 1;</code>
Protocol 49 - Legic UIN only	<code>rdrcfg.lf.lf_disabled = 1;</code> <code>rdrcfg.hf.hf_uin_mode = HF_UIN_LEGIC;</code>
Protocol 113 - Legic UIN rev. endian	<code>rdrcfg.lf.lf_disabled = 1;</code> <code>rdrcfg.hf.hf_uin_mode = HF_UIN_LEGIC;</code> <code>rdrcfg.hf.hf_uin_revendian = 1;</code>
Protocol 74 - Legic Advant UIN	<code>rdrcfg.hf.hf_supported = HF_LEGIC_ADVANT;</code> <code>rdrcfg.lf.lf_disabled = 1;</code> <code>rdrcfg.hf.hf_uin_mode = HF_UIN_LEGIC;</code>
Protocol 107 - Legic Prime UIN	<code>rdrcfg.hf.hf_supported = HF_LEGIC_PRIME;</code> <code>rdrcfg.lf.lf_disabled = 1;</code> <code>rdrcfg.hf.hf_uin_mode = HF_UIN_LEGIC;</code>
Protocol 106 - HF UIN only rev. endian	<code>rdrcfg.hf.hf_uin_revendian = 1</code> <code>rdrcfg.lf.lf_disabled = 1;</code>
Protocol 1500 - HF UIN only continuous	<code>rdrcfg.lf.lf_disabled = 1;</code> <code>rdrcfg.common.continuous = 1;</code>
Protocol 1518 - ISO14443A UIN only	<code>rdrcfg.lf.lf_disabled = 1;</code> <code>rdrcfg.hf.hf_supported = HF_ISO14443A;</code>
Protocol 125 - EM4000 compat.+ HID Prox + HF UIN	<code>rdrcfg.lf.lf_supported_multiple.extend([LF_EM4000, LF_HIDPROX]);</code>
Protocol 90 - EM4000 compat.+ HF UIN	<code>rdrcfg.lf.lf_supported = LF_EM4000;</code>
Protocol 94 - EM4000 compat. + HF UIN cont.	<code>rdrcfg.common.continuous = 1;</code> <code>rdrcfg.lf.lf_supported = LF_EM4000;</code>
Protocol 118 - EM4000 compat.+ ISO14443A UIN	<code>rdrcfg.lf.lf_supported = LF_EM4000;</code> <code>rdrcfg.hf.hf_supported = HF_ISO14443A;</code>
Protocol 114 - EM4000 compatible only	<code>rdrcfg.lf.lf_supported = LF_EM4000;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>

Protocol ID and name	Configuration used
Protocol 91 - ProxLite/Casi-Rusco + HF UIN	<code>rdrcfg.lf.lf_supported = LF_PROXLITE;</code>
Protocol 92 - Paradox/PosiProx + HF UIN	<code>rdrcfg.lf.lf_supported = LF_PARADOX;</code>
Protocol 89 - Nedap + HF UIN	<code>rdrcfg.lf.lf_supported = LF_NEDAP;</code>
Protocol 87 - Awid + HF UIN	<code>rdrcfg.lf.lf_supported = LF_AWID;</code>
Protocol 86 - Pyramid + HF UIN	<code>rdrcfg.lf.lf_supported = LF_PYRAMID;</code>
Protocol 85 - Deister + HF UIN	<code>rdrcfg.lf.lf_supported = LF_DEISTER;</code>
Protocol 84 - Datasec + HF UIN	<code>rdrcfg.lf.lf_supported = LF_DATASEC;</code>
Protocol 83 - G-Prox + HF UIN	<code>rdrcfg.lf.lf_supported = LF_GPROX;</code>
Protocol 82 - ioProx label + HF UIN	<code>rdrcfg.lf.lf_supported = LF_IOPROX;</code>
Protocol 121 - EM4050 f/64 std. mode + HF UIN	<code>rdrcfg.lf.lf_supported = LF_EM4050R0;</code>
Protocol 96 - PAC/KeyPAC + HF UIN	<code>rdrcfg.lf.lf_supported_multiple.extend([LF_PAC, LF_KEYPAC]);</code>
Protocol 120 - URMET/FDI + HF UIN	<code>rdrcfg.lf.lf_supported = LF_URMET;</code>
Protocol 1601 - Cardax + HF UIN	<code>rdrcfg.lf.lf_supported = LF_CARDAX;</code>
Protocol 1602 - Jablotron + HF UIN	<code>rdrcfg.lf.lf_supported = LF_JABLOTRON;</code>
Protocol 1603 - Keri + HF UIN	<code>rdrcfg.lf.lf_supported = LF_KERI;</code>
Protocol 1604 - IDTECK + HF UIN	<code>rdrcfg.lf.lf_supported = LF_IDTECK;</code>
Protocol 1605 - NexKey + HF UIN	<code>rdrcfg.lf.lf_supported = LF_NEXKEY;</code>
Protocol 1607 - NexKey compat. only	<code>rdrcfg.lf.lf_supported = LF_NEXKEY;</code> <code>rdrcfg.common.cardno_conv = CARDCONV_HEX2DEC;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>

Protocol ID and name	Configuration used
Protocol 1606 - Noralsy + HF UIN	<code>rdrcfg.lf.lf_supported = LF_NORALSY;</code>
Protocol 51 - ISO FDX-B + HF UIN	<code>rdrcfg.lf.lf_supported = LF_ISOFDXB;</code> <code>#rdrcfg.common.cardno_conv = CARDCONV_REVBITSINBYTE;</code>
Protocol 1140 - Indala UIN auto	<code>rdrcfg.lf.lf_supported = LF_INDALA;</code> <code>rdrcfg.lf.indala.format = INDALA_UIN_AUTO;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>
Protocol 1101 - Indala UIN1 short	<code>rdrcfg.lf.lf_supported = LF_INDALA;</code> <code>rdrcfg.lf.indala.format = INDALA_UIN1_SHORT;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>
Protocol 1102 - Indala UIN2 short	<code>rdrcfg.lf.lf_supported = LF_INDALA;</code> <code>rdrcfg.lf.indala.format = INDALA_UIN2_SHORT;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>
Protocol 1103 - Indala UIN1 long	<code>rdrcfg.lf.lf_supported = LF_INDALA;</code> <code>rdrcfg.lf.indala.format = INDALA_UIN1_LONG;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>
Protocol 1104 - Indala UIN2 long	<code>rdrcfg.lf.lf_supported = LF_INDALA;</code> <code>rdrcfg.lf.indala.format = INDALA_UIN2_LONG;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>
Protocol 1105 - Indala W26 - format 10022	<code>rdrcfg.lf.lf_supported = LF_INDALA;</code> <code>rdrcfg.lf.indala.format = INDALA_10022;</code> <code>rdrcfg.lf.indala.wiegand_format = W26;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>
Protocol 1106 - Indala W26 Wallb.- format 10022	<code>rdrcfg.lf.lf_supported = LF_INDALA;</code> <code>rdrcfg.lf.indala.format = INDALA_10022;</code> <code>rdrcfg.lf.indala.wiegand_format = WAUTO;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>
Protocol 1107 - Indala W27 - format 10251	<code>rdrcfg.lf.lf_supported = LF_INDALA;</code> <code>rdrcfg.lf.indala.format = INDALA_10251;</code> <code>rdrcfg.lf.indala.wiegand_format = W27;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>
Protocol 1108 - Indala W27 Wallb.- format 10251	<code>rdrcfg.lf.lf_supported = LF_INDALA;</code> <code>rdrcfg.lf.indala.format = INDALA_10251;</code> <code>rdrcfg.lf.indala.wiegand_format = WAUTO;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>
Protocol 1109 - Indala ABA2 - format 11037	<code>rdrcfg.lf.lf_supported = LF_INDALA;</code> <code>rdrcfg.lf.indala.format = INDALA_11037;</code> <code>rdrcfg.lf.indala.magstripe_format = MAG_ABA;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>

Protocol ID and name	Configuration used
Protocol 1110 - Indala Lite - format 17531	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_17531; rdrcfg.lf.indala.wiegand_format = W26; rdrcfg.hf.hf_disabled = 1;
Protocol 1111 - Indala Lite Wallb.- format 17531	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_17531; rdrcfg.lf.indala.wiegand_format = WAUTO; rdrcfg.hf.hf_disabled = 1;
Protocol 1112 - Indala Kantech - format 15024	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_15024; rdrcfg.lf.indala.wiegand_format = W32; rdrcfg.hf.hf_disabled = 1;
Protocol 1113 - Indala ESMI - format 1771x	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_1771X; rdrcfg.lf.indala.wiegand_format = WAUTO; rdrcfg.hf.hf_disabled = 1;
Protocol 1114 - Indala ECR W26 Wallb. - format 10324	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_10324; rdrcfg.lf.indala.wiegand_format = WAUTO; rdrcfg.hf.hf_disabled = 1;
Protocol 1115 - Indala ECR ABA2 - format 11045	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_11045; rdrcfg.lf.indala.magstripe_format = MAG_ABA; rdrcfg.hf.hf_disabled = 1;
Protocol 1130 - Indala ASP+ default key Wiegand	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_DEFWIE; rdrcfg.lf.indala.wiegand_format = WAUTO; rdrcfg.hf.hf_disabled = 1;
Protocol 1131 - Indala ASP+ default key ABA2	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_DEFABA; rdrcfg.lf.indala.magstripe_format = MAG_ABA; rdrcfg.hf.hf_disabled = 1;
Protocol 1132 - Indala ASP+ default key Serial	rdrcfg.lf.lf_supported = LF_INDALA; rdrcfg.lf.indala.format = INDALA_DEFSER; rdrcfg.hf.hf_disabled = 1;
Protocol 16 - Hitag1/S	rdrcfg.lf.lf_supported = LF_HITAG1S; rdrcfg.hf.hf_disabled = 1;
Protocol 17 - Hitag2	rdrcfg.lf.lf_supported = LF_HITAG2; rdrcfg.hf.hf_disabled = 1;

Protocol ID and name	Configuration used
Protocol 18 - EM4x50 UIN	<code>rdrcfg.lf.lf_supported = LF_EM4550;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>
Protocol 23 - Cotag	<code>rdrcfg.lf.lf_supported = LF_COTAG;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>
Protocol 22 - Tiris	<code>rdrcfg.lf.lf_supported = LF_TIRIS;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>
Protocol 35 - HID Prox + HF UIN	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = WAUTO;</code>
Protocol 124 - HID Prox + HF UIN rev. endian	<code>rdrcfg.hf.hf_uin_revendian = 1;</code> <code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = WAUTO;</code>
Protocol 1035 - HID Prox UDF + HF UIN	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = WAUTO;</code> <code>rdrcfg.lf.hidprox.hid_udf_enable = 1;</code>
Protocol 37 - HID iClass UIN	<code>rdrcfg.hf.hf_supported = HF_ICLASS;</code> <code>rdrcfg.lf.lf_disabled = 1;</code>
Protocol 76 - HID iClass UIN + HID Prox	<code>rdrcfg.hf.hf_supported = HF_ICLASS;</code> <code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = WAUTO;</code>
Protocol 93 - HID Prox only	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = WAUTO;</code> <code>rdrcfg.hf.hf_disabled = 1;</code>
Protocol 108 - Legic short KGH seg. 1	<code>rdrcfg.hf.legic_read.mode = LEGIC_KGH;</code> <code>rdrcfg.hf.legic_read.flags = LEGIC_KGH_SHORT;</code> <code>rdrcfg.hf.legic_read.from_seg = 1;</code> <code>rdrcfg.lf.lf_disabled = 1;</code>
Protocol 109 - Legic short KGH seg. 2	<code>rdrcfg.hf.legic_read.mode = LEGIC_KGH;</code> <code>rdrcfg.hf.legic_read.flags = LEGIC_KGH_SHORT;</code> <code>rdrcfg.hf.legic_read.from_seg = 2;</code> <code>rdrcfg.lf.lf_disabled = 1;</code>
Protocol 110 - Legic short KGH seg. 3	<code>rdrcfg.hf.legic_read.mode = LEGIC_KGH;</code> <code>rdrcfg.hf.legic_read.flags = LEGIC_KGH_SHORT;</code> <code>rdrcfg.hf.legic_read.from_seg = 3;</code> <code>rdrcfg.lf.lf_disabled = 1;</code>
Protocol 111 - Legic short KGH seg. 4	<code>rdrcfg.hf.legic_read.mode = LEGIC_KGH;</code> <code>rdrcfg.hf.legic_read.flags = LEGIC_KGH_SHORT;</code> <code>rdrcfg.hf.legic_read.from_seg = 4;</code> <code>rdrcfg.lf.lf_disabled = 1;</code>

Protocol ID and name	Configuration used
Protocol 64 - Legic KGH seg. 1	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_COMPATV2; rdrcfg.hf.legic_read.from_seg = 1; rdrcfg.lf.lf_disabled = 1;</pre>
Protocol 65 - Legic KGH seg. 2	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_COMPATV2; rdrcfg.hf.legic_read.from_seg = 2; rdrcfg.lf.lf_disabled = 1;</pre>
Protocol 66 - Legic KGH seg. 3	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_COMPATV2; rdrcfg.hf.legic_read.from_seg = 3; rdrcfg.lf.lf_disabled = 1;</pre>
Protocol 67 - Legic KGH seg. 4	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_COMPATV2; rdrcfg.hf.legic_read.from_seg = 4; rdrcfg.lf.lf_disabled = 1;</pre>
Protocol 29 - Legic KGH seg. 1 compat.	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_OLD; rdrcfg.hf.legic_read.from_seg = 1; rdrcfg.lf.lf_disabled = 1;</pre>
Protocol 30 - Legic KGH seg. 2 compat.	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_OLD; rdrcfg.hf.legic_read.from_seg = 2; rdrcfg.lf.lf_disabled = 1;</pre>
Protocol 31 - Legic KGH seg. 3 compat.	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_OLD; rdrcfg.hf.legic_read.from_seg = 3; rdrcfg.lf.lf_disabled = 1;</pre>
Protocol 32 - Legic KGH seg. 4 compat.	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_OLD; rdrcfg.hf.legic_read.from_seg = 4; rdrcfg.lf.lf_disabled = 1;</pre>
Protocol 75 - Interflex card ID	<pre>rdrcfg.common.cardno_conv = CARDCONV_INTERFLEX; rdrcfg.common.allow_empty_uin = 1; rdrcfg.lf.lf_disabled = 1;</pre>
Protocol 80 - FIPS201 ID from PIV-II	<pre>rdrcfg.hf.hf_iso_select = 1; rdrcfg.hf.iso_read.read = ISO_CHUID; rdrcfg.hf.hf_supported = HF_ISO14443A; rdrcfg.lf.lf_disabled = 1;</pre>

Protocol ID and name	Configuration used
Protocol 1070 - CEPAS CAN	<pre>rdrcfg.hf.hf_iso_select = 1; rdrcfg.hf.hf_supported = HF_ISO14443B; rdrcfg.hf.iso_read.read = ISO_CEPAS; rdrcfg.common.no_uin_check = 1; rdrcfg.lf.lf_disabled = 1;</pre>
<i>Protocol 117 - LF testing + HF testing</i>	<pre>rdrcfg.common.testing_mode = 1; rdrcfg.common.exact_cardtype = 1; rdrcfg.sam.sam_mode = SAM_AUTO;</pre>

8.26 B-084 - MAGNETIC CARDS V2

Reader description	Magnetic cards v2	
Reader in production	Yes	
Reader technology	Magnetic stripe	
Reader frequency	N/A	
Notes		
Protocol 117 should not be used for user identification, it is a special protocol intended as an aid for card testing and some limitations may apply.		
Supported cfg items	Supported values	Note
rdrcfg.common.cardno_conv	(CARDCONV_APPENDLRC, CARDCONV_HEX2DEC, CARDCONV_HEX2DEC10, CARDCONV_HEX2DECLEFT16, CARDCONV_INTERFLEX, CARDCONV_ISO14443ATRUNCREVENDIAN, CARDCONV_JAVACARDSERIAL, CARDCONV_REVBITSINBYTE, CARDCONV_REVENDIAN, CARDCONV_REVENDIAN2DEC, CARDCONV_REVENDIANHEX2DEC10, CARDCONV_REVENDIANWOHIDPROX)	Builtin card number conversions. Will be overwritten if rdrcfg.cardno.cardnoconv specified
rdrcfg.cardno.cardnoconv		Custom specified card number conversion. CM2CNO("text") or CNOASM(FILE("filename"))).
rdrcfg.magstripe.no_error_report	(0, 1)	
rdrcfg.magstripe.magstripe_format_multiple	[MAG_FMT_DEFAULT, MAG_ABA, MAG_IATA, MAG_NTT, MAG_CADMV1, MAG_CADMV3, MAG_SECURITAS, MAG_BINARY]	

Supported cfg items	Supported values	Note
rdrcfg.magstripe.magstripe_format	(MAG_FMT_DEFAULT, MAG_ABA, MAG_IATA, MAG_NTT, MAG_CADMV1, MAG_CADMV3, MAG_SECURITAS, MAG_BINARY)	
rdrcfg.magstripe.include_error_or_empty_data	(0, 1)	
rdrcfg.magstripe.tracks	[TRACK1, TRACK2, TRACK3]	
rdrcfg.magstripe.track_separator	string	
Default configuration equivalent		
rdrcfg.magstripe.tracks.extend([TRACK2]);		
Protocol ID and name	Configuration used	
Protocol 58 - Track 2	rdrcfg.magstripe.tracks.extend([TRACK2]);	
Protocol 57 - Track 1	rdrcfg.magstripe.tracks.extend([TRACK1]);	
Protocol 59 - Track 3	rdrcfg.magstripe.tracks.extend([TRACK3]);	
Protocol 60 - Tracks 1-2	rdrcfg.magstripe.tracks.extend([TRACK1, TRACK2]);	
Protocol 61 - Tracks 2-3	rdrcfg.magstripe.tracks.extend([TRACK2, TRACK3]);	
Protocol 62 - Tracks 1-3	rdrcfg.magstripe.tracks.extend([TRACK1, TRACK3]);	
Protocol 63 - Tracks 1-2-3	rdrcfg.magstripe.tracks.extend([TRACK1, TRACK2, TRACK3]);	
Protocol 116 - Tracks 1-2-3 testing	rdrcfg.magstripe.tracks.extend([TRACK1, TRACK2, TRACK3]); rdrcfg.magstripe.include_error_or_empty_data = 1;	

8.27 B-086 - READER 3 MAGSTRIPE

Reader description	Magnetic stripe swipe card reader
Reader in production	Yes
Reader technology	Magnetic stripe
Reader frequency	N/A
Notes	
Protocol 117 should not be used for user identification, it is a special protocol intended as an aid for card testing and some limitations may apply.	

Supported cfg items	Supported values	Note
rdrcfg.common.cardno_conv	(CARDCONV_APPENDLRC, CARDCONV_HEX2DEC, CARDCONV_HEX2DEC10, CARDCONV_HEX2DECLEFT16, CARDCONV_INTERFLEX, CARDCONV_ISO14443ATRUNCREVENDIAN, CARDCONV_JAVACARDSERIAL, CARDCONV_REVBITSINBYTE, CARDCONV_REVENDIAN, CARDCONV_REVENDIAN2DEC, CARDCONV_REVENDIANHEX2DEC10, CARDCONV_REVENDIANWOHIDPROX)	Builtin card number conversions. Will be overwritten if rdrcfg.cardno.cardnoconv specified
rdrcfg.cardno.cardnoconv		Custom specified card number conversion. CM2CNO("text") or CNOASM(FILE("filename"))).
rdrcfg.magstripe.no_error_report	(0, 1)	
rdrcfg.magstripe.magstripe_format_multiple	[MAG_FMT_DEFAULT, MAG_ABA, MAG_IATA, MAG_NTT, MAG_CADMV1, MAG_CADMV3, MAG_SECURITAS, MAG_BINARY]	
rdrcfg.magstripe.magstripe_format	(MAG_FMT_DEFAULT, MAG_ABA, MAG_IATA, MAG_NTT, MAG_CADMV1, MAG_CADMV3, MAG_SECURITAS, MAG_BINARY)	
rdrcfg.magstripe.include_error_or_empty_data	(0, 1)	
rdrcfg.magstripe.tracks	[TRACK1, TRACK2, TRACK3]	
rdrcfg.magstripe.track_separator	string	
Default configuration equivalent		
rdrcfg.magstripe.tracks.extend([TRACK2]);		
Protocol ID and name	Configuration used	
Protocol 58 - Track 2	rdrcfg.magstripe.tracks.extend([TRACK2]);	
Protocol 57 - Track 1	rdrcfg.magstripe.tracks.extend([TRACK1]);	
Protocol 59 - Track 3	rdrcfg.magstripe.tracks.extend([TRACK3]);	
Protocol 60 - Tracks 1-2	rdrcfg.magstripe.tracks.extend([TRACK1, TRACK2]);	

Protocol ID and name	Configuration used
Protocol 61 - Tracks 2-3	<code>rdrcfg.magstripe.tracks.extend([TRACK2, TRACK3]);</code>
Protocol 62 - Tracks 1-3	<code>rdrcfg.magstripe.tracks.extend([TRACK1, TRACK3]);</code>
Protocol 63 - Tracks 1-2-3	<code>rdrcfg.magstripe.tracks.extend([TRACK1, TRACK2, TRACK3]);</code>
<i>Protocol 116 - Tracks 1-2-3 testing</i>	<code>rdrcfg.magstripe.tracks.extend([TRACK1, TRACK2, TRACK3]);</code> <code>rdrcfg.magstripe.include_error_or_empty_data = 1;</code>

8.28 B-087 - MULTIREADER HF

Reader description	Universal 13.56 MHz supporting various 13.56MHz standards
Reader in production	Yes
Reader technology	Contactless
Reader frequency	13.56MHz

Notes

Reading of Mifare Classic sectors supported.
Reading of Mifare Classic sectors from 7 byte UIN or random UIN cards supported.
Reading of Mifare DESFire data files supported.
Reading of Mifare Plus sectors not supported at the moment (customization).
Reading of Inside Contactless PicoPass blocks supported.
ISO14443A/B-4 APDU commands supported.
Processing of multiple HF cards at the same time supported.
Only one ISO14443-B card supported in the field.

Supported cfg items	Supported values	Note
<code>rdrcfg.common.cardno_conv</code>	(<code>CARDCONV_APPENDLRC</code> , <code>CARDCONV_HEX2DEC</code> , <code>CARDCONV_HEX2DEC10</code> , <code>CARDCONV_HEX2DECLEFT16</code> , <code>CARDCONV_INTERFLEX</code> , <code>CARDCONV_ISO14443ATRUNCREVENDIAN</code> , <code>CARDCONV_JAVACARDSERIAL</code> , <code>CARDCONV_REVBITSINBYTE</code> , <code>CARDCONV_REVENDIAN</code> , <code>CARDCONV_REVENDIAN2DEC</code> , <code>CARDCONV_REVENDIANHEX2DEC10</code> , <code>CARDCONV_REVENDIANWOHIDPROX</code>)	Builtin card number conversions. Will be overwritten if <code>rdrcfg.cardno.cardnoconv</code> specified
<code>rdrcfg.cardno.cardnoconv</code>		Custom specified card number conversion. CM2CNO("text") or CNOASM(FILE("filename")).

Supported cfg items	Supported values	Note
rdrcfg.common.continuous	(0, 1)	Special setting that enables card requested for login and card remove notification on some products.
rdrcfg.common.no_uin_check	(0, 1)	Do not check if the card UIN is still the same. Required for cards with random UIN
rdrcfg.hf.hf_supported	(HF_DEFAULT, HF_ISO14443A, HF_ISO14443B, HF_ISO15693, HF_INSIDE, HF_FELICA, HF_SIELOX)	
rdrcfg.hf.hf_supported_multiple	[HF_ISO14443A, HF_ISO14443B, HF_ISO15693, HF_INSIDE, HF_FELICA, HF_SIELOX]	Multiple values supported
rdrcfg.hf.hf_uin_revendian	(0, 1)	Reverse endianness of UIN
rdrcfg.hf.report_read_error	(REP_NONE, REP_AUTO, REP_ERROR, REP_REFUSED)	Change default behaviour of error reporting if the card is not read correctly. By default the card is ignored but the reader may be configured to issue card read error or card refused messages.
rdrcfg.hf.hf_iso_select	(0, 1)	ISO14443A/B-4 ISO select. Required for DESFire/ISO14443A/B-4 commands.
rdrcfg.hf.hf_uin_mode	(HF_UIN_AUTO, HF_UIN_NORMAL, HF_UIN_LEGIC)	Configuration of how UIN should be reported. Some readers may have a different endianness on some technologies.
rdrcfg.hf.read_error_return_uin	(0, 1)	Any error in reading data areas will result in returning the card UIN. Use with caution as this lowers overall security.

Supported cfg items	Supported values	Note
rdrcfg.other.piv_mode	(PIV_MODE_AUTO, PIV_MODE_FASCN, PIV_MODE_GUID)	Specify mode of reading of PIV-II cards
rdrcfg.hf.inside_read.key_type	(KEY_KD, KEY_KC)	Type of used key
rdrcfg.hf.inside_read.key	data[8]	key for protected area
rdrcfg.hf.inside_read.page	uint	Book and page in the book, lower nibble (4 bits) are page, bit 4 (LSB of higher nibble) is book. Works only on paged media. Example: book 1, page 2 = 0x12
rdrcfg.hf.inside_read.address	<0,255>	Read from n-th block (1 block = 8 bytes), 2K page: <0,31>; 16K page: <0,255>
rdrcfg.hf.inside_read.read_count	<1,128>	Read this many blocks (each has 8 bytes)
rdrcfg.hf.mifare_read.mode	(MIFARE_CLASSIC)	
rdrcfg.hf.mifare_read.block	<0,240>	0-63 (Mifare 1k), 0-240 (Mifare 4k)
rdrcfg.hf.mifare_read.mifare_key.key_type	(KEY_MIFARE_A, KEY_MIFARE_B)	
rdrcfg.hf.mifare_read.mifare_key.key_data	data[6]	key data
rdrcfg.hf.mifare_read.card_type	uint	Card type to report instead of the default ISO standard
rdrcfg.hf.desfire_read.desfire_aid	data[3]	3 byte DESFire Application ID in endiannes reported from card
rdrcfg.hf.desfire_read.desfire_fid	<0,15>	file id
rdrcfg.hf.desfire_read.desfire_foffset	<0,65535>	offset in file
rdrcfg.hf.desfire_read.desfire_flen	<0,230>	0 returns complete file up to 230 bytes

Supported cfg items	Supported values	Note
rdrcfg.hf.desfire_read.desfire_fopts	{DESFIRE_FILE_AUTO, DESFIRE_FILE_PLAIN, DESFIRE_FILE_MACED, DESFIRE_FILE_ENCRYPTED, DESFIRE_FILE_STD, DESFIRE_FILE_VALUE, DESFIRE_FILE_RECORD}	file encryption options for read
rdrcfg.hf.desfire_read.desfire_key_no	<0,15>	
rdrcfg.hf.desfire_read.desfire_key.key_type	(KEY_DES, KEY_2K3DES, KEY_3K3DES, KEY_AES128, KEY_NONE)	
rdrcfg.hf.desfire_read.desfire_key.key_data	data[8,24]	key data
rdrcfg.hf.desfire_read.card_type	uint	Card type to report instead of the default ISO standard

Default configuration equivalent

```
rdrcfg.hf.hf_supported = HF_DEFAULT;
```

Protocol ID and name	Configuration used
Protocol 105 - Card UIN	# Default reader settings
Protocol 118 - ISO14443A UIN only	rdrcfg.hf.hf_supported = HF_ISO14443A;
Protocol 106 - Card UIN rev. endian	rdrcfg.hf.hf_uin_revendian = 1 rdrcfg.lf.lf_disabled = 1;
Protocol 100 - Card UIN continuous	rdrcfg.common.continuous = 1;

8.29 B-088 - READER 3 MF AND LEGIC

Reader description	Reader supporting Legic Advant and Legic Prime HF technologies, ASK and FSK modulation on LF and various HF standards. Includes support for HID Prox.
Reader in production	Yes
Reader technology	Contactless
Reader frequency	13.56MHz / 125kHz

Notes

YSoft Configuration Cards supported within 2 minutes after startup and if processing of configuration cards is enabled.

Procedure for programming using YSoft Configuration cards: place the card for approximately 10s on the reader, wait for successful programming indication.

Reading of Legic Data segments supported.

Reading of Legic KGH segments supported.

Limited support of KGH segments on Advant cards.

Reading of Legic Access segments supported.

Reading of Mifare DESFire data files supported.

Reading of Mifare Classic sectors NOT supported.

Reading of Mifare Plus sectors not supported at the moment (customization).

ISO14443A/B-4 APDU commands supported.

Processing of multiple HF cards at the same time supported.

Legic Access segments on Prime cards not tested.

Only one ISO14443-B card supported in the field.

Usage of Legic Prime cards together with ISO14443-A/B cards with random UIN is limited.

Legic SAM-63 (Launch) and SAM-64 (Delaunch) cards supported.

Procedure for Launch record delete: Configure special protocol 27132, configure back to default UIN protocol.

On LF UIN mode it may take longer to read the card for the first time.

Protocol 117 should not be used for user identification, it is a special protocol intended as an aid for card testing and some limitations may apply.

Supported cfg items	Supported values	Note
rdrcfg.common.cardno_conv	(CARDCONV_APPENDLRC, CARDCONV_HEX2DEC, CARDCONV_HEX2DEC10, CARDCONV_HEX2DECLEFT16, CARDCONV_INTERFLEX, CARDCONV_ISO14443ATRUNCREVENDIAN, CARDCONV_JAVACARDSERIAL, CARDCONV_REVBITSINBYTE, CARDCONV_REVENDIAN, CARDCONV_REVENDIAN2DEC, CARDCONV_REVENDIANHEX2DEC10, CARDCONV_REVENDIANWOHIDPROX)	Builtin card number conversions. Will be overwritten if rdrcfg.cardno.cardnoconv specified
rdrcfg.cardno.cardnoconv		Custom specified card number conversion. CM2CNO("text") or CNOASM(FILE("filename"))).
rdrcfg.common.continuous	(0, 1)	Special setting that enables card requested for login and card remove notification on some products.
rdrcfg.lf.lf_disabled	(0, 1)	Completely disable processing of LF technologies

Supported cfg items	Supported values	Note
rdrcfg.lf.lf_supported	(LF_AUTO, LF_EM4000, LF_PROXLITE, LF_PARADOX, LF_NEDAP, LF_AWID, LF_PYRAMID, LF_DEISTER, LF_DATASEC, LF_GPROX, LF_IOPROX, LF_EM4050RO, LF_PAC, LF_URMET, LF_CARDAX, LF_JABLOTRON, LF_HIDPROX, LF_CUSTOM)	
rdrcfg.lf.lf_supported_multiple	[LF_AUTO, LF_EM4000, LF_PROXLITE, LF_PARADOX, LF_NEDAP, LF_AWID, LF_PYRAMID, LF_DEISTER, LF_DATASEC, LF_GPROX, LF_IOPROX, LF_EM4050RO, LF_PAC, LF_URMET, LF_CARDAX, LF_JABLOTRON, LF_HIDPROX]	Combining other technologies with LF_CUSTOM is limited.
rdrcfg.lf.hidprox.wiegand_format	(WAUTO, W26, W27, W32, W34, W37, W40, W40_PAR, W44, WABA2, WMAGSTRIPE)	
rdrcfg.lf.hidprox.hid_udf_enable	(0, 1)	Enable processing of HID Prox User Defined Formats - cards with non-standard customer code
rdrcfg.lf.hidprox.no_wiegand_parity_check	(0, 1)	Do not check parity on wiegand formats that contain parity
rdrcfg.lf.hidprox.no_error_report	(0, 1)	Do not report parity or other errors
rdrcfg.lf.lf_custom.modulation	(LF_ASK, LF_FSK2)	Modulation type
rdrcfg.lf.lf_custom.bitrate	(LF_F16, LF_F32, LF_F50, LF_F64)	Card bitrate
rdrcfg.lf.lf_custom.start_decoding	(LF_DECODE_NRZ, LF_DECODE_MANCHESTER, LF_DECODE_DIFFBIPHASE)	Decoding of start condition
rdrcfg.lf.lf_custom.data_decoding	(LF_DECODE_NRZ, LF_DECODE_MANCHESTER, LF_DECODE_DIFFBIPHASE)	Decoding of data
rdrcfg.lf.lf_custom.start_cond	u32	32 bit start condition
rdrcfg.lf.lf_custom.start_mask	u32	32 bit Start condition mask. Must be continuous sequence of bits on ASKFSK frontend

Supported cfg items	Supported values	Note
rdrcfg.lf.lf_custom.count_bits	int	Data bits count without start condition. If you do not specify complete bith length, then adjust ignore_bits for complete data reading
rdrcfg.lf.lf_custom.start_decode_invert	(0, 1)	Invert start condition data. Not applicable if LF_DECODE_DIFFBI PHASE is used
rdrcfg.lf.lf_custom.data_decode_invert	(0, 1)	Invert data. Not applicable if LF_DECODE_DIFFBI PHASE is used
rdrcfg.lf.lf_custom.input_invert	(0, 1)	Invert data incoming to decodes
rdrcfg.lf.lf_custom.read_count	int	How many times the data should be read and checked and compared as identical
rdrcfg.lf.lf_custom.padding_bits	int	Padding 0 bits before actual data output
rdrcfg.lf.lf_custom.ignore_bits	int	Bits at the end of the data stream that should be ignored (i.e. void bits), used for proper timing calculation
rdrcfg.lf.lf_custom.card_bootup_time	int	Value to add to card timeout - card bootup overhead in us
rdrcfg.lf.lf_custom.decode	(LF_DECODE_RAW, LF_DECODE_EM4K)	Use particular decode function
rdrcfg.lf.lf_custom.remove_timeout	uint	Timeout for card remove
rdrcfg.lf.lf_custom.card_type	uint	Card type to report

Supported cfg items	Supported values	Note
rdrcfg.hf.hf_disabled	(0, 1)	Completely disable processing of HF technologies
rdrcfg.common.no_uin_check	(0, 1)	Do not check if the card UIN is still the same. Required for cards with random UIN
rdrcfg.hf.hf_supported	(HF_DEFAULT, HF_LEGIC_PRIME, HF_LEGIC_ADVANT, HF_LEGIC_ADVANT_ISO14443A, HF_LEGIC_ADVANT_ISO1693, HF_ISO14443A, HF_ISO14443B, HF_ISO15693, HF_INSIDE, HF_FELICA)	
rdrcfg.hf.hf_supported_multiple	[HF_DEFAULT, HF_LEGIC_PRIME, HF_LEGIC_ADVANT, HF_LEGIC_ADVANT_ISO14443A, HF_LEGIC_ADVANT_ISO1693, HF_ISO14443A, HF_ISO14443B, HF_ISO15693, HF_INSIDE, HF_FELICA]	Multiple values supported
rdrcfg.hf.hf_uin_revendian	(0, 1)	Reverse endianness of UIN
rdrcfg.hf.report_read_error	(REP_NONE, REP_AUTO, REP_ERROR, REP_REFUSED)	Change default behaviour of error reporting if the card is not read correctly. By default the card is ignored but the reader may be configured to issue card read error or card refused messages.
rdrcfg.hf.hf_iso_select	(0, 1)	ISO14443A/B-4 ISO select. Required for DESFire/ISO14443A/B-4 commands.
rdrcfg.hf.hf_uin_mode	(HF_UIN_AUTO, HF_UIN_NORMAL, HF_UIN_LEGIC)	Configuration of how UIN should be reported. Some readers may have a different endianness on some technologies.

Supported cfg items	Supported values	Note
rdrcfg.hf.read_error_return_uin	(0, 1)	Any error in reading data areas will result in returning the card UIN. Use with caution as this lowers overall security.
rdrcfg.other.piv_mode	(PIV_MODE_AUTO, PIV_MODE_FASCN, PIV_MODE_GUID)	Specify mode of reading of PIV-II cards
rdrcfg.hf.config_card_disabled	(0, 1)	Disable processing of YSoft Configuration Cards
rdrcfg.hf.config_card_process_time	int	Time in msec for which the config card will be checked for presence before processing, default 8000
rdrcfg.hf.config_card_allow_time	int	Time in sec after startup in which configuration card is allowed, -1 means always enabled, default 120
rdrcfg.hf.legic_read.mode	(LEGIC_DATA_SEG, LEGIC_KGH, LEGIC_ACCESS)	
rdrcfg.hf.legic_read.flags	{LEGIC_FLAG_NONE, LEGIC_KGH_SHORT, LEGIC_KGH_OLD, LEGIC_KGH_COMPATV2, LEGIC_ACCESS_STAMP, LEGIC_ACCESS_ID, LEGIC_ACCESS_USER}	OR combinations possible. LEGIC_KGH_OLD - compatibility with old Legic Prime/ Legic Advant readers. LEGIC_KGH_COMPA TV2 - compatibility with Legic Advant v2 card readers. LEGIC_KGH_SHORT - returns just stamp and card number, preferred option for KGH.
rdrcfg.hf.legic_read.from_seg	<1,128>	
rdrcfg.hf.legic_read.read_len	<1,199>	
rdrcfg.hf.legic_read.read_offset	<0,65535>	

Supported cfg items	Supported values	Note
rdrcfg.hf.legic_read.search_string	data[1,12]	1 to 12 bytes of segment stamp
rdrcfg.hf.legic_read.card_type	uint	Card type to report instead of the default Legic type
rdrcfg.hf.legic.legic_prime_disabled	(0, 1)	Disable processing of Legic Prime cards
rdrcfg.hf.legic.legic_advant_disabled	(0, 1)	Disable processing of Legic Advant cards
rdrcfg.hf.legic.enabled_cards_multiple	[CARD_LEGICMIM22, CARD_LEGICMIM256, CARD_LEGICMIM1024, CARD_LEGICATC128_MV210, CARD_LEGICATC256_MV210, CARD_LEGICATC512_MP110, CARD_LEGICATC1024_MV110, CARD_LEGICATC2048_MP110, CARD_LEGICATC4096_MP310, CARD_LEGICATC4096_MP311, CARD_LEGICATC4096_MP312, CARD_LEGICAFS4096_JP1X, CARD_LEGICATC1024_MV010, CARD_LEGICATC256_MV410, CARD_LEGICCTC4096_MP410, CARD_LEGICCTC4096_MM410]	List of Legic card types that are allowed
rdrcfg.hf.legic.disabled_cards_multiple	[CARD_LEGICMIM22, CARD_LEGICMIM256, CARD_LEGICMIM1024, CARD_LEGICATC128_MV210, CARD_LEGICATC256_MV210, CARD_LEGICATC512_MP110, CARD_LEGICATC1024_MV110, CARD_LEGICATC2048_MP110, CARD_LEGICATC4096_MP310, CARD_LEGICATC4096_MP311, CARD_LEGICATC4096_MP312, CARD_LEGICAFS4096_JP1X, CARD_LEGICATC1024_MV010, CARD_LEGICATC256_MV410, CARD_LEGICCTC4096_MP410, CARD_LEGICCTC4096_MM410]	List of Legic card types that should be rejected
rdrcfg.hf.legic.clear_launch_records	(0, 1)	Clear all launch records
rdrcfg.hf.legic.launch_media_disabled	(0, 1)	Disable launch/ delaunch media processing

Supported cfg items	Supported values	Note
rdrcfg.hf.desfire_read.desfire_aid	data[3]	3 byte DESFire Application ID in endiannes reported from card
rdrcfg.hf.desfire_read.desfire_fid	<0,15>	file id
rdrcfg.hf.desfire_read.desfire_foffset	<0,65535>	offset in file
rdrcfg.hf.desfire_read.desfire_flen	<0,230>	0 returns complete file up to 230 bytes
rdrcfg.hf.desfire_read.desfire_fopts	{DESFIRE_FILE_AUTO, DESFIRE_FILE_PLAIN, DESFIRE_FILE_MACED, DESFIRE_FILE_ENCRYPTED, DESFIRE_FILE_STD, DESFIRE_FILE_VALUE, DESFIRE_FILE_RECORD}	file encryption options for read
rdrcfg.hf.desfire_read.desfire_key_no	<0,15>	
rdrcfg.hf.desfire_read.desfire_key.key_type	(KEY_DES, KEY_2K3DES, KEY_3K3DES, KEY_AES128, KEY_NONE)	
rdrcfg.hf.desfire_read.desfire_key.key_data	data[8,24]	key data
rdrcfg.hf.desfire_read.card_type	uint	Card type to report instead of the default ISO standard
rdrcfg.hf.iso_read.read	(ISO_CHUID, ISO_CEPAS)	reading of predefined HF technologies
rdrcfg.hf.iso_read.card_type	uint	Card type to report instead of the default ISO standard
Default configuration equivalent		
<pre>rdrcfg.lf.lf_supported = LF_AUTO; rdrcfg.hf.hf_supported = HF_DEFAULT;</pre>		
Protocol ID and name	Configuration used	
Protocol 1322 - LF UIN + Legic UIN	<pre>rdrcfg.lf.lf_supported = LF_AUTO; rdrcfg.hf.hf_supported = HF_DEFAULT; rdrcfg.hf.hf_uin_mode = HF_UIN_LEGIC;</pre>	
Protocol 1320 - LF UIN + Legic UIN rev. endian	<pre>rdrcfg.lf.lf_supported = LF_AUTO; rdrcfg.hf.hf_supported = HF_DEFAULT; rdrcfg.hf.hf_uin_revendian = 1; rdrcfg.hf.hf_uin_mode = HF_UIN_LEGIC;</pre>	

Protocol ID and name	Configuration used
Protocol 49 - Legic UIN only	<code>rdrcfg.lf.lf_disabled = 1;</code> <code>rdrcfg.hf.hf_uin_mode = HF_UIN_LEGIC;</code>
Protocol 1221 - LF UIN only	<code>rdrcfg.hf.hf_disabled = 1;</code> <code>rdrcfg.lf.lf_supported = LF_AUTO</code>
Protocol 113 - Legic UIN rev. endian	<code>rdrcfg.lf.lf_disabled = 1;</code> <code>rdrcfg.hf.hf_uin_mode = HF_UIN_LEGIC;</code> <code>rdrcfg.hf.hf_uin_revendian = 1;</code>
Protocol 74 - Legic Advant UIN	<code>rdrcfg.hf.hf_supported = HF_LEGIC_ADVANT;</code> <code>rdrcfg.lf.lf_disabled = 1;</code> <code>rdrcfg.hf.hf_uin_mode = HF_UIN_LEGIC;</code>
Protocol 107 - Legic Prime UIN	<code>rdrcfg.hf.hf_supported = HF_LEGIC_PRIME;</code> <code>rdrcfg.lf.lf_disabled = 1;</code> <code>rdrcfg.hf.hf_uin_mode = HF_UIN_LEGIC;</code>
Protocol 80 - FIPS201 ID from PIV-II	<code>rdrcfg.hf.hf_iso_select = 1;</code> <code>rdrcfg.hf.iso_read.read = ISO_CHUID;</code> <code>rdrcfg.hf.hf_supported = HF_ISO14443A;</code> <code>rdrcfg.lf.lf_disabled = 1;</code>
Protocol 90 - EM4000 compat.+ Legic UIN	<code>rdrcfg.lf.lf_supported = LF_EM4000;</code>
Protocol 1349 - Legic UIN+HID Prox	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code>
Protocol 1413 - Legic UIN rev. endian+HID Prox	<code>rdrcfg.hf.hf_uin_revendian = 1;</code> <code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = WAUTO;</code>
Protocol 1374 - Legic Advant UIN+HID Prox	<code>rdrcfg.hf.hf_supported = HF_LEGIC_ADVANT;</code> <code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = WAUTO;</code>
Protocol 1407 - Legic Prime UIN+HID Prox	<code>rdrcfg.hf.hf_supported = HF_LEGIC_PRIME;</code> <code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = WAUTO;</code>
Protocol 1380 - FIPS201 ID from PIV-II+HID Prox	<code>rdrcfg.hf.hf_supported = HF_ISO14443A;</code> <code>rdrcfg.hf.hf_iso_select = 1;</code> <code>rdrcfg.hf.iso_read.read = ISO_CHUID;</code> <code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = WAUTO;</code>
Protocol 125 - EM4000 compat.+ HID Prox + Legic UIN	<code>rdrcfg.lf.lf_supported_multiple.extend([LF_EM4000, LF_HIDPROX]);</code>

Protocol ID and name	Configuration used
Protocol 91 - ProxLite/Casi-Rusco + Legic UIN	<code>rdrcfg.lf.lf_supported = LF_PROXLITE;</code>
Protocol 92 - Paradox/PosiProx + Legic UIN	<code>rdrcfg.lf.lf_supported = LF_PARADOX;</code>
Protocol 89 - Nedap + Legic UIN	<code>rdrcfg.lf.lf_supported = LF_NEDAP;</code>
Protocol 87 - Awid + Legic UIN	<code>rdrcfg.lf.lf_supported = LF_AWID;</code>
Protocol 86 - Pyramid + Legic UIN	<code>rdrcfg.lf.lf_supported = LF_PYRAMID;</code>
Protocol 85 - Deister + Legic UIN	<code>rdrcfg.lf.lf_supported = LF_DEISTER;</code>
Protocol 84 - Datasec + Legic UIN	<code>rdrcfg.lf.lf_supported = LF_DATASEC;</code>
Protocol 83 - G-Prox + Legic UIN	<code>rdrcfg.lf.lf_supported = LF_GPROX;</code>
Protocol 82 - ioProx label + Legic UIN	<code>rdrcfg.lf.lf_supported = LF_IOPROX;</code>
Protocol 121 - EM4050 f/64 std. mode + Legic UIN	<code>rdrcfg.lf.lf_supported = LF_EM4050R0;</code>
Protocol 96 - PAC/KeyPAC + Legic UIN	<code>rdrcfg.lf.lf_supported_multiple.extend([LF_PAC, LF_KEYPAC]);</code>
Protocol 120 - URMET/FDI + Legic UIN	<code>rdrcfg.lf.lf_supported = LF_URMET;</code>
Protocol 1601 - Cardax/Gallagher + Legic UIN	<code>rdrcfg.lf.lf_supported = LF_CARDAX;</code>
Protocol 1602 - Jablotron + Legic UIN	<code>rdrcfg.lf.lf_supported = LF_JABLOTRON;</code>
Protocol 35 - HID Prox + Legic UIN	<code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = WAUTO;</code>
Protocol 124 - HID Prox + Legic UIN rev. endian	<code>rdrcfg.hf.hf_uin_revendian = 1;</code> <code>rdrcfg.lf.lf_supported = LF_HIDPROX;</code> <code>rdrcfg.lf.hidprox.wiegand_format = WAUTO;</code>

Protocol ID and name	Configuration used
Protocol 1035 - HID Prox UDF + Legic UIN	rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO; rdrcfg.lf.hidprox.hid_udf_enable = 1;
Protocol 93 - HID Prox only	rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO; rdrcfg.hf.hf_disabled = 1;
Protocol 114 - EM4000 compatible only	rdrcfg.lf.lf_supported = LF_EM4000; rdrcfg.hf.hf_disabled = 1;
Protocol 108 - Legic short KGH seg. 1	rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_SHORT; rdrcfg.hf.legic_read.from_seg = 1; rdrcfg.lf.lf_disabled = 1;
Protocol 109 - Legic short KGH seg. 2	rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_SHORT; rdrcfg.hf.legic_read.from_seg = 2; rdrcfg.lf.lf_disabled = 1;
Protocol 110 - Legic short KGH seg. 3	rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_SHORT; rdrcfg.hf.legic_read.from_seg = 3; rdrcfg.lf.lf_disabled = 1;
Protocol 111 - Legic short KGH seg. 4	rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_SHORT; rdrcfg.hf.legic_read.from_seg = 4; rdrcfg.lf.lf_disabled = 1;
Protocol 64 - Legic KGH seg. 1	rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_COMPATV2; rdrcfg.hf.legic_read.from_seg = 1; rdrcfg.lf.lf_disabled = 1;
Protocol 65 - Legic KGH seg. 2	rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_COMPATV2; rdrcfg.hf.legic_read.from_seg = 2; rdrcfg.lf.lf_disabled = 1;
Protocol 66 - Legic KGH seg. 3	rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_COMPATV2; rdrcfg.hf.legic_read.from_seg = 3; rdrcfg.lf.lf_disabled = 1;
Protocol 67 - Legic KGH seg. 4	rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_COMPATV2; rdrcfg.hf.legic_read.from_seg = 4; rdrcfg.lf.lf_disabled = 1;

Protocol ID and name	Configuration used
Protocol 29 - Legic KGH seg. 1 compat.	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_OLD; rdrcfg.hf.legic_read.from_seg = 1; rdrcfg.lf.lf_disabled = 1;</pre>
Protocol 30 - Legic KGH seg. 2 compat.	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_OLD; rdrcfg.hf.legic_read.from_seg = 2; rdrcfg.lf.lf_disabled = 1;</pre>
Protocol 31 - Legic KGH seg. 3 compat.	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_OLD; rdrcfg.hf.legic_read.from_seg = 3; rdrcfg.lf.lf_disabled = 1;</pre>
Protocol 32 - Legic KGH seg. 4 compat.	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_OLD; rdrcfg.hf.legic_read.from_seg = 4; rdrcfg.lf.lf_disabled = 1;</pre>
Protocol 75 - Interflex card ID	<pre>rdrcfg.common.cardno_conv = CARDCONV_INTERFLEX; rdrcfg.common.allow_empty_uin = 1; rdrcfg.lf.lf_disabled = 1;</pre>
Protocol 1408 - Legic short KGH seg. 1+HID Prox	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_SHORT; rdrcfg.hf.legic_read.from_seg = 1; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;</pre>
Protocol 1409 - Legic short KGH seg. 2+HID Prox	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_SHORT; rdrcfg.hf.legic_read.from_seg = 2; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;</pre>
Protocol 1410 - Legic short KGH seg. 3+HID Prox	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_SHORT; rdrcfg.hf.legic_read.from_seg = 3; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;</pre>
Protocol 1411 - Legic short KGH seg. 4+HID Prox	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_SHORT; rdrcfg.hf.legic_read.from_seg = 4; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;</pre>

Protocol ID and name	Configuration used
Protocol 1364 - Legic KGH seg. 1+HID Prox	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_COMPATV2; rdrcfg.hf.legic_read.from_seg = 1; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;</pre>
Protocol 1365 - Legic KGH seg. 2+HID Prox	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_COMPATV2; rdrcfg.hf.legic_read.from_seg = 2; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;</pre>
Protocol 1366 - Legic KGH seg. 3+HID Prox	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_COMPATV2; rdrcfg.hf.legic_read.from_seg = 3; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;</pre>
Protocol 1367 - Legic KGH seg. 4+HID Prox	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_COMPATV2; rdrcfg.hf.legic_read.from_seg = 4; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;</pre>
Protocol 1329 - Legic KGH seg. 1 compat.+HID Prox	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_OLD; rdrcfg.hf.legic_read.from_seg = 1; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;</pre>
Protocol 1330 - Legic KGH seg. 2 compat.+HID Prox	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_OLD; rdrcfg.hf.legic_read.from_seg = 2; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;</pre>
Protocol 1331 - Legic KGH seg. 3 compat.+HID Prox	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_OLD; rdrcfg.hf.legic_read.from_seg = 3; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;</pre>
Protocol 1332 - Legic KGH seg. 4 compat.+HID Prox	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_OLD; rdrcfg.hf.legic_read.from_seg = 4; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;</pre>
Protocol 1375 - Interflex card ID+HID Prox	<pre>rdrcfg.common.cardno_conv = CARDCONV_INTERFLEX; rdrcfg.common.allow_empty_uin = 1; rdrcfg.lf.lf_supported = LF_HIDPROX; rdrcfg.lf.hidprox.wiegand_format = WAUTO;</pre>

Protocol ID and name	Configuration used
<i>Protocol 117 - ASK/ FSK testing + Legic UIN</i>	<code>rdrcfg.common.testing_mode = 1;</code> <code>rdrcfg.common.exact_cardtype = 1;</code> <code>rdrcfg.sam.sam_mode = SAM_AUTO;</code>

8.30 B-089 - LEGIC ADVANT V3

Reader description	Legic Advant v3 card reader
Reader in production	Yes
Reader technology	Contactless
Reader frequency	13.56MHz

Notes

Reading of Legic Data segments supported.
 Reading of Legic KGH segments supported.
 Limited support of KGH segments on Advant cards.
 Reading of Legic Access segments supported.
 Reading of Mifare DESFire data files supported.
 Reading of Mifare Classic sectors NOT supported.
 Reading of Mifare Plus sectors not supported at the moment (customization).
 ISO14443A/B-4 APDU commands supported.
 Processing of multiple HF cards at the same time supported.
 Legic Access segments on Prime cards not tested.
 Only one ISO14443-B card supported in the field.
 Usage of Legic Prime cards together with ISO14443-A/B cards with random UIN is limited.
 Legic SAM-63 (Launch) and SAM-64 (Delaunch) cards supported.
 Procedure for Launch record delete: Configure special protocol 27132, configure back to default UIN protocol.

Supported cfg items	Supported values	Note
<code>rdrcfg.common.cardno_conv</code>	(<code>CARDCONV_APPENDLRC</code> , <code>CARDCONV_HEX2DEC</code> , <code>CARDCONV_HEX2DEC10</code> , <code>CARDCONV_HEX2DECLEFT16</code> , <code>CARDCONV_INTERFLEX</code> , <code>CARDCONV_ISO14443ATRUNCREVENDIAN</code> , <code>CARDCONV_JAVACARDSERIAL</code> , <code>CARDCONV_REVBITSINBYTE</code> , <code>CARDCONV_REVENDIAN</code> , <code>CARDCONV_REVENDIAN2DEC</code> , <code>CARDCONV_REVENDIANHEX2DEC10</code> , <code>CARDCONV_REVENDIANWOHIDPROX</code>)	Builtin card number conversions. Will be overwritten if <code>rdrcfg.cardno.cardnoconv</code> specified
<code>rdrcfg.cardno.cardnoconv</code>		Custom specified card number conversion. CM2CNO("text") or CNOASM(FILE("filename")).

Supported cfg items	Supported values	Note
rdrcfg.common.continuous	(0, 1)	Special setting that enables card requested for login and card remove notification on some products.
rdrcfg.common.no_uin_check	(0, 1)	Do not check if the card UIN is still the same. Required for cards with random UIN
rdrcfg.hf.hf_supported	(HF_DEFAULT, HF_LEGIC_PRIME, HF_LEGIC_ADVANT, HF_LEGIC_ADVANT_ISO14443A, HF_LEGIC_ADVANT_ISO1693, HF_ISO14443A, HF_ISO14443B, HF_ISO15693, HF_INSIDE, HF_FELICA)	
rdrcfg.hf.hf_supported_multiple	[HF_DEFAULT, HF_LEGIC_PRIME, HF_LEGIC_ADVANT, HF_LEGIC_ADVANT_ISO14443A, HF_LEGIC_ADVANT_ISO1693, HF_ISO14443A, HF_ISO14443B, HF_ISO15693, HF_INSIDE, HF_FELICA]	Multiple values supported
rdrcfg.hf.hf_uin_revendian	(0, 1)	Reverse endianness of UIN
rdrcfg.hf.report_read_error	(REP_NONE, REP_AUTO, REP_ERROR, REP_REFUSED)	Change default behaviour of error reporting if the card is not read correctly. By default the card is ignored but the reader may be configured to issue card read error or card refused messages.
rdrcfg.hf.hf_iso_select	(0, 1)	ISO14443A/B-4 ISO select. Required for DESFire/ISO14443A/B-4 commands.
rdrcfg.hf.hf_uin_mode	(HF_UIN_AUTO, HF_UIN_NORMAL, HF_UIN_LEGIC)	Configuration of how UIN should be reported. Some readers may have a different endianness on some technologies.

Supported cfg items	Supported values	Note
rdrcfg.hf.read_error_return_uin	(0, 1)	Any error in reading data areas will result in returning the card UIN. Use with caution as this lowers overall security.
rdrcfg.other.piv_mode	(PIV_MODE_AUTO, PIV_MODE_FASCN, PIV_MODE_GUID)	Specify mode of reading of PIV-II cards
rdrcfg.hf.legic_read.mode	(LEGIC_DATA_SEG, LEGIC_KGH, LEGIC_ACCESS)	
rdrcfg.hf.legic_read.flags	{LEGIC_FLAG_NONE, LEGIC_KGH_SHORT, LEGIC_KGH_OLD, LEGIC_KGH_COMPATV2, LEGIC_ACCESS_STAMP, LEGIC_ACCESS_ID, LEGIC_ACCESS_USER}	OR combinations possible. LEGIC_KGH_OLD - compatibility with old Legic Prime/ Legic Advant readers. LEGIC_KGH_COMPA TV2 - compatibility with Legic Advant v2 card readers. LEGIC_KGH_SHORT - returns just stamp and card number, preferred option for KGH.
rdrcfg.hf.legic_read.from_seg	<1,128>	
rdrcfg.hf.legic_read.read_len	<1,199>	
rdrcfg.hf.legic_read.read_offset	<0,65535>	
rdrcfg.hf.legic_read.search_string	data[1,12]	1 to 12 bytes of segment stamp
rdrcfg.hf.legic_read.card_type	uint	Card type to report instead of the default Legic type
rdrcfg.hf.legic.legic_prime_disabled	(0, 1)	Disable processing of Legic Prime cards
rdrcfg.hf.legic.legic_advant_disabled	(0, 1)	Disable processing of Legic Advant cards

Supported cfg items	Supported values	Note
rdrcfg.hf.legic.enabled_cards_multiple	[CARD_LEGICMIM22, CARD_LEGICMIM256, CARD_LEGICMIM1024, CARD_LEGICATC128_MV210, CARD_LEGICATC256_MV210, CARD_LEGICATC512_MP110, CARD_LEGICATC1024_MV110, CARD_LEGICATC2048_MP110, CARD_LEGICATC4096_MP310, CARD_LEGICATC4096_MP311, CARD_LEGICATC4096_MP312, CARD_LEGICAFS4096_JP1X, CARD_LEGICATC1024_MV010, CARD_LEGICATC256_MV410, CARD_LEGICCTC4096_MP410, CARD_LEGICCTC4096_MM410]	List of Legic card types that are allowed
rdrcfg.hf.legic.disabled_cards_multiple	[CARD_LEGICMIM22, CARD_LEGICMIM256, CARD_LEGICMIM1024, CARD_LEGICATC128_MV210, CARD_LEGICATC256_MV210, CARD_LEGICATC512_MP110, CARD_LEGICATC1024_MV110, CARD_LEGICATC2048_MP110, CARD_LEGICATC4096_MP310, CARD_LEGICATC4096_MP311, CARD_LEGICATC4096_MP312, CARD_LEGICAFS4096_JP1X, CARD_LEGICATC1024_MV010, CARD_LEGICATC256_MV410, CARD_LEGICCTC4096_MP410, CARD_LEGICCTC4096_MM410]	List of Legic card types that should be rejected
rdrcfg.hf.legic.clear_launch_records	(0, 1)	Clear all launch records
rdrcfg.hf.legic.launch_media_disabled	(0, 1)	Disable launch/delaunch media processing
rdrcfg.hf.desfire_read.desfire_aid	data[3]	3 byte DESFire Application ID in endiannes reported from card
rdrcfg.hf.desfire_read.desfire_fid	<0,15>	file id
rdrcfg.hf.desfire_read.desfire_ffset	<0,65535>	offset in file
rdrcfg.hf.desfire_read.desfire_flen	<0,230>	0 returns complete file up to 230 bytes

Supported cfg items	Supported values	Note
rdrcfg.hf.desfire_read.desfire_fopts	{DESFIRE_FILE_AUTO, DESFIRE_FILE_PLAIN, DESFIRE_FILE_MACED, DESFIRE_FILE_ENCRYPTED, DESFIRE_FILE_STD, DESFIRE_FILE_VALUE, DESFIRE_FILE_RECORD}	file encryption options for read
rdrcfg.hf.desfire_read.desfire_key_no	<0,15>	
rdrcfg.hf.desfire_read.desfire_key.key_type	(KEY_DES, KEY_2K3DES, KEY_3K3DES, KEY_AES128, KEY_NONE)	
rdrcfg.hf.desfire_read.desfire_key.key_data	data[8,24]	key data
rdrcfg.hf.desfire_read.card_type	uint	Card type to report instead of the default ISO standard
rdrcfg.hf.iso_read.read	(ISO_CHUID, ISO_CEPAS)	reading of predefined HF technologies
rdrcfg.hf.iso_read.card_type	uint	Card type to report instead of the default ISO standard

Default configuration equivalent

```
rdrcfg.hf.hf_supported = HF_DEFAULT;
```

Protocol ID and name	Configuration used
Protocol 49 - Legic UIN	# Default reader settings
Protocol 113 - Legic UIN rev. endian	rdrcfg.hf.hf_uin_revendian = 1; rdrcfg.lf.lf_disabled = 1;
Protocol 74 - Legic Advant UIN	rdrcfg.hf.hf_supported = HF_LEGIC_ADVANT; rdrcfg.lf.lf_disabled = 1; rdrcfg.hf.hf_uin_mode = HF_UIN_LEGIC;
Protocol 107 - Legic Prime UIN	rdrcfg.hf.hf_supported = HF_LEGIC_PRIME; rdrcfg.lf.lf_disabled = 1; rdrcfg.hf.hf_uin_mode = HF_UIN_LEGIC;
Protocol 80 - FIPS201 ID from PIV-II	rdrcfg.hf.hf_iso_select = 1; rdrcfg.hf.iso_read.read = ISO_CHUID; rdrcfg.hf.hf_supported = HF_ISO14443A; rdrcfg.lf.lf_disabled = 1;

Protocol ID and name	Configuration used
Protocol 108 - Legic short KGH seg. 1	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_SHORT; rdrcfg.hf.legic_read.from_seg = 1; rdrcfg.lf.lf_disabled = 1;</pre>
Protocol 109 - Legic short KGH seg. 2	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_SHORT; rdrcfg.hf.legic_read.from_seg = 2; rdrcfg.lf.lf_disabled = 1;</pre>
Protocol 110 - Legic short KGH seg. 3	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_SHORT; rdrcfg.hf.legic_read.from_seg = 3; rdrcfg.lf.lf_disabled = 1;</pre>
Protocol 111 - Legic short KGH seg. 4	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_SHORT; rdrcfg.hf.legic_read.from_seg = 4; rdrcfg.lf.lf_disabled = 1;</pre>
Protocol 75 - Interflex card ID	<pre>rdrcfg.common.cardno_conv = CARDCONV_INTERFLEX; rdrcfg.common.allow_empty_uin = 1; rdrcfg.lf.lf_disabled = 1;</pre>
Protocol 64 - Legic KGH seg. 1	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_COMPATV2; rdrcfg.hf.legic_read.from_seg = 1; rdrcfg.lf.lf_disabled = 1;</pre>
Protocol 65 - Legic KGH seg. 2	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_COMPATV2; rdrcfg.hf.legic_read.from_seg = 2; rdrcfg.lf.lf_disabled = 1;</pre>
Protocol 66 - Legic KGH seg. 3	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_COMPATV2; rdrcfg.hf.legic_read.from_seg = 3; rdrcfg.lf.lf_disabled = 1;</pre>
Protocol 67 - Legic KGH seg. 4	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_COMPATV2; rdrcfg.hf.legic_read.from_seg = 4; rdrcfg.lf.lf_disabled = 1;</pre>
Protocol 29 - Legic KGH seg. 1 compat.	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_OLD; rdrcfg.hf.legic_read.from_seg = 1; rdrcfg.lf.lf_disabled = 1;</pre>

Protocol ID and name	Configuration used
Protocol 30 - Legic KGH seg. 2 compat.	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_OLD; rdrcfg.hf.legic_read.from_seg = 2; rdrcfg.lf.lf_disabled = 1;</pre>
Protocol 31 - Legic KGH seg. 3 compat.	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_OLD; rdrcfg.hf.legic_read.from_seg = 3; rdrcfg.lf.lf_disabled = 1;</pre>
Protocol 32 - Legic KGH seg. 4 compat.	<pre>rdrcfg.hf.legic_read.mode = LEGIC_KGH; rdrcfg.hf.legic_read.flags = LEGIC_KGH_OLD; rdrcfg.hf.legic_read.from_seg = 4; rdrcfg.lf.lf_disabled = 1;</pre>

8.31 B-090 - IBUTTON (DALLAS)

Reader description	Reader of iButton/Dallas single wire contact tokens	
Reader in production	Yes	
Reader technology	Contact	
Reader frequency	N/A	
Notes		
<p>Protection up to 6kV electro static discharge.</p> <p>Please note that under some circumstances and combination of clothing the ESD discharge may be much higher which may damage the reader and the device it is connected to.</p>		
Supported cfg items	Supported values	Note
rdrcfg.common.cardno_conv	(CARDCONV_APPENDLRC, CARDCONV_HEX2DEC, CARDCONV_HEX2DEC10, CARDCONV_HEX2DECLEFT16, CARDCONV_INTERFLEX, CARDCONV_ISO14443ATRUNCREVENDIAN, CARDCONV_JAVACARDSERIAL, CARDCONV_REVBITSINBYTE, CARDCONV_REVENDIAN, CARDCONV_REVENDIAN2DEC, CARDCONV_REVENDIANHEX2DEC10, CARDCONV_REVENDIANWOHIDPROX)	Builtin card number conversions. Will be overwritten if rdrcfg.cardno.cardnoconv specified
rdrcfg.cardno.cardnoconv		Custom specified card number conversion. CM2CNO("text") or CNOASM(FILE("filename"))).

Supported cfg items	Supported values	Note
rdrcfg.other.ibutton_family	u8	Custom family of Dallas/iButton chips
Default configuration equivalent		
None		
Protocol ID and name	Configuration used	
Protocol 11 - iButton UIN	# Default reader settings	

8.32 B-091 - SMARTCARD V2

Reader description	Reader supporting ISO7816 contact SmartCards	
Reader in production	Yes	
Reader technology	Contact	
Reader frequency	N/A	
Notes		
<p>Compliance ISO/IEC 7816-1, 2, 3 and 4</p> <p>Asynchronous T=0 and T=1 cards</p> <p>Support for Class A (5V), Class B (3.3V) and Class C (1.8V) cards</p> <p>9.6 Kbps to 115 Kbps speeds</p> <p>Synchronous (memory) cards not supported</p> <p>It is possible to configure the reader to USB Smart Card CCID class - standard USB Smart Card reader.</p> <p>Driver in most modern OS is installed automatically.</p> <p>Reading of card ID requires a customization depending on card type, card operating system and middleware requirements.</p> <p>SmartCards must be always tested in Y Soft based on required functionality.</p>		
Supported cfg items	Supported values	Note
rdrcfg.common.cardno_conv	(CARDCONV_APPENDLRC, CARDCONV_HEX2DEC, CARDCONV_HEX2DEC10, CARDCONV_HEX2DECLEFT16, CARDCONV_INTERFLEX, CARDCONV_ISO14443ATRUNCREVENDIAN, CARDCONV_JAVACARDSERIAL, CARDCONV_REVBITSINBYTE, CARDCONV_REVENDIAN, CARDCONV_REVENDIAN2DEC, CARDCONV_REVENDIANHEX2DEC10, CARDCONV_REVENDIANWOHIDPROX)	Builtin card number conversions. Will be overwritten if rdrcfg.cardno.cardnoconv specified

Supported cfg items	Supported values	Note
rdrcfg.cardno.cardnoconv		Custom specified card number conversion. CM2CNO("text") or CNOASM(FILE("filename")).
Default configuration equivalent		
None		
Protocol ID and name	Configuration used	
Protocol 78 - FIPS201 ID from PIV-II	rdrcfg.smartcard.mode = SMARTCARD_PIV;	
Protocol 1078 - FIPS201 ID from PIV-II cont.	rdrcfg.smartcard.mode = SMARTCARD_PIV; rdrcfg.common.continuous = 1;	
Protocol 1701 - Java Card IC Serial No.	rdrcfg.smartcard.mode = SMARTCARD_ATR; rdrcfg.common.cardno_conv = CARDCONV_JAVACARDSERIAL;	
<i>Protocol 95 - ATR testing</i>	rdrcfg.smartcard.mode = SMARTCARD_ATR;	

8.33 B-093 - READER 3 SMART CARD

Reader description	Reader supporting ISO7816 contact SmartCards
Reader in production	Yes
Reader technology	Contact
Reader frequency	N/A
Notes	
<p>Compliance ISO/IEC 7816-1, 2, 3 and 4</p> <p>Asynchronous T=0 and T=1 cards</p> <p>Support for Class A (5V), Class B (3.3V) and Class C (1.8V) cards</p> <p>9.6 Kbps to 115 Kbps speeds</p> <p>Synchronous (memory) cards not supported</p> <p>It is possible to configure the reader to USB Smart Card CCID class - standard USB Smart Card reader.</p> <p>Driver in most modern OS is installed automatically.</p> <p>Reading of card ID requires a customization depending on card type, card operating system and middleware requirements.</p> <p>SmartCards must be always tested in Y Soft based on required functionality.</p>	

Supported cfg items	Supported values	Note
rdrcfg.common.cardno_conv	(CARDCONV_APPENDLRC, CARDCONV_HEX2DEC, CARDCONV_HEX2DEC10, CARDCONV_HEX2DECLEFT16, CARDCONV_INTERFLEX, CARDCONV_ISO14443ATRUNCREVENDIAN, CARDCONV_JAVACARDSERIAL, CARDCONV_REVBITSINBYTE, CARDCONV_REVENDIAN, CARDCONV_REVENDIAN2DEC, CARDCONV_REVENDIANHEX2DEC10, CARDCONV_REVENDIANWOHIDPROX)	Builtin card number conversions. Will be overwritten if rdrcfg.cardno.cardnoconv specified
rdrcfg.cardno.cardnoconv		Custom specified card number conversion. CM2CNO("text") or CNOASM(FILE("filename"))).

Default configuration equivalent

None

Protocol ID and name	Configuration used
Protocol 78 - FIPS201 ID from PIV-II	rdrcfg.smartcard.mode = SMARTCARD_PIV;
Protocol 1078 - FIPS201 ID from PIV-II cont.	rdrcfg.smartcard.mode = SMARTCARD_PIV; rdrcfg.common.continuous = 1;
Protocol 1701 - Java Card IC Serial No.	rdrcfg.smartcard.mode = SMARTCARD_ATR; rdrcfg.common.cardno_conv = CARDCONV_JAVACARDSERIAL;
<i>Protocol 95 - ATR testing</i>	rdrcfg.smartcard.mode = SMARTCARD_ATR;

8.34 B-095 - READER 3 IBUTTON

Reader description	Reader of iButton/Dallas single wire contact tokens
Reader in production	Yes
Reader technology	Contact
Reader frequency	N/A
Notes	
Protection up to 6kV electro static discharge. Please note that under some circumstances and combination of clothing the ESD discharge may be much higher which may damage the reader and the device it is connected to.	

Supported cfg items	Supported values	Note
rdrcfg.common.cardno_conv	(CARDCONV_APPENDLRC, CARDCONV_HEX2DEC, CARDCONV_HEX2DEC10, CARDCONV_HEX2DECLEFT16, CARDCONV_INTERFLEX, CARDCONV_ISO14443ATRUNCREVENDIAN, CARDCONV_JAVACARDSERIAL, CARDCONV_REVBITSINBYTE, CARDCONV_REVENDIAN, CARDCONV_REVENDIAN2DEC, CARDCONV_REVENDIANHEX2DEC10, CARDCONV_REVENDIANWOHIDPROX)	Builtin card number conversions. Will be overwritten if rdrcfg.cardno.cardnoconv specified
rdrcfg.cardno.cardnoconv		Custom specified card number conversion. CM2CNO("text") or CNOASM(FILE("filename"))).
rdrcfg.other.ibutton_family	u8	Custom family of Dallas/iButton chips
Default configuration equivalent		
None		
Protocol ID and name	Configuration used	
Protocol 11 - iButton UIN	# Default reader settings	