

## **uFR serial - Communication protocol for uFR series devices**

uFR Series devices can establish communication over FTDI's Virtual COM port, so devices are seen as standard COM port hardware.

Communication parameters are :

*Readers with FTDI serial interface:*

### **uFR Classic and uFR Advance readers with USB connection:**

Serial communication: 1 Mbps, 8-N-1, Flow control: None;

The RTS pin is used to reset the device. When the RTS is set, the device is in a reset state. When the RTS is clear, the device is in normal state.

### **uFR BaseHD readers with “uFR support” firmware installed (ex. XR and uFR XRc readers):**

Serial communication (using VCOM FTDI driver): 250 kbps, 8-N-1, Flow control: None;

*Readers without FTDI serial interface:*

**RS485 (connection without USB/RS-485 converter):** variable baudrate can be set through software tool. Current baud rate must be known when changing baudrate. Default baudrate is 250 kbps.

### **uFR Classic Nano RS232 and Card Size RS232:**

UART / TTL: 115200 bps, 8-N-1, Flow control: None.

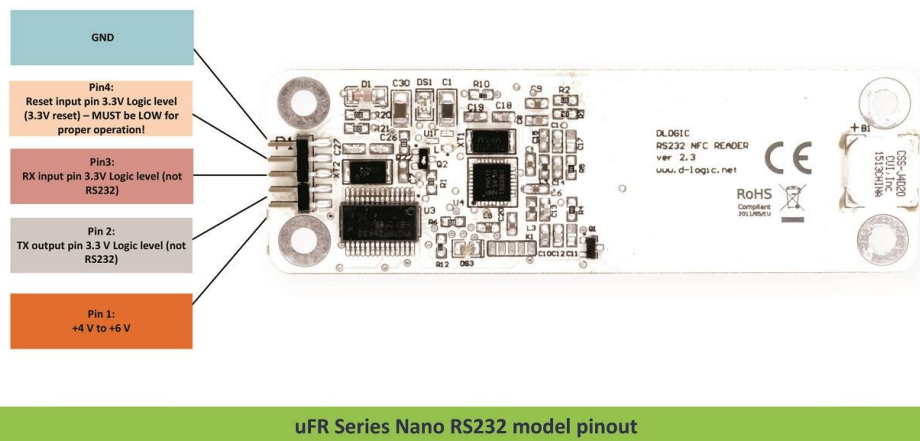
115200 bps is the default baudrate. Variable baudrate can be set through software tool.

Pin number 4 on the connector is used to reset the device. If voltage on this pin is high (3.3 V) then the device is in reset state. If voltage is low (0 V) then the device is in normal working state.

If the device is connected to our RS232 to TTL converter, then the voltage level on pin 4 controls over RTS. When the RTS is clear, the device is in a reset state. When the RTS is set, the device is in normal state.

**During firmware update, the RTS pin must be connected to the pin 4 on the device.**

**Pinout for UART / TTL model is presented below:**



For communication purposes between reader devices and host PC, D-Logic's proprietary protocol called “uFR serial” is created.

All communication is initiated by the host (PC or other platform) to which the device is connected.

Maximum data transferred by single command or received by one device response, from firmware version 3.9.44 is 256 bytes, and before is 192 bytes.

Generally, there are two types of packets:

**CMD** – command sent by host to device

**ANS** – answer sent from device to host

CMD can be short or long set. CMD short set is always 7 byte long while CMD long set – called CMD\_EXT can have variable length.

Answer have following types:

**ACK** – Acknowledgment, everything is OK, device is waiting for next CMD or CMD EXT

**ERR** – Error occurred, error byte defines ERR\_TYPE

**RSP** – Response from device on CMD or CMD\_EXT

Communication constants bytes defines type of packet, which can be seen in first three bytes of each packet.

First byte of each packet is HEADER byte. Second byte is always CMD\_CODE. Third byte is TRAILER byte.

Table1. Communication constants			
CMD_HEADER	0x55	CMD_TRAILER	0xAA
ACK_HEADER	0xAC	ACK_TRAILER	0xCA
RESPONSE_HEADER	0xDE	RESPONSE_TRAILER	0xED
ERR_HEADER	0xEC	ERR_TRAILER	0xCE

## CHECKSUM

All checksums in this document are calculated in the same manner: row of bytes is used for checksum calculation, each byte is XOR-ed with next one until the end of row. Final value is incremented with 0x07.

For example, CMD packet has 7 bytes, where 7<sup>th</sup> byte is checksum of previous 6 bytes:

$$CHECKSUM = (Byte1 \text{ XOR } Byte2 \text{ XOR } Byte3 \text{ XOR } Byte4 \text{ XOR } Byte5 \text{ XOR } Byte6) + 0x07$$

## CMD codes

Each command has its corresponding value - look at [COMMANDS OVERVIEW](#).

## Error codes

If error occurs, device will answer with ERR packet. Each Error has its corresponding value which can be found in table in [Appendix: ERROR CODES](#).

## CMD packet

CMD packet can be short – 7 byte long or EXT-ended with variable length. In case of EXT CMD packet, fourth byte of CMD packet is greater than 0, containing integer value – length of CMD\_EXT packet. When issuing CMD\_EXT, always main CMD 7-byte long packet goes first. If everything as expected, device will answer with ACK packet, waiting for CMD\_EXT packet. On error, device will answer with ERR packet. CMD\_EXT consists of various different parameters, depending on command type, so CMD\_EXT does not have fixed length and order of parameters.

CMD packet has following structure:

Mandatory 7 byte CMD packet structure						
Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CMD_HEADER	CMD_CODE	CMD_TRAILER	CMD_EXT_Length	CMD_Par0	CMD_Par1	CHECKSUM

**Byte 1:** CMD\_HEADER as defined in Table1.Communication constants, 0x55

**Byte 2:** CMD\_CODE as defined in Table2. CMD\_CODE values

**Byte 3:** CMD\_TRAILER as defined in Table1. Communication constants, 0xAA

**Byte 4:** CMD\_EXT\_Length: If 0 than the “CMD EXT” is not used); ELSE value is length of whole CMD\_EXT packet

**Byte 5:** CMD\_Par0: command parameter0, takes different values depending on command

**Byte 6:** CMD\_Par1: command parameter1, takes different values depending on command

**Byte 7:** CHECKSUM – Checksum of Bytes 1 to 6 as explained above

CMD\_EXT packet has following structure:

CMD_EXT packet structure			
Byte 1	..	Byte N	Byte N+1
Parameter bytes 1 to N			CMD_EXT_CHECKSUM

Parameter bytes 1 to N – different parameters, values depends on type of command

**CMD\_EXT\_CHECKSUM** - Checksum of bytes 1 to N

**CMD\_EXT\_Length** is number of all bytes including CMD\_EXT\_CHECKSUM; e.g. length is N+1

### *ANSWER packet types*

The device can answer with following packet types:

#### **ACK – Acknowledgment packet**

If command and CMD packet are properly configured (structure and checksum) and additional CMD\_EXT packet needs to be sent, device will answer with ACK packet.

#### **ERR – Error packet**

If error occurred, device will answer with ERR packet. Some commands can return ERR\_EXT set. In that case ERR\_EXT packet comes immediately after ERR packet.

#### **RSP – Response packet**

If properly configured CMD or CMD\_EXT packet is sent, device will answer with RSP or RSP\_EXT packet, which depends on command issued. For examples, if CMD needs answer which is short enough for RSP packet, there will be no RSP\_EXT packet. Otherwise, if CMD or CMD\_EXT needs answer with more bytes, RSP\_EXT will come immediately after RSP packet. Common situation is when reading data with LinearRead command, where device will answer with row of card data bytes.

#### *ACK – Acknowledgment packet*

ACK packet has following structure:

ACK packet structure						
Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
ACK_HEADER	CMD_CODE	CMD_TRAILER	Irrelevant, not used in ACK packet			CHECKSUM

**Byte 1:** ACK\_HEADER as defined in Table1.Communication constants, 0x55

**Byte 2:** CMD\_CODE as defined in Table2. CMD\_CODE values. Device ACK-knowledge that previous command is properly sent

**Byte 3:** ACK\_HEADER as defined in Table1.Communication constants, 0x55

**Byte 4, Byte 5, Byte 6:** Not used in ACK packet, values are 0x00

**Byte 7:** CHECKSUM – Checksum of Bytes 1 to 6 as explained above

### ERR – error packet

ERR packet has following structure:

Mandatory 7 byte ERR						
Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
ERR_HEADER	ERROR_CODE	ERR_TRAILER	ERR_EXT length	Err_Val0	Err_Val1	CHECKSUM

**Byte 1:** ERR\_HEADER as defined in Table1.Communication constants, 0xEC

**Byte 2:** ERR\_CODE as defined in Table3. ERROR CODES.

**Byte 3:** ERR\_TRAILER as defined in Table1.Communication constants, 0xCE

**Byte 4:** If ERR\_EXT exists, this byte contains length of ERR\_EXT packet (including ERR\_EXT checksum)

**Byte 5:** Possible additional info on error can be defined in ERR\_Val0

**Byte 6:** Possible additional info on error can be defined in ERR\_Val1

**Byte 7:** CHECKSUM – Checksum of Bytes 1 to 6 as explained above

ERR\_EXT and has following structure:

ERR_EXT packet structure			
Byte 1	..	Byte N	Byte N+1
Error bytes 1 to N			ERR_EXT_CHECKSUM

**Byte 1:** First Byte of ERR\_EXT

...

**Byte N:** N-nth Byte of ERR\_EXT

**Byte N+1:** ERR\_EXT\_CHECKSUM, checksum of Bytes 1 to N, calculated as explained earlier.

## RSP – response packet

RSP packet has following structure:

Mandatory 7 byte RSP						
Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
RSP_HEADER	CMD_CODE	RSP_TRAILER	RSP_EXT length	RSP_Val0	RSP_Val1	CHECKSUM

**Byte 1:** RSP\_HEADER as defined in Table1.Communication constants, 0xED

**Byte 2:** CMD\_CODE as defined in Table2. CMD\_CODE values

**Byte 3:** ERR\_TRAILER as defined in Table1.Communication constants, 0xDE

**Byte 4:** If RSP\_EXT exists, this byte contains length of RSP\_EXT packet (including RSP\_EXT checksum)

**Byte 5:** Possible additional info on RESPONSE can be defined in RSP\_Val0

**Byte 6:** Possible additional info on RESPONSE can be defined in RSP\_Val1

**Byte 7:** CHECKSUM – Checksum of Bytes 1 to 6 as explained above

RSP_EXT packet structure			
Byte 1	..	Byte N	Byte N+1
RSP bytes 1 to N			RSP_EXT_CHECKSUM

**Byte 1:** First Byte of RSP\_EXT

...

**Byte N:** N-nth Byte of RSP\_EXT

**Byte N+1:** RSP\_EXT\_CHECKSUM, checksum of Bytes 1 to N, calculated as explained earlier.

## COMMANDS OVERVIEW

Commands are divided into several groups, based on purpose.

## Device related commands

### *General purpose device related commands*

GET_READER_TYPE	0x10
GET_READER_SERIAL	0x11
GET_SERIAL_NUMBER	0x40
GET_HARDWARE_VERSION	0x2A
GET_FIRMWARE_VERSION	0x29
GET_BUILD_NUMBER	0x2B
READER_KEY_WRITE	0x12
USER_DATA_READ	0x1B
USER_DATA_WRITE	0x1C
READER_KEYS_LOCK	0x27
READER_KEYS_UNLOCK	0x28

READER_PASSWORD_WRITE	0x33
SELF_RESET	0x30
SET_SPEED_PERMANENTLY	0x4B
GET_SPEED_PARAMETERS	0x4C
SET_UART_SPEED	0x70
RED_LIGHT_CONTROL	0x71
USER_INTERFACE_SIGNAL	0x26
SET_RF_ANALOG_SETTINGS	0x7D
GET_RF_ANALOG_SETTINGS	0x7E
SET_LED_CONFIG	0x6E
DEFAULT_UART_SPEED_SESSION	0xF1

## Card related commands

### *General purpose card related commands*

GET_CARD_ID	0x13
GET_CARD_ID_EX	0x2C
GET_DLOGIC_CARD_TYPE	0x3C
GET_LAST_CARD_ID_EX	0x7C

### *Trailer block manipulation commands*

SECTOR_TRAILER_WRITE	0x1A
SECTOR_TRAILER_WRITE_UNSAFE	0x2F

### *Block manipulation commands*

BLOCK_READ	0x16
BLOCK_WRITE	0x17
BLOCK_IN_SECTOR_READ	0x18
BLOCK_IN_SECTOR_WRITE	0x19

### *Linear data manipulation commands*

LINEAR_READ	0x14
LINEAR_WRITE	0x15
LINEAR_FORMAT_CARD	0x25
LIN_ROW_READ	0x45



## Value block manipulation commands

### Direct block addressing

VALUE_BLOCK_READ	0x1D
VALUE_BLOCK_WRITE	0x1E
VALUE_BLOCK_INC	0x21
VALUE_BLOCK_DEC	0x22

### Indirect block addressing

VALUE_BLOCK_IN_SECTOR_READ	0x1F
VALUE_BLOCK_IN_SECTOR_WRITE	0x20
VALUE_BLOCK_IN_SECTOR_INC	0x23
VALUE_BLOCK_IN_SECTOR_DEC	0x24

## Commands for DESFIRE cards

GET_DESFIRE_UID	0x80
SET_DESFIRE_KEY	0x81
DESFIRE_WRITE_TO_FILE	0x82
DESFIRE_READ_FROM_FILE	0x83
DESFIRE_CREATE_APPLICATION	0x84
DESFIRE_CREATE_FILE	0x85
DESFIRE_CREATE_AES_KEY	0x86
DESFIRE_GET_KEY_CONFIG	0x87
DESFIRE_CHANGE_KEY_CONFIG	0x88
DESFIRE_DELETE_APPLICATION	0x89
DESFIRE_DELETE_FILE	0x8A
DESFIRE_SET_CONFIGURATION	0x8B
DESFIRE_FORMAT_CARD	0x8C
DESFIRE_FREE_MEM	0x8D
DESFIRE_WRITE_AES_KEY	0x8E
DESFIRE_CREATE_VALUE_FILE	0x8F
DESFIRE_READ_VALUE_FILE	0x9A
DESFIRE_INCREASE_VALUE_FILE	0x9B
DESFIRE_DECREASE_VALUE_FILE	0x9C
DESFIRE_CREATE_RECORD_FILE	0x97
DESFIRE_WRITE_RECORD	0x98
DESFIRE_READ_RECORDS	0x99
DESFIRE_CLEAR_RECORD	0x6D
DESFIRE_GET_APPLICATION_IDS	0xC0

**Commands for Mifare Desfire cards**

MFP_FIRST_AUTHENTICATE	0x6A
MFP_CHANGE_REG_KEY	0x6B
MFP_GET_UID	0x6C

**Commands for NFC Type 2 Tags**

GET_NFC_T2T_VERSION	0xB0
READ_COUNTER	0xB1
INCREMENT_COUNTER	0xB2

**Command for NFC Type 4 Tags**

NT4H_COMMON_CMD	0xB3
-----------------	------

**Originality checking commands**

READ_ECC_SIGNATURE	0xBF
--------------------	------

**Commands for “asynchronous UID sending” feature**

SET_CARD_ID_SEND_CONF	0x3D
GET_CARD_ID_SEND_CONF	0x3E
SET_BAD_SELECT_NR_MAX	0x3F
GET_BAD_SELECT_NR_MAX	0x44

**Power saving commands**

ENTER_SLEEP_MODE	0x46
LEAVE_SLEEP_MODE	0x47
AUTO_SLEEP_SET	0x4D
AUTO_SLEEP_GET	0x4E

**Light and display commands**

SET_DISPLAY_DATA	0x72
SET_SPEAKER_FREQUENCY	0x73
SET_DISPLAY_INTENSITY	0x74
GET_DISPLAY_INTENSITY	0x75

**uFR BASE Control commands**

UFR_XRC_LOCK_OPEN	0x60
UFR_XRC_SET_RELAY_STATE	0x61
UFR_XRC_GET_IO_STATE	0x62

**Shared Ram card emulation commands**

ENTER_SHARE_RAM_COMM_MODE	0x78
EXIT_SHARE_RAM_COMM_MODE	0x79
READ_SHARE_RAM	0x7A
WRITE_SHARE_RAM	0x7B

### *ISO 14443-4 protocol commands*

I_BLOCK_TRANSCEIVE	0x90
R_BLOCK_TRANSCEIVE	0x91
S_BLOCK_DESELECT	0x92
SET_ISO14433_4_MODE	0x93
APDU_TRANSCEIVE	0x94

### *uFR Online commands*

ESP_SET_IO_STATE	0xF3
ESP_GET_IO_STATE	0xF4
ESP_READER_TIME_WRITE	0xF5
ESP_READER_TIME_READ	0xF6
ESP_READER_EEPROM_READ	0xF7
ESP_SET_DISPLAY_DATA	0xF8
ESP_READER_RESET	0xF9
ESP_READER_PASSWORD_WRITE	0xFA
ESP_READER_EEPROM_WRITE	0xFB
ESP_GET_READER_SERIAL	0xE7

### *Miscellaneous functions*

CHECK_UID_CHANGE	0xE4
RF_RESET	0xE5
GET_READER_STATUS	0xE6
READ_TT_STATUS	0xB4

## **DEVICE RELATED COMMANDS**

### **GENERAL PURPOSE DEVICE RELATED COMMANDS**

#### ***GET\_READER\_TYPE (0x10)***

It gives a device (reader) type in size of 4 bytes which is hard coded in the firmware.

uFR Classic has a value of 0xD1150021.

CMD\_EXT set is not in use.

CMD\_Par0 and CMD\_Par1 are not in use.

If everything operates as expected the RSP packet is sent and after that also the RSP\_EXT packet of 5 bytes which contains 4 byte DeviceType values (little-endian) and CHECKSUM byte.

**Example:**

Send CMD GET\_READER\_TYPE  
**55 10 AA 00 00 00 F6**

**Where**

55 - CMD\_HEADER  
10 - CMD\_CODE  
AA - CMD\_TRAILER  
00 00 00 - CMD\_EX\_Length and CMD\_Par0 and CMD\_Par1 not used  
F6 - CHECKSUM

Reader answer with RESPONSE – RSP packet followed by RSP\_EXT packet

**DE 10 ED 05 00 00 2D 21 00 15 D1 EC**

**Where RSP PACKET contains**

DE - RSP\_HEADER  
10 - CMD\_CODE  
ED - RSP\_TRAILER  
05 - RSP\_EXT\_Length  
00 00 - RSP\_Val0 and RSP\_Val1 not used  
2D - CHECKSUM

**and RSP\_EXT contains**

21 00 15 D1 - Device type (currently uFR Classic D1 15 00 21, little-endian notation)  
EC - CHECKSUM

**GET\_READER\_SERIAL (0x11)**

It gives the device (reader) serial number with length of 4 bytes. On the older devices, this serial number has been read from the EEPROM MFRC chip.

The CMD\_EXT set is not in use.

The CMD\_Par0 and CMD\_Par1 are not in use.

If everything operates as expected the RESPONSE set is sent and after that also the RESPONSE EXT set of 5 bytes which contains 4 byte ReaderSerialNumber values (little-endian) and at the end one checksum byte.

**Example:**

Send CMD GET\_READER\_SERIAL  
**55 11 AA 00 00 00 F5**

**Where**

55 - CMD\_HEADER  
11 - CMD\_CODE  
AA - CMD\_TRAILER  
00 00 00 - CMD\_EX\_Length and CMD\_Par0 and CMD\_Par1 not used  
F5 - CHECKSUM

Reader answer with RESPONSE – RSP packet followed by RSP\_EXT packet

**DE 11 ED 05 00 00 2E 54 7E 1A 5D 74**

Where RSP PACKET contains

DE - RSP\_HEADER  
11 - CMD\_CODE  
ED - RSP\_TRAILER  
05 - RSP\_EXT\_Length  
00 00 - RSP\_Val0 and RSP\_Val1 not used  
2E - CHECKSUM

and RSP\_EXT contains

54 7E 1A 5D - Device type (currently serial is 5D 1A 7E 54, little-endian notation)  
74 - CHECKSUM

### **GET\_SERIAL\_NUMBER (0x40)**

Command returns reader serial number in string representation, like “UF123456”.

The CMD\_EXT set is not in use.

The CMD\_Par0 and CMD\_Par1 are not in use.

Example:

Send CMD GET\_SERIAL\_NUMBER  
**55 40 AA 00 AA CC E0**

Where

55 - CMD\_HEADER  
40 - CMD\_CODE  
AA - CMD\_TRAILER  
00 AA CC - CMD\_EX\_Length and CMD\_Par0 and CMD\_Par1 not used  
E0 - CHECKSUM

Reader answer with RESPONSE – RSP packet followed by RSP\_EXT packet

**DE 40 ED 09 00 00 81 55 46 31 32 33 34 35 36 1B**

Where RSP PACKET contains

DE - RSP\_HEADER  
40 - CMD\_CODE  
ED - RSP\_TRAILER  
09 - RSP\_EXT\_Length  
00 00 - RSP\_Val0 and RSP\_Val1 not used  
81 - CHECKSUM

and RSP\_EXT contains

55 46 31 32 33 34 35 36 - Device readers number (currently serial is “UF123456”)  
1B - CHECKSUM

### **GET\_HARDWARE\_VERSION (0x2A)**

Returns reader hardware version as two byte representation of higher and lower byte.

The CMD\_EXT set is not in use.

The CMD\_Par0 and CMD\_Par1 are not in use.

High byte of the hardware version is RSP\_Val0.

Low byte of hardware version is PSP\_Val1

**Example:**

```
CMD      55 2A AA 00 00 00 DC
RSP      DE 2A ED 00 01 01 20
```

**GET\_FIRMWARE\_VERSION (0x29)**

Returns reader firmware version as two byte representation of higher and lower byte.

The CMD\_EXT set is not in use.

The CMD\_Par0 and CMD\_Par1 are not in use.

High byte of the firmware version is RSP\_Val0.

Low byte of the firmware version is PSP\_Val1.

**Example:**

```
CMD      55 29 AA 00 00 00 DD
RSP      DE 29 ED 00 03 09 17
```

**GET\_BUILD\_NUMBER (0x2B)**

Returns reader firmware build version as one byte representation.

The CMD\_EXT set is not in use.

The CMD\_Par0 and CMD\_Par1 are not in use.

Build number of the firmware version is RSP\_Val0.

**Example:**

```
CMD      55 2B AA 00 00 00 DB
RSP      DE 2B ED 00 C8 00 D7
```

**READER\_KEY\_WRITE (0x12)**

This function writes MIFARE key into internal EEPROM, at key index location (0 – 31).

- CMD\_Par0 is key index
- CMD\_Par1 is not in use
- array from 1st to 6th byte of CMD\_EXT set contains 6-byte key
- 7th byte of CMD\_EXT set is CHECKSUM

**Example:**

Write Key FF FF FF FF FF FF into key index 00

```
CMD      55 12 AA 07 00 00 F1
ACK      AC 12 CA 07 00 00 7A

CMD_EXT  FF FF FF FF FF FF 07
RSP      DE 12 ED 00 00 00 28
```

### **USER\_DATA\_READ (0X1B)**

Function gives the 16 bytes from internal EEPROM user space.

The CMD\_Par0 and CMD\_Par1 are not in use.

- array from 1st to 16th byte of RSP\_EXT set contains 16 bytes of user data
- 17th byte of RSP\_EXT set is CHECKSUM.

#### **Example:**

```
CMD      55 1B AA 00 00 00 EB
RSP      DE 1B ED 11 00 00 40
RSP_EXT  6A 6A 00 00 36 00 00 00 30 00 32 00 38 00 41 00 54
```

### **USER\_DATA\_WRITE (0X1C)**

This function writes 16 bytes into user space.

The CMD\_Par0 and CMD\_Par1 are not in use.

- array from 1st to 16th byte of CMD\_EXT set contains 16 bytes of user data
- 17th byte of CMD\_EXT set is CHECKSUM.

#### **Example:**

write into user space values we read in previous example (6A 6A 00 00 36 00 00 00 30 00 32 00 38 00 41 00 54)

```
CMD      55 1C AA 11 00 00 F9
ACK      AC 1C CA 11 00 00 72

CMD_EXT  6A 6A 00 00 36 00 00 00 30 00 32 00 38 00 41 00 54
RSP      DE 1C ED 00 00 00 36
```

### **READER\_KEYS\_LOCK (0x27)**

If the keys (Mifare, AES, ...) in the reader are not locked - that means everyone can change it. If you want to protect the reader of changing keys then must lock the keys. Initially, uFReader is not locked. You can provide any password what you want, but must contain 8 bytes.

#### **Example:**

Lock keys with password "22222222" (we use printable characters for test)

```
CMD      55 27 AA 09 00 00 D8
ACK      AC 27 CA 09 00 00 4F
```

```
CMD_EXT  32 32 32 32 32 32 32 32 07
RSP      DE 27 ED 00 00 00 1B
```

### **READER\_KEYS\_UNLOCK (0x28)**

If you want to change the keys (Mifare, AES, ...) in the reader, reader must be unlocked first. The same password must be used to unlock as when we locked the reader. If you mistype the password - reader would reset.

#### **Example:**

Unlock keys with password "22222222" (we use printable characters for test)

```
CMD      55 28 AA 09 00 00 E5
ACK      AC 28 CA 09 00 00 4E
```

```
CMD_EXT  32 32 32 32 32 32 32 32 07
RSP      DE 28 ED 00 00 00 22
```

### **READER\_PASSWORD\_WRITE (0x33)**

This function is used in Common, Advance and Access Control set of functions.

It defines/changes password which I used for:

- Locking/unlocking keys stored into reader
- Setting date/time of RTC

The CMD\_Par0 and CMD\_Par1 are not in use.

- array from 1st to 8th byte of CMD\_EXT set contains current password, 9th to 16th byte contains new password
- 17th byte of CMD\_EXT set is CHECKSUM.

#### **Example:**

Current password is "11111111" , new password is "22222222"

```
CMD      55 33 AA 11 00 00 E4
ACK      AC 33 CA 11 00 00 4B
CMD_EXT  31 31 31 31 31 31 31 31 31 32 32 32 32 32 32 32 32 07
RSP      DE 33 ED 00 00 00 07
```

### **SELF\_RESET (0X30)**

Function performs soft restart of device.

The CMD\_EXT set is not in use.

The CMD\_Par0 and CMD\_Par1 are not in use



**Example:**

```

CMD          55 30 AA 00 00 00 D6

RSP          DE 30 ED 00 00 00 0A
RSP_EXT      03 55 55 BB

```

**SET\_UART\_SPEED (0X70)**

Function writes new value of UART's baud rate. For example 115200. Command sending is at current baud rate, ACK is at current baud rate, but response is at new baud rate. In future, the device will communicate at a new baud rate.

The CMD\_Par0 and CMD\_Par1 are not in use.

- array from 1st to 4th byte of CMD\_EXT set contains 4 byte long baud rate (little-endian)
- 5th byte of CMD\_EXT set is CHECKSUM.

**Example:**

```

CMD          55 70 AA 05 00 00 91
ACK          AC 70 CA 00 00 00 1D
CMD_EXT      00 C2 01 00 CA
RSP          ED 70 DE 00 00 00 4A

```

**DEFAULT\_UART\_SPEED\_SESSION (0xF1)**

Command starts the session on default UART baud rate, regardless of the setting speed of the reader. That is a specific command. First you must reset the reader over the RTS pin. After that you will receive four bytes from the bootloader on default UART baud rate. Command is then sent. It is useful to set UART to default speed if you forget the current speed by executing SET\_UART\_SPEED with default UART speed.

CMD\_Par0 = 1 and CMD\_Par1 = 1.

CMD\_EXT not in use.

RSP\_EXT not in use.

**Example:**

```

RESET OVER RTS PIN
BOOTLOADER ACK      03 55 55 BB
CMD          55 F1 AA 00 00 00 15
RSP          ED 70 DE 00 00 00 4A

```

**RED\_LIGHT\_CONTROL (0X71)**

This function turns on or off red LED lights. If turned on, the green LED will stop flashing.

The CMD\_EXT set is not in use.

CMD\_Par0 – 0x01 turn red LED on, 0x00 – turn red LED off.

CMD\_Par1 is not in use.

**Example:**

To turn red LED ON, send CMD packet

```
CMD      55 71 AA 00 01 00 96
RSP      DE 71 ED 00 00 00 49
```

To turn red LED OFF, send CMD packet

```
CMD      55 71 AA 00 00 00 95
RSP      DE 71 ED 00 00 00 49
```

For classic uFR Classic, uFR Classic CS and uFR and uFR XL devices.

The function prohibits the blinking of the green diode (if this option is set), and sets color on RGB diodes.

CMD\_Par0 – 0x01 set RGB color, 0x00 – turn RGB led off, green is blinking.

CMD\_Par1 = 0xC5

CMD\_EXT

- 1st byte is intensity of RED light (0 - 255)
- 2nd byte is intensity of GREEN light (0 - 255)
- 3rd byte is intensity of BLUE light (0 - 255)
- 4th byte is intensity of light in % (0 - 100)
- 5th byte is checksum

**Example:**

RED = 255, GREEN = 255, BLUE = 0, intensity = 50%

```
CMD      55 71 AA 05 01 C5 56
ACK      AC 71 CA 05 01 C5 DD
CMD_EXT  FF FF 00 32 39
RSP      DE 71 ED 00 00 00 49
```

To turn red LED OFF, send CMD packet

```
CMD      55 71 AA 00 00 00 95
RSP      DE 71 ED 00 00 00 49
```

From version 5.0.55.

Before the function calls, the command SET\_LED\_CONFIG with the CMD\_Par0 = 0 must be called, or the reader is already in mode of blocking automatic signalization. Function sets the color of the RGB diodes. This color stays on the RGB until the command SET\_LED\_CONFIG with the CMD\_Par0 = 0 is called. Intensity of light is defined by a parameter stored using the command SET\_DISPLAY\_INTENSITY.

CMD\_Par0 – 0x02 set RGB color.

CMD\_Par1 = 0xC5

CMD\_EXT

- 1st byte is intensity of RED light (0 - 255)
- 2nd byte is intensity of GREEN light (0 - 255)
- 3rd byte is intensity of BLUE light (0 - 255)
- 4th byte exists for compatibility reasons, value doesn't matter.
- 5th byte is checksum

**Example:**

RED = 255, GREEN = 255, BLUE = 0

CMD           55 71 AA 05 02 C5 53

ACK           AC 71 CA 05 02 C5 DC

CMD\_EXT      FF FF 00 32 39

RSP           DE 71 ED 00 00 00 49

**USER\_INTERFACE\_SIGNAL (0x26)**

This function turns sound and light reader signals. Sound signals are performed by a reader buzzer and light signals are performed by reader LEDs.

There are predefined signal values for sound and light:

light_signal_mode:		beep_signal_mode:	
0	None	0	None
1	Long Green	1	Short
2	Long Red	2	Long
3	Alternation	3	Double Short
4	Flash	4	Triple Short
		5	Triplet Melody

The CMD\_EXT set is not in use.

CMD\_Par0 is value of light signal mode (0 - 4)

CMD\_Par1 is value of beep signal mode (0 - 5)

**Example:**

light signal mode is Long Green (1), beep signal mode is Long (2)

```
CMD      55 26 AA 00 01 02 E1
RSP      DE 26 ED 00 00 00 1C
```

### **SET\_DISPLAY\_DATA (0x72)**

This feature works with the LED RING 24 display module.

Function enables sending data to the display. A string of data contains information about the intensity of color in each cell of the display. Each cell has three LEDs (red, green and blue). For each cell of the three bytes is necessary. The first byte indicates the intensity of the green color, the second byte indicates the intensity of the red color, and the third byte indicates the intensity of blue color. For example, if the display has 16 cells, an array contains 48 bytes. Value of intensity is in the range from 0 to 255.

CMD\_Par0 number of bytes

CMD\_Par1 not in use

CMD\_EXT contains data for display with checksum

#### **Example:**

green = 0, red = 0xFF, blue = 0x80

```
CMD      55 72 AA 49 48 00 93
ACK      AC 72 CA 49 48 00 1C
CMD_EXT  00 FF 80 00 FF 80 00 FF 80 00 FF 80 00 FF 80 00 FF 80 00 FF 80
00 FF 80 00 FF 80 00 FF 80 00 FF 80 00 FF 80 00 FF 80 00 FF 80 00 FF 80
00 FF 80 00 FF 80 00 FF 80 00 FF 80 00 FF 80 00 FF 80 00 FF 80 00 FF 80
00 FF 80 07
RSP      DE 72 ED 00 00 00 48
```

From version 5.0.55

New feature is the RGB port selection. Internal port uses RGB diodes on the reader PCB. Card size reader has two diodes. XL reader has four diodes. External port uses LED RING with RGB diodes.

Before the function calls, the command SET\_LED\_CONFIG with the CMD\_Par0 = 0 must be called, or the reader is already in mode of blocking automatic signalization. Function sets the color of the RGB diodes. This color stays on the RGB until the command SET\_LED\_CONFIG with the CMD\_Par0 = 1 is called. Intensity of light is defined by a parameter stored using the command SET\_DISPLAY\_INTENSITY.

CMD\_Par0 number of bytes

CMD\_Par1 - external RGB port = 0, internal RGB port = 1

CMD\_EXT contains data for display with checksum

**Example:**

First RGB green = 0xFF, red = 0, blue = 0. Second RGB green = 0, red = 0, blue = 0xFF

```
CMD      55 72 AA 07 06 01 94
ACK      AC 72 CA 07 06 01 1B
CMD_EXT  FF 00 00 00 00 FF 07
RSP      DE 72 ED 00 00 00 48
```

**SET\_DISPLAY\_INTENSITY (0x74)**

Function sets the intensity of light on the display. Value of intensity is in the range 0 to 100.

CMD\_Par0 is display intensity

CMD\_Par1 not in use

CMD\_EXT not in use

**Example:**

display intensity is 50

```
CMD      55 74 AA 00 32 00 C0
RSP      DE 74 ED 00 00 00 4E
```

**GET\_DISPLAY\_INTENSITY (0x75)**

Function gets the intensity of light on the display.

CMD\_Par0 not in use

CMD\_Par1 not in use

CMD\_EXT not in use

RSP\_EXT 1st byte is intensity, 2nd byte is checksum

**Example:**

```
CMD      55 75 AA 00 00 00 91
RSP      DE 75 ED 02 00 00 4B
RSP_EXT  32 39
```

**SET\_SPEAKER\_FREQUENCY (0x73)**

Function sets the frequency of the speaker. The speaker is working on this frequency until a new frequency setting. To stop the operation set frequency to zero.

Period of sound frequency calculated according to the following formula

$$\text{period} = 65535 - 1500000 / (2 * \text{frequency in Hertz})$$

CMD\_Par0 is low byte of sound's period

CMD\_Par1 is high byte of sound's period

**Example:**

set frequency of 1600Hz

**CMD**            55 73 AA 00 2B FE 60

**RSP**            DE 73 ED 00 00 00 47

**SET\_RF\_ANALOG\_SETTINGS (0x7D)**

This function allows you to adjust the value of several registers on PN512. These are registers: RFCfgReg, RxThresholdReg, GsNOnReg, GsNOffReg, CWGsPReg, ModGsPReg. This can be useful if you want to increase the operation distance of card, or when it is necessary to reduce the impact of environmental disturbances.

CMD\_Par0 type of communication with tag

ISO14443 type A	0x01
ISO14443 type B	0x02
ISO14443-4 212 Kbps	0x03
ISO14443-4 424 Kbps	0x04

CMD\_Par1 0 - user settings, 1 - factory default settings

CMD\_EXT

- 1st byte is value of RFCfgReg
- 2nd byte is value of RxThresholdReg
- 3rd byte is value of GsNOnReg
- 4th byte is value of CWGsPReg
- 5th byte is value of GsNOffReg for Type A or ModGsPReg for type B

For ISO14443-4 212 Kbps and ISO14443-4 424 Kbps CMD\_EXT contains just first 2 bytes

**Example:**

RFCfgReg = 0x79, RxThesholdReg = 0x87, GsNonReg = 0x88, CWGsPReg = 0x20, GsNOffReg = 0x88

```

CMD      55 7D AA 06 01 00 8C
ACK      AC 7D CA 06 01 00 23
CMD_EXT  79 87 88 20 88 E5
RSP      DE 7D ED 00 00 00 55

```

**GET\_RF\_ANALOG\_SETTINGS (0x7E)**

The function reads the value of the registers RFCfgReg, RxThresholdReg, GsNOnReg, GsNOffReg, CWGsPReg, ModGsPReg.

CMD\_Par0 type of communication with tag

ISO14443 type A	0x01
ISO14443 type B	0x02
ISO14443-4 212 Kbps	0x03
ISO14443-4 424 Kbps	0x04

The CMD\_EXT set is not in use.

## RSP\_EXT

- 1st byte is value of RFCfgReg
- 2nd byte is value of RxThresholdReg
- 3rd byte is value of GsNOnReg
- 4th byte is value of CWGsPReg
- 5th byte is value of GsNOffReg for Type A or ModGsPReg for type B

For ISO14443-4 212 Kbps and ISO14443-4 424 Kbps RSP\_EXT contains just first 2 bytes

## **SET\_LED\_CONFIG (0x6E)**

Minimal firmware version is 3.9.53

Light signalization configuration. Parameters are written into the device, and they are reloaded after reset or power up.

CMD\_Par0 configuration low byte

CMD\_Par1 configuration high byte

Green light blinking on - CMD\_Par0 bit 0 is 1

Green light blinking off - CMD\_Par0 bit 0 is 0

### **Example:**

Green light blinking turn on

```
CMD      55 6E AA 00 01 00 97
RSP      DE 6E ED 00 00 00 64
```

Green light blinking turn off

```
CMD      55 6E AA 00 00 00 98
RSP      DE 6E ED 00 00 00 64
```

## **UFR\_BASE\_HD\_LOCK\_OPEN (0x60)**

BASE HD uFR only.

Electric strike switches when the function is called. Pulse duration determined by function.

CMD\_Par0 pulse duration in ms low byte

CMD\_Par1 pulse duration in ms high byte

### **Example:**

Pulse duration is 300ms (0x12C)

```
CMD      55 60 AA 00 2C 01 B9
RSP      DE 60 ED 00 00 00 5A
```



## BARRIER CONTROL device command differences

Function controls two electric actuators on the barrier access control device. If the most significant bit in CMD\_Par1 is set will be activated actuator 1, else will be activated actuator 2. The maximum time that can be set is 0x7FFF ms.

CMD\_Par0 duration of active state in ms low byte

CMD\_Par1 & 0x7F duration of active state in ms high byte

If CMD\_Par1 & 0x80 actuator 1 is active, else actuator 2 is active

### Example:

Duration of active state is 5000ms (0x1388), actuator 1 set

```
CMD      55 60 AA 00 88 93 8B
RSP      DE 60 ED 00 00 00 5A
```

## UFR\_BASE\_HD\_SET\_RELAY\_STATE (0x61)

BASE HD uFR only.

Function switches relay.

CMD\_Par0 1 - relay on, 0 - relay off

### Example:

Relay on.

```
CMD      55 61 AA 00 01 00 A6
RSP      DE 61 ED 00 00 00 59
```

## BARRIER CONTROL device command differences

Function switches relay, and sets state of OUT1 to OUT3 outputs

CMD\_Par0 1 - relay on, 0 - relay off

Bit 0 of CMD\_Par1 1 - OUT1 is high, 0 - OUT1 is low

Bit 1 of CMD\_Par1 1 - OUT2 is high, 0 - OUT2 is low

Bit 2 of CMD\_Par1 1 - OUT3 is high, 0 - OUT3 is low

### Example:

Relay on, OUT 1 is low, OUT2 is low, OUT3 is high

```
CMD      55 61 AA 00 01 04 A2
RSP      DE 61 ED 00 00 00 59
```

### **UFR\_BASE\_HD\_GET\_IO\_STATE (0x62)**

BASE HD uFR only.

Function returns states of 3 IO pins.

#### **RSP\_EXT**

1st byte 1- voltage at the intercom terminals detected, 0 - no voltage at the intercom terminals

2nd byte 1 - voltage at DIGIN pin is high, 0 - voltage at DIGIN pin is low.

3rd byte 1 - relay is turn on, 0 - relay is turn off

#### **Example:**

```
CMD          55 62 AA 00 00 00 A4
RSP          DE 62 ED 04 00 00 5C
RSP_EXT     01 00 01 07
```

### **BARRIER CONTROL device command differences**

Function returns state of five input pins, four output pins, and two actuators.

#### **RSP\_EXT**

Bit 0 of 1st byte	1- voltage detected at the IN1, 0 - no voltage at IN1
Bit 1 of 1st byte	1- voltage detected at the IN2, 0 - no voltage at IN2
Bit 2 of 1st byte	1- voltage detected at the IN3, 0 - no voltage at IN3
Bit 3 of 1st byte	1- voltage detected at the IN4, 0 - no voltage at IN4
Bit 4 of 1st byte	1- proximity sensor activated, 0 - proximity sensor is not active
Bit 0 of 2nd byte	1 - relay is on, 0 - relay is off
Bit 1 of 2nd byte	1 - OUT1 is high, 0 - OUT1 is low
Bit 2 of 2nd byte	1 - OUT2 is high, 0 - OUT2 is low
Bit 3 of 2nd byte	1 - OUT3 is high, 0 - OUT3 is low
Bit 0 of 3rd byte	1 - actuator 1 is active, 0 - actuator 1 is not active
Bit 1 of 3rd byte	1 - actuator 2 is active, 0 - actuator 2 is not active

#### **Example:**

Voltage on IN1, proximity sensor is active, relay is on, actuator 1 is active

```
CMD      55 62 AA 00 00 00 A4
RSP      DE 62 ED 04 00 00 5C
RSP_EXT  11 01 01 18
```

## **CARD RELATED COMMANDS**

For all the functions for operations with cards the following applies:

- They operates only with one card in the device field
- If there is no card in the field device return error NO\_CARD (0x08).
- If there is more than one card in the field the behavior of the device is unpredictable but some of the next cases are possible:
  - Gives NO\_CARD error or
  - Just one card is detected and the device gives its type (this is due to the lack of a cascade of selection and the collision process as described in the ISO14443 standard).

## **GENERAL PURPOSE CARD RELATED COMMANDS**

### ***GET\_CARD\_ID (0x13)***

This function return the serial number of the card which is currently in the readers field and the one byte value that represents its type. For Mifare Classic 1K the type is 0x08, Mifare Classic 4k type is 0x18 and Mifare Classic Mini cards type is 0x09.

The CMD\_EXT set is not in use.

The CMD\_Par0 and CMD\_Par1 are not in use.

If everything operates as expected the RESPONSE set is sent and after that also the RESPONSE EXT set of 5 bytes which contains 4 byte Card UID values (little-endian) and CHECKSUM byte.

RSP\_Val0 contains value of the card type.

This function applies only for card with 4-byte UID. For longer UID's, use GET\_CARD\_ID\_EX (0x2C)

### **Example:**

```
CMD      55 13 AA 00 00 00 F3
RSP      DE 13 ED 05 08 00 34
RSP_EXT  13 E2 0A 87 83
```

Where in RSP packet byte 05 represents RSP\_EXT\_length and byte 08 represents CardType – 0x08 – Mifare Classic.

RSP\_EXT returns Card UID (little-endian) and CHECKSUM of UID bytes.

If error occurs, like NO\_CARD, device will answer with ERR packet

```
CMD      55 13 AA 00 00 00 F3
ERR      EC 08 CE 00 00 00 31
```

Where byte 08 represents ERR\_CODE for NO\_CARD error.

### **GET\_CARD\_ID\_EX (0x2C)**

Use this function for cards with UID longer than 4 byte.

This function return the serial number of the card which is currently in the readers field, length of serial number (4 (UID size: single), 7 (UID size: double) or 10 (UID size: triple)), and the one byte value that represents its type. For Mifare Classic 1K the type is 0x08, Mifare Classic 4k type is 0x18 and Mifare Classic Mini cards type is 0x09.

The CMD\_EXT set is not in use.

The CMD\_Par0 and CMD\_Par1 are not in use.

If everything operates as expected the RSP packet is sent and after that also the RSP\_EXT packet of 11 bytes which contains card serial number and at the end one checksum byte.

RSP\_Val0 contains value of the card type.

RSP\_Val1 contains length of card serial number.

#### **Example:**

```
CMD      55 2C AA 00 00 00 DA
  
RSP      DE 2C ED 0B 08 04 1F
RSP_EXT  13 E2 0A 87 00 00 00 00 00 00 83
```

Where in RSP packet byte 0B represents RSP\_EXT\_Length, byte 08 means Card Type – Mifare Classic 1K, and byte 04 is length of card UID in RSP\_EXT packet.

RSP\_EXT packet contains card UID bytes and CHECKSUM.

If error occurs, like NO\_CARD, device will answer with ERR packet

```
CMD      55 2C AA 00 00 00 DA
ERR      EC 08 CE 00 00 00 31
```

Where byte 08 represents ERR\_CODE for NO\_CARD error.

### **GET\_LAST\_CARD\_ID\_EX (0x7C)**

This function returns UID of last card which was present in RF field of reader. It can handle all three known types: 4, 7 and 10 byte long UIDs. Difference with GetCardIdEx is that card does not be in RF field mandatory, UID value is stored in temporary memory area.

The CMD\_EXT set is not in use.

The CMD\_Par0 and CMD\_Par1 are not in use.

If everything operates as expected the RSP packet is sent and after that also the RSP\_EXT packet of 11 bytes which contains card serial number and at the end one checksum byte.

RSP\_Val0 contains value of the card type.

RSP\_Val1 contains length of card serial number.

**Example:**

```

CMD          55 7C AA 00 AA CC EC

RSP          DE 7C ED 0B 08 04 4F
RSP_EXT      52 DA D9 95 00 00 00 00 00 00 CB
  
```

Where in RSP packet byte 0B represents RSP\_EXT\_Length, byte 08 means Card Type – Mifare Classic 1K, and byte 04 is length of card UID in RSP\_EXT packet. RSP\_EXT packet contains card UID bytes and CHECKSUM.

If error occurs, like NO\_CARD, device will answer with ERR packet

```

CMD          55 7C AA 00 AA CC EC
ERR          EC 08 CE 00 AA CC 53
  
```

Where byte 08 represents ERR\_CODE for NO\_CARD error.

**GET\_DLOGIC\_CARD\_TYPE (0x3C)**

This function returns card type according to following enumeration list:

DL_MIFARE_ULTRALIGHT	0x01
DL_MIFARE_ULTRALIGHT_EV1_11	0x02
DL_MIFARE_ULTRALIGHT_EV1_21	0x03
DL_MIFARE_ULTRALIGHT_C	0x04
DL_NTAG_203	0x05
DL_NTAG_210	0x06
DL_NTAG_212	0x07
DL_NTAG_213	0x08
DL_NTAG_215	0x09
DL_NTAG_216	0x0A
MIKRON_MIK640D	0x0B
NFC_T2T_GENERIC	0x0C
DL_NT3H_1101	0x0D
DL_NT3H_1201	0x0E
DL_NT3H_2111	0x0F
DL_NT3H_2211	0x10
DL_NTAG_413_DNA	0x11

DL_NTAG_424_DNA	0x12
DL_NTAG_413_DNA_TT	0x13
DL_NTAG_210U	0x14
DL_NTAG_213_TT	0x15
DL_MIFARE_MINI	0x20
DL_MIFARE_CLASSIC_1K	0x21
DL_MIFARE_CLASSIC_4K	0x22
DL_MIFARE_PLUS_S_2K_SL0	0x23
DL_MIFARE_PLUS_S_4K_SL0	0x24
DL_MIFARE_PLUS_X_2K_SL0	0x25
DL_MIFARE_PLUS_X_4K_SL0	0x26
DL_MIFARE_DESFIRE	0x27
DL_MIFARE_DESFIRE_EV1_2K	0x28
DL_MIFARE_DESFIRE_EV1_4K	0x29
DL_MIFARE_DESFIRE_EV1_8K	0x2A
DL_MIFARE_DESFIRE_EV2_2K	0x2B
DL_MIFARE_DESFIRE_EV2_4K	0x2C
DL_MIFARE_DESFIRE_EV2_8K	0x2D
DL_MIFARE_PLUS_S_2K_SL1	0x2E
DL_MIFARE_PLUS_X_2K_SL1	0x2F
DL_MIFARE_PLUS_EV1_2K_SL1	0x30
DL_MIFARE_PLUS_X_2K_SL2	0x31
DL_MIFARE_PLUS_S_2K_SL3	0x32
DL_MIFARE_PLUS_X_2K_SL3	0x33
DL_MIFARE_PLUS_EV1_2K_SL3	0x34
DL_MIFARE_PLUS_S_4K_SL1	0x35
DL_MIFARE_PLUS_X_4K_SL1	0x36
DL_MIFARE_PLUS_EV1_4K_SL1	0x37
DL_MIFARE_PLUS_X_4K_SL2	0x38
DL_MIFARE_PLUS_S_4K_SL3	0x39
DL_MIFARE_PLUS_X_4K_SL3	0x3A
DL_MIFARE_PLUS_EV1_4K_SL3	0x3B
DL_MIFARE_PLUS_SE_SL0	0x3C

DL_MIFARE_PLUS_SE_SL1	0x3D
DL_MIFARE_PLUS_SE_SL3	0x3E
DL_MIFARE_DESFIRE_LIGHT	0x3F
DL_GENERIC_ISO14443_4	0x40
DL_GENERIC_ISO14443_4_TYPE_B	0x41
DL_GENERIC_ISO14443_3_TYPE_B	0x42
DL_MIFARE_PLUS_EV1_2K_SL0	0x43
DL_MIFARE_PLUS_EV1_4K_SL0	0x44
DL_IMEI_UID	0x80

**Example:**

```

CMD      55 3C AA 00 00 00 CA
RSP      DE 3C ED 00 21 00 35

```

Where byte 21 in RSP packet represents card type – 0x21 – Mifare Classic 1K.

If error occurs, like NO\_CARD, device will answer with ERR packet

```

CMD      55 3C AA 00 00 00 CA
ERR      EC 08 CE 00 00 00 31

```

Where byte 08 represents ERR\_CODE for NO\_CARD error.

## FUNCTIONS FOR READING AND WRITING THE DATA INTO THE CARD

### *Authentication mode considerations for Mifare Classic tags and Mifare Plus tags*

The parameter AUTH\_MODE affects all the functions and determines authorization before reading or entering data in the card sector. This parameter can have the following values:

- RKA\_AUTH1A 0x00
- RKA\_AUTH1B 0x01
- AKM1\_AUTH1A 0x20
- AKM1\_AUTH1B 0x21
- AKM2\_AUTH1A 0x40
- AKM2\_AUTH1B 0x41
- PK\_AUTH1A 0x60
- PK\_AUTH1B 0x61
- PK\_AUTH1A\_AES 0x80 (Mifare Plus tags and NT4H tags uFR PLUS only)
- PK\_AUTH1B\_AES 0x81 (Mifare Plus tags uFR PLUS only)
- SAM\_KEY\_AUTH1A 0x10 (key A stored in SAM)
- SAM\_KEY\_AUTH1B 0x11 (key B stored in SAM)

For firmware versions from 5.0.29.

- `MFP_RKA_AUTH1A` 0x02 (Mifare Plus tags in SL3 mode and NT4H tags)
- `MFP_RKA_AUTH1B` 0x03 (Mifare Plus tags in SL3 mode)
- `MFP_AKM1_AUTH1A` 0x22 (Mifare Plus tags in SL3 mode)
- `MFP_AKM1_AUTH1B` 0x23 (Mifare Plus tags in SL3 mode)
- `MFP_AKM2_AUTH1A` 0x42 (Mifare Plus tags in SL3 mode)
- `MFP_AKM2_AUTH1B` 0x43 (Mifare Plus tags in SL3 mode)

From the names of each of these constants can be concluded that the suffixes 1A and 1B indicate that you want to perform authentication key A or key B.

**Prefixes in the names of constants represents modes of authentication, as following:**

RKA – abbreviation of Reader Key Authentication. This means that authentication will be done with one of the 32 keys (16 AES keys for Mifare Plus tags) that are stored in reader device. It is assumed that as one of the command parameter that is sent to the reader is the index of the desired key. Indexes are in range 0..31 (0..15 for AES keys).

Mifare Plus card using.

For firmware versions from 5.0.1 to 5.0.28. and `RKA_AUTH1A` or `RKA_AUTH1B` uses AES keys from reader AES keys space (index 0 - 15).

For firmware versions from 5.0.29 and `RKA_AUTH1A` or `RKA_AUTH1B` uses AES keys which are calculate from Crypto1 keys from reader Crypto1 keys space (index 0 - 31), and for `MFP_RKA_AUTH1A` or `MFP_RKA_AUTH1B` uses AES keys from reader AES keys space (index 0 - 15).

AKM1 and AKM2 – abbreviation of Automatic Key Modes. This means that the authentication will be done automatically with the keys stored in reader device and they are indexed on the basis of the block or sector address where the writing or reading is currently done.

This applies to any function for card writing and reading, even for linear modes. I

When using AKM1 mode, keys in range 0 to 15 (0 to 7 for Mifare Plus tags for sectors 0 - 7, and 8 - 15 again) are used as Key A for corresponding sectors, while keys indexed from 16 to 31 (8 to 15 for Mifare Plus tags for sectors 16 - 23, and 24 - 31) are used as Key B for corresponding sectors.

**Example for AKM1 keys indexes:**

```
Key[00] = Key A Sector 0; Key [01] = Key A Sector [1]; ... Key [15] = Key A Sector 15;  
Key[16] = Key B Sector 0; Key [17] = Key B Sector [1]; ... Key [31] = Key B Sector 15;
```

When using AKM2, keys are indexed by odd and even order, so even keys indexes are used as Key A and odd keys indexes are used as Key B (for Mifare Plus tags key indexes are 0 - 15 for sectors 0 - 15, and they are repeated for sectors 15 - 30).

**Example for AKM2 keys indexes:**

```
Key[00] = Key A Sector 0; Key [02] = Key A Sector [1]; ... Key [30] = Key A Sector 15;  
Key[1] = Key B Sector 0; Key [3] = Key B Sector [1]; ... Key [31] = Key B Sector 15;
```

For 4k cards, which have 24 sectors more than 1k cards (total 40) for sectors 16 to 31 is used the same method as for indexing sectors 0 to 15 and for sectors 32 to 39 used the same method of indexing and for sectors 0 to 8.



Mifare Plus card using.

For firmware versions from 5.0.29 and AKM1\_AUTH1A or AKM1\_AUTH1B or AKM2\_AUTH1A or AKM2\_AUTH1B, reader keys uses in same manner as for Mifare classic card. AES key calculated from Crypto1 key.

For firmware versions from 5.0.1 to 5.0.28 in AKM1\_AUTH1A or AKM1\_AUTH1B or AKM2\_AUTH1A or AKM1\_AUTH1A, and version 5.0.29 in MFP\_AKM1\_AUTH1A or MFP\_AKM1\_AUTH1B or MFP\_AKM2\_AUTH1A or MFP\_AKM1\_AUTH1B, uses reader keys from AES keys space (index 0 - 15).

#### Example for AKM1 keys indexes:

```
Key[00] = Key A Sector 0; Key [01] = Key A Sector 1; ... Key [07] = Key A Sector 7;
Key[00] = Key A Sector 8; Key [01] = Key A Sector 9; ... Key [07] = Key A Sector 15;
Key[00] = Key A Sector 16; Key [01] = Key A Sector 17; ... Key [07] = Key A Sector 23;
Key[00] = Key A Sector 24; Key [01] = Key A Sector 25; ... Key [07] = Key A Sector 31;
Key[00] = Key A Sector 32; Key [01] = Key A Sector 33; ... Key [07] = Key A Sector 39;
Key[08] = Key B Sector 0; Key [09] = Key B Sector 1; ... Key [15] = Key B Sector 7;
Key[08] = Key B Sector 8; Key [09] = Key B Sector 9; ... Key [15] = Key B Sector 15;
Key[08] = Key B Sector 16; Key [09] = Key B Sector 17; ... Key [15] = Key B Sector 23;
Key[08] = Key B Sector 24; Key [09] = Key B Sector 25; ... Key [15] = Key B Sector 31;
Key[08] = Key B Sector 32; Key [09] = Key B Sector 33; ... Key [15] = Key B Sector 39;
```

#### Example for AKM2 keys indexes:

```
Key[00] = Key A Sector 0; Key [02] = Key A Sector 1; ... Key [14] = Key A Sector 7;
Key[01] = Key B Sector 0; Key [03] = Key B Sector 1; ... Key [15] = Key B Sector 7;
Key[00] = Key A Sector 8; Key [02] = Key A Sector 9; ... Key [14] = Key A Sector 15;
Key[01] = Key B Sector 8; Key [03] = Key B Sector 9; ... Key [15] = Key B Sector 15;
Key[00] = Key A Sector 16; Key [02] = Key A Sector 17; ... Key [14] = Key A Sector 23;
Key[01] = Key B Sector 16; Key [03] = Key B Sector 17; ... Key [15] = Key B Sector 23;
Key[00] = Key A Sector 24; Key [02] = Key A Sector 25; ... Key [14] = Key A Sector 31;
Key[01] = Key B Sector 24; Key [03] = Key B Sector 25; ... Key [15] = Key B Sector 31;
Key[00] = Key A Sector 32; Key [02] = Key A Sector 33; ... Key [14] = Key A Sector 39;
Key[01] = Key B Sector 32; Key [03] = Key B Sector 33; ... Key [15] = Key B Sector 39;
```

PK – abbreviation for Provided Key refers to the authentication which is performed with key that is sent as a command parameter. Generally, this mode of authentication should be avoided due to the low level of security it provides, since key is passed as command parameter.

Mifare Plus using.

For firmware versions from 5.0.1 in PK\_AUTH1A\_AES or PK\_AUTH1B\_AES mode, 16 bytes AES key provided to reader.

For firmware versions from 5.0.29 in PK\_AUTH1A or PK\_AUTH1B mode, 6 bytes Crypto1 key provided to reader. AES key calculated from this Crypto1 key.

SAM\_KEY - abbreviation for Key stored into SAM (working with uFR CS reader with SAM, and firmware versions 5.100.xx only)

### **Authentication mode considerations for NTAG 21x and other T2T tags (supported from firmware version 3.9.10)**

NTAG 21x and some other T2T tags (such as Ultralight EV1) support different authentication method from the Mifare Classic tags. NTAG 21x tags authentication is done using ISO 14443A-3 PWD\_AUTH command, requiring from the reader to transmit secret code (PWD) of 4 bytes the tag, which responds with a PACK (PWD ACKNOWLEDGE). If the transmitted code is equal to that programmed in the tag, he responds with the correct PACK (length 2 bytes). PWD and PACK is typically written into the tag during the personalization process. The configuration pages are used to configure the memory access restriction of the tag. In order to familiarize with the methods of authentication of the NTAG 21x we recommend that you read "NTAG210 / 212, NFC Forum Type 2 Tag IC compliant with 48/128 bytes user memory Product data sheet" or "NTAG213 / 215/216, NFC Forum Type 2 Tag IC compliant with 144/504/888 bytes user memory data sheet Product" or "MF0ULx1, MIFARE Ultralight EV1 - Contactless IC ticket Product data sheet" that can be found on the manufacturer website. All these documents are marked "PUBLIC COMPANY".

NTAG 21x, Ultralight EV2 and other T2T tags supporting PWD\_AUTH, practically use 6 bytes (4 bytes that make up the PWD and 2 bytes of the PACK response) in our uFR readers we use the same mechanism as for Mifare Classic tags. The only difference is that a combined PWD (first 4 bytes of the key) and PACK (the last 2 bytes of the key) now forming a key (6 bytes in length). The resultant key can be prepared in advance and written in the card reader internal EEPROM (NV Memory) for using with Reader Key Authentication (RKA) method, or sent as a parameter of the uFR\_COM protocol command using Provided Key (PK) methods.

Note: Reader Key Authentication (RKA) methods with NTAG 21x, Ultralight EV2 and other T2T tags can not be used with uFR Classic and uFR Advanced commercial readers. These methods are possible only with newer reader series like uFR nano, uFR card size readers and HD Base with uFR support installed. On older models for this purpose can be used only Provided Key (PK) methods.

The following constants are declared for the parameter that determines the method for PWD\_AUTH for NTAG 21x, Ultralight EV2 and other T2T tags:

```
T2T_NO_PWD_AUTH 0x00  
T2T_RKA_PWD_AUTH 0x01  
T2T_PK_PWD_AUTH 0x61
```

These constants are used with the following uFR\_COM protocol commands:

**BLOCK\_READ**  
**BLOCK\_WRITE**  
**LINEAR\_READ**  
**LINEAR\_WRITE**  
**LIN\_ROW\_READ**

and passed as a parameter value controls AUTH\_MODE. If you use any other undeclared value as AUTH\_MODE, the effect will be the same as if you sent T2T\_NO\_PWD\_AUTH.

When for the AUTH\_MODE command parameter you send T2T\_RKA\_PWD\_AUTH or T2T\_PK\_PWD\_AUTH reader will always try to perform PWD\_AUTH regardless of the settings in the configuration pages of the tag. For the implementation of the adequate authentication scheme developer is responsible to use T2T\_NO\_PWD\_AUTH for access of the public data that are not protected by a pair of PWD, PACK.

## TRAILER BLOCK MANIPULATION COMMANDS

Special blocks called “trailer blocks” defines access bits and rights for Keys A and B for each sector. To read more, refer to NXP documentation about Mifare cards, see [http://www.nxp.com/documents/data\\_sheet/M001053\\_MF1ICS50\\_rev5\\_3.pdf](http://www.nxp.com/documents/data_sheet/M001053_MF1ICS50_rev5_3.pdf) and [http://www.nxp.com/documents/data\\_sheet/MF1S50YYX.pdf](http://www.nxp.com/documents/data_sheet/MF1S50YYX.pdf)

### SECTOR\_TRAILER\_WRITE (0x1A)

Function is used to write keys and access bits into the trailers of the sector. It could be used or sector address mode (without need for block\_in\_sector\_address to be sent because the given sector is always known) either the block address mode that determines the addressing\_mode u CMD\_EXT set parameter which can have the following values:

BLOCK\_ADDRESS\_MODE = 0

SECTOR\_ADDRESS\_MODE = 1

Access bits are sent separately as 4 bytes that has possible values 0 up to 7.

The device Firmware is formatting the access bits according to the cards specification irreversible blocking of that sector.

The CMD\_EXT set is used and its length depends on the authentication mode that is in use. CMD\_Par0 contains AUTH\_MODE.

Depending on AUTH\_MODE, CMD and CMD\_EXT set contains:

### RKA\_AUTH1x:

- CMD\_Par1 in CMD set contains readers index key
- 1<sup>st</sup> byte of the set contains sector\_(block\_)address
- 2<sup>nd</sup> byte of the set contains dummy value
- 3<sup>rd</sup> byte of the set contains addressing mode
- 4<sup>th</sup> byte contains 9-byte sector trailer value (anything could be written)

- in 5<sup>th</sup> to 10<sup>th</sup> byte of the set is an unencrypted key A for writing
- in 11<sup>th</sup> to 14<sup>th</sup> byte are the access bits values for 0 to 3 blocks inside the sector respectively (for Classic 4k cards also the second half of their address space – the rest 2K of space, 11<sup>th</sup> byte of CMD\_EXT set determines the access bits values for the blocks 0 to 4, the 12<sup>th</sup> byte for blocks 5 to 9 and the 13<sup>th</sup> byte for blocks 10 to 14 and at the end 14<sup>th</sup> byte for sector trailer)
- the 15<sup>th</sup> to 20<sup>th</sup> byte of the set contains an unencrypted key B for writing
- 21<sup>st</sup> byte contains checksum

#### **AKMy\_AUTH1x:**

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of the set contains sector\_(block\_)address
- 2<sup>nd</sup> byte of the set contains dummy value
- 3<sup>rd</sup> byte of the set contains addressing mode
- 4<sup>th</sup> byte contains 9-byte sector trailer value (anything could be written)
- in 5<sup>th</sup> to 10<sup>th</sup> byte of the set is an unencrypted key A for writing
- in 11<sup>th</sup> to 14<sup>th</sup> byte are the access bits values for 0 to 3 blocks inside the sector respectively (for Classic 4k cards also the second half of their address space – the rest 2K of space, 11<sup>th</sup> byte of CMD\_EXT set determines the access bits values for the blocks 0 to 4, the 12<sup>th</sup> byte for blocks 5 to 9 and the 13<sup>th</sup> byte for blocks 10 to 14 and at the end 14<sup>th</sup> byte for sector trailer)
- the 15<sup>th</sup> to 20<sup>th</sup> byte of the set contains an unencrypted key B for writing
- 21<sup>st</sup> byte contains checksum

#### **PK\_AUTH1x:**

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of the set contains sector\_(block\_)address
- 2<sup>nd</sup> byte of the set contains dummy value
- 3<sup>rd</sup> byte of the set contains addressing\_mode
- 4<sup>th</sup> byte contains 9-byte sector trailer value (anything could be written)
- array from 5<sup>th</sup> up to 10<sup>th</sup> byte contains 6-byte key.
- in 11<sup>th</sup> to 16<sup>th</sup> byte of the set is an unencrypted key A for writing
- in 17<sup>th</sup> to 20<sup>th</sup> byte are the access bits values for 0 to 3 blocks inside the sector respectively (for Classic 4k cards also the second half of their address space – the rest 2K of space, 11<sup>th</sup> byte of CMD\_EXT set determines the access bits values for the blocks 0 to 4, the 12<sup>th</sup> byte for blocks 5 to 9 and the 13<sup>th</sup> byte for blocks 10 to 14 and at the end 14<sup>th</sup> byte for sector trailer)
- the 21<sup>st</sup> do 26<sup>th</sup> byte of the set contains an unencrypted key B for writing
- 27<sup>th</sup> byte contains checksum

If everything is done as it should it returns the RESPONSE set.

RESPONSE\_EXT is not used.

#### **Example:**

authentication RKA key A, key number 0, sector address 0, addressing mode 1, key A = 0xFFFFFFFFFFFF, key B = 0xFFFFFFFFFFFF, access bits values 0, 0, 0, 1

```
CMD          55 1A AA 15 00 00 F7
ACK          AC 1A CA 15 00 00 70

CMD_EXT     00 00 01 69 FF FF FF FF FF FF 00 00 00 01 FF FF FF FF FF FF 70
RESP        DE 1A ED 00 00 00 30
```

Mifare Plus using.

For firmware versions from 5.0.29.

For RKA\_AUTH1x or AKMy\_AUTH1x or PK\_AUTH1x mode AES key for authentication, and new AES key A and key B, are calculate from Crypto1 keys. Commands uses in same manner as for Mifare Classic card.

### **MFP\_RKA\_AUTH1x:**

- CMD\_Par1 in CMD set contains readers index of AES keys ( 0 - 15)
- 1<sup>st</sup> byte of the set contains sector\_(block\_)address
- 2<sup>nd</sup> byte of the set contains dummy value
- 3<sup>rd</sup> byte of the set contains addressing mode
- 4<sup>th</sup> byte contains 9-byte sector trailer value (anything could be written)
- in 5<sup>th</sup> to 10<sup>th</sup> byte of the set is first 6 bytes of an unencrypted key A for writing
- in 11<sup>th</sup> to 14<sup>th</sup> byte are the access bits values for 0 to 3 blocks inside the sector respectively (for Classic 4k cards also the second half of their address space – the rest 2K of space, 11<sup>th</sup> byte of CMD\_EXT set determines the access bits values for the blocks 0 to 4, the 12<sup>th</sup> byte for blocks 5 to 9 and the 13<sup>th</sup> byte for blocks 10 to 14 and at the end 14<sup>th</sup> byte for sector trailer)
- the 15<sup>th</sup> to 20<sup>th</sup> byte of the set contains first 6 bytes of an unencrypted key B for writing
- the 21<sup>st</sup> to 30<sup>th</sup> byte of the set contains second 10 bytes of unencrypted key A for writing
- the 31<sup>st</sup> to 40<sup>th</sup> byte of the set contains second 10 bytes of unencrypted key B for writing
- 41<sup>st</sup> byte contains checksum

### **MFP\_AKMy\_AUTH1x:**

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of the set contains sector\_(block\_)address
- 2<sup>nd</sup> byte of the set contains dummy value
- 3<sup>rd</sup> byte of the set contains addressing mode
- 4<sup>th</sup> byte contains 9-byte sector trailer value (anything could be written)
- in 5<sup>th</sup> to 10<sup>th</sup> byte of the set is an unencrypted key A for writing
- in 11<sup>th</sup> to 14<sup>th</sup> byte are the access bits values for 0 to 3 blocks inside the sector respectively (for Classic 4k cards also the second half of their address space – the rest 2K of space, 11<sup>th</sup> byte of CMD\_EXT set determines the access bits values for the blocks 0 to 4, the 12<sup>th</sup> byte for blocks 5 to 9 and the 13<sup>th</sup> byte for blocks 10 to 14 and at the end 14<sup>th</sup> byte for sector trailer)
- the 15<sup>th</sup> to 20<sup>th</sup> byte of the set contains an unencrypted key B for writing
- the 21<sup>st</sup> to 30<sup>th</sup> byte of the set contains second 10 bytes of unencrypted key A for writing
- the 31<sup>st</sup> to 40<sup>th</sup> byte of the set contains second 10 bytes of unencrypted key B for writing

- 41<sup>st</sup> byte contains checksum

### **PK\_AUTH1x\_AES:**

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of the set contains sector\_(block\_)address
- 2<sup>nd</sup> byte of the set contains dummy value
- 3<sup>rd</sup> byte of the set contains addressing\_mode
- 4<sup>th</sup> byte contains 9-byte sector trailer value (anything could be written)
- array from 5<sup>th</sup> up to 20<sup>th</sup> byte contains 16-byte AES key.
- in 21<sup>st</sup> to 26<sup>th</sup> byte of the set is an unencrypted key A for writing
- in 27<sup>th</sup> to 30<sup>th</sup> byte are the access bits values for 0 to 3 blocks inside the sector respectively (for Classic 4k cards also the second half of their address space – the rest 2K of space, 11<sup>th</sup> byte of CMD\_EXT set determines the access bits values for the blocks 0 to 4, the 12<sup>th</sup> byte for blocks 5 to 9 and the 13<sup>th</sup> byte for blocks 10 to 14 and at the end 14<sup>th</sup> byte for sector trailer)
- the 31<sup>st</sup> do 36<sup>th</sup> byte of the set contains an unencrypted key B for writing
- the 37<sup>th</sup> to 46<sup>th</sup> byte of the set contains second 10 bytes of unencrypted key A for writing
- the 47<sup>th</sup> to 56<sup>th</sup> byte of the set contains second 10 bytes of unencrypted key B for writing
- 57<sup>th</sup> byte contains checksum

If everything is done as it should it returns the RESPONSE set.

RESPONSE\_EXT is not used.

### **SECTOR\_TRAILER\_WRITE\_UNSAFE (0x2F)**

It operates as SECTOR\_TRAILER\_WRITE except it send already formatted sector trailer block to be written without the access bits value check. The command is unsafe because it could lead to irreversible blocking of the entire sector of the card due to improperly formatted value of access bits. Made only for advanced users.

The CMD\_EXT set is used and its length depends on the authentication mode that is in use.

CMD\_Par0 contains AUTH\_MODE.

Depending on AUTH\_MODE, CMD and CMD\_EXT set contains:

### **RKA\_AUTH1x:**

- CMD\_Par1 u CMD set contains readers index key
- 1<sup>st</sup> byte of the set contains sector\_(block\_)address
- 2<sup>nd</sup> byte of the set contains dummy value
- 3<sup>rd</sup> byte of the set contains addressing\_mode
- 4<sup>th</sup> byte of the set contains dummy value
- in 5<sup>th</sup> to 20<sup>th</sup> byte of the set is the content of the sector trailer for writing
- 21<sup>st</sup> byte contains checksum

### AKMy\_AUTH1x:

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of the set contains sector\_(block\_)address
- 2<sup>nd</sup> byte of the set contains dummy value
- 3<sup>rd</sup> byte of the set contains addressing\_mode
- 4<sup>th</sup> byte of the set contains dummy value
- in 5<sup>th</sup> to 20<sup>th</sup> byte of the set is the content of the sector trailer for writing
- 21<sup>st</sup> byte contains checksum

### PK\_AUTH1x:

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of the set contains sector\_(block\_)address
- 2<sup>nd</sup> byte of the set contains dummy value
- 3<sup>rd</sup> byte of the set contains addressing\_mode
- 4<sup>th</sup> byte of the set contains dummy value
- array from 5<sup>th</sup> up to 10<sup>th</sup> bytes contains 6-byte key.
- in 11<sup>th</sup> to 26<sup>th</sup> byte of the set is the content of the sector trailer for writing
- 27<sup>th</sup> byte contains checksum

If everything is done as it should it returns the RESPONSE set.

RESPONSE\_EXT is not used.

### Example:

authentication RKA key A, key number 0, sector address 0, addressing mode 1, key A = 0xFFFFFFFFFFFFFFF, key B = 0xFFFFFFFFFFFFFFF, access bits values 0xFF078069 (default configuration)

```
CMD      55 2F AA 15 00 00 CC
ACK      AC 2F CA 15 00 00 63
```

```
CMD_EXT  00 00 01 00 FF FF FF FF FF FF FF 07 80 69 FF FF FF FF FF FF 17
RESP     DE 2F ED 00 00 00 23
```

## BLOCK MANIPULATION COMMANDS

Following commands used direct block addressing, meaning that blocks are indexed in range 0 to 63 for Mifare 1K cards.

### **BLOCK\_READ (0x16)**

Reads the whole data block from the card which is in the reader field.

The CMD\_EXT set is used and its length depends on authentication mode that is used.

CMD\_Par0 contains AUTH\_MODE.



Depending on AUTH\_MODE, CMD and CMD\_EXT set contains:

### RKA\_AUTH1x:

- CMD\_Par1 in CMD set contains key index in the reader
- 1<sup>st</sup> byte of CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains dummy data
- 5<sup>th</sup> byte contains checksum

### Example:

read block 01 with RKA\_AUTH1A

**CMD**            55 16 AA 05 00 00 F3

**ACK**            AC 16 CA 05 00 00 7C

**CMD\_EXT**       01 00 00 00 08

**RSP**            DE 16 ED 11 00 00 3B

**RSP\_EXT**       00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 07

### AKMy\_AUTH1x:

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains dummy data
- 5<sup>th</sup> byte contains checksum

### PK\_AUTH1x:

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains dummy data
- array from 5<sup>th</sup> to 10<sup>th</sup> byte contains 6-byte key.
- 11<sup>th</sup> byte contains checksum

If all operates as it should it turns the RESPONSE set and the RESPONSE\_EXT is following with 16 read bytes and checksum at the end.

### PK\_AUTH1x\_AES: (uFR PLUS devices only Mifare Plus tags)

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains dummy data
- array from 5<sup>th</sup> to 20<sup>th</sup> byte contains 16-byte AES key.
- 21<sup>st</sup> byte contains checksum

If all operates as it should it turns the RESPONSE set and the RESPONSE\_EXT is following with 16 read bytes and checksum at the end.



Mifare Plus using.

For firmware versions from 5.0.1 to 5.0.28 in RKA\_AUTH1x or AKMy\_AUTH1x mode uses AES key from reader AES keys space (index 0 - 15).

For firmware versions from 5.0.29 in RKA\_AUTH1x or AKMy\_AUTH1x mode uses AES key which calculated from reader Crypto1 key (index 0 - 31).

Firmware versions from 5.0.29

#### **MFP\_RKA\_AUTH1x:**

- CMD\_Par1 in CMD set contains AES key index in the reader (0 -15)
- 1<sup>st</sup> byte of CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains dummy data
- 5<sup>th</sup> byte contains checksum

#### **MFP\_AKMy\_AUTH1x:**

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains dummy data
- 5<sup>th</sup> byte contains checksum

#### **SAM\_KEY\_AUTH1x: (uFR CS with SAM and firmware versions 5.100.xx)**

- CMD\_Par1 in CMD set contains key index in the SAM (1 - 127)
- 1<sup>st</sup> byte of CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains dummy data
- 5<sup>th</sup> byte contains checksum

#### ***BLOCK\_WRITE (0x17)***

Writes the whole data block into the card that is currently in the readers field. Address mode is used for so called block addressing where for example the first block on Mifare Classic 1k has an address 0 and the last one has the address 63. This command doesn't allow the direct writing into the sector trailer and in the case of its addressing it gives back the FORBIDDEN\_DIRECT\_WRITE\_IN\_SECTOR\_TRAILER.

The CMD\_EXT set is used and its length depends on the authentication mode that is in use.

CMD\_Par0 contains AUTH\_MODE.

Depending on AUTH\_MODE, CMD and CMD\_EXT set contains:

#### **RKA\_AUTH1x:**

- CMD\_Par1 in CMD set contains readers index key
- 1<sup>st</sup> byte of CMD\_EXT set contains block\_address

- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains dummy data
- in 5<sup>th</sup> to 20<sup>th</sup> byte of set are placed data for writing into the data block
- 21<sup>st</sup> byte contains checksum

#### **AKMy\_AUTH1x:**

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains dummy data
- in 5<sup>th</sup> to 20<sup>th</sup> byte of the set are placed the data for writing into the data block
- 21<sup>st</sup> byte contains checksum

#### **PK\_AUTH1x:**

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte CMD\_EXT set contains dummy data
- array from 5<sup>th</sup> to 10<sup>th</sup> byte contains 6-byte key.
- in 11<sup>th</sup> too 26<sup>th</sup> byte are placed the data for writing into the data block
- 27<sup>th</sup> byte contains checksum.

#### **PK\_AUTH1x\_AES: (uFR PLUS devices only Mifare Plus tags)**

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte CMD\_EXT set contains dummy data
- array from 5<sup>th</sup> to 20<sup>th</sup> byte contains 16-byte AES key.
- in 21<sup>th</sup> too 36<sup>th</sup> byte are placed the data for writing into the data block
- 37<sup>th</sup> byte contains checksum.

Mifare Plus using.

For firmware versions from 5.0.1 to 5.0.28 in RKA\_AUTH1x or AKMy\_AUTH1x mode uses AES key from reader AES keys space (index 0 - 15).

For firmware versions from 5.0.29 in RKA\_AUTH1x or AKMy\_AUTH1x mode uses AES key which calculated from reader Crypto1 key (index 0 - 31).

Firmware versions from 5.0.29

#### **MFP\_RKA\_AUTH1x:**

- CMD\_Par1 in CMD set contains readers index key
- 1<sup>st</sup> byte of CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains dummy data
- in 5<sup>th</sup> to 20<sup>th</sup> byte of set are placed data for writing into the data block
- 21<sup>st</sup> byte contains checksum

#### **MFP\_AKMy\_AUTH1x:**

- CMD\_Par1 is not used.

- 1<sup>st</sup> byte of CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains dummy data
- in 5<sup>th</sup> to 20<sup>th</sup> byte of the set are placed the data for writing into the data block
- 21<sup>st</sup> byte contains checksum

### **SAM\_KEY\_AUTH1x: (uFR CS with SAM and firmware versions 5.100.xx)**

- CMD\_Par1 in CMD set contains key index in the SAM (0 - 127)
- 1<sup>st</sup> byte of CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains dummy data
- in 5<sup>th</sup> to 20<sup>th</sup> byte of set are placed data for writing into the data block
- 21<sup>st</sup> byte contains checksum

If everything is done as it should device answer with RSP packet.

#### **Example:**

write "01 02 03 04 05 06 07 08" into block 1 using key "FF FF FF FF FF FF"

**CMD**            55 17 AA 1B 60 00 9A

**ACK**            AC 17 CA 1B 60 00 11

**CMD\_EXT**    01 00 00 00 FF FF FF FF FF FF 01 02 03 04 05 06 07 08 00 00 00  
00 00

00 00 00 10

**RSP**            DE 17 ED 00 00 00 2B

### ***BLOCK\_IN\_SECTOR\_READ (0x18)***

It has the same function as the BLOCK\_READ but uses the different address mode for so called sector addressing where is always given the address of the sector and the sector block (as specified in the NXP documentation for Mifare Classic cards). The first sector of the Mifare Classic 1k card for example has the address 0 and the last one has 15. The block addresses of the sector are defined in the interval from 0 to 3 (3<sup>rd</sup> block of each sector is sector trailer) excluding Mifare Classic 4k cards for which in its second line of address space (the second 2k that is 32<sup>nd</sup> up to 39<sup>th</sup> sector) have the block addresses in sector 0 to 15 and the 15<sup>th</sup> is sector trailer.

Communication command protocol is the same as with BLOCK\_READ with following exception:

- 1<sup>st</sup> byte of the CMD\_EXT set contains block\_in\_sector\_address
- 2<sup>nd</sup> byte of the CMD\_EXT set contains sector\_address
- 3<sup>rd</sup> and 4<sup>th</sup> byte of the CMD\_EXT set contains dummy data

#### **Example:**

read block 0 in sector 0 with RKA\_AUTH1A, key number 0

```

CMD      55 18 AA 05 00 00 E9
ACK      AC 18 CA 05 00 00 82

```

```

CMD_EXT  00 00 00 00 07

```

```

RSP      DE 18 ED 11 00 00 41

```

```

RSP_EXT  47 8F 90 61 39 08 04 00 01 F1 0A F0 1A A2 EB 1D 4F

```

### **BLOCK\_IN\_SECTOR\_WRITE (0x19)**

Has the same function as the BLOCK\_WRITE but uses the different address mode, so called sector addressing where the sector address and the address of the block in the sector is always given (as mentioned in NXP documentation for Mifare Classic cards). For example the first sector on Mifare Classic 1k card has the address 0 and the last one has the address 15. The block addresses in sector are in the interval from 0 to 3 (3<sup>rd</sup> block of each sector is sector trailer) excluding Mifare Classic 4k cards for which in its second line of address space (the second 2k that is 32<sup>nd</sup> up to 39<sup>th</sup> sector) have the block addresses in sector 0 to 15 and the 15<sup>th</sup> is sector trailer. Communication command protocol is the same as with BLOCK\_WRITE with following exception:

- 1<sup>st</sup> byte of CMD\_EXT set contains block\_in\_sector\_address
- 2<sup>nd</sup> byte of CMD\_EXT set contains sector\_address
- 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains dummy data

#### **Example:**

write block 1 in sector 0 with RKA\_AUTH1A, key number 0

```

CMD      55 19 AA 15 00 00 FA

```

```

ACK      AC 19 CA 15 00 00 71

```

```

CMD_EXT  01 00 00 00 00 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF 17

```

```

RSP      DE 19 ED 00 00 00 31

```

## **LINEAR DATA MANIPULATION COMMANDS**

### **LINEAR\_READ (0x14)**

Linear read data from the card. This command concatenates data for successive blocks and sectors into one array of data. It performs something like “continuous reading” of data. It is very convenient for reading data from more blocks or sectors which are in successive order.

uFR PLUS only Mifare Plus tags support. In security level 3 for Mifare Plus tags, multi sector authentication can be used to optimize the performance and minimize the number of authentications. AES keys for sectors which contains blocks for linear read, must be equal. Then you can use a multi block read with authentication for first sector only.

The CMD\_EXT set is used whose length depends on the mode of authentication that is used. CMD\_Par0 contains AUTH\_MODE.

Depending on AUTH\_MODE, CMD and CMD\_EXT sets contains:

**RKA\_AUTH1x:**

- CMD\_Par1 in CMD set contains key index in the reader
- 1<sup>st</sup> and 2<sup>nd</sup> byte of CMD\_EXT set contains linear\_address (little endian)
- 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains data\_length (little endian)
- 5<sup>th</sup> byte contains checksum

**Example:**

Read linear data from 0 to 63, length is 64 bytes, using RK AUTH1A

```
CMD      55 14 AA 05 00 00 F5
ACK      AC 14 CA 05 00 00 7E

CMD_EXT  00 00 40 00 47
RSP      DE 14 ED 41 00 00 6D
```

and DATA we asked for in RSP\_EXT

```
31 32 33 34 35 36 37 38 39 30 00 00 00 00 00 31
32 33 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

With checksum

38

**AKMy\_AUTH1x:**

- CMD\_Par1 is not used.
- 1<sup>st</sup> and 2<sup>nd</sup> byte of CMD\_EXT set contains linear\_address (little endian)
- 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains data\_length (little endian)
- 5<sup>th</sup> byte contains checksum

**Example:** Read linear data from 0 to 31, length is 32 bytes, using AKM1 AUTH1A

```
CMD      55 14 AA 05 20 00 D5
ACK      AC 14 CA 05 20 00 5E

CMD_EXT  00 00 20 00 27
RSP      DE 14 ED 21 00 00 0D
```

and DATA we asked for in RSP\_EXT

```
31 32 33 34 35 36 37 38 39 30 00 00 00 00 00 31
32 33 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

With checksum

38

**Example:** Read linear data from 0 to 31, length is 32 bytes, using AKM1 AUTH1B

```
CMD      55 14 AA 05 21 00 D6
ACK      AC 14 CA 05 21 00 5D
```

```
CMD_EXT  00 00 20 00 27
RSP      DE 14 ED 21 00 00 0D
```

and DATA we asked for in RSP\_EXT

```
31 32 33 34 35 36 37 38 39 30 00 00 00 00 00 31
32 33 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

With checksum

38

Same applies to AKM2 AUTHA and AUTHB commands.

### PK\_AUTH1x:

- CMD\_Par1 is not used.
- 1<sup>st</sup> and 2<sup>nd</sup> byte of CMD\_EXT set contains linear\_address (little endian)
- 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains data\_length (little endian)
- array from 5<sup>th</sup> to 10<sup>th</sup> byte contains 6-byte key.
- 11<sup>th</sup> byte contains checksum.

**Example:** Read linear data from 16 to 31, length is 16 bytes, using PK AUTH1B and provided key 6 x FF

```
CMD      55 14 AA 0B 61 00 88
ACK      AC 14 CA 0B 61 00 1F
```

```
CMD_EXT  10 00 10 00 FF FF FF FF FF FF 07
RSP      DE 14 ED 11 00 00 3D
```

and DATA we asked for in RSP\_EXT

```
32 33 00 00 00 00 00 00
00 00 00 00 00 00 00 00
```

with checksum

08

### **SAM\_KEY\_AUTH1x: (uFR CS with SAM and firmware versions 5.100.xx)**

- CMD\_Par1 in CMD set contains key index in the SAM (1 - 127)
- 1<sup>st</sup> and 2<sup>nd</sup> byte of CMD\_EXT set contains linear\_address (little endian)
- 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains data\_length (little endian)
- 5<sup>th</sup> byte contains checksum

If everything operates as expected the RSP packet is sent and after that also the RSP\_EXT with number of bytes according to the data\_length command with checksum at the end.

In case the card is removed from the field or in case of wrong authentication including that some block is read anyway, it turns ERR set with NO\_CARD error code or AUTH\_ERROR and then the ERR\_EXT set which contains the array of the read bytes and CHECKSUM at the end.

LINEAR\_READ command utilise FAST\_READ ISO 14443-3 command with NTAG21x and Mifare Ultralight EV1 tags.

uFR PLUS devices only. Mifare Plus tags. Firmware versions from 5.0.1 to 5.0.28

### **RKA\_AUTH1x:**

- CMD\_Par1 in CMD set contains AES key index in the reader (0 - 15)
- 1<sup>st</sup> and 2<sup>nd</sup> byte of CMD\_EXT set contains linear\_address (little endian)
- 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains data\_length to 192 bytes (little endian)
- 5<sup>th</sup> and 6<sup>th</sup> byte of CMD\_EXT set contains true data length if data length bigger than 192 bytes (little endian)
- 7<sup>th</sup> byte contains checksum
- For reasons of compatibility there is expected Error packet with Error code MFP\_MULTI\_BLOCKS\_READ = 0xB9
- Reading the data is specific and is done in a loop. Reads one data, and if it is 0, then reads another that indicates how much data follows in the package. This is repeated until the required amount of data read. If the first data is different from 0, then loop stops.
- RSP\_EXT not in use

### **PK\_AUTH1x\_AES:**

- CMD\_Par1 is not used.
- 1<sup>st</sup> and 2<sup>nd</sup> byte of CMD\_EXT set contains linear\_address (little endian)
- 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains data\_length (little endian)
- array from 5<sup>th</sup> to 20<sup>th</sup> byte contains 16-byte key.
- 21<sup>st</sup> byte contains checksum.
- For reasons of compatibility there is expected Error packet with Error code MFP\_MULTI\_BLOCKS\_READ = 0xB9

- Reading the data is specific and is done in a loop. Reads one data, and if it is 0, then reads another that indicates how much data follows in the package. This is repeated until the required amount of data read. If the first data is different from 0, then loop stops.
- RSP\_EXT not in use

**Example:**

Read linear data from 0 - 299, length = 300. AES key is 16 x 0xFF

```

CMD      55 14 AA 17 81 00 84
ACK      AC 14 CA 17 81 00 EB
CMD_EXT  00 00 B8 00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
            2C 01 9C
    
```

```

ERR      EC B9 CE 00 FF FF A2
    
```

```

DATA     00 30 41 53 43 49 20 74 65 78 74 20 72 65 61 64 69 6E
            67 20 74 65 73 74 00 00 00 00 00 00 00 00 00 00
            00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

            00 30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
            00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
            00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

            00 30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
            00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
            00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

            00 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
            00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

            00 30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
            00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
            00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

            00 1C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    
```



00 00 00 00 00 00 00 00 00 00 00 00 00

DD (OK)

RSP DE 14 ED 00 00 00 2E

### **SAM\_KEY\_AUTH1x: (uFR CS with SAM and firmware versions 5.100.xx)**

- CMD\_Par1 in CMD set contains key index in the
- 1<sup>st</sup> and 2<sup>nd</sup> byte of CMD\_EXT set contains linear\_address (little endian)
- 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains data\_length to 192 bytes (little endian)
- 5<sup>th</sup> and 6<sup>th</sup> byte of CMD\_EXT set contains true data length if data length bigger than 192 bytes (little endian)
- 7<sup>th</sup> byte contains checksum
- For reasons of compatibility there is expected Error packet with Error code  
MFP\_MULTI\_BLOCKS\_READ = 0xB9
- Reading the data is specific and is done in a loop. Reads one data, and if it is 0, then reads another that indicates how much data follows in the package. This is repeated until the required amount of data read. If the first data is different from 0, then loop stops.
- RSP\_EXT not in use

For firmware versions from 5.0.29

In RKA\_AUTH1x or AKMy\_AUTH1x mode, commands are used in the same manner as for Mifare Classic card. AES key calculated from Crypto1 reader key (index 0 - 31).

### **MFP\_RKA\_AUTH1x:**

- CMD\_Par1 in CMD set contains AES key index in the reader (0 - 15)
- 1<sup>st</sup> and 2<sup>nd</sup> byte of CMD\_EXT set contains linear\_address (little endian)
- 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains data\_length to 192 bytes (little endian)
- 5<sup>th</sup> and 6<sup>th</sup> byte of CMD\_EXT set contains true data length if data length bigger than 192 bytes (little endian)
- 7<sup>th</sup> byte contains checksum
- For reasons of compatibility there is expected Error packet with Error code  
MFP\_MULTI\_BLOCKS\_READ = 0xB9
- Reading the data is specific and is done in a loop. Reads one data, and if it is 0, then reads another that indicates how much data follows in the package. This is repeated until the required amount of data read. If the first data is different from 0, then loop stops.
- RSP\_EXT not in use

### **MFP\_AKMy\_AUTH1x:**

- CMD\_Par1 is not used.
- 1<sup>st</sup> and 2<sup>nd</sup> byte of CMD\_EXT set contains linear\_address (little endian)
- 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains data\_length (little endian)

- 5<sup>th</sup> byte contains checksum

### **LINEAR\_WRITE (0x15)**

Linear data writing into the card which is currently in the field of the reader. The verification of each written block is done during the writing.

The CMD\_EXT set is used and its length depends on the authentication mode that is used

CMD\_Par0 contains AUTH\_MODE.

Depending on AUTH\_MODE, CMD and CMD\_EXT sets contains:

#### **RKA\_AUTH1x:**

- CMD\_Par1 in CMD set contains key index in the reader
- 1<sup>st</sup> and 2<sup>nd</sup> byte of CMD\_EXT set contains linear\_address (little endian)
- 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains data\_length (little endian)
- from 5<sup>th</sup> byte up (data\_length + 4) contains data array for writing
- (data\_length + 5) byte contains checksum

**Example:** Write 8 bytes into card string at linear address 08, using RK\_AUTH1A, bytes are 10 11...17

```
CMD      55 15 AA 0D 00 00 EE
ACK      AC 15 CA 0D 00 00 85
```

```
CMD_EXT  08 00 08 00 10 11 12 13 14 15 16 17 07
RSP      DE 15 ED 00 00 00 2D
```

We can check now if bytes are written using previous examples of LinearRead command.

#### **AKMy\_AUTH1x:**

- CMD\_Par1 is not used.
- 1<sup>st</sup> and 2<sup>nd</sup> byte of CMD\_EXT set contains linear\_address (little endian)
- 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains data\_length (little endian)
- from 5<sup>th</sup> byte up (data\_length + 4) contains data array for writing
- (data\_length + 5) byte contains checksum

#### **PK\_AUTH1x:**

- CMD\_Par1 is not used.
- 1<sup>st</sup> and 2<sup>nd</sup> byte of CMD\_EXT set contains linear\_address (little endian)
- 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains data\_length (little endian)

- array from 5<sup>th</sup> to 10<sup>th</sup> byte contains 6- byte key
- 11<sup>th</sup> byte and up to (data\_length + 10) contains data array for writing
- (data\_length + 11) byte contains checksum.

uFR PLUS devices only. Mifare Plus tags. Firmware versions from 5.0.1 to 5.0.28.

#### **PK\_AUTH1x\_AES:**

- CMD\_Par1 is not used.
- 1<sup>st</sup> and 2<sup>nd</sup> byte of CMD\_EXT set contains linear\_address (little endian)
- 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains data\_length (little endian)
- array from 5<sup>th</sup> to 20<sup>th</sup> byte contains 16- byte key
- 21<sup>st</sup> byte and up to (data\_length + 20) contains data array for writing
- (data\_length + 21) byte contains checksum.

#### **SAM\_KEY\_AUTH1x: (uFR CS with SAM and firmware versions 5.100.xx)**

- CMD\_Par1 in CMD set contains key index in the SAM
- 1<sup>st</sup> and 2<sup>nd</sup> byte of CMD\_EXT set contains linear\_address (little endian)
- 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains data\_length (little endian)
- from 5<sup>th</sup> byte up (data\_length + 4) contains data array for writing
- (data\_length + 5) byte contains checksum

If everything went as expected device answer with RSP packet.

In error case it turns the ERR packet where the RSP\_Val0 contains the number of eventual written bytes.

For firmware versions from 5.0.29

In RKA\_AUTH1x or AKMy\_AUTH1x mode, commands are used in the same manner as for Mifare Classic card. AES key calculated from Crypto1 reader key (index 0 - 31).

#### **MFP\_RKA\_AUTH1x:**

- CMD\_Par1 in CMD set contains AES key index in the reader
- 1<sup>st</sup> and 2<sup>nd</sup> byte of CMD\_EXT set contains linear\_address (little endian)
- 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains data\_length (little endian)
- from 5<sup>th</sup> byte up (data\_length + 4) contains data array for writing
- (data\_length + 5) byte contains checksum

#### **MFP\_AKMy\_AUTH1x:**

- CMD\_Par1 is not used.
- 1<sup>st</sup> and 2<sup>nd</sup> byte of CMD\_EXT set contains linear\_address (little endian)
- 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains data\_length (little endian)
- from 5<sup>th</sup> byte up (data\_length + 4) contains data array for writing

- (data\_length + 5) byte contains checksum

### **LINEAR\_FORMAT\_CARD (0x25)**

The CMD\_EXT set is used and its length depends on the authentication mode that is used. Since this command can erase data or block card reading if wrong access bits are provided, we strongly suggest to test it first through SDK API examples to figure out what this command does. For pure erasing data or filling card with 0x00 without changing the keys, it is much easier to use Linear\_Write command.

#### **Usage:**

CMD\_Par0 contains AUTH\_MODE.

Depending on AUTH\_MODE, CMD and CMD\_EXT set contains:

#### **RKA\_AUTH1x:**

- CMD\_Par1 in CMD set contains readers index key
- 1<sup>st</sup> byte of the set contains access bits value for blocks in sector
- 2<sup>nd</sup> byte of the set contains access bits value for sector trailers
- 3<sup>rd</sup> byte of the set contains dummy value
- 4<sup>th</sup> byte of the set has 9-byte sector trailer value (anything could be written)
- in 5<sup>th</sup> to 10<sup>th</sup> byte of the set is new key A
- in 11<sup>th</sup> to 16<sup>th</sup> byte of the set is new key B
- 17<sup>th</sup> byte contains checksum

#### **AKMy\_AUTH1x:**

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of the set contains access bits value for blocks in sector
- 2<sup>nd</sup> byte of the set contains access bits value for sector trailers
- 3<sup>rd</sup> byte of the set contains dummy value
- 4<sup>th</sup> byte of the set has 9-byte sector trailer value (anything could be written)
- in 5<sup>th</sup> to 10<sup>th</sup> byte of the set is new key A
- in 11<sup>th</sup> to 16<sup>th</sup> byte of the set is new key B
- 17<sup>th</sup> byte contains checksum

#### **PK\_AUTH1x:**

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of the set contains access bits value for blocks in sector
- 2<sup>nd</sup> byte of the set contains access bits value for sector trailers
- 3<sup>rd</sup> byte of the set contains dummy value
- 4<sup>th</sup> byte of the set has 9-byte sector trailer value (anything could be written)
- array from 5<sup>th</sup> up to 10<sup>th</sup> byte contains 6-byte key for authentication (previous)
- in 11<sup>th</sup> to 16<sup>th</sup> byte of the set is new key A
- in 17<sup>th</sup> to 22<sup>nd</sup> byte of the set is new key B

- 23<sup>rd</sup> byte contains checksum

If everything is done as it should device answer with RSP packet.

RSP\_EXT is not used.

**Example:**

Key A is 0xFFFFFFFFFFFF, Key B is 0xFFFFFFFFFFFF, access bits value for blocks is 0, access bits value for sector trailers is 1, authentication mode is RKA\_AUTH1A, key number is 0

```

CMD          55 25 AA 11 00 00 D2
ACK          AC 25 CA 11 00 00 59
CMD_EXT     00 01 00 69 FF FF FF FF FF FF FF FF FF FF FF 6F
RSP         DE 25 ED 00 10 00 0D
    
```

Mifare Plus using.

Firmware versions from 5.0.29.

In RKA\_AUTH1x or AKMy\_AUTH1x or PK\_AUTH1x mode, commands are used in the same manner as for Mifare Classic card. AES key for authentication calculated from Crypto1 reader key (index 0 - 31) or provided Crypto1 key. New AES key A and key B are calculate from provided Crypto1 keys. 4K card formatting is about 10 seconds, so it is periodically sent keep alive frame, before response frame.

```

CMD          55 25 AA 11 00 00 D2
ACK          AC 25 CA 11 00 00 59
CMD_EXT     00 01 00 69 FF FF FF FF FF FF FF FF FF FF FF 6F
KEEP_ALIVE A1 25 85 00 00 00 08
...
KEEP_ALIVE A1 25 85 00 00 00 08
RSP         DE 25 ED 00 10 00 0D
    
```

**MFP\_RKA\_AUTH1x:**

- CMD\_Par1 in CMD set contains readers index key
- 1<sup>st</sup> byte of the set contains access bits value for blocks in sector
- 2<sup>nd</sup> byte of the set contains access bits value for sector trailers
- 3<sup>rd</sup> byte of the set contains dummy value
- 4<sup>th</sup> byte of the set has 9-byte sector trailer value (anything could be written)
- in 5<sup>th</sup> to 10<sup>th</sup> byte of the set are first 6 bytes of new AES key A
- in 11<sup>th</sup> to 16<sup>th</sup> byte of the set are first 6 bytes of new AES key B
- in 17th to 26th bytes of the set are last 10 bytes of new AES key A
- in 27th to 36th bytes of the set are last 10 bytes of new AES key B
- 37<sup>th</sup> byte contains checksum

**MFP\_AKMy\_AUTH1x:**

- CMD\_Par1 is not used.

- 1<sup>st</sup> byte of the set contains access bits value for blocks in sector
- 2<sup>nd</sup> byte of the set contains access bits value for sector trailers
- 3<sup>rd</sup> byte of the set contains dummy value
- 4<sup>th</sup> byte of the set has 9-byte sector trailer value (anything could be written)
- in 5<sup>th</sup> to 10<sup>th</sup> byte of the set are first 6 bytes of new AES key A
- in 11<sup>th</sup> to 16<sup>th</sup> byte of the set are first 6 bytes of new AES key B
- in 17<sup>th</sup> to 26<sup>th</sup> bytes of the set are last 10 bytes of new AES key A
- in 27<sup>th</sup> to 36<sup>th</sup> bytes of the set are last 10 bytes of new AES key B
- 37<sup>th</sup> byte contains checksum

#### **PK\_AUTH1x\_AES:**

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of the set contains access bits value for blocks in sector
- 2<sup>nd</sup> byte of the set contains access bits value for sector trailers
- 3<sup>rd</sup> byte of the set contains dummy value
- 4<sup>th</sup> byte of the set has 9-byte sector trailer value (anything could be written)
- array from 5<sup>th</sup> up to 20<sup>th</sup> byte contains 16-byte AES key for authentication (previous)
- in 21<sup>th</sup> to 26<sup>th</sup> byte of the set are first 6 bytes of new AES key A
- in 27<sup>th</sup> to 32<sup>nd</sup> byte of the set are first 6 bytes of new AES key B
- in 33<sup>rd</sup> to 42<sup>nd</sup> bytes of the set are last 10 bytes of new AES key A
- in 43<sup>rd</sup> to 52<sup>nd</sup> bytes of the set are last 10 bytes of new AES key B
- 53<sup>rd</sup> byte contains checksum

#### ***LIN\_ROW\_READ(0x45)***

Functions allow you to quickly read data from the card including the sector trailer blocks. These functions are very similar to the functions for linear reading of users data space. Using this command is the same as using the command `LINEAR_READ(0x14)`

The `CMD_EXT` set is used whose length depends on the mode of authentication that is used. `CMD_Par0` contains `AUTH_MODE`.

Depending on `AUTH_MODE`, `CMD` and `CMD_EXT` sets contains:

#### **RKA\_AUTH1x:**

- `CMD_Par1` in `CMD` set contains key index in the
- 1<sup>st</sup> and 2<sup>nd</sup> byte of `CMD_EXT` set contains `linear_address` (little endian)
- 3<sup>rd</sup> and 4<sup>th</sup> byte of `CMD_EXT` set contains `data_length` (little endian)
- 5<sup>th</sup> byte contains checksum

#### **AKMy\_AUTH1x:**

- `CMD_Par1` is not used.
- 1<sup>st</sup> and 2<sup>nd</sup> byte of `CMD_EXT` set contains `linear_address` (little endian)

- 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains data\_length (little endian)
- 5<sup>th</sup> byte contains checksum

#### PK\_AUTH1x:

- CMD\_Par1 is not used.
- 1<sup>st</sup> and 2<sup>nd</sup> byte of CMD\_EXT set contains linear\_address (little endian)
- 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains data\_length (little endian)
- array from 5<sup>th</sup> do 10<sup>th</sup> byte contains 6-byte key.
- 11<sup>th</sup> byte contains checksum.

#### Example:

Read data from 0 to 47, length is 48 bytes, using RK AUTH1A key number 0

```

CMD          55 45 AA 05 00 00 C6
ACK          AC 45 CA 05 00 00 2D
CMD_EXT      00 00 30 00 37
RSP          DE 45 ED 31 00 00 4E
RSP_EXT      47 8F 90 61 39 08 04 00 01 F1 0A F0 1A A2 EB 1D 00 00 00 00 00
00 FF
              07 80 69 FF FF FF FF FF FF 00 00 00 00 00 00 FF 07 80 69 FF FF
FF FF
              FF FF 4F
    
```

## VALUE BLOCK MANIPULATION COMMANDS

From firmware version 5.0.36. Mifare Plus X, SE or EV1 value block manipulation support.

### DIRECT BLOCK ADDRESSING

#### VALUE\_BLOCK\_READ (0x1D)

Reads the 4-byte value of the “value block” of the card which is currently in the reading field. Address mode that is used is so called block addressing where for example the first block of Mifare Classic 1k card has the address 0 and the last one has the address 63.

The CMD\_EXT set is used and its length depends on the authentication mode that is used. CMD\_Par0 contains AUTH\_MODE.

Depending on AUTH\_MODE, CMD and CMD\_EXT set contains:

#### RKA\_AUTH1x:

- CMD\_Par1 in CMD set contains readers index key
- 1<sup>st</sup> byte of the CMD\_EXT set contains block\_address

- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte of the CMD\_EXT set contains dummy data
- 5<sup>th</sup> byte contains checksum

#### **AKMy\_AUTH1x:**

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of the CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte of the CMD\_EXT set contains dummy data
- 5<sup>th</sup> byte contains checksum

#### **PK\_AUTH1x:**

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of the CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte of the CMD\_EXT set contains dummy data
- array from 5<sup>th</sup> to 10<sup>th</sup> byte contains 6-byte key.
- 11<sup>th</sup> byte contains checksum

#### **SAM\_KEY\_AUTH1x: (uFR CS with SAM and firmware versions 5.100.xx)**

- CMD\_Par1 in CMD set contains key index in the SAM
- 1<sup>st</sup> byte of the CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte of the CMD\_EXT set contains dummy data
- 5<sup>th</sup> byte contains checksum

#### **Mifare Plus using. Firmware version from 5.0.36**

#### **PK\_AUTH1x\_AES: (FR PLUS devices only Mifare Plus tags)**

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains dummy data
- array from 5<sup>th</sup> to 20<sup>th</sup> byte contains 16-byte AES key.
- 21<sup>st</sup> byte contains checksum

For firmware versions from 5.0.1 to 5.0.28 in RKA\_AUTH1x or AKMy\_AUTH1x mode uses AES key from reader AES keys space (index 0 - 15).

For firmware versions from 5.0.29 in RKA\_AUTH1x or AKMy\_AUTH1x mode uses AES key which calculated from reader Crypto1 key (index 0 - 31).

Firmware versions from 5.0.29

#### **MFP\_RKA\_AUTH1x:**

- CMD\_Par1 in CMD set contains AES key index in the reader (0 -15)
- 1<sup>st</sup> byte of CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains dummy data
- 5<sup>th</sup> byte contains checksum



### **MFP\_AKMy\_AUTH1x:**

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte of CMD\_EXT set contains dummy data
- 5th byte contains checksum

If everything is OK, device answer with RSP packet followed by RSP\_EXT containing 4-byte value and checksum.

RSP\_Val0 contains block address (read from block value for powerful backup as mentioned in the Mifare card documentation).

In the case of error the VALUE\_BLOCK\_ADDR\_INVALID (read value of the value block is formatted properly but the address bytes aren't) it returns ERR\_EXT set which contains the value of the value block.

Notice that value is in little-endian notation, where negative values are stored as "Two complement's".

### **Example:**

Read Value Block 05 with PK\_AUTH1A:

```
CMD          55 1D AA 0B 60 00 90
ACK          AC 1D CA 0B 60 00 17

CMD_EXT      05 00 00 00 FF FF FF FF FF FF 0C
RSP          DE 1D ED 05 00 00 32
RSP_EXT      00 00 00 00 07
```

### **VALUE\_BLOCK\_WRITE (0x1E)**

Store 4-byte value into "value block".

This command disallow the writing into the trailers of the sector and in case of their addressing it returns the FORBIDEN\_DIRECT\_WRITE\_IN\_SECTOR\_TRAILER.

The CMD\_EXT set is used and its length depends on the authentication mode that is used.

CMD\_Par0 contains AUTH\_MODE.

Depending on AUTH\_MODE, CMD and CMD\_EXT set contains:

### **RKA\_AUTH1x:**

- CMD\_Par1 in CMD set contains readers index key
- 1<sup>st</sup> byte of the CMD\_EXT set contains block\_address

- 2<sup>nd</sup> and 3<sup>rd</sup> byte of the CMD\_EXT set contains dummy data
- 4<sup>th</sup> byte contains value address
- in 5<sup>th</sup> to 8<sup>th</sup> byte of the set is placed the data for writing into the value block
- 9<sup>th</sup> byte contains checksum

#### **AKMy\_AUTH1x:**

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of the CMD\_EXT set contains block\_address
- 2<sup>nd</sup> and 3<sup>rd</sup> byte of the CMD\_EXT set contains dummy data
- 4<sup>th</sup> byte contains value address
- in 5<sup>th</sup> to 8<sup>th</sup> byte of the set is placed the data for writing into the value block
- 9<sup>th</sup> byte contains checksum

#### **PK\_AUTH1x:**

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of the CMD\_EXT set contains block\_address
- 2<sup>nd</sup> and 3<sup>rd</sup> byte of the CMD\_EXT set contains dummy data
- 4<sup>th</sup> byte contains value address
- array from 5<sup>th</sup> up to 10<sup>th</sup> byte contains 6-byte key.
- in 11<sup>th</sup> to 14<sup>th</sup> byte of the set is placed the data for writing into the value block
- 15<sup>th</sup> byte contains checksum

Example: Store value 01 01 01 01 into block 5 using PK\_AUTH1A key FF FF FF FF FF FF

```
CMD      55 1E AA 0F 60 00 95
ACK      AC 1E CA 0F 60 00 1E
```

```
CMD_EXT  05 00 00 05 FF FF FF FF FF FF 01 01 01 01 07
RSP      DE 1E ED 00 00 00 34 DE
```

#### **SAM\_KEY\_AUTH1x: (uFR CS with SAM and firmware versions 5.100.xx)**

- CMD\_Par1 in CMD set contains key index in the SAM
- 1<sup>st</sup> byte of the CMD\_EXT set contains block\_address
- 2<sup>nd</sup> and 3<sup>rd</sup> byte of the CMD\_EXT set contains dummy data
- 4<sup>th</sup> byte contains value address
- in 5<sup>th</sup> to 8<sup>th</sup> byte of the set is placed the data for writing into the value block
- 9<sup>th</sup> byte contains checksum

#### **Mifare Plus using. Firmware version from 5.0.36**

#### **PK\_AUTH1x\_AES: (FR PLUS devices only Mifare Plus tags)**

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of the CMD\_EXT set contains block\_address
- 2<sup>nd</sup> and 3<sup>rd</sup> byte of the CMD\_EXT set contains dummy data

- 4<sup>th</sup> byte contains value address
- array from 5<sup>th</sup> up to 20<sup>th</sup> byte contains 16-byte key.
- in 21<sup>st</sup> to 24<sup>th</sup> byte of the set is placed the data for writing into the value block
- 25<sup>th</sup> byte contains checksum

For firmware versions from 5.0.1 to 5.0.28 in RKA\_AUTH1x or AKMy\_AUTH1x mode uses AES key from reader AES keys space (index 0 - 15).

For firmware versions from 5.0.29 in RKA\_AUTH1x or AKMy\_AUTH1x mode uses AES key which calculated from reader Crypto1 key (index 0 - 31).

Firmware versions from 5.0.29

#### **MFP\_RKA\_AUTH1x:**

- CMD\_Par1 in CMD set contains AES key index in the reader (0 -15)
- 1<sup>st</sup> byte of the CMD\_EXT set contains block\_address
- 2<sup>nd</sup> and 3<sup>rd</sup> byte of the CMD\_EXT set contains dummy data
- 4<sup>th</sup> byte contains value address
- in 5<sup>th</sup> to 8<sup>th</sup> byte of the set is placed the data for writing into the value block
- 9<sup>th</sup> byte contains checksum

#### **MFP\_AKMy\_AUTH1x:**

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of the CMD\_EXT set contains block\_address
- 2<sup>nd</sup> and 3<sup>rd</sup> byte of the CMD\_EXT set contains dummy data
- 4<sup>th</sup> byte contains value address
- in 5<sup>th</sup> to 8<sup>th</sup> byte of the set is placed the data for writing into the value block
- 9<sup>th</sup> byte contains checksum

If everything is OK, device answer with RSP packet. RSP\_EXT is not used.

Notice that value is in little-endian notation, where negative values are stored as “Two complement's”. For example, decimal value 65535 should be stored as FF FF 00 00.

#### **VALUE\_BLOCK\_INC (0x21)**

It increases the value of the addressed value block for the 4-byte value increment\_val that is send as a command parameter and is been used for so-called block address mode.

The CMD\_EXT set is used and its length depends on the authentication mode that is used.

CMD\_Par0 contains AUTH\_MODE.

Depending on AUTH\_MODE, CMD and CMD\_EXT set contains:

### **RKA\_AUTH1x:**

- CMD\_Par1 in CMD set contains readers index key
- 1<sup>st</sup> byte of the CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte of the CMD\_EXT set contains dummy data
- in 5<sup>th</sup> to 8<sup>th</sup> byte set is increment\_val
- 9<sup>th</sup> byte contains checksum

### **AKMy\_AUTH1x:**

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of the CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte of the CMD\_EXT set contains dummy data
- in 5<sup>th</sup> to 8<sup>th</sup> byte set is increment\_val
- 9<sup>th</sup> byte contains checksum

### **PK\_AUTH1x:**

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of the CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte of the CMD\_EXT set contains dummy data
- array from 5<sup>th</sup> up to 10<sup>th</sup> byte contains 6-byte key
- in 11<sup>th</sup> to 14<sup>th</sup> bytes of the set is increment\_val
- 15<sup>th</sup> byte contains checksum.

### **SAM\_KEY\_AUTH1x: (uFR CS with SAM and firmware versions 5.100.xx)**

- CMD\_Par1 in CMD set contains key index into SAM
- 1<sup>st</sup> byte of the CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte of the CMD\_EXT set contains dummy data
- in 5<sup>th</sup> to 8<sup>th</sup> byte set is increment\_val
- 9<sup>th</sup> byte contains checksum

### **Mifare Plus using. Firmware version from 5.0.36**

#### **PK\_AUTH1x\_AES: (FR PLUS devices only Mifare Plus tags)**

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of the CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte of the CMD\_EXT set contains dummy data
- array from 5<sup>th</sup> up to 20<sup>th</sup> byte contains 16-byte key.
- in 21<sup>st</sup> to 24<sup>th</sup> byte of the set is increment\_val
- 25<sup>th</sup> byte contains checksum

For firmware versions from 5.0.1 to 5.0.28 in RKA\_AUTH1x or AKMy\_AUTH1x mode uses AES key from reader AES keys space (index 0 - 15).

For firmware versions from 5.0.29 in RKA\_AUTH1x or AKMy\_AUTH1x mode uses AES key which calculated from reader Crypto1 key (index 0 - 31).

Firmware versions from 5.0.29

### **MFP\_RKA\_AUTH1x:**

- CMD\_Par1 in CMD set contains AES key index in the reader (0 -15)
- 1<sup>st</sup> byte of the CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte of the CMD\_EXT set contains dummy data
- in 5<sup>th</sup> to 8th byte of the set is increment\_val
- 9<sup>th</sup> byte contains checksum

### **MFP\_AKMy\_AUTH1x:**

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of the CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte of the CMD\_EXT set contains dummy data
- in 5<sup>th</sup> to 8th byte of the set is increment\_val
- 9<sup>th</sup> byte contains checksum

If everything is OK, device answer with RSP packet. RSP\_EXT packet is not used.

### **Example:**

Increase Value Block 5 with "F0 F0 F0 F0" using PK\_AUTH1A with key FF FF FF FF FF FF

```
CMD          55 21 AA 0F 60 00 B8
ACK          AC 21 CA 0F 60 00 2F
```

```
CMD_EXT     05 00 00 00 FF FF FF FF FF FF F0 F0 F0 F0 0C
RSP         DE 21 ED 00 00 00 19 DE
```

Notice that when we read now Value Block 5 we will get

RSP and RSP\_EXT DE 1D ED 05 05 00 35 F1 F1 F1 71 87,

with value F1 F1 F1 71, stored in little-endian notation, where byte 71 is represented in Two Complement's manner (change of sign +/-).

### **VALUE\_BLOCK\_DEC (0x22)**

Decrement the value of the addressed value block for 4-byte value decrement\_val which is sent as the command parameter. The so-called block address mode is used.

The CMD\_EXT set is used and the length of the authentication mode is used.

CMD\_Par0 contains AUTH\_MODE.

Depending on AUTH\_MODE, CMD and CMD\_EXT set contains:

**RKA\_AUTH1x:**

- CMD\_Par1 in CMD set contains readers index key
- 1<sup>st</sup> byte of the CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte CMD\_EXT set contains dummy data
- in 5<sup>th</sup> to 8<sup>th</sup> byte of the set is decrement\_val
- 9<sup>th</sup> byte contains checksum

**AKMy\_AUTH1x:**

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of the CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte CMD\_EXT set contains dummy data
- in 5<sup>th</sup> to 8<sup>th</sup> byte of the set is decrement\_val
- 9<sup>th</sup> byte contains checksum

**PK\_AUTH1x:**

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of the CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte of the CMD\_EXT set contains dummy data
- array from 5<sup>th</sup> up to 10<sup>th</sup> byte contains 6-byte key.
- in 11<sup>th</sup> to 14<sup>th</sup> byte of the set is decrement\_val
- 15<sup>th</sup> byte contains checksum.

**SAM\_KEY\_AUTH1x: (uFR CS with SAM and firmware versions 5.100.xx)**

- CMD\_Par1 in CMD set contains key index into SAM (1 - 127)
- 1<sup>st</sup> byte of the CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte CMD\_EXT set contains dummy data
- in 5<sup>th</sup> to 8<sup>th</sup> byte of the set is decrement\_val
- 9<sup>th</sup> byte contains checksum

**Mifare Plus using. Firmware version from 5.0.36**

**PK\_AUTH1x\_AES: (FR PLUS devices only Mifare Plus tags)**

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of the CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte of the CMD\_EXT set contains dummy data
- array from 5<sup>th</sup> up to 20<sup>th</sup> byte contains 16-byte key.
- in 21<sup>st</sup> to 24<sup>th</sup> byte of the set is decrement\_val
- 25<sup>th</sup> byte contains checksum

For firmware versions from 5.0.1 to 5.0.28 in RKA\_AUTH1x or AKMy\_AUTH1x mode uses AES key from reader AES keys space (index 0 - 15).

For firmware versions from 5.0.29 in RKA\_AUTH1x or AKMy\_AUTH1x mode uses AES key which calculated from reader Crypto1 key (indec 0 - 31).

Firmware versions from 5.0.29

#### **MFP\_RKA\_AUTH1x:**

- CMD\_Par1 in CMD set contains AES key index in the reader (0 -15)
- 1<sup>st</sup> byte of the CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte of the CMD\_EXT set contains dummy data
- in 5<sup>th</sup> to 8th byte of the set is decrement\_val
- 9<sup>th</sup> byte contains checksum

#### **MFP\_AKMy\_AUTH1x:**

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of the CMD\_EXT set contains block\_address
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte of the CMD\_EXT set contains dummy data
- in 5<sup>th</sup> to 8th byte of the set is decrement\_val
- 9<sup>th</sup> byte contains checksum

If everything is OK, device answer with RSP packet. RSP\_EXT packet is not used

#### **Example:**

Decrement Value Block 5 with 00 00 00 F0 using PK\_AUTH1A with key FF FF FF FF FF FF

```
CMD          55 22 AA 0F 60 00 B9
ACK          AC 22 CA 0F 60 00 32

CMD_EXT      05 00 00 00 FF FF FF FF FF FF 00 00 00 F0 FC
RSP          DE 22 ED 00 00 00 18
```

Notice that when we read now Value Block 5 we will get

```
RSP and RSP_EXT  DE 1D ED 05 05 00 35  F1 F1 F1 01 F7
```

with value F1 F1 F1 01, stored in little-endian notation, where byte 01 is represented in Two Complement's manner (change of sign +/-).

## **INDIRECT BLOCK ADDRESSING**

### **VALUE\_BLOCK\_IN\_SECTOR\_READ (0x1F)**

It operates as VALUE\_BLOCK\_READ but uses the different address mode, so-called sector addressing where are always given the sector address and the block address in the sector (as mentioned in NXP documentation for Mifare Classic cards).

For example the first sector of the Mifare Classic 1k card has the 0 and the last one has the address 15. Block addresses in the sector are in the interval from 0 to 3 (3<sup>rd</sup> block of each sector is sector trailer) excluding Mifare Classic 4k cards for which in its second half of address space

(second 2k with 32 to 39 sector) the addresses of the blocks in sector 0 to 15 and the block 15 is sector trailer.

Communication command protocol is the same as with VALUE\_BLOCK\_READ with following exception:

- 1<sup>st</sup> byte of the CMD\_EXT set contains block\_in\_sector\_address
- 2<sup>nd</sup> byte of the CMD\_EXT set contains sector\_address
- 3<sup>rd</sup> and 4<sup>th</sup> byte of the CMD\_EXT set contains dummy data.

Device will answer with RSP and RSP\_EXT. RSP\_Val0 contains direct block address.

**Example:**

Read Value Block 01 in Sector 01 (is equal to Value Block 5 using direct addressing) using PK\_AUTH1A mode with key FF FF FF FF FF FF

**CMD**            55 1F AA 0B 60 00 92  
**ACK**            AC 1F CA 0B 60 00 19

**CMD\_EXT**      01 01 00 00 FF FF FF FF FF FF 07  
**RSP**            DE 1F ED 05 05 00 33  
**RSP\_EXT**      F1 F1 F1 01 F7

**VALUE\_BLOCK\_IN\_SECTOR\_WRITE (0x20)**

It operates as VALUE\_BLOCK\_WRITE but uses different address mode, so-called sector addressing where are always given the sector address and the block address in the sector (as mentioned in NXP documentation for Mifare Classic cards). For example the first sector of the Mifare Classic 1k card has the 0 and the last one has the address 15. Block addresses in the sector are in the interval from 0 to 3 (3<sup>rd</sup> block of each sector is sector trailer) excluding Mifare Classic 4k cards for which in its second half of address space (second 2k with 32 to 39 sector) the addresses of the blocks in sector 0 to 15 and the block 15 is sector trailer.

Communication command protocol is the same as with VALUE\_BLOCK\_IN\_SECTOR\_READ with following exception:

- 1<sup>st</sup> byte of the CMD\_EXT set contains block\_in\_sector\_address
- 2<sup>nd</sup> byte of the CMD\_EXT set contains sector\_address
- 3<sup>rd</sup> and 4<sup>th</sup> byte of the CMD\_EXT set contains dummy data

**Example:**

Write Value Block 00 in Sector 01 (is equal to Value Block 5 using direct addressing) value “80 80 80 80” using PK\_AUTH1A mode with key FF FF FF FF FF FF



```

CMD      55 20 AA 0F 60 00 B7
ACK      AC 20 CA 0F 60 00 30

```

```

CMD_EXT 01 01 00 00 FF FF FF FF FF FF 80 80 80 80 07
RSP      DE 20 ED 00 00 00 1A

```

### **VALUE\_BLOCK\_IN\_SECTOR\_INC (0x23)**

It operates as VALUE\_BLOCK\_IN\_SECTOR\_INC but uses the different address mode, so-called sector addressing where are always given the sector address and the block address in the sector (as mentioned in NXP documentation for Mifare Classic cards). For example the first sector of the Mifare Classic 1k card has the 0 and the last one has the address 15. Block addresses in the sector are in the interval from 0 to 3 (3<sup>rd</sup> block of each sector is sector trailer) excluding Mifare Classic 4k cards for which in its second half of address space (second 2k with 32 to 39 sector) the addresses of the blocks in sector 0 to 15 and the block 15 is sector trailer.

Communication command protocol is the same as with VALUE\_BLOCK\_INC with following exception:

- 1<sup>st</sup> byte of the CMD\_EXT set contains block\_in\_sector\_address
- 2<sup>nd</sup> byte of the CMD\_EXT set contains sector\_address
- 3<sup>rd</sup> and 4<sup>th</sup> byte of the CMD\_EXT set contains dummy data.

#### **Example:**

```

CMD      55 23 AA 0F 60 00 BA
ACK      AC 23 CA 0F 60 00 31

```

```

CMD_EXT 01 01 00 00 FF FF FF FF FF FF 60 60 60 60 07
RSP      DE 23 ED 00 00 00 17

```

### **VALUE\_BLOCK\_IN\_SECTOR\_DEC (0x24)**

It operates as VALUE\_BLOCK\_IN\_SECTOR\_DEC but uses different address mode, so-called sector addressing where are always given the sector address and the block address in the sector (as mentioned in NXP documentation for Mifare Classic cards). For example the first sector of the Mifare Classic 1k card has the 0 and the last one has the address 15. Block addresses in the sector are in the interval from 0 to 3 (3<sup>rd</sup> block of each sector is sector trailer) excluding Mifare Classic 4k cards for which in its second half of address space (second 2k with 32 to 39 sector) the addresses of the blocks in sector 0 to 15 and the block 15 is sector trailer.

Communication command protocol is the same as with VALUE\_BLOCK\_DEC with following exception:

- 1<sup>st</sup> byte of the CMD\_EXT set contains block\_in\_sector\_address
- 2<sup>nd</sup> byte of the CMD\_EXT set contains sector\_address
- 3<sup>rd</sup> and 4<sup>th</sup> byte of the CMD\_EXT set contains dummy data

**Example:**

```
CMD      55 24 AA 0F 60 00 BB
ACK      AC 24 CA 0F 60 00 34
```

```
CMD_EXT  01 01 00 00 FF FF FF FF FF FF 60 60 60 60 07
RSP      DE 24 ED 00 00 00 1E
```

**Commands for NFC Type 2 Tags*****GET\_NFC\_T2T\_VERSION (0xB0)*****supported from firmware version 3.8.19**

This command returns 8 bytes of the T2T version. All modern T2T chips support this functionality and have in common a total of 8 byte long version response. This function is primarily intended to use with NFC\_T2T\_GENERIC tags (i.e. tags for which command GET\_DLOGIC\_CARD\_TYPE returns 0x0C in RSP\_Val0).

```
CMD_Par0 not in use.
CMD_Par1 not in use.
CMD_EXT not in use.
```

On success:

```
RSP_Val0 not in use.
RSP_Val1 not in use.
```

RSP\_EXT will contain 8 bytes of the T2T version. For exact meaning of this version bytes, you have to consult the card manufacturer's documentation.

If card in field doesn't have originality checking support, returned error code is:

```
UNSUPPORTED_CARD_TYPE (0x11)
```

**Example:**

```
CMD      55 B0 AA 00 AA CC 30
RSP      DE B0 ED 09 00 00 91
RSP_EXT  00 04 04 02 01 00 13 03 1A
```

**Commands supporting NFC T2T Counters*****READ\_COUNTER (0xB1)*****supported from firmware version 3.9.11**

This function is used to read one of the three 24-bit one-way counters in Ultralight EV1 chip family or to read 24-bit NFC counter in NTAG 213, NTAG 215 and NTAG 216 chips.

Counters in the Ultralight EV1 can't be password protected. NFC counters in NTAG 213, NTAG

215 and NTAG 216 chips can be password protected.

CMD\_Par0 contains AUTH\_MODE.

AUTH\_MODE using with this function can be:

T2T\_NO\_PWD\_AUTH (0x00) {same constant value as RKA\_AUTH1A}

T2T\_RKA\_PWD\_AUTH (0x01) {same constant value as RKA\_AUTH1B}

T2T\_PK\_PWD\_AUTH (0x61) {same constant value as PK\_AUTH1B}

Depending on **AUTH\_MODE**, CMD and CMD\_EXT set contains:

#### **T2T\_NO\_PWD\_AUTH:**

- CMD\_Par1 contains **counter address** (For Ultralight EV1: 0, 1 or 2. For NTAG21x: 0).
- CMD\_EXT not in use.

#### **T2T\_RKA\_PWD\_AUTH:**

- CMD\_Par1 in CMD set contains readers index key.
- CMD\_EXT not in use.

#### **T2T\_PK\_PWD\_AUTH:**

- CMD\_Par1 is not used.
- 1<sup>st</sup> byte of CMD\_EXT set contains block\_address.
- 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> byte CMD\_EXT set contains dummy data.
- array from 5<sup>th</sup> to 8<sup>th</sup> byte contains 4-byte T2T password.
- 9<sup>th</sup> and 10<sup>th</sup> byte of CMD\_EXT set contains 2-byte PAK (password acknowledge).
- 11<sup>th</sup> byte contains checksum.

If you issue this command without using password authentication but access to the NFC counter is configured to be password protected, this function will return COUNTER\_ERROR.

If access to NFC counter is configured to be password protected and PWD-PACK pair sent as a 6-byte provided key disagrees with PWD-PACK pair configured in tag, this function will return UFR\_AUTH\_ERROR. If access to NFC counter isn't configured to be password protected, this function will return UFR\_AUTH\_ERROR.

#### **Example:**

```
CMD      55 B1 AA 00 00 01 56
RSP      DE B1 ED 05 00 00 8E
RSP_EXT  07 00 00 00 0E
```

### ***INCREMENT\_COUNTER (0xB2)***

#### **supported from firmware version 3.9.11**

This command is used to increment one of the three 24-bit one-way counters in Ultralight EV1 chip family. Those counters can't be password protected. If the sum of the addressed counter value and the increment value is higher than 0xFFFFFFFF, the tag replies with an error and does not update the respective counter.

CMD\_Par0 not in use.

CMD\_Par1 contains counter address (0, 1 or 2).

CMD\_EXT contains 4-byte increment value in little endian format, only the 3 least significant bytes are relevant.

RSP\_EXP not in use.

**Example:**

CMD	55 B2 AA 05 00 01 50
ACK	AC B2 CA 05 00 01 D7
CMD_EXT	04 00 00 00 0B
RSP	DE B2 ED 00 00 00 88

## **COMMANDS FOR “ASYNCHRONOUS UID SENDING” FEATURE**

This feature “Async UID sending” is capability of reader device to send Card UID immediately when card enters into device RF field, without any action initiated by host. This is also exception from rule that communication is always initiated by host to device. Feature can be turned on and off. Baudrate for this feature is different than baudrate of device, e.g. it can be different. Prefix and suffix are bytes that are used to diversify UID's, like header and trailer bytes of UID.

Device can send UID encapsulated in [Prefix] and [Suffix] when card enters into RF field.

Device can also send “empty UID” when card leaves RF field, meaning only [Prefix][Suffix] will be sent.

Best practice is to set Baud rate different than device communication speed, anything bigger than 9600 Bps to avoid collision with standard communication between device and host.

### **SET\_CARD\_ID\_SEND\_CONF (0x3D)**

Set the asynchronously card ID sending parameters.

CMD\_Par0 contains send enable flag (bit 0), prefix enable flag (bit 1) and send removed enable flag (bit2).

When using option Send removed flag, Prefix byte is mandatory

1<sup>st</sup> byte of the CMD\_EXT contains prefix character

2<sup>nd</sup> byte of the CMD\_EXT contains suffix character

array from 3<sup>rd</sup> byte up to 6<sup>th</sup> byte of the CMD\_EXT contains baud rate value

7<sup>th</sup> byte of the CMD\_EXT contains internal CRC (xor of bytes CMD\_Par0 to 6<sup>th</sup> byte + 7)

8<sup>th</sup> byte of the CMD\_EXT contains checksum

If everything is OK, device answer with RSP packet. RSP\_EXT is not used.

**Example:**

**CMD**            55 3D AA 08 07 00 D4 (send command 3D, bits 0,1,2 high), D4  
**checksum**

**ACK**            AC 3D CA 08 07 00 5B (ACK OK)

**CMD\_EXT**      CC EE 80 25 00 00 87 07 (prefix CC, suffix EE, speed 9600  
(0x2580) ,

(87 checksum -

07,00,CC,EE,80,25,00,00) ,

(07 - checksum of CMD\_EXT)

**RSP**            DE 3D ED 00 00 00 15            (RESPONSE OK) speed 9600 (0x2580) ,

When card enter the field, event will occur:

<b>HEX</b>	CC	30	34	32	32	43	33	36	32	34	42	32	44	38	31	EE
<b>ASCII</b>	?	0	4	2	2	C	3	6	2	4	B	2	D	8	1	?

meaning card UID is 04 22 C3 62 4B 2D 81

On card removal, event will occur:

**CC EE**

To disable feature, send bits 0,1,2 low:

```
CMD  55 3D AA 00 00 00 C9
RSP  DE 3D ED 00 00 00 15
```

### **GET\_CARD\_ID\_SEND\_CONF (0x3E)**

Get the asynchronously card ID sending parameters.

The CMD\_EXT set is not in use.

The CMD\_Par0 and CMD\_Par1 are not in use.

If everything is OK, device answer with RSP packet and after that also the RSP\_EXT packet of 9 bytes.

RSP\_Val0 and RSP\_Val1 are not in use.

1<sup>st</sup> byte of the RESPONSE\_EXT contains send enable flag (bit 0), prefix enable flag (bit 1) and send removed enable flag (bit2).

2<sup>nd</sup> byte of the RESPONSE\_EXT contains prefix character

3<sup>rd</sup> byte of the RESPONSE\_EXT contains suffix character

array from 4<sup>th</sup> byte up to 7<sup>th</sup> byte of the RESPONSE\_EXT contains baud rate value

8<sup>th</sup> byte of the RESPONSE\_EXT contains internal CRC

9<sup>th</sup> byte of the RESPONSE\_EXT contains checksum

### **Example:**

```
CMD      55 3E AA 00 00 00 C8 (send CMD 3E, C8 checksum)
RSP      DE 3E ED 09 00 00 0B (RSP command 3E, 9 byte follows, 0B
checksum)
RSP_EXT  07 CC EE 80 25 00 00 87 0E (07 -bits 0,1,2 high, CC Prefix, EE
suffix,
                                speed 9600 (0x2580),
                                87 - checksum (
07,CC,EE,80,25,00,00),
                                0E - checksum of RSP_EXT)
```

## **COMMANDS FOR WORKS WITH DESFIRE CARDS**

For uFR CS with SAM and firmware versions 5.100.xx all types of keys into SAM support added.

For uFR PLUS devices and firmware version from 5.0.25 DES, 2K3DES and 3K3DES key support added.

enum KEY\_TYPE

```
{
    AES_KEY_TYPE = 0, //AES key KEY_LENGTH = 16 bytes
    DES3K_KEY_TYPE = 1, //3K3DES key KEY_LENGTH = 24 bytes
```

```

DES_KEY_TYPE = 2, //DES key  KEY_LENGTH = 8 bytes
DES2K_KEY_TYPE = 3 //2K3DES key KEY_LENGTH = 16 bytes
};

```

### **DESFIRE\_WRITE\_AES\_KEY (0x8E)**

Command writes AES key into reader.

(Old firmwares and AES key)

CMD\_Par0 and CMD\_Par1 are 0

1<sup>st</sup> byte of the CMD\_EXT contains ordinal number of AES key into reader

array from 2<sup>nd</sup> byte up to 17<sup>th</sup> byte of the CMD\_EXT contains AES key

18<sup>th</sup> byte of the CMD\_EXT contains checksum

(Firmware version from 5.0.25)

CMD\_Par0 = KEY\_TYPE and CMD\_Par1 = 0

1<sup>st</sup> byte of the CMD\_EXT contains ordinal number of key into reader

array from byte 2 to byte (1 + KEY\_LENGTH) of CMD\_EXT contains key

byte (2 + KEY\_LENGTH) contains checksum

(For 3K3DES key 2 fields into reader will be occupied. For example, if key stored into field 0, then field 1 also used for this key, first free field is 2)

Device answer with RSP packet.

RSP\_EXT

1st byte is 0

2nd byte is error code look at [Appendix: ERROR CODES](#)

3rd byte is checksum

#### **Example:**

AES key is 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF, and ordinal number is 3

```

CMD      55 8E AA 12 00 00 6A (send command 8E) , 6A checksum
ACK      AC 8E CA 12 00 00 01 (ACK OK)
CMD_EXT  03 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 0A
RSP      DE 8E ED 03 00 00 C5
RSP_EXT  00 00 07

```

### **GET\_DESFIRE\_UID (0x80)**

Command returns Unique ID of card, if the Random ID is used.

From firmware version 5.0.32 Desfire Light tag support

(Old firmwares and AES key)

CMD\_Par0 and CMD\_Par1 are 0

1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal AES key, or 0 if uses external AES key

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal AES key, or 0 if uses external AES key

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains AES key  
array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)  
22<sup>nd</sup> byte contains ordinal key number into application  
23<sup>rd</sup> byte contains checksum

(Firmware version from 5.0.25)

CMD\_Par0 = (KEY\_TYPE << 4) and CMD\_Par1 = 0

1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal key, or 0 if uses external key

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal key, or 0 if uses external key

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains key (for AES and 2K3DES all key bytes, for DES 8 key bytes and 8 zeros, for 3K3DES first 16 key bytes)

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte contains ordinal key number into application

(for AES, DES and 2K3DES) 23<sup>rd</sup> byte contains checksum

(for 3K3DES) array from byte 23 to byte 30 contains last 8 key bytes, and byte 31 contains

checksum

(uFR CS with SAM and firmware versions 5.100.xx)

CMD\_Par0 = (KEY\_TYPE << 4) and CMD\_Par1 = 0

1<sup>st</sup> byte of the CMD\_EXT is 2 (using key into SAM)

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of key into SAM (1 - 127)

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains 16 zeros

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte contains ordinal key number into application

23<sup>rd</sup> byte contains checksum

Response:

If no error, i.e. error code is CARD\_OPERATION\_OK, device answer with RSP packet and after that also the RSP\_EXT packet of 12 bytes.

RSP\_Val0 and RSP\_Val1 are not in use.

array from 1<sup>st</sup> to 7<sup>th</sup> byte of RSP\_EXT contains 7 bytes length card UID

8<sup>th</sup> and 9<sup>th</sup> bytes represents card's error code of operation (b9 \* 256 + b8), look at [Appendix: ERROR CODES for DESFire card operations](#)

10<sup>th</sup> and 11<sup>th</sup> bytes represents execution time of command

12<sup>th</sup> byte is checksum.

If error code is READER\_ERROR or NO\_CARD\_DETECTED, device answer with RSP\_EXT packet of 3 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents execution time of command

3<sup>rd</sup> byte is checksum.

In other cases, device answer with RSP\_EXT packet of 5 bytes.



1<sup>st</sup> and 2<sup>nd</sup> bytes represents card's error code of operation ( $b_2 * 256 + b_1$ ), look at [Appendix: ERROR CODES for DESFire card operations](#)

3<sup>rd</sup> and 4<sup>th</sup> bytes represents execution time of command

5<sup>th</sup> byte is checksum.

**Example:**

Authentication using the internal key ordinal number 3, AID = 0xF00001, ordinal key number into application is 1.

CMD 55 80 AA 17 00 00 6F

(send command 80), 6F checksum

ACK AC 80 CA 17 00 00 F8

(ACK OK)

CMD\_EXT 01 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 F0 01 F9

(internal key uses so AES key bytes may have any value (all 00), F9 checksum)

RSP DE 80 ED 0C 00 00 AC

(RSP command 80, 12 bytes follows, 0B checksum)

RSP\_EXT 04 01 02 03 05 06 07 B9 0B 0A 00 BF

(UID is 04010203050607, error code is 0BB9, execution time is 000A, checksum is BF)

**DESFIRE\_FREE\_MEM (0x8D)**

Command returns the available bytes on the card

The CMD\_EXT set is not in use.

The CMD\_Par0 and CMD\_Par1 are not in use.

If no error, i.e. error code is CARD\_OPERATION\_OK, device answer with RSP packet and after that also the RSP\_EXT packet of 9 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents error code of operation ( $b_2 * 256 + b_1$ ), look at [Appendix: ERROR CODES for DESFire card operations](#)

3<sup>rd</sup> and 4<sup>th</sup> bytes represents execution time of command

array from 5<sup>th</sup> to 8<sup>th</sup> of RSP\_EXT contains quantity of available bytes on card

9<sup>th</sup> byte is checksum

**Example:**

CMD 55 8D AA 00 00 00 79

RSP DE 8D ED 09 00 00 BE

RSP\_EXT B9 0B 0A 00 E8 03 00 00 5A

(error code 0BB9, execution time 000A, free mem 000003E8 i.e. 1000)

## DESFIRE\_FORMAT\_CARD(0x8C)

Function releases all allocated user memory on the card. All applications will be deleted, also all files within those applications will be deleted. Only the card master key, and card master key settings will not be deleted. This operation requires authentication with the card master key.

(Old firmwares and AES key)

CMD\_Par0 and CMD\_Par1 are 0

1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal AES key, or 0 if uses external AES key

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal AES key, or 0 if uses external AES key

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains AES key

19<sup>th</sup> byte is checksum

(Firmware version from 5.0.25)

CMD\_Par0 = (KEY\_TYPE << 4) and CMD\_Par1 = 0

1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal key, or 0 if uses external key

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal key, or 0 if uses external key

array from byte 3 to byte (2 + KEY\_LENGTH) contains key

byte 3 + KEY\_LENGTH is checksum

(uFR CS with SAM and firmware versions 5.100.xx)

CMD\_Par0 = (KEY\_TYPE << 4) and CMD\_Par1 = 0

1<sup>st</sup> byte of the CMD\_EXT is 2 (using key into SAM)

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of key into SAM (0 -127)

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains 16 zeros

19<sup>th</sup> byte is checksum

If error code is READER\_ERROR or NO\_CARD\_DETECTED, device answer with RSP\_EXT packet of 3 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents execution time of command

3<sup>rd</sup> byte is checksum.

In other cases, device answer with RSP\_EXT packet of 5 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents error code of operation (b2 \* 256 + b1)

3<sup>rd</sup> and 4<sup>th</sup> bytes represents execution time of command

5<sup>th</sup> byte is checksum.

### Example:

Authentication using the internal key ordinal number 1



- 1<sup>st</sup> byte of the CMD\_EXT is 2 (using key into SAM)
- 2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of key into SAM (0 -127)
- array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains 16 zeros
- 19<sup>th</sup> byte is 1 if Random ID enabled or 0 if Random ID disabled
- 20<sup>th</sup> byte is 1 if format card disabled or 0 if format card enabled
- 21<sup>st</sup> byte is checksum

If error code is `READER_ERROR` or `NO_CARD_DETECTED`, device answer with `RSP_EXT` packet of 3 bytes.

- 1<sup>st</sup> and 2<sup>nd</sup> bytes represents execution time of command
- 3<sup>rd</sup> byte is checksum.

In other cases, device answer with `RSP_EXT` packet of 5 bytes.

- 1<sup>st</sup> and 2<sup>nd</sup> bytes represents error code of operation ( $b2 * 256 + b1$ )
- 3<sup>rd</sup> and 4<sup>th</sup> bytes represents execution time of command
- 5<sup>th</sup> byte is checksum.

### Example:

Authentication using the internal key ordinal number 1, Random ID enabled, format card disabled

<b>CMD</b>	<code>55 8B AA 15 00 00 68</code>	(send command 8B), 68
<b>checksum</b>		
<b>ACK</b>	<code>AC 8B CA 15 00 00 FF</code>	(ACK OK)
<b>CMD_EXT</b>	<code>01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 08</code>	(internal key uses so AES key bytes may have any value (all 00), Random ID 01, format card 00, 08 checksum)
<b>RSP</b>	<code>DE 8B ED 05 00 00 C4</code>	(RSP command 8B, 5 byte follows, BD checksum)
<b>RSP_EXT</b>	<code>B9 0B 1A 00 AF</code>	(error code 0BB9, execution time 001A)

### **DESFIRE\_GET\_KEY\_CONFIG(0x87)**

Function allows to get card master key and application master key configuration settings. In addition it returns the maximum number of keys which can be stored within selected application. (Old firmwares and AES key)

CMD\_Par0 and CMD\_Par1 are 0

1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal AES key, or 0 if uses external AES key  
2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal AES key, or 0 if uses external AES key  
array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains AES key  
array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)  
22<sup>nd</sup> byte contains checksum.

(Firmware version from 5.0.25)

CMD\_Par0 = (KEY\_TYPE << 4) and CMD\_Par1 = 0

1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal key, or 0 if uses external key  
2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal key, or 0 if uses external key  
array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains key (for AES and 2K3DES all key bytes, for DES 8 key bytes and 8 zeros, for 3K3DES first 16 key bytes)  
array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)  
(for AES, DES and 2K3DES) 22<sup>nd</sup> byte contains checksum  
(for 3K3DES) array from byte 22 to byte 29 contains last 8 key bytes, and byte 30 contains

checksum

(uFR CS with SAM and firmware versions 5.100.xx)

CMD\_Par0 = (KEY\_TYPE << 4) and CMD\_Par1 = 0

1<sup>st</sup> byte of the CMD\_EXT is 2 (using key into SAM)  
2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of key into SAM (0 -127)  
array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains 16 zeros  
array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)  
22<sup>nd</sup> byte contains checksum.

If no error, i.e. error code is CARD\_OPERATION\_OK, device answer with RSP packet and after that also the RSP\_EXT packet of 7 bytes.

RSP\_Val0 and RSP\_Val1 are not in use.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents error code of operation ( $b2 * 256 + b1$ )  
3<sup>rd</sup> and 4<sup>th</sup> bytes represents execution time of command  
5<sup>th</sup> byte is key settings  
6<sup>th</sup> byte is maximum number of keys within selected application.  
7<sup>th</sup> byte is checksum

If error code is READER\_ERROR or NO\_CARD\_DETECTED, device answer with RSP\_EXT packet of 3 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents execution time of command  
3<sup>rd</sup> byte is checksum.

In other cases, device answer with RSP\_EXT packet of 5 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents error code of operation ( $b2 * 256 + b1$ )  
3<sup>rd</sup> and 4<sup>th</sup> bytes represents execution time of command  
5<sup>th</sup> byte is checksum.

**Example:**

Authentication using the internal key ordinal number 2, AID = 0xF00001

```
CMD      55 87 AA 16 00 00 75 (send command 87), 75 checksum
ACK      AC 87 CA 16 00 00 FE (ACK OK)
```

```
CMD_EXT  01 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 F0
CA(internal key uses so AES key bytes may have any value (all 00), CA
checksum)
```

```
RSP      DE 87 ED 07 00 00 BA          (RSP command 87, 7 bytes
follows, BA checksum)
```

```
RSP_EXT  B9 0B 1A 00 09 03 A9          (error code 0BB9, execution time
001A, key settings 9, maximum number of key 3)
```

**DESFIRE\_CHANGE\_KEY\_CONFIG(0x88)**

Function allows to set card master key, and application master key configuration settings.  
(Old firmwares and AES key)

CMD\_Par0 and CMD\_Par1 are 0

1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal AES key, or 0 if uses external AES key

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal AES key, or 0 if uses external AES key

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains AES key

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is key settings

23<sup>rd</sup> byte contains checksum.

(Firmware version from 5.0.25)

CMD\_Par0 = (KEY\_TYPE << 4) and CMD\_Par1 = 0

1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal key, or 0 if uses external key

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal key, or 0 if uses external key

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains key (for AES and 2K3DES all key bytes, for DES 8 key bytes and 8 zeros, for 3K3DES first 16 key bytes)

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is key settings

(for AES, DES and 2K3DES) 23<sup>rd</sup> byte contains checksum

(for 3K3DES) array from byte 23 to byte 30 contains last 8 key bytes, and byte 31 contains checksum

(uFR CS with SAM and firmware versions 5.100.xx)

CMD\_Par0 = (KEY\_TYPE << 4) and CMD\_Par1 = 0

1<sup>st</sup> byte of the CMD\_EXT is 2 (using key into SAM)

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of key into SAM (0 -127)  
array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains 16 zeros  
array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)  
22<sup>nd</sup> byte is key settings  
23<sup>rd</sup> byte contains checksum.

RSP\_Val0 and RSP\_Val1 are not in use.

If error code is READER\_ERROR or NO\_CARD\_DETECTED, device answer with RSP\_EXT packet of 3 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents execution time of command  
3<sup>rd</sup> byte is checksum.

In other cases, device answer with RSP\_EXT packet of 5 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents error code of operation (b2 \* 256 + b1)  
3<sup>rd</sup> and 4<sup>th</sup> bytes represents execution time of command  
5<sup>th</sup> byte is checksum.

### Example:

Authentication using the internal key ordinal number 2, AID = 0xF00001, key settings is 9

```
CMD      55 88 AA 17 00 00 67 (send command 88) , 67 checksum
ACK      AC 88 CA 17 00 00 00 (ACK OK)
```

```
CMD_EXT  01 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 F0
09 02 (internal key uses so AES key bytes may have any value (all 00) , 02
checksum)
```

```
RSP      DE 88 ED 05 00 00 C6          (RSP command 88, 5 bytes
follows, C5 checksum)
```

```
RSP_EXT  B9 0B 1A 00 AF (error code 0BB9, execution time 001A)
```

### **DESFIRE\_CHANGE\_AES\_KEY(0x86)**

Function allow to change any AES key on the card. Changing the card master key require current card master key authentication. Authentication for the application keys changing depend on the application master key settings (which key uses for authentication).

From firmware version 5.0.32 Desfire Light tag support

(Old firmwares and AES key)

CMD\_Par0 and CMD\_Par1 are 0



1<sup>st</sup> byte of the CMD\_EXT bit 0 set if uses internal AES key for authentication, bit 1 set if internal AES key uses as new key, bit 2 set if internal AES key uses as old key, high nibble is ordinal number of internal AES key which uses as old key, if they uses.

2<sup>nd</sup> byte of the CMD\_EXT low nibble is ordinal number of internal AES key which uses for authentication or 0 if uses external AES key, high nibble is ordinal number of internal AES key which uses as new key of 0 if uses external AES key

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains AES key for authentication

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is key number into application which uses for authentication

array from 23<sup>rd</sup> to 38<sup>th</sup> byte of CMD\_EXT contains new AES key

38<sup>th</sup> byte is key number into application that will be changed

array from 39<sup>th</sup> to 54<sup>th</sup> byte of CMD\_EXT contains new AES key

55<sup>th</sup> byte contains checksum.

(Firmware version from 5.0.25)

CMD\_Par0 = AUTH\_KEY\_TYPE | (NEW\_KEY\_TYPE << 2) and CMD\_Par1 = 0

1<sup>st</sup> byte of the CMD\_EXT bit 0 set if uses internal key for authentication, bit 1 set if internal key uses as new key, bit 2 set if internal key uses as old key, high nibble is ordinal number of internal key which uses as old key, if they uses.

2<sup>nd</sup> byte of the CMD\_EXT low nibble is ordinal number of internal key which uses for authentication or 0 if uses external key, high nibble is ordinal number of internal key which uses as new key of 0 if uses external key

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains key for authentication (for AES and 2K3DES all key bytes, for DES 8 key bytes and 8 zeros, for 3K3DES first 16 key bytes)

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is key number into application which uses for authentication

array from 23<sup>rd</sup> to 38<sup>th</sup> byte of CMD\_EXT contains new key (for AES and 2K3DES all key bytes, for DES 8 key bytes and 8 zeros, for 3K3DES first 16 key bytes)

38<sup>th</sup> byte is key number into application that will be changed

array from 39<sup>th</sup> to 54<sup>th</sup> byte of CMD\_EXT contains new key (for AES and 2K3DES all key bytes, for DES 8 key bytes and 8 zeros, for 3K3DES first 16 key bytes)

(for AES, DES and 2K3DES) 55<sup>th</sup> byte contains checksum.

(for 3K3DES as authentication key) array from byte 55 to byte 62 contains last 8 key bytes of authentication key

(for 3K3DES as new key) array from byte 63 to byte 70 contains last 8 key bytes of new key

(for 3K3DES as new key) array from byte 71 to byte 78 contains last 8 key bytes of old key

(for 3K3DES as authentication and new key) byte 79 is checksum

(for 3K3DES as authentication key and not new key) byte 63 is checksum

(uFR CS with SAM and firmware versions 5.100.xx)

CMD\_Par0 = index of key for authentication into SAM | 0x80

CMD\_Par1 = index of new key into SAM | 0x80

1<sup>st</sup> byte of the CMD\_EXT = AUTH\_KEY\_TYPE | (NEW\_KEY\_TYPE << 2)

2<sup>nd</sup> byte of the CMD\_EXT = index of old key into SAM | 0x80



array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains 16 zeros  
array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)  
22<sup>nd</sup> byte is key number into application which uses for authentication  
array from 23<sup>rd</sup> to 38<sup>th</sup> byte of CMD\_EXT contains 16 zeros  
38<sup>th</sup> byte is key number into application that will be changed  
array from 39<sup>th</sup> to 54<sup>th</sup> byte of CMD\_EXT contains 16 zeros  
55<sup>th</sup> byte contains checksum.

RSP\_Val0 and RSP\_Val1 are not in use.

If error code is READER\_ERROR or NO\_CARD\_DETECTED, device answer with RSP\_EXT packet of 3 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents execution time of command  
3<sup>rd</sup> byte is checksum.

In other cases, device answer with RSP\_EXT packet of 5 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents error code of operation ( $b2 * 256 + b1$ )  
3<sup>rd</sup> and 4<sup>th</sup> bytes represents execution time of command  
5<sup>th</sup> byte is checksum.

### Example:

Change the key number 2, into AID 0xF00001. Authentication with master application key key number 0.

Key for authentication is internal key number 1, new key is internal key number 2, and old key is internal key number 3.

**CMD**            55 86 AA 37 00 00 55 (send command 88, 0x37 bytes follows 55 checksum)

**ACK**            AC 86 CA 37 00 00 DE (ACK OK)

**CMD\_EXT**      33 21 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 F0  
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 02 00 00 00 00 00  
00 00 00 00 00 00 00 00 00 00 00 00 00 00 E8 (internal key uses so AES key bytes may have any value (all 00), E8 checksum)

**RSP**            DE 86 ED 05 00 00 B7                    (RSP command 86, 5 bytes follows, C5 checksum)

**RSP\_EXT**      B9 0B 1A 00 AF (error code 0BB9, execution time 001A)

### **DESFIRE\_CREATE\_APPLICATION(0x84)**

Function allows to create new application on the card. Is the card master key authentication is required, depend on the card master key settings. Maximal number of applications on the card is 28. Each application is linked to set of up 14 different user definable access keys.

(Old firmwares and AES key)

CMD\_Par0 and CMD\_Par1 are 0

1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal AES key, or 0 if uses external AES key

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal AES key, or 0 if uses external AES key

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains AES key

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is 1 if authentication required, or 0 if no need the authentication

23<sup>rd</sup> byte is application key settings

24<sup>th</sup> byte is maximal number of keys into application

25<sup>th</sup> contains checksum.

(Firmware version from 5.0.25)

CMD\_Par0 = APP\_TYPE | (KEY\_TYPE << 4) and CMD\_Par1 = 0

(Application master key type: AES -> APP\_TYPE = 0, 3K3DES -> APP\_TYPE = 1, DES ->

APP\_TYPE = 2)

1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal key, or 0 if uses external key

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal key, or 0 if uses external key

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains key (for AES and 2K3DES all key bytes, for DES 8 key bytes and 8 zeros, for 3K3DES first 16 key bytes)

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is 1 if authentication required, or 0 if no need the authentication

23<sup>rd</sup> byte is application key settings

24<sup>th</sup> byte is maximal number of keys into application

(for AES, DES and 2K3DES) 25<sup>th</sup> byte contains checksum

(for 3K3DES) array from byte 25 to byte 32 contains last 8 key bytes, and byte 32 contains

checksum

(uFR CS with SAM and firmware versions 5.100.xx)

CMD\_Par0 = APP\_TYPE | (KEY\_TYPE << 4) and CMD\_Par1 = 0

(Application master key type: AES -> APP\_TYPE = 0, 3K3DES -> APP\_TYPE = 1, DES ->

APP\_TYPE = 2)

1<sup>st</sup> byte of the CMD\_EXT is 2 (using key into SAM)

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of key into SAM (0 -127)

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains 16 zeros

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is 1 if authentication required, or 0 if no need the authentication

23<sup>rd</sup> byte is application key settings

24<sup>th</sup> byte is maximal number of keys into application

25<sup>th</sup> contains checksum.

RSP\_Val0 and RSP\_Val1 are not in use.

If error code is `READER_ERROR` or `NO_CARD_DETECTED`, device answer with `RSP_EXT` packet of 3 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents execution time of command  
 3<sup>rd</sup> byte is checksum.

In other cases, device answer with `RSP_EXT` packet of 5 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents error code of operation ( $b2 * 256 + b1$ )  
 3<sup>rd</sup> and 4<sup>th</sup> bytes represents execution time of command  
 5<sup>th</sup> byte is checksum.

**Example:**

Authentication using the internal key ordinal number 1, AID = 0xF00002, key settings is 9, maximal number of application keys is 3, authentication required

**CMD**            55 84 AA 19 00 00 69 (send command 84), 69 checksum  
**ACK**            AC 84 CA 19 00 00 02 (ACK OK)

**CMD\_EXT**    01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 02 00 F0  
 01 09 03 00 (internal key uses so AES key bytes may have any value (all 00), 00 checksum)

**RSP**            DE 84 ED 05 00 00 B9                    (RSP command 84, 5 bytes follows, B9 checksum)

**RSP\_EXT**    B9 0B 1A 00 AF (error code 0BB9, execution time 001A)

**DESFIRE\_DELETE\_APPLICATION(0x89)**

Function allows to deactivate application on the card. AID allocation is removed, but deleted memory blocks can only recovered by using Format card function.

(Old firmwares and AES key)

CMD\_Par0 and CMD\_Par1 are 0  
 1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal AES key, or 0 if uses external AES key  
 2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal AES key, or 0 if uses external AES key  
 array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains AES key  
 array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)  
 22<sup>nd</sup> byte contains checksum

(Firmware version from 5.0.25)

CMD\_Par0 = KEY\_TYPE << 4 and CMD\_Par1 = 0  
 1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal key, or 0 if uses external key  
 2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal key, or 0 if uses external key

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains key (for AES and 2K3DES all key bytes, for DES 8 key bytes and 8 zeros, for 3K3DES first 16 key bytes)

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

(for AES, DES and 2K3DES) 22<sup>nd</sup> byte contains checksum

(for 3K3DES) array from byte 22 to byte 29 contains last 8 key bytes, and byte 30 contains checksum

(uFR CS with SAM and firmware versions 5.100.xx)

CMD\_Par0 = (KEY\_TYPE << 4) and CMD\_Par1 = 0

1<sup>st</sup> byte of the CMD\_EXT is 2 (using key into SAM)

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of key into SAM (0 -127)

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains 16 zeros

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte contains checksum

(Firmware version from 5.0.37)

CMD\_Par0 = KEY\_TYPE << 4

CMD\_Par1 = 0 -> delete with card master key, 1 -> delete with application master key

1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal key, or 0 if uses external key

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal key, or 0 if uses external key

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains key (for AES and 2K3DES all key bytes, for DES 8 key bytes and 8 zeros, for 3K3DES first 16 key bytes)

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

(for AES, DES and 2K3DES) 22<sup>nd</sup> byte contains checksum

(for 3K3DES) array from byte 22 to byte 29 contains last 8 key bytes, and byte 30 contains checksum

checksum

(uFR CS with SAM and firmware versions from 5.100.37)

CMD\_Par0 = (KEY\_TYPE << 4)

CMD\_Par1 = 0 -> delete with card master key, 1 -> delete with application master key

1<sup>st</sup> byte of the CMD\_EXT is 2 (using key into SAM)

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of key into SAM (0 -127)

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains 16 zeros

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte contains checksum

RSP\_Val0 and RSP\_Val1 are not in use.

If error code is READER\_ERROR or NO\_CARD\_DETECTED, device answer with RSP\_EXT packet of 3 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents execution time of command

3<sup>rd</sup> byte is checksum.

In other cases, device answer with RSP\_EXT packet of 5 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents error code of operation (b2 \* 256 + b1)

3<sup>rd</sup> and 4<sup>th</sup> bytes represents execution time of command

5<sup>th</sup> byte is checksum.

### Example:

Authentication using the internal key ordinal number 1, AID = 0xF00002

```
CMD      55 89 AA 16 00 00 67 (send command 89), 67 checksum
ACK      AC 89 CA 16 00 00 00 (ACK OK)
```

```
CMD_EXT  01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 02 00 F0
F9 (internal key uses so AES key bytes may have any value (all 00), F9
checksum)
```

```
RSP      DE 89 ED 05 00 00 C6          (RSP command 89, 5 bytes
follows, C6 checksum)
```

```
RSP_EXT  B9 0B 1A 00 AF (error code 0BB9, execution time 001A)
```

### **DESFIRE\_CREATE\_STD\_FILE(0x85)**

Function allows to create file for the storage unformatted user data within existing application on the card. Maximal number of files into application is 32. The file will be created in the currently selected application. Is the application master key authentication is required, depend on the application master key settings.

Communication settings define communication mode between reader and card. The communication modes are:

- plain communication communication settings value is 0x00
- plain communication secured by MACing communication settings value is 0x01
- fully enciphered communication communication settings value is 0x11

Access rights for read, write, read&write and changing, references certain key within application's keys (0 – 13). If value is 14, this means free access, independent of previous authentication. If value is 15, this means deny access (for example if write access is 15 then the file type is read only).

(Old firmwares and AES key)

CMD\_Par0 and CMD\_Par1 are 0

1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal AES key, or 0 if uses external AES key

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal AES key, or 0 if uses external AES key

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains AES key

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is ID of file that will be created (0 – 31)

23<sup>rd</sup> and 24<sup>th</sup> bytes represented access rights for read, write, read&write and changing (byte 23 = read&write\_key\_no (high 4 bits) | changing\_key\_no (low 4 bits)

byte 24 = read\_key\_no (high 4 bits) | write\_key\_no (low 4 bits))

array from 25<sup>th</sup> to 28<sup>th</sup> of CMD\_EXT contains file size in bytes

29<sup>th</sup> byte is 1 if authentication required, or 0 if no need the authentication

30<sup>th</sup> byte is communication settings

31<sup>st</sup> byte is checksum

(Firmware version from 5.0.25)

CMD\_Par0 = KEY\_TYPE << 4 and CMD\_Par1 = 0

1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal key, or 0 if uses external key

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal key, or 0 if uses external key

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains key (for AES and 2K3DES all key bytes, for DES 8 key bytes and 8 zeros, for 3K3DES first 16 key bytes)

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is ID of file that will be created (0 – 31)

23<sup>rd</sup> and 24<sup>th</sup> bytes represented access rights for read, write, read&write and changing  
(byte 23 = read&write\_key\_no (high 4 bits) | changing\_key\_no (low 4 bits)

byte 24 = read\_key\_no (high 4 bits) | write\_key\_no (low 4 bits))

array from 25<sup>th</sup> to 28<sup>th</sup> of CMD\_EXT contains file size in bytes

29<sup>th</sup> byte is 1 if authentication required, or 0 if no need the authentication

30<sup>th</sup> byte is communication settings

(for AES, DES and 2K3DES) 31<sup>st</sup> byte contains checksum

(for 3K3DES) array from byte 31 to byte 38 contains last 8 key bytes, and byte 39 contains checksum

(uFR CS with SAM and firmware versions 5.100.xx)

CMD\_Par0 = (KEY\_TYPE << 4) and CMD\_Par1 = 0

1<sup>st</sup> byte of the CMD\_EXT is 2 (using key into SAM)

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of key into SAM (0 -127)

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains 16 zeros

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is ID of file that will be created (0 – 31)

23<sup>rd</sup> and 24<sup>th</sup> bytes represented access rights for read, write, read&write and changing  
(byte 23 = read&write\_key\_no (high 4 bits) | changing\_key\_no (low 4 bits)

byte 24 = read\_key\_no (high 4 bits) | write\_key\_no (low 4 bits))

array from 25<sup>th</sup> to 28<sup>th</sup> of CMD\_EXT contains file size in bytes

29<sup>th</sup> byte is 1 if authentication required, or 0 if no need the authentication

30<sup>th</sup> byte is communication settings

31<sup>st</sup> byte is checksum

RSP\_Val0 and RSP\_Val1 are not in use.

If error code is READER\_ERROR or NO\_CARD\_DETECTED, device answer with RSP\_EXT packet of 3 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents execution time of command

3<sup>rd</sup> byte is checksum.

In other cases, device answer with RSP\_EXT packet of 5 bytes.

- 1<sup>st</sup> and 2<sup>nd</sup> bytes represents error code of operation ( $b2 * 256 + b1$ )
- 3<sup>rd</sup> and 4<sup>th</sup> bytes represents execution time of command
- 5<sup>th</sup> byte is checksum.

**Example:**

Authentication using the internal key ordinal number 1, AID = 0xF00002, authentication required, file ID is 1, communication settings is 0x11, access rights is 0x2110 (read with key 2, write with key 1, read&write with key 1, changing with key 0), file size is 1000 (0x000003E8)

**CMD**            55 85 AA 1F 00 00 67 (send command 89), 67 checksum  
**ACK**            AC 85 CA 1F 00 00 00 (ACK OK)

**CMD\_EXT**      01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 02 00 F0  
 01 10 21 E8 03 00 00 01 11 40 (internal key uses so AES key bytes may  
 have any value (all 00), 40 checksum)

**RSP**            DE 85 ED 05 00 00 BA                    (RSP command 85, 5 bytes  
 follows, BA checksum)

**RSP\_EXT**      B9 0B 1A 00 AF (error code 0BB9, execution time 001A)

**DESFIRE\_DELETE\_FILE(0x8A)**

Function deactivates a file within currently selected application. Allocated memory blocks associated with deleted file not set free. Only format card function can delete the memory blocks. Is the application master key authentication is required, depend on the application master key settings.

(Old firmwares and AES key)

- CMD\_Par0 and CMD\_Par1 are 0
- 1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal AES key, or 0 if uses external AES key
- 2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal AES key, or 0 if uses external AES key
- array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains AES key
- array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)
- 22<sup>nd</sup> byte is ID of file that will be deleted (0 – 31)
- 23<sup>rd</sup> byte is 1 if authentication required, or 0 if no need the authentication
- 24<sup>th</sup> byte is checksum

(Firmware version from 5.0.25)

- CMD\_Par0 = KEY\_TYPE << 4 and CMD\_Par1 = 0
- 1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal key, or 0 if uses external key
- 2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal key, or 0 if uses external key
- array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains key (for AES and 2K3DES all key bytes, for DES 8 key bytes and 8 zeros, for 3K3DES first 16 key bytes)



array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)  
 22<sup>nd</sup> byte is ID of file that will be deleted (0 – 31)  
 23<sup>rd</sup> byte is 1 if authentication required, or 0 if no need the authentication  
 (for AES, DES and 2K3DES) 24<sup>th</sup> byte contains checksum  
 (for 3K3DES) array from byte 24 to byte 31 contains last 8 key bytes, and byte 32 contains  
 checksum

(uFR CS with SAM and firmware versions 5.100.xx)

CMD\_Par0 = (KEY\_TYPE << 4) and CMD\_Par1 = 0  
 1<sup>st</sup> byte of the CMD\_EXT is 2 (using key into SAM)  
 2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of key into SAM (0 -127)  
 array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains 16 zeros  
 array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)  
 22<sup>nd</sup> byte is ID of file that will be deleted (0 – 31)  
 23<sup>rd</sup> byte is 1 if authentication required, or 0 if no need the authentication  
 24<sup>th</sup> byte is checksum

RSP\_Val0 and RSP\_Val1 are not in use.

If error code is READER\_ERROR or NO\_CARD\_DETECTED, device answer with RSP\_EXT  
 packet of 3 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents execution time of command  
 3<sup>rd</sup> byte is checksum.

In other cases, device answer with RSP\_EXT packet of 5 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents error code of operation (b2 \* 256 + b1)  
 3<sup>rd</sup> and 4<sup>th</sup> bytes represents execution time of command  
 5<sup>th</sup> byte is checksum.

### Example:

Authentication using the internal key ordinal number 1, AID = 0xF00002, authentication required, file ID is 1

**CMD**            55 8A AA 18 00 00 74 (send command 8A), 74 checksum

**ACK**            AC 8A CA 18 00 00 FB (ACK OK)

**CMD\_EXT**      01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 02 00 F0  
 01 01 F9 (internal key uses so AES key bytes may have any value (all  
 00), F9 checksum)

**RSP**            DE 8A ED 05 00 00 C3                    (RSP command 8A, 5 bytes  
 follows, C3 checksum)

**RSP\_EXT**      B9 0B 1A 00 AF (error code 0BB9, execution time 001A)



## **DESFIRE\_READ\_FROM\_STD\_FILE(0x83)**

Function allow to read data from Standard Data File. Read command requires a preceding authentication either with the key specified for Read or Read&Write access.

From firmware version 5.0.32 Desfire Light tag support

(Old firmwares and AES key)

CMD\_Par0 and CMD\_Par1 are 0

1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal AES key, or 0 if uses external AES key

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal AES key, or 0 if uses external AES key

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains AES key

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is application key number for reading

23<sup>rd</sup> byte is ID of file (0 – 31)

23<sup>rd</sup> byte is 1 if authentication required, or 0 if no need the authentication

24<sup>th</sup> and 25<sup>th</sup> bytes represents start position for read operation within file

26<sup>th</sup> and 27<sup>th</sup> bytes represents number of data to be read

28<sup>th</sup> byte is communication settings

29<sup>th</sup> byte is checksum

(Firmware version from 5.0.25)

CMD\_Par0 = KEY\_TYPE << 4 and CMD\_Par1 = 0

1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal key, or 0 if uses external key

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal key, or 0 if uses external key

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains key (for AES and 2K3DES all key bytes, for DES 8 key bytes and 8 zeros, for 3K3DES first 16 key bytes)

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is application key number for reading

23<sup>rd</sup> byte is ID of file (0 – 31)

23<sup>rd</sup> byte is 1 if authentication required, or 0 if no need the authentication

24<sup>th</sup> and 25<sup>th</sup> bytes represents start position for read operation within file

26<sup>th</sup> and 27<sup>th</sup> bytes represents number of data to be read

28<sup>th</sup> byte is communication settings

(for AES, DES and 2K3DES) 29<sup>th</sup> byte contains checksum

(for 3K3DES) array from byte 29 to byte 36 contains last 8 key bytes, and byte 37 contains checksum

(uFR CS with SAM and firmware versions 5.100.xx)

CMD\_Par0 = (KEY\_TYPE << 4) and CMD\_Par1 = 0

1<sup>st</sup> byte of the CMD\_EXT is 2 (using key into SAM)

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of key into SAM (0 -127)

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains 16 zeros

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

- 22<sup>nd</sup> byte is application key number for reading
- 23<sup>rd</sup> byte is ID of file (0 – 31)
- 23<sup>rd</sup> byte is 1 if authentication required, or 0 if no need the authentication
- 24<sup>th</sup> and 25<sup>th</sup> bytes represents start position for read operation within file
- 26<sup>th</sup> and 27<sup>th</sup> bytes represents number of data to be read
- 28<sup>th</sup> byte is communication settings
- 29<sup>th</sup> byte is checksum

Reading the data is specific and is done in a loop. Reads one data, and if it is 0, then reads another that indicates how much data follows in the package. This is repeated until the required amount of data read. If the first data is different from 0, then reader will be sent standard response.

RSP\_Val0 and RSP\_Val1 are not in use.

If error code is `READER_ERROR` or `NO_CARD_DETECTED`, device answer with `RSP_EXT` packet of 3 bytes.

- 1<sup>st</sup> and 2<sup>nd</sup> bytes represents execution time of command
- 3<sup>rd</sup> byte is checksum.

In other cases, device answer with `RSP_EXT` packet of 5 bytes.

- 1<sup>st</sup> and 2<sup>nd</sup> bytes represents error code of operation ( $b2 * 256 + b1$ )
- 3<sup>rd</sup> and 4<sup>th</sup> bytes represents execution time of command
- 5<sup>th</sup> byte is checksum.

### Example:

Authentication using the internal key ordinal number 3, AID = 0xF00002, authentication required, file ID is 1, reading key number is 2, bytes for read 50 from start address 10, communication settings 0x11

```
CMD      55 83 AA 1D 00 00 68 (send command 83), 68 checksum
ACK      AC 83 CA 1D 00 00 FB (ACK OK)
```

```
CMD_EXT  01 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 02 00 F0
02 01 01 0A 00 32 00 11 E2 (internal key uses so AES key bytes may have
any value (all 00), E2 checksum)
```

```
DATA     00 32 01 02 03 04 05 06 07 08 09 0A 01 02 03 04 05 06 07 08 09
0A 01 02 03 04 05 06 07 08 09 0A 01 02 03 04 05 06 07 08 09 0A 01 02 03
04 05 06 07 08 09 0A
```

```
RSP      DE 8A ED 05 00 00 C3          (RSP command 8A, 5 bytes
follows, C3 checksum)
```

```
RSP_EXT  B9 0B 1A 00 AF (error code 0BB9, execution time 001A)
```

## **DESFIRE\_WRITE\_TO\_STD\_FILE(0x82)**

Function allow to write data to Standard Data File, or to Backup Data File. Write command requires a preceding authentication either with the key specified for Write or Read&Write access. From firmware version 5.0.32 Desfire Light tag support

(Old firmwares and AES key)

CMD\_Par0 and CMD\_Par1 are 0

1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal AES key, or 0 if uses external AES key

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal AES key, or 0 if uses external AES key

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains AES key

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is application key number for writing

23<sup>rd</sup> byte is ID of file (0 – 31)

24<sup>th</sup> byte is 1 if authentication required, or 0 if no need the authentication

25<sup>th</sup> and 26<sup>th</sup> bytes represents start position for read operation within file

27<sup>th</sup> and 28<sup>th</sup> bytes represents number of data to be write

29<sup>th</sup> byte is communication settings

array from 30<sup>th</sup> to 30 + block size number of data for writing contains maximal 160 data for writing

31 + block size byte is checksum

(Firmware version from 5.0.25)

CMD\_Par0 = KEY\_TYPE << 4 and CMD\_Par1 = 0

1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal key, or 0 if uses external key

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal key, or 0 if uses external key

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains key (for AES and 2K3DES all key bytes, for DES 8 key bytes and 8 zeros, for 3K3DES first 16 key bytes)

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is application key number for writing

23<sup>rd</sup> byte is ID of file (0 – 31)

24<sup>th</sup> byte is 1 if authentication required, or 0 if no need the authentication

25<sup>th</sup> and 26<sup>th</sup> bytes represents start position for read operation within file

27<sup>th</sup> and 28<sup>th</sup> bytes represents number of data to be write

29<sup>th</sup> byte is communication settings

array from 30<sup>th</sup> to 30 + block size number of data for writing contains maximal 160 data for writing

(for AES, DES and 2K3DES) (31 + block size) byte is checksum

(for 3K3DES) array from byte (31 + block size) to byte (38 + block size) contains last 8 key bytes, and byte (39 + block size) contains checksum

(uFR CS with SAM and firmware versions 5.100.xx)

$CMD\_Par0 = (KEY\_TYPE \ll 4)$  and  $CMD\_Par1 = 0$

1<sup>st</sup> byte of the  $CMD\_EXT$  is 2 (using key into SAM)

2<sup>nd</sup> byte of the  $CMD\_EXT$  contains ordinal number of key into SAM (0 -127)

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of  $CMD\_EXT$  contains 16 zeros

array from 19<sup>th</sup> to 21<sup>st</sup> byte of  $CMD\_EXT$  contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is application key number for writing

23<sup>rd</sup> byte is ID of file (0 – 31)

24<sup>th</sup> byte is 1 if authentication required, or 0 if no need the authentication

25<sup>th</sup> and 26<sup>th</sup> bytes represents start position for read operation within file

27<sup>th</sup> and 28<sup>th</sup> bytes represents number of data to be write

29<sup>th</sup> byte is communication settings

array from 30<sup>th</sup> to 30 + block size number of data for writing contains maximal 160 data for writing

31 + block size byte is checksum

If you want to enter more than 160 bytes, then it is done in blocks of up to 160 bytes. After the first block of data reader sent 0xAD if necessary to receive more data, or 0xDD if no need more data, or at any error. When you receive 0xAD then sends a packet in which the first byte indicates how many bytes follow. When you receive 0xDD then follow standard response.

$RSP\_Val0$  and  $RSP\_Val1$  are not in use.

If error code is  $READER\_ERROR$  or  $NO\_CARD\_DETECTED$ , device answer with  $RSP\_EXT$  packet of 3 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents execution time of command

3<sup>rd</sup> byte is checksum.

In other cases, device answer with  $RSP\_EXT$  packet of 5 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents error code of operation ( $b2 * 256 + b1$ )

3<sup>rd</sup> and 4<sup>th</sup> bytes represents execution time of command

5<sup>th</sup> byte is checksum.

### Example:

Authentication using the internal key ordinal number 3, AID = 0xF00002, authentication required, file ID is 1, writing key number is 1, bytes for write 50 from start address 10, communication settings 0x11

```
CMD      55 82 AA 51 00 00 33 (send command 82), 33 checksum
ACK      AC 82 CA 51 00 00 BC (ACK OK)

CMD_EXT  01 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 02 00 F0
01 01 01 0A 00 32 00 11 01 02 03 04 05 06 07 08 09 0A 01 02 03 04 05 06
07 08 09 0A 01 02 03 04 05 06 07 08 09 0A 01 02 03 04 05 06 07 08 09 0A
01 02 03 04 05 06 07 08 09 0A CRC (internal key uses so AES key bytes
may have any value (all 00), CRC checksum)

DATA      DD (no need more data)

RSP      DE 82 ED 05 00 00 BB (RSP command 82, 5 bytes
follows, BB checksum)
RSP_EXT  B9 0B 1A 00 AF (error code 0BB9, execution time 001A)
```

### **DESFIRE\_CREATE\_VALUE\_FILE(0x8F)**

For uFR PLUS devices only.

Function allows to create file for the storage and manipulation of 32 bit signed integer values within existing application on the card. Maximal number of files into application is 32. The file will be created in the currently selected application. Is the application master key authentication is required, depend on the application master key settings.

Communication settings define communication mode between reader and card. The communication modes are:

- plain communication communication settings value is 0x00
- plain communication secured by MACing communication settings value is 0x01
- fully enciphered communication communication settings value is 0x11

Access rights for read, write, read&write and changing, references certain key within application's keys (0 – 13). If value is 14, this means free access, independent of previous authentication. If value is 15, this means deny access (for example if write access is 15 then the file type is read only).

(Old firmwares and AES key)

CMD\_Par0 and CMD\_Par1 are 0

1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal AES key, or 0 if uses external AES key

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal AES key, or 0 if uses external AES key

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains AES key

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is ID of file that will be created (0 – 31)

23<sup>rd</sup> and 24<sup>th</sup> bytes represented access rights for read, write, read&write and changing (byte 23 = read&write\_key\_no (high 4 bits) | changing\_key\_no (low 4 bits))

byte 24 = read\_key\_no (high 4 bits) | write\_key\_no (low 4 bits))  
array from 25<sup>th</sup> to 28<sup>th</sup> byte contains value of lower limit (lowest byte is first)  
array from 29<sup>th</sup> to 32<sup>nd</sup> byte contains value of upper limit (lowest byte is first)  
array from 33<sup>rd</sup> to 36<sup>th</sup> byte contains initial value of value file (lowest byte is first)  
37<sup>th</sup> byte  
bit 0 – limited credit enabled (1 – yes, 0 – no)  
bit 1 – free get value (1 – yes, 0 – no)  
38<sup>th</sup> byte is 1 if authentication required, or 0 if no need the authentication  
39<sup>th</sup> byte is communication settings  
40<sup>st</sup> byte is checksum

(Firmware version from 5.0.25)

CMD\_Par0 = KEY\_TYPE << 4 and CMD\_Par1 = 0  
1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal key, or 0 if uses external key  
2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal key, or 0 if uses external key  
array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains key (for AES and 2K3DES all key bytes, for DES 8 key bytes and 8 zeros, for 3K3DES first 16 key bytes)  
array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)  
22<sup>nd</sup> byte is ID of file that will be created (0 – 31)  
23<sup>rd</sup> and 24<sup>th</sup> bytes represented access rights for read, write, read&write and changing  
(byte 23 = read&write\_key\_no (high 4 bits) | changing\_key\_no (low 4 bits))  
byte 24 = read\_key\_no (high 4 bits) | write\_key\_no (low 4 bits))  
array from 25<sup>th</sup> to 28<sup>th</sup> byte contains value of lower limit (lowest byte is first)  
array from 29<sup>th</sup> to 32<sup>nd</sup> byte contains value of upper limit (lowest byte is first)  
array from 33<sup>rd</sup> to 36<sup>th</sup> byte contains initial value of value file (lowest byte is first)  
37<sup>th</sup> byte  
bit 0 – limited credit enabled (1 – yes, 0 – no)  
bit 1 – free get value (1 – yes, 0 – no)  
38<sup>th</sup> byte is 1 if authentication required, or 0 if no need the authentication  
39<sup>th</sup> byte is communication settings  
(for AES, DES and 2K3DES) 40<sup>st</sup> byte is checksum byte contains checksum  
(for 3K3DES) array from byte 40 to byte 47 contains last 8 key bytes, and byte 48 contains checksum

(uFR CS with SAM and firmware versions 5.100.xx)

CMD\_Par0 = (KEY\_TYPE << 4) and CMD\_Par1 = 0  
1<sup>st</sup> byte of the CMD\_EXT is 2 (using key into SAM)  
2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of key into SAM (0 -127)  
array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains 16 zeros  
array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)  
22<sup>nd</sup> byte is ID of file that will be created (0 – 31)  
23<sup>rd</sup> and 24<sup>th</sup> bytes represented access rights for read, write, read&write and changing  
(byte 23 = read&write\_key\_no (high 4 bits) | changing\_key\_no (low 4 bits))  
byte 24 = read\_key\_no (high 4 bits) | write\_key\_no (low 4 bits))

array from 25<sup>th</sup> to 28<sup>th</sup> byte contains value of lower limit (lowest byte is first)  
 array from 29<sup>th</sup> to 32<sup>nd</sup> byte contains value of upper limit (lowest byte is first)  
 array from 33<sup>rd</sup> to 36<sup>th</sup> byte contains initial value of value file (lowest byte is first)  
 37<sup>th</sup> byte  
 bit 0 – limited credit enabled (1 – yes, 0 – no)  
 bit 1 – free get value (1 – yes, 0 – no)  
 38<sup>th</sup> byte is 1 if authentication required, or 0 if no need the authentication  
 39<sup>th</sup> byte is communication settings  
 40<sup>st</sup> byte is checksum

If error code is `READER_ERROR` or `NO_CARD_DETECTED`, device answer with `RSP_EXT` packet of 3 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents execution time of command  
 3<sup>rd</sup> byte is checksum.

In other cases, device answer with `RSP_EXT` packet of 5 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents error code of operation ( $b2 * 256 + b1$ )  
 3<sup>rd</sup> and 4<sup>th</sup> bytes represents execution time of command  
 5<sup>th</sup> byte is checksum.

### Example:

Authentication using the internal key ordinal number 3, AID = 0xF00008, authentication required, file ID is 20, access rights is 0x0000 (read with key 0, write with key 0, read&write with key 0, changing with key 0), lower limit is 100, upper limit is 300, initial value is 200, communication settings 0x0. Upper limit must be bigger than or equal to the lower limit and initial value.

```
CMD      55 8F AA 28 00 00 5F (send command 8F) , 5F checksum
ACK      AC 8F CA 28 00 00 C8 (ACK OK)
```

```
CMD_EXT  01 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 08 00
F0 14 00 00 64 00 00 00 2C 01 00 00 C8 00 00 00 00 01 00 75 CRC
(internal key uses so AES key bytes may have any value (all 00) , CRC
checksum)
```

```
RSP      DE 8F ED 05 00 00 C0
```

```
RSP_EXT  B9 0B 46 00 FB (error code 0x0BB9, execution time 0x0046)
```

### **DESFIRE\_READ\_VALUE\_FILE( 0x9A)**

For uFR PLUS devices only.

Function allow to read value from value files. Read command requires a preceding authentication either with the key specified for Read or Read&Write access.

From firmware version 5.0.32 Desfire Light tag support

(Old firmwares and AES key)



CMD\_Par0 and CMD\_Par1 are 0

1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal AES key, or 0 if uses external AES key

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal AES key, or 0 if uses external AES key

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains AES key

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is application key number for reading

23<sup>rd</sup> byte is ID of file (0 – 31)

24<sup>th</sup> byte is 1 if authentication required, or 0 if no need the authentication

25<sup>th</sup> and 26<sup>th</sup> bytes represents start position for read operation within file

27<sup>th</sup> and 28<sup>th</sup> bytes represents number of data to be write

29<sup>th</sup> byte is communication settings

(Firmware version from 5.0.25)

CMD\_Par0 = KEY\_TYPE << 4 and CMD\_Par1 = 0

1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal key, or 0 if uses external key

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal key, or 0 if uses external key

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains key (for AES and 2K3DES all key bytes, for DES 8 key bytes and 8 zeros, for 3K3DES first 16 key bytes)

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is application key number for reading

23<sup>rd</sup> byte is ID of file (0 – 31)

24<sup>th</sup> byte is 1 if authentication required, or 0 if no need the authentication

25<sup>th</sup> and 26<sup>th</sup> bytes represents start position for read operation within file

27<sup>th</sup> and 28<sup>th</sup> bytes represents number of data to be write

29<sup>th</sup> byte is communication settings

(for AES, DES and 2K3DES) 29<sup>th</sup> byte is checksum byte contains checksum

(for 3K3DES) array from byte 29 to byte 36 contains last 8 key bytes, and byte 37 contains checksum

(uFR CS with SAM and firmware versions 5.100.xx)

CMD\_Par0 = (KEY\_TYPE << 4) and CMD\_Par1 = 0

1<sup>st</sup> byte of the CMD\_EXT is 2 (using key into SAM)

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of key into SAM (0 -127)

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains 16 zeros

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is application key number for reading

23<sup>rd</sup> byte is ID of file (0 – 31)

24<sup>th</sup> byte is 1 if authentication required, or 0 if no need the authentication

25<sup>th</sup> and 26<sup>th</sup> bytes represents start position for read operation within file

27<sup>th</sup> and 28<sup>th</sup> bytes represents number of data to be write

29<sup>th</sup> byte is communication settings



If no error, i.e. error code is CARD\_OPERATION\_OK, device answer with RSP packet and after that also the RSP\_EXT packet of 9 bytes.

RSP\_Val0 and RSP\_Val1 are not in use.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents error code of operation ( $b2 * 256 + b1$ )

3<sup>rd</sup> and 4<sup>th</sup> bytes represents execution time of command

array from 5<sup>th</sup> to 8<sup>th</sup> byte is value of value file

9<sup>th</sup> byte is checksum

If error code is READER\_ERROR or NO\_CARD\_DETECTED, device answer with RSP\_EXT packet of 3 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents execution time of command

3<sup>rd</sup> byte is checksum.

In other cases, device answer with RSP\_EXT packet of 5 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents error code of operation ( $b2 * 256 + b1$ )

3<sup>rd</sup> and 4<sup>th</sup> bytes represents execution time of command

5<sup>th</sup> byte is checksum.

### Example:

Authentication using the internal key ordinal number 3, AID = 0xF00008, authentication required, file ID is 20, application reading key number is 0.

**CMD**            55 9A AA 1A 00 00 86 (send command 9A), 86 checksum

**ACK**            AC 9A CA 1A 00 00 ED (ACK OK)

**CMD\_EXT**      01 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 08 00  
 F0 00 14 01 00 F6 (internal key uses so AES key bytes may have any value  
 (all 00))

**RSP**            DE 9A ED 09 00 00 A7

**RSP\_EXT**      B9 0B 46 00 C8 00 00 00 43      (error code 0x0BB9, execution  
 time 0x0046, value 0x000000C8)

### **DESFIRE\_INCREASE\_VALUE\_FILE(0x9B)**

For uFR PLUS devices only.

Function allows to increase a value stored in a value files. Credit command requires a preceding authentication with the key specified for Read&Write access.

From firmware version 5.0.32 Desfire Light tag support

From firmware version 5.0.38 Transaction MAC for Desfire Light and Desfire EV2 support

(Old firmwares and AES key)

CMD\_Par0 and CMD\_Par1 are 0

1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal AES key, or 0 if uses external AES key

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal AES key, or 0 if uses external AES key

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains AES key

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is application key number for read&write

23<sup>rd</sup> byte is ID of file (0 – 31)

24<sup>th</sup> byte is 1 if authentication required, or 0 if no need the authentication

25<sup>th</sup> byte is communication settings

array from 26<sup>th</sup> and 29<sup>th</sup> bytes represents value (must be positive number)

30<sup>st</sup> byte is checksum byte contains checksum

(Firmware version from 5.0.25)

CMD\_Par0 = KEY\_TYPE << 4 and CMD\_Par1 = 0

1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal key, or 0 if uses external key

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal key, or 0 if uses external key

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains key (for AES and 2K3DES all key bytes, for DES 8 key bytes and 8 zeros, for 3K3DES first 16 key bytes)

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is application key number for read&write

23<sup>rd</sup> byte is ID of file (0 – 31)

24<sup>th</sup> byte is 1 if authentication required, or 0 if no need the authentication

25<sup>th</sup> byte is communication settings

array from 26<sup>th</sup> and 29<sup>th</sup> bytes represents value (must be positive number)

(for AES, DES and 2K3DES) 30<sup>st</sup> byte is checksum byte contains checksum

(for 3K3DES) array from byte 30 to byte 37 contains last 8 key bytes, and byte 38 contains checksum

(uFR CS with SAM and firmware versions 5.100.xx)

CMD\_Par0 = (KEY\_TYPE << 4) and CMD\_Par1 = 0

1<sup>st</sup> byte of the CMD\_EXT is 2 (using key into SAM)

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of key into SAM (0 -127)

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains 16 zeros

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is application key number for read&write

23<sup>rd</sup> byte is ID of file (0 – 31)

24<sup>th</sup> byte is 1 if authentication required, or 0 if no need the authentication

25<sup>th</sup> byte is communication settings

array from 26<sup>th</sup> and 29<sup>th</sup> bytes represents value (must be positive number)

30<sup>st</sup> byte is checksum byte contains checksum

(Firmware version from 5.0.38)

tmc\_file = 0 -> Transaction MAC is not used

tmc\_file = 1 -> Transaction MAC is used Reader ID is not used

tmc\_file = 3 -> Transaction MAC is used Reader ID is used

CMD\_Par0 = KEY\_TYPE << 4 and CMD\_Par1 = tmc\_file

1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal key, or 0 if uses external key  
2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal key, or 0 if uses external key  
array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains key (for AES and 2K3DES all key bytes, for DES 8 key bytes and 8 zeros, for 3K3DES first 16 key bytes)  
array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)  
22<sup>nd</sup> byte is application key number for read&write  
23<sup>rd</sup> byte is ID of file (0 – 31)  
24<sup>th</sup> byte is 1 if authentication required, or 0 if no need the authentication  
25<sup>th</sup> byte is communication settings  
array from 26<sup>th</sup> and 29<sup>th</sup> bytes represents value (must be positive number)  
(for AES, DES and 2K3DES) 30<sup>st</sup> byte is checksum byte contains checksum  
(for 3K3DES) array from byte 30 to byte 37 contains last 8 key bytes, and byte 38 contains checksum

(uFR CS with SAM and firmware from version 5.100.38)

tmc\_file = 0 -> Transaction MAC is not used  
tmc\_file = 1 -> Transaction MAC is used Reader ID is not used  
tmc\_file = 3 -> Transaction MAC is used Reader ID is used  
CMD\_Par0 = (KEY\_TYPE << 4) and CMD\_Par1 = tmc\_file  
1<sup>st</sup> byte of the CMD\_EXT is 2 (using key into SAM)  
2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of key into SAM (0 -127)  
array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains 16 zeros  
array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)  
22<sup>nd</sup> byte is application key number for read&write  
23<sup>rd</sup> byte is ID of file (0 – 31)  
24<sup>th</sup> byte is 1 if authentication required, or 0 if no need the authentication  
25<sup>th</sup> byte is communication settings  
array from 26<sup>th</sup> and 29<sup>th</sup> bytes represents value (must be positive number)  
30<sup>st</sup> byte is checksum byte contains checksum

If error code is READER\_ERROR or NO\_CARD\_DETECTED, device answer with RSP\_EXT packet of 3 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents execution time of command  
3<sup>rd</sup> byte is checksum.

In other cases, device answer with RSP\_EXT packet of 5 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents error code of operation ( $b_2 * 256 + b_1$ )  
3<sup>rd</sup> and 4<sup>th</sup> bytes represents execution time of command  
5<sup>th</sup> byte is checksum.

From version 5.0.38. if tmc\_file > 0

1<sup>st</sup> and 2<sup>nd</sup> bytes represents error code of operation ( $b_2 * 256 + b_1$ )  
3<sup>rd</sup> and 4<sup>th</sup> bytes represents execution time of command  
5th to 20th bytes represent Reader ID

21st to 36th bytes represent Previous Encrypted Reader ID  
 37th to 40th bytes represent Transaction MAC counter  
 41st to 48th bytes represent Transaction MAC  
 49th byte is checksum.

**Example:**

Authentication using the internal key ordinal number 3, AID = 0xF00008, authentication required, file ID is 20, application read&write key number is 0, increase value is 100

```
CMD      55 9B AA 1E 00 00 81
ACK      AC 9B CA 1E 00 00 EA
```

```
CMD_EXT  01 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 08 00
F0 00 14 01 00 64 00 00 00 92 (internal key uses so AES key bytes may
have any value (all 00))
```

```
RSP      DE 9B ED 05 00 00 B4
```

```
RSP_EXT  B9 0B 67 00 DC (error code 0x0BB9, execution time 0x0067)
```

**DESFIRE\_DECREASE\_VALUE\_FILE(0x9C)**

For uFR PLUS devices only.

Function allows to decrease value from value files. Debit command requires a preceding authentication with on of the keys specified for Read, Write or Read&Write access.

From firmware version 5.0.32 Desfire Light tag support

From firmware version 5.0.38 Transaction MAC for Desfire Light and Desfire EV2 support

(Old firmwares and AES key)

CMD\_Par0 and CMD\_Par1 are 0

1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal AES key, or 0 if uses external AES key

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal AES key, or 0 if uses external AES key

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains AES key

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is application key number for read, write or read&write

23<sup>rd</sup> byte is ID of file (0 – 31)

24<sup>th</sup> byte is 1 if authentication required, or 0 if no need the authentication

25<sup>th</sup> byte is communication settings

array from 26<sup>th</sup> and 29<sup>th</sup> bytes represents value (must be positive number)

30<sup>st</sup> byte is checksum byte contains checksum

(Firmware version from 5.0.25)

CMD\_Par0 = KEY\_TYPE << 4 and CMD\_Par1 = 0

1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal key, or 0 if uses external key

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal key, or 0 if uses external key

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains key (for AES and 2K3DES all key bytes, for DES 8 key bytes and 8 zeros, for 3K3DES first 16 key bytes)  
array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)  
22<sup>nd</sup> byte is application key number for read&write  
23<sup>rd</sup> byte is ID of file (0 – 31)  
24<sup>th</sup> byte is 1 if authentication required, or 0 if no need the authentication  
25<sup>th</sup> byte is communication settings  
array from 26<sup>th</sup> and 29<sup>th</sup> bytes represents value (must be positive number)  
(for AES, DES and 2K3DES) 30<sup>st</sup> byte is checksum byte contains checksum  
(for 3K3DES) array from byte 30 to byte 37 contains last 8 key bytes, and byte 38 contains checksum

(uFR CS with SAM and firmware versions 5.100.xx)

CMD\_Par0 = (KEY\_TYPE << 4) and CMD\_Par1 = 0  
1<sup>st</sup> byte of the CMD\_EXT is 2 (using key into SAM)  
2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of key into SAM (0 -127)  
array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains 16 zeros  
array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)  
22<sup>nd</sup> byte is application key number for read, write or read&write  
23<sup>rd</sup> byte is ID of file (0 – 31)  
24<sup>th</sup> byte is 1 if authentication required, or 0 if no need the authentication  
25<sup>th</sup> byte is communication settings  
array from 26<sup>th</sup> and 29<sup>th</sup> bytes represents value (must be positive number)  
30<sup>st</sup> byte is checksum byte contains checksum

(Firmware version from 5.0.38)

tmc\_file = 0 -> Transaction MAC is not used  
tmc\_file = 1 -> Transaction MAC is used Reader ID is not used  
tmc\_file = 3 -> Transaction MAC is used Reader ID is used  
CMD\_Par0 = KEY\_TYPE << 4 and CMD\_Par1 = tmc\_file  
1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal key, or 0 if uses external key  
2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal key, or 0 if uses external key  
array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains key (for AES and 2K3DES all key bytes, for DES 8 key bytes and 8 zeros, for 3K3DES first 16 key bytes)  
array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)  
22<sup>nd</sup> byte is application key number for read&write  
23<sup>rd</sup> byte is ID of file (0 – 31)  
24<sup>th</sup> byte is 1 if authentication required, or 0 if no need the authentication  
25<sup>th</sup> byte is communication settings  
array from 26<sup>th</sup> and 29<sup>th</sup> bytes represents value (must be positive number)  
(for AES, DES and 2K3DES) 30<sup>st</sup> byte is checksum byte contains checksum  
(for 3K3DES) array from byte 30 to byte 37 contains last 8 key bytes, and byte 38 contains checksum

(uFR CS with SAM and firmware from version 5.100.38)

tmc\_file = 0 -> Transaction MAC is not used  
 tmc\_file = 1 -> Transaction MAC is used Reader ID is not used  
 tmc\_file = 3 -> Transaction MAC is used Reader ID is used  
 CMD\_Par0 = (KEY\_TYPE << 4) and CMD\_Par1 = tmc\_file  
 1<sup>st</sup> byte of the CMD\_EXT is 2 (using key into SAM)  
 2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of key into SAM (0 -127)  
 array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains 16 zeros  
 array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)  
 22<sup>nd</sup> byte is application key number for read, write or read&write  
 23<sup>rd</sup> byte is ID of file (0 – 31)  
 24<sup>th</sup> byte is 1 if authentication required, or 0 if no need the authentication  
 25<sup>th</sup> byte is communication settings  
 array from 26<sup>th</sup> and 29<sup>th</sup> bytes represents value (must be positive number)  
 30<sup>st</sup> byte is checksum byte contains checksum

If error code is READER\_ERROR or NO\_CARD\_DETECTED, device answer with RSP\_EXT packet of 3 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents execution time of command  
 3<sup>rd</sup> byte is checksum.

In other cases, device answer with RSP\_EXT packet of 5 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents error code of operation (b2 \* 256 + b1)  
 3<sup>rd</sup> and 4<sup>th</sup> bytes represents execution time of command  
 5<sup>th</sup> byte is checksum.

From version 5.0.38. if tmc\_file > 0

1<sup>st</sup> and 2<sup>nd</sup> bytes represents error code of operation (b2 \* 256 + b1)  
 3<sup>rd</sup> and 4<sup>th</sup> bytes represents execution time of command  
 5th to 20th bytes represent Reader ID  
 21st to 36th bytes represent Previous Encrypted Reader ID  
 37th to 40th bytes represent Transaction MAC counter  
 41st to 48th bytes represent Transaction MAC  
 49th byte is checksum.

### Example:

Authentication using the internal key ordinal number 3, AID = 0xF00008, authentication required, file ID is 20, application read&write key number is 0, increase value is 100

```
CMD      55 9C AA 1E 00 00 84
ACK      AC 9C CA 1E 00 00 EB
```

```
CMD_EXT  01 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 08 00
F0 00 14 01 00 64 00 00 00 92 (internal key uses so AES key bytes may
have any value (all 00))
```

RSP DE 9C ED 05 00 00 B1  
RSP\_EXT B9 0B 67 00 DC (error code 0x0BB9, execution time 0x0067)

### DESFIRE\_GET\_APPLICATION\_IDS (0xC0)

For uFR PLUS devices only.

Function returns the Application Identifiers for all active applications on a card. Maximal number of application ids is 28.

(Old firmwares and AES key)

CMD\_Par0 and CMD\_Par1 are 0

1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal AES key, or 0 if uses external AES key

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal AES key, or 0 if uses external AES key

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains AES key

19<sup>th</sup> byte is 1 if authentication required, or 0 if no need the authentication

20<sup>st</sup> byte is checksum byte contains checksum

(Firmware version from 5.0.25)

CMD\_Par0 = KEY\_TYPE << 4 and CMD\_Par1 = 0

1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal key, or 0 if uses external key

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal key, or 0 if uses external key

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains key (for AES and 2K3DES all key bytes, for DES 8 key bytes and 8 zeros, for 3K3DES first 16 key bytes)

19<sup>th</sup> byte is 1 if authentication required, or 0 if no need the authentication

(for AES, DES and 2K3DES) 20<sup>st</sup> byte is checksum byte contains checksum

(for 3K3DES) array from byte 20 to byte 27 contains last 8 key bytes, and byte 28 contains checksum

(uFR CS with SAM and firmware versions 5.100.xx)

CMD\_Par0 = (KEY\_TYPE << 4) and CMD\_Par1 = 0

1<sup>st</sup> byte of the CMD\_EXT is 2 (using key into SAM)

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of key into SAM (0 -127)

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains 16 zeros

19<sup>th</sup> byte is 1 if authentication required, or 0 if no need the authentication

20<sup>st</sup> byte is checksum byte contains checksum

If error code is READER\_ERROR or NO\_CARD\_DETECTED, device answer with RSP\_EXT packet of 3 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents execution time of command

3<sup>rd</sup> byte is checksum.

In other cases, device answer with RSP\_EXT packet of 3 \* number\_of\_application\_ids + 7 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents error code of operation (b2 \* 256 + b1)

3<sup>rd</sup> and 4<sup>th</sup> bytes represents execution time of command

5<sup>th</sup> byte is number of application identifiers



6<sup>th</sup> to (6 + 3 \* number\_of\_application\_ids)<sup>th</sup> are triplets of bytes which represents application identifier (little endian numbers)

(7 + 3 \* number\_of\_application\_ids)<sup>th</sup> is checksum

### Example:

Authentication using the internal key ordinal number 2, authentication required.

There are 2 application ID-s 0xA10000 and 0xA20000

**CMD**            55 C0 AA 13 00 00 33

**ACK**            AC C0 CA 13 00 00 DC

**CMD\_EXT**       01 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 09

(internal key uses so AES key bytes may have any value (all 00))

**RSP**            DE C0 ED 0C 00 00 06

**RSP\_EXT**       B9 0B 67 00 02 00 00 A1 00 00 A2 DB        (error code 0x0BB9,  
execution time 0x0067)

### **DESIRE\_CREATE\_RECORD\_FILE (0xC1)**

For uFR PLUS devices only.

Function allows to create file for multiple storage of structural data, within an existing application.

Linear Record File.

Once the file filled completely with data records, further writing to file is not possible unless it is cleared.

Cyclic Record File.

Once the file filled completely with data records, the card automatically overwrites the oldest record with latest written one.

CMD\_Par0 = KEY\_TYPE << 4 and CMD\_Par1 = 0

1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal key, or 0 if uses external key

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal key, or 0 if uses external key array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains key (for AES and 2K3DES all key bytes, for DES 8 key bytes and 8 zeros, for 3K3DES first 16 key bytes)

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is ID of file that will be created (0 – 31)

23<sup>rd</sup> and 24<sup>th</sup> bytes represented access rights for read, write, read&write and changing (byte 23 = read&write\_key\_no (high 4 bits) | changing\_key\_no (low 4 bits)

byte 24 = read\_key\_no (high 4 bits) | write\_key\_no (low 4 bits))

array from 25<sup>th</sup> to 28<sup>th</sup> of CMD\_EXT contains record size in bytes

array from 29<sup>th</sup> to 32<sup>nd</sup> of CMD\_EXT contains maximal number of records

33<sup>rd</sup> byte is 1 if authentication required, or 0 if no need the authentication

34<sup>th</sup> byte is communication settings

(for AES, DES and 2K3DES) 35<sup>th</sup> byte contains checksum



(for 3K3DES) array from byte 35 to byte 42 contains last 8 key bytes, and byte 43 contains checksum

(uFR CS with SAM and firmware versions 5.100.xx)

$CMD\_Par0 = (KEY\_TYPE \ll 4)$  and  $CMD\_Par1 = 0$

1<sup>st</sup> byte of the  $CMD\_EXT$  is 2 (using key into SAM)

2<sup>nd</sup> byte of the  $CMD\_EXT$  contains ordinal number of key into SAM (0 -127)

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of  $CMD\_EXT$  contains 16 zeros

array from 19<sup>th</sup> to 21<sup>st</sup> byte of  $CMD\_EXT$  contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is ID of file that will be created (0 – 31)

23<sup>rd</sup> and 24<sup>th</sup> bytes represented access rights for read, write, read&write and changing

(byte 23 = read&write\_key\_no (high 4 bits) | changing\_key\_no (low 4 bits))

byte 24 = read\_key\_no (high 4 bits) | write\_key\_no (low 4 bits))

array from 25<sup>th</sup> to 28<sup>th</sup> of  $CMD\_EXT$  contains record size in bytes

array from 29<sup>th</sup> to 32<sup>nd</sup> of  $CMD\_EXT$  contains maximal number of records

33<sup>rd</sup> byte is 1 if authentication required, or 0 if no need the authentication

34<sup>th</sup> byte is communication settings

35<sup>th</sup> byte contains checksum

RSP\_Val0 and RSP\_Val1 are not in use.

If error code is `READER_ERROR` or `NO_CARD_DETECTED`, device answer with `RSP_EXT` packet of 3 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents execution time of command

3<sup>rd</sup> byte is checksum.

In other cases, device answer with `RSP_EXT` packet of 5 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents error code of operation ( $b2 * 256 + b1$ )

3<sup>rd</sup> and 4<sup>th</sup> bytes represents execution time of command

5<sup>th</sup> byte is checksum.

### ***DESFIRE\_WRITE\_RECORD (0x98)***

For uFR PLUS devices only.

Function allows to write data to a record in a Linear Record File or Cyclic Record File. Write command requires a preceding authentication either with the key specified for Write or Read&Write access.

From firmware version 5.0.32 Desfire Light tag support

From firmware version 5.0.38 Transaction MAC for Desfire Light and Desfire EV2 support

(Firmware version from 5.0.xx)

$CMD\_Par0 = KEY\_TYPE \ll 4$  and  $CMD\_Par1 = 0$

1<sup>st</sup> byte of the  $CMD\_EXT$  is 1 if uses internal key, or 0 if uses external key

2<sup>nd</sup> byte of the  $CMD\_EXT$  contains ordinal number of internal key, or 0 if uses external key

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains key (for AES and 2K3DES all key bytes, for DES 8 key bytes and 8 zeros, for 3K3DES first 16 key bytes)

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is application key number for writing

23<sup>rd</sup> byte is ID of file (0 – 31)

24<sup>th</sup> byte is 1 if authentication required, or 0 if no need the authentication

25<sup>th</sup> and 26<sup>th</sup> bytes represents start position for write operation within file

27<sup>th</sup> and 28<sup>th</sup> bytes represents number of data to be write

29<sup>th</sup> byte is communication settings

array from 30<sup>th</sup> to 30 + block size number of data for writing contains maximal 160 data for writing

(for AES, DES and 2K3DES) (31 + block size) byte is checksum

(for 3K3DES) array from byte (31 + block size) to byte (38 + block size) contains last 8 key bytes, and byte (39 + block size) contains checksum

(uFR CS with SAM and firmware from versions 5.100.xx)

CMD\_Par0 = (KEY\_TYPE << 4) and CMD\_Par1 = 0

1<sup>st</sup> byte of the CMD\_EXT is 2 (using key into SAM)

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of key into SAM (0 -127)

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains 16 zeros

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is application key number for writing

23<sup>rd</sup> byte is ID of file (0 – 31)

24<sup>th</sup> byte is 1 if authentication required, or 0 if no need the authentication

25<sup>th</sup> and 26<sup>th</sup> bytes represents start position for write operation within file

27<sup>th</sup> and 28<sup>th</sup> bytes represents number of data to be write

29<sup>th</sup> byte is communication settings

array from 30<sup>th</sup> to 30 + block size number of data for writing contains maximal 160 data for writing

(31 + block size) byte is checksum

(Firmware version from 5.0.38)

tmc\_file = 0 -> Transaction MAC is not used

tmc\_file = 1 -> Transaction MAC is used Reader ID is not used

tmc\_file = 3 -> Transaction MAC is used Reader ID is used

CMD\_Par0 = KEY\_TYPE << 4 and CMD\_Par1 = tmc\_file

1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal key, or 0 if uses external key

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal key, or 0 if uses external key

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains key (for AES and 2K3DES all key bytes, for DES 8 key bytes and 8 zeros, for 3K3DES first 16 key bytes)

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is application key number for writing

23<sup>rd</sup> byte is ID of file (0 – 31)

24<sup>th</sup> byte is 1 if authentication required, or 0 if no need the authentication

25<sup>th</sup> and 26<sup>th</sup> bytes represents start position for write operation within file

27<sup>th</sup> and 28<sup>th</sup> bytes represents number of data to be write

29<sup>th</sup> byte is communication settings

array from 30<sup>th</sup> to 30 + block size number of data for writing contains maximal 160 data for writing

(for AES, DES and 2K3DES) (31 + block size) byte is checksum

(for 3K3DES) array from byte (31 + block size) to byte (38 + block size) contains last 8 key bytes, and byte (39 + block size) contains checksum

(uFR CS with SAM and firmware from version 5.100.38)

tmc\_file = 0 -> Transaction MAC is not used

tmc\_file = 1 -> Transaction MAC is used Reader ID is not used

tmc\_file = 3 -> Transaction MAC is used Reader ID is used

CMD\_Par0 = (KEY\_TYPE << 4) and CMD\_Par1 = tmc\_file

1<sup>st</sup> byte of the CMD\_EXT is 2 (using key into SAM)

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of key into SAM (0 -127)

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains 16 zeros

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is application key number for writing

23<sup>rd</sup> byte is ID of file (0 – 31)

24<sup>th</sup> byte is 1 if authentication required, or 0 if no need the authentication

25<sup>th</sup> and 26<sup>th</sup> bytes represents start position for write operation within file

27<sup>th</sup> and 28<sup>th</sup> bytes represents number of data to be write

29<sup>th</sup> byte is communication settings

array from 30<sup>th</sup> to 30 + block size number of data for writing contains maximal 160 data for writing

(31 + block size) byte is checksum

If error code is READER\_ERROR or NO\_CARD\_DETECTED, device answer with RSP\_EXT packet of 3 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents execution time of command

3<sup>rd</sup> byte is checksum.

In other cases, device answer with RSP\_EXT packet of 5 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents error code of operation (b2 \* 256 + b1)

3<sup>rd</sup> and 4<sup>th</sup> bytes represents execution time of command

5<sup>th</sup> byte is checksum.

From version 5.0.38. if tmc\_file > 0

1<sup>st</sup> and 2<sup>nd</sup> bytes represents error code of operation (b2 \* 256 + b1)

3<sup>rd</sup> and 4<sup>th</sup> bytes represents execution time of command

5th to 20th bytes represent Reader ID

21st to 36th bytes represent Previous Encrypted Reader ID

37th to 40th bytes represent Transaction MAC counter

41st to 48th bytes represent Transaction MAC

49th byte is checksum.

### **DESFIRE\_READ\_RECORDS (0x99)**

For uFR PLUS devices only.

Function allows to read data from a record in a Linear Record File or Cyclic Record File. Read command requires a preceding authentication either with the key specified for Write or Read&Write access.

From firmware version 5.0.32 Desfire Light tag support

CMD\_Par0 = KEY\_TYPE << 4 and CMD\_Par1 = 0

1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal key, or 0 if uses external key

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal key, or 0 if uses external key

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains key (for AES and 2K3DES all key bytes, for DES 8 key bytes and 8 zeros, for 3K3DES first 16 key bytes)

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is application key number for reading

23<sup>rd</sup> byte is ID of file (0 – 31)

24<sup>th</sup> byte is 1 if authentication required, or 0 if no need the authentication

25<sup>th</sup> and 26<sup>th</sup> bytes represents start position for read operation within file

27<sup>th</sup> and 28<sup>th</sup> bytes represents number of records to be read

29<sup>th</sup> byte is communication settings

30<sup>th</sup> and 31<sup>st</sup> bytes represents size of record

(for AES, DES and 2K3DES) 32<sup>nd</sup> byte contains checksum

(for 3K3DES) array from byte 32 to byte 39 contains last 8 key bytes, and byte 40 contains checksum

(uFR CS with SAM and firmware versions 5.100.xx)

CMD\_Par0 = (KEY\_TYPE << 4) and CMD\_Par1 = 0

1<sup>st</sup> byte of the CMD\_EXT is 2 (using key into SAM)

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of key into SAM (0 -127)

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains 16 zeros

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is application key number for reading

23<sup>rd</sup> byte is ID of file (0 – 31)

24<sup>th</sup> byte is 1 if authentication required, or 0 if no need the authentication

25<sup>th</sup> and 26<sup>th</sup> bytes represents start position for read operation within file

27<sup>th</sup> and 28<sup>th</sup> bytes represents number of records to be read

29<sup>th</sup> byte is communication settings

30<sup>th</sup> and 31<sup>st</sup> bytes represents size of record

32<sup>nd</sup> byte contains checksum

Reading the data is specific and is done in a loop. Reads one data, and if it is 0, then reads another that indicates how much data follows in the package. This is repeated until the required amount of data read. If the first data is different from 0, then reader will be sent standard response.

RSP\_Val0 and RSP\_Val1 are not in use.

If error code is `READER_ERROR` or `NO_CARD_DETECTED`, device answer with `RSP_EXT` packet of 3 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents execution time of command

3<sup>rd</sup> byte is checksum.

In other cases, device answer with `RSP_EXT` packet of 5 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents error code of operation ( $b2 * 256 + b1$ )

3<sup>rd</sup> and 4<sup>th</sup> bytes represents execution time of command

5<sup>th</sup> byte is checksum.

### **DESFIRE\_CLEAR\_RECORD (0x6D)**

For uFR PLUS devices only.

Function allows to reset a Linear Record File or Cyclic Record file to the empty state. Clear command requires a preceding authentication with the key specified for Read&Write access.

From firmware version 5.0.32 Desfire Light tag support

From firmware version 5.0.38 Transaction MAC for Desfire Light and Desfire EV2 support

(Firmware version 5.0.xx)

`CMD_Par0 = KEY_TYPE << 4` and `CMD_Par1 = 0`

1<sup>st</sup> byte of the `CMD_EXT` is 1 if uses internal key, or 0 if uses external key

2<sup>nd</sup> byte of the `CMD_EXT` contains ordinal number of internal key, or 0 if uses external key

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of `CMD_EXT` contains key (for AES and 2K3DES all key bytes, for DES 8 key bytes and 8 zeros, for 3K3DES first 16 key bytes)

array from 19<sup>th</sup> to 21<sup>st</sup> byte of `CMD_EXT` contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is ID of file that will be deleted (0 – 31)

23<sup>rd</sup> byte is 1 if authentication required, or 0 if no need the authentication

(for AES, DES and 2K3DES) 24<sup>th</sup> byte contains checksum

(for 3K3DES) array from byte 24 to byte 31 contains last 8 key bytes, and byte 32 contains checksum

(uFR CS with SAM and firmware versions 5.100.xx)

`CMD_Par0 = (KEY_TYPE << 4)` and `CMD_Par1 = 0`

1<sup>st</sup> byte of the `CMD_EXT` is 2 (using key into SAM)

2<sup>nd</sup> byte of the `CMD_EXT` contains ordinal number of key into SAM (0 -127)

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of `CMD_EXT` contains 16 zeros

array from 19<sup>th</sup> to 21<sup>st</sup> byte of `CMD_EXT` contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is ID of file that will be deleted (0 – 31)

23<sup>rd</sup> byte is 1 if authentication required, or 0 if no need the authentication

24<sup>th</sup> byte contains checksum

(Firmware version from 5.0.38)

tmc\_file = 0 -> Transaction MAC is not used

tmc\_file = 1 -> Transaction MAC is used Reader ID is not used

tmc\_file = 3 -> Transaction MAC is used Reader ID is used

CMD\_Par0 = KEY\_TYPE << 4 and CMD\_Par1 = tmc\_file

1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal key, or 0 if uses external key

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal key, or 0 if uses external key

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains key (for AES and 2K3DES all key bytes, for DES 8 key bytes and 8 zeros, for 3K3DES first 16 key bytes)

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is ID of file that will be deleted (0 – 31)

23<sup>rd</sup> byte is 1 if authentication required, or 0 if no need the authentication

24<sup>th</sup> byte is Application key number

(for AES, DES and 2K3DES) 25<sup>th</sup> byte contains checksum

(for 3K3DES) array from byte 25 to byte 32 contains last 8 key bytes, and byte 33 contains checksum

(uFR CS with SAM and firmware versions 5.100.xx)

tmc\_file = 0 -> Transaction MAC is not used

tmc\_file = 1 -> Transaction MAC is used Reader ID is not used

tmc\_file = 3 -> Transaction MAC is used Reader ID is used

CMD\_Par0 = (KEY\_TYPE << 4) and CMD\_Par1 = tmc\_file

1<sup>st</sup> byte of the CMD\_EXT is 2 (using key into SAM)

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of key into SAM (0 -127)

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains 16 zeros

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is ID of file that will be deleted (0 – 31)

23<sup>rd</sup> byte is 1 if authentication required, or 0 if no need the authentication

24<sup>th</sup> byte contains checksum

RSP\_Val0 and RSP\_Val1 are not in use.

If error code is READER\_ERROR or NO\_CARD\_DETECTED, device answer with RSP\_EXT packet of 3 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents execution time of command

3<sup>rd</sup> byte is checksum.

In other cases, device answer with RSP\_EXT packet of 5 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents error code of operation ( $b2 * 256 + b1$ )

3<sup>rd</sup> and 4<sup>th</sup> bytes represents execution time of command

5<sup>th</sup> byte is checksum.

From version 5.0.38. if tmc\_file > 0

1<sup>st</sup> and 2<sup>nd</sup> bytes represents error code of operation ( $b2 * 256 + b1$ )



3<sup>rd</sup> and 4<sup>th</sup> bytes represents execution time of command  
5<sup>th</sup> to 20<sup>th</sup> bytes represent Reader ID  
21<sup>st</sup> to 36<sup>th</sup> bytes represent Previous Encrypted Reader ID  
37<sup>th</sup> to 40<sup>th</sup> bytes represent Transaction MAC counter  
41<sup>st</sup> to 48<sup>th</sup> bytes represent Transaction MAC  
49<sup>th</sup> byte is checksum.

### **DESFIRE\_CREATE\_TRANS\_MAC\_FILE (0xC2)**

From firmware version 5.0.38

Function allows to create TransactionMAC file.

CMD\_Par0 = KEY\_TYPE << 4 and CMD\_Par1 = 0

1<sup>st</sup> byte of the CMD\_EXT is 1 if uses internal key, or 0 if uses external key

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of internal key, or 0 if uses external key

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains key (for AES and 2K3DES all key bytes, for DES 8 key bytes and 8 zeros, for 3K3DES first 16 key bytes)

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is ID of file that will be created (0 – 31)

23<sup>rd</sup> byte is communication settings

24<sup>th</sup> and 25<sup>th</sup> bytes represented access rights for read, write, read&write and changing  
(byte 24 = commit\_reader\_id\_key\_no (high 4 bits) | changing\_key\_no (low 4 bits)

byte 25 = read\_key\_no (high 4 bits) | 0x0F)

array from 26<sup>th</sup> to 41<sup>st</sup> byte contains Transaction MAC key

(for AES, DES and 2K3DES) 42<sup>nd</sup> byte contains checksum

(for 3K3DES) array from byte 42 to byte 49 contains last 8 key bytes, and byte 50 contains checksum

(uFR CS with SAM and firmware versions 5.100.38)

CMD\_Par0 = (KEY\_TYPE << 4) and CMD\_Par1 = 0

1<sup>st</sup> byte of the CMD\_EXT is 2 (using key into SAM)

2<sup>nd</sup> byte of the CMD\_EXT contains ordinal number of key into SAM (0 -127)

array from 3<sup>rd</sup> to 18<sup>th</sup> byte of CMD\_EXT contains 16 zeros

array from 19<sup>th</sup> to 21<sup>st</sup> byte of CMD\_EXT contains AID (Application ID 3 bytes)

22<sup>nd</sup> byte is ID of file that will be created (0 – 31)

23<sup>rd</sup> byte is communication settings

24<sup>th</sup> and 25<sup>th</sup> bytes represented access rights for read, write, read&write and changing  
(byte 24 = commit\_reader\_id\_key\_no (high 4 bits) | changing\_key\_no (low 4 bits)

byte 25 = read\_key\_no (high 4 bits) | 0x0F)

array from 26<sup>th</sup> to 41<sup>st</sup> byte contains Transaction MAC key

(for AES, DES and 2K3DES) 42<sup>nd</sup> byte contains checksum

(for 3K3DES) array from byte 42 to byte 49 contains last 8 key bytes, and byte 50 contains checksum

RSP\_Val0 and RSP\_Val1 are not in use.

If error code is `READER_ERROR` or `NO_CARD_DETECTED`, device answer with RSP\_EXT packet of 3 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents execution time of command

3<sup>rd</sup> byte is checksum.

In other cases, device answer with RSP\_EXT packet of 5 bytes.

1<sup>st</sup> and 2<sup>nd</sup> bytes represents error code of operation ( $b2 * 256 + b1$ )

3<sup>rd</sup> and 4<sup>th</sup> bytes represents execution time of command

5<sup>th</sup> byte is checksum.

From version 5.0.38. if `tmc_file > 0`

1<sup>st</sup> and 2<sup>nd</sup> bytes represents error code of operation ( $b2 * 256 + b1$ )

3<sup>rd</sup> and 4<sup>th</sup> bytes represents execution time of command

5th to 20th bytes represent Reader ID

21st to 36th bytes represent Previous Encrypted Reader ID

37th to 40th bytes represent Transaction MAC counter

41st to 48th bytes represent Transaction MAC

49th byte is checksum.

## **COMMANDS FOR MIFARE PLUS CARDS**

### ***MFP\_FIRST\_AUTHENTICATE (0x6A)***

Function is used for optional authentication with AES key when the card is in security level 1 and for switching to the security level 3.

CMD\_Par0 is authentication mode (`RKA_AUTH1A=0x00` or `PK_AUTH1A_AES=0x80`)

CMD\_Par1 is ordinal number of AES key from reader (0 - 15)

CMD\_EXT

1<sup>st</sup> and 2<sup>nd</sup> bytes represents card key address

array from 3<sup>rd</sup> to 18<sup>th</sup> byte contains AES key

19<sup>th</sup> byte is checksum

The RSP\_EXT is not in use

### **Example:**

Switch to security level 3 from security level 1. AES key must be equivalent with key entered into SL3 switch key register during personalization of card, for example key number 4 stored into reader.



```

CMD          55 6A AA 13 00 04 89
ACK          AC 6A CA 13 00 04 22
CMD_EXT     03 90 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 9A
    (internal key uses so AES key bytes may have any value (all 00))
RSP          DE 6A ED 00 00 00 60
    
```

### **MFP\_CHANGE\_REG\_KEY(0x6B)**

Function is used for registers or keys changing when the card is in security level 3.  
 CMD\_Par0 is authentication mode (RKA\_AUTH1A=0x00 or PK\_AUTH1A\_AES=0x80)  
 CMD\_Par1 is ordinal number of AES key from reader (0 - 15)  
 CMD\_EXT

- 1<sup>st</sup> and 2<sup>nd</sup> bytes represents card key or register address
- array from 3<sup>rd</sup> to 18<sup>th</sup> byte contains new AES key or register data
- 19<sup>th</sup> and 20<sup>th</sup> bytes represents card key for authentication address
- array from 21<sup>st</sup> to 36<sup>th</sup> byte contains AES key for authentication
- 37<sup>th</sup> byte is checksum

(uFR CS with SAM and firmware versions 5.100.xx)

- CMD\_Par0 is authentication mode (SAM\_KEY\_AUTH1A = 0x10)
- CMD\_Par1 is ordinal number of AES key for authentication from SAM (1 - 127)
- CMD\_EXT

- 1<sup>st</sup> and 2<sup>nd</sup> bytes represents card key or register address
- array from 3<sup>rd</sup> to 18<sup>th</sup> byte contains 16 zeros or register data
- 19<sup>th</sup> and 20<sup>th</sup> bytes represents card key for authentication address
- 21<sup>st</sup> byte is ordinal number of new AES key from SAM (1 - 127)
- 22<sup>nd</sup> byte is checksum

The RSP\_EXT is not in use

### **Example:**

Change the configuration AES key. Authenticate with same key. Old key is 0x11111111111111111111111111111111, new key is 0x22222222222222222222222222222222.

```

CMD          55 6B AA 25 80 00 36
ACK          AC 6B CA 25 80 00 AF
CMD_EXT     01 90 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 01 90
    11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 07
RSP          DE 6B ED 00 00 00 5F
    
```

### **MFP\_GET\_UID(0x6C)**

Function is used to read card UID when the Random ID enabled. VC polling ENC, and VC polling MAC key entered during personalization. These keys use in card UID reading process.

CMD\_Par0 is authentication mode (RKA\_AUTH1A=0x00 or PK\_AUTH1A\_AES=0x80)

CMD\_Par1 is 0

if authentication mode is PK\_AUTH1A\_AES

CMD\_EXT

array from 1<sup>st</sup> to 16<sup>th</sup> byte contains VC polling ENC key

array from 17<sup>th</sup> to 32<sup>nd</sup> byte contains VC polling MAC key

33<sup>rd</sup> byte is checksum

else if authentication mode is RKA\_AUTH1A

CMD\_EXT

1<sup>st</sup> byte is ordinal number of internal key contain VC polling ENC key

2<sup>nd</sup> byte is ordinal number of internal key contain VC polling MAC key

3<sup>rd</sup> byte is checksum

(uFR CS with SAM and firmware versions 5.100.xx)

CMD\_Par0 is authentication mode (SAM\_KEY\_AUTH1A = 0x10)

1<sup>st</sup> byte is ordinal number of SAM key contain VC polling ENC key

2<sup>nd</sup> byte is ordinal number of SAM key contain VC polling MAC key

3<sup>rd</sup> byte is checksum

RSP\_EXT

1<sup>st</sup> byte is UID length (7 or 4)

array from 2<sup>nd</sup> to 2 + length byte contains card UID

### Example:

VC polling ENC key is 0x22222222222222222222222222222222, VC polling MAC key is 0x11111111111111111111111111111111.

```
CMD      55 6C AA 21 80 00 35
ACK      AC 6C CA 21 80 00 AC
CMD_EXT  22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22
11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 07
RSP      DE 6C ED 09 00 00 66
RSP_EXT  07 01 02 03 04 05 06 07 0E
```

## COMMANDS FOR NT4H CARDS

### From firmware version 5.0.32

Supported cards are NT4H1321 (NTAG 413 DNA), NT4H2421Gx (NTAG 424 DNA), and NT4H2421Tx (NTAG 424 DNA TT) card.

NTAG 424 DNA is fully compliant with the NFC Forum Type 4 Tag IC specification

(Certification ID: 58562), with the contactless proximity protocol according to ISO/IEC14443-4 and the ISO/IEC 7816-4 based file system and command frames.

NTAG 424 DNA TT comes with smart status awareness, detecting the status of a tamper loop. Same command is used for the Desfire Light tag in a couple of cases.

### **NT4H\_COMMON\_CMD (0xB3)**

This command is used for various NT4H commands.

### **NT4H\_SET\_GLOBAL\_PARAMETERS**

Command sets file number, key number, and communication mode, before the using functions for reading and writing data into cards which are used for NTAG 2xx cards. This makes it possible to use existing functions for linear reading and writing.

CMD\_PAR0 = 1, CMD\_PAR1 = 0

CMD\_EXT

1st byte is file number (NTAG 413 - 1 or 2, NTAG 424 - 1 to 3)

2nd byte is application key number (NTAG 413 - 0 to 2, NTAG 424 - 0 to 4)

3rd byte is communication mode of selected file (0 - plain, 1 - macked, 3 - enciphered)

4th byte is checksum

The RSP\_EXT is not in use

#### **Example:**

File number = 2, key number = 0, communication mode = 0 (plain)

```
CMD      55 B3 AA 04 01 00 50
ACK      AC B3 CA 04 01 00 D7
CMD_EXT  02 00 00 09
RSP      DE B3 ED 00 00 00 87
```

### **NT4H\_CHANGE\_FILE\_SETTINGS**

The commands change the access parameters of an existing standard data file. Length of settings data, and its content may be various according to NXP documentation.

CMD\_PAR0 = 2, CMD\_PAR1 = 0

CMD\_EXT

1st byte defines internal key using (1 - reader key, 0 - provided key)

2nd byte is ordinal AES key number into reader (0 - 15)

array 3 - 18 is provided AES key

19th byte is card type (NT4H cards = 1, Desfire light = 2)

20th byte is file number (NTAG 413 - 1 or 2, NTAG 424 - 1 to 3, Desfire light 0, 1, 3, 4, 15 or 31)

21st byte is application key number (NTAG 413 - 0 to 2, NTAG 424 - 0 to 4)

22nd byte is communication mode (3 - enciphered)

23rd byte is settings data length

array of settings data length bytes

last byte is checksum

The RSP\_EXT is not in use

**Example:**

File number = 2, current change key number = 0, read key number = 2, write key number = 3, read/write key number = 3, new change key number = 0, communication mode = 0 (plain), authentication mode provided, AES key 16x 0x00.

```
CMD          55 B3 AA 1B 02 00 5C
ACK          AC B3 CA 1B 02 00 D3
CMD_EXT     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 02 00
03 03 00 30 23 17
RSP          DE B3 ED 00 00 00 87
```

**NT4H\_SET\_CARD\_CONFIGURATION**

Command set card configuration. Authentication with master key required. Length of configuration data, and its content may be various according to NXP documentation.

CMD\_PAR0 = 3, CMD\_PAR1 = 0

CMD\_EXT

1st byte defines internal key using (1 - reader key, 0 - provided key)

2nd byte is ordinal AES key number into reader (0 - 15)

array 3 - 18 is provided AES key

19th byte is card type (NT4H cards = 1)

20th byte is card command options according to NXP documentation.

21st byte is configuration data length

array of configuration data length

last byte is checksum

The RSP\_EXT is not in use

**Example:**

Set Random ID. Option = 0 (PICC configuration), Authentication with provided master key.

```
CMD          55 B3 AA 17 03 00 5F
ACK          AC B3 CA 17 03 00 C8
CMD_EXT     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 01
02 09
RSP          DE B3 ED 00 00 00 87
```

**NT4H\_CHANGE\_KEY**

Command changes AES key. Authentication with the application master key is required.

CMD\_PAR0 = 4, CMD\_PAR1 = 0

CMD\_EXT

1st byte defines internal key using (1 - reader key, 0 - provided key)

2nd byte is ordinal AES key number into reader (0 - 15)

array 3 - 18 is provided AES key

byte 19 is application key number which will be changed (NTAG 413 - 0 to 2, NTAG 424 - 0 to 4)

array 20 - 35 is new AES key

array 36 - 52 is old AES key if application key number is different from 0

byte 53 is checksum.

The RSP\_EXT is not in use

**Example:**

Key number 2 changing. Master AES key is 16 x 0x00. New AES key is 16 x 0x11. Old AES key is 16 x 0x00. Provided key authentication mode.

```

CMD      55 B3 AA 34 04 00 83
ACK      AC B3 CA 34 04 00 EC
CMD_EXT  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 02 11 11
11 11 11 11 11 11 11 11 11 11 11 11 11 11 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 09
RSP      DE B3 ED 00 00 00 87
    
```

**NT4\_GET\_UID**

Command returns card UID if Random ID activated. Valid authentication is required.

CMD\_PAR0 = 5, CMD\_PAR1 = 0

CMD\_EXT

1st byte defines internal key using (1 - reader key, 0 - provided key)

2nd byte is ordinal AES key number into reader (0 - 15)

array 3 - 18 contains provided AES key

byte 19 is application key number (NTAG 413 - 0 to 2, NTAG 424 - 0 to 4)

RSP\_EXT

array 1 - 7 contains UID

8th byte is checksum.

**Example:**

Provided key authentication mode. Key number = 2. AES key is 16 x 0x11.

```

CMD      55 B3 AA 14 05 00 64
ACK      AC B3 CA 14 05 00 CB
CMD_EXT  00 00 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 02 09
RSP      DE B3 ED 08 00 00 8F
RSP_EXT  04 5B A8 92 76 63 80 F7
    
```

**NT4H\_GET\_FILE\_SETTINGS**

Command returns file settings. Length of settings data may be various according to NXP documentation.

CMD\_PAR0 = 6, CMD\_PAR1 = 0

CMD\_EXT

1st and 2nd bytes are 0 (no authentication required)

3rd byte is card type (NT4H cards = 1, Desfire light = 2)

4th byte is file number (NTAG 413 - 1 or 2, NTAG 424 - 1 to 3, Desfire light 0, 1, 3, 4, 15 or 31)

RSP\_EXT

Settings data length bytes

the last byte is checksum.

**Example:**

File number = 2, File is in secure dynamic message mode.

**CMD** 55 B3 AA 05 06 00 56

**ACK** AC B3 CA 05 06 00 DD

**CMD\_EXT** 00 00 01 02 0A

**RSP** DE B3 ED 14 00 00 9B

**RSP\_EXT** 00 40 E0 EE 00 01 00 C1 FE 22 22 00 00 44 00 00 44 00 00 77

**NT4H\_GET\_SDM\_READING\_COUNTER**

Function supports retrieving of the current values of the SDM reading counter.

CMD\_PAR0 = 7, CMD\_PAR1 = 0

CMD\_EXT

1st byte defines internal key using (1 - reader key, 0 - provided key, 0xFF no authentication)

2nd byte is ordinal AES key number into reader (0 - 15)

array 3 - 18 is provided AES key

19th byte is file number (NTAG 413 - 1 or 2, NTAG 424 - 1 to 3)

20th byte is application key number (NTAG 413 - 0 to 2, NTAG 424 - 0 to 4)

21st byte is checksum

RSP\_EXT

array 1 - 3 value of counter (little endian)

byte 4 is checksum

**Example:**

Get SDM reading counter without authentication.

**CMD** 55 B3 AA 15 07 00 65

**ACK** AC B3 CA 15 07 00 CE

**CMD\_EXT** FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 02 00 04

**RSP** DE B3 ED 04 00 00 8B

**RSP\_EXT** 02 00 00 09

## DFL\_DELETE\_TRANSACTION\_MAC\_FILE

Command delete transaction MAC file.

**NOTE: Transaction MAC file exists by factory default. To use the operations with value file, and cyclic record file, this file must be deleted.**

CMD\_PAR0 = 8, CMD\_PAR1 = 0

CMD\_EXT

1st byte defines internal key using (1 - reader key, 0 - provided key)

2nd byte is ordinal AES key number into reader (0 - 15)

array 3 - 18 is provided AES key

19th byte is file number = 15

20th byte is checksum

RSP\_EXT not in use

## NT4H\_GET\_TT\_STATUS

Firmware version 5.0.43. NTAG 424 TT only.

Command supports retrieving of the permanent and current Tag Tamper Status.

CMD\_PAR0 = 9, CMD\_PAR1 = 0

CMD\_EXT

1st byte defines internal key using (1 - reader key, 0 - provided key, 0xFF - no authentication)

2nd byte is ordinal AES key number into reader (0 - 15)

array 3 - 18 is provided AES key

19th byte is tag tamper status key number (0 - 4)

20th byte is checksum

RSP\_EXT

1st byte is tag tamper permanent status

2nd byte is tag tamper current status

3rd byte is checksum

### Example:

Get tag tamper status. Authentication with provided key  
0x00000000000000000000000000000000. Tag tamper status key is 0.

```

CMD          55 B3 AA 14 09 00 58
ACK          AC B3 CA 14 09 00 CF
CMD_EXT      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 07
RSP          DE B3 ED 03 00 00 8A
RSP_EXT      43 43 07
    
```

## COMMANDS FOR READER SETTINGS

### *SET\_BAD\_SELECT\_NR\_MAX (0x3F)*

The function allows you to set the number of unsuccessful card selections before it can be considered that the card is not placed on the reader. Period between two card selections is approximately 10ms. Default value of this parameter is 20 i.e. 200ms. This parameter can be set in the range of 0 to 254.

The CMD\_EXT set is not in use.

CMD\_Par0 is bad select card number maximal

CMD\_Par1 = (CMD\_Par0 xor A3) + 7

The RSP\_EXT is not in use

#### **Example:**

Bad select card maximal is 10

CMD\_Par0 = 0x0A, CMD\_Par1 = (0A xor A3) + 7 = B0

```

CMD          55 3F AA 00 0A B0 81 (send command 3F) , 81 checksum
RSP          DE 3F ED 00 00 00 13
    
```

### *GET\_BAD\_SELECT\_NR\_MAX(0x44)*

The function returns value of maximal unsuccessful card selections, which is set in reader.

The CMD\_EXT set is not in use.

CMD\_Par0 and CMD\_Par1 are 0

RSP\_EXT - 1<sup>st</sup> byte is maximal value of bad select card number

#### **Example:**

```

CMD          55 44 AA 00 00 00 C2 (send command 44) , C2 checksum
RSP          DE 44 ED 02 00 00 7C
RSP_EXT      0A 11 (number is 0x0A)
    
```



## **FUNCTIONS FOR THE READER LOW POWER MODE CONTROL**

### ***ENTER\_SLEEP\_MODE (0x46)***

Function allows the low power reader mode. Reader is in sleep mode. RF field is turned off. The reader is waiting for the command to return to normal working mode.

The CMD\_EXT set is not in use.

CMD\_Par0 and CMD\_Par1 are 0

The RSP\_EXT is not in use.

#### **Example:**

```
CMD      55 46 AA 00 00 00 C0 (send command 46) , C0 checksum
RSP      DE 46 ED 00 00 00 7C
```

### ***LEAVE\_SLEEP\_MODE (0x47)***

Function allows return from low power reader mode to normal working mode.

The CMD\_EXT set is not in use.

CMD\_Par0 and CMD\_Par1 are 0

The RSP\_EXT is not in use.

From version 5.0.23 after the wake up byte sent, must wait 10 ms before the command sending.

#### **Example:**

```
WAKE UP BYTE  00
```

from version 5.0.23 wait 10 ms after the wake up byte sent

```
CMD      55 47 AA 00 00 00 BF (send command 47) , BF checksum
RSP      DE 47 ED 00 00 00 7B
```

### ***AUTO\_SLEEP\_SET (0x4D)***

supported from firmware version 3.8.18

**Command description:**

This function permanently set auto-sleep functionality of the device. Valid value for the CMD\_Par0 range is from 1 to 254 seconds. To permanently disable auto-sleep functionality use 0 or 0xFF for the CMD\_Par0 value.

The CMD\_EXT is not in use.

CMD\_Par1 are 0 (not in use).

The RSP\_EXT is not in use.

***AUTO\_SLEEP\_GET (0x4E)***

supported from firmware version 3.8.18

**Command description:**

This command returns permanently configured auto-sleep wait seconds.

The CMD\_EXT is not in use.

CMD\_Par0 and CMD\_Par1 are 0 (not in use).

The RSP\_EXT is not in use.

RSP\_Val0 containing configured auto-sleep wait seconds.

RSP\_Val1 is 0 (not in use).

**Commands for Reader NTAG Emulation Mode**

***WRITE\_EMULATION\_NDEF (0x4A)***

supported from firmware version 3.8.0

**Command description:**

Command stores a message record for NTAG emulation mode into the reader EEPROM. The CMD\_EXT is used and contains an NDEF message for tag emulation mode. Maximum total size for emulated NDEF message is 144 bytes.

CMD\_Par0 and CMD\_Par1 are 0 (not in use).

1<sup>st</sup> and 2<sup>nd</sup> byte of the CMD\_EXT set contains length of the following NDEF message (parameter called ndef\_len) maximal length is 144 bytes.

The next ndef\_len bytes contains an NDEF message.

last byte of the CMD\_EXT set contains checksum

**Example:**

(NDEF message is URI type with "[www.d-logic.net](http://www.d-logic.net)" payload):

```

CMD          55 4A AA 16 00 00 AA
ACK          AC 4A CA 16 00 00 41
CMD_EXT      14 00 03 10 D1 01 0C 55 01 64 2D 6C 6F 67 69 63 2E 6E 65 74 FE
0E
RSP          DE 4A ED 00 00 00 80

```

**Possible error codes:**

WRITE\_VERIFICATION\_ERROR = 0x70

MAX\_SIZE\_EXCEEDED = 0x10

**Write emulation NDEF into reader RAM from firmware version 5.0.33**

Command store a message record for NTAG emulation mode in to the reader RAM. The CMD\_EXT is used and contains NDEF message for tag emulation mode. Maximum total size for emulated NDEF message is 1008 bytes. The data is not written into EEPROM of the reader, so they cannot be loaded after the reader reset. This command must be execute after reader reset to use the NTAG emulation.

CMD\_Par0 is 1 and CMD\_Par1 is 0.

1<sup>st</sup> and 2<sup>nd</sup> byte of the CMD\_EXT set contains length of the following NDEF message (parameter called ndef\_len) maximal length is 1008 bytes.

next part of ndef\_len (maximal part size is 240 bytes)

last byte of the CMD\_EXT set contains checksum

If you want to enter more than 240 bytes, then it is done in blocks of up to 240 bytes. After the first block of data reader sent 0xAD if necessary to receive more data, or 0xDD if no need more data, or at any error. When you receive 0xAD then sends a packet in which the first byte indicates how many bytes follow. When you receive 0xDD then follow standard response.

RSP\_Val0 and RSP\_Val1 are not in use.

**Example:**

NDEF message with maximal length of 1008 bytes. Type Text

**CMD**            55 4A AA F3 01 00 4E

**ACK**            AC 4A CA F3 01 00 E5

**CMD\_EXT\_1**

F0 03 03 FF 03 EB C1 01 00 00 03 E4 54 02 65 6E 33 34 35 36 37 38 39 30  
31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34  
35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38  
39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32  
33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36  
37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30  
31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34  
35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38  
39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32  
33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36  
37 38 9D

**ACK**            AD

**CMD\_EXT\_2**

F0 37 38 9D 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39  
30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33  
34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37  
38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31  
32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35  
36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39  
30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33  
34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37  
38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31  
32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35  
36

```
ACK          AD
CMD_EXT_3
F0 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39
30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33
34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37
38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31
32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35
36
ACK          AD
CMD_EXT_4
F0 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39
30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33
34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37
38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31
32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35
36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39
30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33
34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37
38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31
32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35
36
ACK          AD
CMD_EXT_5
30 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39
30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33
34
ACK          DD (NO MORE DATA)
RESP        DE 4A ED 00 00 00 80
```

## **TAG\_EMULATION\_START (0x48)**

supported from firmware version 3.8.0

Put the reader permanently in a NDEF tag emulation mode. Only way for a reader to exit from this mode is to receive the TAG\_EMULATION\_STOP command.

In this mode, the reader can only answer to the following commands:

```
WRITE_EMULATION_NDEF (0x4A)
TAG_EMULATION_STOP (0x49)
TAG_EMULATION_START (0x48)
GET_READER_TYPE (0x10)
GET_READER_SERIAL (0x11)
GET_FIRMWARE_VERSION (0x29)
GET_HARDWARE_VERSION (0x2A)
GET_BUILD_NUMBER (0x2B)
GET_SERIAL_NUMBER (0x40)
```

Issuing another commands in this mode, results with the following error code:

```
FORBIDDEN_IN_TAG_EMULATION_MODE = 0x90
```

CMD\_Par0 and CMD\_Par1 are 0 (not in use).

### **Possible error codes:**

```
WRITE_VERIFICATION_ERROR = 0x70
```

*(command resulting in a direct write to a device non-volatile memory)*

### **Example:**

```
CMD      55 48 AA 00 00 00 BE
RSP      DE 48 ED 00 00 00 82
```

### **TAG emulation into RAM start from firmware version 5.0.33**

Put the reader permanently in a NDEF tag in RAM emulation mode. Only way for a reader to exit from this mode is to receive the TAG\_EMULATION\_STOP command, or by reader reset. Use the command GET\_READER\_STATUS to check if the reader is still in emulation mode (maybe the reader was reset for some reason).

CMD\_Par0 is 1 and CMD\_Par1 is 0.

### **TAG\_EMULATION\_STOP (0x49)**

supported from firmware version 3.8.0

Allows the reader permanent exit from a NDEF tag emulation mode.

CMD\_Par0 and CMD\_Par1 are 0 (not in use).

#### **Possible error codes:**

**WRITE\_VERIFICATION\_ERROR = 0x70**

(command resulting in a direct write to a device non-volatile memory)

#### **Example:**

```
CMD      55 49 AA 00 00 00 BD
RSP      DE 49 ED 00 00 00 81
```

### **TAG emulation into RAM stop from firmware version 5.0.33**

CMD\_Par0 is 1 and CMD\_Par1 is 0.

### **T2T\_MIRROR\_EMULATION (0xE7)**

supported from firmware version 5.0.61

Command supports T2T ASCII mirror functionality into card emulation mode.

#### **TAG emulation T2T mirror counter with reset counter**

Command enables the 24 bit NFC counter. Counter increased by the first valid READ command in the NTAG emulation mode, after the external RF field detected. Counter is represented in 6 bytes of ASCII code, when the NDEF message is read. For example if the counter value is 0x56, it will be represented as 000056, at the end of the NDEF message. Position of the counter mirror start byte must be entered as a function parameter. This is the absolute position in the card emulation data array.

#### **Counter value sets to 0.**

CMD\_Par0 is 1 and CMD\_Par1 is 0.

CMD\_EXT 1st byte is 0xEA

CMD\_EXT 2nd byte is a low byte of position of the counter.

CMD\_EXT 3rd byte is a high byte of position of the counter.

CMD\_EXT 4th to 7th bytes are counter value 0.

CMD\_EXT 8th byte is checksum.

#### **Example:**

Position of the counter is 0x25

```
CMD      55 E7 AA 09 01 00 17
ACK      AC E7 CA 09 01 00 90
CMD_EXT  EA 25 00 00 00 00 00 00 D6
RSP      DE E7 ED 00 00 00 DB
```

### TAG emulation T2T mirror counter without reset counter

Same as the previous command, except the reset the counter. **Counter keeps the current value.**

```
CMD_EXT 4th byte is 0xFF
CMD_EXT 5th byte is 0xFF
CMD_EXT 6th byte is 0xFF
CMD_EXT 7th byte is 0x00
```

#### Example:

Position of the counter is 0x25

```
CMD      55 E7 AA 09 01 00 17
ACK      AC E7 CA 09 01 00 90
CMD_EXT  EA 25 00 FF FF FF 00 00 37
RSP      DE E7 ED 00 00 00 DB
```

### TAG emulation T2T mirror counter disabled

Command disables the 24 bit NFC counter.

CMD\_Par0 is 1 and CMD\_PAR1 is 0.

CMD\_EXT 1th to 7th bytes are 0.

CMD\_EXT 8th byte is checksum.

#### Example:

```
CMD      55 E7 AA 09 01 00 17
ACK      AC E7 CA 09 01 00 90
CMD_EXT  00 00 00 00 00 00 00 00 07
RSP      DE E7 ED 00 00 00 DB
```

### Ad-Hoc emulation mode:

This mode enables user controlled emulation from the user application. There is “nfc-rfid-reader-sdk/ufr-examples-ad\_hoc\_emulation-c” console example written in C, using our uFCoder library (see uFR API). This example demonstrate usage of the uFCoder library functions that implement sending of the following commands:



## **AD\_HOC\_EMULATION\_START (0x76)**

supported from firmware version 3.9.34

Put uFR in emulation mode with ad-hoc emulation parameters (see. SET\_AD\_HOC\_EMULATION\_PARAMS and GET\_AD\_HOC\_EMULATION\_PARAMS). uFR stays in emulation mode until AD\_HOC\_EMULATION\_STOP command is sent or reader reset.

The CMD\_EXT set is not in use.

CMD\_Par0 and CMD\_Par1 are not in use.

The RSP\_EXT is not in use

### **Example:**

```
CMD      55 76 AA 00 AA CC F6
RSP      DE 76 ED 00 00 00 4C
```

## **AD\_HOC\_EMULATION\_STOP (0x77)**

supported from firmware version 3.9.34

Terminate uFR ad-hoc emulation mode.

The CMD\_EXT set is not in use.

CMD\_Par0 and CMD\_Par1 are not in use.

The RSP\_EXT is not in use

### **Example:**

```
CMD      55 77 AA 00 AA CC F5
RSP      DE 77 ED 00 00 00 4B
```

## **GET\_EXTERNAL\_FIELD\_STATE (0x9F)**

supported from firmware version 3.9.34

This command returns external field state when uFR is in ad-hoc emulation mode.

The CMD\_EXT set is not in use.

CMD\_Par0 and CMD\_Par1 are not in use.

RSP\_Val0 is 0 if external field isn't present or 1 if field is present.

RSP\_Val1 is not in use.

The RSP\_EXT is not in use

**Example:**

```
CMD      55 9F AA 00 AA CC 0D
RSP      DE 9F ED 00 01 00 B4
```

***GET\_AD\_HOC\_EMULATION\_PARAMS (0x9D)***

supported from firmware version 3.9.35

This command returns current ad-hoc emulation parameters. On uFR power on or reset ad-hoc emulation parameters are set back to their default values.

The CMD\_EXT set is not in use.

CMD\_Par0 and CMD\_Par1 are not in use.

RSP\_Val0 contains current ad-hoc threshold parameters. Default value is 0xF7.

RSP\_Val1 contains current ad-hoc receiver gain and RF level values of the RFCfgReg register (most significant bit of this value should be 0 all the time). Default value is 0x79.

The RSP\_EXT is not in use

**Example:**

```
CMD      55 9D AA 00 AA CC 0B
RSP      DE 9D ED 00 F7 79 27
```

***SET\_AD\_HOC\_EMULATION\_PARAMS (0x9E)***

supported from firmware version 3.9.35

This command set ad-hoc emulation parameters. On uFR power on or reset ad-hoc emulation parameters are set back to their default values.

The CMD\_EXT set is not in use.

CMD\_Par0 contains current ad-hoc threshold parameters. Default value is 0xF7.

CMD\_Par1 contains current ad-hoc receiver gain and RF level values of the RFCfgReg register (most significant bit of this value should be 0 all the time). Default value is 0x79.

**Example:**

```
CMD      55 9E AA 00 F7 79 F6
RSP      DE 9E ED 00 00 00 B4
```

***SET\_SPEED\_PERMANENTLY (0x4B)***

supported from firmware version 3.8.4

Permanently set the requested transceive data rates between reader and ISO14443 – 4A card / tag.

CMD\_EXT set not in use.

CMD\_Par0 containing requested transmit speed constant

CMD\_Par1 containing requested receive speed constant

The RSP\_EXT not in use.

Valid speed constants are:

<i>Const</i>	<i>Requested speed</i>
0	106 kbps (default)
1	212 kbps
2	424 kbps

**Possible error codes:**

`WRITE_VERIFICATION_ERROR = 0x70`

(command resulting in a direct write to a device non-volatile memory)

**Example:**

```
CMD      55 4B AA 00 02 02 BB
RSP      DE 4B ED 00 00 00 7F
```

**GET\_SPEED\_PARAMETERS (0x4C)**

supported from firmware version 3.8.4

This command returns permanently configured transceive data rates between reader and ISO14443 – 4A card / tag.

CMD\_EXT set not in use.

The RSP\_EXT not in use.

RSP\_Val0 containing configured transmit speed constants

RSP\_Val1 containing configured receive speed constants

Valid speed constants are:

<i>Const</i>	<i>Configured speed</i>
0	106 kbps (default)
1	212 kbps
2	424 kbps

**Example:**

```
CMD      55 4C AA 00 00 00 BA
RSP      DE 4C ED 00 02 02 86
```

**Support for ISO 14443-4 protocol commands**

**Basic commands**

***SET\_ISO14433\_4\_MODE (0x93)***

supported from firmware version 3.9.36

After issuing this command, ISO 14443-4 tag in a field will be selected and RF field polling will be stopped. Furthermore all the other ISO 14443-4 protocol commands can be issued in a sequence (including APDU\_TRANSCEIVE). Last command in those sequences should be S\_BLOCK\_DESELECT.

**Example:**

```
CMD      55 93 AA 00 AA CC 11
RSP      DE 93 ED 00 00 00 A7
```

***SET\_ISO14443\_4\_DL\_STORAGE (0x97)***

supported from firmware version 4.0.20

After issuing this command, ISO 14443-4 tag in a field will be selected and RF field polling will be stopped. Furthermore all the other ISO 14443-4 protocol commands can be issued in a sequence (including APDU\_TRANSCEIVE). Last command in those sequences should be S\_BLOCK\_DESELECT.

This command is identical to SET\_ISO14433\_4\_MODE with a difference that enables fast reading mechanism for a JC DL Storage cards using extended APDU format for case 2E in APDU\_TRANSCEIVE command (APDU in format: CLA, INS, P1, P2, 0x00, 0x7F, 0xFF) where 0x7F, 0xFF bytes represents maximum of 0x7FFF = 32767 bytes (big endian convention is in use in this case). When C-APDU is formatted in that way, I.E. using case 2E APDU format, after 7 bytes of the RSP packet will be two bytes which will define size in bytes (big endian convention) of the following data stream.

**Example:**

```
CMD      55 97 AA 00 AA CC 26
RSP      DE 97 ED 00 00 00 AB
```

***I\_BLOCK\_TRANSCEIVE (0x90)***

supported from firmware version 3.9.36

Used to convey information for use by the application layer.

CMD\_Par0 contains command specific flags (0x0C additional chained i block , 0x04 single i block)

CMD\_Par1 containing timeout value in [ms]

CMD\_EXT contains an i-block body.

RSP\_EXT contains an i-block response.

### ***R\_BLOCK\_TRANSCEIVE (0x91)***

**supported from firmware version 3.9.36**

Used to convey positive or negative acknowledgements. An R-block never contains an INF field.

The acknowledgement relates to the last received block.

CMD\_Par0 contains acknowledge flag (1 = ACK, 0 = NOT ACK)

CMD\_Par1 containing timeout value in [ms]

CMD\_EXT not in use.

RSP\_EXT contains an i-block response.

### ***S\_BLOCK\_DESELECT (0x92)***

**supported from firmware version 3.9.36**

Issue this command to deselect tag and restore RF field polling. This command is mandatory at the end of any

ISO 14443-4 protocol command sequence.

#### **Example:**

<b>CMD</b>	<b>55 92 AA 00 64 00 10</b>
<b>RSP</b>	<b>DE 92 ED 00 00 00 A8</b>

## **Support for APDU commands in ISO 14443-4 tags**

### ***APDU\_TRANSCEIVE (0x94)***

**supported from firmware version 3.9.39**

The majority of the ISO 14443-4 tags supports the APDU message structure according to ISO/IEC 7816-4. For more details you have to check the manual for the tags that you plan to use.

Issuing APDU\_TRANSCEIVE command you will send C-APDU to ISO 14443-4 tag selected using SET\_ISO14433\_4\_MODE. After successfully executing APDU\_TRANSCEIVE command uFR returns a byte array which contains R-APDU including data field (body) followed by the trailer (SW1 and SW2 APDU status bytes).

CMD\_Par0 not in use

CMD\_Par1 containing timeout value in [ms]

CMD\_EXT contains C-APDU (i.e. {CLA, INS, P0, P1, Lc, ... Nc bytes ... , Le} )

RSP\_EXT contains R-APDU including a data field (body) followed by the trailer (SW1 and SW2 APDU status bytes).

### Short APDU Support

The transceiver communication buffer in uFR devices is accommodated so that it can support all standard i.e short APDU commands and their responses.

The Short APDU command is characterized by the fact that the C-APDU (sent to the uFR device) can have a maximum of 261 bytes (4 bytes of C-APDU header: {CLS, INS, P1, P2}, 1 byte Lc, data bytes {maximum 255 bytes} and 1 Le byte). R-APDU, in the case of the short APDU, can have a maximum of 258 bytes (256 data bytes and 2 SW bytes at the end of the R-APDU stream).

The R-APDU must always contain a minimum of 2 SW bytes.

Since the maximum length of the C-APDU command must be encoded in at least 2 bytes (261 is greater than 255 which is the maximum value that can be encoded in one byte only), the uFR COM protocol has been extended to use the fifth byte of the CMD packet (CMD\_Par0) as a most significant byte (MSB) of the length of the following CMD\_EXT packet, while the least significant byte (LSB) - CMD\_EXT\_Length is located in the fourth byte of the CMD packet (Little Endian sequence). So, APDU\_TRANSCEIVE does not contain CMD\_Par0 and this parameter is replaced by CMD\_EXT\_Length\_MSB and the length of the next CMD\_EXT packet is encoded in bytes CMD\_EXT\_Length\_MSB and CMD\_EXT\_Length, which is a 16-bit word. The maximum length of the CMD\_EXT packet in the case of a short APDU command can be 262 bytes (261 bytes for C-APDU including one additional CheckXOR7 byte at the end of the stream).

In case of the short APDU, the CMD\_Par1 parameter must not be equal to APDU\_STREAM\_INDICATOR = 0x5A, which will be discussed later in the "Extended APDU support" chapter.

After receiving the ACK and sending the CMD\_EXT packet, RSP and RSP\_EXT packets are expected to be received. In general if the RSP packet was not received after 1.00 s since the last byte of the CMD\_EXT packet was sent, it can be considered that a communication timeout had occurred due to some critical error. However, there are many APDU commands whose execution time takes much longer than 1.00 s, so the uFR KEEP ALIVE mechanism is used here, which is active as long as the ISO14443-4 tag maintains a connection to the uFR reader according to "Half-duplex block transmission" (T = CL) protocol. The uFR KEEP ALIVE mechanism is described in the next chapter. So, as long as the APDU command is executed on the ISO 14443-4 tag, which lasts longer than the estimated timeout duration (cca 1.00s), uFR will send KEEP\_ALIVE packets. After successful execution of the APDU command, the RSP and RSP\_EXT packets are returned. Since the short R-APDU can have 258 bytes, which means that RSP\_EXT in this case has a length of 259 bytes (additional CheckXOR7 byte at the end of the stream is mandatory), the length of this packet must be encoded in at least 2 bytes similar to the CMD\_EXT packet length. Thus, in a similar way, the length of the RSP\_EXT packet is encoded in the bytes RSP\_EXT\_Length and RSP\_Val0, where RSP\_Val0 has the role of the RSP\_EXT\_Length\_MSB.

In case of the short APDU, the RSP\_Val1 command must not be equal to APDU\_STREAM\_INDICATOR = 0x5A, which will be discussed later in the “Extended APDU support” chapter.

### Short APDU Cases:

#### Case 1:

C-APDU Length is exactly 4 bytes. There are no C-APDU data bytes. After APDU command execution, only 2 SW bytes are expected within the R-APDU (APDU response).

C-APDU:

Byte1	Byte2	Byte3	Byte4
CLS	INS	P1	P2

R-APDU:

Byte1	Byte2
SW1	SW2

It is obvious that there is no extended variant of this APDU case.

#### Case 2 Short (2S):

C-APDU Length is exactly 5 bytes. There are no C-APDU data bytes.  $N_e$  can be from 1 to 256 which is encoded in the  $L_e$  field so, when the  $L_e = 0$ ,  $N_e$  is 256 and in all other cases  $N_e = L_e$  { $N_e = L_e > 0 ? L_e : 256$ ; // C operation for the  $N_e$  assignment}

After APDU command execution, maximum  $N_e$  bytes and additional two SW bytes are expected within the R-APDU (APDU response).  $N_e$  only means that the host device is able to receive so many bytes after the successful execution of the current APDU command. The actual number of response data bytes returned by the tag can be less than  $N_e$  or even 0.

C-APDU:

Byte1	Byte2	Byte3	Byte4	Byte5
CLS	INS	P1	P2	$L_e$

R-APDU:

Byte[0]...Byte[n-3]	Byte1	Byte2
0... $N_e$ response data bytes	SW1	SW2

#### Case 3 Short (3S):

C-APDU Length is minimum 6 bytes and maximum  $5+N_c$  bytes.  $N_c=L_c$  can't be 0 and its maximum value is 255. After APDU command execution, only two SW bytes are expected within the R-APDU (APDU response).

**C-APDU:**

Byte1	Byte2	Byte3	Byte4	Byte5	Byte[6]...Byte[Lc+5]
CLS	INS	P1	P2	$L_c$	0... $N_c$ data bytes

**R-APDU:**

Byte1	Byte2
SW1	SW2

**Case 4 Short (4S):**

C-APDU Length is minimum 7 bytes and maximum  $6+N_c$  bytes.  $N_c=L_c$  can't be 0 and its maximum value is 255.

$N_e$  can be from 1 to 256 which is encoded in the  $L_e$  field so, when the  $L_e=0$ ,  $N_e$  is 256 and in all other cases  $N_e=L_e$  { $N_e = L_e > 0 ? L_e : 256$ ; // C operation for the  $N_e$  assignment}

After APDU command execution, maximum  $N_e$  bytes and additional two SW bytes are expected within the R-APDU (APDU response).  $N_e$  only means that the host device is able to receive so many bytes after the successful execution of the APDU command. The actual number of response data bytes returned by the tag can be less than  $N_e$  or even 0.

**C-APDU:**

Byte1	Byte2	Byte3	Byte4	Byte5	Byte[6]...Byte[Lc+5]	Byte[Lc+6]
CLS	INS	P1	P2	$L_c$	0... $N_c$ data bytes	$L_e$

**R-APDU:**

Byte[0]...Byte[n-3]	Byte1	Byte2
0... $N_e$ response data bytes	SW1	SW2

**Example:**

Issuing NDEF Tag Application Select command (Case 1S):

'00 A4 04 00 07 D2 76 00 00 85 01 01 00'



```

CMD      55 94 AA 0E 00 CC B0
ACK      AC 94 CA 0E 00 CC 37
CMD_EXT  00 A4 04 00 07 D2 76 00 00 85 01 01 00 8D
RSP      DE 94 ED 03 00 00 AB
RSP_EXT  90 00 97
    
```

### uFR KEEP ALIVE mechanism

To support the APDU commands defined in the ISO 7816-4 standard via the "Half-duplex block transmission" (T=CL) protocol, as a physical layer of data transmission, defined in the ISO 14443-4 standard, it was necessary to expand the uFR COM protocol with a KEEP\_ALIVE packet type.

*KEEP\_ALIVE packet:*

Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
KEEP_ALIVE_HDR	CMD_CODE	KEEP_ALIVE_TRL	0	0	0	CHECKSUM

KEEP\_ALIVE\_HDR = 0xA1

CMD\_CODE = APDU\_TRANSCEIVE = 0x94 {in this case}

KEEP\_ALIVE\_TRL = 0x85

uFR sends this packet perpetually as long as the current APDU command is executed on the NFC tag and this execution is longer than 1.00s and the tag maintains connection to the uFR via the appropriate physical layer protocol. If the execution of the APDU command takes significantly longer, uFR will repeat the KEEP\_ALIVE packet every second until the RSP packet returns (or ERR packet in case of an error).

Attention: In the case of APDU commands whose execution on the NFC tag takes less than approx. 1.00s, no single one KEEP\_ALIVE packet will be returned, but RSP and RSP\_EXT packets will follow immediately.

### Extended APDU Support

Next generation devices that support ISO7816-4 and ISO14443-4 protocols also support Extended APDU commands. The difference between the Short and Extended APDU commands is in the maximum lengths of the C-APDU and R-APDU response and the way these lengths are encoded. The C-APDU header is the same for both short and extended APDU commands and contains {CLS, INS, P1, P2} bytes. So the way of encoding  $N_c$  and  $N_e$  lengths differs depending on the case of the APDU Extended commands (Cases 2E, 3E and 4E).

#### Case 2 Extended (2E):

C-APDU Length is exactly 7 bytes. There are no C-APDU data bytes.  $N_e$  can be from 1 to 65536 which is encoded in  $L_e\_MSB$  and  $L_e\_LSB$  fields so, when the  $L_e=0x0000$ ,  $N_e$  is 65536 and in all other cases  $N_e=L_e \{N_e = L_e > 0 ? L_e : 65536; // C operation for the N_e assignment\}$

After APDU command execution, maximum  $N_e$  bytes and additional two SW bytes are expected within the R-APDU (APDU response).  $N_e$  only means that the host device is able to receive so many bytes after the successful execution of the current APDU command. The actual number of response data bytes returned by the tag can be less than  $N_e$  or even 0.

**C-APDU:**

Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
CLS	INS	P1	P2	0	$L_e\_MSB$	$L_e\_LSB$

MSB stands for Most Significant Byte

LSB stands for Least Significant Byte (Big Endian byte sequence is in use here)

**R-APDU:**

Byte[0]...Byte[n-3]	Byte1	Byte2
0... $N_e$ response data bytes	SW1	SW2

**Case 3 Extended (3E):**

C-APDU length is minimum 8 bytes and maximum  $7+N_c$  bytes.  $N_c$  is encoded in  $L_c\_MSB$  and  $L_c\_LSB$  bytes which represent a 16-bit word in the Big Endian sequence.  $N_c=L_c$  can't be 0 and its maximum value is 65535. After APDU command execution, only two SW bytes are expected within the R-APDU (APDU response).

**C-APDU:**

Byte1	Byte2	Byte3	Byte4	Byte5	Byte5	Byte6	Byte[8]...Byte[ $L_c+7$ ]
CLS	INS	P1	P2	0	$L_c\_MSB$	$L_c\_LSB$	0... $N_c$ data bytes

MSB stands for Most Significant Byte

LSB stands for Least Significant Byte (Big Endian byte sequence is in use here)

**R-APDU:**

Byte[0]...Byte[n-3]	Byte1	Byte2
0... $N_e$ response data bytes	SW1	SW2

**Case 4 Extended (4E):**

C-APDU Length is minimum 10 bytes and maximum  $9+N_c$  byte.  $N_c$  is encoded in  $L_c\_MSB$  and  $L_c\_LSB$  bytes which represent a 16-bit word in the Big Endian sequence.  $N_c=L_c$  can't be 0 and its maximum value is 65535.

$N_e$  can be from 1 to 65536 which is encoded in  $L_e\_MSB$  and  $L_e\_LSB$  fields so, when the  $L_e=0x0000$ ,  $N_e$  is 65536 and in all other cases  $N_e=L_e$  { $N_e = L_e > 0 ? L_e : 65536$ ; // C operation for the  $N_e$  assignment}

After APDU command execution, maximum  $N_e$  bytes and additional two SW bytes are expected within the R-APDU (APDU response).  $N_e$  only means that the host device is able to receive so many bytes after the successful execution of the current APDU command. The actual number of response data bytes returned by the tag can be less than  $N_e$  or even 0.

**C-APDU:**

Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte[8]...Byte[Lc+7]	Byte[Lc+8]	Byte[Lc+9]
CLS	INS	P1	P2	0	L <sub>c</sub> _MSB	L <sub>c</sub> _LSB	0...N <sub>e</sub> data bytes	L <sub>e</sub> _MSB	L <sub>e</sub> _LSB

MSB stands for Most Significant Byte

LSB stands for Least Significant Byte (Big Endian byte sequence is in use here)

**R-APDU:**

Byte[0]...Byte[n-3]	Byte1	Byte2
0...N <sub>e</sub> response data bytes	SW1	SW2

In order to support data streams of maximum lengths up to 65536 bytes (64KB), the uFR COM protocol needed to be further expanded. This extension of the uFR COM protocol involves the chained transmission of smaller data chunks of a maximum of 262 bytes. The key to this protocol extension is to use CMD packets with CMD\_Par1 set to the value APDU\_STREAM\_INDICATOR = 0x5A immediately followed by the CMD\_EXT packet containing a current chunk, as long there is continued transmission i.e. the next C-APDU chunk.

APDU\_STREAM\_INDICATOR = 0x5A

After each valid CMD package containing CMD\_Par1 = APDU\_STREAM\_INDICATOR, followed by the CMD\_EXT packet containing a current chunk, uFR will respond with a modified ACK packet which is shown in the table below:

Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
ACK_HEADER	CMD_CODE	ACK_TRAILER	0	0	APDU_STREAM_INDICATOR	CHECKSUM

In case of an error, the standard ERR packet with the status code is returned and the data transfer process is interrupted.

Poslednji CMD paket C-APDU strima ima CMD\_Par1 različit od APDU\_STREAM\_INDICATOR. Nakon takvog paketa, počinje izvršavanje APDU komande u ciljnom uređaju/tagu i na red dolazi eventualni uFR KEEP\_ALIVE mehanizam dok se čeka RSP i RSP\_EXT paket.

Napomena: u slučaju Extended APDU komandi čije izvršavanje na ciljnom uređaju traje manje od cca. 1.00s, neće doći do vraćanja niti jednog KEEP\_ALIVE paketa već će odziv biti odmah u vidu RSP i RSP\_EXT paketa.

Extended APDU response (R-APDU) reception can also be chained i.e. in chunks, and indicator for it is the existence of `RSP_Val1 = APDU_STREAM_INDICATOR` in the RSP packet. Each such RSP packet is followed by an `RSP_EXT` packet which contains a valid R-APDU chunk. The last (or only one) RSP packet, followed by an `RSP_EXT` chunk which contains R-APDU and 2 SW bytes at the end of the R-APDU stream, has an `RSP_Val1` different from `APDU_STREAM_INDICATOR` e.g. `RSP_Val1 = 0`. The size of individual R-APDU chunks is determined by the buffer size of the ISO14443-4 tag itself, so it can vary from tag to tag. The maximum size of individual R-APDU chunks can be 256 bytes, so 2 bytes are used to encode the length of the `RSP_EXT` packet in the RSP packet in a similar way to `CMD` and `CMD_EXT` (Little Endian) uFR COM protocol extensions, which is already described in the "Short APDU Support" section. So, the `RSP_EXT_Length` field actually represents `RSP_EXT_Length_LSB` and `RSP_Val0` is `RSP_EXT_Length_MSB`. It is not possible to know in advance the length of the R-APDU response, so the protocol must rely solely on the described mechanism. It is only known that the R-APDU can be a maximum total length of  $65536 + 2 \text{ (SW)} = 65538$  bytes, not counting data overhead due to the use of an additional layer of uFR COM protocol.

In case of an error, the standard ERR packet with the status code is returned and the data transfer process is interrupted.

## PKI infrastructure and digital signature support

### Fully supported from firmware version 3.9.55

In our product range, we have special cards called "D-Logic JCAp" (working title), which contains support for PKI infrastructure and digital signing. To use these features you have to implement specific APDU command sequences using the `APDU_TRANSCEIVE` command described before. We have implemented PKI infrastructure and digital signature support in our API (for reference read "**uFR Series NFC reader API**").

## **Support for ISO 7816 protocol**

The device communicates via ISO7816 UART with the smart card located into the mini smart card holder. Supports synchronous cards which do not use C4/C8.

### **Basic commands**

#### **OPEN\_ISO7816\_INTERFACE (0x95)**

Function activates the smart card and returns an ATR (Answer To Reset) array of bytes from the smart card.

After the successfully executed function, the same APDU commands as for ISO 14443-4 tags can be used, but not at the same time.

#### **SAM AV2 ACTIVATION**

If NXP SAM AV2 is locked, this function will unlock the SAM.

CMD\_Par0 = 1  
CMD\_EXT not in use

#### **Example:**

```
CMD      55 95 AA 00 01 00 72
RSP      DE 95 ED 1D 00 00 C2
RSP_EXT  3B DF 18 FF 81 F1 FE 43 00 3F 03 83 4D 49 46 41 52 45 20 50 6C
75 73 20 53 41 4D 3B 42
```

#### **GENERAL 7816 SMART CARD ACTIVATION**

CMD\_Par0 = 4  
CMD\_EXT not in use  
RSP\_EXT not in use

#### **Example:**

```
CMD      55 95 AA 00 04 00 75
RSP      DE 95 ED 14 00 00 B9
RSP_EXT  3B F9 96 00 00 80 31 FE 45 4A 54 61 78 43 6F 72 65 56 0F 42
```

#### **APDU\_switch\_to\_ISO7816\_interface**

Function switches the use of APDU to ISO 7816 interface from ISO 14443-4 interface. The smart card must be in the active state.

CMD\_Par0 = 2  
CMD\_EXT not in use  
RSP\_EXT not in use

**Example:**

```
CMD      55 95 AA 00 02 00 6F
RSP      DE 95 ED 00 00 00 AD
```

**CLOSE\_ISO7816\_INTERFACE (0x96)**

**close\_ISO7816\_interface\_no\_APDU**

Function deactivates the smart card. APDU commands are not used.

CMD\_Par0 = 2  
CMD\_EXT not in use  
RSP\_EXT not in use

**Example:**

```
CMD      55 96 AA 00 02 00 72
RSP      DE 96 ED 00 00 00 AC
```

**close\_ISO7816\_interface\_APDU\_ISO14443\_4**

Function deactivates the smart card. APDU commands are used by ISO 14443-4 tags. Tag must already be in ISO 14443-4 mode.

CMD\_Par0 = 1  
CMD\_EXT not in use  
RSP\_EXT not in use

**Example:**

```
CMD      55 96 AA 00 01 00 6F
RSP      DE 96 ED 00 00 00 AC
```

**APDU\_switch\_to\_ISO14443\_4\_interface**

Function switches the use APDU to ISO 14443-4 tags. The smart card stays in active state. Tag must already be in ISO 14443-4 mode.

CMD\_Par0 = 3  
CMD\_EXT not in use  
RSP\_EXT not in use

**Example:**

```
CMD      55 96 AA 00 03 00 71
RSP      DE 96 ED 00 00 00 AC
```

**APDU\_switch\_off\_from\_ISO7816\_interface**

APDU commands are not used. The smart card stays in active state.

```
CMD_Par0 = 4
CMD_EXT not in use
RSP_EXT not in use
```

**Example:**

```
CMD      55 96 AA 00 04 00 74
RSP      DE 96 ED 00 00 00 AC
```

**Originality checking**

**Supported from firmware version 3.9.8**

Some card chips support the originality checking mechanism using Elliptic Curve Digital Signature Algorithm (ECDSA). Chip families that support originality checking mechanisms are NTAG 21x and Mifare Ultralight EV1. For details on originality checking, you must have a non-disclosure agreement (NDA) with the manufacturer who will provide you with the relevant documentation.

uFR originality checking support is based on READ\_ECC\_SIGNATURE command. For the rest of originality checking procedure you need to use the instructions from the manufacturer documentation.

We have originality checking support completely implemented in our API using uFCoder library function **OriginalityCheck()** (for reference read “**uFR Series NFC reader API**”).

***READ\_ECC\_SIGNATURE (0xBF)***

**Supported from firmware version 3.9.8**

This command reads the ECC signature of the card chip UID. Card chip UID is signed using EC private key known only to the manufacturer.

```
CMD_Par0 not in use.
CMD_Par1 not in use.
CMD_EXT not in use.
```

On success:

```
RSP_Val0 will contain the DlogicCardType code of the card in the field.
RSP_Val1 will contain the UID length of the card in the field.
```

RSP\_EXT will contain an ECC signature from the card in the field, in the first 32 bytes, followed by the 10 bytes of UID. UID field in the RSP\_EXT data will always have 10 bytes but the RSP\_Val1 defines how many of them are relevant.

If card in field doesn't have originality checking support, returned error code is:

**UNSUPPORTED\_CARD\_TYPE (0x11)**

**Example:**

```

CMD      55 BF AA 00 00 00 47
RSP      DE BF ED 2B 0A 07 B1
RSP_EXT  AA 7B 0D 58 CE 43 D7 1A D1 CB 8B 37 56 6B 1E 86
           27 97 34 D7 14 4A 59 40 50 93 B4 B6 F8 7A 53 70
           04 13 95 6A 64 34 80 00 00 00 92
    
```

From firmware version 5.0.43.

Command supports ECC with variable length.

CMD\_PAR0 is 1

CMD\_Par1 not in use.

CMD\_EXT not in use.

RSP\_Val0 will contain the DlogicCardType code of the card in the field.

RSP\_Val1 will contain the UID length of the card in the field.

RSP\_EXT will contain an ECC signature from the card in the field, followed by the 10 bytes of UID. UID field in the RSP\_EXT data will always have 10 bytes but the RSP\_Val1 defines how many of them are relevant.

**Example:**

Read ECC signature from NTAG 424 TT without authentication. ECC signature length is 56 bytes.

```

CMD      55 BF AA 00 01 00 48
RSP      DE BF ED 43 13 07 E2
RSP_EXT  02 D9 33 90 43 1C 8B 37 1F 6C 15 67 0F 7F 52 97 26 D6 E3 C5 EC
           D5 81 30 6F 61 89 73 48 F2 0D BC 69 3D 4B 1C 16 E3 A3 88 77 C5 AC 82 A2
           DA 15 B7 26 D0 5E 2D 1E B3 48 39 04 75 7C AA 5C 5E 80 00 00 00 70
    
```

Command supports NTAG 424 and NTAG 424 TT cards if the Random ID is activated.

CMD\_PAR0 = 2, CMD\_PAR1 = 0

CMD\_EXT

1st byte defines internal key using (1 - reader key, 0 - provided key, 0xFF - no authentication)

2nd byte is ordinal AES key number into reader (0 - 15)

array 3 - 18 is provided AES key

19th byte is key number (0 - 4)

20th byte is checksum

RSP\_Val0 will contain the DlogicCardType code of the card in the field.



RSP\_EXT will contain a 56 bytes long ECC signature from the card in the field, and Random ID.  
Note: UID must read with the NT4\_GET\_UID command.

**Example:**

Read ECC signature from NTAG 424 TT. Authentication with provided key 0x00000000000000000000000000000000. Key number is 0.

```
CMD      55 BF AA 13 02 00 58
ACK      AC BF CA 13 02 00 CF
CMD_EXT  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 07
RSP      DE BF ED 43 13 04 DF
RSP_EXT6701 6D 2C C2 0C 5B 21 0C 22 AE F0 57 2E 4B 35 F8 68 84 8E EA E4
D3 25 4E 72 DB 04 66 96 A5 DF 70 B4 E4 C0 45 6E 4B 4F D2 07 DD E5 5C 42
51 C1 08 C9 4D 96 64 3E 20 BA 08 40 B1 05 00 00 00 00 00 00 00 6C
```

From firmware version 5.0.43.

Command supports Desfire EV2 and Desfire Light cards.

Command supports NTAG 424 and NTAG 424 TT cards if the Random ID is activated.

CMD\_PAR0 = KEY\_TYPE | 0x03

AES\_KEY\_TYPE = 0x00,

DES3K\_KEY\_TYPE = 0x10,

DES\_KEY\_TYPE = 0x20,

DES2K\_KEY\_TYPE = 0x30

CMD\_PAR1 = 0

CMD\_EXT

1st byte defines internal key using (1 - reader key, 0 - provided key)

2nd byte is ordinal AES key number into reader (0 - 15)

array 3 - 18 is provided key (8 bytes DES, 16 bytes AES and 2K3DES, or first 16 bytes of 3K3DES)

array 19 - 21 is AID

22st byte is application key number

23rd byte is 1 if authentication is required, or 0 if not.

24rd byte is checksum.

(if 3K3DES key, then array 24 - 31 is last 8 bytes of 3K3DES key, and 32nd byte is checksum)

RSP\_Val0 will contain the DlogicCardType code of the card in the field.

RSP\_EXT will contain a 56 bytes long ECC signature from the card in the field, and Random ID or UID.

Note: If Random ID is activated, then the UID must be read with the GET\_DESFIRE\_UID command.

**Example:**

Random ID isn't activated.

```
CMD          55 BF AA 18 03 00 62
ACK          AC BF CA 18 03 00 C9
CMD_EXT      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 07
RSP          DE BF ED 43 3F 07 FE
RSP_EXT      1A E4 0E 9A 35 57 5A 13 47 56 76 46 B7 0F 0B 0D 85 BD 25 DE 02
62 36 2A 1C C0 DF 59 3C 48 D8 6E F9 D8 06 E9 0F 83 46 6A 40 96 5C 0A 26
26 9F AA 9C 1F 7D 4F 11 FD 5C A2 04 43 5B 0A FF 64 80 00 00 00 93
```

**Example:**

Random ID is activated. Provided 2K3DES key 0x01020304050607080910111213141516. AID = 0xD20000. Application key number is 0. Authentication is required.

```
CMD          55 BF AA 18 33 00 72
ACK          AC BF CA 18 33 00 F9
CMD_EXT      00 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 00 00 D2
00 01 CC
RSP          DE BF ED 43 2B 04 E7
RSP_EXT      24 1F 8C E1 07 1B 6D FE 3E E1 F2 40 97 82 33 5D 17 86 35 14 65
30 6A 9F 54 5E 6D 48 1D 5A FA 98 71 EE 36 17 45 B8 F3 3E DD E8 2A 8F 18
EB 49 79 96 C8 9F F8 D8 6A E0 4B 08 FC CB 23 00 00 00 00 00 00 3B
```

**Anti-collision support i.e. multi card reader mode**

**Supported from firmware version 5.0.1 (for uFR PLUS devices only)**

After power on or resetting the reader it is in a “single card” mode of operation. In this mode the reader can only work with one card in the field and the card is selected automatically.

uFR PLUS devices can be placed in so-called “anti-collision” mode of operation using ENABLE\_ANTI\_COLLISION command. In that mode reader can work with multiple cards in the field. Fundamental problem in a “anti-collision” mode of operation is the amount of energy that is required to power the cards in the field. Different types of cards require more or less energy. So the maximum number of cards with which reader can work simultaneously depends on specific needs for powering different cards in the field. The reader can work with up to 4 cards that have low average consumption, at a time. Cards that have low average consumption include the following models: Mifare Ultralight, Mifare Classic, Ntag series.

All the card models which supports modern cryptography mechanisms have higher power consumption. So in the case of Mifare Desfire, Mifare Ultralight C, Mifare Plus, Java Cards and other high consumption cards there should be no more then 2 cards in the reader field at a time.

### **ENABLE\_ANTI\_COLLISION (0x2D)**

This command puts the reader in an “anti-collision” mode of operation.

CMD\_Par0 and CMD\_Par1 not in use.

CMD\_EXT and RSP\_EXT not in use.

#### **Example:**

```
CMD      55 2D AA 00 00 00 D9
RSP      DE 2D ED 00 00 00 25
```

### **DISABLE\_ANTI\_COLLISION (0x2E)**

Exits from “anti-collision” mode of operation i.e. put the reader into “single card” mode of operation.

CMD\_Par0 and CMD\_Par1 not in use.

RSP\_Val0 and RSP\_Val1 not in use.

CMD\_EXT and RSP\_EXT not in use.

#### **Example:**

```
CMD      55 2E AA 00 00 00 D8
RSP      DE 2E ED 00 00 00 24
```

### **ENUM\_CARDS (0x37)**

If the reader is in an “anti-collision” mode of operation, this command enumerates cards which are found in the reader field. Otherwise the command reports ANTI\_COLLISION\_DISABLED error code.

All the following commands: LIST\_CARDS, SELECT\_CARD and DESELECT\_CARD, work with UIDs from the actual UID list of the enumerated cards, which is obtained by the last ENUM\_CARDS command issuing.

CMD\_Par0 and CMD\_Par1 not in use.

RSP\_Val0 and RSP\_Val1 not in use.

CMD\_EXT and RSP\_EXT not in use.

#### **Example:**

```
CMD      55 37 AA 00 00 00 CF
RSP      DE 37 ED 00 02 16 17
```

### **LIST\_CARDS (0x38)**

Before issuing this command you must issue the ENUM\_CARDS command first.

CMD\_Par0 and CMD\_Par1 not in use.

CMD\_EXT not in use.

RSP\_Val0 contains the number of the cards detected in the reader field.

RSP\_Val1 contains the length of the UID list, in bytes.

RSP\_EXT contains the UID list of the card in the reader field.

For each UID, of the cards detected in the reader field, there are 11 “UID record bytes” allocated in the list. First of those 11 bytes allocated designate actual UID length immediately followed by the exactly 10 bytes of UID (which is maximum hypothetical UID size). E.g, if the actual UID length is 4 bytes, you should ignore the last 6 bytes of the UID record.

**Example 1 (there is only 1 card in the field):**

```
CMD      55 38 AA 00 00 00 CE
RSP      DE 38 ED 0C 01 0B 14
RSP_EXT  07 04 48 76 B2 04 35 80 00 00 00 45
```

**Example 2 (there is 2 cards in the field):**

```
CMD      55 38 AA 00 00 00 CE
RSP      DE 38 ED 17 02 16 0F
RSP_EXT  07 04 13 95 6A 64 34 80 00 00 00
          04 C5 58 3E E6 00 00 00 00 00 00 85
```

**Example 3 (there is 3 cards in the field):**

```
CMD      55 38 AA 00 00 00 CE
RSP      DE 38 ED 22 03 21 12
RSP_EXT  07 04 13 95 6A 64 34 80 00 00 00
          04 C5 58 3E E6 E2 00 00 00 00 00
          07 04 48 76 B2 04 35 80 00 00 00 A9
```

**SELECT\_CARD (0x39)**

This command selects one of the cards whose UID is on the actual UID list of the enumerated cards. If there are any of the cards previously selected by issuing this command you will get an error CARD\_ALREADY\_SELECTED and you should issue the DESELECT\_CARD command prior using this command, in such a case.

CMD\_Par0 contains card UID size

CMD\_Par1 not in use.

CMD\_EXT contains card UID (have to be “card UID size” bytes as designated by the CMD\_Par0).

RSP\_EXT not in use.

RSP\_Val0 contains selected card type (see GET\_DLOGIC\_CARD\_TYPE enumeration).

RSP\_Val1 not in use.

**Example:**

```

CMD      55 39 AA 05 04 00 CE
ACK      AC 39 CA 05 04 00 65
CMD_EXT  C5 58 3E E6 4C
RSP      DE 39 ED 00 21 00 32

```

**DESELECT\_CARD (0x3A)**

Deselects previously selected card issuing SELECT\_CARD command.

CMD\_Par0 and CMD\_Par1 not in use.

RSP\_Val0 and RSP\_Val1 not in use.

CMD\_EXT and RSP\_EXT not in use.

**Example:**

```

CMD      55 3A AA 00 00 00 CC
RSP      DE 3A ED 00 00 00 10

```

**GET\_ANTI\_COLLISION\_STATUS (0x3B)**

Using this command you can get the current anti-collision status of the reader.

CMD\_Par0 and CMD\_Par1 not in use.

CMD\_EXT and RSP\_EXT not in use.

RSP\_Val0 contains 1 if the reader is in a “anti-collision” mode of operation, otherwise 0.

RSP\_Val1 contains 1 if the reader is in a “anti-collision” mode of operation and there is selected card, otherwise 0.

**Example:**

```

CMD      55 3B AA 00 00 00 CB
RSP      DE 3B ED 00 01 01 0F

```

**Commands for uFR Online****ESP\_SET\_IO\_STATE (0xF3)**

uFR Online only.

Function set IO pin state.

CMD\_Par0 pin number

CMD\_Par1 IO state 0 - low, 1 - high, 2 - input

RSP\_Val0 and RSP\_Val1 not in use.

CMD\_EXT is optional.

CMD\_EXT 1<sup>st</sup> byte is a time-delayed state (low in example). 2<sup>nd</sup> and 3<sup>rd</sup> are time in milliseconds (1388 is 5000ms). The 4<sup>th</sup> byte is CMD\_EXT checksum.

RSP\_EXT not in use.

**Example:**

IO pin 3 high level.

```
CMD      55 F3 AA 00 03 01 15
RSP      DE F3 ED 00 00 00 C7
```

**Example 1:**

IO pin 3 high level and set low level after 5000ms.

```
CMD      55 F3 AA 00 03 01 15 00 13 88 A2
RSP      DE F3 ED 00 00 00 C7
```

***ESP\_GET\_IO\_STATE (0xF4)***

uFR Online only.

Function gets IO pin states.

CMD\_Par0 and CMD\_Par1 not in use.

RSP\_Val0 and RSP\_Val1 not in use.

CMD\_EXT not in use.

**Example:**

Get IO pins state. All pins set as input

```
CMD      55 F4 AA 00 00 00 12
RSP      DE F4 ED 00 00 00 CE
RSP_EXT  02 02 02 02 02 02 07
```

***ESP\_READER\_TIME\_WRITE (0xF5)***

uFR Online only.

Function to set RTC date/time.

CMD\_Par0 and CMD\_Par1 not in use.

RSP\_Val0 and RSP\_Val1 not in use.

CMD\_EXT contains year, month, day, hour, minutes, seconds

**Example:**

Set date and time to 2019-06-20 10:01:02

```
CMD      55 F5 AA 07 00 00 14
ACK      AC F5 CA 07 00 00 60
CMD_EXT  13 06 14 0A 01 02 0F
RSP      DE F5 ED 00 00 00 CD
```

### **ESP\_READER\_TIME\_READ (0xF6)**

uFR Online only.

Function to get RTC date/time.

CMD\_Par0 and CMD\_Par1 not in use.

RSP\_Val0 and RSP\_Val1 not in use.

RSP\_EXT 1<sup>st</sup> to 8<sup>th</sup> byte contains password, 9<sup>th</sup> to 14<sup>th</sup> byte contains date/time.

#### **Example:**

Get 2019-06-20 10:01:02 date and time from device. Password is '11111111'.

```
CMD      55 F6 AA 00 00 00 10
RSP      DE F6 ED 00 00 00 CC
RSP_EXT  31 31 31 31 31 31 31 31 13 06 14 0A 01 02 0F
```

### **ESP\_READER\_EEPROM\_READ (0xF7)**

uFR Online only.

Function to read uFR Online EEPROM data.

CMD\_Par0 and CMD\_Par1 not in use.

CMD\_EXT 1<sup>st</sup> to 4<sup>th</sup> byte contains EEPROM address, 5<sup>th</sup> to 8<sup>th</sup> byte contains length of data to read. (little endian)

RSP\_Val0 and RSP\_Val1 not in use

RSP\_EXT contains requested EEPROM data

#### **Example:**

Read 5 bytes (0xFF, 0xFF, 0xFF, 0xFF, 0xFF) from address 0x00

```
CMD      55 F7 AA 09 00 00 08
ACK      AC F7 CA 09 00 00 9F
CMD_EXT  00 00 00 00 00 00 00 05 0C
RSP      DE F7 ED 05 00 00 C8
RSP_EXT  FF FF FF FF FF 06
```

### **ESP\_READER\_EEPROM\_WRITE (0xFB)**

uFR Online only.

Function to write uFR Online EEPROM data.

CMD\_Par0 and CMD\_Par1 not in use.

CMD\_EXT 1<sup>st</sup> to 4<sup>th</sup> byte contains EEPROM address, 5<sup>th</sup> to 8<sup>th</sup> byte contains length of data to read, 9<sup>th</sup> to 16<sup>th</sup> byte contains password, bytes from 17<sup>th</sup> contain data. (little endian)

RSP\_Val0 and RSP\_Val1 not in use

RSP\_EXT not in use

#### **Example:**

Write 5 bytes (0xFF, 0xFF, 0xFF, 0xFF, 0xFF) to address 0x00. Password is '11111111'.

```
CMD      55 FB AA 16 00 00 19
ACK      AC FB CA 16 00 00 92
CMD_EXT  00 00 00 00 00 00 00 05 31 31 31 31 31 31 31 31
          FF FF FF FF FF 01
RSP      DE FB ED 00 00 00 CF
```

### **ESP\_SET\_DISPLAY\_DATA (0xF8)**

uFR Online only.

Function enables sending data to the uFR Online LED. A string of data contains information about the intensity of color in each cell. Each cell has three LED (red, green and blue). For each cell of the three bytes is necessary. The first byte indicates the intensity of the red color, the second byte indicates the intensity of the green color, and the third byte indicates the intensity of blue color.

CMD\_Par0 and CMD\_Par1 contain LED light duration in ms. If duration is 0, light will never turn off

CMD\_EXT contains data for display with checksum

RSP\_Val0 and RSP\_Val1 not in use

RSP\_EXT not in use

#### **Example:**

red = 0x10, green = 0xFF, blue = 0x20, duration = 100ms



```
CMD      55 F8 AA 02 00 64 0C
ACK      AC F8 CA 02 00 64 FF
CMD_EXT  10 FF 20 10 FF 20 07
RSP      DE F8 ED 00 00 00 D2
```

### **ESP\_READER\_RESET (0xF9)**

uFR Online only.

Function resets device connected to uFR Online.

CMD\_Par0 - always set to 0.

CMD\_Par1 not in use.

RSP\_Val0 and RSP\_Val1 not in use.

CMD\_EXT and RSP\_EXT not in use.

#### **Example:**

Reset device.

```
CMD      55 F9 AA 00 00 00 0D
RSP      DE F9 ED 00 00 00 D1
```

### **ESP\_SET\_TRANSPARENT\_READER (0xF9)**

uFR Online only.

Function set transparent reader connected to uFR Online.

CMD\_Par0 - set 1 for first device(default) or 2 for external connected reader.

CMD\_Par1 not in use.

RSP\_Val0 and RSP\_Val1 not in use.

CMD\_EXT and RSP\_EXT not in use.

#### **Example:**

Set first reader as transparent device.

```
CMD      55 F9 AA 00 01 00 0E
RSP      DE F9 ED 00 00 00 D1
```

Set external reader as transparent device.

```
CMD      55 F9 AA 00 02 00 0B
RSP      DE F9 ED 00 00 00 D1
```

### **ESP\_READER\_PASSWORD\_WRITE (0xFA)**

uFR Online only.

Function to write uFR Online password.

CMD\_Par0 and CMD\_Par1 not in use.

CMD\_EXT 1<sup>st</sup> to 8<sup>th</sup> byte contains old password, bytes from 9<sup>th</sup> to 16<sup>th</sup> contains new password.

RSP\_Val0 and RSP\_Val1 not in use

RSP\_EXT not in use

### Example:

Write a new password '22222222'. Old password is '11111111'.

```
CMD      55 FA AA 11 00 00 1B
ACK      AC FA CA 11 00 00 94
CMD_EXT  31 31 31 31 31 31 31 31 32 32 32 32 32 32 32 32 07
RSP      DE FA ED 00 00 00 D0
```

### ESP\_GET\_READER\_SERIAL (0xE7)

It gives the uFR Online serial number with length of 4 bytes. The CMD\_EXT set is not in use.

The CMD\_Par0 and CMD\_Par1 are not in use.

If everything operates as expected the RESPONSE set is sent and after that also the RESPONSE EXT set of 5 bytes which contains 4 byte ReaderSerialNumber values (little-endian) and at the end one checksum byte.

### Example:

Send CMD GET\_READER\_SERIAL

```
55 E7 AA 00 00 00 1F
```

### Where

55 - CMD\_HEADER  
E7 - CMD\_CODE  
AA - CMD\_TRAILER  
00 00 00 - CMD\_EX\_Length and CMD\_Par0 and CMD\_Par1 not used  
1F - CHECKSUM

Reader answer with RESPONSE – RSP packet followed by RSP\_EXT packet

```
DE E7 ED 05 00 00 D8 54 7E 1A 5D 74
```

### Where RSP PACKET contains

DE - RSP\_HEADER  
E7 - CMD\_CODE  
ED - RSP\_TRAILER  
05 - RSP\_EXT\_Length  
00 00 - RSP\_Val0 and RSP\_Val1 not used  
D8 - CHECKSUM

### and RSP\_EXT contains

54 7E 1A 5D - Device serial number (currently serial is 5D 1A 7E 54, little-endian notation)  
74 - CHECKSUM

## **Miscellaneous commands**

### **CHECK\_UID\_CHANGE (0xE4)**

From firmware version 5.0.27

Function tries to change the UID on the card. On some cards (e.g. Magic Classic) changing UID is possible.

CMD\_Par0 and CMD\_Par1 not in use.

CMD\_EXT not in use

RSP\_EXT not in use

#### **Example:**

If “Magic Classic” card is tested, then function returns OK, else function returns error code.

```
CMD      55 E4 AA 00 00 00 22
RSP      DE E4 ED 00 00 00 DE
```

### **RF\_RESET (0xE5)**

From firmware version 5.0.27

Command reset RF field at the reader. The RF field will be off, and then on after 50ms.

CMD\_Par0 and CMD\_Par1 not in use.

CMD\_EXT not in use

RSP\_EXT not in use

#### **Example:**

```
CMD      55 E5 AA 00 00 00 21
RSP      DE E5 ED 00 00 00 DD
```

From firmware version 5.0.51.

In the multi card reader mode.

### **RF\_ON**

Command switch on RF field at the reader.

CMD\_Par0 = 1

CMD\_EXT not in use

RSP\_EXT not in use

**Example:**

```

CMD          55 E5 AA 00 01 00 22
RSP          DE E5 ED 00 00 00 DD

```

**RF\_OFF**

Command switch off RF field at the reader. The RF field can be switched on by RF\_ON, or ENUM\_CARDS, or DISABLE\_ANTICOLISION command.

```

CMD_Par0 = 2
CMD_EXT not in use
RSP_EXT not in use

```

**Example:**

```

CMD          55 E5 AA 00 02 00 1F
RSP          DE E5 ED 00 00 00 DD

```

**GET\_READER\_STATUS (0xE6)**

From firmware version 5.0.33

Function returns various reader states. The reader states are defined into following structures. This function is useful for checking if the reader is still in emulation mode after the command TAG\_EMULATION\_START.

```

typedef enum E_EMULATION_MODES
{
    TAG_EMU_DISABLED = 0,
    TAG_EMU_DEDICATED,
    TAG_EMU_COMBINED,
    TAG_EMU_AUTO_AD_HOC
}emul_modes_t;

```

```

typedef enum E_EMULATION_STATES
{
    EMULATION_NONE = 0,
    EMULATION_IDLE,
    EMULATION_AUTO_COLL,
    EMULATION_ACTIVE,
    EMULATION_HALT,
    EMULATION_POWER_OFF
}emul_states_t;

```

```

typedef enum E_PCD_MGR_STATES

```

```
{
    PCD_MGR_NO_RF_GENERATED = 0,
    PCD_MGR_14443A_POLLING,
    PCD_MGR_14443A_SELECTED,
    PCD_MGR_CE_DEDICATED,
    PCD_MGR_CE_COMBO_START,
    PCD_MGR_CE_COMBO,
    PCD_MGR_CE_COMBO_IN_FIELD
}pcd_states_t;
```

CMD\_Par0 and CMD\_Par1 not in use.

CMD\_EXT not in use

RSP\_EXT

1st byte is reader state from pcd\_states\_t structure

- normal working mode states are PCD\_MGR\_NO\_RF\_GENERATED or PCD\_MGR\_14443A\_POLLING or PCD\_MGR\_14443A\_SELECTED.

- NTAG emulation mode state is PCD\_MGR\_CE\_DEDICATED

2nd byte is emulation mode from emul\_modes\_t structure

- normal working mode state is TAG\_EMU\_DISABLED

- NTAG emulation mode state is TAG\_EMU\_DEDICATED

3rd byte is emulation state form emul\_states\_t structure

4th bytes is reader sleep mode indicator

0 - reader is in normal or emulation mode

1 - reader is in sleep mode

5th byte is checksum

### Example:

```
WAKE UP BYTE    00    (send just before command)
CMD             55 E6 AA 00 00 00 20
RSP             DE E6 ED 05 00 00 D7
RSP_EXT        03 01 01 00 0A
```

### **READ\_TT\_STATUS** (0xB4)

From firmware version 5.0.60

Function provides the information about the tag tamper status which is detected when the NTAG 213 TT is powered by an RF field.

CMD\_Par0 and CMD\_Par1 not in use.

CMD\_EXT not in use

RSP\_EXT

array 1 - 4 is Tag Tamper Message

byte 5 is Tag Tamper status

byte 6 is checksum

**Example:**

Tag tamper message is A1 B2 C3 D4, tag tamper status is open 'O' 0x4F

**CMD**            55 B4 AA 00 00 00 52

**RSP**            DE B4 ED 06 00 00 88

**RSP\_EXT**      A1 B2 C3 D4 4F 52

**Appendix: ERROR CODES**

<b>ERROR</b>	<b>VALUE</b>
OK	0x00
COMMUNICATION_ERROR	0x01
CHKSUM_ERROR	0x02
READING_ERROR	0x03
WRITING_ERROR	0x04
BUFFER_OVERFLOW	0x05
MAX_ADDRESS_EXCEEDED	0x06
MAX_KEY_INDEX_EXCEEDED	0x07
NO_CARD	0x08
COMMAND_NOT_SUPPORTED	0x09
FORBIDDEN_DIRECT_WRITE_IN_SECTOR_TRAILER	0x0A
ADDRESSED_BLOCK_IS_NOT_SECTOR_TRAILER	0x0B
WRONG_ADDRESS_MODE	0x0C
WRONG_ACCESS_BITS_VALUES	0x0D
AUTH_ERROR	0x0E
PARAMETERS_ERROR	0x0F
MAX_SIZE_EXCEEDED	0x10
UNSUPPORTED_CARD_TYPE	0x11
COUNTER_ERROR	0x12
WRITE_VERIFICATION_ERROR	0x70
BUFFER_SIZE_EXCEEDED	0x71
VALUE_BLOCK_INVALID	0x72
VALUE_BLOCK_ADDR_INVALID	0x73
VALUE_BLOCK_MANIPULATION_ERROR	0x74
WRONG_UI_MODE	0x75
KEYS_LOCKED	0x76
KEYS_UNLOCKED	0x77

WRONG_PASSWORD	0x78
CAN_NOT_LOCK_DEVICE	0x79
CAN_NOT_UNLOCK_DEVICE	0x7A
DEVICE_EEPROM_BUSY	0x7B
RTC_SET_ERROR	0x7C
EEPROM_ERROR	0x7D
NO_CARDS_ENUMERATED	0x7E
CARD_ALREADY_SELECTED	0x7F
WRONG_CARD_TYPE	0x80
FORBIDDEN_IN_TAG_EMULATION_MODE	0x90
<b>Mifare Plus tags errors</b>	
MFP_COMMAND_OVERFLOW	0xB0
MFP_INVALID_MAC	0xB1
MFP_INVALID_BLOCK_NR	0xB2
MFP_NOT_EXIST_BLOCK_NR	0xB3
MFP_COND_OF_USE_ERROR	0xB4
MFP_LENGTH_ERROR	0xB5
MFP_GENERAL_MANIP_ERROR	0xB6
MFP_SWITCH_TO_ISO14443_4_ERROR	0xB7
MFP_ILLEGAL_STATUS_CODE	0xB8
MFP_MULTI_BLOCKS_READ	0xB9
<b>NT4H tags errors</b>	
NT4H_COMMAND_ABORTED	0xC0
NT4H_LENGTH_ERROR	0xC1
NT4H_PARAMETER_ERROR	0xC2
NT4H_NO_SUCH_KEY	0xC3
NT4H_PERMISSION_DENIED	0xC4
NT4H_AUTHENTICATION_DELAY	0xC5
NT4H_MEMORY_ERROR	0xC6
NT4H_INTEGRITY_ERROR	0xC7



<b>NT4H_FILE_NOT_FOUND</b>	<b>0xC8</b>
<b>NT4H_BOUNDARY_ERROR</b>	<b>0xC9</b>
<b>NT4H_INVALID_MAC</b>	<b>0xCA</b>
<b>NT4H_NO_CHANGES</b>	<b>0xCB</b>

## **Appendix: ERROR CODES for DESFire card operations**

```

#define DATA_OVERFLOW          2990
#define READER_ERROR           2999
#define NO_CARD_DETECTED       3000
#define CARD_OPERATION_OK      3001
#define WRONG_KEY_TYPE         3002
#define KEY_AUTH_ERROR         3003
#define CARD_CRYPTO_ERROR     3004
#define READER_CARD_COMM_ERROR 3005
#define PC_READER_COMM_ERROR   3006
#define COMMIT_TRANSACTION_NO_REPLY 3007
#define COMMIT_TRANSACTION_ERROR 3008
#define NO_ISO1444_4_CARD     3009
#define NOT_SUPPORTED_KEY_TYPE 3010

```

```
/* Status and error codes */
```

```

#define OPERATION_OK           0x0C00
#define NO_CHANGES           0x0C0C
#define OUT_OF_EEPROM_ERROR    0x0C0E
#define ILLEGAL_COMMAND_CODE  0x0C1C
#define INTEGRITY_ERROR        0x0C1E
#define NO_SUCH_KEY            0x0C40
#define LENGTH_ERROR           0x0C7E
#define PERMISSION_DENIED      0x0C9D
#define PARAMETER_ERROR        0x0C9E
#define APPLICATION_NOT_FOUND  0x0CA0
#define APPL_INTEGRITY_ERROR    0x0CA1
#define AUTHENTICATION_ERROR    0x0CAE
#define ADDITIONAL_FRAME       0x0CAF
#define BOUNDARY_ERROR         0x0CBE
#define PICC_INTEGRITY_ERROR    0x0CC1
#define COMMAND_ABORTED        0x0CCA
#define PICC_DISABLED_ERROR     0x0CCD
#define COUNT_ERROR            0x0CCE
#define DUPLICATE_ERROR        0x0CDE
#define EEPROM_ERROR_DES       0x0CEE
#define FILE_NOT_FOUND         0x0CF0
#define FILE_INTEGRITY_ERROR    0x0CF1

```

**Change log:****Firmware version 5.0.1 and later apply only to uFR PLUS devices**

<b>Date</b>	<b>Description</b>	<b>doc. revision</b>	<b>refers to the firmware ver.</b>
2022-02-24	Tag emulation T2T mirror counter support	1.30	5.0.61
2022-01-19	NTAG 213 TT support	1.29	5.0.60
2021-12-27	Extended APDU support	1.28	5.0.57
2021-10-18	SET_DISPLAY_DATA command has new feature for internal RGB modules, RED_LIGHT_CONTROL command changed	1.27	5.0.55
2021-09-01	Support for ISO 7816 protocol	1.26	5.0.44
2021-08-31	uFR Online GPIO time control added	1.25	
2021-01-11	RF field on/off in the multi card mode	1.23	5.0.51
2020-10-19	Desfire EV2 and Desfire Light ECC signature read support	1.22	5.0.44
2020-10-09	NTAG 424 TT support.	1.21	5.0.43
2020-07-17	Leave sleep mode command bug fix	1.20	5.0.23
2020-04-10	Transaction MAC for Desfire Light and Desfire EV2 support	1.19	5.0.38
2020-02-27	Mifare Plus X, SE or EV1 value block operations support	1.18	5.0.36
2020-02-20	Desfire light tag support	1.17	5.0.32
2020-02-20	COMMANDS FOR NT4H CARDS	1.16	5.0.32
2020-02-18	Default UART speed session.	1.15	5.0.1
2020-02-18	NTAG emulation mode in RAM (1008 bytes user memory). Get reader status	1.14	5.0.33
2019-10-30	For Mifare Plus card in SL3 uses functions for Mifare Classic card. AES key calculated from Crypto1 key.	1.13	5.0.29
2019-10-1	Check if UID changeable and RF reset	1.12	5.0.27
2019-10-1	SAM support for uFR CS with SAM	1.12	5.100.27
2019-08-15	Desfire operations with Linear and Cyclic records.	1.11	5.0.25
2019-08-14	Desfire DES, 2K3DES, and 3K3DES internal key support	1.10	5.0.25
2019-06-21	Added uFR Online commands.	1.9	

2019-05-17	Added description for a new command: code 0x97, SET_ISO14443_4_DL_STORAGE.	1.8	5.0.20
2019-05-17	All references to “ISO 14443-4A” have been changed to “ISO 14443-4” because uFR firmwares support ISO 14443-4A and ISO 14443-4B types both from 3.9.49 firmware version.	1.8	from 3.9.49
2019-05-16	Desfire get application identifiers added	1.7	5.0.19
2018-10-01	Anti-collision support (multi card reader mode) added	1.6	5.0.1
2018-07-05	Mifare Plus commands added. Differences for block read and write and linear read. uFR PLUS devices only.	1.5	
2018-07-04	Mifare Desfire value file manipulation functions. uFR PLUS devices only.	1.4	
2018-06-08	Added missing descriptions for READER_KEYS_LOCK, READER_KEYS_UNLOCK, and READER_PASSWORD_WRITE commands. Added hardware reset explanation.	1.3	
2018-06-08	Originality checking and READ_ECC_SIGNATURE command.	1.3	3.9.8
2018-06-08	Added missing descriptions for READ_COUNTER and INCREMENT_COUNTER commands (NFC Type 2 Tags)	1.3	3.9.11
2018-06-08	Added missing description for GET_NFC_T2T_VERSION command (NFC Type 2 Tags)	1.3	3.8.19
2018-06-08	Added missing card type constants in GET_DLOGIC_CARD_TYPE table.	1.3	
2018-05-31	SET_LED_CONFIG command added	1.2	3.9.53
2018-05-30	DESFIRE_WRITE_AES_KEY, and GET_DESFIRE_UID examples are corrected	1.1	
2018-05-30	Appendix: ERROR CODES for DESFire card operations	1.1	
2018-05-29	PKI infrastructure and digital signature support	1.1	3.9.55
2018-05-29	Changed date format in a Change log. Now we use a more universal ‘yyyy-mm-dd’ date format.	1.1	-
2017-06-29	Support for APDU commands in ISO 14443-4A tags	1.0	3.9.39
2017-05-23	Support for ISO 14443-4A protocol commands	1.0	3.9.36
2017-05-03	Commands for Ad-Hoc emulation mode parameters manipulation. (GET_AD_HOC_EMULATION_PARAMS and SET_AD_HOC_EMULATION_PARAMS).	1.0	3.9.35
2017-05-03	Ad-Hoc emulation mode commands.	1.0	3.9.34
2016-08-06	FAST_READ ISO14443-3 command with LINEAR_READ utilisation.	1.0	3.9.14

2016-06-06	Title "Authentication mode considerations" changed to "Authentication mode considerations for Mifare Classic tags"	1.0	
2016-06-06	New Title "Authentication mode considerations for NTAG 21x and other T2T tags"	1.0	3.9.10