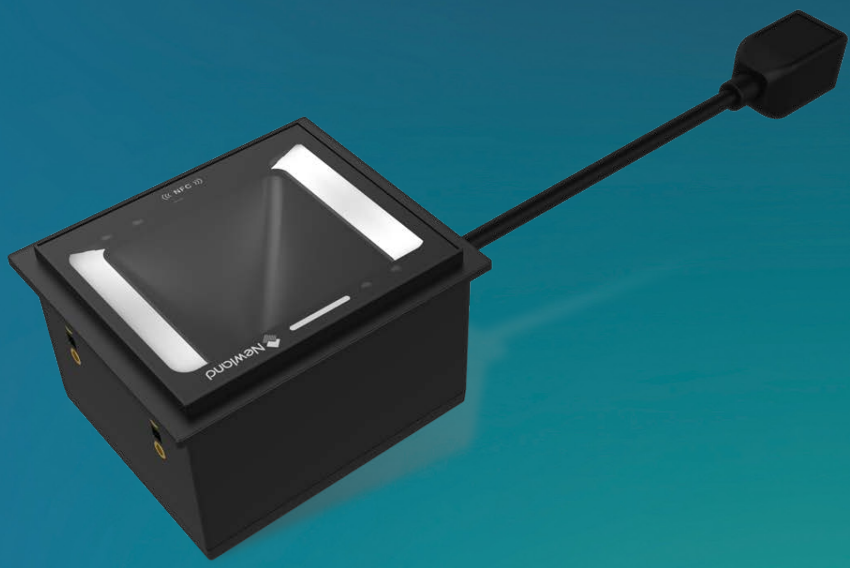
Newland AIDC

SCANNING MADE SIMPLE



FM3281 GROUPER

Fixed-Mount Scanners

Version: V1.0.1

NFC Technical Guide

Disclaimer

Please read the manual carefully before using the product and operate it according to the manual. It is advised that you keep this manual for future reference.

Do not disassemble the device or remove the seal label from the device; doing so will void the product warranty provided by Newland Europe BV.

All pictures in this manual are for reference only, and the actual product may differ.

Regarding product modification and update, Newland Europe BV reserves the right to make changes to any software or hardware to improve reliability, function, or design at any time without notice. The information contained herein is subject to change without prior notice.

The products depicted in this manual may include software copyrighted by Newland Europe BV or a third party. The user, corporation or individual shall not duplicate, in whole or in part, distribute, modify, decompile, disassemble, decode, reverse engineer, rent, transfer, or sublicense such software without prior written consent from the copyright holders.

This manual is copyrighted. No part of this publication may be reproduced, distributed, or used in any form without Newland Europe BV's written permission.

Risk Warning Regarding Unauthorized System Updates:
You should use the Newland-provided tool to update this product's system. Modifying system files by installing a third-party ROM system or using any cracking method may result in product malfunction or data loss and void your warranty.

Newland Europe BV reserves the right to make a final interpretation of the statement above.

Newland Europe BV

Rolweg 25, 4104 AV, Culemborg,
The Netherlands
www.newland-id.com

Newland Europe BV is a subsidiary of Newland Digital Technology Co., Ltd. Our general conditions of Purchase, Sale and Delivery are filed with the Record Office of the Chamber of Commerce of Utrecht, The Netherlands.

K.v.K. H.R. Utrecht / Chamber of Commerce Utrecht: Reg. nr. 17109876

Revision History

Version	Description	Date
V1.0.0	Initial release.	12 Jan 2024
V1.0.1	New sections: 5.7.2 Communication Data Rate 5.7.4 Extension Send Command (001D) 5.7.5 Pass-through Transmission (001C) 5.16.7 Read Data (00B9) Updated section: Example 7: MIFARE PLUS-L1	15 Jan 2026

Table of Contents

Chapter 1 Getting Started

Near-field communication, or NFC, is an emerging technology that allows devices like smartphones to exchange data with other devices in a short distance. NFC is an evolution from the integration of contactless RFID and interconnection technology. By integrating functions of inductive card readers, inductive cards and point-to-point communication on a chip, it can realize mobile payment, e-ticketing, access control, mobile identification and anti-counterfeiting by using mobile terminals.

The transmission distance of NFC is much smaller than that of RFID. The transmission distance of RFID is between 0 and 1m. Compared with RFID, NFC has the characteristics of low cost, high bandwidth and low energy consumption with attenuation technology.

NFC is a short-range (<10cm) wireless communication technology.

- Operating frequency: 13.56MHz.
- Compatible with ISO 14443, ISO 15693 and FeliCa standard.
- Transmission speed: 106kbit/s, 212kbit/s and 424kbit/s.

Chapter 2 Supported NFC Protocol

FM3281 supports four protocol standards as below:

1. ISO 14443-A
2. ISO 14443-B
3. ISO 15693
4. FeliCa

Chapter 3 Supported NFC Card Type

Introduction

FM3281 supports:

1. MIFARE series card
2. NTAG213/215/216 card
3. Ultralight/C/EV1 card
4. NTAG 42x DNA/ TT card
5. ICODE2 card
6. FeliCa card

3.1 MIFARE Series Card

Including MIFARE DESfire, MIFARE Classic, MIFARE Ultralight, and MIFARE Plus.

3.1.1 MIFARE DESfire

The MIFARE DESfire series provides highly secure ICs based on microcontrollers. DESfire stands for the use of DES, 2K3DES, 3K3DES, and AES hardware cryptographic engines for protecting transmission data.

This series is suitable for developers and operators to establish reliable, inter-active and extensible non-contact solutions. MIFARE DESfire series can be integrated into mobile schemes. It is widely used for identity, access control, loyalty, payment, and transport ticketing.

Including MIFARE DESfire EVx and MIFARE DESfire light.

Products	Explain	ISO/IEC	Security	Certification
MIFARE DESfire Evx	High-security IC for contact-free smart urban services.	ISO/IEC 14443 A 1-4 & ISO/IEC 7816	DES/2K3DES/3K2DES/AES Encryption Algorithm	CC EAL5+
MIFARE DESfire light	Safe, easy to integrate, cost-effective,	ISO/IEC 14443 A	AES 128-bit and LRP authentication and secure	CC EAL4

Products	Explain	ISO/IEC	Security	Certification
	contact-free, integrated circuit	1-4 & ISO/IEC 7816	messaging	

3.1.1.1 MIFARE DESfire EVx

	MIFARE DESfire EV3	MIFARE DESfire EV2	MIFARE DESfire EV1
ISO/IEC 14443 A 1-4	Support	Support	Support
Supports ISO/IEC 7816-4	Extensible	Extensible	Extensible
EEPROM data memory	2/4/8KB	2/4/8/16/32KB	2/4/8KB
Flexible file structures	Support	Support	Support
NFC Forum Class 4 Tags	Support	Support	Support
Unique ID	7 B UID or 4 B RID	7 B UID or 4 B RID	7 B UID or 4 B RID
Number of applications	Supported Memory Size	Supported Memory Size	28
Number of files per app	32	32	32
Data transfer rates supported	Up to 848 Kbit/s	Up to 848 Kbit/s	Up to 848 Kbit/s
Supported encryption algorithms	DES/2K3DES/ 3K3DES/ AES128	DES/2K3DES/ 3K3DES/ AES128	DES/2K3DES/ 3K3DES/ AES128

	MIFARE DESfire EV3	MIFARE DESfire EV2	MIFARE DESfire EV1
Delegated Application Management (Multiple Applications)	Yes, preloaded key	Support	-
SUN (Secure Unique NFC Message)	Yes, compatible with NTAG DNA	-	-
Transaction MAC per application	Support	Support	-
Multiple sets of keys per application	Up to 16 keys	Up to 16 keys	-
Multiple files can be accessed	Up to 8 keys	Up to 8 keys	-
File sharing between applications	Support	Support	
Transaction timer	Support	-	-
Virtual Card Architecture	Support	Support	-
Near-field verification	Support	Support	-
Transport type	Wafer, MOA4 and MOA8	Wafer, MOA4 and MOA6	Wafer, MOA4 and MOA8

3.1.1.2 MIFARE DESfire Light

The MIFARE DESfire Light is a contactless IC designed for easy integration into new and existing systems. Its predefined file system and 640 bytes of total available memory (equivalent to 1 kB of MIFARE Classic) make it an excellent choice for single-application designs in a variety of use cases. The MIFARE DESfire Light is compatible with MIFARE DESfire EV2.

MIFARE DESfire Light are designed for limited and extended use applications and include appropriate protection mechanisms to support trusted services. Depending on the use case, up to five AES 128-bit keys can be used to manage access, while secure messaging options enhance data and privacy protection. All hardware and software security features of the chip have been externally reviewed, tested and certified in accordance with Common Criteria EAL4.

3.1.2 MIFARE Classic

Can be used in applications such as public transport ticketing; Major cities have chosen MIFARE as their electronic ticketing solution. Applications:

- Public Transport
- Access Control
- Event Ticketing
- Gaming and Authentication

The Smart Anti-Jamming feature allows multiple cards to be operated simultaneously in the field. The anti-tamper algorithm selects each card separately and ensures that the selected card performs the transaction correctly and does not cause data corruption due to the presence of other cards. MIFARE classic is designed for easy integration and convenience. This allows a complete ticketing transaction to be completed within 100 ms. As a result, users using the MIFARE class card will not spend a lot of time on ticket transactions, thereby avoiding congestion at the entrance and reducing the boarding time of the bus. The MIFARE card can be placed in the wallet to conduct transactions, even if there are coin-like metal objects in the wallet, there will be no impact on communication.

Products	ISO/IEC	Bit Rate	Security	UID Type	Write Operation Tolerance	EEPROM
MIFARE Classic EV1	14443-Type 3A	106	MIFARE CRYPTO1	4NUID & 7 UID	100000	1KB – 4KB

3.1.3 MIFARE Plus

The MIFARE Plus series adds security to smart city services by supporting the seamless migration of contactless infrastructure to higher security, and its backward compatibility allows for cost-effective upgrading of the security level of native smart card applications. Special features, such as support for mobile and wireless top-ups, make it easier to use smart city services.

Products	Explain	ISO/IEC	Security
MIFARE Plus	It is designed both as a gateway for a new	ISO/IEC 14443 A 1-	48-bit

Products	Explain	ISO/IEC	Security
EVx	Smart City and a security upgrade for legacy infrastructures.	4 & ISO 7816-4	Crypto-1, 128-bit AES
MIFARE Plus SE	An entry-level version of the MIFARE Plus series.	ISO/IEC 14443 A 1-4	48-bit Crypto-1, 128-bit AES

3.1.3.1 MIFARE Plus Evx

Memory	MIFARE Plus EV2	MIFARE Plus X
Memory configuration	Block/sector structures	Block/sector structures
Memory size	2 kB / 4 kB	2 kB / 4 kB
ISO/IEC	ISO/IEC 14443 A 1-4 ISO/IEC 7816	ISO/IEC 14443 A 1-4 ISO/IEC 7816
UID/ONUID	7 B UID or 4 B ONUID	7 B UID or 4 B ONUID
Data transfer rate	Up to 848 kbps according to ISO/IEC 14444 -4	Up to 848 kbps according to ISO/IEC 14444 -4
Algorithm	AES 128-bit, Secure Messaging, Legacy Encryption1	AES 128-bit, Secure Messaging, Legacy Encryption1
Security level concept	Sector-by-sector or card-by-card	Card only
SL1SL3MixMode	Secure backend connected to SL1 sector	-
Transaction MAC (TMAC)	Secure verification of back-end Transactions	-

Memory	MIFARE Plus EV2	MIFARE Plus X
Transaction timer	Mitigating man-in-the-middle attack	-

3.1.3.2 MIFARE Plus SE

MIFARE Plus is the only mainstream smart card family compatible with MIFARE Classic 1K and MIFARE Classic 4K that provides pre-issued cards pending security upgrades to the infrastructure. With security upgraded to Level 3, MIFARE Plus uses the Advanced Encryption Standard (AES) for authentication, data integrity and encryption.

MIFARE Plus SE is the entry-level version of the proven, reliable MIFARE Plus product family. It is fully compatible with MIFARE Classic 1K functionality, providing complete support for MIFARE Classic value blocks. MIFARE Plus SE is the first choice for customers who are ready to switch to a higher level of security, and who will be preparing for the future by introducing ready-to-use cards for AES security into their existing system environments.

The MIFARE Plus SE card is easy to distribute and can run MIFARE Classic systems because it uses a linear memory structure compatible with MIFARE Classic and because MIFARE Plus SE supports all MIFARE Classic value block operations in security levels SL1 and SL3.MIFARE Plus SE stores its 128-bit AES key at the top of the data block. optional AES authentication in SL1 effectively detects cards that do not belong to the system.

3.1.4 MIFARE Ultralight

Ideal for low-cost, high-volume applications such as public transportation, loyalty cards, and event tickets.

Product	Explain	ISO/IEC	Security
MIFARE Ultralight AES (new)	Secure, easy to integrate, contact-free IC for limited applications	ISO/IEC 14443 A 1-3	128-bit AES authentication for data and counter protection and CMAC data integrity protection via RF interface
MIFARE Ultralight C	Contact-free IC supports 3DES Encryption in limited applications	ISO/IEC 14443 A 1-3	112-bit 3DES
MIFARE Ultralight	Contact-free IC with	ISO/IEC	32-bit password + password

Product	Explain	ISO/IEC	Security
EV 1	password protection for limited smart paper tickets and cards	14443 A 1-3	confirmation

3.2 NTAG Series Cards

NTAG is the market leading portfolio of NFC tag IC solutions for the IoT consumer and industrial sectors. These powerful NFC tags offer different levels of security and functionality to meet a wide range of applications and customer requirements. Companies can now introduce smart, digitally connected products that catalyze new value throughout the lifecycle. With a broad NTAG feature set, they can also enable a whole new user experience through more dynamic and advanced levels of personalization.

Passively powered NFC tag and tag IC solutions include the NTAG 21x, NTAG 213 tag tamper detection and the cryptographically secure NTAG DNA series. NTAG products are fully compliant with the NFC Forum standards and cover both Class 2 and Class 4 tags. NTAG ICs can store NDEF (NFC Data Exchange Format) data, making them fully compatible with any NFC device.

Product	Memory Access Protection	Encrypted Communication	NFC Hit Counter	Originality Signature	Secure NDEF (SUN) Message	Tamper Detection
NTAG 424 DNA Tag Tamper Detection	AES-128 bit key (5)			56 bytes		Primary open and current status
NTAG 424 DNA	AES-128 bit key (5)			56 bytes		-
NTAG 213 Tag Tamper Detection	32-bit password	-		32 bytes programmable	-	Once-open and current state
NTAG 213/215/216	32-bit password	-		32 bytes	-	-

Product	Standard Edition	User Memory [bytes]	Data Retention Period [years]	Write Operation Endurance [times]	Label Certification
NTAG 424 DNA Tag Tamper Detection	ISO /IEC 14443-A NFC Forum T4T	416, including 128b security data files	50	200,000	AES-128 bit key (5)
NTAG 424 DNA	ISO /IEC 14443-A NFC Forum T4T	416, including 128b security data files	50	200,000	AES-128 bit key (5)
NTAG 213 Tag Tamper Detection	ISO 14443A 1-3 NFC Forum T2T	144	10	100,000	-
NTAG 213/215/216	ISO 14443A 1-3 NFC Forum T2T	144 / 504 / 888	10	100,000	-

3.3 ICODE

The short-range RFID solution is compliant with ISO/IEC 15693 and ISO/IEC 18000-3 standards and follows the NFC Forum Tag Type 5 specification. ICODE offers a working range of up to 1.5 meters and a remote reader, ICODE also offers an additional read range compared to the ISO/IEC 14443 and a standard ISO/IEC 15693 reader for an ultra-small form factor and NFC handset readability: The ICODE product specification has a range of valuable features. Such as protection of EAS, Application Family Identifier (AFI), memory access, and privacy.

ICODE products are used in a wide range of applications, as listed below:

- Library management
- Identification of consumables and accessories
- Brand protection and anti-counterfeiting
- Supply chain visibility and control
- Industrial use

Product	Standard Edition	User memory [bit]	EPC Code Size [Bits]	EAS protection	EAS Selectivity

Product	Standard Edition	User memory [bit]	EPC Code Size [Bits]	EAS protection	EAS Selectivity
ICODE SLIX 2	ISO 18000-3M1 NFC Forum T5T	2528	-	32-bit password	
ICODE SLIX	ISO 18000-3M1 NFC Forum T5T	896	-	32-bit password	-
ICODE SLIX-L	ISO 18000-3M1 NFC Forum T5T	256	-	32-bit password	
ICODE SLIX-S	ISO 18000-3M1 NFC Forum T5T	1280	-	32-bit password	

	AFI	AFI Protection	Keep quiet	Original signature	Memory protection
ICODE SLIX 2		32-bit password			32-bit password
ICODE SLIX		32-bit password	-	-	-
ICODE SLIX-L		32-bit password	-	-	32-bit password
ICODE SLIX-S		32-bit password	-	-	32-bit password

3.4 FeliCa

FeliCa has the same technology that is applicable to cash or identification cards as regular IC cards, but the instruction set is specialized for applications that require high-speed processing features (automatic recharge devices, building access controls, etc.) or checkout (convenience stores), etc. Therefore, it is not compatible with the basic instructions of ISO 7816-3. In addition, the internal memory of the IC chip is fixed as 16-byte records, which is incompatible with the file structure stipulated by ISO 7816-3.

In terms of encryption processing, Triple DES is used for mutual authentication and DES or Triple DES for communication. There is no public key encryption specification. Dual model (contact/non-contact) is only used for contact communication, although it can be encrypted with a public key.

In mutual authentication, the indented code is used as the password of the solution. Instead of saying that each item is authenticated individually, it is encrypted by a complex access code. The key generated is called a flinch code, and this flinch code can be used by up to 16 items. The indented code does not produce the original password. In this way, high speed processing is achieved without compromising the level of security

Applies to:
Public transport
Automatic recharge device
Building access control
Checkout (Convenience Store)

Chapter 4 General Function

4.1 Working Mode Setting

FM3281 supports command mode, reporting mode and service mode to meet different application requirements.

1. Reporting mode: The equipment is always in the process of finding the card. When the card enters the valid range, the UID is automatically output, and other business instructions are not allowed.
2. Command mode: card request, anti-collision, card selection and other operations are controlled by the upper computer software.
3. Business mode: When the equipment is initialized in the process of card searching, the UID will be automatically output when the card enters the valid range. It can also be operated by other business instructions. After the operation is completed, it is necessary to re-control the access to the card search according to its own needs.

Commands

NFCMOD0 --Reporting Mode

NFCMOD1 --Command Mode

NFCMOD2 --Business Mode

4.2 Time Interval for Searching Card

Set the time interval for card searching, and the unit is ms. The smaller the time interval for card searching is, the higher the power consumption of the device is, and the faster the card is read.

Command

NFCDUR$ (Range of $ if from 10 to 1000ms)

4.3 Return to Card Type

Whether to return card type and output ASCII character string when automatically outputting UID. Card type includes protocol type (2 bytes) + ATQA (4 bytes) + SAK (2 bytes). The protocol type contains:

01：ISO14443-A

02：ISO14443-B

03：ISO15693

04：FELICA

05: Chinese 2nd generation ID card

Commands

NFCCTP0: Disable Output Card Type

NFCCTP1: Enable Output Card Type

4.4 Enable Switch

NFC enable switch.

Commands

NFCENA0: Disable NFC function

NFCENA 1: Enable NFC function

4.5 Reread Delay

NFC reread delay switch.

Commands

NFCRDE0: NFC reread delay is invalid. The same NFC tag can be read continuously at any time.

NFCRDE1: If an NFC tag is read and the NFC tag is read for the secondary consecutive time within the NFC re-read delay time, the NFC tag read for the secondary time will be ignored and will not be output.

4.6 Reread Delay Time Settings

Reread delay.

Command

NFCRDT$: Range of $ is from 1ms to 3600000ms.

Chapter 5 General Function

5.1 Command Format

Host → Scanner

CMD	Function Code	Data
NFCCMD	4 Bytes	N Bytes

- CMD: Newland Unified Command.
- Function Code: Card Operation Function Code.
- Data: Host sends to the card data in hexadecimal string format. For example, HEX: 0x41, ASCII character 41. (Can be empty).

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	4 Bytes	N Bytes	2 Bytes	N Bytes

- CMD: The unified command NFC CMD is consistent with that sent by Host.
- Function Code: Card operation function code is consistent with that sent by Host.
- Data: Host sends to the card data in hexadecimal string format. For example, HEX: 0x41, ASCII character 41. (Can be null) is consistent with that sent by Host.
- Status: 00, 71 for success, others for failure. 2 bytes.
- Card Data: Data returned by the card. (In hexadecimal string format, for example, if the data obtained from the card is HEX: 0x41, then the output of the scanner is ASCII character 41, which can be null).

5.2 Select Protocol

If the reader supports cards of multiple protocols, the corresponding card protocol needs to be selected before card operation. Applies to the command mode.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0010	2 Bytes

Data: the protocol type to be selected, and the value is 00--TypeA; 01--TypeB; 02--ICODE2; 03 – FeliCa 2 bytes.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0010	N byte	2 Bytes	NULL

5.3 Re-find Card

When the working mode is set as the business mode, you need to use this command to actively find a card again after finding the card.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0011	NULL

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0011	NULL	2 Bytes	NULL

5.4 Format KeyEntry

Format a key entry to a new KeyType into KeyStore.

Host → Scanner

CMD	Function Code	Data
NFCCMD	002B	8 Bytes

Data: The following length and contents are before conversion to hexadecimal strings.

The data packet data is wKeyNo (2 bytes, LSB first) and wKeyType (2 bytes, LSB first).

wKeyNo: Key number of the key to be loaded. Range: from 0 to 3.

wKeyType: New Key type of the KeyEntry (predefined type of KeyType).

The values are as follows:

0x00 --AES 128 Key [16 bytes]

0x04 --2 Key Triple Des.[16 bytes]

0x05 --3 Key Triple Des.[24 bytes]

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	002B	8 Bytes	2 Bytes	NULL

5.5 SetKey

Change a key entry at a given version in Key Store.

Host → Scanner

CMD	Function Code	Data
NFCCMD	002C	48/64 Bytes

Data: The following length and contents are before conversion to hexadecimal strings.

Packet data is wKeyNo, wKeyVersion, wNewKeyType, pNewKey, wNewKeyVersion

wKeyNo: Key number of the key in Key Store, 2 bytes and LSB first.

The value is from 0 to 3.

wKeyVersion: Key version of the **wKeyNo** in Key Store, 2 bytes and LSB first.

wNewKeyType: New Key type of the **wKeyNo**, 2 bytes and LSB first.

The values are as follows:

0x04 --2 Key Triple Des [16 bytes].

0x05 --3 Key Triple Des [24 bytes].

pNewKey: The new key to be changed, 16 bytes or 24 bytes.

wNewKeyVersion: New Key Version of the **wKeyNo** to set, 2 bytes and LSB first.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	002C	48/64 bytes	2 bytes	NULL

Call **Format Key Entry** and Set Key to load the key into Key Store to authenticate the DESfire EV1/EV2/EV3/Light or NTag42X card key.

5.6 ISO14443-A Series Card Activation Instruction

5.6.1 Request Card

Host → Scanner

CMD	Function Code	Data
NFCCMD	0012	2 Bytes

Data: request mode, 2 bytes

00: With the halt command, the card can only be requested to be placed on the scanner once. If it is operated again, it needs to be removed and placed again to prevent multiple operations on the card.

01: The request has been successful.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0012	2 Bytes	2 Bytes	4 Bytes

Card Data: ATQA Data.

5.6.2 Anti-collision

Host → Scanner

CMD	Function Code	Data
NFCCMD	0013	NULL

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0013	NULL	2 Bytes	8 Bytes

Card Data: default serial number of card.

5.6.3 Select Card

Host → Scanner

CMD	Function Code	Data
NFCCMD	0014	8 Bytes

Data: serial number of the card returned for anti-collision.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0014	8 Bytes	2 Bytes	2 Bytes

Card Data: SAK Data.

5.6.4 Secondary anti-collision

Secondary anti-collision and secondary card selection are required for the 7-byte UID card.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0015	NULL

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0015	NULL	2 Bytes	8 Bytes

Card Data: the secondary serial number returned by the card.

Note: 0 byte is removed from the card serial number returned by the first anti-collision, and the remaining 3 bytes and the 4 bytes returned by the secondary anti-collision finally form a 7-byte UID.
For example, the serial number returned by the first anti-collision is {0x88, 0x04, 0xFD, 0xF4}.
The sequence number returned by the secondary anti-collision is {0xEA, 0 xA9, 0x6A, 0x80},
The final card serial number is {0x04, 0xFD, 0 xF4, 0xEA, 0xA9, 0x6A, 0x80}

5.6 Stop Card

Host → Scanner

CMD	Function Code	Data
NFCCMD	0017	NULL

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0017	NULL	2 Bytes	NULL

5.7 Contactless CPU Card (ISO14443 TypeA)

Operation process of non-connected CPU card:

- Command mode: request-> anti-collision-> select card-> secondary anti-collision-> secondary select card-> reset-> send command-> stop
- Service mode: UID received-> reset-> send command-> stop-> find card

5.7.1 Reset

Host → Scanner

CMD	Function Code	Data
NFCCMD	0018	NULL

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0018	NULL	2 Bytes	N Bytes

Card Data: It is the reset response information returned by the card (ATS).

5.7.2 Communication Data Rate

To modify the communication data rate, the command must be sent as the first one after a reset. The default communication data rate is 106 kbps.

Host → Scanner

CMD	Function Code	Data
NFCCMD	001F	6 Bytes

Data: The following length and contents are before conversion to hexadecimal strings.

Packets data are **Protocol, bDri, bDsi**

Protocol: 0x01

bDri : (Current) Divisor Receive (PCD to PICC) Integer; 0-3; */

0x00 :106 kBit/s

0x01: 212 kBit/s

0x02: 424 kBit/s

0x03: 848 kBit/s

bDsi : Current) Divisor Send (PICC to PCD) Integer; 0-3;

0x00 :106 kBit/s

0x01: 212 kBit/s

0x02: 424 kBit/s

0x03: 848 kBit/s

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	001F	6 Bytes	2 Bytes	NULL

5.7.3 Send Command

Pass-through application protocol command.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0019	N Bytes

Data:

NAD (default 00)

CID (default 00)

PCB (default 00)

LEN: (COS instruction length)

Data [4] -DATA [LEN + 4] COS command

For example, ISO7816-4 gets the random number.

CLA 00

INS 84

P1 00

P2 00

Le 08

Send command: NFCCMD0019000000050084000008

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0019	N Bytes	2 Bytes	N Bytes

Card Data: data returned by pass through command.

5.7.4 Extension Send Command

This command sends an APDU command with extended length (LC/LE is 3 bytes, the maximum value of LC is 490, and the maximum value of LE is 502). The data can be sent completely at one time without packet segmentation. If LC/LE exceeds their maximum values, the ISO14443 transparent transmission command can be used for packet segmentation processing.

Host → Scanner

CMD	Function Code	Data
NFCCMD	001D	N Bytes

The data packet 'data' is the COS command to be sent, and N is the length of the APDU command.

For example, when writing a binary file, write 264 bytes of data starting from offset 0.

data[0] = 0x00; //CLA

data[1] = 0xD6; //INS

data[2] = 0x00; //P1

data[3] = 0x00; //P2

data[4] = 0x00; //LC high byte

data[5] = 0x01; //LC

data[6] = 0x08; //LC low byte

data[7]...data[N - 1]: 264 bytes of data to be written.

Example: Send 264-byte data

Step 1: Discovery

Request: NFCCMD0011

Response: NFCCMD001100<ACK>;

Step 2: Card near output UID

Device: 040E4102BC1E90

Step 3: Reset

Request: NFCCMD0018

Response: NFCCMD001800067877810280<ACK>;

Step 4: Select MF

Request: NFCCMD001D00A40000020100

Response: NFCCMD001D00A40000020100009000<ACK>;

Step 5: Select File

Request: NFCCMD001D00A40000020100

Response: NFCCMD001D00A40000020100009000<ACK>;

Step 6: Update Binary

Request:

NFCCMD001D00D6000000010811223344556677889900112233445566778899001122334455667788
99001122334455667788990011223344556677889900112233445566778899001122334455667788990
01122334455667788990011223344556677889900112233445566778899001122334455667788990011
22334455667788990011223344556677889900112233445566778899001122334455667788990011223
34455667788990011223344556677889900112233445566778899001122334455667788990011223344
55667788990011223344556677889900112233445566778899001122334455667788990011223344556
677889900112233445566778899001122334455667788990011223344

Response:

NFCCMD001D00D6000000010811223344556677889900112233445566778899001122334455667788
99001122334455667788990011223344556677889900112233445566778899001122334455667788990
01122334455667788990011223344556677889900112233445566778899001122334455667788990011
22334455667788990011223344556677889900112233445566778899001122334455667788990011223
34455667788990011223344556677889900112233445566778899001122334455667788990011223344
55667788990011223344556677889900112233445566778899001122334455667788990011223344556
677889900112233445566778899001122334455667788990011223344009000<ACK>;

Step 7: Read Binary

Request: NFCCMD001D00B00000000108

Response:

NFCCMD001D00B00000000108001122334455667788990011223344556677889900112233445566778
89900112233445566778899001122334455667788990011223344556677889900112233445566778899
00112233445566778899001122334455667788990011223344556677889900112233445566778899001
12233445566778899001122334455667788990011223344556677889900112233445566778899001122

3344556677889900112233445566778899001122334455667788990011223344556677889900112233445566778899001122334455667788990011223344556677889900112233445566778899001122334455667788990011223344556677889900112233449000<ACK>;

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	001D	N Bytes	2 Bytes	N Bytes

Card Data: data returned by pass through command.

5.7.5 Pass-through Transmission

Pass-through application protocol command. This command can be used to send commands that transmit data exceeding 243 bytes. Data larger than 243 bytes must be sent in fragmented packets, requiring PCB configuration. If the data length of an APDU exceeds 255 bytes, the LC field must use extended length (3 bytes). For specific implementation methods, please refer to the ISO7816-4 documentation

Host → Scanner

CMD	Function Code	Data
NFCCMD	001C	N Bytes

The data packet data contains the data to be sent, where N denotes the length defined by the ISO14443 protocol. Specific meanings are as follows:

data[0] = 0x11; // Fixed

ISO14443 CMD:

data[1] = PCB;

// Bit 0: Block number (alternates between 0 and 1)

// Bit 1: Fixed to 1

// Bit 2: For I-blocks, 1 indicates NAD follows; for R-blocks, fixed to 0

// Bit 3: 1 indicates CID follows

// Bit 4: For I-blocks: Chaining bit (1=more data follows; 0=final block)

For R-blocks: 0=ACK (acknowledge); 1=NAK (negative acknowledge)

// Bit 5: 0 for I-blocks; 1 for R-blocks

// Bits 6–7: '00' for I-blocks; '10' for R-blocks

data[2]: CID or APDU command

data[3]: NAD or APDU command

data[4] to data[N]: APDU command"

Example 1：Get a random number

Step 1: Discovery

Request: NFCCMD0011

Response: NFCCMD001100<ACK>;

Step 2: Card near output UID

Device: 040E4102BC1E90

Step 3: Reset

Request: NFCCMD0018

Response: NFCCMD001800067877810280<ACK>;

Step 4:

Send command: NFCCMD001C11020084000008

response command： NFCCMD001C110200840000080002C2E58CC3314865FC9000

Example 2: Send 264 bytes of data

Step 1: Discovery

Request: NFCCMD0011

Response: NFCCMD001100<ACK>;

Step 2: Card near output UID

Device: 040E4102BC1E90

Step 3: Reset

Request: NFCCMD0018

Response: NFCCMD001800067877810280<ACK>;

Step 4: Select MF

Request: NFCCMD001C110200A40000020100

Response: NFCCMD001C110200A400000201000029000<ACK>;

Step 5: Select File

Request: NFCCMD001C110300A40000020100

Response: NFCCMD001C110300A400000201000039000<ACK>;

Step 6: UPDATE BINARY (APDU extension 3 bytes Lc)

Request:

NFCCMD001C111200D6000000010811111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111111111111

Response:

NFCCMD001C111200D6000000010811111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111100A2<A
CK>;

Request:

NFCCMD001C11132222222222222222222222222222222222222222222222222222222222222222222
222222222222222222222222222222222222222222222222222222222222222

Response:

NFCCMD001C11132222222222222222222222222222222222222222222222222222222222222222222
22222222222222222222222222222222222222222222222222222222222222200A3<ACK>;

Request:

NFCCMD001C11123333333333333333333333333333333333333333333333333333333333333333333
333333333333333333333333333333333333333333333333333333333333333

Response:

NFCCMD001C11123333333333333333333333333333333333333333333333333333333333333333333
33333333333333333333333333333333333333333333333333333333333333300A2<ACK>;

Request:

NFCCMD001C11134444444444444444444444444444444444444444444444444444444444444444444
444444444444444444444444444444444444444444444444444444444444444

Response:
NFCCMD001C1113444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444400A3<ACK>;
Request: NFCCMD001C11025555555555555555
Response: NFCCMD001C1102555555555555555500029000<ACK>;

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	001C	N Bytes	2 Bytes	N Bytes

Card Data: data returned by pass through command.

The result data contains the length of ISO14443 protocol information returned by the card. Specific meanings are as follows:

- Data[0]: PCB; // See PCB in sent data
- Data[1]: CID or card response data
- Data[2]: NAD or card response data
- Data[3] to data[N - 1]: Card response data

5.7.6 Stop Card

Host → Scanner

CMD	Function Code	Data
NFCCMD	001A	NULL

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	001A	NULL	2 byte	NULL

5.8 MIFARE DESfire Light

Operation process of MIFARE DESfire light card:

- Command mode: request-> anti-collision-> select card-> secondary anti-collision-> secondary select card-> reset-> send command to verify key, read and write-> stop
- Service mode: receive UID-> reset-> send command to verify key, read and write-> stop-> find card

5.8.1 Authenticate EV2

Performs an authentication based on standard AES. This will be using the AES128 keys and will generate and verify the contents based on generic AES algorithm.

Host → Scanner

CMD	Function Code	Data
NFCCMD	002D	N Bytes

Data: The following length and contents are before conversion to hexadecimal strings

Packets data are bFirstAuth, wOption, wKeyNo, wKeyVer, bKeyNoCard, bDivLen, pDivInput, bLenPcdCapsIn, bPcdCapsIn.

bFirstAuth: 1 byte,One of the below options

0x00 --Non First Auth in regular EV2 auth Mode.

0x01 --First Auth in regular EV2 auth Mode.

0x02 --Non First Auth in LRP mode.

0x03 --First Auth in LRP mode.

wOption: Diversification option, 2 bytes. The values are as follows

0xFFFF --No diversification.

0x0000 --Encryption based method of diversification.

0x0001 --CMAC based method of diversification.

wKeyNo: The key number in keystore to authenticate, 2 bytes and LSB first.

wKeyVer: Key version in the key store, 2 bytes and LSB first.

bKeyNoCard: Key number on card, 1 byte.

bDivLen: Length of diversification input, 1 byte.

pDivInput: Diversification input, up to 16 bytes. Can be NULL.

bLenPcdCapsIn: Length of PcdCapsIn, 1 byte.Always 0 for following authentication.

pPcdCapsIn: PCD Capabilities. Up to 6 bytes.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	002D	N Bytes	2 Bytes	24 Bytes

Card Data: The following length and contents are before conversion to hexadecimal strings

Result data data is bPcdCapsOut, bPdCapsOut.

bPcdCapsOut: PCD Capabilities, 6 bytes.

bPdCapsOut: PD Capabilities, 6 bytes.

5.8.2 Get Key Version

Read out the current key version of any key stored on the PICC.

Host → Scanner

CMD	Function Code	Data
NFCCMD	002F	4 Bytes

Data: The following length and contents are before conversion to hexadecimal strings

The data packet data is bKeyNo and bKeySetNo in turn

bKeyNo: Key number of the targeted key, 1 byte.

bKeySetNo: Key set number, 1 byte. Optional as it is passed only when bit6 of **bKeyNo** is set.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	002F	4 Bytes	2 Bytes	N Bytes

Card Data: data is the version of the specified key.

5.8.3 Get Version

Returns manufacturing related data of the PICC.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0030	NULL

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0030	NULL	2 Bytes	56

5.8.4 Set Configuration

Configures the card and pre personalizes the card with a key, defines if the UID or the random ID is sent back during communication setup and configures the ATS string.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0031	N Bytes

Data: The following length and contents are before conversion to hexadecimal strings

The data packets data are bOption, bDataLen, pData.

bOption:Define length and content of the **pData**, 1 byte.

0x00 --Update the PICC Configuration.

0x02 --Update the ATS.

0x03 --Update the SAK.

0x04 --Update the Secure Messaging.

0x05 --Update Capibility Data.

0x06 --Application Renaming.

0x08 --File Renaming.

0x09 --Value file type configuration.

0x0A --Failed Authentication Counter Configuration.

0x0B --Hardware Configuration.

bDataLen:The size of **pData**, 1 byte.

pData:Configuration data for the option specified.The length of bytes is specified by **bDataLen**.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0031	N Bytes	2 Bytes	NULL

5.8.5 Get Card UID

Returns the file identifiers of all active files within the currently selected application. Each File ID is coded in one byte.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0032	NULL

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0032	NULL	2 Bytes	14 Bytes

The following length and contents are before conversion to hexadecimal strings.

Card data: data is the 7-byte UID returned.

5.8.6 Get File IDs

Returns the file identifiers of all active files within the currently selected application. Each File ID is coded in one byte.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0033	NULL

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0033	NULL	2 Bytes	N Bytes

The following length and contents are before conversion to hexadecimal strings.

Card Data: Results data is the fields, if the Transaction MAC file is present 6 file IDs will be returned, otherwise 5 file IDs returned.

5.8.7 Get ISO File IDs

Returns the file identifiers of all active files within the currently selected application. Each File ID is coded in one byte.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0035	2 Bytes

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0035	2 Bytes	2 Bytes	N Bytes

The following length and contents are before conversion to hexadecimal strings.

Card Data: Data is the fields of the 3 Standard data files and the Cyclic Record file.

5.8.8 Get File Settings

Get information on the properties of a specific file.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0035	2 Bytes

Data: data is the file number of the targeted file.

Scanner → Host

→	Function Code	Data	Status	Card Data
NFCCMD	0035	2 Bytes	2 Bytes	N Bytes

Card Data: N bytes of the file settings.

5.8.9 Read Data

Read data from standard data files or backup data files.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0039	18 Bytes

Data: The following length and contents are before conversion to hexadecimal strings.

The data packet data is bOption, bIns, bFileNo, pOffset, pLength.

bCommOption:Communication settings for the file, 1 byte.

0x00 --Plain mode of communication.

0x10 --MAC mode of communication.

0x30 --Enciphered mode of communication.

bIns: The ISO14443-4 chaining format, 1 byte. This should always be set to 1.

bFileNo: File number to be read data, 1 byte.

pOffset: Starting position for the read operation, 3 bytes and LSB first.

pLength:The number of bytes to be read, 3 bytes and LSB first.

Ifitis 0x000000, Read the entire data file, starting from the position specified in the **pOffset** value.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0039	18 Bytes	2 Bytes	N Bytes

Card Data: If more data is to be read, the status **0x71** is returned, then should call This command again with **bCommOption=| 0x02**.

5.8.10 Write Data

Write data to standard data files or backup data files.

Host → Scanner

CMD	Function Code	Data
NFCCMD	003A	N Bytes

Data: The following length and contents are before conversion to hexadecimal strings

Packet data is bOption, bIns, bFileNo, pOffset, pTxDataLen, pTxData.

bOption:Communication settings for the file, 1 byte.

0x00 --Plain mode of communication.

0x10 --MAC mode of communication.

0x30 --Enciphered mode of communication.

bIns:The ISO14443-4 chaining format, 1 byte. This should always be set to 1.

bFileNo: File number to be writedata, 1 byte.

pOffset: Starting position for the writeoperation, 3 bytes and LSB first.

pTxDataLen:The length of data to be written, 3 bytes and LSB first.

pTxData: Data to be written, the length of bytes is specified by **pTxDataLen**.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	003A	N Bytes	2 Bytes	NULL

5.8.11 ISO Select File

This command is a standard ISO/IEC 7816-4 command. It selects either the PICC level, an application or a file within the application.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0046	N Bytes

Data: The following length and contents are before conversion to hexadecimal strings.

Packet data are bOption, **bSelector**, **pFid**, **pDFName**, **bDFNameLen**,**bExtendedLenApdu**.

bOption: Indicates whether to return File Control Information (FCI), 1 byte.

0x00 --Return FCI;

0x0C --No return FCI.

bSelector:Selection Control, 1 byte.

0x00 --Select MF, DF or EF, by file identifier.

0x01 --Select child DF.

0x02 --Select EF under the current DF, by file identifier.

0x03 --Select parent DF of the current DF.

0x04 --Select by DF name.

pFid: File Identifier, 2 bytes. Valid only when **bSelector**= 0x00 or 0x02.

bDFNameLen: The length of DFName, 1 byte. Valid only when **bOption**= 0x04.

pDFName: DF Name, up to 16 bytes. Valid only when **bSelector**=0x04.

bExtendedLenApdu: Default 0x00.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0046	N Bytes	2 Bytes	N Bytes

Card Data: The following length and contents are before conversion to hexadecimal strings.

data is the returned FCI stored in file ID 1Fh of the DF.

Valid only when **bOption**= 0x00.

5.8.12 Get Config

Perform a Get Config command.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0049	4 Bytes

Data: The following length and contents are before conversion to hexadecimal strings.

Packet data is wConfig.

wConfig: Configuration to read, 2 bytes. Will be one of the below values:

0xA100 --Get additional info of a generic error.

0xA200 --Get status of command wrapping in ISO 7816-4 APDUs.

0xA300 --Get Short Length APDU wrapping in ISO 7816-4 APDUs.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0049	4 Bytes	2 Bytes	N Bytes

Card Data: data is the value for the mentioned configuration.

5.8.13 Set Config

Perform a Set Config command.

Host → Scanner

CMD	Function Code	Data
NFCCMD	004A	4 Bytes

Data: The following length and contents are before conversion to hexadecimal strings.

Packets data are wConfig, wValue.

wConfig: Configuration to set, 2 bytes. Will be one of the below values:

0xA100 --Set additional info of a generic error.

0xA200 --Set current status of command wrapping in ISO 7816-4 APDUs.

0xA300 --Set Short Length APDU wrapping in ISO 7816-4 APDUs.

wValue: The value for the mentioned configuration, 2 bytes and LSB first.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	004A	4 Bytes	2 Bytes	NULL

Card Data: data is the value for the mentioned configuration.

5.8.14 Reset Authentication

Reset Authentication status.

Host → Scanner

CMD	Function Code	Data
NFCCMD	004B	NULL

Scanner Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	004B	NULL	2 Bytes	NULL

5.8.15 Read Sign

Performs the originality check to verify the genuineness of PICC.

Host → Scanner

CMD	Function Code	Data
NFCCMD	004E	NULL

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	004E	NULL	2 Bytes	112 Bytes

Card Data: The following length and contents are before conversion to hexadecimal strings.

data is the plain 56 bytes originality signature of the PICC.

5.8.16 Change Key

This command is used to change the application keys. Authentication with application key number 0 is required to change the key.

Host → Scanner

CMD	Function Code	Data
NFCCMD	002E	N Bytes

The following lengths and contents are before conversion to hexadecimal strings.

The packet data is **wOption, wOldKeyNo, wOldKeyVer, wNewKeyNo, wNewKeyVer, bKeyNoCard, bDivLen,pDivInput** in order.

wOption: Diversification option, 2 bytes. The values are one of the below options:

0xFFFF

0x0002 | 0x0020

0x0004 | 0x0020

0x0002 | 0x0008

0x0004 | 0x0010

0x0002 | 0x0004

0x0002 | 0x0004 | 0x0020

0x0002 | 0x0004 | 0x0008 | 0x0010

The value description:

0xFFFF--No diversification.

0x0002--Diversification of new key required (bit 1).

0x0004--Old key was diversified (bit 2).

0x0008--New key diversification using one rnd (bit 3).

Default is two rounds:

0x0010--Old key diversification using one rnd (bit 4).

Default is two rounds:

0x0020--Key diversification method based on CMAC (bit 5).

Default is Encryption method:

wOldKeyNo: Old key number in keystore, 2 bytes and LSB first.

wOldKeyVer: Old key version in keystore, 2 bytes and LSB first.

wNewKeyNo: New key number in keystore,2 bytes and LSB first.

wNewKeyVer: New key version in keystore,2 bytes and LSB first.

bKeyNoCard:Key number of the key to be changed, 1 byte.

bDivLen: Length of **pDivInput**, 1 byte.

pDivInput: Diversification input.Up to 16bytes,can be NULL.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	002E	N Bytes	2 Bytes	NULL

5.8.17 Get Value

Read the currently stored value from value files.

Host → Scanner

CMD	Function Code	Data
NFCCMD	003B	4 Bytes

The following lengths and contents are before conversion to hexadecimal strings.

Data: The packet data are **bOption** and **bFileNo** in order.

bOption:Communication settings for the file, 1 byte.

0x00 --Plain mode of communication.

0x10 --MAC mode of communication.

0x30 --Enciphered mode of communication.

bFileNo: File number to be read value, 1 byte.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	003B	4 Bytes	2 Bytes	8 Bytes

The following lengths and contents are before conversion to hexadecimal strings.

Card Data: The value read out, 4 bytes and LSB first.

5.8.18 Credit

Increase a value stored in a Value File.

Host → Scanner

CMD	Function Code	Data
NFCCMD	003C	12 Bytes

The following lengths and contents are before conversion to hexadecimal strings.

Data: The packet data is **bOption**, **bFileNo**, and **pValue** in order.

bOption: communication setting of the file, 1 byte.

0x00 --Plain communication mode.

0x10 --MAC communication mode.

0x30 --Enciphered mode of communication.

bFileNo: The file number to which the value should be credited, 1 byte.

pValue: Value to be credited, 4 bytes and LSB first.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	003C	12 Bytes	2 Bytes	NULL

5.8.19 Debit

Decrease a value stored in a Value File.

Host → Scanner

CMD	Function Code	Data
NFCCMD	003D	12 Bytes

The following lengths and contents are before conversion to hexadecimal strings.

Data: The packet data is **bOption**, **bFileNo** and **pValue** in order.

bOption:Communication settings for the file, 1 byte.

0x00 --Plain mode of communication.

0x10 --MAC mode of communication.

0x30 --Enciphered mode of communication.

bFileNo: The file number to which the value should be debited, 1 byte.

pValue: Value to be debited, 4 bytes and LSB first.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	003D	12 Bytes	2 Bytes	NULL

5.8.20 Limited Credit

Allows a limited increase of a value stored in a Value File without having full credit permissions to the file.

Host → Scanner

CMD	Function Code	Data
NFCCMD	003E	12 Bytes

The following lengths and contents are before conversion to hexadecimal strings.

Data: The packet data is **bOption**, **bFileNo** and **pValue** in order.

bOption:Communication settings for the file, 1 byte.

0x00 --Plain mode of communication.

0x10 --MAC mode of communication.

0x30 --Enciphered mode of communication.

bFileNo: The file number to which the value should be credited, 1 byte.

pValue: Value to be credited, 4 bytes and LSB first.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	003E	12 Bytes	2 Bytes	NULL

5.8.21 Commit Reader ID

Commit a Reader ID for the ongoing transaction. This will allow a backend to identify theat tacking merchant in case of fraud detected.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0043	32 Bytes

The following lengths and contents are before conversion to hexadecimal strings.

Data: The packet data is the transaction MAC reader ID, 16 bytes.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0043	32 Bytes	2 Bytes	32 Bytes

The following lengths and contents are before conversion to hexadecimal strings.

Card Data: Encrypted Transaction MAC Reader ID of last successful transaction, present if authenticated, 16 bytes.

5.8.22 Decrypt Reader ID

Decrypted the reader ID as performed on the PICC.

Host → Scanner

CMD	Function Code	Data
NFCCMD	004D	N Bytes

The following lengths and contents are before conversion to hexadecimal strings.

Data: The packet data is **wOption**,**wKeyNoTMACKey**, **wKeyVerTMACKey**, **wRamKeyNo**, **wRamKeyVer**,
bDivInputLen, **pDivInput**, **pTMC**, **bUidLen**, **pUid** and **pEncTMRI** in order.

wOption: Diversification option, 2 bytes.

0xFFFF --Disable key diversification.

0x0000 --Enable key diversification.

wKeyNoTMACKey: 2 bytes key number to be used from software keystore, LSB first.

wKeyVerTMACKey: 2 byts key version to be used from software keystore, LSB first.

wRamKeyNo: 2 byts key number of Destination Key where the computed session TMAC key will be stored, LSB first. To be used for SAM AV3 only.

wRamKeyVer: 2 bytes key version of Destination Key where the computed session TMAC key will be stored, LSB first. To be used for SAM AV3 only.

bDivInputLen: Length of **pDivInput**, 1 byte.

pDivInput: Diversification input to diversify TMACKey, the length of bytes is specified by **pDivInputLen**.

pTMC: Transaction MAC Counter, 4 byts.

bUidLen: Length of **pUid**, 1 byte.

pUid: UID of the card, the length of bytes is specified by **bUidLen**.

pEncTMRI: Encrypted Transaction MAC Reader ID of the latest successful transaction, 16 bytes.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	004D	N Bytes	2 Bytes	32 Bytes

Card Data: 16 bytes Decrypted Reader ID of the last successful transaction.

5.8.23 Commit Transaction

Validate all previous write access on Backup Data files, value files and record files within one application.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0044	2 Bytes

The following lengths and contents are before conversion to hexadecimal strings.

Data: The packet data is **bOption**.

bOption: One of the below options, 1 byte.

0x80 --No TMC and TMV returned.

0x81 --TMC and TMV returned.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0044	2 Bytes	2 Bytes	N Bytes

The resultant data is **pTMC**, **pTMV** in that order.

pTMC: Transaction MAC Counter (TMC), when bOption = 0x81, 4 bytes.

pTMV: Transaction MAC Value (TMV), when bOption = 0x81,8 bytes.

5.8.24 Abort Transaction

Aborts all previous write accesses on Backup Data files, value files and record files within one application.

Host->Scanner

CMD	Function Code	Data
NFCCMD	0045	NULL

Scanner ->Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0045	NULL	2 Bytes	NULL

5.9 MIFARE DESfire EV1/EV2/EV3

Operation process of MIFARE DESfire card:

- Command mode: request-> anti-collision-> select card-> secondary anti-collision-> secondary select card-> reset-> send command to verify key, read and write-> stop
- Service mode: receive UID-> reset-> send command to verify key, read and write-> stop-> find card

5.9.1 Authencation

Performs an Authentication with the specified key number， This command supports the backward compatible D40 authentication. The command can be used with DES and 2K3DES keys and performs DESfire native authentication.

Host → Scanner

CMD	Function Code	Data
NFCCMD	004F	N Bytes

The following length and contents are before conversion to hexadecimal strings.

Data: **wOption, wKeyNo, wKeyVer, bKeyNoCard, pDivInput and bDivLen** in order.

wOption: Diversification option, 2 bytes. The values are as follows

0xFFFF --No diversification.

0x8000 --Encryption based method, 2K3DES half key diversification.

0x0000 --Encryption based method, 2K3DES full key diversification.

0x0001 --CMAC based method of diversification.

wKeyNo: The key number in keystore to authenticate,2 bytes and LSB first.

wKeyVer: Key version in the key store, 2 bytes and LSB first.

bKeyNoCard:Key number on card, 1 byte.

ORed with **0x80** to indicate Shared application identifier (SAI).

pDivInput: Diversification input, up to 16bytes. Can be NULL.

bDivLen: Length of diversification input, 1 byte.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	004F	N byte	2 Bytes	NULL

5.9.2 AES Authentication

Perform an AES Authentication. The command should be used with AES128 keys.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0051	N Bytes

The following length and contents are before conversion to hexadecimal strings.

Data: **wOption**, **wKeyNo**, **wKeyVer**, **bKeyNoCard**, **pDivInput**, and **bDivLen in order.**

wOption: Diversification option, 2 bytes. The values are as follows

0xFFFF --No diversification.

0x0000 --Encryption based method of diversification.

0x0001 --CMAC based method of diversification.

wKeyNo: The key number in keystore to authenticate, 2 bytes and LSB first.

wKeyVer: Key version in the key store, 2 bytes and LSB first.

bKeyNoCard: Key number on card, 1 byte.

ORed with **0x80** to indicate Shared application identifier (SAI).

pDivInput: Diversification input, up to 16 bytes. Can be NULL.

bDivLen: Length of diversification input, 1 byte.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0051	N Bytes	2 Bytes	NULL

5.9.3 Authenticate EV2

Performs an Ev2 First or Non-first Authentication. This will be using the AES128 keys and will generate and verify the contents based on generic AES algorithm.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0052	N Bytes

The following length and contents are before conversion to hexadecimal strings.

Data: **bFirstAuth**, **wOption**, **wKeyNo**, **wKeyVer**, **bKeyNoCard**, **bDivLen**,**pDivInput**, **bLenPcdCapsIn** and **bPcdCapsIn** in order.

bFirstAuth: 1 byte,One of the below options

0x00 --Following (NonFirst) Authentication;

0x01 --First Authentication.

wOption: Diversification option, 2 bytes. The values are as follows:

0xFFFF --No diversification.

0x0000 --Encryption based method ofdiversification.

0x0001 --CMAC based method of diversification.

wKeyNo: The key number in keystore to authenticate, 2 bytes and LSB first.

wKeyVer: Key version in the key store, 2 bytes and LSB first.

bDivLen: Length of diversification input, 1 byte.

pDivInput: Diversification input, up to 16 bytes. Can be NULL.

bLenPcdCapsIn: Length of PcdCapsIn, 1 byte.Always 0 for following authentication.

pPcdCapsIn: PCD Capabilities. Up to 6 bytes.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0052	N Bytes	2 Bytes	24 Bytes

Card Data: The following length and contents are before conversion to hexadecimal strings.

Result data is bPcdCapsOut, bPdCapsOut.

bPcdCapsOut: PCD Capabilities, 6 bytes.

bPdCapsOut: PD Capabilities, 6 bytes.

5.9.4 Change Key Settings

Changes the master key settings on PICC and application level.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0054	NULL

Data: data is New key settings to be updated**.**

If AID = 0 selected, its PICCKeySettiongs, else its AppKeySettings.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0054	NULL	2 Bytes	4/6 Bytes

Card Data: data is the KeySettings, can be 4 or 6 bytes.

If AID = 0 selected, its PICCKeySettings, else its AppKeySettings.

5.9.5 Get Key Settings

Gets information on the PICC and application master key settings.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0054	NULL

Data: data is New key settings to be updated**.**

If AID = 0 selected, its PICCKeySettiongs, else its AppKeySettings.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0054	NULL	2 Bytes	4/6 Bytes

Card Data: data is the KeySettings, Can be 4 or 6 bytes.

If AID = 0 selected, its PICCKeySettings, else its AppKeySettings.

5.9.6 Change Key

Changes any key on the PICC

Host → Scanner

CMD	Function Code	Data
NFCCMD	0055	N Bytes

The following length and contents are before conversion to hexadecimal strings

Data: is wOption, wOldKeyNo, wOldKeyVer, wNewKeyNo, wNewKeyVer, bKeyNoCard, pDivInput, bDivLen in order.

wOption:Diversification option, 2 bytes. The values are one of the below options.

0xFFFF

0x0002 | 0x0020

0x0004 | 0x0020

0x0002 | 0x0008

0x0004 | 0x0010

0x0002 | 0x0004

0x0002 | 0x0004 | 0x0020

0x0002 | 0x0004 | 0x0008 | 0x0010

The value description:

0xFFFF--No diversification.

0x0002--Diversification of new key required (bit 1).

0x0004--Old key was diversified (bit 2).

0x0008--New key diversification using one rnd (bit 3).

Default is two rounds.

0x0010--Old key diversification using one rnd (bit 4).

Default is two rounds.

0x0020--Key diversification method based on CMAC (bit 5).

Default is Encryption method

wOldKeyNo: Old key number in keystore, 2 bytes and LSB first.

wOldKeyVer: Old key version in keystore, 2 bytes and LSB first.

wNewKeyNo: New key number in keystore,2 bytes and LSB first.

wNewKeyVer: New key version in keystore,2 bytes and LSB first.

bKeyNoCard:Key number of the key to be changed, 1 byte.

ORed with **0x80**to indicate Shared application identifier (SAI).

pDivInput: Diversification input.Up to 16bytes,can be NULL.

bDivLen: Length of **pDivInput**, 1 byte.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0055	N Bytes	2 Bytes	NULL

5.9.7 Change Key EV2

Changes any key present in keyset on the PICC. The key on the PICC is changed to the new key. The key type of the application keys cannot be changed. The key type of only the PICC master key can be changed. The keys changeable are PICCDAMAuthKey, PICCDAMMACKey, PICCDAMEncKey, VCConfigurationKey, SelectVCKey, VCProximityKey, VCPollingEncKey, VCPollingMACKey.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0056	N Bytes

The following length and contents are before conversion to hexadecimal strings

Data: **wOption, wOldKeyNo, wOldKeyVer, wNewKeyNo, wNewKeyVer, bKeySetNo, bKeyNoCard, pDivInput, bDivLen** in order.

wOption:Diversification option, 2 bytes. Same as **5.8.6 Change Key.**

wOldKeyNo: Old key number in keystore, 2 bytes and LSB first.

wOldKeyVer: Old key version in keystore, 2 bytes and LSB first.

wNewKeyNo: New key number in keystore,2 bytes and LSB first.

wNewKeyVer: New key version in keystore, 2 bytes and LSB first.

bKeySetNo: Key set number within targeted application, 1 byte.

bKeyNoCard:Key number of the key to be changed, 1 byte.

ORed with **0x80** to indicate Shared application identifier (SAI).

pDivInput: Diversification input.Up to 16bytes,can be NULL.

pDivLen: Length of **pDivInput**, 1 byte.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0056	N Bytes	2 Bytes	NULL

5.9.8 Get Key Version

Read out the current key version of any key stored on the PICC.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0057	4 Bytes

The following length and contents are before conversion to hexadecimal strings

Data: bKeyNo, bKeySetNo.

bKeyNo: Key number of the targeted key, 1 byte.ORed with one of the below options

0x80 --Secondaryary application indicator (SAI).

0x00 --**KeySetNo**not available in the command.

0x70 --**KeySetNo**available in the command.

bKeySetNo: Key set number, 1 byte. Optional as it is passed only when bit6 of **bKeyNo** is set. ORed with one of the below options

0x00 --Key version retrieval from specific key set.

0x80 --Key set versions retrieva.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0057	4 Bytes	2 Bytes	N Bytes

Card Data: data is Key set versions of the selected application ordered by ascending key set number, i.e. starting with the AKS.

5.9.9 Create Application

Create new applications on the PICC.

Host → Scanner

CMD	Function Code	Data
NFCCMD	005B	N Bytes

The following length and contents are before conversion to hexadecimal strings

Data: bOption, pAid, bKeySettings 1, bKeySettings 2, bKey Settings 3,

pKeySetValues,**pISOFileId**, **pISODFName**, **bISODFNameLen**.

bOption:Option to represent the present of ISO information, 1 byte.

0x01 meaning **pISOFileId**is supplied.

0x02 meaning **pISODFName**is present.

0x03 meaning both **pISOFileId**and **pISODFName**are present.

0x00 meaning both not present.

pAid: Application Identifier of the application to be created, 3 bytes. The value ranges from 0x000001 to 0xFFFFFF, 0x000000 is reserved for the PICC level.

bKeySettings1: Application Key settings, 1 byte.

bKeySettings2: Several other key settings, 1 byte.

bKeySettings3: Additional optional key settings,present if **bKeySettings2**[b4]

is set, 1 byte.

pKeySetValues: The Key set values for the application, present if **bKeySettings3**[b4] is set. Should consists of the following data

Byte0 = AKS ver

Byte1 = Number of Keysets

Byte2 = MaxKeysize

Byte3 = Application KeySet Settings

pISOFileId: ISO File ID if present, 2 bytes and LSB first.

pISODFName: ISO DF Name if present, upto 16 bytes. Can be NULL.

bISODFNameLen: Size of **pISODFName**if that is present, 1 byte.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	005B	N Bytes	2 Bytes	NULL

5.9.10 Delete Application

Permanently deactivates applications on the PICC.

Host → Scanner

CMD	Function Code	Data
NFCCMD	005D	N Bytes

The following length and contents are before conversion to hexadecimal strings

Data: pAid, pDAMMAC, and bDAMMAC _ Len. In turn.

pAid: Application Identifier of the application to be deleted, 3 bytes. The value ranges from 0x000001 to 0xFFFFFF, 0x000000 is reserved for the PICC level and cannot be deleted.

pDAMMAC: The MAC calculated by the card issuer to allow delegatedapplication deletion, present if KeyID.PICCDAMAuthKey or KeyID.NXPDAMAuthKey authentication, 8 bytes.

bDAMMAC_Len: Size of **pDAMMAC** if that is present, 1 byte.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	005D	N Bytes	2 Bytes	NULL

5.9.11 Select Application

Select 1 or 2 applications or the PICC level specified by their application identifier.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0061	N Bytes

The following length and contents are before conversion to hexadecimal strings

Data: bOption, pAid, pAid2.

bOption: indicates the presence of secondary Aid, 1 byte. One of the below options

0x00 --Primary application selection.

0x01 --Secondary application selection.

pAid: The primary application identifier to be selected, 3 bytes and LSB first.

pAid2: The secondary application identifier to be selected when **bOption**= 0x01, 3bytes and LSB first.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0061	N Bytes	2 Bytes	NULL

5.9.12 Format PICC

Release the PICC user memory.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0062	NULL

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0062	NULL	2 Bytes	NULL

5.9.13 Get Version

Returns manufacturing related data of the PICC.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0063	NULL

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0063	NULL	2 Bytes	56 Bytes

5.9.14 Free Memory

Returns free memory available on the PICC.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0064	NULL

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0064	NULL	2 Bytes	6 Bytes

Card Data: The size of free memory and LSB first.

5.9.15 Set Configuration

Configures the card and pre personalizes the card with a key, defines if the UID or the random ID is sent back during communication setup and configures the ATS string.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0065	N Bytes

The following length and contents are before conversion to hexadecimal strings.

Data: bOption, bDataLen, pData in order.

bOption:Define length and content of the **pData**, 1 byte.

0x00 --Update the PICC Configuration.

0x01 --Update the Default Keys.

0x02 --Update the ATS.

0x03 --Update the SAK.

0x04 --Update the Secure Messaging.

0x05 --Update Capibility Data.

0x06 --Update VC Installation Identifier.

0x0C --Update the ATQA information.

bDataLen:The size of **pData**, 1 byte.

pData:Configuration data for the option specified. The length of bytes is specified by **bDataLen.**

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0065	N Bytes	2 Bytes	NULL

5.9.16 Get Card UID

Return the Unique ID of the PICC. The key must be authenticated before this command.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0066	4 Bytes

The following length and contents are before conversion to hexadecimal strings.

Data: bExchangeOption and bOption

bExchangeOption: Flag to indicate whether the parameter information bOption to be exchanged to PICC or not.

0x00 -- Not exchanging the Option information to PICC. (EV1/EV2)

0x01 -- Exchanging the Option information to PICC. (EV3)

bOption: Indicates whether a 4-byte NUID is returned from PICC.

0x00 -- No returned 4 bytes NUID.

0x01 -- Returned 4 bytes NUID.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0066	4 Bytes	2 Bytes	14 Bytes

5.9.17 Get File IDs

Returns the File IDentifiers of all active files within the currently selected application.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0067	NULL

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0067	NULL	2 Bytes	N Bytes

The following length and contents are before conversion to hexadecimal strings

Card Data: data is the field. Each File ID is coded in one byte and is in the range from 0x00 to 0x1F.

5.9.18 Get File Settings

Get information on the properties of a specific file.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0069	2 Bytes

Data: data is File number of the targeted file.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0069	2 Bytes	2 Bytes	N Bytes

Card data: N bytes of the file settings.

5.9.19 Change File Settings

Changes the access parameters of an existing file.

Host → Scanner

CMD	Function Code	Data
NFCCMD	006B	N Bytes

Data: The following length and contents are before conversion to hexadecimal strings

Data: bOption, bFileNo, bFileOption, pAccessRights, bAddInfoLen, pAddInfo.

bOption: Indicates the mode of communication to be used while exchanging the data to PICC, 1 byte.

One of the below mentioned options:

0x00 --Plain mode of communication

0x10 --MAC mode of communication.

0x30 --Enciphered mode of communication.

Ored with **0x80**to exchange the information available in **pAddInfo**buffer as is.

bFileNo: File number for which the setting needs to be updated, 1 byte.

ORed with **0x80**to indicate secondaryary application indicator.

bFileOption: New communication settings for the file, 1 byte.

0x00 --Plain mode of communication.

0x01--MAC mode of communication.

0x03--Enciphered mode of communication.

Ored with one of the below flags if required:

0x80 --Additional Access Rigths enabled.

0x40 –-Secure Dynamic Messaging and Mirroring support enabled

0x20 --5th Bit indicating TMC limit config.

pAccessRights: The new access right to be applied for the file, 2 bytes.

bAddInfoLen: The length of **pAddInfo**, 1 byte.

pAddInfo: Contains the following optional information.

[Additional access rights] || [SDMOption || SDM AccessRights ||

VCUIDOffset || SDMReadCtrOffset || PICCDataOffset ||

SDMACInputOffset || SDMENCOffset || SDMENCLength ||

SDMMACOffset] || [TMCLimit]

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	006B	N Bytes	2 Bytes	NULL

5.9.20 Create StdData File

Creates files for storage of plain unformatted user data within an existing application on the PICC.

Host → Scanner

CMD	Function Code	Data
NFCCMD	006C	N Bytes

Data: The following length and contents are before conversion to hexadecimal strings

Data: bOption, bFileNo, pISOFileId, bCommSett, pAccessRights, pFileSize.

bOption: Indicates ISO file ID is present or not, 1 byte.

0x00 --ISOFileId is not provided.

0x01 --ISOFileId is provided and is valid.

bFileNo:File number of the file to be created, 1 byte.

plSOFileId: When **bOption**= 0x00, No this parameter. When **bOption**= 0x01, Indicates ISO/IEC 7816-4 File ID for the file to be created, 2 byts. Excluding the following values reserved by ISO: 0x0000, 0x3F00, 0x3FFF, 0xFFFF.

bCommSett: Communication settings for the file, 1 byte.

0x00 --Plain mode of communication.

0x01 --MAC mode of communication.

0x03 --Enciphered mode of communication.

Ored with one of the below options

0x20 --MIFARE Classic contactless IC mapping support enabled.

0x40 --Secure Dynamic Messaging and Mirroring support enabled.

0x80 --Additional Access Rigths enabled.

pAccessRights: Access Rights, 2 bytes.

pFileSize: File size in bytes for the file to be created, 3 bytes and LSB first.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	006C	N Bytes	2 Bytes	NULL

5.9.21 Create Backup Data File

Creates files for the storage of plain unformatted user data within an existing application on the PICC, additionally supporting the feature of an integrated backup mechanism.

Host → Scanner

CMD	Function Code	Data
NFCCMD	006D	N Bytes

The following length and contents are before conversion to hexadecimal strings.

Data: bOption, bFileNo, plSOFileId, bCommSett, pAccessRights, pFileSize.

bOption: Indicates ISO file ID is present or not, 1 byte.

0x01 --ISOFileId is provided and is valid.

bFileNo:File number of the file to be created, 1 byte.

plSOFileId: When **bOption**= 0x00, No this parameter. When **bOption**= 0x01, Indicates ISO/IEC 7816-4 File ID for the file to be created, 2 byts. Excluding the following values reserved by ISO: 0x0000, 0x3F00, 0x3FFF, 0xFFFF.

bCommSett: Communication settings for the file, 1 byte.

0x00 --Plain mode of communication.

0x01 --MAC mode of communication.

0x03 --Enciphered mode of communication.

Ored with one of the below options

0x20 --MIFARE Classic contactless IC mapping support enabled.

0x40 --Secure Dynamic Messaging and Mirroring support enabled.

0x80 --Additional Access Rigths enabled.

pAccessRights: Access Rights, 2 bytes.

pFileSize: File size in bytes for the file to be created, 3 bytes and LSB first.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	006D	N Bytes	2 Bytes	NULL

5.9.22 Create Value File

Creates files for the storage and manipulation of 32bit signed integer values within an existing application on the PICC.

Host->Scanner

CMD	Function Code	Data
NFCCMD	006E	34 Bytes

The following length and contents are before conversion to hexadecimal strings.

Data: bFileNo, bCommSett, pAccessRights, pLowerLmit, pUpperLmit, pValue, bLimitedCredit.

bFileNo:File number of the file to be created, 1 byte.

ORed with **0x80** to indicate secondaryary application indicator.

bCommSett:Communication settings for the file, 1 byte.

0x00 --Plain mode of communication.

0x01 --MAC mode of communication.

0x03 --Enciphered mode of communication.

Ored with one of the below options

0x20 --MIFARE Classic contactless IC mapping support enabled.

0x80 --Additional Access Rigths enabled.

pAccessRights: Access Rights, 2 bytes.

PValue: Initial value, 4 bytes and LSB first.

pUpperLimit: Upper limit value, 4 bytes and LSB first.

pLowerLmit: Lower limit value, 4 bytes and LSB first.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	006E	N Bytes	2 Bytes	NULL

5.9.23 Create Linear Record File

Creates files for multiple storage of structural similar data, for example for loyalty programs within an existing application. Once the file is filled, further writing is not possible unless it is cleared.

Host → Scanner

CMD	Function Code	Data
NFCCMD	006F	N Bytes

The following length and contents are before conversion to hexadecimal strings.

Data: bOption, bFileNo, plSOFileId, bCommSett, pAccessRights, pRecordSize, pMaxNoOfRec.

bOption: Indicates ISO file ID is present or not, 1 byte.

0x01 --ISOFileId is provided and is valid.

bFileNo:File number of the file to be created, 1 byte. ORed with **0x80**to indicate secondaryary application indicator.

plSOFileId: When **bOption**= 0x00, No this parameter. When **bOption**= 0x01, Indicates ISO/IEC 7816-4 File ID for the file to be created, 2 byts. Excluding the following values reserved by ISO: 0x0000, 0x3F00, 0x3FFF, 0xFFFF.

bCommSett: Communication settings for the file, 1 byte.

0x00 --Plain modeof communication.

0x01 --MAC mode of communication.

0x03 --Enciphered mode of communication.

Ored with the below options

0x80 --Additional Access Rigths enabled.

pAccessRights: Access Rights, 2 bytes.

pRecordSize: Size of onesingle record in bytes, 3 bytes and LSB first.Empty record not allowed.

pMaxNoOfRec:Max Number of Records, 3 bytes and LSB first.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	006F	N Bytes	2 Bytes	NULL

5.9.24 Create Cyclic Record File

Creates files for multiple storage of structural similar data, for example for logging transactions, within an existing application. Once the file is filled, the PICC automatically overwrites the oldest record with the latest written one.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0070	N Bytes

The following length and contents are before conversion to hexadecimal strings

Data: bOption, bFileNo, plSOFileId, bCommSett, pAccessRights, pRecordSize, pMaxNoOfRec.

bOption: Indicates ISO file ID is present or not, 1 byte. 0x01 --ISOFileIdis provided and is valid.

bFileNo:File number of the file to be created, 1 byte. ORed with **0x80**to indicate secondary application indicator.

plSOFileId: When **bOption**= 0x00, No this parameter. When **bOption**= 0x01, Indicates ISO/IEC 7816-4 File IDfor the file to be created, 2 byts. Excluding the following values reserved by ISO: 0x0000, 0x3F00, 0x3FFF, 0xFFFF.

bCommSett: Communication settings for the file, 1 byte.

0x00 --Plain mode of communication.

0x01 --MAC mode of communication.

0x03 --Enciphered mode of communication.

Ored with the below options

0x80 --Additional Access Rigths enabled.

pAccessRights: Access Rights, 2 bytes.

pRecordSize: Size of onesingle record in bytes, 3 bytes and LSB first. Empty record is not allowed.

pMaxNoOfRec: Max Number of Records, 3 bytes and LSB first.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0070	N Bytes	2 Bytes	NULL

5.9.25 Delete File

Permanently deactivates a file within the file directory of the currently selected application.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0072	2 Bytes

The following length and contents are before conversion to hexadecimal strings.

Data: The file number of the file to be deleted. ORed with 0x80 to indicate secondary application indicator.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0072	2 Bytes	2 Bytes	NULL

5.9.26 Read Data

Read data from standard data files or backup data files.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0073	18 Bytes

The following length and contents are before conversion to hexadecimal strings.

Data: bOption, bIns, bFileNo, pOffset, pLength.

bCommOption: Communication settings for the file, 1 byte.

0x00 --Plain mode of communication.

0x10 --MAC mode of communication.

0x30 --Enciphered mode of communication.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0073	18 Bytes	2 Bytes	N bytes

Card Data: If more data is to be read, the status **0x71** is returned, then should call This command again with **bCommOption=| 0x02**.

5.9.27 Write Data

Write data to standard data files or backup data files.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0074	N Bytes

The following length and contents are before conversion to hexadecimal strings.

Data: bOption,**bIns**,**bFileNo**, **pOffset**,**pTxDataLen**, **pTxData** in order.

bOption: Communication settings for the file, 1 byte.

0x00 --Plain mode of communication.

0x10 --MAC mode of communication.

0x30 --Enciphered mode of communication.

bIns: One of the below mentioned options, 1 byte.

0x00 --Represent the application chaining format.

0x01 --Represent the ISO14443-4 chaining format.

bFileNo: File number to be writedata, 1 byte.

ORed with **0x80** to indicate secondary application indicator.

pOffset: Starting position for the write operation, 3 bytes and LSB first.

pTxDataLen:The length of data to be written, 3 bytes and LSB first. Maximum length 512 byte.

pTxData: Data to be written, the length of bytes is specified by **pTxDataLen**.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0074	N Bytes	2 Bytes	NULL

5.9.28 Get Value

Read the currently stored value from value files.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0075	4 Bytes

The following length and contents are before conversion to hexadecimal strings.

Data: **bOption**,**bIns**, **bFileNo** in order.

bOption:Communication settings for the file, 1 byte.

0x00 --Plain mode of communication.

0x10 --MAC mode of communication.

0x30 --Enciphered mode of communication.

bFileNo: File number to be read value, 1 byte.

ORed with **0x80** to indicate secondary application indicator.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0075	4 Bytes	2 Bytes	8 Bytes

5.9.29 Credit

Increase a value stored in a Value File.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0076	12 Bytes

The following length and contents are before conversion to hexadecimal strings.

Data: **bOption**, **bFileNo**, **pValue** in order.

bOption:Communication settings for the file, 1 byte.

0x00 --Plain mode of communication.

0x10 --MAC mode of communication.

0x30 --Enciphered mode of communication.

bFileNo: The file number to which the value should be credited, 1 byte.

ORed with **0x80**to indicate secondary application indicator.

pValue: Value to be credited, 4 bytes and LSB first.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0076	12 Bytes	2 Bytes	NULL

5.9.30 Debit

Decrease a value stored in a Value File.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0077	12 Bytes

The following length and contents are before conversion to hexadecimal strings.

Data: **bOption, bFileNo, pValue** in order.

bOption: Communication settings for the file, 1 byte.

0x00 --Plain mode of communication.

0x10 --MAC mode of communication.

0x30 --Enciphered mode of communication.

bFileNo: The file number to which the value should be debited, 1 byte.

ORed with 0x80 to indicate secondary application indicator.

pValue: Value to be debited, 4 bytes and LSB first.

Scanner ->Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0077	12 Bytes	2 Bytes	NULL

5.9.31 Limited Credit

Allows a limited increase of a value stored in a Value File without having full credit permissions to the file.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0078	12 Bytes

The following length and contents are before conversion to hexadecimal strings

Data: bOption, bFileNo, pValue.

bOption:Communication settings for the file, 1 byte.

0x00 --Plain mode of communication.

0x10 --MAC mode of communication.

0x30 --Enciphered mode of communication.

bFileNo: The file number to which the value should be credited, 1 byte.ORed with **0x80** to indicate secondaryary application indicator.

pValue: Value to be credited, 4 bytes and LSB first.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0078	12 Bytes	2 Bytes	NULL

5.9.32 Commit Transaction

Validate all previous write access on Backup Data files, value files and record files within one application.

Host → Scanner

CMD	Function Code	Data
NFCCMD	007E	2 Bytes

The following length and contents are before conversion to hexadecimal strings.

Data: bOption.

bOption: One of the below options, 1 byte.

0x00 --Not exchanged to the PICC(EV2).

0x80 --Exchanged to PICC and represent no return of TMC and TMV(EV3).

0x81 --Exchanged to PICC and represent return of TMC and TMV(EV3).

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	007E	2 Bytes	2 Bytes	N Bytes

The following length and contents are before conversion to hexadecimal strings.

Card Data: Results Data are **pTMC** and **pTMV**.

pTMC: TransactionMAC Counter (TMC), when **bOption** = 0x80, 4 bytes.

pTMV: Transaction MAC Value (TMV), when **bOption** = 0x81, 8 bytes.

5.9.33 Abort Transaction

Aborts all previous write accesses on Backup Data files, value files and record files within the selected application(s). If applicable, the Transaction MAC calculation is aborted.

Host->Scanner

CMD	Function Code	Data
NFCCMD	007F	NULL

Scanner ->Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	007F	NULL	2 Bytes	NULL

5.9.34 Get Config

Perform a Get Config command.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0089	4 Bytes

The following length and contents are before conversion to hexadecimal strings

Data: wConfig.

wConfig: Configuration to read, 2 bytes. Will be one of the below values

0xA100 --Get additional info of a generic error or some length exposed by interfaces.

0xA200 --Get current status of command wrapping in ISO 7816-4 APDUs.

0xA300 --Get current status of Short Length APDU wrapping in ISO 7816-4 APDUs.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0089	4 byte	2 Bytes	N Bytes

Card Data: data is the value for the mentioned configuration.

5.9.35 Set Config

Perform a Set Config command.

Host → Scanner

CMD	Function Code	Data
NFCCMD	008A	4 Bytes

The following length and contents are before conversion to hexadecimal strings

Data: wConfig, wValue.

wConfig: Configuration to set, 2 bytes. Will be one of the below values exposed by interfaces.

0xA200 --Get current status of command wrapping in ISO 7816-4 APDUs.

0xA300 --Get current status of Short Length APDU wrapping in ISO 7816-4 APDUs.

wValue: The value for the mentioned configuration, 2 bytes and LSB first.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	008A	4 Bytes	2 Bytes	NULL

Card Data: data is the value for the mentioned configuration.

5.9.36 Reset Authentication

Reset Authentication status.

Host → Scanner

CMD	Function Code	Data
NFCCMD	008B	NULL

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	008B	NULL	2 Bytes	NULL

5.9.37 Read Sign

Performs the originality check to verify the genuineness of PICC.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0091	NULL

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0091	NULL	2 Bytes	112 Bytes

Card Data: The following length and contents are before conversion to hexadecimal strings. Data is the plain 56 bytes originality signature of the PICC.

5.9.38 Get DF Names

Returns the Application IDentifiers together with a File ID and (optionally) a DF Name of active applications with ISO/IEC 7816-4 support.

Host → Scanner

CMD	Function Code	Data
NFCCMD	005F	2 Bytes

Scanner ->Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	005F	2 Bytes	2 Bytes	N Bytes

Card Data:

The following length and contents are before conversion to hexadecimal strings.

The data is DFNames containing 3 bytes of AID + 2 bytes of ISO Fid + 1 to 16 bytes of DF Name. if more DF Names are to be returned, the status **0x71** is returned, and this command should then be called again with **bOption** = 0x02.

5.10 MIFARE Classic Series Cards

Operation process of MIFARE classic series cards:

- Command mode: request > anti-collision > select card > send command to verify key, read and write > stop
- Business mode: receive UID > send command to verify key, read and write > stop > find card

5.10.1 Password Verification

Host → Scanner

CMD	Function Code	Data
NFCCMD	000A	16 Bytes

Data: Block + Key type + Key

Block: The block number to verify, 2 bytes.

Key Type: 00 - keyA; 01 - keyB, 2 bytes.

Key: 12 bytes

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	000A	16 Bytes	2 Bytes	NULL

5.10.2Read Data

Host → Scanner

CMD	Function Code	Data
NFCCMD	0003	2 Bytes

Data: 2 bytes of block number to be read.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0003	2 Bytes	2 Bytes	32 Bytes

Card Data: read block data.

5.10.3 Write Data

Host → Scanner

CMD	Function Code	Data
NFCCMD	0008	34 Bytes

Data: Block+ Write Data.

Block: 2 bytes.

Write Data: 32 bytes.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0008	34 Bytes	2 Bytes	NULL

5.10.4 Data Block Initialization

Change data blocks to value format for increment, decrease, restore, and transfer operations.

Host → Scanner

CMD	Function Code	Data
NFCCMD	00D0	12 Bytes

Data: Block + Value + Address.

Block: 2 bytes.

Value: 8 bytes.

Address: 2 bytes (same as Block).

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00D0	12 Bytes	2 Bytes	NULL

5.10.5 Read Value

Read the value of the specified block.

Host → Scanner

CMD	Function Code	Data
NFCCMD	00D1	2 Bytes

Data: Block

Block: 2 Bytes

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00D1	2 Bytes	2 Bytes	10 Bytes

Card Data: Value + Address.

Value: 8 bytes.

Address: 2 bytes.

5.10.6 Add Value

Add value to a specified block. The added value is stored in TransferBuffer and the value in the original data block remains unchanged.

Host → Scanner

NFCCMD	00D2	10 Bytes

Data: Block + Value.

Block: 2 bytes.

Value: 8 bytes.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00D2	10 Bytes	2 Bytes	NULL

5.10.7 Subtract value

Decreases the value of a specified block. The reduced value is stored in TransferBuffer, and the value in the original data block remains unchanged.

Host → Scanner

CMD	Function Code	Data
NFCCMD	00D3	10 Bytes

The low byte of the address in the zone address is the block number to be decremented. Data is the value to be decremented, low byte first.

Data: Block + Value.

Block: 2 bytes.

Value: 8 bytes.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00D3	10 Bytes	2 Bytes	NULL

5.10.8 Transfer

Transfers a value data in the TransferBuffer to a specified block.

Host → Scanner

CMD	Function Code	Data
NFCCMD	00D4	2 Bytes

Data is the block number of the data to be received.

Data: Block.

Block: 2 bytes.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00D4	2 Bytes	2 Bytes	NULL

5.10.9 Restore

Transfers value data in the TransferBuffer to a specified block.

Host → Scanner

CMD	Function Code	Data
NFCCMD	00D5	2 Bytes

Data is the block number of the data to be transmitted.

Data: Block.

Block: 2 bytes.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00D5	2 Bytes	2 Bytes	NULL

5.11 Ultralight/C/EV1/NTAG21x

Ultralight/C/EV1/NTAG21x card operation process:

- Command mode: request > anti-collision > select card > secondary anti-collision > secondary select card > send command > stop
- Service mode: UID received > send command > stop > find card

5.11.1 Read Data

Read data of a specified block/page of Ultralight, UltralightC, UltralightEV1, NTAG213/215/216.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0092	2 Bytes

Data contains the address of the block.

Scanner → Host

CMD	Function Code	Data	Status	Card Data

NFCCMD	0092	2 Bytes	2 Bytes	32 Bytes

Card Data: is the read block data. Since the data of each block/page is 4 bytes, the card will return data of 4 consecutive blocks/pages.

For example, when block = 0, the card will return data of 4 blocks/pages: 0, 1, 2 and 3, totaling 16 * 2 bytes.

5.11.2 Write Data

Write data to designated block/page of Ultralight, UltralightC, UltralightEV1, NTAG213/215/216.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0093	10 Bytes

Data contains:

Block	Data
2 Bytes	8 Bytes

Block: The block number to be written.

Data: data to be written.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0093	10 Bytes	2 Bytes	NULL

5.11.3 Ultralight C Card Password Verification

Verify the password of Ultralight C Card.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0094	32 Bytes

Data: Key

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0094	32 Bytes	2 Bytes	NULL

5.11.4 Password Verification

Verify the password of Ultralight EV1, NTAG213/215/216 cards.

Host → Scanner

CMD	Function Code	Data

NFCCMD	0095	8 Bytes

Data: Key

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0095	8byte	2 Bytes	NULL

5.11.5 Obtaining Version Information

Get NTAG21x version information to retrieve information about the NTAG family, product version, storage size, and other product data needed to identify a specific NTAG21x.

Host->Scanner

CMD	Function Code	Data
NFCCMD	0096	NULL

Scanner ->Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0096	NULL	2 Bytes	N Bytes

5.11.6 Reading Counter Value

Used to read the current value of the NFC one-way counter for NTAG213, NTAG215, and NTAG216. This command has an argument that specifies the counter number and returns the 24-bit counter value of the corresponding counter.
If the NFC _ CNT _ PWD _ PROT bit is set to 1b, the counter is password protected and cannot be read until a valid password is authenticated.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0097	2 Bytes

Data: Counter assigned number

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0097	2 Bytes	2 Bytes	6 Bytes

5.11.7 Read Signature Information

Read the signature information of NTAG21x. Returns an IC-specific 32-byte ECC signature to verify that NXP Semiconductor is the silicon supplier. The signature is programmed in the chip production.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0098	NULL

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0098	NULL	2 Bytes	64 Bytes

Card Data: signature information.

5.12 ICODE2(ISO15693)

Operation process of ICODE2 cards:

- Command mode: count Label > select Label > send Command
- Service mode: UID > send command > find card

5.12.1 Inventory Labels

Count and query the tags within the antenna range and perform anti-collision operation.

Host → Scanner

CMD	Function Code	Data
NFCCMD	0020	6 Bytes

Data: Afi_flag + AFI+ Slot_flag.

Afi _ flag: whether to match the value of AFI, 00--no match, 01--match; 2 bytes.

AFI: The value of the AFI. If it does not match the AFI, it defaults to 00. 2 bytes.

Slot _ flag: Channel Type 0 indicates 16 channels, which can operate multiple cards within the antenna range. 1 means 1 channel, only one card can be operated. 2 bytes.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0020	6 Bytes	2 Bytes	N Bytes

Card Data: len+DSFID0+UID0…DSFIDN+UIDN.

Len: Returns the total length of the card information, 2 bytes.

DSFID0: The DSFID of card 0. 2 bytes.

UID0: UID of card 0. 16 bytes.

…

DSFIDN: DSFID 2 bytes of card N.

UIDN: UID of card N.

5.12.2 Select Label

Host → Scanner

CMD	Function Code	Data
NFCCMD	0021	16 Bytes

Data: UID of the card, obtained by the counting label process.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0021	16 Bytes	2 Bytes	NULL

5.12.3 Read Block Information

Host → Scanner

CMD	Function Code	Data
NFCCMD	0022	N Bytes

Data contains:

StartBlock	select_flag	address_flag	option_flag	UID	Block No
2 Bytes	2 Bytes	2 Bytes	2 Bytes	16 Bytes	2 Bytes

StartBlock: that star address of the data block to be read

Select _ flag:

00 -- Select the tab to execute the command based on the setting of Address _ flag.

01--Only the card in the selected state can execute this command.

Address _ flag:

00 -- The request is not in address mode, the UID is invalid, and any card executes;

01 -- The request is in address mode, and the UID matches. Only the card with a matching UID will execute the command.

Option _ flag:

00 – does not return the security status of the block;

01 -- Returns the security status of the block.

UID: The tag UID to read the data from.

Block No: The number of blocks to be read.

If select _ flag is set to 1, then address _ flag should be 0 and the UID of the card is invalid.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0022	N Bytes	2 Bytes	N Bytes

Card Data:

Len: Total length of returned data: 2 bytes

When option _ flag = 1

Block 1 security status: 2 bytes

Block 1 content: 8 bytes

…

BlockN Security Status 2 bytes

BlockN content: 8 bytes

When option _ flag = 0

Block 1 content: 8 bytes

…

BlockN content: 8 bytes

5.12.4 Write Block Data

Host → Scanner

CMD	Function Code	Data
NFCCMD	0023	N Bytes

Data contains:

StartBlock	select_flag	address_flag	UID	Block No	Data
2 Bytes	2 Bytes	2 Bytes	16 Bytes	2 Bytes (M)	M*4*2 Bytes

StartBlock: that star address of the data block to be written

Select _ flag:

00 -- Select the tab to execute the command based on the setting of Address _ flag;

01--Only the card in the selected state can execute this command.

Address _ flag:

00 -- The request is not in address mode, the UID is invalid, and any card executes;

01 -- The request is in address mode, and the UID matches. Only the card with a matching UID will execute the command.

UID: Tag UID to write data to

Block No: Number of blocks to be written and fetched M

Data: Data to be written M * 4 * 2

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0023	N Bytes	2 Bytes	NULL

5.12.5 Permanent Locking Block

Host → Scanner

CMD	Function Code	Data
NFCCMD	0024	22 Bytes

Data contains:

Block	select_flag	address_flag	UID
2 Bytes	2 Bytes	2 Bytes	16 Bytes

Block: is the block address to be locked.

Select _ flag:

00 -- Select the tab to execute the command based on the setting of Address _ flag;

01--Only the card in the selected state can execute this command.

Address _ flag:

00 -- The request is not in address mode, the UID is invalid, and any card executes;

01 -- The request is in address mode, and the UID has a match. Only the card with a matching UID will execute the command.

UID: The label UID of the block to lock

If select _ flag is set to 1, then address _ flag should be 0 and the UID of the card is invalid.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0024	22 Bytes	2 Bytes	NULL

5.12.6 Write AFI

Host → Scanner

CMD	Function Code	Data
NFCCMD	0025	22 Bytes

Data Contains

select_flag	address_flag	UID	AFI
2 Bytes	2 Bytes	16 Bytes	16 Bytes

Select _ flag:

00 -- Select the tab to execute the command based on the setting of Address _ flag.

01--Only the card in the selected state can execute this command

Address _ flag:

00 -- The request is not in address mode, the UID is invalid, and any card executes;

01 -- The request is in address mode, and the UID has a match. Only the card with a matching UID will execute the command.

UID: The label UID of the block to lock

AFI: The value of the AFI to be written

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0025	22 Bytes	2 Bytes	NULL

5.12.7 Lock AFI

Host → Scanner

CMD	Function Code	Data
NFCCMD	0026	20 Bytes

Data contains:

select_flag	address_flag	UID
2 Bytes	2 Bytes	16 Bytes

Select _ flag:

00 -- Select the tab to execute the command based on the setting of Address _ flag;

01--Only the card in the selected state can execute this command.

Address _ flag:

00 -- The request is not in address mode, the UID is invalid, and any card executes;

01 -- The request is in address mode, and the UID has a match. Only the card with a matching UID will execute the command.

UID: The label UID of the block to lock.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0026	20 Bytes	2 Bytes	NULL

5.12.8 Write DSFID

Host → Scanner

CMD	Function Code	Data
NFCCMD	0027	22 Bytes

Data contains

select_flag	address_flag	UID	DSFID
2 Bytes	2 Bytes	16 Bytes	2 Bytes

Select _ flag:

00 -- Select the tab to execute the command based on the setting of Address _ flag;

01--Only the card in the selected state can execute this command.

Address _ flag:

00 -- The request is not in address mode, the UID is invalid, and any card executes;

01 -- The request is in address mode, and the UID has a match. Only the card with a matching UID will execute the command.

UID: The label UID of the block to lock

DSFID: The value of the DSFID to be written.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0027	22 Bytes	2 Bytes	NULL

5.12.9 Lock DSFID

Host → Scanner

CMD	Function Code	Data
NFCCMD	0028	20 Bytes

Data contains:

select_flag	address_flag	UID

2 Bytes	2 Bytes	16 Bytes

Select _ flag:

00 -- Select the tab to execute the command based on the setting of Address _ flag;

01--Only the card in the selected state can execute this command.

Address _ flag:

00 -- The request is not in address mode, the UID is invalid, and any card executes;

01 -- The request is in address mode, and the UID has a match. Only the card with a matching UID will execute the command.

UID: Tag UID to lock ADSFID.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0028	20 Bytes	2 Bytes	NULL

5.12.10 Set EAS

Host → Scanner

CMD	Function Code	Data
NFCCMD	0029	4 Bytes

Data contains:

EAS	Request_flag
2 Bytes	2 Bytes

EAS:

00 EAS bit set to 0;

01 EAS bit set to 1.

Request _ flag: is the request flag, and the value is

00 -- the request can be executed by any card;

01--Only the card in the selected state is executed.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	0029	4 Bytes	2 Bytes	NULL

5.12.11 Lock EAS

Host → Scanner

CMD	Function Code	Data
NFCCMD	002A	2 Bytes

Data: Request _ flag. It is the request flag, and the value is

00 -- the request can be executed by any card;

01--Only the card in the selected state is executed.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	002A	2 Bytes	2 Bytes	NULL

5.13 NTAG 42x DNA/TT

NTAG 42x DNA/TT Card Operation Procedure:

- Command mode: request > anti-collision > select card > secondary anti-collision > secondary select card > reset > send command to verify key, read and write > stop
- Service mode: receive UID > reset > send command to verify key, read and write > stop > find card

5.13.1 Authenticate EV2

Performs a First or non-first Authentication depending upon bFirstAuth Parameter. This will be using the AES128 keys and will generate and verify the contents based on generic AES algorithm.

Host → Scanner

CMD	Function Code	Data
NFCCMD	009B	N Bytes

The following length and contents are before conversion to hexadecimal strings.

Data: bFirstAuth, **wOption**, **wKeyNo**, **wKeyVer**, **bKeyNoCard**, **bDivLen**,

pDivInput, **bLenPcdCapsIn**, **pPcdCapsIn**.

bFirstAuth: Authentication mode,1 byte.

0x00 --Non First Auth in regular EV2 auth Mode.

0x01 --First Auth in regular EV2 auth Mode.

0x02 --Non First Auth in LRP mode.

0x03 --First Auth in LRP mode.

wOption: Diversification option, 2 bytes.

0xFFFF --No diversification.

0x0000 --Encryption based method of diversification.

0x0001 --CMAC based method of diversification

wKeyNo: Key number in keystore, 2 bytes and LSB first.

wKeyVer: Key version in keystore, 2 bytes and LSB first.

bKeyNoCard: Key number on card, 1 byte.

bDivLen: Length of diversification input, 1 byte.

pDivInput: Diversification input, up to 16 bytes. Can be NULL.

bLenPcdCapsIn: Length of PcdCapsIn, 1 byte.Always 0 for following authentication.

pPcdCapsIn: PCD Capabilities. Up to 6 bytes.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	009B	N Bytes	2 Bytes	24 Bytes

Card Data:

bPcdCapsOut: PCD Capabilities, 12 bytes.

bPdCapsOut: PD Capabilities, 12 bytes.

5.13.2 Set Configuration

Configures the card and pre personalizes the card with a key, defines if the UID or the random ID is sent back during communication setup and configures ATS string.

Host → Scanner

CMD	Function Code	Data
NFCCMD	009C	N Bytes

The following length and contents are before conversion to hexadecimal strings.

Data: **bOption, bDataLen and pData** in order.

bOption: Define length and content of the **pData**, 1 byte.

0x00 --Update the PICC Configuration.

0x04 --Update the Secure Messaging.

0x05 --Update Capibility Data.

0x07 --Update Tag Tamper configuration

0x0A --Update Failed Authentication Counter Configuration.

0x0B --Update Hardware Configuration.

pData: Configuration data for the option specified. The length of bytes is specified by **bDataLen.**

bDataLen: The size of **pData**, 1 byte.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	009C	N Bytes	2 Bytes	NULL

5.13.3 Get Version

Returns manufacturing related data of the PICC.

Host → Scanner

CMD	Function Code	Data
NFCCMD	009D	NULL

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	009D	NULL	2 Bytes	N Bytes

Card Data: Get version information for.

5.13.4 Get Card ID

Return the Unique ID of the PICC. The key must be authenticated before this command.

Host → Scanner

CMD	Function Code	Data
NFCCMD	009E	NULL

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	009E	NULL	2 Bytes	14 Bytes

Card Data: UID returned by the card.

5.13.5 Change Key

Change any key on the PICC.

Host → Scanner

CMD	Function Code	Data
NFCCMD	009F	N Bytes

The following length and contents are before conversion to hexadecimal string.

Data: wOption, wOldKeyNo, wOldKeyVer, wNewKeyNo, wNewKeyVer, bKeyNoCard, pDivInput, bDivLen.

wOption: Diversification option, 2 bytes. The values are one of the below options:

0xFFFF

0x0002 | 0x0020

0x0004 | 0x0020

0x0002 | 0x0008

0x0004 | 0x0010

0x0002 | 0x0004

0x0002 | 0x0004 | 0x0020

0x0002 | 0x0004 | 0x0008 | 0x0010

The value description:

0xFFFF--No diversification.

0x0002--Diversification of new key required (bit 1).

0x0004--Old key was diversified (bit 2).

0x0008--New key diversification using one rnd (bit 3).

Default is two rounds.

0x0010--Old key diversification using one rnd (bit 4).

Default is two rounds.

0x0020--Key diversification method based on CMAC (bit 5).

Default is Encryption method.

wOldKeyNo: Old key number in keystore, 2 bytes and LSB first.

wOldKeyVer: Old key version in keystore, 2 bytes and LSB first.

wNewKeyNo: New key number in keystore,2 bytes and LSB first.

wNewKeyVer: New key version in keystore,2 bytes and LSB first.

bKeyNoCard: Key number of the key to be changed, 1 byte.

pDivInput: Diversification input. Up to 16bytes, can be NULL.

bDivLen: Length of **pDivInput**, 1 byte.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	009F	N Bytes	2 Bytes	NULL

5.13.6 Gey Key Version

Read out the current key version of any key stored on the PICC.

Host → Scanner

CMD	Function Code	Data
NFCCMD	00A0	4 Bytes

Data: bKeyNo, **bKeySetNo**.

bKeyNo: Key number on card, 2byte.

bKeySetNo: Key set number, 2 bytes. Optional as it is passed only when bit6 of bKeyNo is set.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00A0	4 Bytes	2 Bytes	N Bytes

5.13.7 Get File Settings

Get information on the properties of a specific file.

Host → Scanner

CMD	Function Code	Data
NFCCMD	00A1	2 Bytes

Data: The file number of the targeted file.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00A1	2 Bytes	2 Bytes	N Bytes

5.13.8 Get File Counters

Returns manufacturing related data of the PICC.

Host → Scanner

CMD	Function Code	Data
NFCCMD	00A2	4 Bytes

Data: **bOption**, **bFileNo.**

bOption: Indicates the mode of communication to be used while exchanging the data to PICC, 2 bytes.

00 --Plain mode of communication.

30 --Enciphered mode of communication.

bFileNo: File number for which the Counter information need to be received, 2 bytes.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00A2	4 Bytes	2 Bytes	N Bytes

5.13.9 Change File Settings

Changes the access parameters of an existing file.

Host → Scanner

CMD	Function Code	Data
NFCCMD	00A3	70 Bytes

Data: **bOption**, **bFileNo**, **bFileOption**, **pAccessRights, bSdmOptions**, **pSdmAccessRights**, **dwVCUIDOffset**, **dwSDMReadCtrOffset**, **dwPICCDataOffset**, **dwTTPermStatusOffset**, **dwSDMMACInputOffset**, **dwSDMENCOffset**, **dwSDMENCLen**, **dwSDMMACOffset**, **dwSDMReadCtrLimit** in order.

bOption: Indicates the mode of communication to be used while exchanging the data to PICC.

0x00 --Plain mode of communication.

0x10 --MAC mode of communication.

0x30 --Enciphered mode of communication.

bFileNo: File number for which the setting needs to be updated, 1 byte.

bFileOption: New communication settings for the file, 1 byte.

0x00 --Plain mode of communication.

0x01 --MAC mode of communication.

0x03 --Enciphered mode of communication.

Ored with below option

0x40 --Secure Dynamic Messaging and Mirroring is enabled.

pAccessRights: The new access right to be applied for the file, 2 bytes.

bSdmOptions: Secure Dynamic Messaging options. One of the below values to be used, 1 byte.

0x80 --Only VCUID is considred for SDM MAC calculation.

0x40 --Only RDCTR is considred for SDM MAC calculation.

0x20 --Indicates the presence of SDM Read Counter Limit.

0x10 --Indicates the presence of SDM ENC File data.

0x08 --Indicates the presence of SDM TT Status.

0x01 --Indicates the encoding as ASCII.

Must be ored with the above values.

pSdmAccessRights: The SDM access rights to be applied, 2 bytes.

dwVCUIDOffset: VCUID Offset information, 3 bytes and LSB first.

dwSDMReadCtrOffset: SDMReadCtr value, 3 bytes and LSB first.

dwPICCDataOffset: Mirror position for encrypted PICC Data, 3 bytes and LSB first.

dwTTPermStatusOffset: Tag Tamper Permanent Status Offset value, 3 bytes and LSB first.

dwSDMMACInputOffset: Offset in the file where the SDM MAC computation starts, 3 bytes and LSB first.

dwSDMENCOffset: SDMENCFileData mirror position, 3 bytes and LSB first.

dwSDMENCLen: Length of the SDMENCFileData, 3 bytes and LSB first.

dwSDMMACOffset: SDMMAC mirror position, 3 bytes and LSB first.

dwSDMReadCtrLimit: SDM Read Counter Limit value, 3 bytes and LSB first.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00A3	70 Bytes	2 Bytes	NULL

5.13.10 Read Data

Read data from standard data files or backup data files.

Host → Scanner

CMD	Function Code	Data
NFCCMD	00A4	18 Bytes

The following length and contents are before conversion to hexadecimal strings

Data: bOption, bIns, bFileNo, pOffset, pLength in order.

bCommOption:Communication settings for the file, 1 byte.

0x00 --Plain mode of communication.

0x10 --MAC mode of communication.

0x30 --Enciphered mode of communication.

bIns: One of the below mentioned options, 1 byte.

0x00 --Represent the application chaining format.

0x01 --Represent the ISO14443-4 chaining format.

bFileNo: File number to be read data, 1 byte.

pOffset: Starting position for the read operation, 3 bytes and LSB first.

If it is 0x000000, Read the entire data file, starting from the position specified in the offset value.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00A4	18 Bytes	2 Bytes	N Bytes

Card Data: **If more data is to be read, the status 0x71 is returned, then should call This command again with bCommOption=| 0x02.**

5.13.11 Write Data

Write data to standard data files or backup data files.

Host → Scanner

CMD	Function Code	Data
NFCCMD	00A5	N Bytes

The following length and contents are before conversion to hexadecimal strings

Data: bOption, bIns, bFileNo, pOffset, pTxData, pTxDataLen.

bOption:Communication settings for the file, 1 byte.

0x00 --Plain mode of communication.

0x10 --MAC mode of communication.

0x30 --Enciphered mode of communication.

bIns: One of the below mentioned options, 1 byte.

0x00 --Represent the application chaining format.

0x01 --Represent the ISO14443-4 chaining format.

bFileNo: File number to be writedata, 1 byte.

pOffset: Starting position for the writeoperation, 3 bytes and LSB first.

pTxData: Data to be written, the length of bytes is specified by **pTxDataLen**.

pTxDataLen:The length of data to be written, 3 bytes and LSB first.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00A5	N Bytes	2 Bytes	NULL

5.13.12 ISO Select File

Host → Scanner

CMD	Function Code	Data
NFCCMD	00A6	N Bytes

The following length and contents are before conversion to hexadecimal strings

Data: bOption, bSelector, pFid, pDFName, bDFNameLen, bExtendedLenApdu.

bOption: Indicates whether to return File Control Information (FCI), 1 byte.

0x0C --No return FCI.

bSelector:Selection Control, 1 byte.

0x00 --Select MF, DF or EF, by file identifier.

0x01 --Select child DF.

0x02 --Select EF under the current DF, by file identifier.

0x03 --Select parent DF of the current DF.

0x04 --Select by DF name.

pFid: The ISO File number to be selected, 2 bytes.

Valid only when **bSelector**= 0x00 or 0x02.

pDFName: The ISO DFNameto be selected, up to 16 bytes.

Valid only when **bSelector**= 0x04.

bDFNameLen: The length of DFName, 1 byte. Valid only when **bOption**= 0x04.

bExtendedLenApdu: Flag for Extended Length APDU, 1 byte.

0x01 for Extended Length APDUs.

0x00 or any other value for Short APDUs.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00A6	N Bytes	2 Bytes	N Bytes

Card Data: data is the returned FCI. Valid only when **bOption**= 0x00.

5.13.13 Read Sign

Performs the originality check to verify the genuineness of chip.

Host → Scanner

CMD	Function Code	Data
NFCCMD	00A9	NULL

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00A9	NULL	2 Bytes	112 Bytes

Card Data: Data is the plain 56 bytes originality signature of the PICC.

5.13.14 Get TT Status

Returns Tag Tamper Status data of the PICC.

Host → Scanner

CMD	Function Code	Data
NFCCMD	00AA	NULL

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00AA	NULL	2 Bytes	4

Card Data: data is theTag Tamper Protection status

TTPermStatus	TTCurrStatus
2 Bytes	2 Bytes

5.14 FeliCa

Operation process of FeliCa card:

- Command mode: activate card > send command
- Service mode: UID > send command received

5.15 Contactless CPU Card (ISO14443 TypeB)

Before executing a TypeB card, use the **5.2 Select Protocol.** The command selects the current protocol of the reader as TypeB.

5.15.1 Activate Card

Host → Scanner

CMD	Function Code	Data
NFCCMD	00AB	NULL

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00AB	NULL	2 Bytes	N Bytes

The following length and contents are before conversion to hexadecimal strings.

Card Data: Bytes 2-5 are the serial number of the card.

5.15.2 Reset

Host → Scanner

CMD	Function Code	Data
NFCCMD	00AC	14 Bytes

The following length and contents are before conversion to hexadecimal strings

Data: UID, CID, BrTx, BrRx.

UID:5.14.1 Activate CardUID returned, 4 bytes.

CID: The value is 0-14, which can be used only when the card supports CID, 1 byte. Default 0.

BrTx: default 0 x00, 1 byte.

BrTx: default 0 x00, 1byte.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00AC	14 Bytes	2 Bytes	N Bytes

5.15.3 Send Command

APDU commands.

Host → Scanner

CMD	Function Code	Data
NFCCMD	00AD	N Bytes

Data: UID, CID, BrTx, BrRx.

NAD (default 00)

CID (default 00)

PCB (default 00)

LEN: (COS command length)

Data [4] -DATA [LEN + 4] COS command

For example, ISO7816-4 gets the random number

CLA 00

INS 84

P1 00

P2 00

Le 08

Send command: NFCCMD00190000000050084000008

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00AD	N Bytes	2 Bytes	N Bytes

Card Data: Return data by APDU.

5.15.4 Stop Card

Host → Scanner

CMD	Function Code	Data
NFCCMD	00AE	8 Bytes

The following length and contents are before conversion to hexadecimal strings.

Data: **5.14.1 Activate Card** UID returned.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00AE	8 Bytes	2 Bytes	NULL

5.15.5 Obtain Chinese ID Card UID

Host → Scanner

CMD	Function Code	Data
NFCCMD	00AF	NULL

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00AF	NULL	2 Bytes	N Bytes

5.16 MIFARE Plus

MIFARE Plus Card Operation Procedure:

- Command mode:
  - 4-byte serial number card: Request > conflict prevention > select card.
  - 7-byte serial number card: Request > anti-conflict > select card > secondary anti-conflict > secondary card selection.
- Business mode: Receive UID > send command to verify key, read/write > stop > seek card

5.16.1 Write Perso

This command is used to change the data and AES keys from the initial delivery configuration to a customer specific value.

Host → Scanner

CMD	Function Code	Data
NFCCMD	00B0	N Bytes

The following length and contents are before conversion to hexadecimal strings.

Data: bLayer4Comm, wBlockNr, bNumBlocks, pData.

bLayer4Comm: ISO14443 protocol to be used, 1 byte.

0x00 -- Use Iso14443 Layer 3 protocol.

0x01 -- Use Iso14443 Layer 4 protocol.

wBlockNr: Block number to be personalized., 2 bytes and LSB first.

bNumBlocks: Number of blocks to be personalized, 1 byte.

pData: The value for the block mentioned in wBlockNr. If bNumBlocks = 1, the length should be 16 bytes. If bNumBlocks > 1, the length should be (bNumBlocks * 16) bytes.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00B0	NULL	2 Bytes	NULL

5.16.2 Commit Perso

This command commits the written data during WritePerso command and switches the SecurityLevel to 1 or 3 based on the option provided.

If the Option parameter is 0, only the command code will be exchanges to PICC. This is to maintain the backward compatibility with MIFARE Plus PICC.

Host → Scanner

CMD	Function Code	Data

NFCCMD	00B1	2 Bytes

The following length and contents are before conversion to hexadecimal strings.

Data: bOption, bLayer4Comm.

bOption: Option to be used for Security Level switching, 1 byte.

0x00 -- Maintain the backward compatibility with MIFARE Plus PICC.

0x01 -- Switch the Security Level to 1.

0x03 -- Switch the Security Level to 3.

bLayer4Comm: ISO14443 protocol to be used, 1 byte.

0x00 -- Use Iso14443 Layer 3 protocol.

0x01 -- Use Iso14443 Layer 4 protocol.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00B1	NULL	2 Bytes	NULL

5.16.3 Authenticate MFC

Perform MIFARE Authenticate command in Security Level 1 with MIFARE CLASSIC PICC.

Host → Scanner

CMD	Function Code	Data
NFCCMD	00B2	N Bytes

The following length and contents are before conversion to hexadecimal strings.

Data: bBlockNo, bKeyType, wKeyNumber, wKeyVer, bUidLen, pUid.

bBlockNo: The block number to be used for authentication, 1 byte.

bKeyType: Authentication key type to be used, 1 byte.

0x0A -- MIFARE(R) Key A.

0x0B -- MIFARE(R) Key B.

wKeyNumber: Key number to used from key store, 2 byte and LSB first.

wKeyVer: Key version to used from key store, 2 byte and LSB first.

bUidLen: the Length of the pUid, 1 byte.

pUid: UID of the PICC received during anti-collision sequence, the length of byte is specified by bUidLen.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00B2	NULL	2 Bytes	NULL

5.16.4 Authenticate SL1

Performs a MIFARE Plus Authentication for Security Level 1. This command performs basic Authenticate First / Non-First command execution and performs the Authenticate Continue command internally.

Host → Scanner

CMD	Function Code	Data
NFCCMD	00B4	N Bytes

The following length and contents are before conversion to hexadecimal strings.

Data: bLayer4Comm, bFirstAuth, wBlockNr, wKeyNumber, wKeyVer,

bLenDivInput, pDivInput, bLenPcdCap2, pPcdCap2In.

bLayer4Comm: ISO14443 protocol to be used, 1 byte.

0x00 -- Use Iso14443 Layer 3 protocol.

0x01 -- Use Iso14443 Layer 4 protocol.

bFirstAuth: Type of authentication to be performed, 1 byte.

0x01 -- Indicate the authenticate type as first.

0x00 -- Indicate the authenticate type as non-first or following.

wBlockNr: The block number to be used for authentication, 2 bytes and LSB first.

wKeyNumber: Key number to used from key store, 2 byte and LSB first.

wKeyVer: Key version to used from key store, 2 byte and LSB first.

bLenDivInput: The length of diversification input used to diversify the key, 1 byte. If 0, no diversification is performed.

pDivInput: Diversification Input used to diversify the key, the length of byte is specified by bLenDivInput.

bLenPcdCap2: Length of the pPcdCap2In, 1 byte.

pPcdCap2In: The Input PCD Capabilities, the length of byte is specified by bLenPcdCap2.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00B4	12 Bytes	2 Bytes	NULL

Card data: **pPcdCap2Out, pPdCap2.**

pPcdCap2Out: The Output PCD capabilities, 6 bytes.

pPdCap2: The Output PD capabilities, 6 bytes.

5.16.5 Authenticate SL3

Performs a MIFARE Plus Authentication for Security Level 3. This command performs basic Authenticate First / Non-First command execution and performs the Authenticate Continue command internally.

Host → Scanner

CMD	Function Code	Data
NFCCMD	00B5	N Bytes

The following length and contents are before conversion to hexadecimal strings.

Data: bLayer4Comm, bFirstAuth, wBlockNr, wKeyNumber, wKeyVer,

bLenDivInput, pDivInput, bLenPcdCap2, pPcdCap2In.

bLayer4Comm: ISO14443 protocol to be used, 1 byte.

0x00 -- Use Iso14443 Layer 3 protocol.

0x01 -- Use Iso14443 Layer 4 protocol.

bFirstAuth: Type of authentication to be performed, 1 byte.

0x01 -- Indicate the authenticate type as first.

0x00 -- Indicate the authenticate type as non-first or following.

wBlockNr: The block number to be used for authentication, 2 bytes and LSB first.

wKeyNumber: Key number to used from key store, 2 byte and LSB first.

wKeyVer: Key version to used from key store, 2 byte and LSB first.

bLenDivInput: The length of diversification input used to diversify the key,

1 byte. If 0, no diversification is performed.

pDivInput: Diversification Input used to diversify the key, the length of byte is specified by bLenDivInput.

bLenPcdCap2: Length of the pPcdCap2In, 1 byte.

pPcdCap2In: The Input PCD Capabilities, the length of byte is specified by bLenPcdCap2.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00B5	12 Bytes	2 Bytes	NULL

Card data: **pPcdCap2Out, pPdCap2.**

pPcdCap2Out: The Output PCD capabilities, 6 bytes.

pPdCap2: The Output PD capabilities, 6 bytes.

5.16.6 Write Data

This command writes a 16 bytes data to the PICC. The parameter bEncrypted, bWriteMaced are valid only for MFP authenticated state and not for MFC authenticate state.

Host → Scanner

CMD	Function Code	Data
NFCCMD	00B8	N Bytes

The following length and contents are before conversion to hexadecimal strings.

Data: bEncrypted, bWriteMaced, wBlockNr, bNumBlocks, pBlocksData.

bEncrypted: Type of communication to be used. Based on this flag the command

code will be updated, 1 byte.

0x00 -- Indicate the communication between PCD and PICC is plain.

0x01 -- Indicate the communication between PCD and PICC is encrypted.

bWriteMaced: Indicate whether the response should be maced. Based on this flag

the command code will be updated, 1 byte.

0x00 -- Indicate the communication is not maced for PICC to PCD transfer.

0x01 -- Indicate the communication is maced for PICC to PCD transfer.

wBlockNr: The block number to which the data should be written, 2 bytes and LSB first.

bNumBlocks: Number of blocks to write, 1 byte.

pBlocksData: The data to be written. The length of bytes is bNumBlocks * 16.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00B8	(N Bytes)	2 Bytes	NULL

Card data: **pTMC, pTMV.**

pTMC: Only available is the block is a TMProtected block, 4 bytes of Transaction. MAC counter information will be returned.

pTMV: Only available is the block is a TMProtected block. 8 bytes of Transaction. MAC value will be returned.

5.16.7 Read Data

Performs a Read/Read MACed command.

The parameter **bEncrypted**, **bReadMaced** and **bMacOnCmd** are valid only for MFP. Authenticated state and not for MFC authenticate state.

Host → Scanner

CMD	Function Code	Data
NFCCMD	00B9	12 Bytes

Data: The following length and contents are before conversion to hexadecimal strings.

bEncrypted, bReadMaced, bMacOnCmd, wBlockNr, bNumBlocks.

bEncrypted: Type of communication to be used. Based on this flag the command code will be updated, 1 byte.

0x00 -- Indicate the communication between PCD and PICC is plain.

0x01 -- Indicate the communication between PCD and PICC is encrypted.

bReadMaced: Indicate whether the response should be maced. Based on this flag the command code will be updated, 1 byte.

0x00 -- Indicate the communication is not maced for PICC to PCD transfer.

0x01 -- Indicate the communication is not maced for PICC to PCD transfer.

bMacOnCmd: Indicate whether the command should be maced. Based on this flag the command code will be updated, 1 byte.

0x00 -- Indicate the communication is not maced for PCD to PICC transfer.

0x01 -- Indicate the communication is maced for PCD to PICC transfer.

wBlockNr: The block number to which the data should be read, 2 bytes and LSB first.

bNumBlocks: Number of blocks to be read, 1 byte.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00B9	12 Bytes	2 Bytes	N Bytes

Data is the data returned from PICC. The length is (bNumBlocks * 16) + 8,

The last 8 bytes is the MAC received form PICC.

Host → Scanner

CMD	Function Code	Data
NFCCMD	00B8	N Bytes

Data: The following length and contents are before conversion to hexadecimal strings.

bEncrypted, bWriteMaced, wBlockNr, bNumBlocks, pBlocksData.

bEncrypted: Type of communication to be used. Based on this flag the command

code will be updated, 1 byte.

0x00 -- Indicate the communication between PCD and PICC is plain.

0x01 -- Indicate the communication between PCD and PICC is encrypted.

bWriteMaced: Indicate whether the response should be maced. Based on this flag the command code will be updated, 1 byte.

0x00 -- Indicate the communication is not maced for PICC to PCD transfer.

0x01 -- Indicate the communication is maced for PICC to PCD transfer.

wBlockNr: The block number to which the data should be written, 2 bytes and LSB first.

bNumBlocks: Number of blocks to write, 1 byte.

pBlocksData: The data to be written. The length of bytes is bNumBlocks * 16.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00B8	(N Bytes)	2 Bytes	NULL

Result data: **pTMC, pTMV.**

pTMC: Only available is the block is a TMProtected block, 4 bytes of Transaction. MAC counter information will be returned.

pTMV: Only available is the block is a TMProtected block. 8 bytes of Transaction. MAC value will be returned.

5.16.8 Read Value

Performs a Read / Read MACed Value command. The parameter Encrypted, ReadMaced and MacOnCmd are valid only for MFP authenticated state and not for MFC authenticate state.

Host → Scanner

CMD	Function Code	Data
NFCCMD	00BB	5 Bytes

The following length and contents are before conversion to hexadecimal strings.

Data: bEncrypted, bReadMaced, bMacOnCmd, wBlockNr.

bEncrypted: Type of communication to be used. Based on this flag the command code will be updated, 1 byte.

0x00 -- Indicate the communication between PCD and PICC is plain.

0x01 -- Indicate the communication between PCD and PICC is encrypted. **bReadMaced:**

bReadMaced: Indicate whether the response should be maced. Based on this flag the command code will be updated, 1 byte.

0x00 -- Indicate the communication is not maced for PICC to PCD transfer.

0x01 -- Indicate the communication is maced for PICC to PCD transfer.

bMacOnCmd: Indicate whether the command should be maced. Based on this flag the command code will be updated, 1 byte.

0x00 -- Indicate the communication is not maced for PCD to PICC transfer.

0x01 -- Indicate the communication is maced for PCD to PICC transfer.

wBlockNr: The PICC block number to which the value should be read, 2 bytes and LSB first.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00BB	5 Bytes	2 Bytes	NULL

Card data: **pTMC, pTMV.**

dwValue: The value read from the specified block number, 4 bytes and LSB first.

bAddr: The address from the read value information, 1 byte.

5.16.9 Increment

Performs an Increment / Increment MACed command. The parameter IncrementMaced is valid only for MFP authenticated state and not for MFC authenticate state.

Host → Scanner

CMD	Function Code	Data
NFCCMD	00BC	7 Bytes

The following length and contents are before conversion to hexadecimal strings.

Data: bIncrementMaced, wBlockNr, dwValue.

bIncrementMaced: Indicate whether the response should be maced. Based on this flag the command code will be updated, 1 byte.

0x00 -- Indicate the communication is not maced for PICC to PCD transfer.

0x01 -- Indicate the communication is maced for PICC to PCD transfer.

wBlockNr: PICC block number to be used for incrementing the value, 2 bytes andLSB first.

dwValue: The value to be incremented, 4 bytes and LSB first.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00BC	NULL	2 Bytes	NULL

5.16.10 Decrement

Performs a Decrement / Decrement MACed command. The parameter DecrementMaced is valid only for MFP authenticated state and not for
MFC authenticate state.

Host → Scanner

CMD	Function Code	Data
NFCCMD	00BD	7 Bytes

The following length and contents are before conversion to hexadecimal strings.

Data: bDecrementMaced, wBlockNr, dwValue.

bIncrementMaced: Indicate whether the response should be maced. Based on this flag the command code will be updated, 1 byte.

0x00 -- Indicate the communication is not maced for PICC to PCD transfer.

0x01 -- Indicate the communication is maced for PICC to PCD transfer.

wBlockNr: PICC block number to be used for incrementing the value, 2 bytes andLSB first.
dwValue: The value to be incremented, 4 bytes and LSB first.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00BD	NULL	2 Bytes	NULL

5.16.11 Increment Transfer

Performs an Increment Transfer/Increment Transfer MACed command.
The parameter IncrementTransferMaced is valid only for MFP authenticated state and not for MFC authenticate state.

Host → Scanner

CMD	Function Code	Data
NFCCMD	00BE	9 Bytes

The following length and contents are before conversion to hexadecimal strings.
Data: bIncrementTransferMaced, wSourceBlockNr, wDestinationBlockNr,
dwValue.
bIncrementTransferMaced: Indicate whether the response should be maced. Based on this flag the command code will be updated, 1 byte.

0x00 -- Indicate the communication is not maced for PICC to PCD transfer.

0x01 -- Indicate the communication is maced for PICC to PCD transfer.

wSourceBlockNr: PICC block number to be used for incrementing the value, 2 bytes and LSB first.
wDestinationBlockNr: PICC block number to be used for transferring the value, 2 bytes and LSB first.
dwValue: The value to be incremented and transferred, 4 bytes and LSB first.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00BE	12 Bytes	2 Bytes	NULL

Card data: **pTMC, pTMV.**
pTMC: Only available is the block is a TMProtected block.Transaction MAC counter information, 4 bytes.
pTMV: Only available is the block is a TMProtected block.Transaction MAC value, 8 bytes.

5.16.12 Decrement Transfer

Performs a Decrement Transfer / Decrement Transfer MACed command. The parameter DecrementTransferMaced is valid only for MFP authenticated state and
not for MFC authenticate state.

Host → Scanner

CMD	Function Code	Data
NFCCMD	00BF	9 Bytes

The following length and contents are before conversion to hexadecimal strings.

Data: bDecrementTransferMaced, wSourceBlockNr, wDestinationBlockNr, dwValue.

bDecrementTransferMaced: Indicate whether the response should be maced. Based on this flag the command code will be updated, 1 byte.

0x00 -- Indicate the communication is not maced for PICC to PCD transfer.

0x01 -- Indicate the communication is maced for PICC to PCD transfer.

wSourceBlockNr: PICC block number to be used for decrementing the value, 2 bytes and LSB first.

wDestinationBlockNr: PICC block number to be used for transferring the value, 2 bytes and LSB first.

dwValue: The value to be decremented and transferred, 4 bytes and LSB first.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00BF	12 Bytes	2 Bytes	NULL

Card result: **pTMC, pTMV.**

pTMC: Only available is the block is a TMProtected block.Transaction MAC counter information, 4 bytes.

pTMV: Only available is the block is a TMProtected block.Transaction MAC value, 8 bytes.

5.16.13 Transfer

Performs a Transfer/Transfer MACed command. The parameter TransferMaced is valid only for MFP authenticated state and not for
MFC authenticate state.

Host->Scanner

CMD	Function Code	Data
NFCCMD	00C0	3 Bytes

The following length and contents are before conversion to hexadecimal strings.

Data: bTransferMaced, wBlockNr.

bTransferMaced: Indicate whether the response should be maced. Based on this flag the command code will be updated, 1 byte.

0x00 -- Indicate the communication is not maced for PICC to PCD transfer.

0x01 -- Indicate the communication is maced for PICC to PCD transfer.

wBlockNr: PICC block number to be used for transferring the value, 2 bytes and LSB first.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00C0	12 Bytes	2 Bytes	NULL

Card data: **pTMC, pTMV.**

pTMC: Only available is the block is a TMProtected block.Transaction MAC counter information, 4 bytes.

pTMV: Only available is the block is a TMProtected block.Transaction MAC value, 8 bytes.

5.16.14 Restore

Performs a Restore / Restore MACed command. The parameter RestoreMaced is valid only for MFP authenticated state and not for MFC authenticate state.

Host → Scanner

CMD	Function Code	Data
NFCCMD	00C1	3 Bytes

The following length and contents are before conversion to hexadecimal strings.

Data: bRestoreMaced, wBlockNr.

bRestoreMaced: Indicate whether the response should be maced. Based on this flag the command code will be updated, 1 byte.

0x00 -- Indicate the communication is not maced for PICC to PCD transfer.

0x01 -- Indicate the communication is maced for PICC to PCD transfer.

wBlockNr: PICC block number to be used for restoring the value, 2 bytes and LSB first.

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00C1	NULL	2 Bytes	NULL

5.16.15 Get Version

Returns manufacturing related data of the PICC.

Host → Scanner

CMD	Function Code	Data
NFCCMD	00C2	NULL

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00C2	N bytes	2 Bytes	NULL

Card data: data is the version information of the PICC.

If UID is 4 bytes, the buffer will have 27 bytes of version information.

If UID is 7 bytes, the buffer will have 28 bytes of version information.

If UID is 10 bytes, the buffer will have 33 bytes of version information.

5.16.16 Read Signature

Read originality Signature from the PICC.

Host → Scanner

CMD	Function Code	Data
NFCCMD	00C3	2 Bytes

Data: The following length and contents are before conversion to hexadecimal strings.

bLayer4Comm, bAddr.

bLayer4Comm: ISO14443 protocol to be used, 1 byte.

0x00 -- Use ISO14443 Layer 3 protocol.

0x01 -- Use ISO14443 Layer 4 protocol.

bAddr: Targeted ECC originality check signature, 1 byte (default:0x00).

Scanner → Host

CMD	Function Code	Data	Status	Card Data
NFCCMD	00C3	56 Bytes	2 Bytes	NULL

Card data: data is PICC's originality signature, 56 bytes.

Chapter 6 Examples

Example 1: MIFARE Classic

Read data in block 5.

Step 1: Set the working mode to business mode.

Request: NFCMOD2

Response: NFCMOD2<ACK>

Step 2: The MIFARE class card is close to the return UID

Device: 1daf2b9a

Step 3: Verify block5 key

Request: NFCCMD000A0500FFFFFFFFFF key defaults to 0xff, 0xff

Response: NFCCMD000A0500FFFFFFFFFFFF00<ACK>;

Step 4: Read the data of block5

Request: NFCCMD000305

Response: NFCCMD000305003132333435363738394041424344454 6<ACK>;

Example 2: MIFARE DESfire EVx

Read data: APP ID 0x00001 \File ID 0x00

Step 1: Discovery
Request: NFCCMD0011
Response: NFCCMD001100<ACK>;

Step 2: Card near output UID
Device: 04b44803030180

Step 3: Reset
Request: NFCCMD0018
Response: NFCCMD0018000675b3b00280<ACK>;

Step 4: FormatKeyEntry&SetKey AES128 key in hex: 00000000000000000000000000000000
Request: NFCCMD002B00000000
Response: NFCCMD002B0000000000<ACK>;
Request: NFCCMD002C00000000000000000000000000000000000000000000000000
Response: NFCCMD002C00000000000000000000000000000000000000000000000000<ACK>;

Step 5: Get Application IDs
Request: NFCCMD005E00
Response: NFCCMD005E0000010000<ACK>;

Step 6: Select Application 0x000001
Request: NFCCMD006100010000
Response: NFCCMD00610001000000<ACK>;

Step 7: Get File IDs
Request: NFCCMD0067
Response: NFCCMD006700000102<ACK>;

Step 8: AES Authentication (key store 00 card key 00)
Request: NFCCMD0051FFFF000000000000
Response: NFCCMD0051FFFF00000000000000<ACK>;

Step 9: Read Data File 0x00
Request: NFCCMD007300000000000000C0000
Response: NFCCMD007300000000000000C00000000000000000000000000000000<ACK>;

Example 3: MIFARE DESfire Light

Read data: FILE ID 0xDF01

Step1: Discovery
Request: NFCCMD0011
Response: NFCCMD001100<ACK>;

Step 2: Card near output UID
Device: 0447587af16780

Step 3: Reset
Request: NFCCMD0018
Response: NFCCMD001800067777710280<ACK>;

Step 4: FormatKeyEntry&SetKey AES128 key hex : 00000000000000000000000000000000
Request: NFCCMD002B00000000
Response: NFCCMD002B0000000000<ACK>;
Request: NFCCMD002C00000000000000000000000000000000000000000000000000
Response: NFCCMD002C00000000000000000000000000000000000000000000000000<ACK>;

Step 5: ISO Select File 0xDF01
Request: NFCCMD0046000001DF0000
Response:
NFCCMD0046000001DF00000000000000000000000000000000000000000000000000000000000000000000000000000000<ACK>;

Step 6: Authentication EV2
Request: NFCCMD002D01FFFF00000000010000
Response: NFCCMD002D01FFFF00000000010000000000000000000000000000000000<ACK>;
Request: NFCCMD002D00FFFF00000000010000
Response: NFCCMD002D00FFFF00000000010000000000000000000000000000000000<ACK>;

Step 7: Get File IDs
Request: NFCCMD0033
Response: NFCCMD0033000f1f03000104<ACK>;

Step 8: Get File Settings
Request: NFCCMD003500
Response: NFCCMD00350000000003301f000100<ACK>;

Step 9: Read Data
Request: NFCCMD0039300100000000100000

Response: NFCCMD00393001000000001000000000000000000000000000000000000000<ACK>;

Example 4: Ultralight/C/EV1/NTAG21x

Read data: (NTAG21x)

Step 1: Discovery
Request: NFCCMD0011
Response: NFCCMD001100<ACK>;

Step 2: Card near output UID
Device: 045de7aa064f80
Request: NFCCMD0095FFFFFFFF
Response: NFCCMD0095FFFFFFFF00<ACK>;

Step 3: Read Data
Request: NFCCMD009200
Response: NFCCMD00920000045de736aa064f806348337cf1323f37<ACK>;

Example 5: NTAG 42x DNA / TT

Read data

Step 1: Discovery
Request: NFCCMD0011
Response: NFCCMD001100<ACK>;

Step 2: Card near output UID
Device: 046c492aaa6180

Step 3: Reset
Request: NFCCMD0018
Response: NFCCMD001800067777710280<ACK>;

Step 4: ISO Select File 0xE110
Request: NFCCMD00A60C0010E10000
Response: NFCCMD00A60C0010E1000000<ACK>;

Step 5: FormatKeyEntry&SetKey AES128 key hex : 00000000000000000000000000000000
Request: NFCCMD002B00000000
Response: NFCCMD002B0000000000<ACK>;

Request: NFCCMD002C00000000000000000000000000000000000000000000000000

Response: NFCCMD002C00000000000000000000000000000000000000000000000000<ACK>;

Step 6: Authentication EV2

Request: NFCCMD009B01FFFF00000000000000

Response: NFCCMD009B01FFFF00000000000000000000000000000000000000000000<ACK>;

Request: NFCCMD009B00FFFF00000000000000

Response: NFCCMD009B00FFFF00000000000000000000000000000000000000000000<ACK>;

Step 7: Read Data

Request: NFCCMD00A4000101000000200000

Response:

NFCCMD00A4000101000000200000000001720010000ff0406e104010000000506e1050080828300000000000000000<ACK>;

Example 6: ICODE2

Read data.

Step 1: Discovery

Request: NFCCMD0011

Response: NFCCMD001100<ACK>;

Step 2: Card near output UID

Device: e004015097bda21a

Step 3: Select card

Request: NFCCMD0021E004015097BDA21A

Response: NFCCMD0021E004015097BDA21A00<ACK>;

Step 4: Read block

Request: NFCCMD002200010101E004015097BDA21A01

Response: NFCCMD002200010101E004015097BDA21A0100050111223344<ACK>;

Example 7: MIFARE PLUS-L1

Read and write block 28.

Step 1: Business mode

Request: NFCMOD2

Response: NFCMOD2<ACK>

Step 2: Discovery
Request: NFCCMD0011
Response: NFCCMD001100<ACK>;

Step 3: Card near output UID
Device: 24B5B35B

Step 4: FormatKeyEntry&SetKey, MIFARE key defaults to 0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff
Request: NFCCMD002B03000600;
Response: NFCCMD002B0300060000<ACK>;
Request: NFCCMD002C030000000600ffffffffffff0300
Response: NFCCMD002C030000000600ffffffffffff030000<ACK>;

Step 5: Verify the key for block 28
Request:NFCCMD00B21C0A030003000424B5B35B
Response:NFCCMD00B21C0A030003000424B5B35B00<ACK>;

Step 6: Read data from block 28
Request: NFCCMD00B90101011C0001
Response:
NFCCMD00B90101011C000100000000000000000000000000000000004F26FFFFFFFFFFFF<ACK>;

Step 7: Write block 28
Request: NFCCMD00B801011C0001FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Response:
NFCCMD00B801011C0001FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000000000000000000000000
0<ACK>;

Example 8: MIFARE PLUS-L3

Read block 28.

Step 1: Business mode
Request: NFCMOD2
Response: NFCMOD2<ACK>;

Step 2: Discovery
Request: NFCCMD0011
Response: NFCCMD001100<ACK>;

Step 3: Card near output UID
Device: 24B5B35B

Step 4: Reset
Request: NFCCMD0018
Response: NFCCMD0018000C75778002C1052F2F01BCD6<ACK>;

Step 5: Load key. Default key AES loaded key 0: AES key defaults to 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF
Request: NFCCMD002B00000000
Response: NFCCMD002B0000000000<ACK>;
Request: NFCCMD002C000000000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000
Response: NFCCMD002C000000000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000<ACK>;

Step 6: Authorization
Request: NFCCMD00B5010040000000000000
Response: NFCCMD00B501004000000000000000000000000000000000000000<ACK>;

Step 7： Read data from block 28
Request: NFCCMD00B90101011C0001
Response:
NFCCMD00B90101011C000100000000000000000000000000000004F26FFFFFFFFFFF<ACK>;

Step 8： Write block 28
Request: NFCCMD00B801011C0001FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Response:
NFCCMD00B801011C0001FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000000000000000000000000
0<ACK>;

Step 9: Read block 28
Request: NFCCMD00B90101011C0001
Response:
NFCCMD00B90101011C000100FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF4F26FFFFFFFFFFFF<AC
K>;

Appendix: Error Status Number Table

Error Status Number Table

Since the status code is only one byte, there may be duplicates, so the error code should be handled according to the following 2 different processes.

1. If the operated card is the card in Table 1 and Table 2, find the error code of the card first. If the error code is not found, then find the error code in Table 3. If it is not found yet, finally find the error code in Table 4.
2. If the operation card is not the card in Table 1 and Table 2, please directly find the error code in Table 4.

Table 1: MIFARE DESfire EV/Light/NTAG42X Error

Error Number	Description
0x81	MFDFEVx Response - Insufficient NV-Memory.
0x82	MFDFEVx Invalid key number specified.

0x83	MFDFEVx Current configuration/status does not allow the requested command.
0x84	MFDFEVx Requested AID not found on PICC.
0x85	MFDFEVx Attempt to read/write data from/to beyond the files/record's limits.
0x86	MFDFEVx Previous cmd not fully completed. Not all frames were requested or provided by the PCD.
0x87	MFDFEVx Num. of applns limited to 28. No additional applications possible.
0x88	MFDFEVx File/Application with same number already exists.
0x89	MFDFEVx Specified file number does not exist.
0x8A	MFDFEVx Crypto error returned by PICC. CRC or MAC does not match data padding bytes not valid.
0x8B	MFDFEVx Parameter value error returned by PICC.
0x8C	MFDFEVx DESfire Generic error. Invalid Command Error.
0x8D	MFDFEVx ISO 7816 Generic error. Check Additional Info.
0x8E	MFDFEVx ISO 7816 Generic error. Check Additional Info.
0x8F	MFDFEVx Not Supported Error.
0x90	MFDFEVx integrity error.
0x91	MFDFEVx memory error.
0x92	ISO7816 custom response code for memory failure.
0x93	ISO7816 custom response code for wrong length, no further indication.
0x94	ISO7816 custom response code for security status not satisfied.
0x95	ISO7816 custom response code for conditions of use not satisfied.
0x96	ISO7816 custom response code for file or application not found.
0x97	ISO7816 custom response code for incorrect parameters P1-P2.
0x98	ISO7816 custom response code for Lc inconsistent with parameter P1-P2.
0x99	ISO7816 custom response code for wrong LE field.
0x9A	ISO7816 custom response code for instruction code not supported or invalid.
0x9B	MFDFEVx custom error code for Successful operation with limited functionality.
0x9C	MFDFEVx Too many commands in the session or transaction.
0x9D	MFDFEVx Failure in the operation of the PD.
0x9E	MFDFEVx Invalid Block number: not existing in the implementation or not valid to target with this command.
0x9F	MFDFEVx Format of the command is not correct (e.g., too many or too few bytes).

Table 2: MIFARE Plus EVx Error

Error Number	Description
0x80	MFP EVx Authentication Error.
0x81	MFP EVx Command Overflow Error.
0x82	MFP EVx MAC Error.
0x83	MFP EVx Block Number Error.
0x84	MFP EVx Extension Error.
0x85	MFP EVx Invalid Command Error.
0x86	MFP EVx Format Error.
0x87	MFP EVx Generic Error.
0x88	MFP EVx Transaction MAC related Error.
0x89	MFP EVx Not Supported Error.
0x8A	MFP EVx 7816 wrong length error.
0x8B	MFP EVx 7816 wrong params error.
0x8C	MFP EVx 7816 wrong LC error.
0x8D	MFP EVx 7816 wrong LE error.
0x8E	MFP EVx Authentication Error.

Table 3: IC Error

Error Number	Description
0x01	No reply received, e.g., PICC removal.
0x02	Wrong CRC or parity detected.
0x03	A collision occurred.
0x04	Attempt to write beyond buffer size.
0x05	Invalid frame format.
0x06	Received response violates protocol.
0x07	Authentication error.
0x08	A Read or Write error occurred in RAM/ROM or Flash.
0x09	The RC sensors signal over-heating.
0x0A	Error due to RF.
0x0B	An error occurred in RC communication.
0x0C	A length error occurred.
0x0D	A resource error occurred.
0x0E	TX Rejected sanely by the counterpart.
0x0F	RX request Rejected sanely by the counterpart.
0x10	Error due to External RF.
0x11	EMVCo EMD Noise Error.
0x12	Used when HAL Abort is called.
0x13	LPCD is exited, without card detection.
0x7F	An internal error occurred.
0xAD	Authentication Delay.
0x20	Invalid data parameters supplied (layer id check failed).

0x21	Invalid parameter supplied.
0x22	Reading/Writing a parameter would produce an overflow.
0x23	Parameter not supported.
0x24	Command not supported.
0x25	Condition of use not satisfied.
0x26	A key error occurred.
0x27	Error occurred during initialization.
0x28	OSAL failed to perform the requested operation.
0x29	Error occurred during initialization.
0x30	Event failed to perform the requested operation.

Table 4: Device and Other Card Error

Error Number	Description
0x80	Read error.
0x81	Write error.
0x82	Command error.
0x83	Password error.
0x86	Card not detected.
0x8A	BCC checksum error.
0x8B	Card type error.
0x8C	Card dialing error.
0x8D	General error (parameter error).
0x8E	Communication data format error (command header error).
0x8F	Data length error.
0x95	Serial port occupied.
0xCB	Length error.
0xCF	Timeout error (card reply not received).
0xE0	The card holder has a card that is not powered up.
0xE1	Parameter error.
0xE2	Load code error.
0xE3	Request failed.
0xE4	Anti-conflict failure.
0xE5	Select card failed.
0xE6	Failed to get device password.
0xE7	Authentication password error.

0xE8	Value added error.
0xE9	Value reduction error.
0xEA	Restore error.
0xEB	Transfer error.
0xEC	CPU card reset error.
0xED	CPU card APDU command error.
0xEE	Failure to change password.

SCANNING MADE SIMPLE

Newland AIDC EMEA
+31 (0) 345 87 00 33
info@newland-id.com

Rolweg 25
4104 AV Culemborg
The Netherlands