MF3ICD81

MIFARE DESFire Functional specification

Rev. 3.5 — 28 November 2008 Product data sheet
134035 SECURED, STRICTLY CONFIDENTIAL INFORMATION

1. General description

NXP has developed the MIFARE DESFire (MF3ICD81) to be used with Proximity Coupling Devices (PCDs) according to ISO/IEC 14443 Type A. The communication protocol complies to part ISO/IEC 14443-4. The MF3ICD81 is primarily designed for secure contactless transport applications and related loyalty programs.In addition to ISO/IEC 14443 DESFire also supports the use of ISO/IEC 7816-3 compliant APDU message structure and ISO/IEC 7816-4 instruction set. The IC is designed for ID-1 formats and features therefore a resonance capacitor of 17 pF.

2. Features

2.1 RF interface: ISO/IEC 14443 Type A

- Contactless transmission of data and powered by the RF-field (no battery needed)
- Operating distance: Up to 100 mm (depending on antenna geometry)
- Operating frequency: 13.56 MHz
- Fast data transfer: 106 kbit/s, 212 kbit/s, 424 kbit/s, 848 kbit/s
- High data integrity: 16/32 bit CRC, parity, bit coding, bit counting
- True deterministic anticollision
- 7 byte unique identifier (cascade level two according to ISO/IEC 14443-3)
- Uses ISO/IEC 14443-4 protocol

2.2 ISO/IEC 7816 compatibility

- Supports ISO/IEC 7816-3 APDU message structure
- Supports ISO/IEC 7816-4 INS code ‘A4’ for SELECT FILE
- Supports ISO/IEC 7816-4 INS code ‘B0’ for READ BINARY
- Supports ISO/IEC 7816-4 INS code ‘D6’ for UPDATE BINARY
- Supports ISO/IEC 7816-4 INS code ‘B2’ for READ RECORDS
- Supports ISO/IEC 7816-4 INS code ‘E2’ for APPEND RECORD
- Supports ISO/IEC 7816-4 INS code ‘84’ for GET CHALLENGE
- Supports ISO/IEC 7816-4 INS code ‘88’ for INTERNAL AUTHENTICATE
- Supports ISO/IEC 7816-4 INS code ‘82’ for EXTERNAL AUTHENTICATE

2.3 Non-volatile memory

- 8 kB NV-Memory
- Data retention of 10 years
- Write endurance typcial 500 000 cycles

2.4 NV-memory organization

- Flexible file system
- Up to 28 applications simultaneously on one PICC
- Up to 32 files in each application

2.5 Security

- Unique 7 byte serial number for each device
- Optional “RANDOM” ID
- Mutual three pass authentication
- Mutual authentication according to ISO/IEC 7816-4
- Data encryption on RF-channel
- Hardware DES using 56/112/168 bit keys featuring key version, data authenticity by 8 byte CMAC
- Hardware AES using 128-bit keys featuring key version, data authenticity by 8 byte CMAC
- Authentication on application level
- Hardware exception sensors
- Self-securing file system
- Backward compatibility to MF3ICD40: 4 byte MAC
- Hardware exception sensors

2.6 Initial memory content

- Keys initialized to 0x00
- Same MIFARE type identification as for MF3ICD40 except from SAK

3. Ordering information

MIFARE DESFire will be available in the following delivery formats:

- Au-Bumped dies on 8” wafer, sawn on FFC, 120 μm thickness
- MOA4 contactless chip card module

4. Block diagram

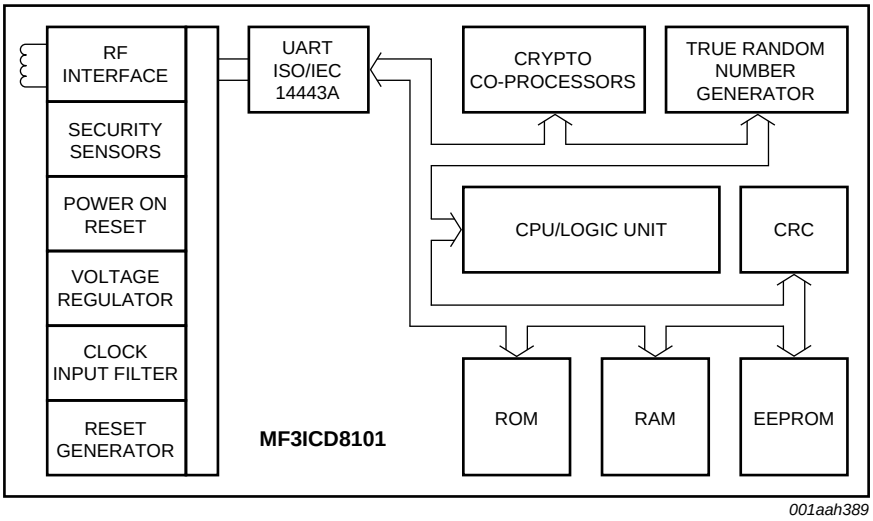

Fig 1. Block diagram

5. Pinning information

5.1 Pinning

See wafer spec addendum of device

6. Functional description

6.1 Contactless energy and data transfer

MIFARE

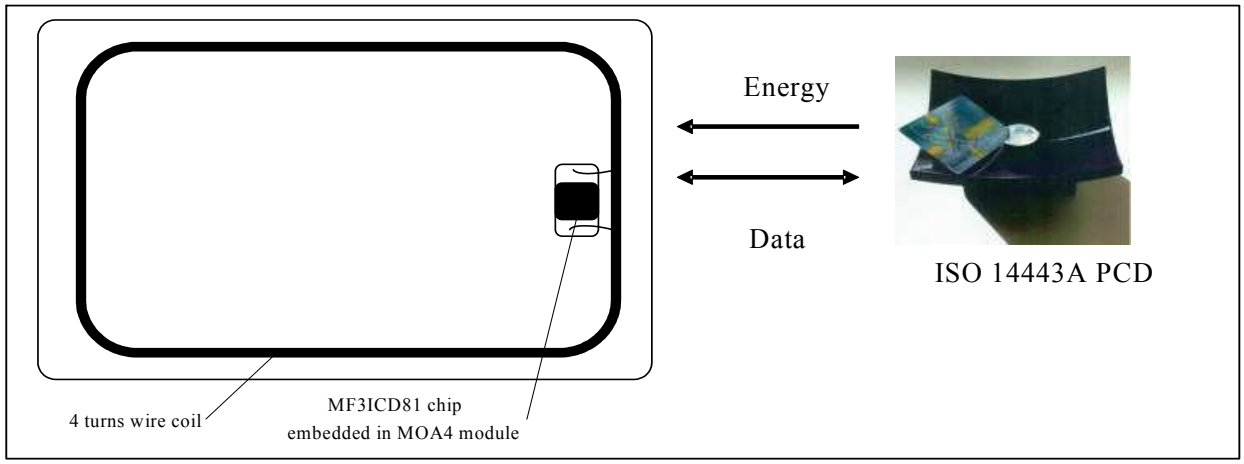

Fig 2. Contactless energy and data transfer

6.2 Delivery types

- Au-Bumped Dies on 8” wafer, sawn on FFC, 120 µm thickness
- MOA4 Contactless Chip Card Module

6.3 Anticollision

An intelligent anticollision mechanism allows to handle more than one PICC in the field simultaneously. The anticollision algorithm selects each PICC individually and ensures that the execution of a transaction with a selected PICC is performed correctly without data corruption resulting from other PICCs in the field.

6.4 UID / serial number

The unique 7 byte serial number (UID) is programmed into a locked part of the NV-memory which is reserved for the manufacturer. Due to security and system requirements these bytes are write-protected after beeing programmed by the IC manufacturer at production time. According to ISO/IEC 14443-3 during the first anticollision loop, see Section 9.1.3, the cascade tag will be returned 0x88 and the first 3 bytes of the UID, SN0 to SN2 and BCC. The second anticollision loop, see Section 9.1.4, will return bytes SN3 to SN6 and BCC.

SN0 holds the Manufacturer ID for NXP (04h) according to ISO/IEC 14443-3 and ISO/IEC 7816-6 AMD 1.

6.5 Random ID number

MF3ICD81 can be configured by the usage of the command ‘Set Configuration’ see Section 9.4.9 to use Random ID Numbers instead of Unique ID numbers during anti-collision procedure. In this case MF3ICD81 only uses a single anti-collision loop.The 3 byte random number is generated after RF-reset of the MF3ICD81.

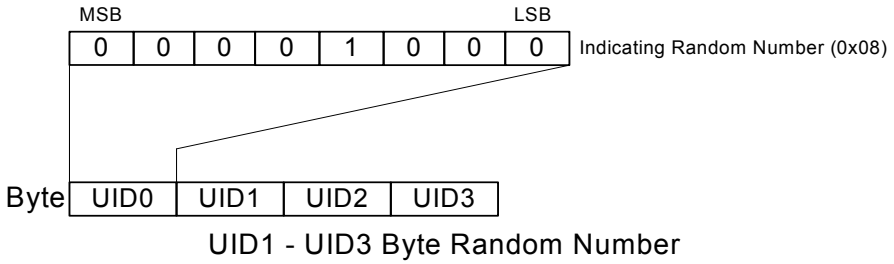

Fig 3. Random ID number

According to ISO/IEC 14443-3 the first anticollision loop, see MIFARE application note ISO/IEC 14443 PICC Selection, will return the Random Number Tag 0x08, the 3 bytes Random Number and the BCC.

6.6 Memory organization

The 8 kbyte NV-memory is organized using a flexible file system. This file system allows a maximum of 28 different applications on one single PICC. Each application provides up to 32 files. Other than for MF3ICD40, there are not limitations on the numbering between the different file types. Every application is represented by it's 3 bytes Application IDentifier, AID.

Five different file types are supported, see Section 8.1.

A guideline to assign DESFire AIDs can be found in the application note “MIFARE Application Directory, MAD”.

Each file can be created either at PICC initialization (card production/card printing), at PICC personalization (vending machine) or in the field.

If a file or application becomes obsolete in operation, it can be permanently invalidated.

Commands which have impact on the file structure itself (e.g. creation or deletion of applications, change of keys) activate an automatic rollback mechanism, which protects the file structure from getting corrupted.

If this rollback is necessary, it is done without user interaction before carrying out further commands.To ensure data integrity on application level, a transaction oriented backup is implemented for all file types with backup. It is possible to mix file types with and without backup within one application.

6.7 Security

The 7 byte UID is unchangeably programmed into each device during production. It cannot be altered and ensures the uniqueness of each device.

The UID may be used to derive diversified keys for each ticket. Diversified PICC keys contribute to gain an effective anti-cloning mechanism.

More information about the security concept can be read in Section 7.

7. Security concept

Prior to data transmission a mutual three pass authentication can be done between PICC and PCD. Depending on the requested authentication, the security level is chosen. Communication between PICC and PCD can be shown as follows (see also Figure 4):

Plain data transfer (only with authentication with the command authenticate () (command code 0x0A))

Plain data transfer with cryptographic checksum (MAC)

- Authentication with authenticate () (command code 0x0A):
  - DES/3DES: 4 byte MAC
- Authentications with 0x1A AuthenticateISO and 0xAA AuthenticateAES:
  - DES/3DES/AES: 8 byte MAC

Encrypted data transfer (secured by CRC before encryption)

- Authentication with authenticate () (command code 0x0A):
  - 16 bit CRC
- Authentications with 0x1A AuthenticateISO and 0xAA AuthenticateAES:
  - CRC32.
  - Be aware that the CRC32 is not inverted (XOR FF) at the end of the calculation.

Remark: The security concept for ISO/IEC 7816-4 commands is explained in Section 9.7.7.

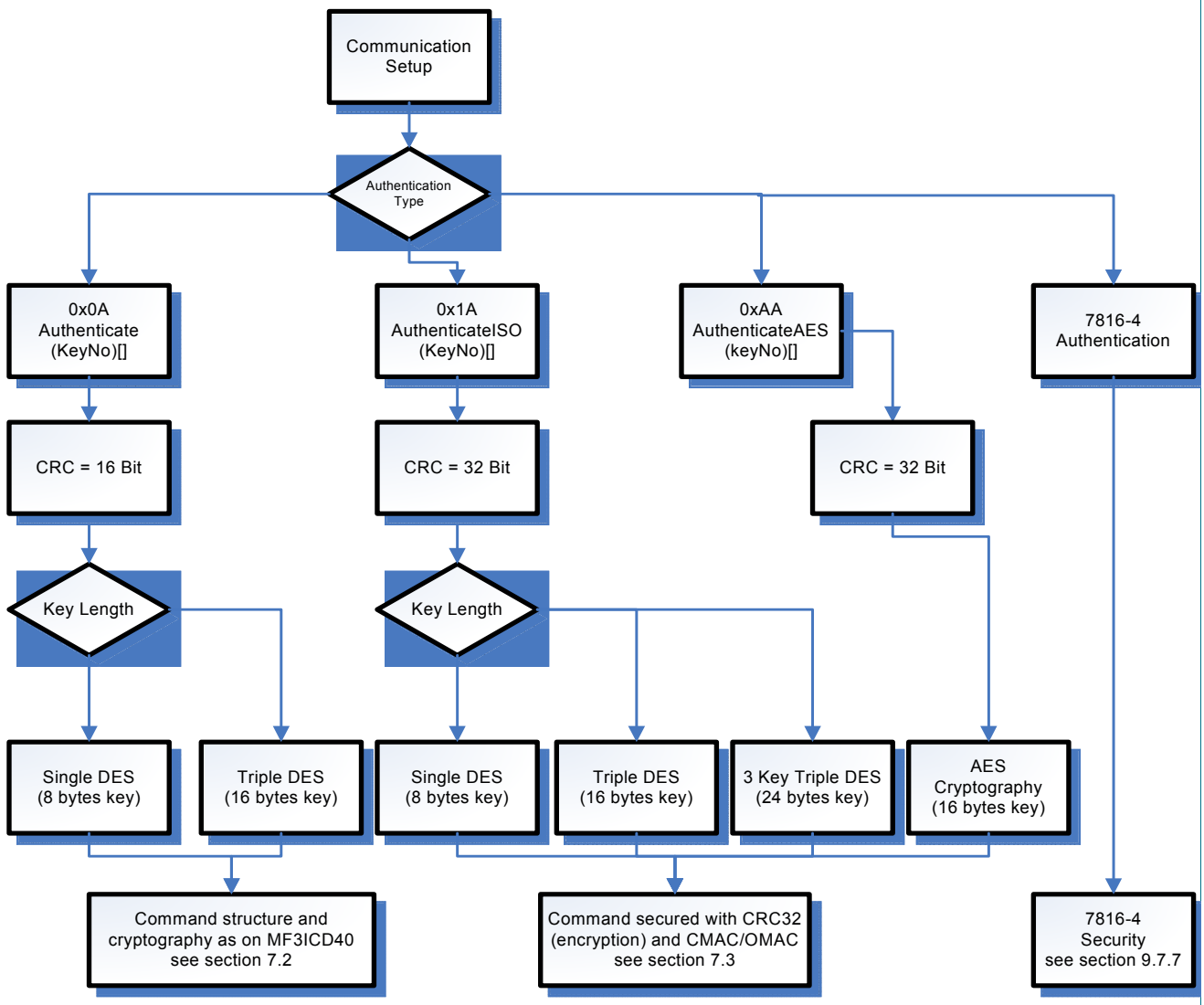

Fig 4. Communication setup

7.1 Authentication

Prior to data transmission a mutual three pass authentication can be done between PICC and PCD depending on the configuration employing either 56 bit DES (single DES, DES) see Section 7.4, 112 bit DES (triple DES, 3DES) see Section 7.4, 168 bit DES (3 key triple DES, 3K3DES) see Section 7.5 or AES see Section 7.6. During the authentication the security level of all further commands during the session is set.

Three pass authentication proves that both parties (PCD and PICC) are owner of a common secret (DES/3DES/3K3DES/AES key). The result of a successful authentication is a trusted link between both parties. The authentication command also yields a session key that you can use to protect the data transmission channel.

Below it is described in detail how the mutual 3-pass authentication procedure is done and how the session key is generated. Parts which differ from the MF3ICD40 backwards compatibility authentication (command code 0x0A) to the new authentications with 0x1A AuthenticateISO and 0xAA AuthenticateAES, are indicated.

Table 1. 3 pass authentication

#	PCD	Data exchanged	PICC
1	The reader device is always the entity which starts an authentication procedure. This is done by sending the command Authenticate. As parameter the key number is passed to the PICC in order to select a certain key stored in its NV-memory (up to 14 different keys per application). If the key number does not reflect a valid key in the PICC memory, an error code is sent by the PICC in response. Depending on the previously selected AID on the PICC, the subsequent authentication procedure is done for this specific AID. If the previously selected AID is 0x00, then the authentication is done using the PICC Master Key. In this case, the parameter key number has to be set to 0x00, too. (The possibilities and usage of the PICC Master Key is described later on.) After power up of the PICC the AID 0x00 is implicitly selected, which means that an Authenticate command after power-up always references the PICC Master Key.	Authenticate (KeyNo) →	
2		8 or 16 bytes ek_{keyNo}(RndB) ←	After a specific key is selected, the PICC generates an 8 byte (DES, 3DES) or 16 byte (3K3DES / AES) random number *RndB*. This random number is **en**ciphered with the selected key, denoted by ek_{keyNo}*(RndB),* and is then transmitted to the PCD.
3	The PCD runs a **de**ciphering operation on the received ek_{keyNo}*(RndB)* and thus retrieves *RndB*. (The used key for the deciphering obviously has to be the same as for the previous enciphering by the PICC.) In the next step *RndB* is rotated left by 8 bits (first byte is moved to the end of *RndB*), yielding *RndB'*. Authentication() (command code 0x0A): The PCD itself generates an 8 byte (DES/3DES) random number RndA. This RndA is concatenated with RndB' and **de**ciphered with the selected key. This token dk_{keyNo}*(RndA + RndB')* is sent to the PICC. AuthenticationISO, command code 0x1A, Authenticate AES, command code 0xAA: The PCD itself generates an 8 byte (DES/3DES) or 16 byte (3K3DES/AES) random number RndA. The RndA is concatenated with RndB' and **en**ciphered with the selected key. This token ek_{keyNo}*(RndA + RndB')* is sent to the PICC.	16 or 32 bytes ek/dk_{keyNo}(RndA+RndB') →	

Table 1. 3 pass authentication

#	PCD	Data exchanged	PICC
4		← 8 or 16 bytes $dk/ek_{keyNo}(RndA')$	Authentication() (command code 0x0A): The PICC runs an **en**cipherment on the received token and thus gains *RndA + RndB'*. The PICC can now verify the sent *RndB'* by comparing it with the *RndB'* obtained by rotating the original *RndB* left by 8 bits internally. AuthenticationISO, command code 0x1A, Authenticate AES, command code 0xAA: The PICC runs an **de**cipherment on the received token and thus gains *RndA + RndB'*. The PICC can now verify the sent *RndB'* by comparing it with the *RndB'* obtained by rotating the original *RndB* left by 8 bits internally. A successful verification proves to the PICC that the PICC and the PCD posses the same secret (key). If the verification fails, the PICC stops the authentication procedure and returns an error message. As the PICC also received the random number *RndA*, generated by the PCD, it can perform a rotate left operation by 8 bits on *RndA* to gain *RndA',* which is **en**ciphered again, resulting in $ek_{keyNo}(RndA')$. This token is sent to the PCD.

Table 1. 3 pass authentication

#	PCD	Data exchanged	PICC
5	The PCD runs a **de/en**cipherment on the received $e_{K_{keyNo}}$ *(RndA')* and thus gains *RndA'* for comparison with the PCD-internally rotated *RndA'*. If the comparison fails, the PCD exits the procedure and may halt the PICC.		
6			The PICC sets the authentication state for the currently selected application or the PICC itself (in case of AID=0x00).
7	If all comparisons are successful, the 8 - 24 bytes session key is obtained by employing *RndA* and *RndB*. The session key is gained by combining them according to the following rule: **DES session key** := $RndA_{1st\ half}$ + $RndB_{1st\ half}$ **3DES session key** := $RndA_{1st\ half}$ + $RndB_{1st\ half}$ + $RndA_{2nd\ half}$ + $RndB_{2nd\ half}$. This scrambling of *RndA* and *RndB* is done to avoid that a malicious PCD could degenerate 3DES cryptography to single DES operation by forcing *RndA* = *RndB*. **3KDES session key** := RndA byte 0..3 + RndB byte 0..3 + RndA byte 6..9 + RndB byte 6..9 + RndA byte 12..15 + RndB byte 12..15 **AES session key** := RndA byte 0..3 + RndB byte 0..3 + RndA byte 12..15 + RndB byte 12..15 With the generation of the session key the mutual 3-pass authentication is successfully completed.		

7.2 Backwards compatibility mode

If the authentication was performed with the command Authenticate, command code 0x0A, the commands shall be sent on the minimum security level. Ensuring backward compatibility, the native commands of MF3ICD40 can be called in exact the same way with the MF3ICD81.

This mode can be only used in combination with DES or 3DES cryptography. The keys used are always 16 bytes long.

Remark: After any error code during the transaction, the status is not set back and the authentication does not need to be renewed.

Security of data streams for reading a data stream from the card is implemented in the following way:

Depending on the communication settings (see Section 8.2) the following cases can be distinguished:

- Crypto Mode 0x00: Plain communication, see Section 7.2.1
- Crypto Mode 0x01: Plain communication secured by MACing, see Section 7.2.2
- Crypto Mode 0x03: Fully enciphered communication, see Section 7.2.3

7.2.1 Plain communication

The data transmitted and received is not secured. The data is completely sent in plain.

7.2.2 Plain communication secured by MACing

The data is padded to a full multiple of 8 and a MAC is calculated according to Section 7.2.4 over the resulting data stream and attached to the unpadded data stream.

7.2.3 Fully enciphered communication

The plain data is secured with CRC16, see Section 7.2.5, and attached to the plain data stream. Padding is performed according to Section 7.2.6 the resulting data stream is encrypted.

7.2.4 MAC

After padding the data stream according to Section 7.2.6 a MAC is calculated with the current session key over the data stream. The MAC is calculated by encryption of the respective stream using DES-algorithm. The 4 MSBs of the result are attached to the data stream sent/received from the card.

7.2.5 CRC16

The CRC is calculated with the following formula over the plain text prior to encryption:

$x^{16} + x^{12} + x^5 + 1$

7.2.6 Padding

For encryption, padding is performed with 0x00 to full multiple of 8 bytes string. Exception is unspecified data length in “Read Data”; where padding is performed with 0x80 and 0x00 (see ISO/IEC 9797 section 1, padding method 2) to a full multiple of 8 bytes string. Example: “01 02” would be padding according to ISO/IEC 9797/2 to “01 02 80 00 00 00 00 00”.

For MACing, padding is performed with 0x00 to a full multiple of 8 bytes string. As the backwards compatibility mode can only be used in combination with DES or 3DES, no other block length then 8 byte is used.

7.2.7 Encryption

The card calculates a CRC16 over the plain data and adds it to the data stream. Padding is performed according to Section 7.2.6. Then the card **en**ciphers the stream and sends it to the reader. If the commands are queued due to a very long data stream, the init vector for the decipherment is always updated. After the reading is finished, the init vector is reset to 0x00. The writing method can be secured in the same way as the reading method.

7.3 New Authentications

In addition to the authentication introduced already in MF3ICD40, which is described in the last section, new authentications were implemented, ensuring higher security level for the product. If the authentication was performed with the command AuthenticateISO, command code 0x1A or AuthenticateAES, command code 0xAA, the commands shall be sent on a high security level. 3DES as well as 3KDES or after AuthenticateAES also AES can be used to secure the communication.

Depending on the communication settings (see Section 8.2) the following cases can be distinguished:

- Crypto Mode 0x00: Plain communication, see Section 7.3.1

- Crypto Mode 0x01: Plain communication secured by MACing, see Section 7.3.2
- Crypto Mode 0x03: Fully enciphered communication, see Section 7.3.3

7.3.1 Plain communication

In principal a real plain communication does not exist after AuthenticateISO and AuthenticateAES. If plain communication is set for the sent data stream, the CMAC (see Section 7.3.4) is calculated to update the init vector, but not attached to the stream. If the answer is set to be plain, the data is always secured by a CMAC (see Section 7.3.4).

7.3.2 Plain communication secured by MACing

The data is padded according to Section 7.3.6 to a full block (8 byte for 3DES/3KDES, 16 byte for AES) according to Section 7.3.6 and the CMAC (see Section 7.3.4 is calculated and attached to the unpadded stream. The CMAC secures the communication from the card IC to the reader IC, it does not secure the communication from the reader IC to the card IC.

7.3.3 Fully enciphered communication

The plain data is secured with CRC32, see Section 7.2.6, and attached to the plain data stream. Padding is performed according to Section 7.2.6 the resulting data stream is encrypted (see Section 7.3.7).

7.3.4 CMAC

The CMAC is calculated according to the NIST special Publication 800-38B, which gives a recommendation for block cipher modes of operation. The init vector is updated after every transmitted and received command. Even if the CMAC is not attached “physically” to the stream, the calculation is performed to update the init vector. The same applies also if the sent stream is encrypted, then the init vector has to be updated. Therefore the init vector is only reset to 0x00 after authentication.

The CMAC is calculated over the data in the following orderlength:

- Cmd
- Header
- Plain Data
- Padding to a multiple of 8 bytes; padding according to CMAC

Table 2. Command sent to the card (not encrypted)

Command sent to the card (not encrypted):			
Cmd	Header	Plain Data	CMAC

The MAC shall not be added to the command, but the MAC needs to be calculated to retrieve the current INIT VECTOR which needs to be available for the next cryptographic calculation. (Even the data stream includes only one byte, the CMAC needs to be calculated).

Table 3. Status received from the card (not encrypted)

Status received from the card (not encrypted):		
Status	Plain Data	CMAC

The CMAC is calculated over the data in the following order:

- Plain data
- Status
- Padding

The CMAC is also attached to the stream, if only the status is returned. The CMAC is not calculated and attached, if an error code is returned. Then the application of the PICC leaves the authenticated state and needs to be authenticated again.

7.3.5 CRC32

The CRC32 is calculated with the following polynomial:

$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

The result is not inverted as described in some exemplary implementations from the web.

If a message to be sent is encrypted, the CRC32 is calculated over the following data in the following order:

- Cmd
- Header
- Plain data

The message to be received is encrypted, the CRC32 is calculated over the following data in the following order:

- Plain data
- Status

7.3.6 Padding

For encryption, padding is performed with 0x00 to full multiple of 8 bytes. Exception is unspecified data length in the commands “Read Data”; where padding is performed with 0x80 and 0x00 (see ISO/IEC 9797 section 1, padding method 2) to a full multiple of 8 bytes stream. Example: “01 02” would be padded according to ISO/IEC 9797/2 to “01 02 80 00 00 00 00 00”.

For the CMAC, the padding is rather different. After authentication a session key is retrieved, which is encrypted with 0x00 and then after shifting mechanism detailed in NIST special publication 800-38B, a key is retrieved. The first half is Key1, the second half is Key 2.

If the data block is a multiple of 8 (3DES/3KDES) or 16 bytes (AES) and does not need to be padded, the last block is XORed with Key1. If the data block is not a multiple of 8 (3DES/3KDES) or 16 bytes (AES) and needs to be padded, the last block (8 byte long for 3DES/3KDES and 16 byte long for AES) is padded according to ISO/IEC 9797 section 1, padding method 2, and on top XORed with Key 1.

Example for padding method according to ISO/IEC 9797 section 1, padding method 2: ’01 02’ would be padded according to ISO/IEC 9797/2 to ’01 02 80 00 00 00 00 00’.

7.3.7 Encryption

A CRC32 (see Section 7.3.5) is calculated over the plain data and added to the data stream. Padding is performed according to Section 7.3.6. Then the card **en**ciphers the stream and sends it to the reader. If the commands are queued due to a very long data stream, the init vector for the decipherment is always updated. After the reading is finished, the init vector is stored for the next command, where it is used for the encryption or CMAC calculation. The writing method can be secured in the same way as the reading method.

Table 4. Command sent to the card (encrypted)

Command sent to the card (encrypted):		
Cmd	Header	Enc (Plain Data+CRC32+padding according to CMAC)

Table 5. Status received from the card (encrypted)

Status received from the card (encrypted):	
Status	Enc (Plain Data + CRC32 + padding according to CMAC)

7.4 (3)DES-crypto operation

The DES standard defines two basic cryptographic operations: **en**cipherment and **de**cipherment.

It is equivalent to TDEA keying option 2 as described in FIPS PUB 46-3.

DESFire supports two modes of DES cipher operation:

Mode after authentication with 0x0A Authenticate:
PCD is ALWAYS deciphering, PICC is ALWAYS enciphering

Mode after authentication with 0x1A AuthenticateISO and 0xAA Authenticate AES:
sender is enciphering, receiver is deciphering or sender is deciphering and receiver is enciphering.

Table 6. 3DES operation

3DES operation	PICC	PCD
Key1= 1st 8 bytes of 3DES Key	Encipher	Decipher
Key2= 2nd 8 bytes of 3DES Key	Decipher	Encipher
Key3= 1st 8 bytes of 3DES Key	Encipher	Decipher

DES/3DES always operates on 8 bytes. Therefore data streams are always padded to a length of multiples of 8 bytes.

All cryptographic operations are done in “cipher block chaining mode”, which defines the result of the previous operation to be the Init-vector of the next cryptographic operation. For sending data the "CBC send mode" is used, for receiving always "CBC receive mode".

The data structure regarding the calculation of the MAC and type of CRC used is detailed on Section 6.7.

The original DES standard specifies one parity bit for every byte in the DES key, causing an overhead of 8 bits per DES key. Modern 3DES Hardware does not need parity bits any more to ensure data integrity. DESFire uses these bits to store one byte of data that can be used to store the version of the key. See one example in Section 8.2. This is called key versioning. When you write a key to DESFire, you always write this version information (in your key generation process, make sure you overwrite the parity bits with the key version, otherwise you endanger the security level of your cryptographic system, as the parity bits (= key version) can by read with the GetKeyVersion command, see Section 9.3.7).

Key Version is only maintained in the left most 8 bytes. All other bytes are not used for the key version.

To store a key version to the PICC, the new version number needs to be coded by the application software into the new key before issuing the ChangeKey command. The key version information is NOT checked by the PICC in any respect.

Remark: Even if the Key versioning feature is NOT used, the application needs to make sure that the parity bits of every single byte in the new DES / 3DES key are overwritten with a default value (e.g. “0”) as they can be read out using the GetKeyVersion command.

In case the parity information is not overwritten, the valid parity bits are returned, which significantly decreases the cryptographic strength of the key.

7.5 3 key 3 DES-crypto operation

3 Key 3 DES is the TDEA keying option 1 as described in FIPS PUB 46-3 and ANSI x9.52-1998.

Table 7. 3 key 3 DES-crypto operation

3DES operation	PICC	PCD
Key1= 1st 8 bytes of 3K3DES Key	Encipher	Decipher
Key2= 2nd 8 bytes of 3K3DES Key	Decipher	Encipher
Key3= 3rd 8 bytes of 3K3DES Key	Encipher	Decipher

3K3DES always operates on 8 bytes. Therefore data streams are always padded to a length of multiples of 8 bytes.

All cryptographic operations are done in “cipher block chaining mode”, which defines the result of the previous operation to be the Init-vector of the next cryptographic operation. For sending data the "CBC send mode" is used, for receiving always "CBC receive mode".

The data structure regarding the calculation of the MAC and type of CRC used is detailed on Section 6.7

The original DES standard specifies one parity bit for every byte in the DES key, causing an overhead of 8 bits per DES key. Modern 3DES Hardware does not need parity bits any more to ensure data integrity. DESFire uses these bits to store one byte of data that can be used to store the version of the key. This is called key versioning. When you write a key to DESFire, you always write this version information (in your key generation process, make sure you overwrite the parity bits with the key version, otherwise you endanger the security level of your cryptographic system, as the parity bits (= key version) can be read with the GetKeyVersion command, see Section 9.3.7.

To store a key version to the PICC, the new version number needs to be coded by the application software into the new key before issuing the ChangeKey command. The key version information is NOT checked by the PICC in any respect.

Key Version is only maintained in the left most 8 bytes. All other bytes are not used for the key version.

Remark: Even if the Key versioning feature is NOT used, the application needs to make sure that the parity bits of every single byte in the new key are overwritten with a default value (e.g. “0”) as they can be read out using the GetKeyVersion command.

In case the parity information is not overwritten, the valid parity bits are returned, which significantly decreases the cryptographic strength of the key.

7.6 AES-crypto operation

For details on AES (also known as “Rijndael”) please refer to publicly available standard FIPS PUB 197, announced by the “National Institute of Standards and Technology”.

The AES standard defines two basic cryptographic operations: encipherment and decipherment.

AES always operates on 16 bytes. Therefore data streams are always padded to a length of multiples of 16 bytes.

The key size is always 128 bit.

The data structure regarding the calculation of the CMAC and type of CRC used is detailed on Section 6.7.

Remark: In case the length of data in the MF3ICD81 read command is given padding with "0x00 00 00" (all data bytes have to be read), padding to n*16 Bytes has to be done with once "0x80" and the rest "0x00".; if length is not indicated (like write data or read data).

To store a key version to the PICC, the new version number needs to be supplied by the application software as a parameter for the ChangeKey command. The key version information is NOT checked by the PICC in any respect.

8. MF3ICD81 – Coding of security-, application- and file- related features

8.1 Coding of file types

The files within an application can be of different types as:

- Standard Data Files (coded as 0x00)
- Backup Data Files (coded as 0x01)
- Value Files with Backup (coded as 0x02)
- Linear Record Files with Backup(coded as 0x03)
- Cyclic Record Files with Backup(coded as 0x04)

Other than for MF3ICD40, there are not limitations on the numbering between the different file types.

8.2 Coding of communication settings – crypto modes

The Communication settings define the level of security for the communication between PCD and PICC. Communication settings always apply on file-level.

If the communication mode is enciphered, but the file is configured for free access, then the communication is either in plain (Authentication 0x0A) or CMACed (other Authentications).

The settings are coded into one byte which needs to be set to

Table 8. Coding of communication settings

Communication mode	bit 7 – bit 2	bit 1	bit 0
Plain communication	RFU = 0	ignored	0
Plain communication secured by MACing	RFU = 0	0	1
Fully enciphered communication	RFU = 0	1	1

Both DES and 3DES keys are stored in strings consisting of 16 bytes:

If the 2nd half of the key string is equal to the 1st half, the key is handled as a single DES key by the PICC.

If the 2nd half of the key string is not equal to the 1st half, the key is handled as a 3DES key.

Even if the first 8 byte key itself is the same as the second 8 byte, but the version is coded into one half, it is used as a 3DES key and not as a single DES key during authentication and session key generation (see example in Table 9).

Examples for single DES keys:
0x00 11 22 3344 55 66 7600 11 22 3344 55 66 76

Examples for 3DES keys:
0x0011 2233 4455 6677 8899 AABB CCDD EEFF 0A1A 2A3A 4A5A 6A7A

A 3KDES key is stored in 24 bytes.

Example for 3KDES key:
0x0011 2233 4455 6677 8899 AABB CCDD EEFF 000A 1A2A 3A4A 5A6A

Key versions:

Apart from AES keys, where the key version is stored in the key settings, the key version for a DES key is stored in the so called parity bit. For single double or triple length keys, only 8 left bytes of the key stream are concerned for the key version. See the following example:

Table 9. Example for single DES keys

Key (hex)	00	11	22	33	44	55	66	76	00	11	22	33	44	55	66	77
Version (bin)	0	1	0	1	0	1	0	0	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a

So the version would be: 0x54

The “real” key is 0x0011 2233 4455 6676, see example in this section about single DES keys.

8.3 Coding of access rights

There are four different access rights (2 bytes for each file) stored for each file within each application:

Read Access (GetValue, Debit for Value files)

Write Access(GetValue Section 9.6.3, Debit, LimitedCredit for Value files)

Read&Write Access (GetValue Section 9.6.3, Debit, LimitedCredit, Credit, for Value files)

ChangeAccessRights

Each of the access rights is coded in 4 bits, one nibble. Each nibble represents a link to one of the keys stored within the respective application's key file.

One nibble allows to code 16 different values. If such a value is set to a number between 0 and 13 (max. 14 keys), this references a certain key within the application's key file, provided that the key exists (selecting a non-existing key is not allowed).

If the number is coded as 14 (0xE) this means “free” access.Thus access is granted always, independent on a previous authentication of the user. The number 15 (0xF) defines the opposite of “free” access and has the meaning “deny” access. Therefore the respective linked access rights is always denied.The most significant 4 bits of the two bytes parameter code the reference number of the key which is necessary to know for getting Read Access (in case of Value files: for the GetValue and the Debit Command).The next 4 bits hold the number of the key for getting Write Access (in case of Value files: GetValue, Debit and LimitedCredit Command).The upper nibble of the lower byte holds the key number for getting Read&Write Access, in Value files this right allows full access (in case of Value files: GetValue, Debit, LimitedCredit and Credit Command; in

case of Record files additionally: ClearRecordFile).The least significant nibble holds the reference number of the key, which is necessary to be authenticated with in order to change the access rights for the file and to link each access right to key numbers.

Table 10. Access rights are always packed in 2 bytes as follows:

15 12	11 8	7 4	3 0
Read Access	Write Access	Read&Write Access	Change Access Rights
MS Bit			LS Bit

Read is possible with Read Access and Read&Write Access.

Write is possible with Write Access and Read&Write Access.

If a file is accessed without valid authentication but free access (0xE) is possible through at least one relevant access right, the communication mode is forced to plain.

If only one of the keys for “Read” and “Read&Write” access (or "Write" and "Read&Write" access) is set to 0xE, the other key is different from 0xE, communication is done MACed/enciphered in case of a valid authentication and done in plain in case of no valid authentication. In the second case the communication settings, see Section 8.2, are ignored by the PICC.

8.4 Chaining

The frame length for MF3ICD81 is 64 bytes. Especially for data manipulation commands, it makes sense to offer chaining.

Chaining can be used in various commands like Read Data, Write Data with “AF” see in the specification of the command. It is not used in the ISO/IEC 7816-4 command set.

8.5 Coding of status- and error codes

Table 11. Coding of status- and error codes

Hex code	Status	Description
0x00	OPERATION_OK	Successful operation
0x0C	NO_CHANGES	No changes done to backup files, CommitTransaction / AbortTransaction not necessary
0x0E	OUT_OF_EEPROM_ERROR	Insufficient NV-Memory to complete command
0x1C	ILLEGAL_COMMAND_CODE	Command code not supported
0x1E	INTEGRITY_ERROR	CRC or MAC does not match data Padding bytes not valid
0x40	NO_SUCH_KEY	Invalid key number specified
0x7E	LENGTH_ERROR	Length of command string invalid
0x9D	PERMISSION_DENIED	Current configuration / status does not allow the requested command
0x9E	PARAMETER_ERROR	Value of the parameter(s) invalid
0xA0	APPLICATION_NOT_FOUND	Requested AID not present on PICC
0xA1	APPL_INTEGRITY_ERROR	Unrecoverable error within application, application will be disabled [1]
0xAE	AUTHENTICATION_ERROR	Current authentication status does not allow the requested command
0xAF	ADDITIONAL_FRAME	Additional data frame is expected to be sent
0xBE	BOUNDARY_ERROR	Attempt to read/write data from/to beyond the file's/record's limits. Attempt to exceed the limits of a value file.
0xC1	PICC_INTEGRITY_ERROR	Unrecoverable error within PICC, PICC will be disabled [1]
0xCA	COMMAND_ABORTED	Previous Command was not fully completed Not all Frames were requested or provided by the PCD
0xCD	PICC_DISABLED_ERROR	PICC was disabled by an unrecoverable error [1]
0xCE	COUNT_ERROR	Number of Applications limited to 28, no additional CreateApplication possible
0xDE	DUPLICATE_ERROR	Creation of file/application failed because file/application with same number already exists
0xEE	EEPROM_ERROR	Could not complete NV-write operation due to loss of power, internal backup/rollback mechanism activated[1]
0xF0	FILE_NOT_FOUND	Specified file number does not exist
0xF1	FILE_INTEGRITY_ERROR	Unrecoverable error within file, file will be disabled [1]

[1] These errors are not expected to appear during normal operation.

8.6 DESFire command set overview - ISO/IEC 14443-3

Table 12. ISO/IEC 14443-3

Command	Description
REQA	REQA and ATQA are implemented fully according to ISO/IEC 14443-3.
WUPA	WAKE-UP is implemented fully according to ISO/IEC 14443-3.
ANTICOLLISION / SELECT Cascade Level 1	The ANTICOLLISION and SELECT commands are implemented fully according to ISO/IEC 14443-3.The response is part 1 of the UID.
ANTICOLLISION / SELECT Cascade Level 2	The ANTICOLLISION and SELECT commands are implemented fully according to ISO/IEC 14443-3.The response is part 2 of the UID.

8.7 DESFire command set overview - ISO/IEC 14443-4

Table 13. ISO/IEC 14443-4

Command	Description
RATS	The response to the RATS command identifies the PICC type to the PCD.
PPS	The PPS command allows to individually select the communication baud rate between PCD and PICC. For DESFire it is possible to individually set the communication baud rate independently for both directions i.e. DESFire allows a non-symmetrical information interchange speed.
WTX	If the PICC needs more time than the defined FWT to respond to a PCD command it will send a request for a waiting time extension.

8.8 MF3ICD81 command set overview – security related commands

Table 14. Security related commands

Hex	Command	Description
0x0A	Authenticate	In this procedure both, the PICC as well as the reader device, show in an encrypted way that they posses the same secret which especially means the same key. This procedure not only confirms that both entities are permitted to do operations on each other but also creates a session key which can be used to keep the further communication path secure. As the name “session key” implicitly indicates, each time a new authentication procedure is successfully completed a new key for further cryptographic operations is generated.
0x1A	AuthenticateISO	In this procedure both, the PICC as well as the reader device, show in an encrypted way that they posses the same secret which especially means the same key. This procedure not only confirms that both entities are permitted to do operations on each other but also creates a session key which can be used to keep the further communication path secure. As the name “session key” implicitly indicates, each time a new authentication procedure is successfully completed a new key for further cryptographic operations is generated.
0xAA	AuthenticateAES	In this procedure both, the PICC as well as the reader device, show in an encrypted way that they posses the same secret which especially means the same key. This procedure not only confirms that both entities are permitted to do operations on each other but also creates a session key which can be used to keep the further communication path secure. As the name “session key” implicitly indicates, each time a new authentication procedure is successfully completed a new key for further cryptographic operations is generated.
0x54	Change KeySettings	Changes the master key settings on PICC and application level.
0x5C	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.
0xC4	Change Key	Changes any key stored on the PICC.
0x64	Get KeyVersion	Reads out the current key version of any key stored on the PICC.

Remark: All command & data frames are exchanged between PICC and PCD by using block format as defined in ISO/IEC 14443-4.

8.9 MF3ICD81 command set overview – PICC level commands

Table 15. PICC level commands

Hex	Command	Description
0xCA	Create Application	Creates new applications on the PICC.
0xDA	Delete Application	Permanently deactivates applications on the PICC.
0x6A	Get Applications IDs	Returns the Application IDentifiers of all applications on a PICC.
0x6E	Free Memory	Returns the free memory available on the card
0x6D	GetDFNames	Returns the DF names
0x45	Get KeySettings	Gets information on the PICC and application master key settings. In addition it returns the maximum number of keys which are configured for the selected application.
0x5A	Select Application	Selects one specific application for further access.
0xFC	FormatPICC	Releases the PICC user memory.
0x60	Get Version	Returns manufacturing related data of the PICC.
0x51	GetCardUID	Returns the UID.

Remark: All command & data frames are exchanged between PICC and PCD by using block format as defined in ISO/IEC 14443-4. Prior to using these commands you also need to select the PICC with 0x5A.

8.10 MF3ICD81 command set overview – application level commands

Table 16. Application level commands

Hex	Command	Description
0x6F	Get FileIDs	Returns the File IDentifiers of all active files within the currently selected application.
0xF5	Get FileSettings	Get information on the properties of a specific file.
0x5F	Change FileSettings	Changes the access parameters of an existing file.
0xCD	Create StdDataFile	Creates files for the storage of plain unformatted user data within an existing application on the PICC.
0xCB	Create BackupDataFile	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.
0xCC	Create ValueFile	Creates files for the storage and manipulation of 32bit signed integer values within an existing application on the PICC.
0xC1	Create LinearRecordFile	Creates files for multiple storage of structural similar data, for example for loyalty programs, within an existing application on the PICC. Once the file is filled completely with data records, further writing to the file is not possible unless it is cleared.
0xC0	Create CyclicRecordFile	Creates files for multiple storage of structural similar data, for example for logging transactions, within an existing application on the PICC. Once the file is filled completely with data records, the PICC automatically overwrites the oldest record with the latest written one. This wrap is fully transparent for the PCD.
0xDF	DeleteFile	Permanently deactivates a file within the file directory of the currently selected application.
0x61	Get ISOFile IDS	Get back the ISO File ID.

Remark: All command & data frames are exchanged between PICC and PCD by using block format as defined in ISO/IEC 14443-4.

8.11 MF3ICD81 command set overview – data manipulation commands

Table 17. Data manipulation commands

Hex	Command	Description
0xBD	Read Data	Reads data from Standard Data Files or Backup Data Files.
0x3D	Write Data	Writes data to Standard Data Files or Backup Data Files.
0x6C	Get Value	Reads the currently stored value from Value Files.
0x0C	Credit	Increases a value stored in a Value File.
0xDC	Debit	Decreases a value stored in a Value File.
0x1C	Limited Credit	Allows a limited increase of a value stored in a Value File without having full Credit permissions to the file.
0x3B	Write Record	Writes data to a record in a Cyclic or Linear Record File.
0xBB	Read Records	Reads out a set of complete records from a Cyclic or Linear Record File.
0xEB	Clear RecordFile	Resets a Cyclic or Linear Record File to empty state.
0xC7	Commit Transaction	Validates all previous write access' on Backup Data Files, Value Files and Record Files within one application.
0xA7	Abort Transaction	Invalidates all previous write access' on Backup Data Files, Value Files and Record Files within one application.

Remark: All command & data frames are exchanged between PICC and PCD by using block format as defined in ISO/IEC 14443-4.

9. DESFire command set

9.1 Command set ISO/IEC 14443-3

The MF3ICD81 accepts the following command set according to ISO/IEC 14443-3 activation:

9.1.1 Request type A (REQA)

Table 18. Request type A (REQA)

Code	Parameter	Data	Integrity mechanism	Response
26 (7 Bit)	-	-	-	0344

Description: The MF3ICD81 accepts the REQA command in Idle state only. The response is the 2-byte ATQA (0344). REQA and ATQA are implemented fully according to ISO/IEC 14443-3.

9.1.2 Wake-Up (WUPA)

Table 19. Wake-Up (WUPA)

Code	Parameter	Data	Integrity mechanism	Response
52 (7Bit)	-	-	-	0344

Description: The MF3ICD81 accepts the WAKE-UP command in the Idle and Halt state only. The response is the 2-byte ATQA (0344). WAKE-UP is implemented fully according to ISO/IEC 14443-3.

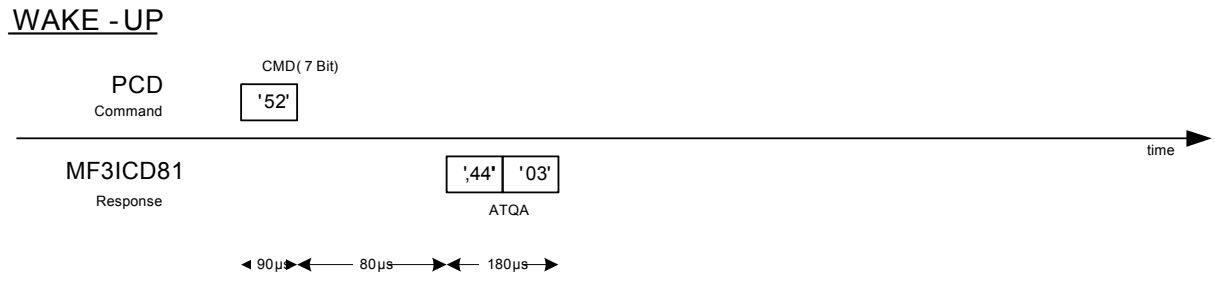

Fig 5. Wake-Up

9.1.3 ANTICOLLISION and SELECT of cascade level 1

Table 20. ANTICOLLISION and SELECT of cascade level 1

Code	Parameter	Data	Integrity mechanism	Response
Anticollision: 93	20 – 67	Part of the UID	Parity	Parts of UID
Select: 93	70	Cascade Tag, First 3 bytes of UID	Parity, BCC, CRC	SAK (24)

Description: The ANTICOLLISION and SELECT commands are based on the same command code. They differ only in the parameter byte. This byte is per definition 70 in case of SELECT. The MF3ICD81 accepts these commands in the Ready1 state only. The response is part 1 of the UID.

For more details on the possible parameters please see ISO/IEC 14443-3.

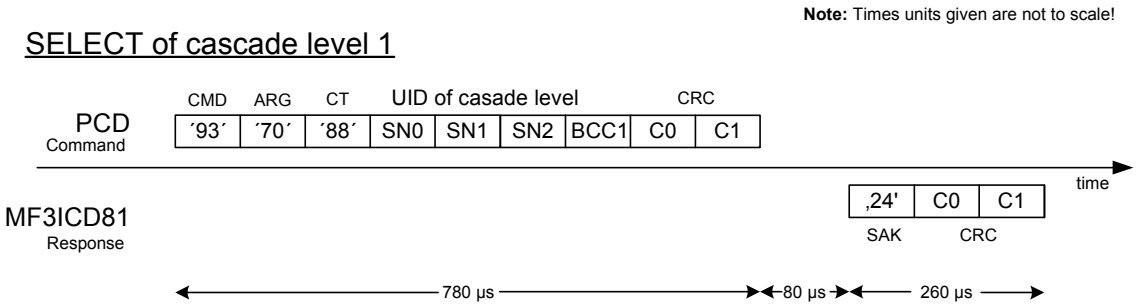

Fig 6. SELECT of cascade level 1

9.1.4 ANTICOLLISION and SELECT of cascade level 2

Table 21. ANTICOLLISION and SELECT of cascade level 2

Code	Parameter	Data	Integrity mechanism	Response
Anticollision: 0x95	20 – 67	Part of the UID	Parity	Parts of UID
Select: 0x95	70	Second 4 bytes of UID	Parity, BCC, CRC	SAK (20)

Description: The ANTICOLLISION and SELECT command are based on the same command code. They differ only in the parameter byte. This byte is per definition 70 in case of SELECT. The MF3ICD81 accepts these commands in the Ready2 state only. The response is part 2 of the UID.

For more details on the possible parameters please see ISO/IEC 14443-3.

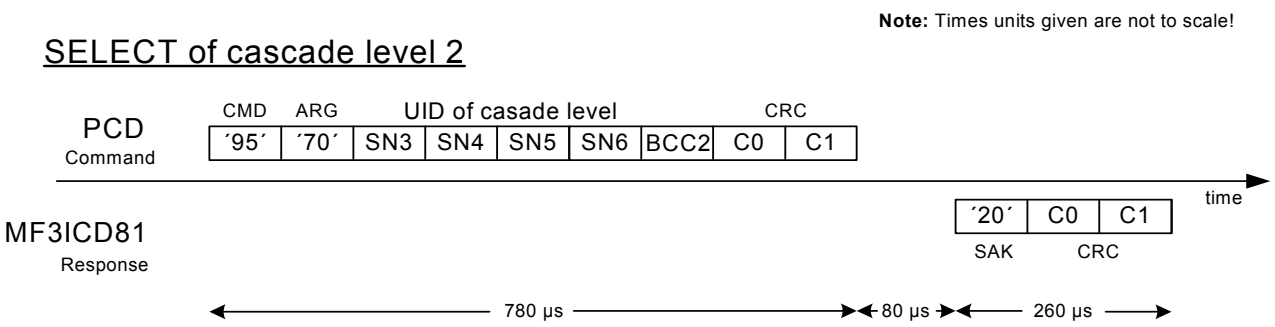

Fig 7. ANTICOLLISION and SELECT of cascade level 2

9.2 Command Set ISO/IEC 14443-4

The MF3ICD81 provides the following command set according to ISO/IEC 14443-4:

9.2.1 Request for answer to select (RATS)

Table 22. Request for answer to select (RATS)

Code	Parameter	Data	Integrity mechanism	Response
0xE0	High Nibble: FSDI Low Nibble: CID	-	CRC	16 Byte Data

Description: The response to the RATS command communicates the PICC capabilities to the PCD. The parameter byte codes two different parameters for communication:

- FSDI: The high nibble of the parameter byte codes the maximum frame size supported by the PCD for communication with the PICC.
- CID: The Low Nibble of the parameter byte codes the logical number of the addressed PICC. This logical number is in the range from 0x00 to 0x0E. This CID is used to distinguish several PICCs simultaneously selected by a single PCD. The DESFire PICC fully supports CIDs, which is coded in TC(1).

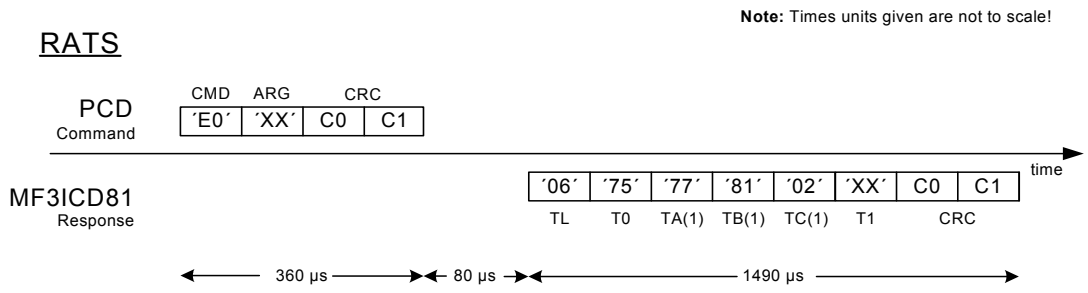

Fig 8. Request for answer to select (RATS)

The response of the MF3ICD81 to the RATS is the 'Answer to Select', ATS. This ATS consists of the following bytes:

TL: 'Length Byte', the length byte TL specifies the length of the transmitted ATS including itself. The two CRC bytes are not included in TL. For the MF3ICD81 device TL is set to 0x06.

T0: 'Format Byte', the format bytes defines the presence of the subsequent bytes TA(1), TB(1) and TC(1). All three are present, resulting is 0x7 for the higher nibble. The lower nibble (FSCI) specifies the maximum size of a frame accepted by the MF3ICD81 which is 64 bytes, coded as 0x5. The format byte therefore is 0x75.

TA(1): The 'Interface byte TA(1)' codes the maximum possible data rates supported by the PICC. As the MF3ICD81 supports date rates up to 848 kbaud in both directions (PICC to PCD and PCD to PICC), the TA(1) byte is set to 0x77.

TB(1): The higher nibble codes the Frame Waiting Time (FWT), which is set to 0x8 for DESFire, indicating 77.33 ms. The lower nibble codes the start-up frame guard time (SFGT). It is set to 0x1, indicating 604µs. The PCD will therefore receive 0x81.

TC(1): CID is supported, NAD is not supported, therefore TC(1) is set to 0x02.

T1: The DESFire PICC sends one byte as historical character which should be ignored by the application software.

9.2.2 Protocol and parameter selection request (PPS)

Table 23. Protocol and parameter selection request (PPS)

Code	Parameter	Data	Integrity mechanism	Response
'PPSS: 'DX	PPS0 PPS1	-	CRC	PPSS: 'D0'

Description: The PPS command allows to individually select the communication baud rate between PCD and PICC. For DESFire it is possible to individually set the communication baud rate independently for both directions i.e. DESFire allows a non-symmetrical information interchange speed.

- PPSS: The higher nibble of the PPSS byte needs to be set to 'D', all other values are RFU. The lower nibble indicates the CID of the selected PICC in the range of 0x00 and 0x0E.
- PPS0: The PPS Parameter 0 byte indicates the presence of PPS1 (which is valid for DESFire) and therefore has to be set to 0x11.
- PPS1: The PPS Parameter 1 byte defines the divisor integer for timings between PCD and PICC which directly defines the baud rate in each direction.

  The higher nibble of the PPS1 byte is RFU and has to be set to '0'. Bits b3 and b2 code the divisor integer from PICC to PCD and are called DSI. Bits b1 and b0 code the divisor integer from PCD to PICC and are called DRI.

The higher nibble of the PPS1 byte is RFU and has to be set to '0'. Bits b3 and b2 code the divisor integer from PICC to PCD and are called DSI. Bits b1 and b0 code the divisor integer from PCD to PICC and are called DRI.

The coding of DRI and DSI is done as specified below:

Table 24. Protocol and parameter selection request (PPS

DRI, DSI bit coding	00	01	10
Divisor	1	2	4

DESFire supports baud rates up to 848 kbaud in both directions. please note that it is possible to independently set the communication speed in both directions, which allows for example to use 106 kbaud for communication from PCD to PICC (DRI=00) and 424 kbaud from PICC to PCD (DSI=10).

For communication with 106 kbaud in both directions the value of PPS1 has to be '00', which is the default if no PPS command is sent to the PICC.

The response of the MF3ICD81 to the PSS command is the PPS start byte (PPSS, 0xD0). Invalid PPS requests are ignored (as defined in ISO).

9.2.3 Frame waiting extensions (WTX)

If the PICC needs more time than the defined FWT to respond to a PCD command it will send a request for a waiting time extension.

A 14443-4 S-block in sent by the PICC. According to ISO the INF field will contain the value 0x01, requesting another FWT interval.

The PCD has to confirm the request by sending another S-block either confirming 0x01 in the INF field.

9.3 MF3ICD81 command set – security related commands

The MF3ICD81 provides the following command set for security related functions:

9.3.1 Authenticate (3)DES [2 bytes]

This command can be used with DES and 3DES.In this procedure both, the PICC as well as the reader device, show in an encrypted way that they posses the same secret which especially means the same key. This procedure not only confirms that both entities can trust each other but also generates a session key which can be used to keep the further communication path secure. As the name “session key” implicitly indicates, each time a new authentication procedure is successfully completed a new key for further cryptographic operations is obtained.

After a successful authentication with this command the reader needs to decipher data to prepare it for sending and decipher data after receiving. This is the authentication, which was already used with MF3ICD40, where the IC could only encipher, no deciphering was possible by the Card IC.

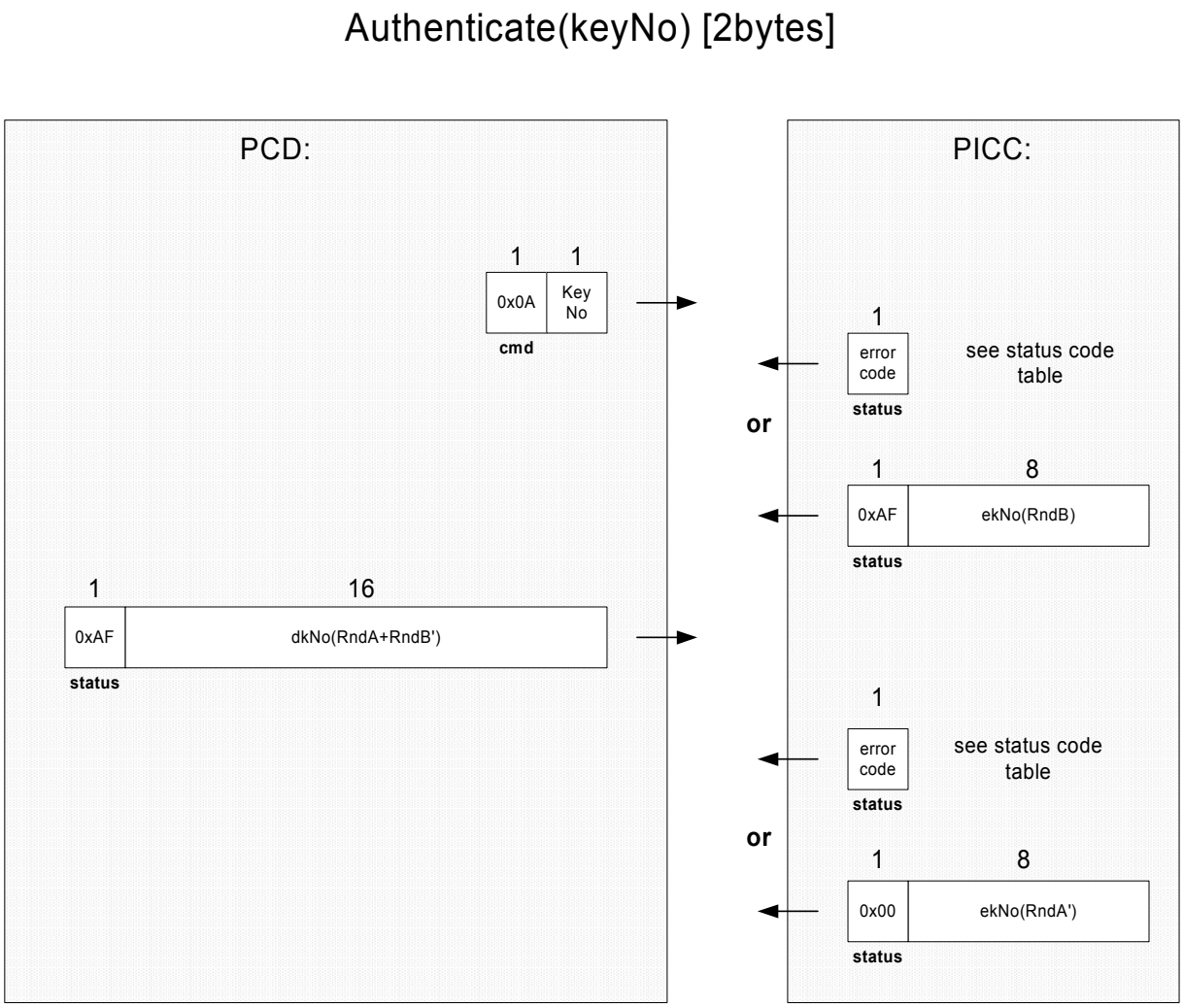

Fig 9. Authenticate (keyNo) [2 bytes]

Depending on the configuration of the application (represented by its AID), an authentication has to be done to perform specific operations:

- Gather information about the application
- Change the keys of the application
- Create and delete files within the application
- Change access rights
- Access data files in the authenticated application

Depending on the security configuration of the PICC, the following commands may require an authentication with the PICC master keys:

- Gather information about the applications on the PICC
- Change the PICC master key itself
- Change the PICC key settings
- Create a new application
- Delete an existing application

The authentication state is invalidated by

- Selecting an application
- Changing the key which was used for reaching the currently valid authentication status
- A failed authentication

The authentication defines the security level of further commands. See Section 6.7 for more details.

Please note: Master keys are identified by their key number 0x00. This is valid on PICC level (this means the selected AID is 0x00) and on Application level (this means the selected AID is not 0x00).

9.3.2 Authenticate DES in ISO CBC send mode [2 bytes]

This command can be used with DES, 3DES and 3K3DES keys.

In this procedure both, the PICC as well as the reader device, show in an encrypted way that they posses the same secret which especially means the same key. This procedure not only confirms that both entities can trust each other but also generates a session key which can be used to keep the further communication path secure. As the name “session key” implicitly indicates, each time a new authentication procedure is successfully completed a new key for further cryptographic operations is obtained.

After a successful authentication with this command the reader needs to encipher data to prepared it for sending and decipher data after receiving.

More information (key length) can be retrieved from Table 1 in Section 7.1

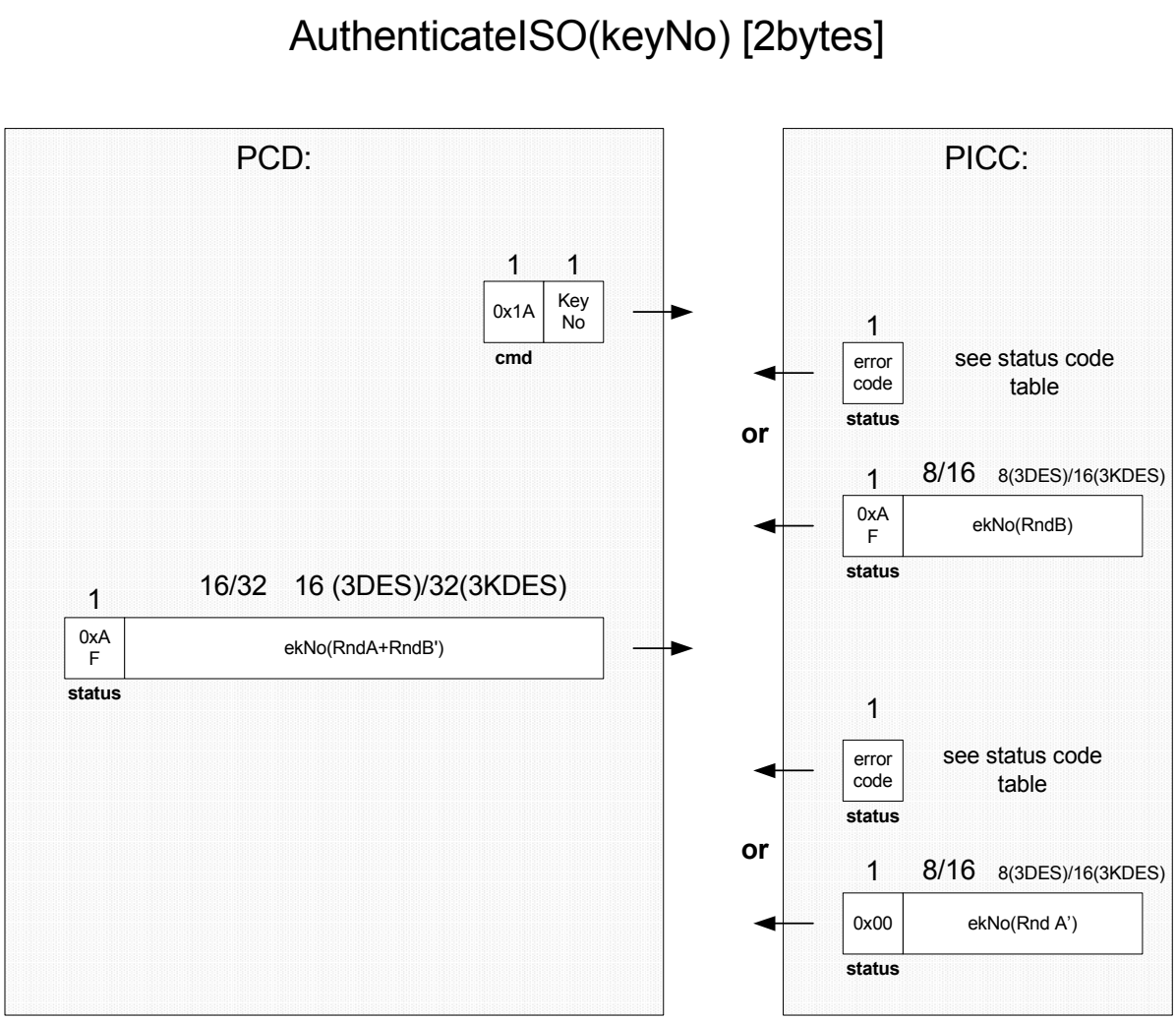

Fig 10. Authenticate DES in ISO CBC send mode [2 bytes]

Depending on the configuration of the application (represented by its AID), an authentication has to be done to perform specific operations:

- Gather information about the application
- Change the keys of the application
- Create and delete files within the application
- Change access rights
- Access data files in the authenticated application

Depending on the security configuration of the PICC, the following commands may require an authentication with the PICC master keys:

- Gather information about the applications on the PICC
- Change the PICC master key itself
- Change the PICC key settings
- Create a new application
- Delete an existing application

The authentication state is invalidated by

- Selecting an application
- Changing the key which was used for reaching the currently valid authentication status
- A failed authentication

Please note: Master keys are identified by their key number 0x00. This is valid on PICC level (this means the selected AID is 0x00) and on Application level (selected AID is not 0x00).

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

9.3.3 Authenticate AES (keyNo) [2 bytes]

In this procedure both, the PICC as well as the reader device, show in an encrypted way that they posses the same secret which especially means the same key. This procedure not only confirms that both entities can trust each other but also generates a session key which can be used to keep the further communication path secure. As the name “session key” implicitly indicates, each time a new authentication procedure is successfully completed a new key for further cryptographic operations is obtained.

After a successful authentication with this command the reader needs to encipher data to prepared it for sending and decipher data after receiving.

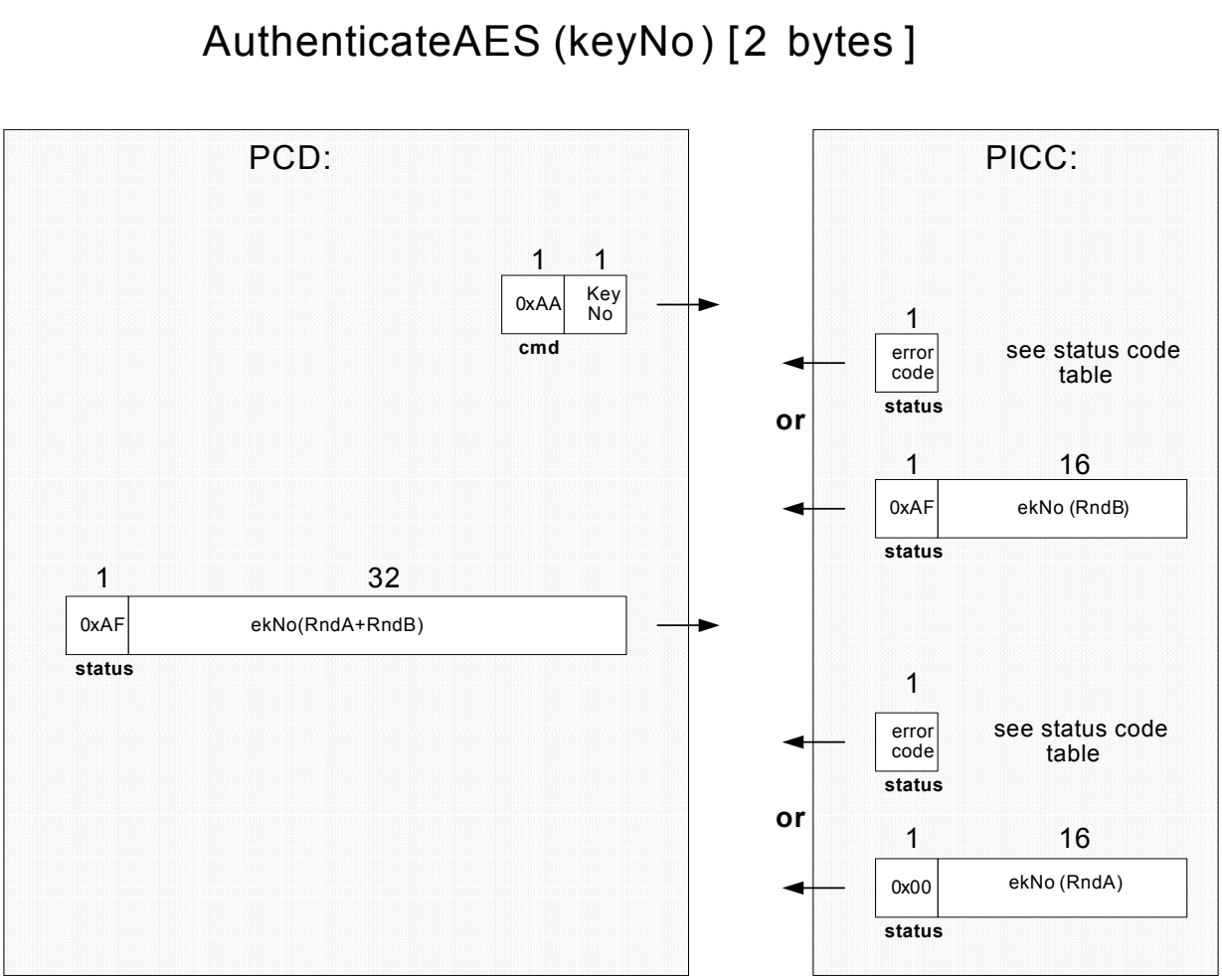

Fig 11. Authenticate AES(keyNo) [2 bytes]

Depending on the configuration of the application (represented by its AID), an authentication has to be done to perform specific operations:

- Gather information about the application
- Change the keys of the application
- Create and delete files within the application
- Change access rights
- Access data files in the authenticated application

Depending on the security configuration of the PICC, the following commands may require an authentication with the PICC master keys:

- Gather information about the applications on the PICC
- Change the PICC master key itself
- Change the PICC key settings
- Create a new application
- Delete an existing application

The authentication state is invalidated by

- Selecting an application
- Changing the key which was used for reaching the currently valid authentication status
- A failed authentication

Please note: Master keys are identified by their key number 0x00. This is valid on PICC level (this means the selected AID is 0x00) and on Application level (this means the selected AID is not 0x00). The authentication defines the security level of further commands. See Section 6.7 for more details.

9.3.4 ChangeKeySettings (KeySettings) [9/17 bytes]

This command changes the master key configuration settings depending on the currently selected AID. If the AID '0x00' has been selected in advance, the change applies to the PICC key settings, otherwise (the selected AID is not 0x00) it applies to the application key settings of the currently selected application.

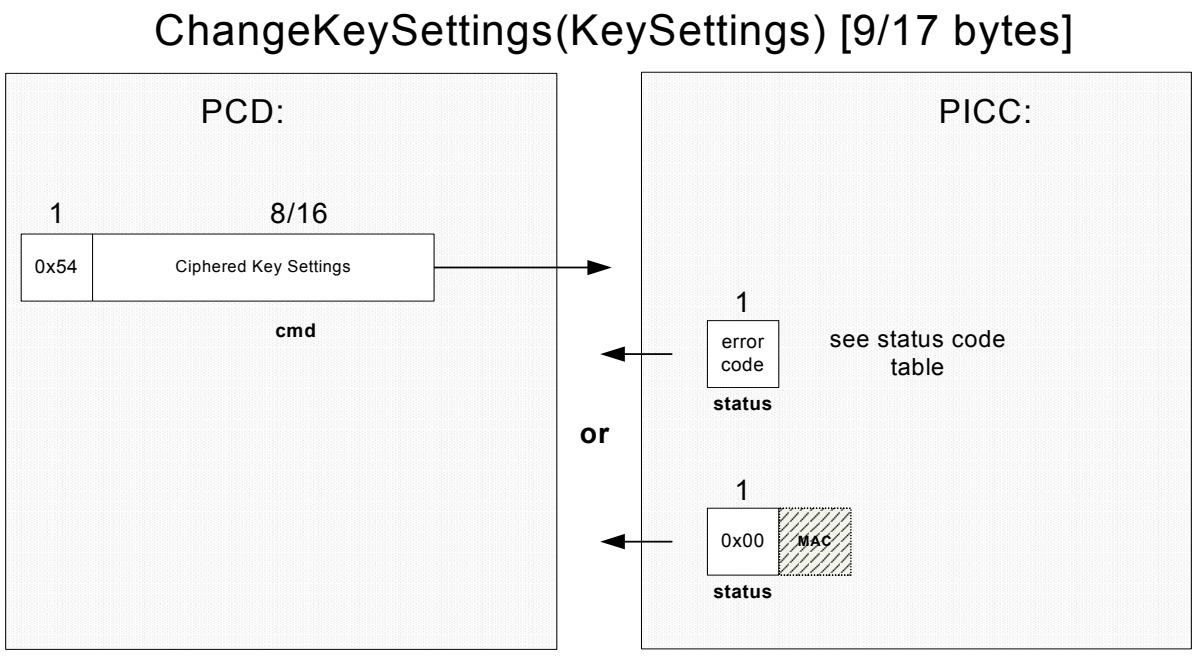

Fig 12. ChangeKeySettings(KeySettings) [9/17 bytes]

This command only succeeds if the “configuration changeable” bit, see below, of the current key settings was not cleared before.

Additionally a successful preceding authentication with the master key is required (PICC master key if AID = 0x00, else with application master key). This command takes one byte as parameter which codes the new master key settings.

To guarantee that the ChangeKeySettings command is sent by the same PCD which did the preceding Authentication command, it is necessary to apply the same security mechanism as for the ChangeKey command, see Section 9.3.6.

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

PICC Master Key Settings:

Table 25. PICC Master Key Settings

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RFU	RFU	RFU	RFU	configuration changeable	PICC master key not required for create / delete	free directory list access without PICC master key	allow changing the PICC master key

On PICC Level (selected AID = 0x00) the coding is interpreted as:

Bit7-Bit 4: RFU, has to be set to 0.

Bit3: codes whether a change of the PICC master key settings is allowed:

- 0: configuration not changeable anymore (frozen).
- 1: this configuration is changeable if authenticated with PICC master key (default setting).

Bit2: codes whether PICC master key authentication is needed before Create- / DeleteApplication

- 0: CreateApplication / DeleteApplication is permitted only with PICC master key authentication.
- 1: CreateApplication is permitted without PICC master key authentication (default setting). DeleteApplication requires an authentication with PICC master key or application master key.[1]

Bit1: codes whether PICC master key authentication is needed for application directory access:

- 0: Successful PICC master key authentication is required for executing the GetApplicationIDs and GetKeySettings commands.
- 1: GetApplicationIDs, GetDFNames and GetKeySettings commands succeed independently of a preceding PICC master key authentication (default setting).

Bit0: codes whether the PICC master key is changeable:

- 0: PICC Master key is not changeable anymore (frozen).
- 1: PICC Master key is changeable (authentication with the current PICC master key necessary, default setting).

1. In case of usage of the application master key for deletion, the application which is about to be deleted needs to be Selected and Authenticated with the application master key prior to the DeleteApplication command.

Application Master Key Settings:

Table 26. Application master key settings

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ChangeKey Access Rights Bit3	ChangeKey Access Rights Bit2	ChangeKey Access Rights Bit1	ChangeKey Access Rights Bit0	configuration changeable	free create / delete without master key	free directory list access without master key	allow change master key

On Application Level (selected AID is not 0x00) the coding is interpreted as:

Bit7-Bit4: hold the Access Rights for changing application keys (ChangeKey command).

- 0x0: Application master key authentication is necessary to change any key (default).
- 0x1.. 0xD: Authentication with the specified key is necessary to change any key. A change key and a PICC master key can only be changed after authentication with the master key. For keys other than the master or change key, an authentication with the same key is needed.
- 0xE: Authentication with the key to be changed (same KeyNo) is necessary to change a key.
- 0xF: All Keys (except application master key, see Bit0) within this application are frozen.

Bit3: codes whether a change of the application master key settings is allowed:

- 0: configuration not changeable anymore (frozen).
- 1: this configuration is changeable if authenticated with the application master key (default setting).

Bit2: codes whether application master key authentication is needed before CreateFile / DeleteFile

- 0: CreateFile / DeleteFile is permitted only with application master key authentication.
- 1: CreateFile / DeleteFile is permitted also without application master key authentication.

Bit1: codes whether application master key authentication is needed for file directory access:

- 0: Successful application master key authentication is required for executing the GetFileIDs GetFileSettings and GetKeySettings commands.
- 1: GetFileIDs, GetISOFileIDs, GetFileSettings and GetKeySettings commands succeed independently of a preceding application master key authentication (default setting).

Bit0: codes whether the application master key is changeable:

- 0: Application master key is not changeable anymore (frozen).
- 1: Application master key is changeable (authentication with the current application master key necessary, default setting).

Remark: A change key and a PICC master key can only be changed after authentication with the master key (see Section 9.3.6).

9.3.5 GetKeySettings() [1 byte]

The GetKeySettings() command allows to get configuration information on the PICC and application master key configuration settings as described in Section 9.3.4. In addition it returns the maximum number of keys which can be stored within the selected application.

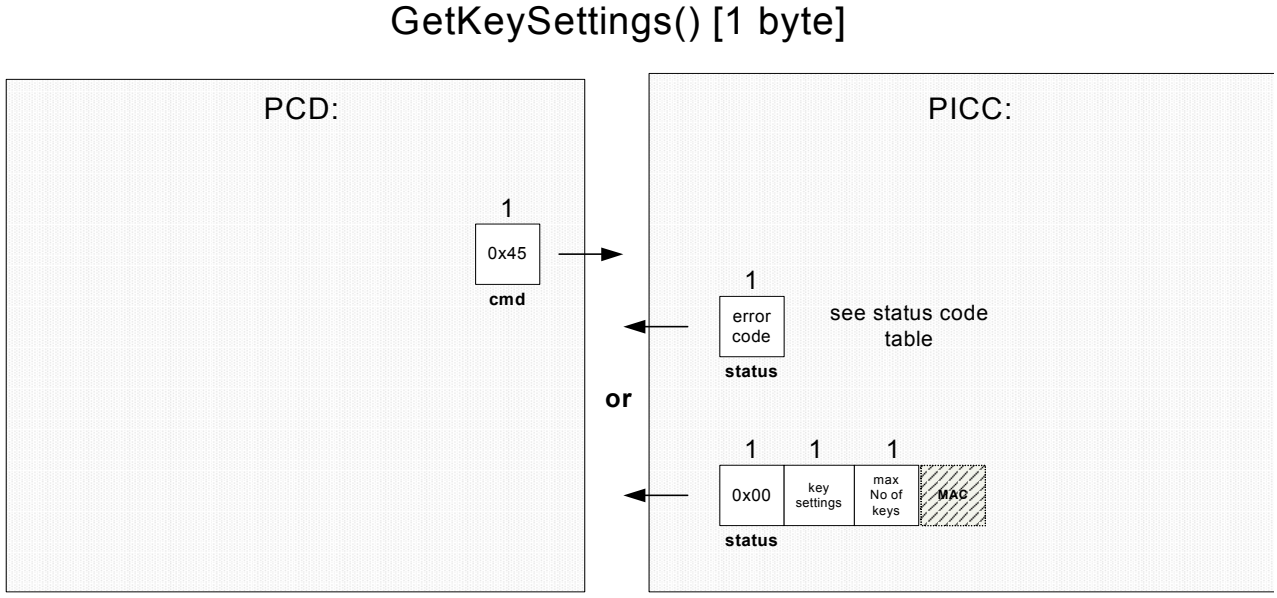

Fig 13. GetKeySettings() [1 byte]

No parameter is passed with this command.

Depending on the master key settings (see Section 9.3.4), a preceding authentication with the master key is required.

If the PICC master key settings are queried (currently selected AID = 0x00), the number of keys is returned as 0x01, as only one PICC master key exists on a PICC.

Within the byte 'max No of keys', also the key type is indicated:

This command allows to read out the keysettings stored on the PICC.The 2 MS-bits specify if the newly written key should be (3)DES, 3K3DES or AES:

- '00' specifies (3)DES operation of PICC master key
- '01' specifies 3K3DES operation of PICC master key
- '10'specifies AES operation of PICC master key

For keys on card level (master key) the key type is not reflected in the 2 MS-bits.

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

9.3.6 ChangeKey (KeyNo) [26-42 bytes]

This command allows to change any key stored on the PICC.

If the AID ’0x00’ is selected (PICC level), the change applies to the PICC master key. As only one PICC master key is stored on DESFire, the lower 6 bits of KeyNo have to be set to ‘0’. The 2 MS-bits specify if the newly written key should be (3)DES, 3K3DES or AES:

- ‘00’ specifies (3)DES operation of PICC master key
- ‘01’ specifies 3K3DES operation of PICC master key
- ‘10’specifies AES operation of PICC master key

In all other cases (the selected AID is not ’0x00’) the change applies to the specified KeyNo within the currently selected application (represented by it’s AID). On Application level (the selected AID is not ’0x00’) it is not possible to change key type after application creation, see Section 9.3.1.

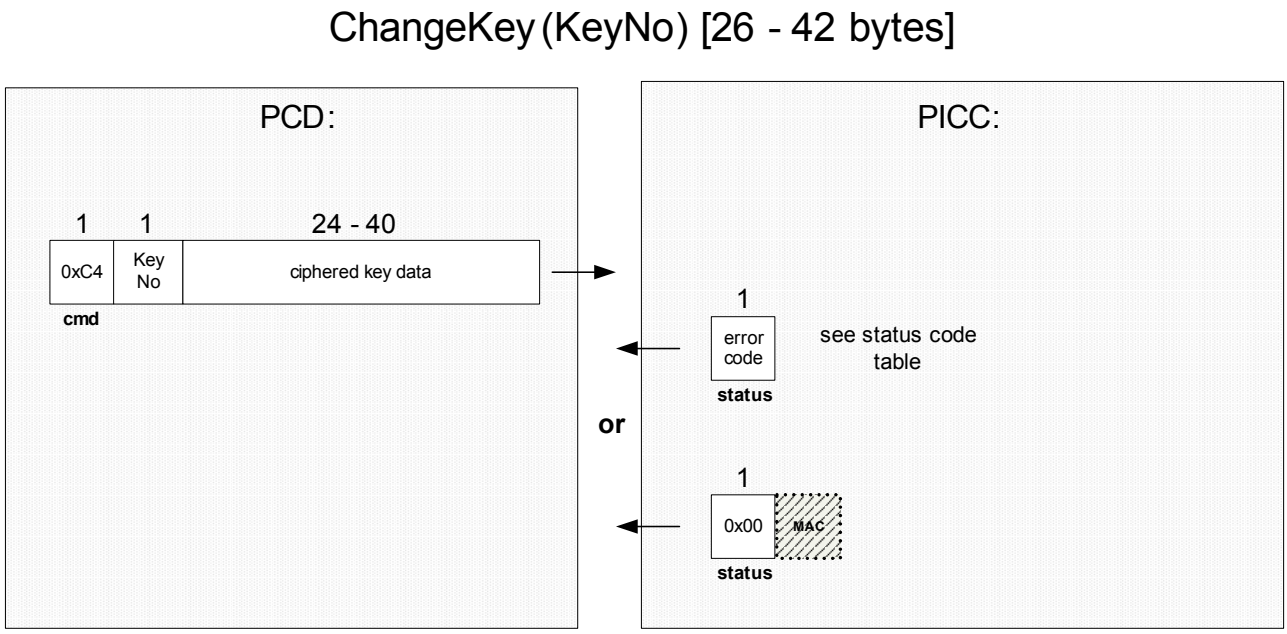

Fig 14. ChangeKey(KeyNo) [26- 42 bytes]

As a parameter this command takes the KeyNo which is of one byte length and has to be in the range from 0x00 to number of application keys - 1.

The ciphered key data include the following:

- the key (can be XORed with the previous section, see next paragraphs)
- the key version (1 byte), if the key is an AES key.

The respective key settings (see Section 9.3.6) define whether a change of keys is permitted or not, additionally they show which key is needed for Authentication before the ChangeKey Command, but are not included in this command, the settings can be changed with the ChangeKeySettings command.

To change any key (except Master Key and the ChangeKey Key), authentication with the ChangeKey is necessary.

- To change the ChangeKey Key or the Master Key, authentication with the Master Key is necessary.
- In case the KeyNo used for authentication is DIFFERENT from the KeyNo to be changed and ChangeKey Key is set to a value 0xE, the PCD needs to generate the data frame “deciphered key data” in the following way:

Authentication with command authenticate() (command code 0x0A) was performed:

CASE I: KeyNo used for authentication is DIFFERENT from KeyNo to be changed and the Change Key Key is set to value 0x0E, the PCD needs to generate the data frame ’deciphered key data’ in the following way:

- The new key and the current key are bit-wise XORed (16/24 byte).
- A CRC16 is calculated over the XORed data and appended at the end.
- Additionally a CRC16 of the new key is appended.
- Padding (see Section 7.2.4) is applied to reach an adequate frame size of multiples of 8 (or 16 (AES)).
- Finally a DES/3DES enciphering operation (Using the current session key) is performed on the whole key data field.
- The cryptogram blocks are all chained using the CBC send mode.

CASE II: KeyNo used for authentication is the same as the KeyNo to be changed or if ChangeKey Key access condition is set to 0x0E, the PCD needs to generate the data from ’enciphered key data’ in the following way:

- A CRC16 is calculated over the new key data and appended at the end.
- After padding of zeros is applied to reach an adequate frame size of multiples of 8.
- Finally an enciphering operation using the current session key is performed on the whole key data field.
- The cryptogram blocks are chained using the CBC send mode.

CASE III: The ChangeKey Key is set to 0x0F (’never’), all keys except the Master Key (see Section 9.3.4, Bit0) are frozen. The ChangeKey command therefore will return an error code when attempting to change a key different from the master key.

Other authentication (0x1A and 0xAA) was performed:

CASE I: KeyNo used for authentication is DIFFERENT from KeyNo to be changed and the Change Key Key is set to value 0x0E, the PCD needs to generate the data frame ’deciphered key data’ in the following way:

- The new key and the current key are bit-wise XORed (16/24 byte).
- For an AES key, the key version (1 byte) is attached.
- A CRC32 is calculated over the XORed data and appended at the end.
- Additionally a CRC32 of the new key is appended.
- Padding (see Section 7.3.6) is applied to reach an adequate frame size of multiples of 8 (or 16 (AES)).

- Finally a DES/3DES/3KDES/AES enciphering operation (Using the current session key) is performed on the whole key data field.
- The cryptogram blocks are all chained using the CBC send mode.

CASE II: KeyNo used for authentication is the same as the KeyNo to be changed or if ChangeKey Key access condition is set to 0x0E, the PCD needs to generate the data fram 'enciphered key data' in the following way:

- For an AES key, the key version (1 byte) is attached.
- A CRC32 is calculated over the new key data and appended at the end.
- After padding of zeros is applied to reach an adequate frame size of multiples of 8 (3DES, 3KDES) or 16 (AES).
- Finally an enciphering operation using the current session key is performed on the whole key data field.
- The cryptogram blocks are chained using the CBC send mode.

CASE III: The ChangeKey Key is set to 0x0F ('never'), all keys except the Master Key (see Section 9.3.4, Bit0) are frozen. The ChangeKey command therefore will return an error code when attempting to change a key different from the master key.

Remark: After a successful change of the key used to reach the current authentication status, this authentication is invalidated i.e. an authentication with the new key is necessary for subsequent operations.

9.3.7 GetKeyVersion (KeyNo) [2 bytes]

The GetKeyVersion command allows to read out the current key version of any key stored on the PICC.

If AID = 0x00 is selected, the command returns the version of the PICC master key and therefore only KeyNo = 0x00 is valid (only one PICC master key is present on a PICC). In all other cases (the AID is not '0x00') the version of the specified KeyNo within the currently selected application (represented by it's AID) is returned.

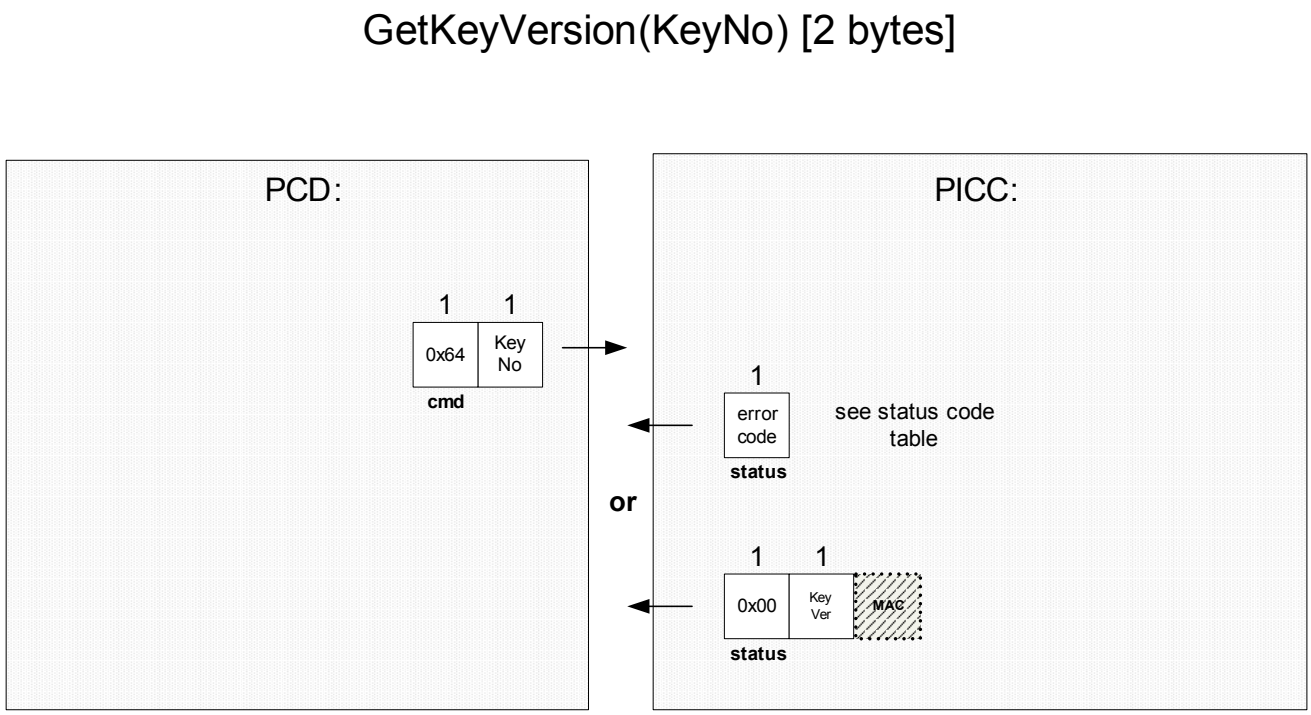

Fig 15. GetKeyVersion (KeyNo) [2 bytes]

One parameter is passed with this command which codes the key number.

The command returns the current version of the specified key as an unsigned byte. For DES keys the version is coded as described in Section 8.2.

To change the key version of any key, the ChangeKey command, see Section 9.3.6, is used.This command can be issued without valid authentication.

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

9.4 MF3ICD81 command set – PICC level commands

9.4.1 CreateApplication [6 (...22) bytes]

The CreateApplication command allows to create new applications on the PICC.

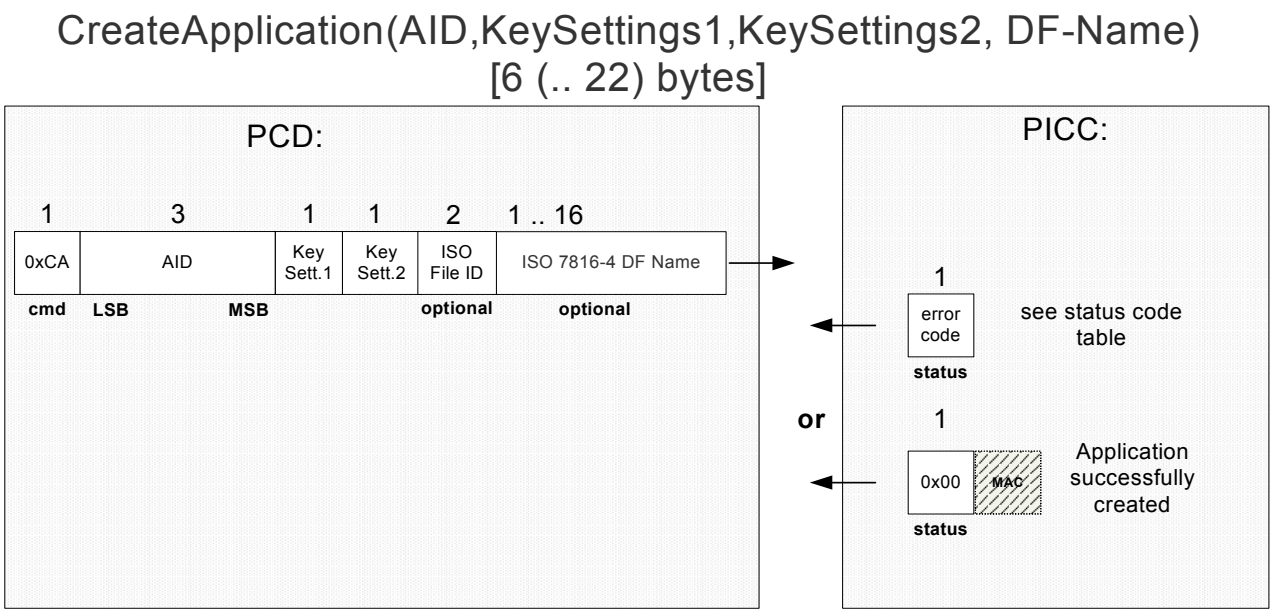

Fig 16. CreateApplication [6 (...22) bytes]

An application is identified by an 'Application Identifier', AID, which is implemented as a 24 bit number. Application Identifier 0x00 00 00 is reserved as a reference to the PICC itself.

In 7816-mode, in Addition to the AID an application can be referenced using a "DF-name" using the ISO SELECT, "selection by DF name". The "DF-Name" can be chosen independently from the AID.

Depending on the PICC master key settings, see Section 9.3.4, a preceding PICC master key authentication may be required.This command requires that the currently selected AID is 0x00 00 00 which references the card level.

One PICC can hold up to 28 Applications. Each application is linked to a set of up to 14 different user definable access keys. To store data in an application, it is necessary to create so called files within that application, see Section 9.5.5 to Section 9.5.9. Up to 32 files of different size and type can be created within each application. Different levels of access rights for each single file can be linked to the keys of the application.

The 24 bit AID is the first parameter of the command.

The second parameter is the KeySettings1. This is the 'Application Master Key Settings' as defined in Section 9.3.4.

The next parameter ‘Key Settings 2’ defines several settings:

- Bit 0..3: Number of keys that can be stored within the application for cryptographic purposes. A maximum of 14 keys can be stored within an application of DESFire. One can also create an application with no keys!
- Bit 4: RFU
- Bit 5: Indicates use of 2 byte ISO/IEC 7816-4 File Identifies for files within the Application:
  - ‘0’ NO 2 byte File Identifiers for files within the application supported
  - ‘1’ 2 byte File Identifiers for files within the application supported
- Bit 6..7: Indicates the crypto method of the application:
  - ‘00’ specifies DES and 3DES operation for the whole application
  - ‘01’ specifies 3K3DES operation for the whole application
  - ‘10’ specifies AES operation for the whole application

One can use “00” for authentication either with “0x0A” or “0x1A”. With the binary value “01” only authentication with “0x1A” is possible, the binary value “10” can only go together with authentication with “0xAA”.

Before any setup of a file system, it is recommended to configure the whole card using the command ‘SetConfiguration’ see Section 9.4.9. This command will initialize all keys of any created application to a specified value which is taken out of the default key and default version from the ’SetConfiguration’ command. Without this command all keys are consisting of 0x00 bytes. This step shall be taken during initialisation at card manufacturing latest. In case of using DES and 3DES all keys are single DES keys by definition after initialisation, see Section 8.2.

ISOFile IDs are used for ISO/IEC 7816-4 file systems. The ISOFile ID parameter will be used to select the application with the ISO SELECT command, file option. The parameter is optional.

In bit 5 of KeySettings2 one defines if for every EF (elementary file) within the application a ISOFile ID is used.

The last parameter is optional and carries the “DF-name” used in 7816-4 mode in combination with the ISO SELECT command. The length of this parameter can be from 1 to 16 bytes.

In case no “DF-name” is specified during creation of the application, the application can NOT be referenced by “DF-name” later on.

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

9.4.2 DeleteApplication(AID) [4 bytes]

The DeleteApplication command allows to permanently deactivate applications on the PICC.

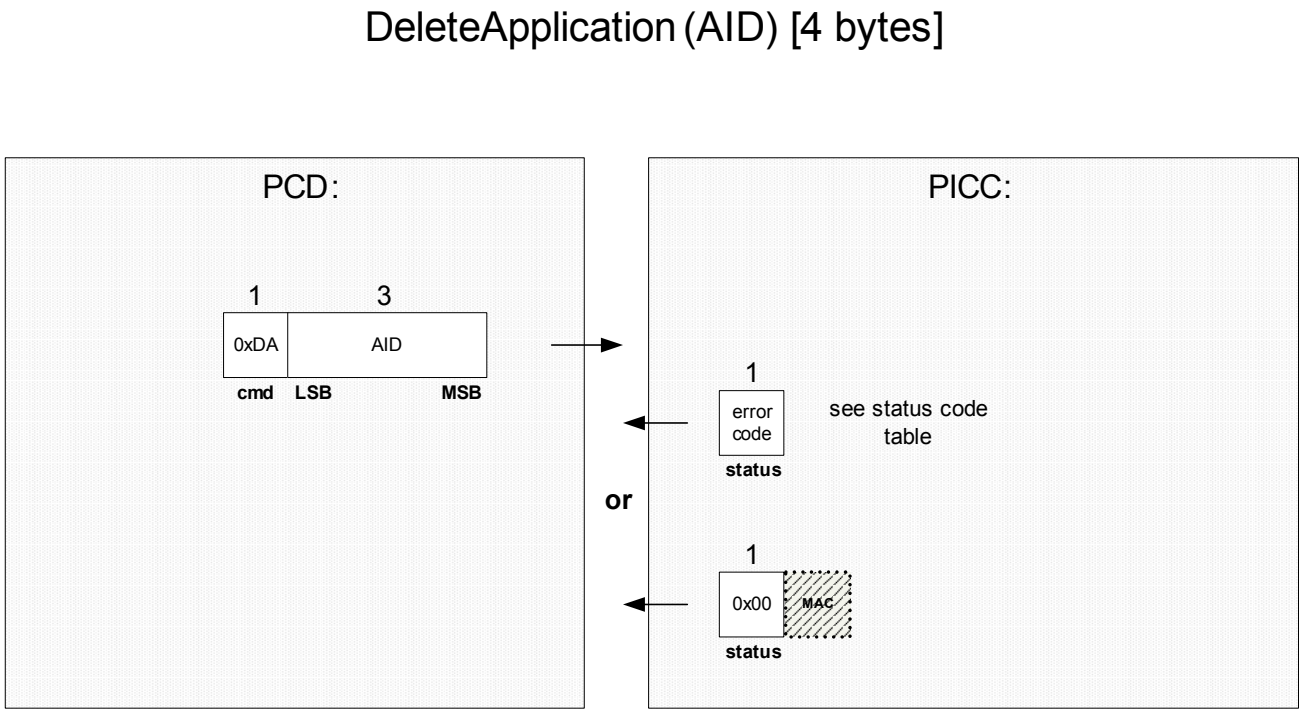

Fig 17. DeleteApplication (AID) [4 bytes]

The application which will be deleted is represented by it's Application Identifier, AID, which is the only parameter of this command.

Depending on the PICC master key settings, see Section 9.3.4, either a preceding PICC master key authentication or an application master key authentication is required.

In the latter case, master key authentication has to be performed prior to the deletion.

The AID allocation is removed, therefore it is possible to create a new application with the deleted application's AID. All keys are overwritten with random values. However, the deleted memory blocks can only be recovered by using the FormatPICC command (see Section 9.4.6) which erases the full user memory of the PICC.

Remark: Even if the PICC master key contains the default value 0 and the bit "free create / delete without PICC master key" is set, it is necessary to be either authenticated with the zero PICC master key or the respective application master key.

Remark: If the currently selected application is deleted, this command automatically selects the PICC level, selected AID = 0x00 00 00.

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

9.4.3 GetApplicationIDs () [1 byte]

The GetApplicationIDs command returns the Application IDentifiers of all active applications on a PICC.

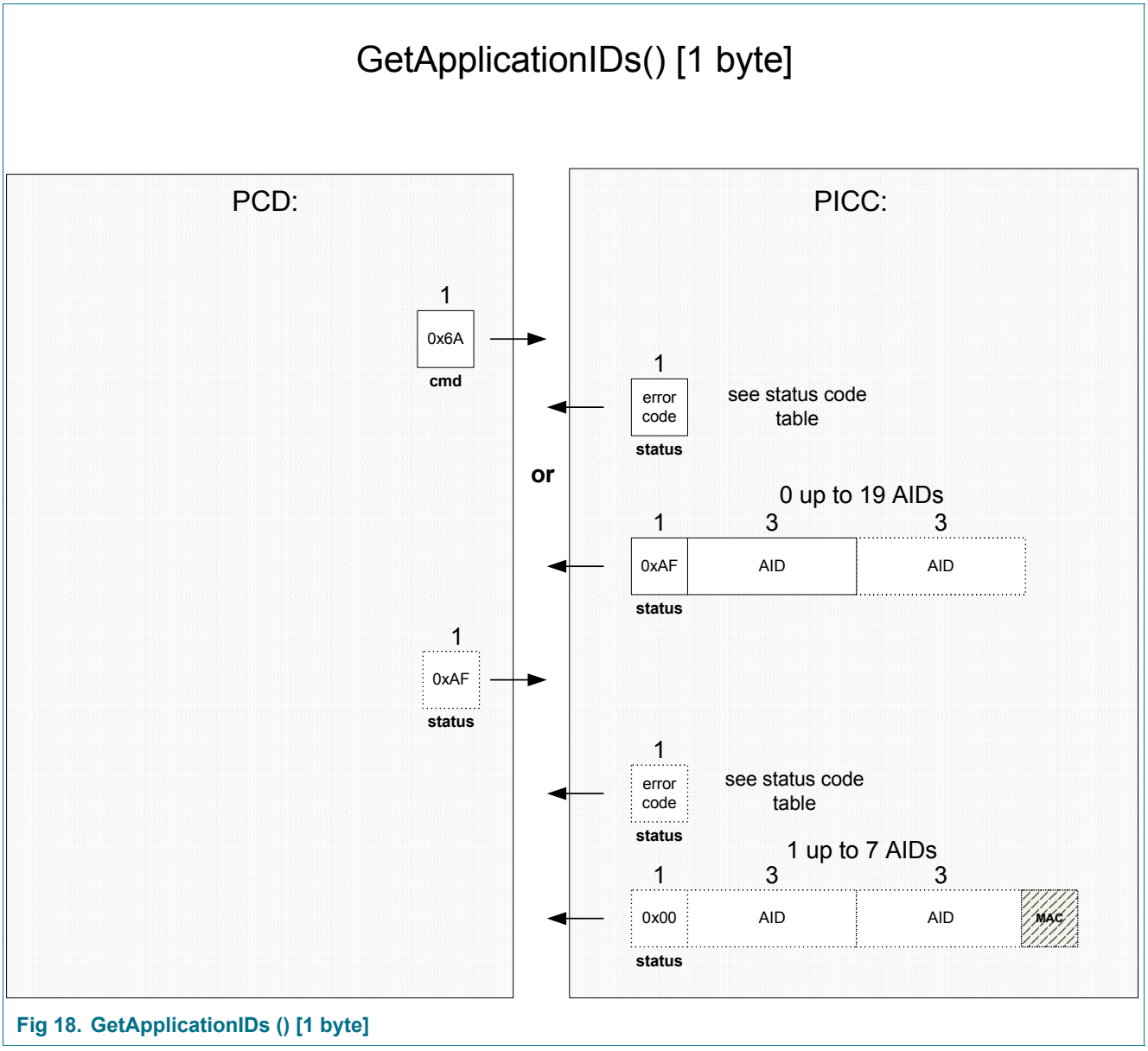

Fig 18. GetApplicationIDs () [1 byte]

This command does not accept any parameters.

Depending on the PICC master key settings (see Section 9.3.4) a successful authentication with the PICC master key might be required to execute this command.

This command requires that the currently selected AID is 0x00 00 00 which references the card level. If no Application ID is available, then an error code is returned.

As response the PICC sends a sequence of all installed AIDs. If this sequence does not fit into one single frame, an additional frame is set by the PICC, indicated by a 0xAF in the status byte of all frames which will be continued.

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

9.4.4 GetDFNames () [1 byte]

The GetDFNames command returns the ISO/IEC 7816-4 DF-Names of all active applications on a PICC.

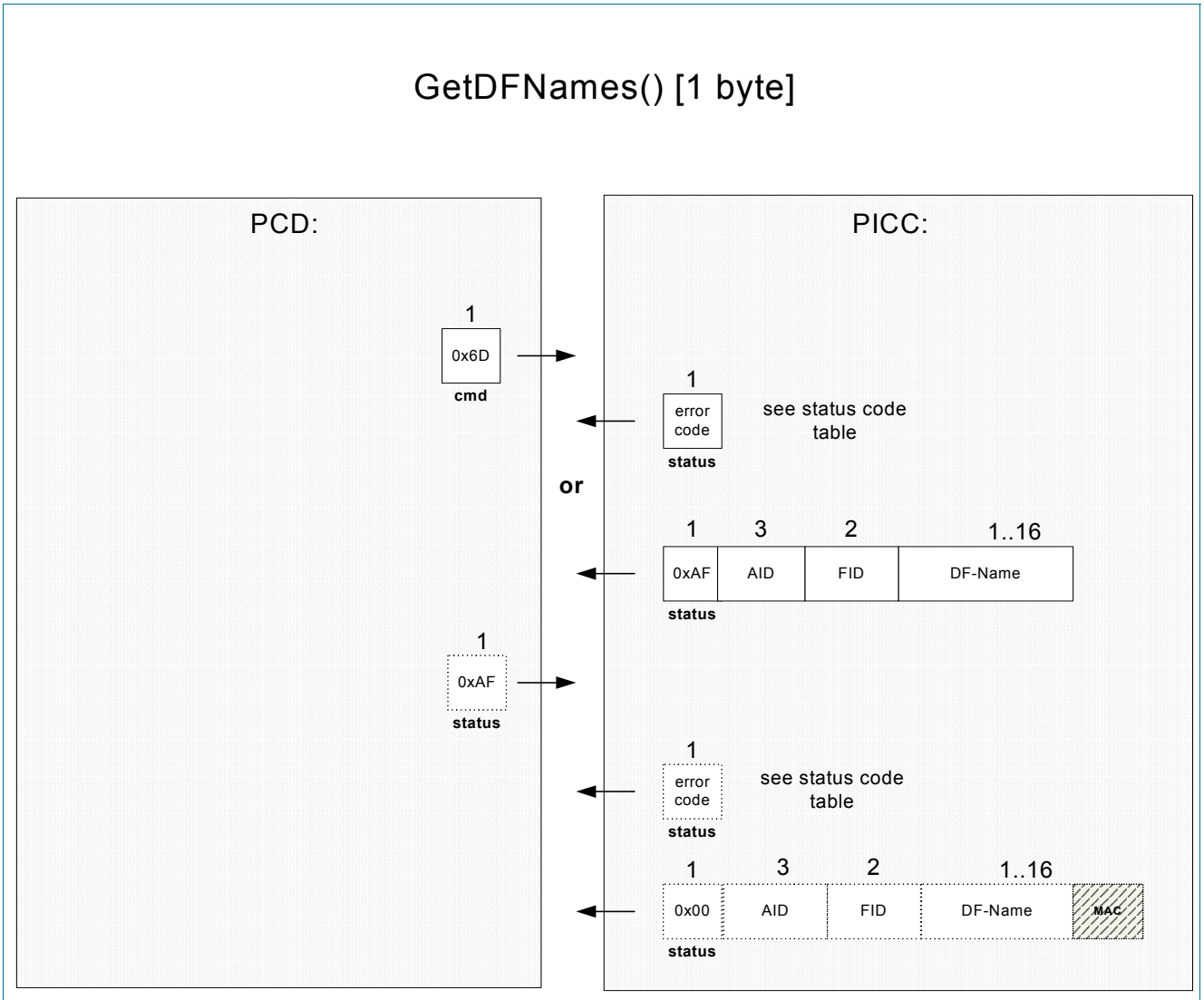

Fig 19. GetDFNames () [1 byte]

This command does not accept any parameters.

Depending on the PICC master key settings (see Section 9.3.4) a successful authentication with the PICC master key might be required to execute this command.

This command requires that the currently selected AID is 0x00 00 00 which references the card level.

As response the PICC sends a sequence of all installed DFs, represented by their DF-Name. Each DF-Name is sent in a separated frame. In case the sequence does not fit into one single frame, an additional frame is set by the PICC, indicated by a 0xAF in the status byte of all frames which will be continued.

If the DC has no DF name it is not sent back within this command.

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

9.4.5 SelectApplication (AID) [4 bytes]

The SelectApplication command allows to select one specific application for further access.

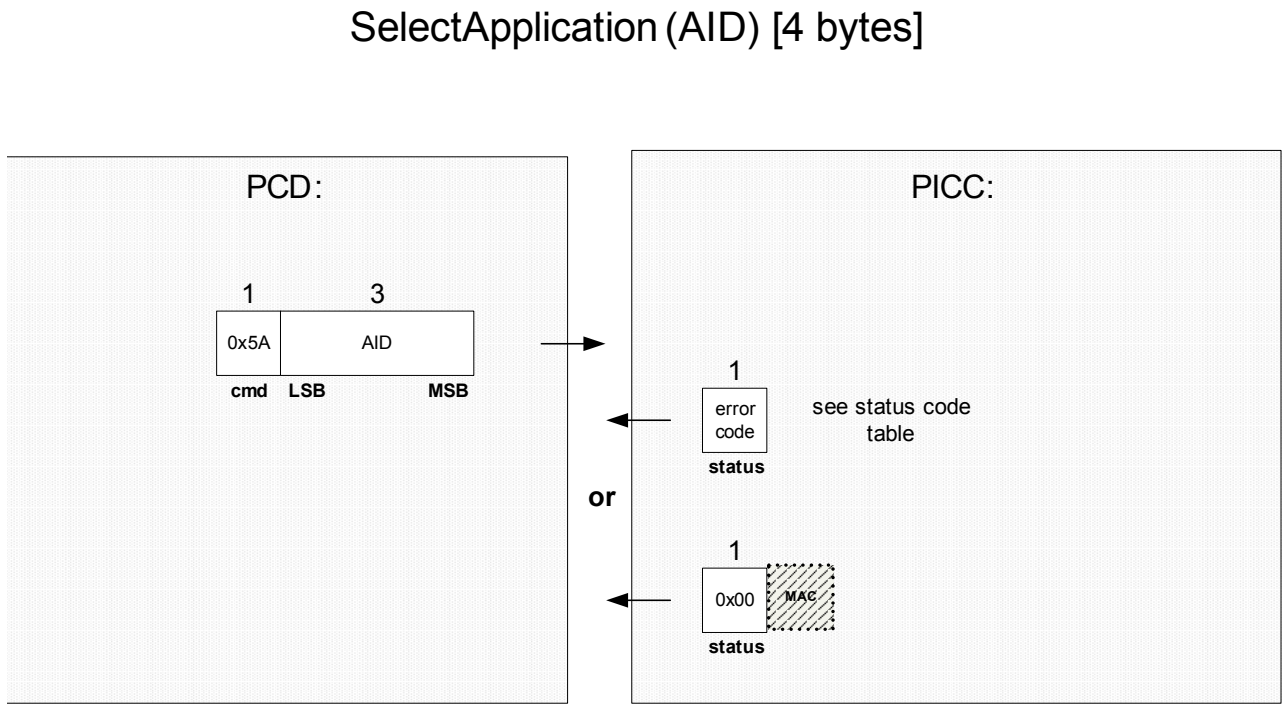

Fig 20. SelectApplication (AID) [4 bytes]

As parameter this command takes three bytes coding the AID.

If this parameter is 0x00 00 00, the PICC level is selected and any further operations (typically commands like CreateApplication, DeleteApplication…) are related to this level.

If an application with the specified AID is found in the application directory of the PICC, the subsequent commands interact with this application.

As mentioned in the description of the authenticate command (see Section 9.3.1), each SelectApplication command invalidates the current authentication status.

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

9.4.6 FormatPICC () [1 byte]

This command releases the PICC user memory.

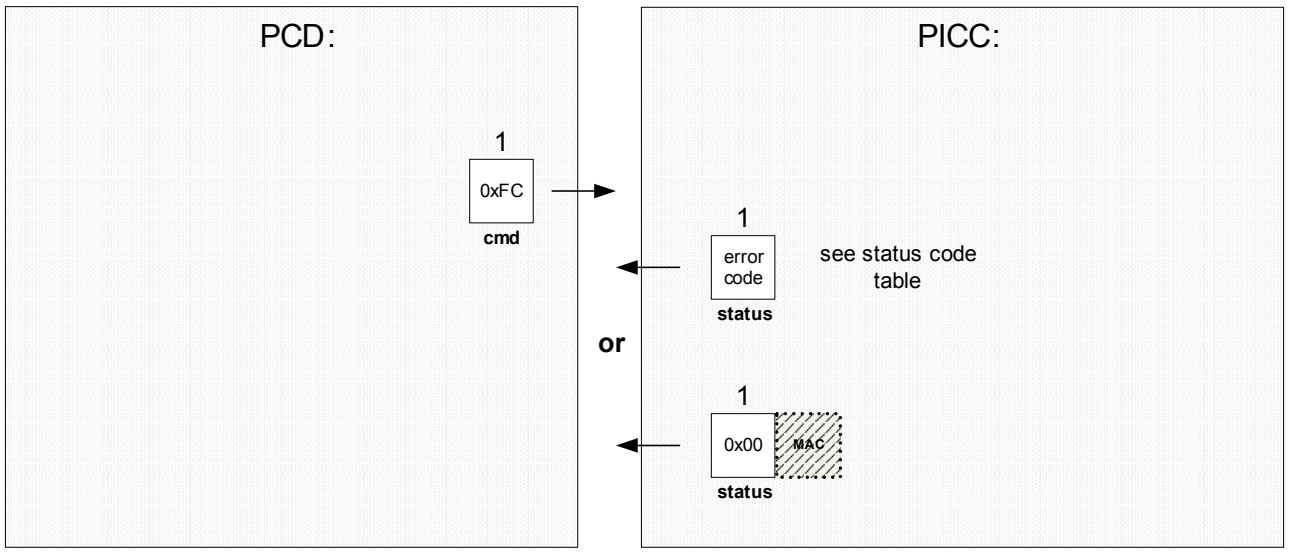

Fig 21. FormatPICC () [1 byte]

No parameters are passed with this command. The FormatPICC Command releases all allocated user memory on the PICC. All applications are deleted and all files within those applications are deleted. The memory is deleted.

Remark: This command cannot be rolled back.

The PICC master key and the PICC master key settings keep their currently set values, they are not influenced by this command. The command can be disabled using SetConfiguration(). This command always requires a preceding authentication with the PICC master key.

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

9.4.7 GetVersion [1 byte]

The GetVersion command returns manufacturing related data of the PICC.

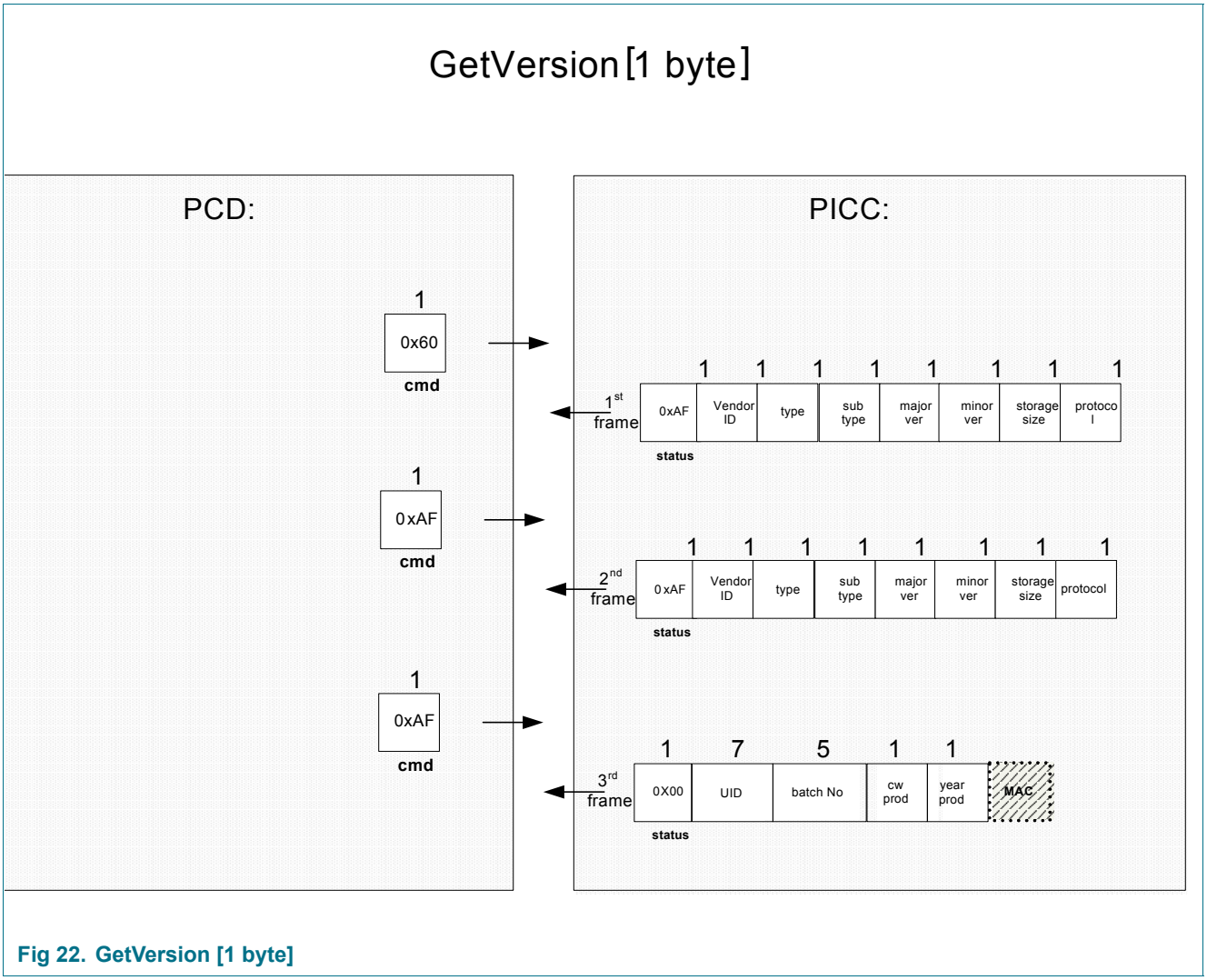

Fig 22. GetVersion [1 byte]

Three frames of manufacturing related data are returned by the PICC:

The 1st frame: contains hardware related information:

byte 1: codes the vendor ID (0x04 for NXP)

byte 2: codes the type (here 0x01)

byte 3: codes the subtype (here 0x01)

byte 4: codes the major version number 01

byte 5: codes the minor version number 00

byte 6: codes the storage size* (here 0x1A = 8192)

byte 7: codes the communication protocol type
(here 0x05 meaning ISO/IEC 14443-2 and -3)

The 2nd frame contains software related information:

byte 1: codes the vendor ID (here 0x04 for NXP)

byte 2: codes the type (here 0x01)

byte 3: codes the subtype (here 0x01)

byte 4: codes the major version 01

byte 5: codes the minor version 03

byte 6: codes the storage size[2] (here 0x1A = 8192 bytes)

byte 7: codes the communication protocol type
(here 0x05 meaning ISO/IEC 14443-3 and -4)

The 3rd frame returns the unique serial number, batch number, year and calendar week of production:

byte 1 to byte 7: code the unique serial number

byte 8 to byte 12: code the production batch number

byte 13: codes the calendar week of production[3]

byte 14: codes the year of production[3].

In case of “Random ID”, see Section 6.5, the UID is sent back as 0x00.

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

2.The 7 MS Bits (= n) code the storage size itself based on 2^n, the Lisped is set to '0' if the size is exactly 2^n and set to '1' if the storage size is between 2^n and 2^(n+1). For this version of DESFire the 7 MSBits are set to 0x0D (2^13 = 8192) and the LSBit is '0'.

3.Calender week 36 in 2006 would be in this case: 0x3606.

9.4.8 FreeMem [1 byte]

The FreeMem command returns the available bytes on the PICC.

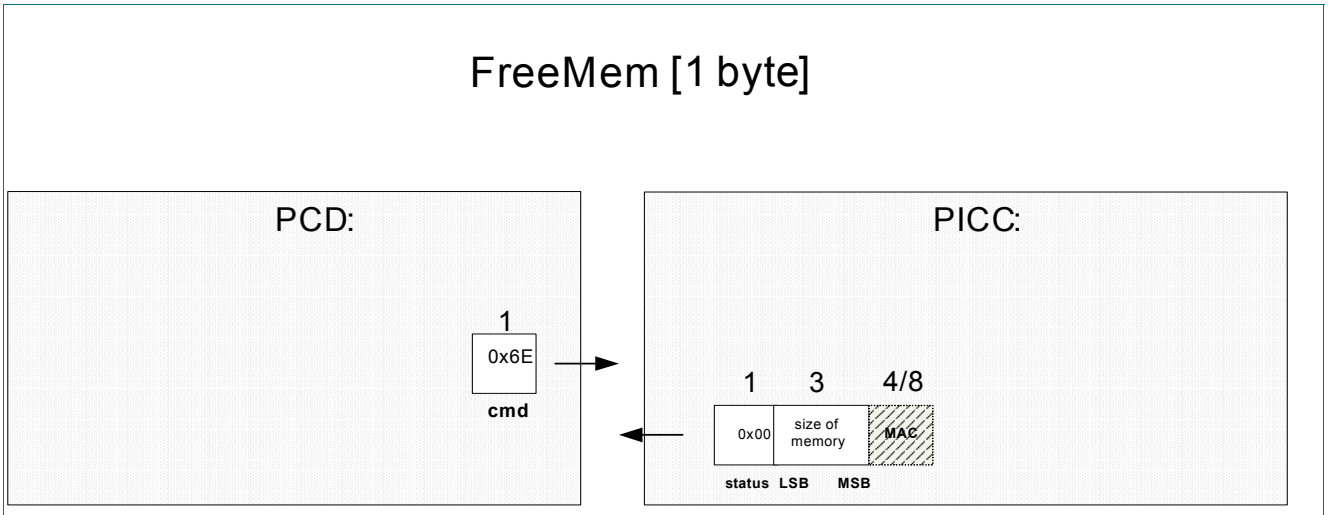

Fig 23. FreeMem [1 byte]

This command takes no parameter. The answer is the size of the free memory, whereas the first byte is the LSB.

Information on structure after authentication with Authenticate 0x0A can be found in Section 7.2.1 for AuthenticateISO 0x1A and Authenticate 0xAA in Section 7.3.1.

9.4.9 SetConfiguration [10 – 34 bytes]

Master key authentication on card level needs to be performed prior to the SetConfiguration command. The command is secured with a 8 byte CMAC.

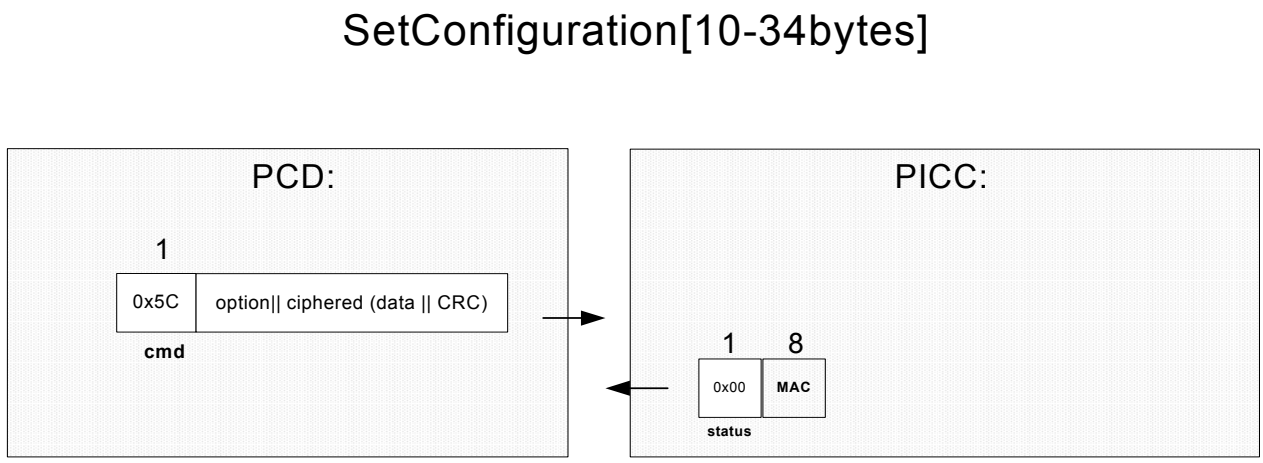

Fig 24. SetConfiguration [10 – 34 bytes]

Option:

- 0x00: data is the configuration byte
- 0x01: data is default key version and default key

  Option= 0x01; all applications will be personalized during creation with this default key and version instead of 0x00
- 0x02: data is the user defined ATS
- 0xXX: RFU

Data:

Case: Option = configuration byte

bit 0= 0 Format card enabled

bit 0= 1 Format card disabled; cannot be reset

bit 1= 0 Random ID disabled

bit 1= 1 Random ID enabled; cannot be reset

Case: Option = 0x01; Key || Key version; 24 bytes key and 1 byte default version; If a key shorter then 24 bytes is used, only the left most bytes will be used.

Case: Option = 0x02; User defined ATS parameter; the ATS string that is returned after RATS excluding the ATS CRC1 and CRC2 bytes. Only the length of the string is checked (20); but no other verification of the ATS string is performed. Therefore the data string must be formatted in the following: TL T0 TA TB TC + Historical bytes.

Standard configuration has Random ID disabled, Formad ID enabled, the ATS according to document “018413 MIFARE interface platform type identification procedure”.

Remark: Option= 0x02 should only be carried out by experts. For this option padding is carried out with ISO/IEC 9797-1 padding method2 before enciphering. For all other options padding is carried out with 0x00. For an ATS longer then 16 bytes, please check the frame size of all used readers, as a reader with frame size of 16 bytes could have problems in the receipt of the ATS.

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

9.4.10 GetCardUID [1 byte]

This command is only applicable if SetConfiguration() was called to use random UIDS.

An authentication with any key needs to be performed prior to the command GetCardUID(). This command returns the UID and gives the opportunity to retrieve the UID, even if the random ID is used.

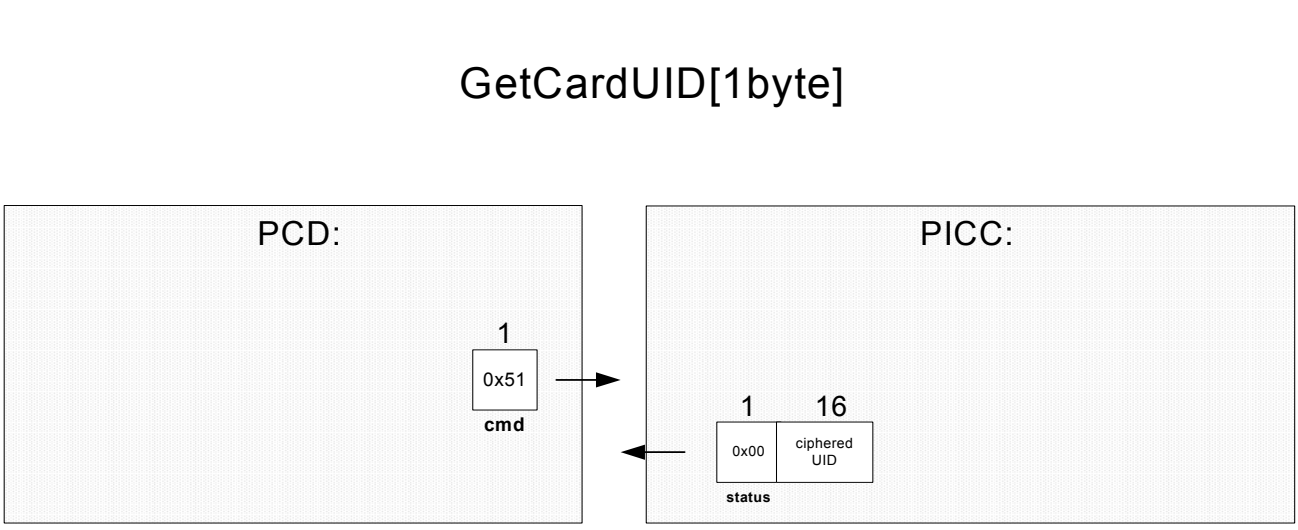

Fig 25. GetCardUID [1 byte]

The command takes no parameter.

The command is encrypted, see Section 7.3.3

9.5 MF3ICD81 command set – application level commands

The MF3ICD81 provides the following command set for Application level functions:

9.5.1 GetFileIDs() [1 byte]

The GetFileIDs command returns the File IDentifiers of all active files within the currently selected application.

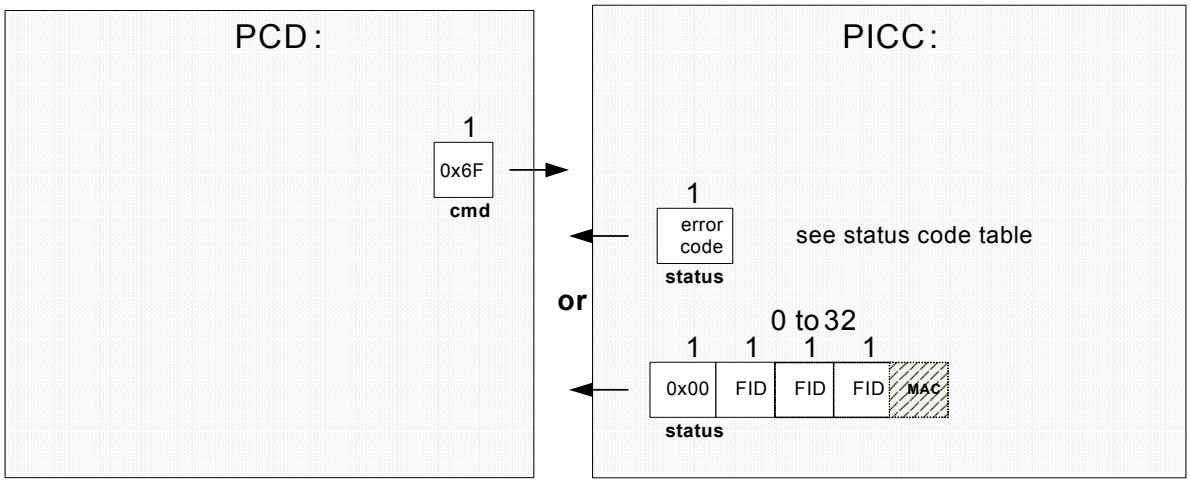

Fig 26. GetFileIDs () [1 byte]

This command takes no parameters.

Depending on the application master key settings (see Section 9.3.4), a preceding authentication with the application master key might be required.

Each File ID is coded in one byte and is in the range from 0x00 to 0x1F.

Duplicate values are not possible as each file must have an unambiguous identifier.

As the number of files is limited to 32 within one application, the response always fits into one single data frame. If there is no file in the application, an error can be returned.

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

9.5.2 GetISOFileIDs () [1 byte]

The GetFISOileIDs command returns the 2 byte ISO/IEC 7816-4 File IDentifiers of all active files within the currently selected application.

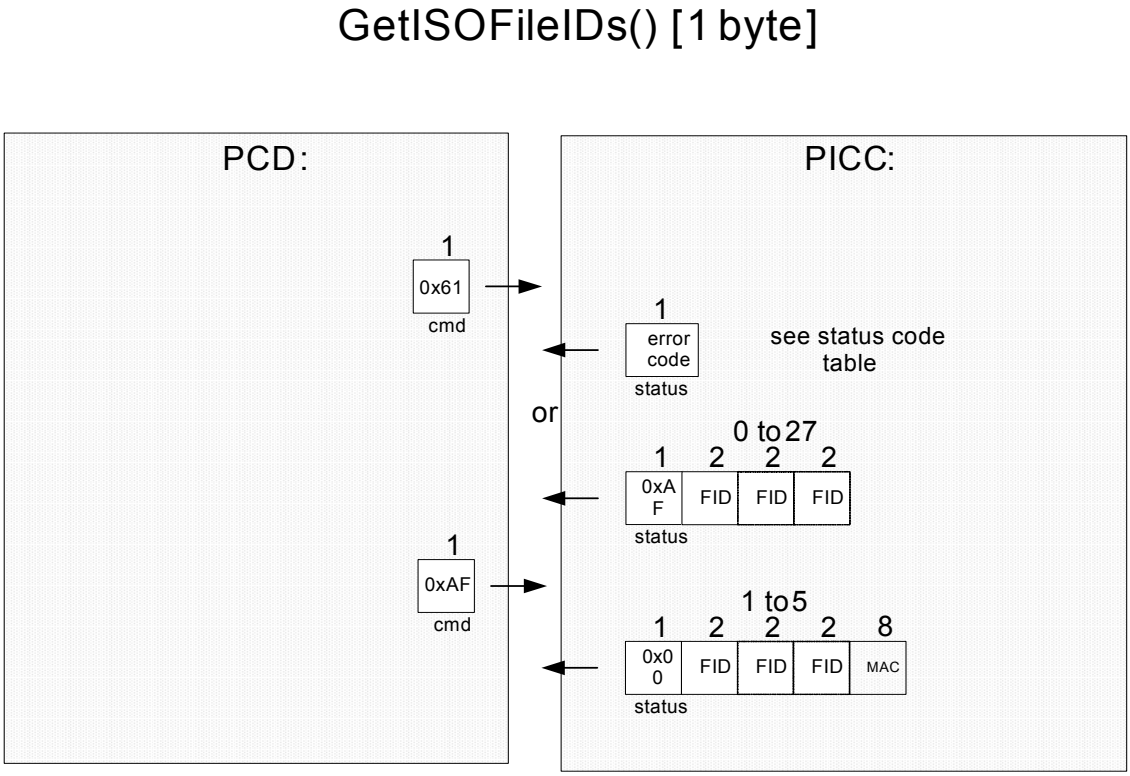

Fig 27. GetISOFileIDs () [1 byte]

This command takes no parameters.

Depending on the application master key settings (see Section 9.3.4), a preceding authentication with the application master key might be required.

Each File ID is coded in two byte. Duplicate values are not possible as each file must have an unambiguous identifier.

As the number of files is limited to 32 within one application, the response not always fits into one single data frame. If there is no ISO File EF, only an error code can be returned.

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

9.5.3 GetFileSettings () [2 bytes]

The GetFileSettings command allows to get information on the properties of a specific file. The information provided by this command depends on the type of the file which is queried.

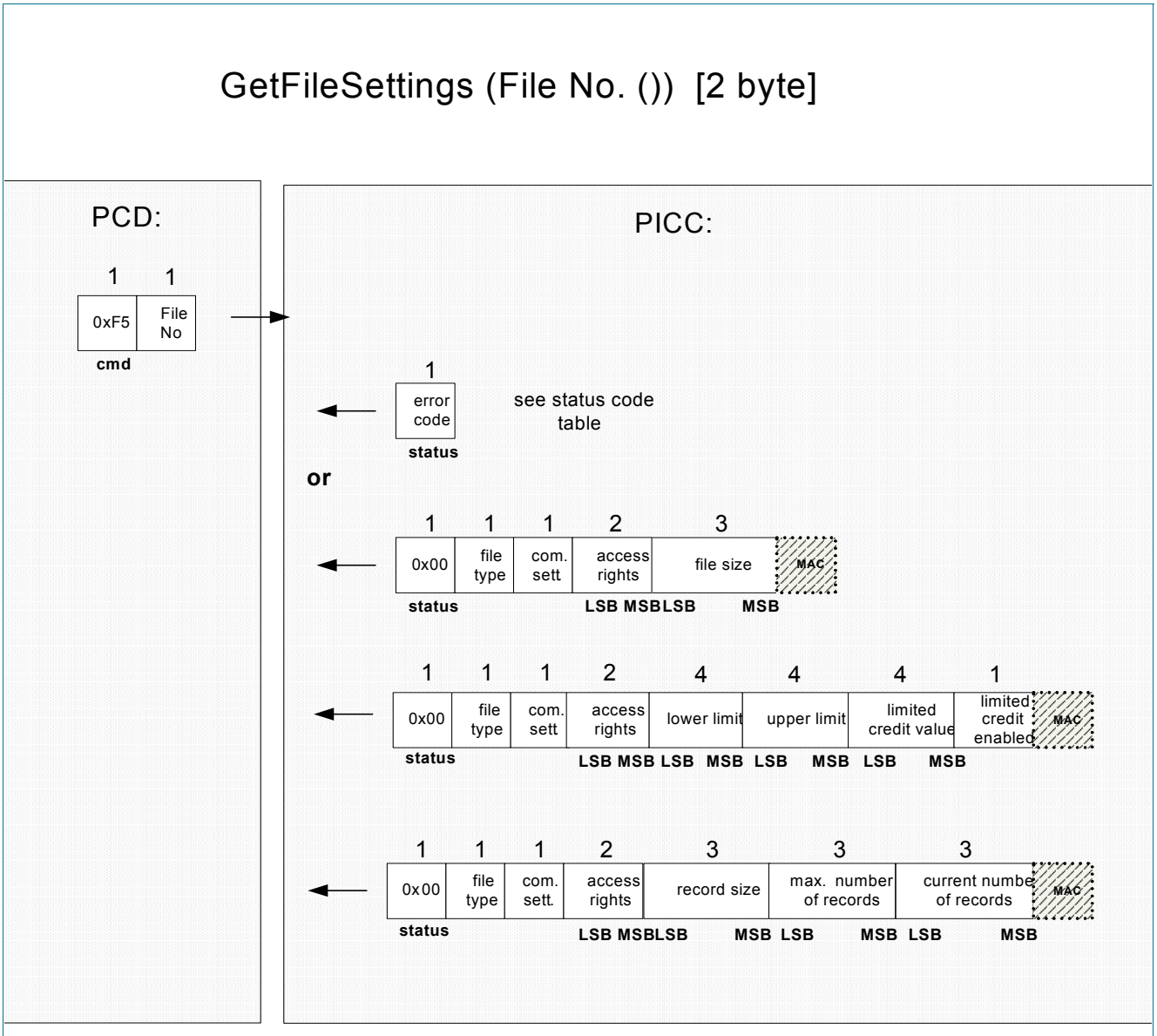

Fig 28. GetFileSettings () [2 bytes]

The GetFileSettings command takes one parameter, coding the file number of the file to be queried within the currently selected application. This file number must be in the range between 0x00 and 0x1F.

Depending on the application master key settings (see Section 9.3.4), a preceding authentication with the application master key might be required.

After updating a value file's value but before issuing the CommitTransaction command, the GetFileSettings command will always retrieve the old, unchanged limit for the limited credit value.

The first part of the returned message is the same for all file types:

The first byte indicates the file's type, see Section 8.1.

The next byte provides information on the file's communication settings (plain/MACed/Enciphered), see Section 8.2.

This information is followed by the 2 byte file Access Rights field, see Section 8.3.

All subsequent bytes in the response have a special meaning depending on the file type:

- Standard Data Files and Backup Data Files:

  One field of three bytes length returns the user file size in bytes.
- Value Files:

  Three fields, each of four bytes length, are returned whereby the first field returns the "lower limit" of the file (as defined at file creation), the second field returns the "upper limit" (as defined at file creation) and the next field returns the current maximum "limited credit" value, see Section 9.6.5. If the limited credit functionality is not in use, the last field contains all zeros. The last byte codes, if the LimitedCredit command is allowed for this file (0x00 for disabled, 0x01 for enabled).
- Linear Record Files and Cyclic Record Files:

  Three fields, each of three bytes length, are returned whereby the first field codes the size of one single record (as defined at file creation), the second field codes the maximum number of records within the record file (as defined at file creation) and the last field returns the current number of records within the record file. This number equals the maximum number of records which currently can be read, see Section 9.6.8.

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

9.5.4 ChangeFileSettings [5/10 bytes]

This command changes the access parameters of an existing file.

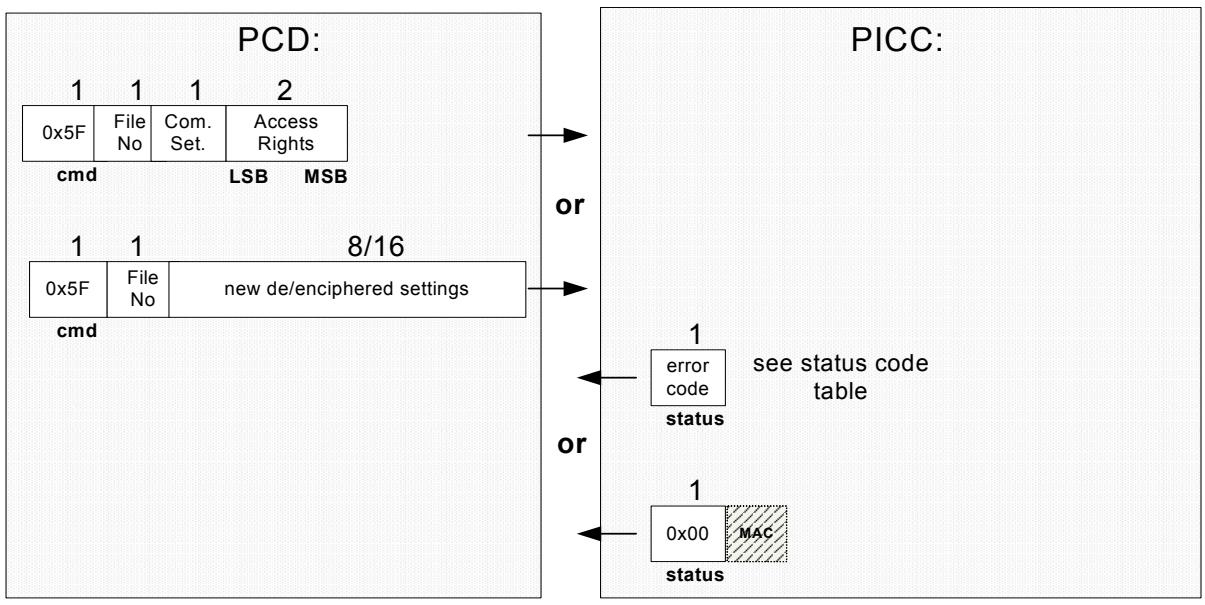

Fig 29. ChangeFileSettings [5/10 bytes]

As first parameter one byte is to be passed which codes the file number within the currently selected application.

The next byte defines the new communication settings, see Section 8.2.

Finally a two byte field defines the new Access Rights, see Section 8.3.

This change only succeeds if the current Access Rights for “Change Access Rights” is different from “never”, see also Section 8.3.

To guarantee that the ChangeFileSettings command is coming from the same party which did the preceding authentication, it is necessary to apply basically the same security mechanism as used with the ChangeKey command, see Section 9.3.6:

However, if the ChangeAccessRights Access Rights is set with the value “free”, no security mechanism is necessary and therefore the data is sent as plain text (5 byte overall length).

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

9.5.5 CreateStdDataFile [8 (10) bytes]

The CreateStdDataFile command is used to create files for the storage of plain unformatted user data within an existing application on the PICC.

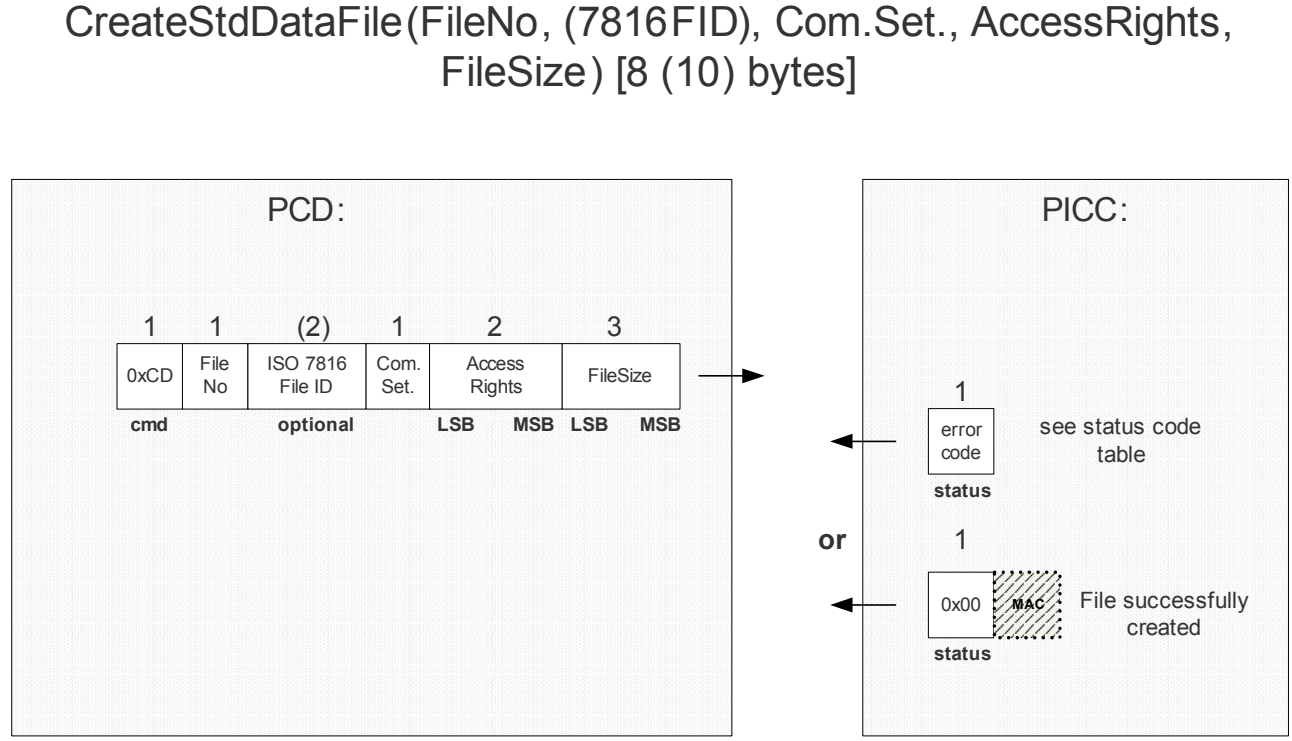

Fig 30. CreateStdDataFile [8 (10) bytes]

As first parameter this command needs the file number of the new file within the range 0x00 to 0x1F. The file will be created in the currently selected application. It is not necessary to create the files within the application in a special order. If a file with the specified number already exists within the currently selected application, an error code is returned.

The second parameter is only transmitted if ISO SELECT IDs are enabled for this application, see Section 9.4.1. The ISO SELECT ID is used to access the File via the ISO SELECT command, see Section 9.7.8.

The next byte defines the communication settings, see Section 8.2.

Then a two byte field defines the access rights for the new file, see Section 8.3.

The last parameter of three byte length specifies the size of the file in bytes.

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

Remark: The MF3ICD81 internally allocates NV-memory in a multiple of 32 bytes. Therefore a file creation command with FileSize parameter 0x00 00 01 (1 byte file size) will internally consume the same amount of NV-memory as a 0x00 00 20 (32 byte file size), namely 32 bytes.

9.5.6 CreateBackupDataFile [8(10) bytes]

The CreateBackupDataFile command is used to create 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.

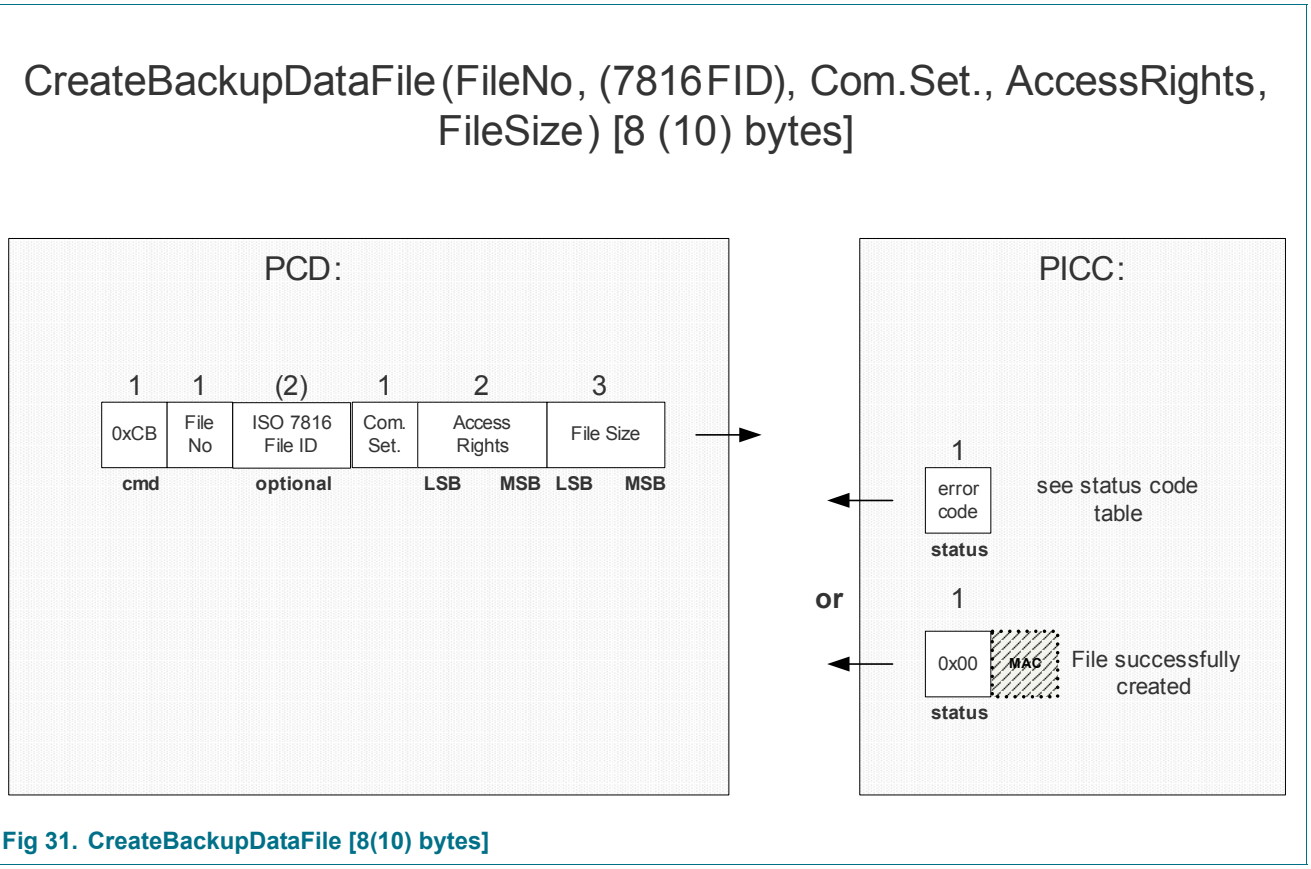

Fig 31. CreateBackupDataFile [8(10) bytes]

As first parameter this command needs the file number of the new file within the range 0x00 to 0x1F.

As the name “BackupDataFile” implies, files of this type feature an integrated backup mechanism.

Every write command is done in a independent mirror image of this file. To validate a write access to this file type, it is necessary to confirm it with a CommitTransaction command, see Section 9.6.10. If no CommitTransaction command is send by the PCD, only the mirror image is changed, the original data remains unchanged and valid.

All parameters have the same format as for the CreateStdDataFile command, see Section 9.5.5. Due to the mirror image a BackupDataFile always consumes DOUBLE the NV-memory on the PICC compared to a StdDataFile with the same specified FileSize.

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

9.5.7 CreateValueFile [18 bytes]

The CreateValueFile command is used to create files for the storage and manipulation of 32bit signed integer values within an existing application on the PICC.

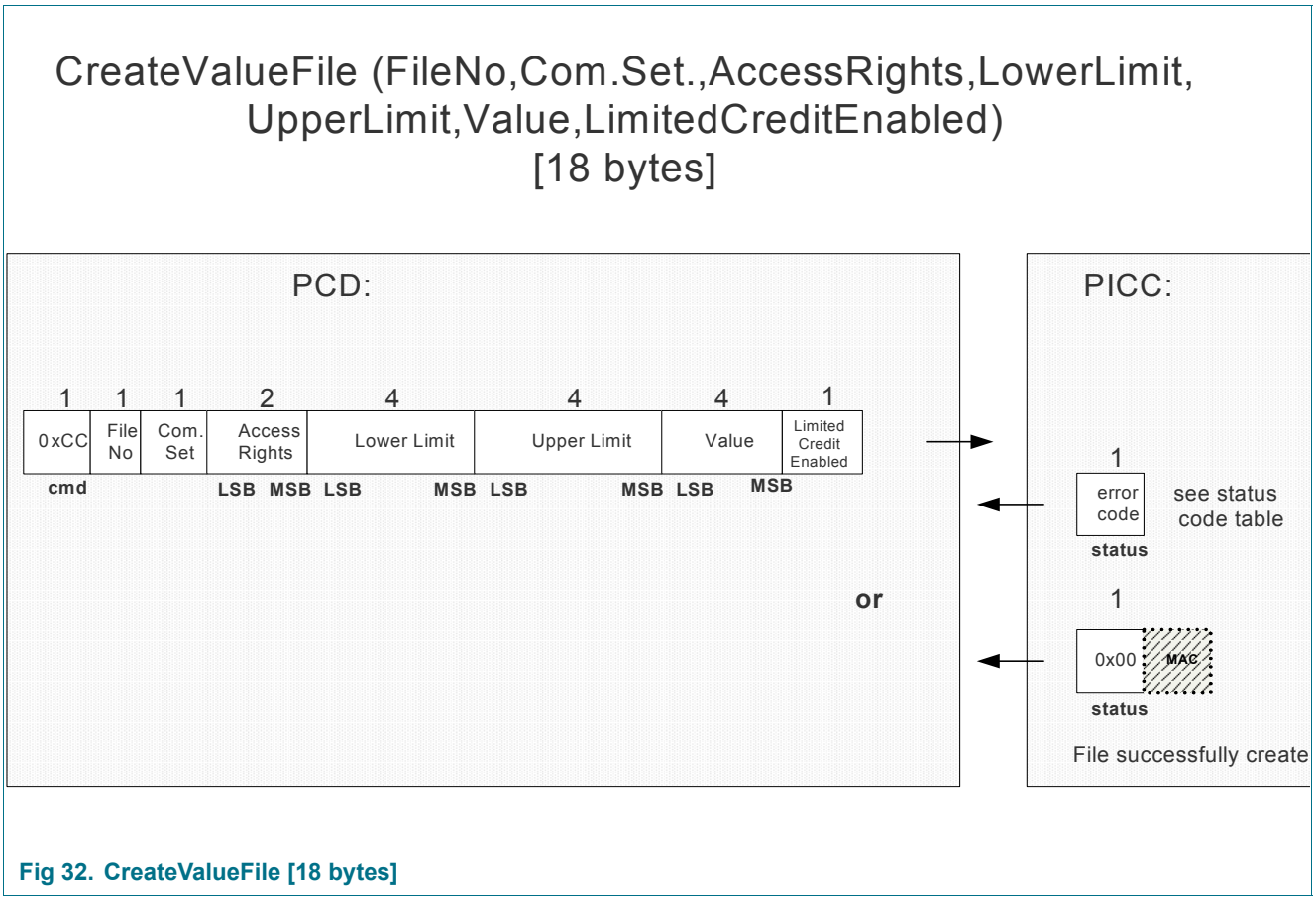

Fig 32. CreateValueFile [18 bytes]

As first parameter one byte codes the file number in the range 0x00 to 0x1F which the new created file should get within the currently selected application.

The next byte defines the communication settings, see Section 8.2

Then a two byte field defines the Access Rights for the new file, see Section 8.3.

The next parameter is of 4 byte length and codes the lower limit which is valid for this file. The lower limit marks the boundary which must not be passed by a Debit calculation on the current value, see Section 9.6.5. The lower limit is a 4 byte signed integer and thus may be negative too.

After this again 4 bytes are used to code the upper limit which sets the boundary in the same manner but for the Credit operation, see Section 9.6.4. This parameter is also a 4 byte signed integer.

The upper limit has to be higher than the lower limit, otherwise an error message would be sent by the PICC and thus the file would not be created.

The next parameter is a 4 byte signed integer again and specifies the initial value of the value file. The upper and lower limit is checked by the PICC, in case of inconsistency the file is not created and an error message is sent by the PICC.

The last byte code has several options:

- Bit 0: Enable “LimitedCredit” feature, see Section 9.6.6. Here ‘0’ means that “LimitedCredit” is disabled and ‘1’ enables this feature.
- Bit 1: Enable “Free GetValue” feature, which allows free read access to the value file. Here ‘0’ means that “Free GetValue” is disabled and ‘1’ enables this feature.If the access rights are set to disable any reading, this feature cannot be used.

ValueFiles feature always the integrated backup mechanism. Therefore every access changing the value needs to be validated using the CommitTransaction command, see Section 9.6.8.

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

9.5.8 CreateLinearRecordFile [11 (...14) bytes]

The CreateLinearRecordFile command is used to create files for multiple storage of structural data, for example for loyalty programs, within an existing application on the PICC. Once the file is filled completely with data records, further writing to the file is not possible unless it is cleared, see command ClearRecordFile, Section 9.6.9.

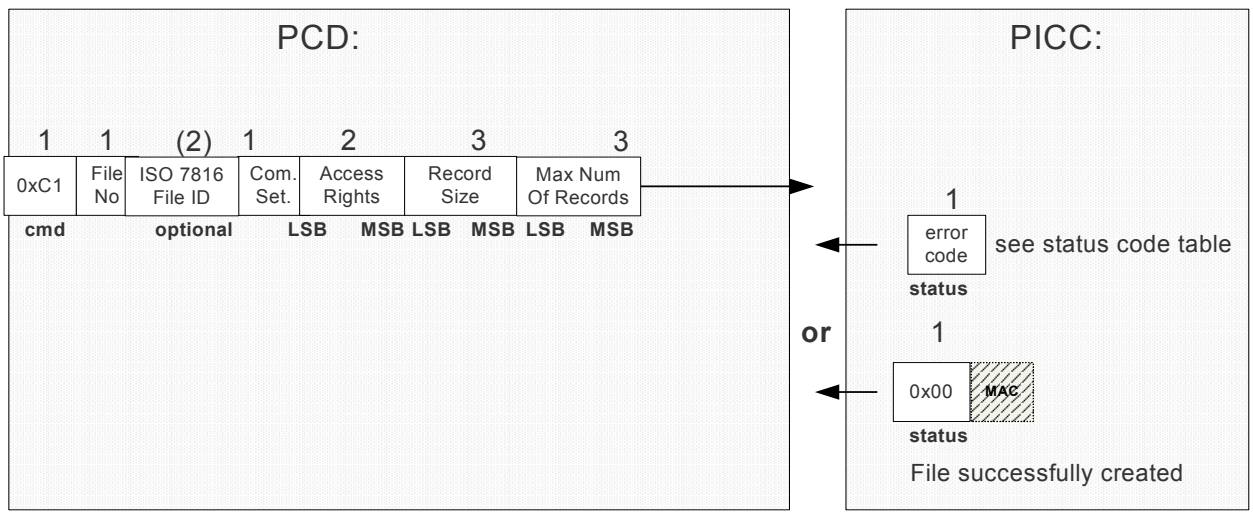

Fig 33. CreateLinearRecordFile [11 (...14) bytes]

As first parameter one byte codes the file number in the range 0x00 to 0x1F which the new created file should get within the currently selected application.

The second parameter is only transmitted if ISO SELECT IDs are enabled for this application, see Section 9.4.1. The ISO 7816 SELECT is used to access the File via the ISO SELECT command, see Section 9.7.8.

The next byte defines the communication settings, see Section 8.2.

Then a two byte field defines the access rights for the new file, see Section 8.3.

The next parameter is of three bytes length and codes the size of one single record in bytes. This parameter has to be in the range from 0x00 00 01 to 0xFF FF FF.

The next parameter is also of three bytes length and codes the number of records. This parameter has to be in the range from 0x00 00 01 to 0xFF FF FF, too.

Thus the entire file size in the PICC NV-memory is given by RecordSize*MaxNumOfRecords.

Linear Record Files feature always the integrated backup mechanism. Therefore every access appending a record needs to be validated using the CommitTransaction command, see Section 9.6.10.

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

9.5.9 CreateCyclicRecordFile [11 (13) bytes]

The CreateCyclicRecordFile command is used to create files for multiple storage of structural data, for example for logging transactions, within an existing application on the PICC. Once the file is filled completely with data records, the PICC automatically overwrites the oldest record with the latest written one. This wrap is fully transparent for the PCD.

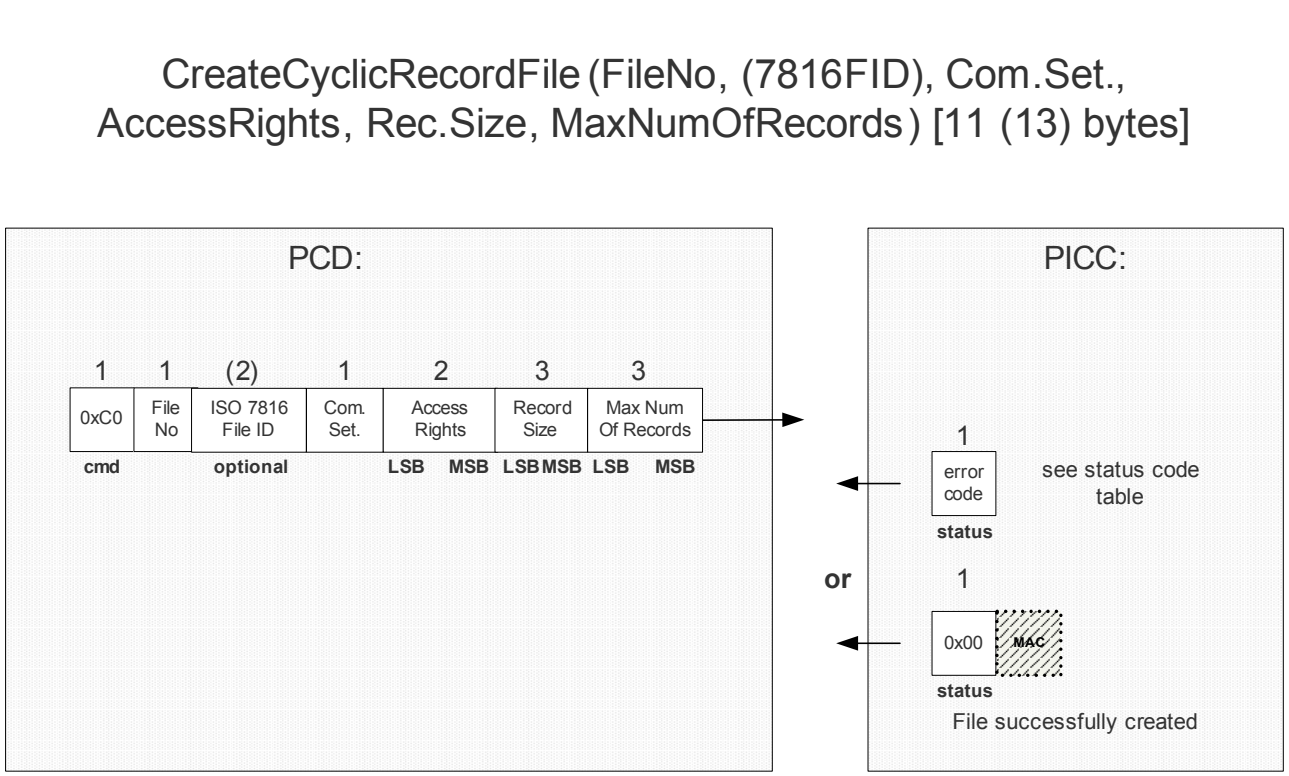

Fig 34. CreateCyclicRecordFile [11 (13) bytes]

As first parameter one byte codes the file number in the range 0x00 to 0x1F which the new created file should get within the currently selected application.

The second parameter is only transmitted if ISO SELECT IDs are enabled for this application, see Section 9.4.1. The ISO SELECT ID is used to access the File via the ISO SELECT command, see Section 9.7.8

The next byte defines the communication settings, see Section 8.2.

Then a two byte field defines the access rights for the new file, see Section 8.3.

The next parameter is of three bytes length and codes the size of one single record in bytes. This parameter has to be in the range from 0x00 00 01 to 0xFF FF FF.

The last parameter is also of three bytes length and codes the number of records. This parameter has to be in the range from 0x00 00 02 to 0xFF FF FF, too.

Thus the entire file size in the PICC NV-memory is given by RecordSize*MaxNumOfRecords.

Cyclic Record Files feature always the integrated backup mechanism. Therefore every access appending a record needs to be validated using the CommitTransaction command, see Section 9.6.10.

Information on communication security can be found in Section 7.

Note: as the backup feature consumes one record, the ‘Max. Num Of Records’ needs to be one larger than the application requires.

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

9.5.10 DeleteFile (FileNo) [2 bytes]

The DeleteFile command permanently deactivates a file within the file directory of the currently selected application.

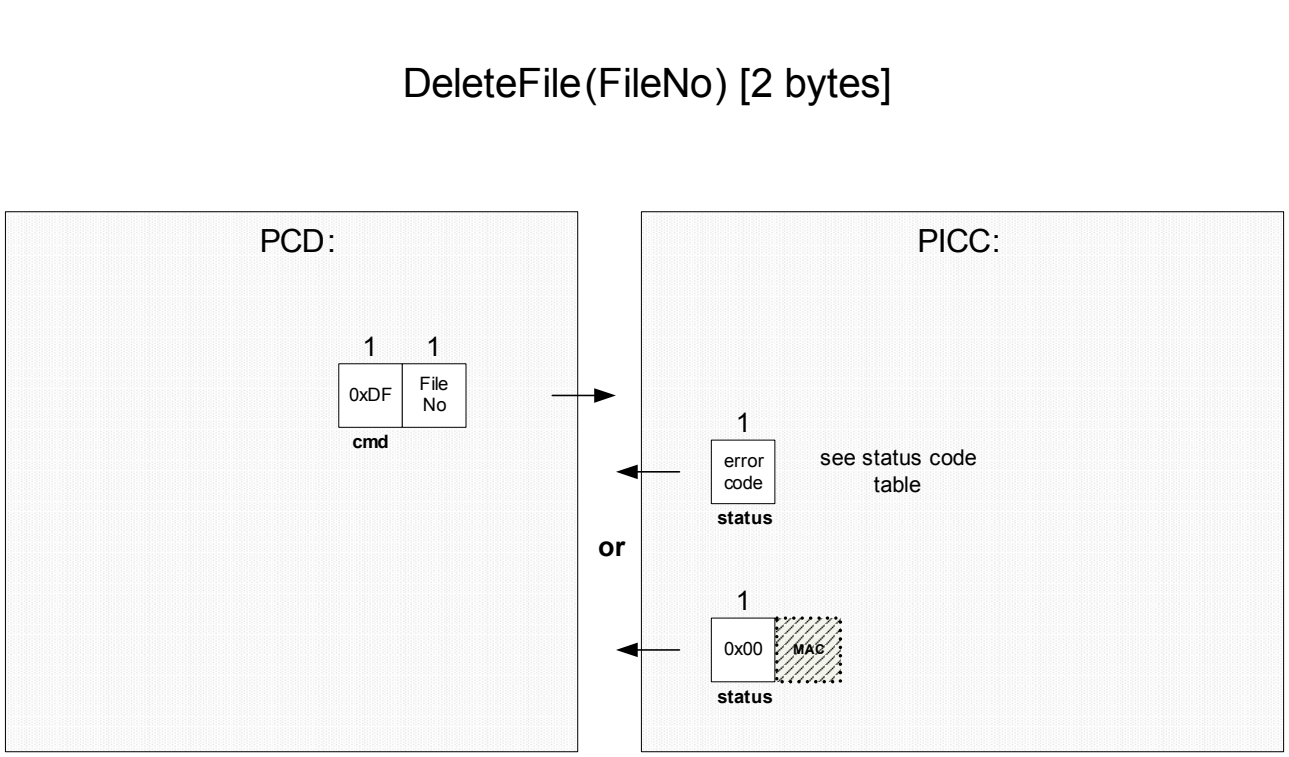

Fig 35. DeleteFile (FileNo) [2 bytes]

This command takes one byte as parameter coding the file number which is to be in the range from 0x00 to 0x1F

The operation of this command invalidates the file directory entry of the specified file which means that the file can not be accessed anymore.

Depending on the application master key settings, see Section 9.4.3, a preceding authentication with the application master key is required.

Allocated memory blocks associated with the deleted file are not set free. The File number of the deleted file can be re-used to create a new file within that application.

To release memory blocks for re-use, the whole PICC user NV-memory needs to be erased using the FormatPICC command, see Section 9.4.6.

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

9.6 MF3ICD81 Command Set – Data Manipulation Commands

The MF3ICD81 provides the following command set for Data Manipulation:

9.6.1 ReadData [8 bytes]

The ReadData command allows to read data from Standard Data Files or Backup Data Files.

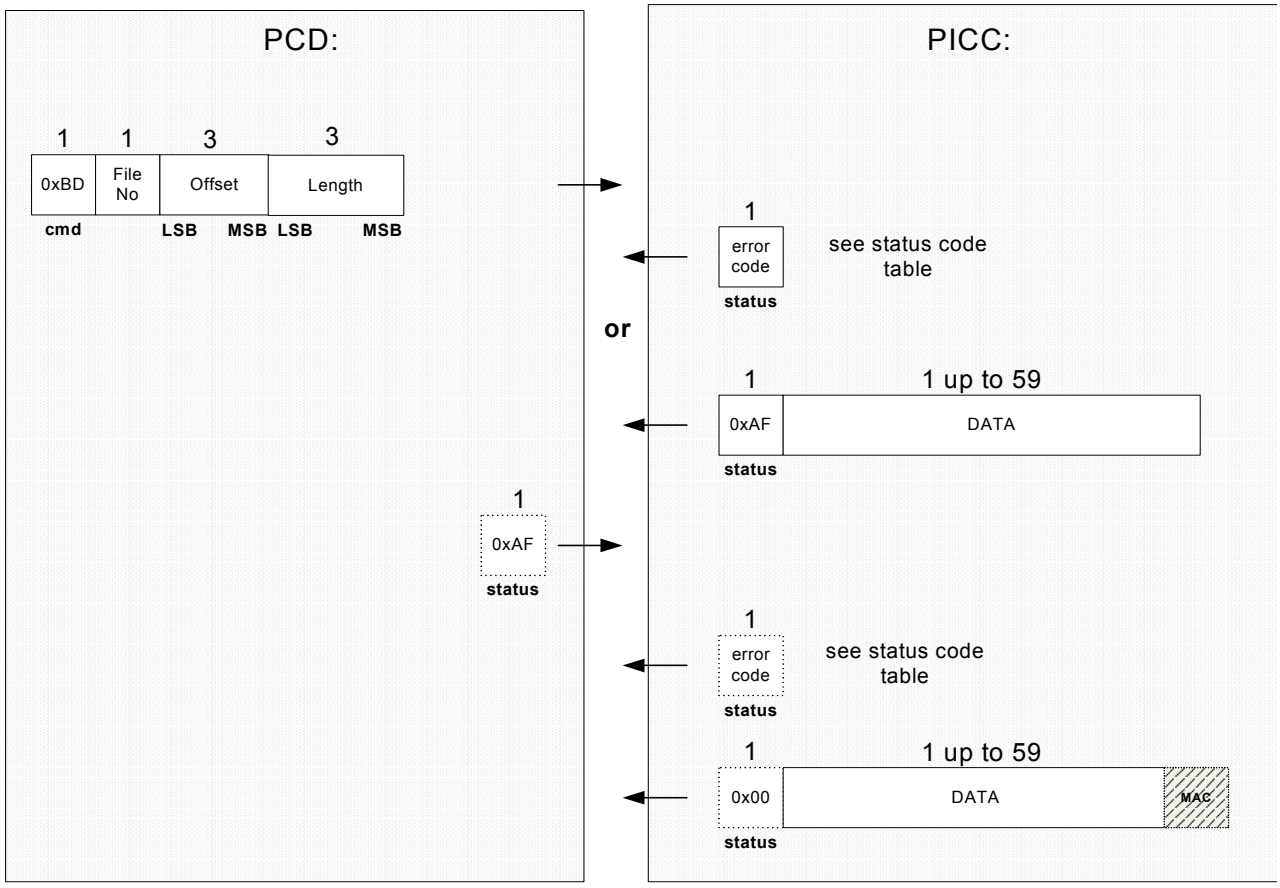

Fig 36. ReadData [8 bytes]

The first parameter is of 1 byte length and defines the file number to be read from. This parameter has to be in the range from 0x00 to 0x1F.

The next parameter is of three byte length and codes the starting position for the read operation within the file (= offset value). This parameter has to be in the range from 0x00 00 00 to file size -1.

The third parameter is also three byte long and specifies the number of data bytes to be read. This parameter can be in the range from 0x00 00 00 to 0xFF FF FF.

If the third parameter is coded as 0x00 00 00, the entire data file, starting from the position specified in the offset value, is read.

If the number of bytes to send to the PCD does not fit into one single frame, the PICC waits for a status frame with status byte 0xAF, before the next frame is sent to the PCD.

If Backup Data Files are read after writing to them, but before issuing the CommitTransaction command, see Section 9.6.10, the ReadData command will always retrieve the old, unchanged data stored in the PICC. All data written to a Backup Data File is validated and externally “visible” for a ReadData command only after a CommitTransaction command.

The Read command requires a preceding authentication either with the key specified for “Read” or “Read&Write” access, see Section 8.3.

Depending on the communication settings, see Section 8.2, linked to the file, data will be sent by the PICC either plain, MACed or enciphered. All cryptographic operations are done in CBC mode.

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

Remark: Please check the communication mode and the access rights of the file.

9.6.2 WriteData [8+ bytes]

The WriteData command allows to write data to Standard Data Files and Backup Data Files.

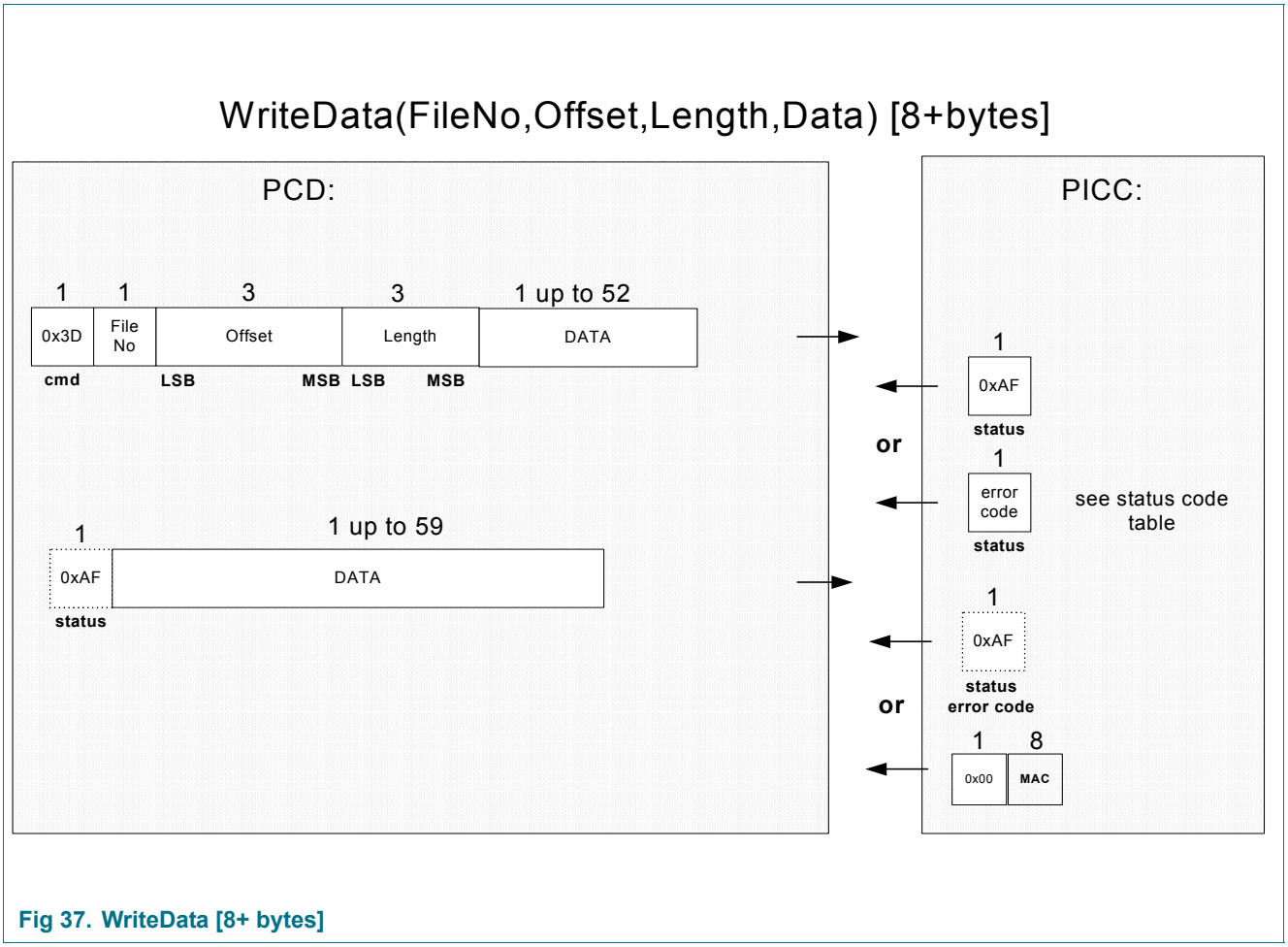

Fig 37. WriteData [8+ bytes]

The first parameter is of 1 byte length and defines the file number to be written to; valid range is 0x00 to 0x1F.

The next parameter is of three bytes length and codes the starting position for the write operation within the file (= offset value). This parameter has to be in the range from 0x00 00 00 to file size -1.

The third parameter is also three bytes long and specifies the number of data bytes to be written. This parameter can be in the range from 0x00 00 01 to 0xFF FF FF.

If the number of bytes to send does not fit into one single frame, the PCD has to wait for a status frame with status byte 0xAF before sending the next frame to the PICC.

The Write command requires a preceding authentication either with the key specified for “Write” or “Read&Write” access, see Section 8.3.

Depending on the communication settings, see Section 8.2, linked to the file, data needs to be sent by the PCD either plain, MACed or enciphered. All cryptographic operations are done in CBC mode.

For MACed and enciphered communication padding of the data string is necessary to reach an overall data-length of multiples of eight. This padding is done with all 0x00.

If the WriteData operation is performed on a Backup Data File, it is necessary to validate the written data with a CommitTransaction command, see Section 9.6.10. An AbortTransaction command will invalidate all changes, see Section 9.6.11.

If data is written to Standard Data Files (without integrated backup mechanism), data is directly programmed into the visible NV-memory of the file. The new data is immediately available to any following ReadData commands performed on that file.

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

Remark: In case of MACed or enciphered communication the validity of data is verified by the PICC by checking the MAC or the CRC (including necessary padding bytes) which is transmitted at the end of the data frame. If the verification fails (MAC / CRC does not fit data, padding bytes invalid), the PICC stops further NV-programming and returns an Integrity Error to the PCD, see Section 9.6.11. As a consequence of the Integrity Error any transaction, which might have begun, is automatically aborted.

Remark: Getting an Integrity Error when writing on a Standard Data File can corrupt the content of the file.

Remark: In case of MACing the padding bytes are only used for cryptographic purpose but NOT exchanged between PCD and PICC.

Remark: If the communication mode is enciphered; but the file is configured for free access, then the communication is either in plain (Authenticate 0x0A) or CMACed (other Authentications).

9.6.3 GetValue [2 bytes]

The GetValue command allows to read the currently stored value from Value Files.

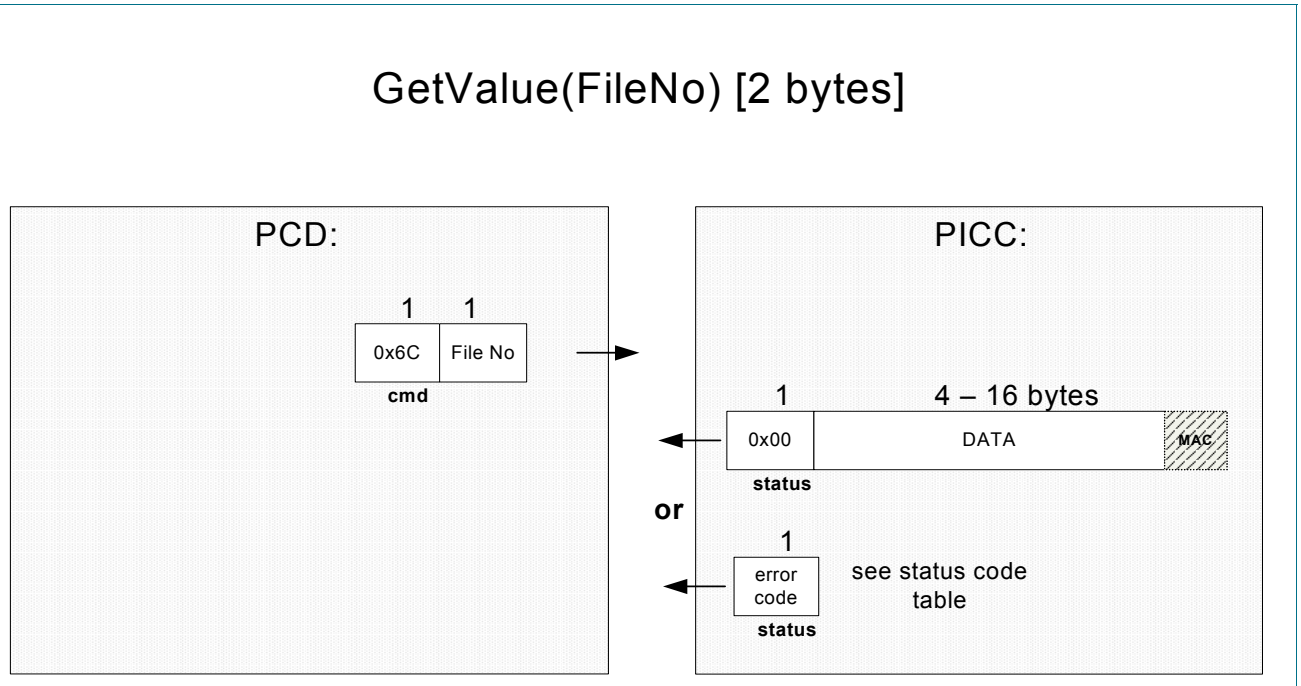

Fig 38. GetValue [2 bytes]

The only parameter sent with this command is of one byte length and codes the file number. This parameter has to be in the range from 0x00 to 0x1F.

The PICC response holds the current value of the Value file. Depending on the communication mode data and the type of authentication the following data structure will be sent back (see more information on CRC and CMAC in Section 7):

- Authentication 0x0A; transfer in plain: 4 bytes
- Authentication 0x0A; transfer deciphered: 8 bytes (4 bytes value, 2 bytes CRC, 2 bytes padding with 0x00)
- Authentication 0x0A: transfer maced: 8 bytes (4 bytes value, 4 bytes MAC)
- Other authentication 3DES/DES: transfer enciphered: 8 bytes (4 bytes value, 4 bytes CRC)
- Other authentication: transfer maced: 12 bytes (4 bytes value, 8 bytes CMAC)
- AES: transfer enciphered: 16 bytes

The value is always represented LSB first.

The GetValue command requires a preceding authentication with the key specified for Read, Write or Read&Write access, see Section 8.3.

After updating a value file's value but before issuing the CommitTransaction command, the GetValue command will always retrieve the old, unchanged value which is still the valid one.

If “GetFreeValue” is enabled within “CreateValueFile”; then the communication mode is plain (authenticate 0x0A) or CMACed (other authentications).

Depending on the option parameter in the CreateValueFile command, the command GetValue(FileNo)[] the access can be granted freely.

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

9.6.4 Credit (FileNo, Data) [6/14 bytes]

The Credit command allows to increase a value stored in a Value File.

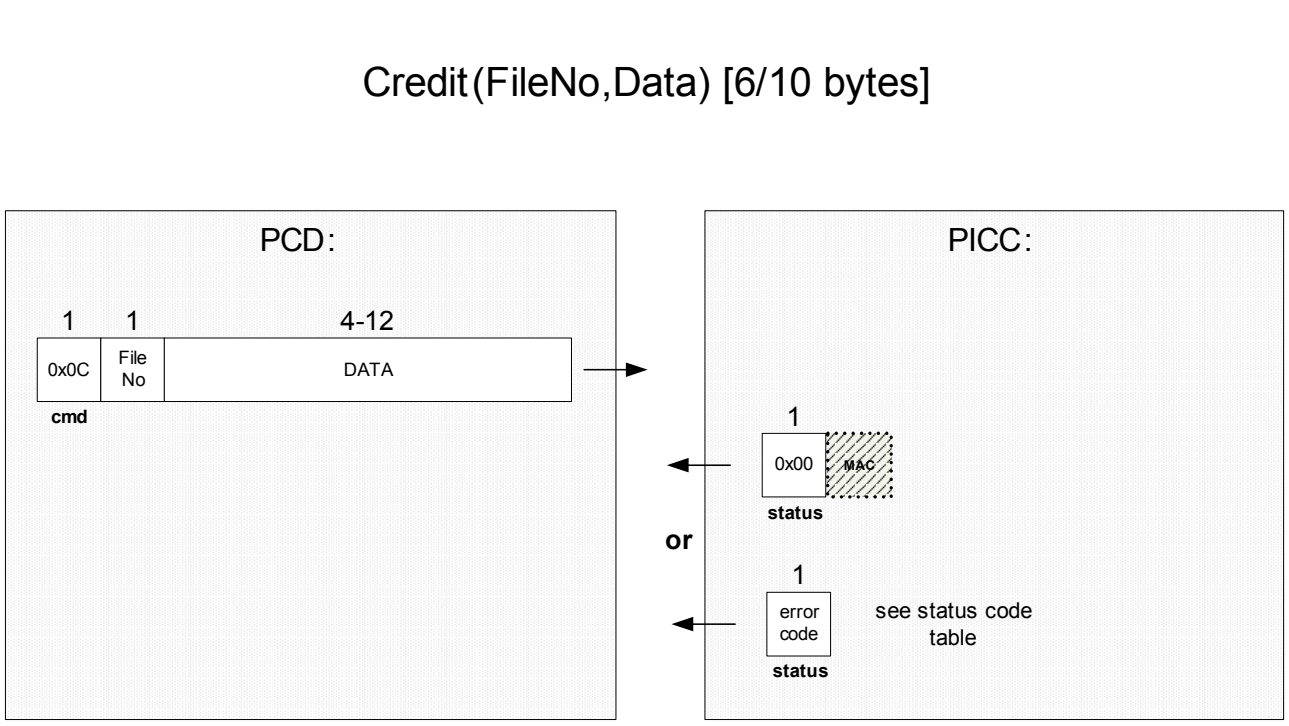

Fig 39. Credit [6/10 bytes]

The first parameter sent with this command is of one byte length and codes the file number. This parameter has to be in the range from 0x00 to 0x1F.

This command increases the current value stored in the file by a certain amount (4 byte signed integer) which is transmitted in the data field. Only positive values are allowed for the Credit command.

Depending on the communication mode data and the type of authentication the following data structure shall be written to the card:

- Authentication 0x0A; transfer in plain: 4 bytes
- Authentication 0x0A; transfer deciphered: 8 bytes (4 bytes value, 2 bytes CRC, 2 bytes padding with 0x00)

- Authentication 0x0A: transfer maced: 8 bytes (4 bytes value, 4 bytes MAC)
- Other authentication: transfer in plain: 4 bytes (the CMAC shall be stored for the next commands)
- Other authentication: transfer enciphered: 16 bytes (4 bytes value, 4 bytes CRC)
- Other authentication: transfer maced: 12 bytes (4 bytes value, 8 bytes CMAC)

The value is always represented LSB first.

It is necessary to validate the updated value with a CommitTransaction command, see Section 9.6.10. An AbortTransaction command will invalidate all changes, see Section 9.6.11.

The value modifications of Credit, Debit and LimitedCredit commands are cumulated until a CommitTransaction command is issued. Credit commands do NEVER modify the Limited Credit Value of a Value file. However, if the Limited Credit Value needs to be set to 0, a LimitedCredit with value 0 can be used.

The Credit command requires a preceding authentication with the key specified for “Read&Write” access, see Section 8.3.

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

9.6.5 Debit (FileNo, Data) [6/10 bytes]

The Debit command allows decreasing a value stored in a Value File.

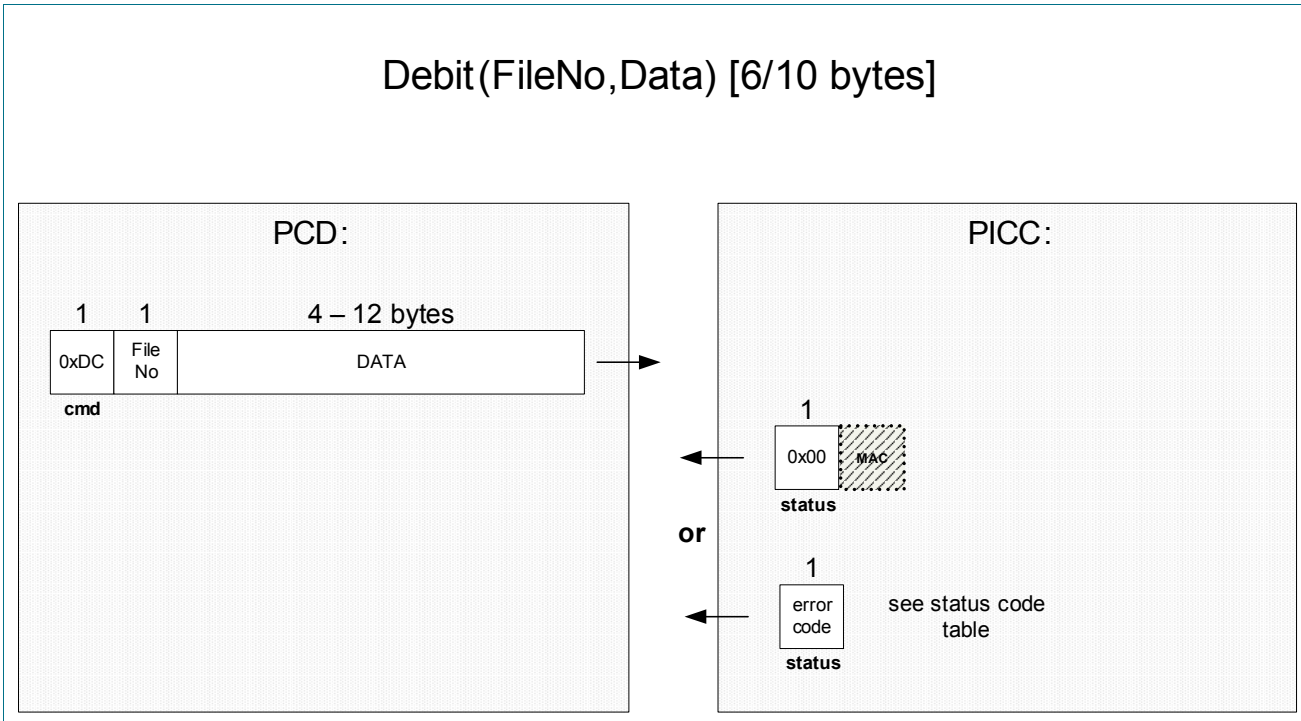

Fig 40. Debit [6/10 bytes]

The first parameter is of one byte length and codes the file number. This parameter has to be in the range from 0x00 to 0x1F.

The next parameter (4 byte signed integer) holds the value which will be subtracted from the current value stored in the file. Only positive values are allowed for the Debit command.

Depending on the communication mode data and the type of authentication the following data structure shall be written to the card:

- Authentication 0x0A; transfer in plain: 4 bytes
- Authentication 0x0A; transfer deciphered: 8 bytes (4 bytes value, 2 bytes CRC, 2 bytes padding with 0x00)
- Authentication 0x0A: transfer maced: 8 bytes (4 bytes value, 4 bytes MAC)
- Other authentication: transfer in plain: 4 bytes (the CMAC shall be stored for the next commands)
- Other authentication: transfer enciphered: 16 bytes (4 bytes value, 4 bytes CRC, 8 bytes padding)
- Other authentication: transfer maced: 12 bytes (4 bytes value, 8 bytes CMAC)

The value is always represented LSB first.

It is necessary to validate the updated value with a CommitTransaction command, see Section 9.6.10. An AbortTransaction command will invalidate all changes, see Section 9.6.11.

The value modifications of Credit, Debit and LimitedCredit commands are cumulated until a CommitTransaction command is issued.

The Debit command requires a preceding authentication with one of the keys specified for Read, Write or Read&Write access, see Section 8.3.

If the usage of the LimitedCredit feature is enabled, the new limit for a subsequent LimitedCredit command is set to the sum of Debit commands within one transaction before issuing a CommitTransaction command. This assures that a LimitedCredit command can not re-book more values than a debiting transaction deducted before.

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

9.6.6 LimitedCredit (FileNo, Data) [6/10 bytes]

The LimitedCredit command allows a limited increase of a value stored in a Value File without having full Read&Write permissions to the file. This feature can be enabled or disabled during value file creation.

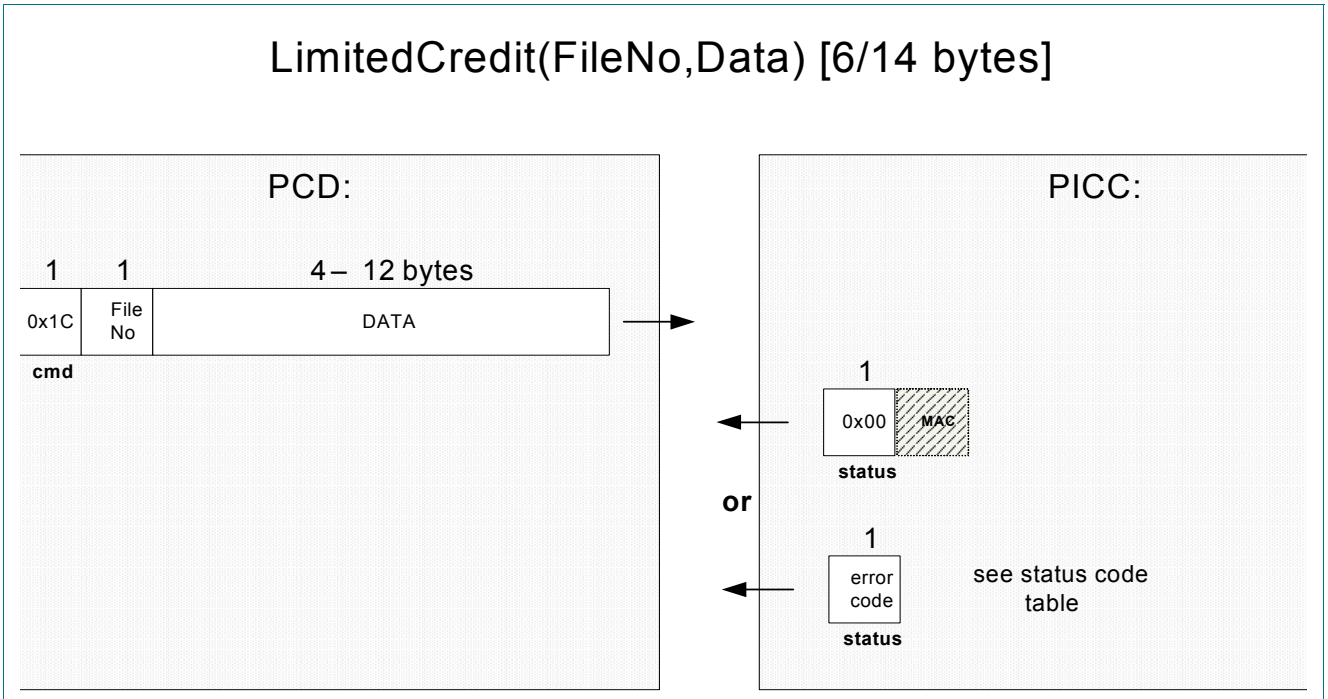

Fig 41. LimitedCredit [6/10 bytes]

The first parameter sent with this command is of one byte length and codes the file number. This parameter has to be in the range from 0x00 to 0x1F.

This command increases the current value stored in the file by a certain amount (4 byte signed integer) which is transmitted in the data field. Only positive values are allowed for the LimitedCredit command.

Depending on the communication mode data and the type of authentication the following data structure shall be written to the card:

- Authentication 0x0A; transfer in plain: 4 bytes
- Authentication 0x0A; transfer deciphered: 8 bytes (4 bytes value, 2 bytes CRC, 2 bytes padding with 0x00)
- Authentication 0x0A: transfer maced: 8 bytes (4 bytes value, 4 bytes MAC)
- Other authentication: transfer in plain: 4 bytes (the CMAC shall be stored for the next commands)
- Other authentication: transfer enciphered: 16 bytes (4 bytes value, 4 bytes CRC, 8 bytes padding)
- Other authentication: transfer maced: 12 bytes (4 bytes value, 8 bytes CMAC)

The value is always represented LSB first.

It is necessary to validate the updated value with a CommitTransaction command, see Section 9.6.10. An AbortTransaction command will invalidate all changes, see Section 9.6.11.

The value modifications of Credit, Debit and LimitedCredit commands are cumulated until a CommitTransaction command is issued.

The LimitedCredit command requires a preceding authentication with the key specified for “Write” or "Read & Write" access, see Section 8.3.

The value for LimitedCredit is limited to the sum of the Debit commands on this value file within the most recent transaction containing at least one Debit. After executing the LimitedCredit command the new limit is set to 0 regardless of the amount which has been re-booked. Therefore the LimitedCredit command can only be used once after a Debit transaction.

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

9.6.7 WriteRecord (FileNo, Offset, Length, Data) [8+ bytes]

The WriteRecord command allows to write data to a record in a Cyclic or Linear Record File.

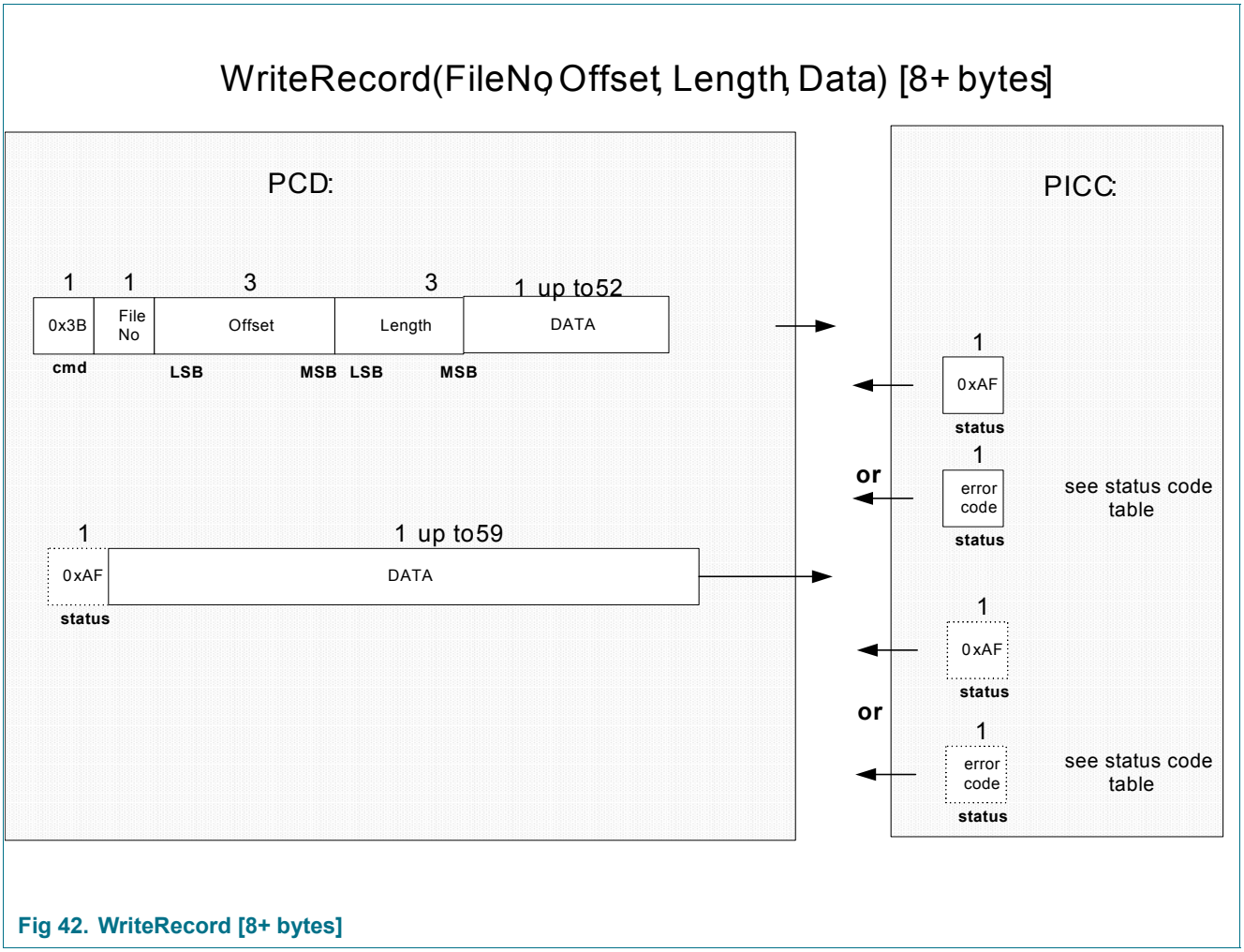

Fig 42. WriteRecord [8+ bytes]

The first parameter is of one byte length and codes the file number. This parameter has to be in the range from 0x00 to 0x1F.

The next three bytes code the offset within one single record (in bytes). This parameter has to be in the range from 0x00 00 00 to record size - 1.

This parameter is followed by another three bytes coding the length of data which is to be written to the record file. This parameter has to be in the range from 0x00 00 01 to record size - offset.

The WriteRecord command appends one record at the end of the linear record file, it erases and overwrites the oldest record in case of a cyclic record file if it is already full. The entire new record is cleared before data is written to it.

If no CommitTransaction command (see Section 9.6.10) is sent after a WriteRecord command, the next WriteRecord command to the same file writes to the already created record. After sending a CommitTransaction command, a new WriteRecord command will create a new record in the record file. An AbortTransaction command will invalidate all changes, see Section 9.6.11.

After issuing a ClearRecordFile command, but before a CommitTransaction / AbortTransaction command, a WriteRecord command to the same record file will fail.

Depending on the communication settings, see Section 8.2, linked to the file, data needs to be sent by the PCD either plain, MACed or enciphered.

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

9.6.8 ReadRecords (FileNo, Offset, Length) [8 bytes]

The ReadRecords command allows to read out a set of complete records from a Cyclic or Linear Record File.

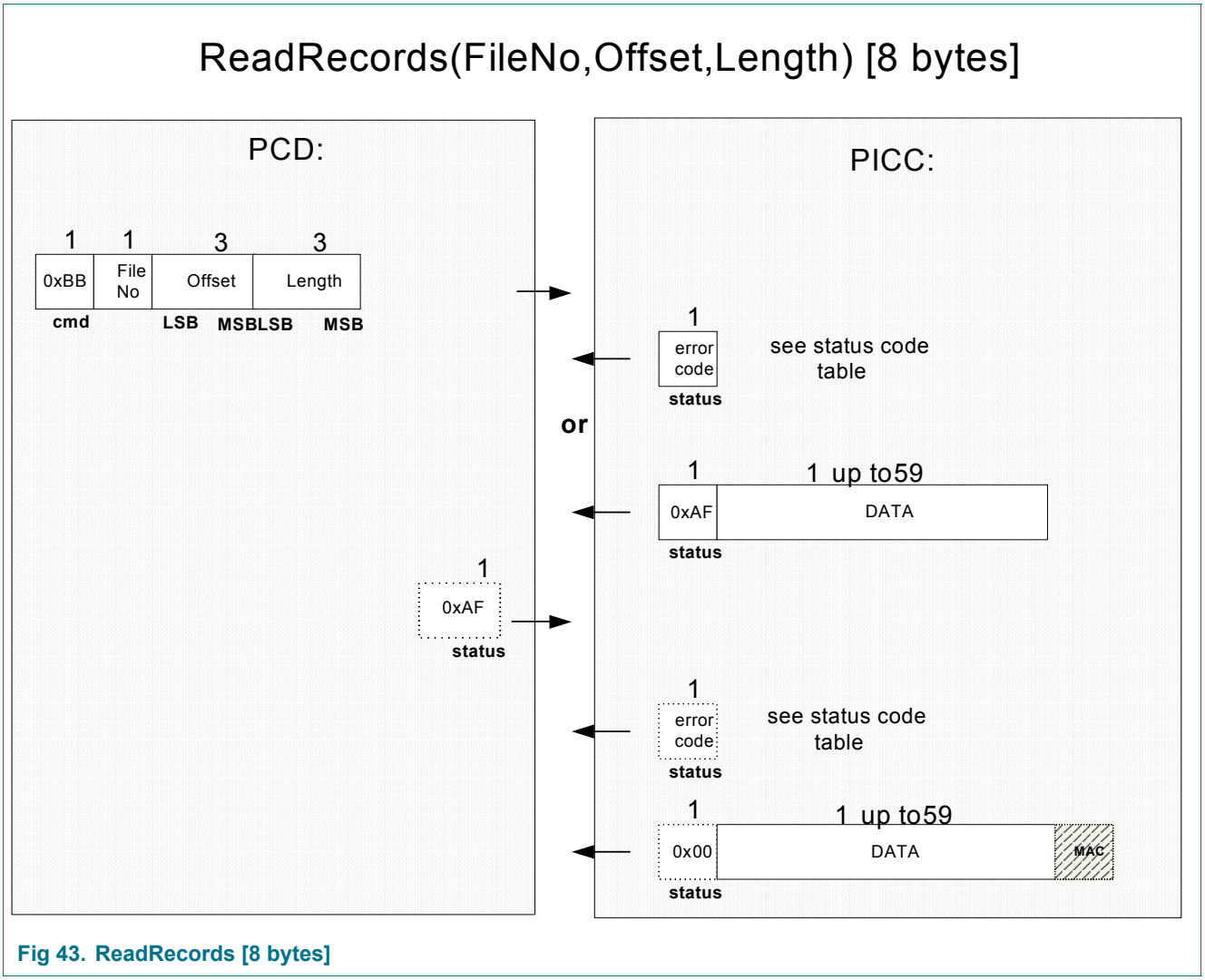

Fig 43. ReadRecords [8 bytes]

The first parameter is of one byte length and codes the file number in the range from 0x00 to 0x1F.

The next parameter is three bytes long and codes the offset of the newest record which is read out. In case of 0x00 00 00 the latest record is read out. The offset value must be in the range from 0x00 to number of existing records – 1.

The third parameter is another three bytes which code the number of records to be read from the PICC. Records are always transmitted by the PICC in chronological order (= starting with the oldest, which is number of records – 1 before the one addressed by the given offset). If this parameter is set to 0x00 00 00 then all records, from the oldest record up to and including the newest record (given by the offset parameter) are read.

The allowed range for the number of records parameter is from 0x00 00 00 to number of existing records – offset.

Remark: In cyclic record files the maximum number of stored valid records is one less than the number of records specified in the CreateCyclicRecordFile command.

A ReadRecords command on an empty record file (directly after creation or after a committed clearance, see Section 9.6.9) will result in an error.

The ReadRecords command requires a preceding authentication either with the key specified for “Read” or “Read&Write” access, see Section 8.3.

Depending on the communication setting, see Section 8.2, and authentication linked to the file, data will be sent by the PICC either plain, MACed or enciphered. All cryptographic operations are done in CBC mode.

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

9.6.9 ClearRecordFile (FileNo) [2 bytes]

The ClearRecordFile command allows to reset a Cyclic or Linear Record File to the empty state.

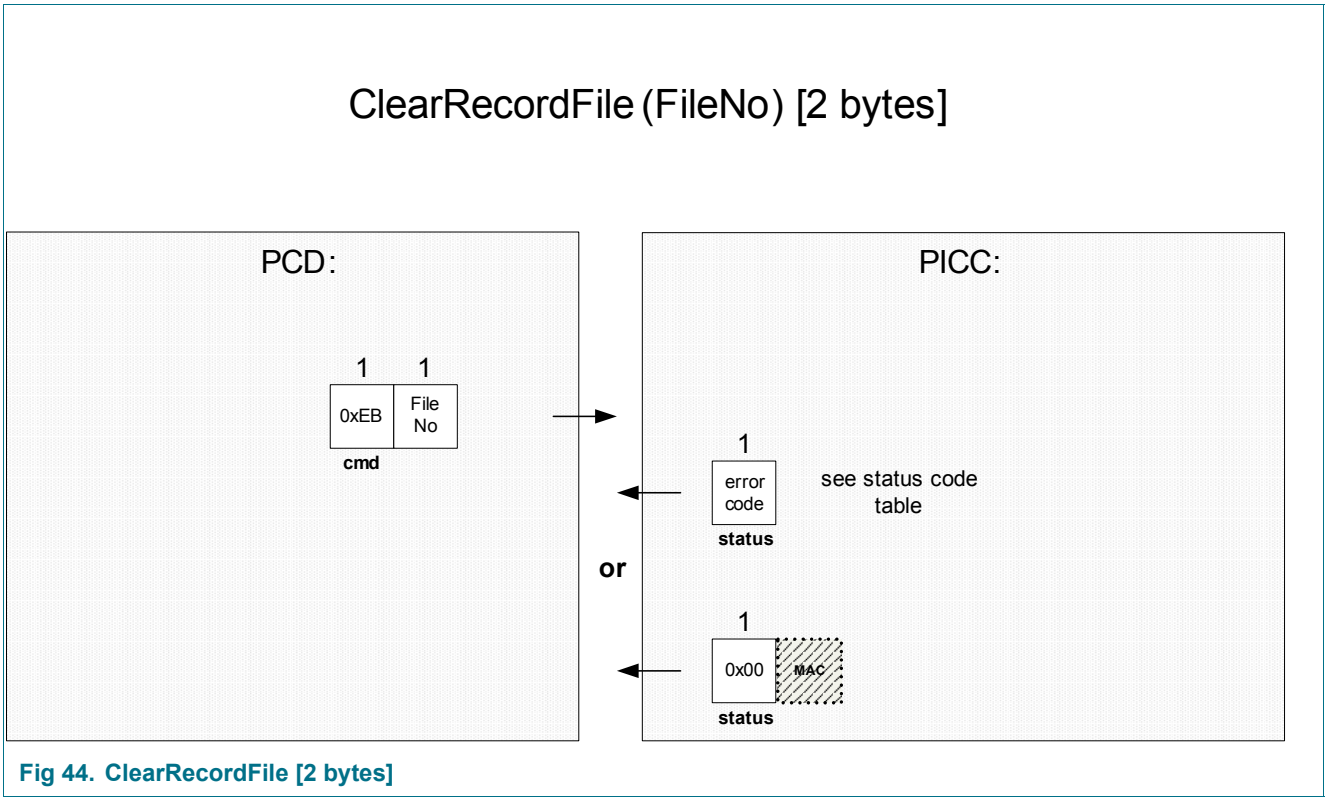

Fig 44. ClearRecordFile [2 bytes]

The only parameter sent with this command is of one byte length and codes the file number in the range from 0x00 to 0x1F.

After executing the ClearRecordFile command but before CommitTransaction, all subsequent WriteRecord commands, see Section 9.6.7, will fail. The ReadRecords command, see Section 9.6.8, will return the old still valid records.

After the CommitTransaction command is issued, a ReadRecords command will fail, WriteRecord commands will be successful.

An AbortTransaction command (instead of CommitTransaction) will invalidate the clearance, see Section 9.6.11.

Full “Read&Write” permission on the file is necessary for executing this command, see Section 8.3.

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

9.6.10 CommitTransaction [1 byte]

The CommitTransaction command allows to validate all previous write access on Backup Data Files, Value Files and Record Files within one application.

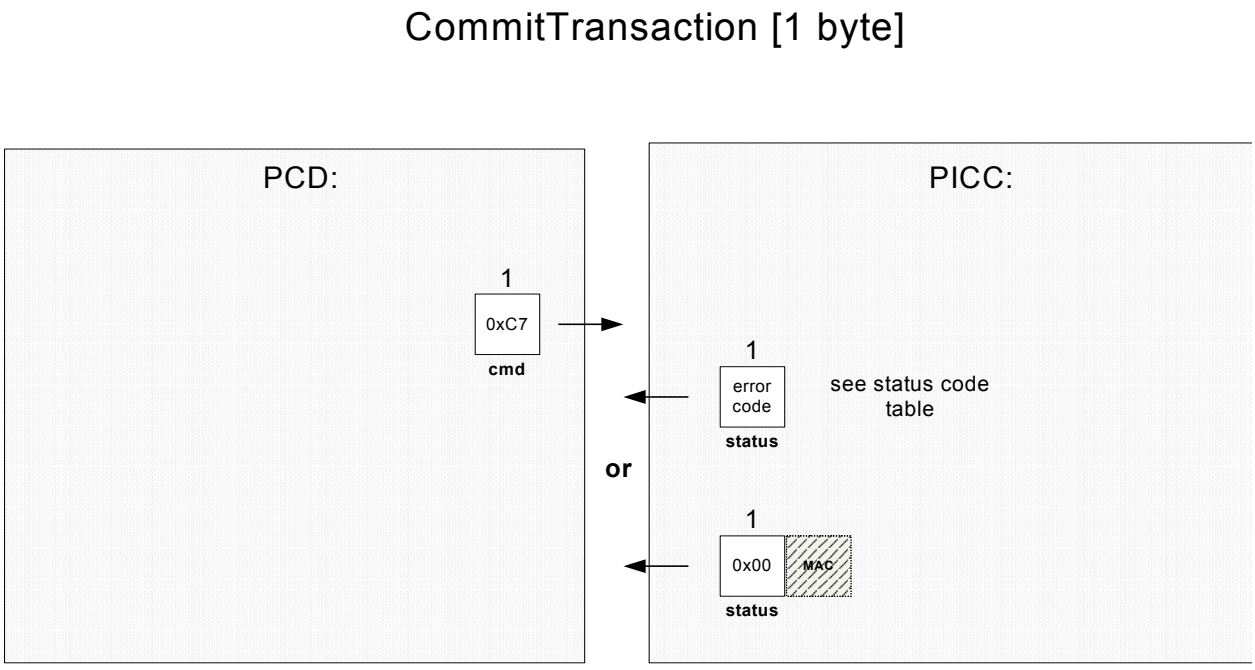

Fig 45. CommitTransaction [1 byte]

This command takes no parameter.

The CommitTransaction command validates all write access to files with integrated backup mechanisms:

- Backup Data Files
- Value Files
- Linear Record Files
- Cyclic Record Files

The CommitTransaction is typically the last command of a transaction before the ISO/IEC 14443-4 Deselect command or before proceeding with another application (SelectApplication command).

As logical counter-part of the CommitTransaciton command the AbortTransaction command allows to invalidate changes on files with integrated backup management, see Section 9.6.11.

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

9.6.11 AbortTransaction [1 byte]

The AbortTransaction command allows to invalidate all previous write access on Backup Data Files, Value Files and Record Files within one application.

This is useful to cancel a transaction without the need for re-authentication to the PICC, which would lead to the same functionality.

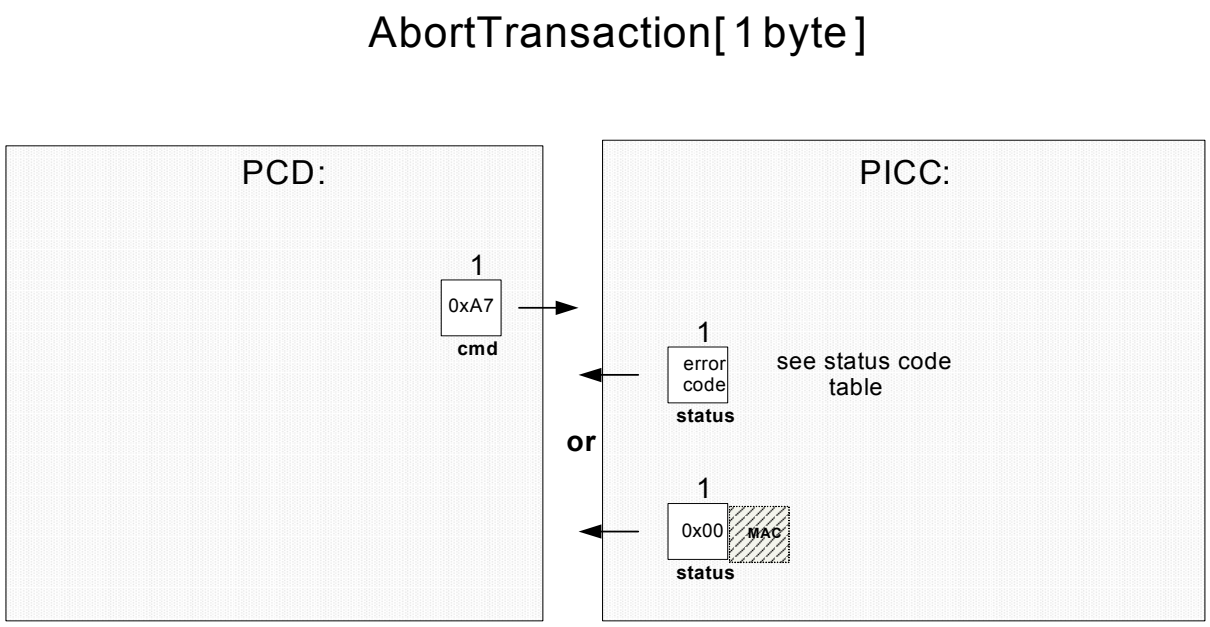

Fig 46. AbortTransaction [1 byte]

This command takes no parameter.

The AbortTransaction command invalidates all write access to files with integrated backup mechanisms without changing the authentication status:

- Backup Data Files
- Value Files
- Linear Record Files
- Cyclic Record Files

Information on authentication and communication dependent structure of command can be found in Section 7.3 for communication after 0x0A Authenticate or in Section 7.4 for communication AuthenticateISO 0x1A or AuthenticateAES 0xAA.

9.7 Command Set ISO/IEC 7816-4 – Basic inter-industry commands

The MF3ICD81 provides the following command set according to ISO/IEC 7816-4 clause 6:

- INS code ‘A4’ SELECT FILE
- INS code ‘B0’ READ BINARY
- INS code ‘D6’ UPDATE BINARY
- INS code ‘B2’ READ RECORDS
- INS code ‘D2’ UPDATE RECORD
- INS code ‘E2’ APPEND RECORD
- INS code ‘84’ GET CHALLENGE
- INS code ‘88’ INTERNAL AUTHENTICATE
- INS code ‘82’ EXTERNAL AUTHENTICATE

9.7.1 ISO/IEC 7816-4 APDU message structure

DESFire supports the APDU message structure according to ISO/IEC 7816-4 for

- an optional wrapping of the native DESFire APDU format
- for the additionally implemented 7816-4 commands, as described later on.

9.7.2 DESFire implementation of ISO/IEC 7816-4

MF3ICD81 supports the following **communication modes:**

- plain communication
- communication secured with CMAC only for ’Read Binary’ and ’Read Records’.

As the files could have different communication settings, the implementation needs to take care that the communication settings of the file fit to the communication mode used during the implementation. A communication mode with encryption is not possible with ISO/IEC 7816-4 commands and therefore the PICC will answer with an error. The length of the CMAC is not included in the Le field.

The byte order of the native command set uses a different byte order then the ISO 7816-4 command set. If a file is created with the native command set, data like FID are in a different byte order, which needs to take into account. (Difference big endian and little endian)

Remark: For the wrapping option (ISO 7816-4 INS is used) used together with Omnikey Reader 5321, please check the correct registry key:
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\CardMan\RFID]
"DesfireNative"=dword:00000001
For further questions, please download the developer guide from the Omnikey website:
http://omnikey.aaitg.com

9.7.3 Selection of native DESFire APDU Framing versus ISO/IEC 7816-4 framing and commands

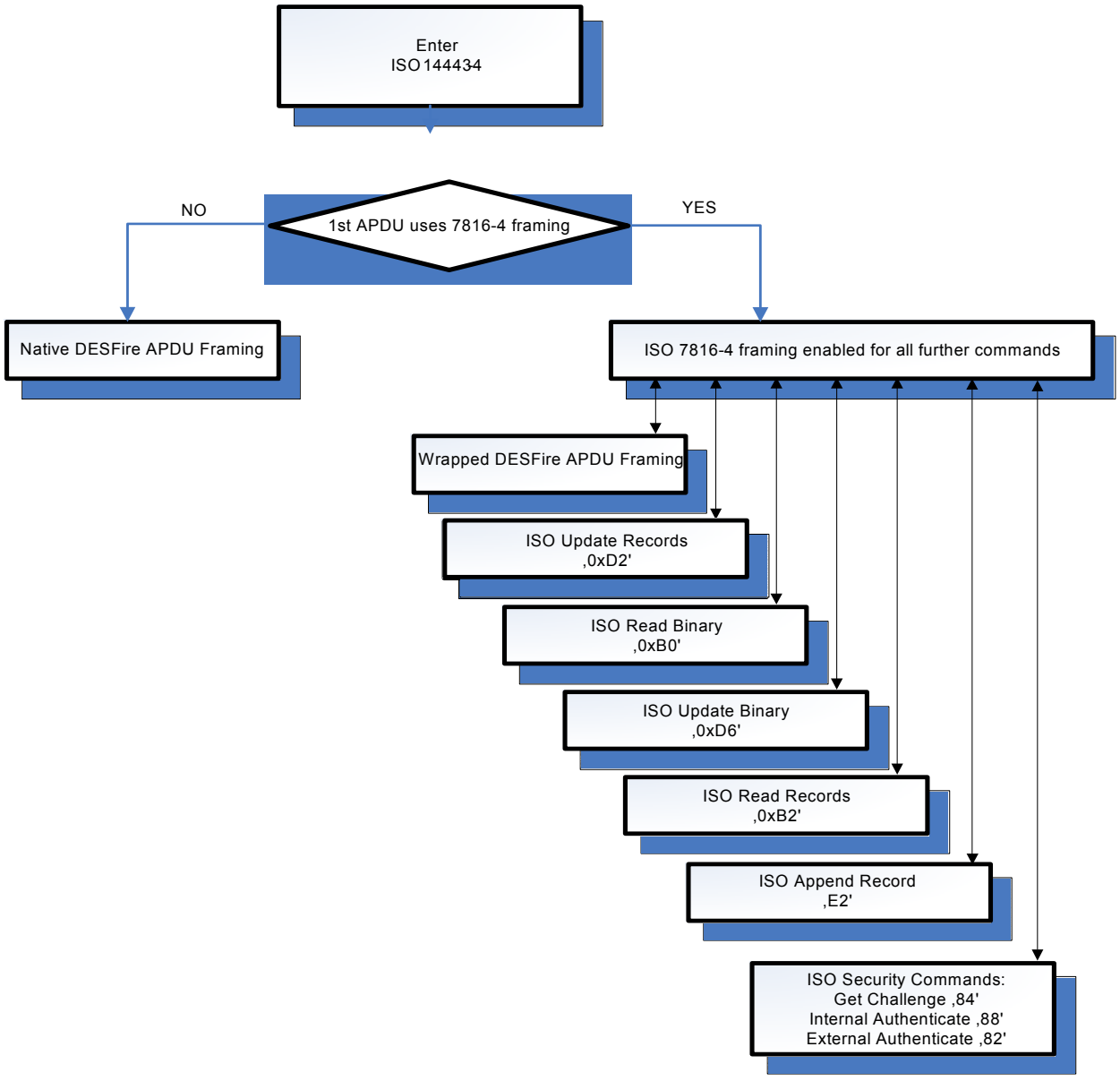

Fig 47. Selection of native DESFire APDU Framing versus ISO/IEC 7816-4 framing and commands

9.7.4 Wrapping of native DESFire APDUs

Wrapping of the native DESFire APDU Format is done as a “case 4” 7816-4 APDU message structure:

Table 27. Wrapping of native DESFire APDUs

CLA	INS	P1	P2	LC	Data	LE
0x90	DESFire CMD Code	0x00	0x00	Length of wrapped data	DESFire command parameter(s)	0x00

Remark: The LE byte shall be set to 0x00, which defines that any length of the PICC response is allowed. Other values than 0x00 are not allowed. The LE byte has to be always present.

Remark: The length of the total wrapped DESFire command is not longer then 55 byte long. This is change to the mode which is used in MF3ICD40, which can wrap longer DESFire commands into APDUs.

The response of DESFire is wrapped as follows:

Table 28. Response of DESFire i

Data	SW1	SW2	Remarks
DESFire Response Data	0x91	0xYY	The SW2 byte holds the Status byte of native DESFire

The first command after authentication defines, if wrapping is used or the commands are all sent without wrapping. Once wrapping is established after authentication, all further commands shall be wrapped or a ISO 7816-4 command, which is specified in this specification.

9.7.5 ISO/IEC 7816-4 format

MF3ICD81 uses for all ISO/IEC 7816-4 commands the following format:

Table 29. ISO/IEC 7816-4 format

CLS	INS	P1	P2	LC	Data	LE

Parameters 'CLS', 'INS', 'P1', 'P2' are detailed in every command description.

'Lc' states the length of the data parameter (max. 0xFF), it is only used if there exists a data in the command stream.

'Le' states the length of the data (max 0xFF), that should be sent back from the card.If 'Le'==0x00, then all data should be sent back, which is available. Exception is wrapping, see last section, where only ’0x00’ can be used. The length of the CMAC which can be used in the ’Read Binary’ and ’Read Records’ command shall be included in Le.

See more information on the principle structure of APDUs in ISO/IEC 7816-4.

9.7.6 Pre-Selection after entering 14443-4 (only in 7816-4 Framing mode)

If the first APDU after establishment of a successfully ISO/IEC 14443-4 communication, is compliant to ISO 7816-4 message format, an ISO SELECT File command, see Section 9.7.8, is required as first step in communication.

For legacy support, DESFire allows to skip this ISO SELECT File command, see Section 9.7.8, as the DESFire activates per default the DESFire AID, see Section 9.7.8.

For enhanced legacy support, DESFire allows to skip the ISO SELECT command.

Remark: For maximum compatibility with future versions of DESFire (or DESFire emulations on Dual Interface Smart Cards) we do NOT recommend to use this legacy features in newly designed applications.

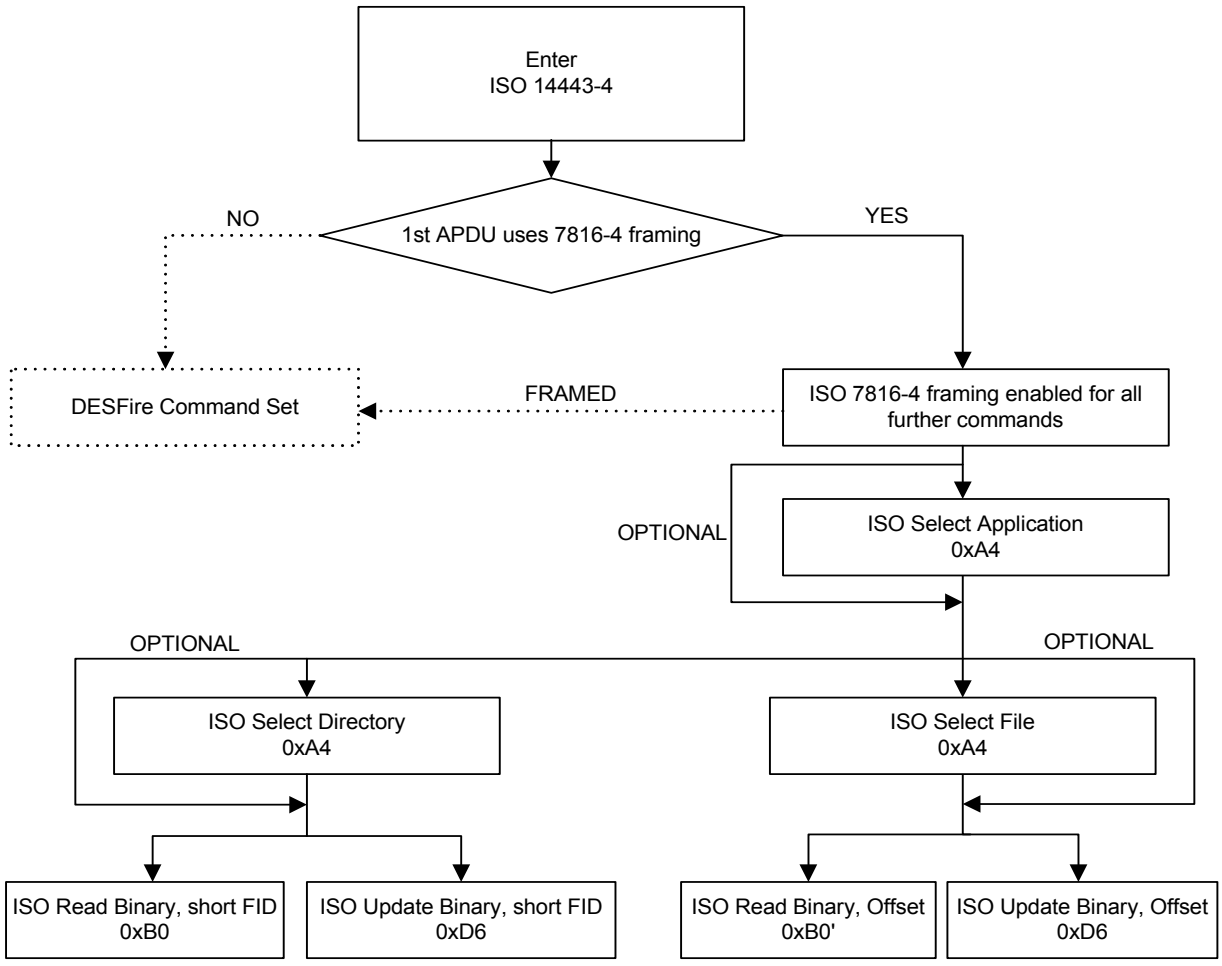

Fig 48. Pre-Selection after entering 14443-4

9.7.7 Security concept ISO/IEC 7816-4 commands

Table 30. Security concept ISO/IEC 7816-4 commands

Command	PCD		PICC
Get Challenge	-	→	
			Generate Random Number R PICC1
		←	RPICC1
	Generate Random Number: RPCD1 Generate Cryptogram: $e(R_{PCD1}\|\|R_{PICC1})$		
External Authenticate	Key Reference, $e(R_{PCD1}\|\|R_{PICC1})$	→	
			Verify R_{PICC1}Store R_{PCD1}
		←	Status of Verification
	Generate Random Number R_{PCD2}		
Internal Authenticate	Key Reference, R_{PCD2}	→	
			Generate Random Number: R_{PICC2}Generate Cryptogram: $e(R_{PICC2}\|\|R_{PCD2})$
		←	$e(R_{PICC2}\|\|R_{PCD2})$
	Verify R_{PCD2} Store R_{PICC2} Reset init vector Calculate session key.		

Key reference for EXTERNAL AUTHENTICATE and INTERNAL AUTHENTICATE needs to be identical.:

Single DES: 64bit session key required:

bit 0..4(R_{PCD1}) || bit 0..4(R_{PICC2})

2 Key Triple DES: 128bit session key required:

bit 1..32(R_{PCD1}) || bit 1..32(R_{PICC2}) || bit 33..64(R_{PCD1}) || bit 33..64(R_{PICC2})

If first half of the key is the same as for the second half, a single DES session key is used.

3 Key Triple DES: 192bit session key required:

bit 1..32(R_{PCD1}) || bit 1..32(R_{PICC2}) || bit 49..80(R_{PCD1}) || bit 49..80(R_{PICC2}) || bit 97..128(R_{PCD1}) || bit 97..128(R_{PICC2})

AES 128: 128bit session key required:

bit 1..32(bitR_{PCD1})|| bit 1..32(R_{PICC2})|| bit 97..128(R_{PCD1}) || bit 97..128(R_{PICC2})

Depending on the communication settings, only two modes are available after ISO/IEC 7816-4 authentication for ISO 7816-4 commands:

- Plain data transfer for all ISO/IEC 7816-4 commands
- Plain data transfer with cryptographic checksum
  - Only applicable for the commands 'Read Binary' and 'Read Record'
  - 8 byte CMAC (algorithm is the same as for native commands, (see Section 7.3.4), the CMAC is calculated over data and the SW2 (in this order))
  - Update Binary, Update Record and Append Record can only be carried out in plain. If the communication settings of the files are not set for plain communication, and these commands are sent to the PICC, an error will be sent back.

Wrapping of native commands can be also carried out in this mode (see Section 9.6.4)

Wrapping of native DESFire APDUs:

- After wrapped authentication with 0x0A Authenticate, the ISO 7816 commands is secured with a 4 byte MAC (see Section 7.2.4) for backward compatibility reasons. The MAC is calculated only over data field of the APDU.
- After wrapped authentication with 0x1A ISO authenticate or 0xAA AES authenticate, the ISO 7816 commands are secured with a 8 byte CMAC. The CMAC is calculated over the the whole APDU except the CMAC itself, (see Section 7.3.4)

9.7.8 ISO SELECT FILE command

This command is implemented in compliance with ISO/IEC 7816-4.

The registered ISO AID of DESFire itself is ”0xD2 76 00 00 85 01 00”. When selecting this application, the MF is selected. This is done in order to ensure compliance with future µc cards having a DESFire emulation inside.

Table 31. Detailed Description of the APDU Fields

Byte	Data	Remarks
INS	‘A4’	
P1	Selection Control	Selection by File Identifier: 0 … Select MF, DF or EF 2 … Select EF under current DF Selection by DF name: 4 … Directory Selection by DF name
P2	Option	File Control information option: Return FCI, optional template
L_c	Length of Data	Length of subsequent data filed
Data	Reference	Acc. To P1 – P2
L_e	Length of Response	Empty or length of expected response

Option:

- P2 = ’00’; the FCI stored in the file with ID 31 shall be returned
- P2 = ’0C’; no FCI shall be returned

For ISO SELECT FILE “00 A4 00 00” or “00 A4 00 02 3F 00” the masterfile (MF) will be selected.

As the FID is written with a native command (see Section 9.3.1) and the FID is read out with a ISO/IEC 7816-4 command, the different byte order needs to be taken into account.

Remark: If the FCI shall be returned, the file with ID 31 has to have FREE read access. The file with ID 0x31 is a standard data file and includes the whole data of the FCI, which is sent back if the application is selected with the 'Select DF' command.There is no specific FCI template format checked, the data which is stored in the file will be sent back without any check from OS to the terminal.

Table 32. Response APDU

Data	SW1	SW2	Remarks
-	0x67	0x00	wrong length
Optional: FCI stored in File 0x31 of the DF	0x90	0x00	correct execution
-	0x6A	0x82	application not found, currently selected application remains selected.
-	0x6A	0x86	wrong parameters P1 and/or P2
-	0x6A	0x87	wrong parameters Lc inconsistent with P1-P2
-	0x6F	0x00	no precise diagnostics

9.7.9 ISO READ BINARY command

This command is implemented in compliance with ISO/IEC 7816-4.

Table 33. Detailed description of the APDU Fields

Byte	Data	Remarks
INS	'B0'	
P1 / P2	Short File ID /	If bit 8 is set to zero in P1, then P1-P2 code the offset from zero to 32767 (fifteen bits).
	Offset	If bit 8 is set to one in P1, then bits 7 to 6 of P1 are set to 0 (RFU bits), bits 5 to 1 of P1 code a short ISO FID and P2 codes the offset from zero to 255 (eight bits).
L_E	Bytes to Read	The number of bytes to read from the file. The length of the CMAC/MAC (depending on communication settings) shall be included in this value. If L_E is '08' and a CMAC will be included in the answer, the whole content of the file will be sent back. If L_E is '04' and a MAC will be included in the answer, the whole content of the file will be sent back.

Table 34. Response APDU

Data	SW1	SW2	Remarks
-	0x62	0x82	end of file reached before reading L_E bytes
-	0x67	0x00	wrong length
-	0x69	0x82	file access not allowed
-	0x69	0x85	file empty
Data	0x90	0x00	command OK
Data || CMAC	0x90	0x00	command OK, communication secured with CMAC
-	0x6A	0x86	wrong parameter P1 and/or P2
-	0x6A	0x82	wrong parameter P1 and/or P2
-	0x6C	0x00	file not found
-	0x6F	0x00	no precise diagnostics

After authentication the returned stream will be secured by a 8 byte CMAC, if the communication settings are accordingly. If no authentication was performed/needed, no CMAC will be added.

9.7.10 ISO UPDATE BINARY command

This command is implemented in compliance with ISO/IEC 7816-4, the command is only possible with plain communication.

If a “Backup Data File” is used, the DESFire issues an implicit “Commit Transaction” command. This ensures that every update of a binary file is treated as a transaction.

Table 35. Detailed description of the APDU Fields

Byte	Data	Remarks
INS	‘D6’	
P1 / P2	Short File ID /	If bit 8 is set to zero in P1, then P1-P2 code the offset from zero to 32767 (fifteen bits).
	Offset	If bit 8 is set to one in P1, then bits 7 to 6 of P1 are set to 0 (RFU bits), bits 5 to 1 of P1 code a short ISO FID and P2 codes the offset from zero to 255 (eight bits).
L_c	Bytes to Write	The number of bytes to write to the file.
Data	Data	String of data units to be updated

Table 36. Response APDU

Data	SW1	SW2	Remarks
	0x90	0x00	command OK
-	0x69	0x82	file access not allowed
-	0x65	0x81	memory failure (unsuccessful updating)
-	0x67	0x00	wrong length
-	0x6A	0x86	file not found
-	0x6A	0x86	wrong parameter P1 and/or P2
-	0x6A	0x82	wrong parameter P1 and/or P2
-	0x6F	0x00	no precise diagnostics

9.7.11 ISO READ RECORD(S) command

This command is implemented in compliance with ISO/IEC 7816-4, only plain communication or communication secured with MAC is possible.

Table 37. Detailed description of the APDU Fields

Byte	Data	Remarks
INS	'B2'	
P1	Record number	Number of first record to be read; 00 indicates the last written record
P2	Reference Control	Bit 8 – bit 4: short EF identifier or '0' == currently selected file Bit 3 – bit 1: Usage of record number in P1: 100 … Read record P1 101 … Read all records from P1 to last
L_E	Bytes to Read	The number of bytes to read can be only a multiple of the record size. The length of the CMAC (depending on communication settings) shall not be included in this value. If L_E is '08' and a CMAC will be included in the answer, the whole content of the record will be sent back. If L_E is '04' and a MAC will be included in the answer, the whole content of the record will be sent back.

Table 38. Response APDU

Data	SW1	SW2	Remarks
-	0x67	0x00	wrong length
-	0x69	0x82	file access not allowed
-	0x69	0x85	Access conditions not satisfied
Data	0x90	0x00	command OK
Data || CMAC	0x90	0x00	command OK, communication secured with MAC.
-	0x6A	0x82	file not found
-	0x6A	0x86	wrong parameter P1 and/or P2
-	0x6B	0x00	wrong parameter P1 and/or P2
-	0x6F	0x00	no precise diagnostics

After authentication the returned stream will be secured by a 8 byte CMAC. If no authentication was performed/needed, no MAC will be added.

9.7.12 ISO APPEND RECORD command

This command is implemented in compliance with ISO/IEC 7816-4, only plain communication is possible.

If a "Backup Data File" is used, the DESFire issues an implicit "Commit Transaction" command. This ensures that every update of a binary file is treated as a transaction.

Table 39. Detailed description of the APDU Fields

Byte	Data	Remarks
INS	‘E2’	
P1	‘00’	
P2	Reference Control	Bit 8 – bit 4: short EF identifier or ‘0’ == currently selected file Bit 3 – bit 1: ‘000’
L_c	Length of Data	
Data	Record to be appended	
L_e	-	

Table 40. Response APDU

Data	SW1	SW2	Remarks
-	0x69	0x82	file access not allowed
Data	0x90	0x00	command OK
-	0x6A	0x86	file not found
-	0x6A	0x86	wrong parameter P1 and/or P2
-	0x6A	0x82	wrong parameter P1 and/or P2
-	0x6F	0x00	wrong length
-	0x6F	0x00	no precise diagnostics

9.7.13 ISO GET CHALLENGE command

This command is implemented in compliance with ISO/IEC 7816-4.

Table 41. Detailed Description of the APDU Fields

Byte	Data	Remarks
INS	‘84’	
P1	‘00’	
P2	‘00’	
L_E	Length of Challenge expected	RPICC1: 2 Key Triple DES:64bit required 3 Key Triple DES:128bit required AES 128:128bit required

Table 42. Response APDU

Data	SW1	SW2	Remarks
-	0x67	0x00	wrong length
-	0x69	0x82	file access not allowed
Challenge: RPICC1	0x90	0x00	command OK
-	0x6A	0x82	file not found
-	0x6A	0x86	wrong parameter P1 and/or P2
-	0x6C	0x00	wrong Le field
-	0x6E	0x00	wrong CLA
-	0x6F	0x00	no precise diagnostics

9.7.14 ISO EXTERNAL AUTHENTICATE command

This command is implemented in compliance with ISO/IEC 7816-4.

Table 43. Detailed description of the APDU Fields

Byte	Data	Remarks
INS	'82'	
P1	Reference to algorithm	'00' .. no information given, algorithm is defined by context '02' .. 2 Key Triple DES – CBC mode '04' .. 3 Key Triple DES – CBC mode '09' .. AES 128 – CBC mode
P2	Reference to secret	Bit 8: '0' .. PICC master key used (only available if current AID context is '3F00') '1' .. DF specific key used Bit 4.. Bit 1: Key reference number '0' .. 'D'; Bit 5=0
L_c	Length of Data Field	2 Key Triple DES: 128bit 3 Key Triple DES: 256bit AES 128:256bit
Data	Response to Challenge	e(RPCD1||RPICC1) Random number RPCD1: 2 Key Triple DES:64bit required 3 Key Triple DES:128bit required AES 128:128bit required
L_e	-	

Table 44. Response APDU

Data	SW1	SW2	Remarks
-	0x67	0x00	wrong length
-	0x69	0x82	file access not allowed
-	0x90	0x00	command OK, RPICC1 in e(RPCD1||RPICC1) correctly verified
-	0x6A	0x82	file not found
-	0x6A	0x86	wrong parameter P1 and/or P2
-	0x6A	0x87	LC inconsistent with P1/P2
-	0x6D	0x00	Instruction not supported
-	0x6F	0x00	no precise diagnostics

9.7.15 ISO INTERNAL AUTHENTICATE command

This command is implemented in compliance with ISO/IEC 7816-4.

PCD needs to verify R_{PCD2} in e(R_{PICC2}||R_{PCD2}) to release security status.

Table 45. Detailed description of the APDU Fields

Byte	Data	Remarks
INS	'88'	
P1	Reference to algorithm	'00' .. no information given, algorithm is defined by context '02' .. 2 Key Triple DES – CBC mode '04' .. 3 Key Triple DES – CBC mode '09' .. AES 128 – CBC mode
P2	Reference to secret	Bit 8: '0' .. PICC master key used (only available if current AID context is '3F00') '1' .. DF specific key used (AID is not '3F00') Bit 5 .. Bit 1: Key reference number '0' .. 'D'
Lc	Length of Data Field	
Data	Challenge	Random number RPCD2: 2 Key Triple DES:64bit required 3 Key Triple DES:128bit required AES 128:128bit required
Le	Expected Length of Response to Challenge	2 Key Triple DES:128bit 3 Key Triple DES:256bit AES 128:256bit

Table 46. Response APDU

Data	SW1	SW2	Remarks
-	0x67	0x00	wrong length
-	0x69	0x82	file access not allowed
Response to challenge: e(RPICC2||RPCD2)	0x90	0x00	command OK
-	0x6A	0x82	file not found
-	0x6C	0x00	wrong length (Le wrong)
-	0x6C	0x00	wrong parameter P1 and/or P2
-	0x6F	0x00	no precise diagnostics

10. Revision history

Table 47. Revision history

Document ID	Release date	Data sheet status	Change notice	Supersedes
134035	20081128	Product data sheet		134034
Modifications:	• Changed SW minor version to '3' in document Section 9.4.7 on page 55. Product did not change.			
134034	20080930	Product data sheet		134033
Modifications:	• Remove 'string' from first sentence in Section 7.3.6 on page 15. • Rephrased to 'Communication between PICC and PICD can be shown as follows' Section 7.1 on page 7 • Removed 'Replay Attack' from Section 2.5 • Rephrased the usage of init vector in Section 7.3.4 • Removed 'Write Data' as exception for padding in Section 7.2.6 • Removed 'Write Data' as exception for padding in Section 7.3.6 • Security status: added "strictly confidential information"			
134033	20080716	Product data sheet		134032
Modifications:	• Added the sentence "If L_E is '08' and a CMAC will be included in the answer, the whole content of the file will be sent back. If L_E is '04' and a MAC will be included in the answer, the whole content of the file will be sent back." in Section 9.7.9 on page 99, for the L_E. • Added the sentence "If L_E is '08' and a CMAC will be included in the answer, the whole content of the record will be sent back. If L_E is '04' and a MAC will be included in the answer, the whole content of the record will be sent back." in Section 9.7.11 on page 101, for the L_E. • The SAK in Figure 6 on page 28 is changed from '04' to '24' • The SAK in Table 20 on page 27 is changed from '04' to '24'			
134032	20080602	Product data sheet		134031

Table 47. Revision history *...continued*

Document ID	Release date	Data sheet status	Change notice	Supersedes
Modifications:	• Rephrasing of clauses • Added the sentence “The CMAC secures the communication,...” in Section 7.3.2 on page 13 • Change of “dechiphers” in “enchipers” in Section 7.2.7 on page 12 • Change of value from “0x6D” in “0x54” in Table 14 on page 24 • Added the sentence “This is the authentication,..” in Section 9.3.1 on page 31 • Change of value “00” of byte 5 from minor version in “04” on page 56 in Section 9.4.7 “GetVersion [1 byte]” • Change of value from “0x2406” in “0x3606” in Footnote on page 56 • Removed the sentence “This command is only available,...” in Section 9.4.10 on page 59 • Removed the sentence in “The last paramater is optional ...” on page 69 in Section 9.5.8 on page 69 • Remove of “ISO Update Record 0xD2” in Figure 47 on page 93 • Removed the word “not” in Section 9.7.5 on page 94 • Added the sentence “If first half of the key,..” in Section 9.7.7 on page 96 • Change the file number range for cyclic record files to 0x1F in Section 9.5.9 on page 71 • Include SW1=0x6A SW2=0x87 in Section 9.7.14 on page 103 • Some spelling and format mistakes in Section 9.3.4 on page 37 • Remove ’RndAccess’ in the figure Figure 33 on page 69 • Include the CMAC in the valid answer in Section 9.7.11 on page 101 • Include the CMAC in the valid answer in Section 9.7.9 on page 99			
134031	11 March 2008	Product data sheet	-	134030
Modifications:	• Added the remark “If the ISO 7816-4- wrapping is used...” in Section 9.7.2 “DESFire implementation of ISO/IEC 7816-4” on page 92			
134030	7 February 2008	Product data sheet		134020
Modifications:	• Change of product status • General update			
134020	23 January 2008	Preliminary data sheet		-
	• Initial version			

11. Legal information

11.1 Data sheet status

Document status[1][2]	Product status[3]	Definition
Objective [short] data sheet	Development	This document contains data from the objective specification for product development.
Preliminary [short] data sheet	Qualification	This document contains data from the preliminary specification.
Product [short] data sheet	Production	This document contains the product specification.

[1] Please consult the most recently issued document before initiating or completing a design.

[2] The term 'short data sheet' is explained in section "Definitions".

[3] The product status of device(s) described in this document may have changed since this document was published and may differ in case of multiple devices. The latest product status information is available on the Internet at URL http://www.nxp.com.

11.2 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

Short data sheet — A short data sheet is an extract from a full data sheet with the same product type number(s) and title. A short data sheet is intended for quick reference only and should not be relied upon to contain detailed and full information. For detailed and full information see the relevant full data sheet, which is available on request via the local NXP Semiconductors sales office. In case of any inconsistency or conflict with the short data sheet, the full data sheet shall prevail.

11.3 Disclaimers

General — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in medical, military, aircraft, space or life support equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Limiting values — Stress above one or more limiting values (as defined in the Absolute Maximum Ratings System of IEC 60134) may cause permanent damage to the device. Limiting values are stress ratings only and operation of the device at these or any other conditions above those given in the Characteristics sections of this document is not implied. Exposure to limiting values for extended periods may affect device reliability.

Terms and conditions of sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, including those pertaining to warranty, intellectual property rights infringement and limitation of liability, unless explicitly otherwise agreed to in writing by NXP Semiconductors. In case of any inconsistency or conflict between information in this document and such terms and conditions, the latter will prevail.

No offer to sell or license — Nothing in this document may be interpreted or construed as an offer to sell products that is open for acceptance or the grant, conveyance or implication of any license under any copyrights, patents or other industrial or intellectual property rights.

Quick reference data — The Quick reference data is an extract of the product data given in the Limiting values and Characteristics sections of this document, and as such is not complete, exhaustive or legally binding.

11.4 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

MIFARE — is a trademark of NXP B.V.

DESFire — is a trademark of NXP B.V.

12. Contact information

For additional information, please visit: **http://www.nxp.com**

For sales office addresses, send an email to: **salesaddresses@nxp.com**

13. Tables

continued >>

14. Figures

continued >>

15. Contents

continued >>

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 28 November 2008
Document identifier: 134035