MF2DL(H)x0

MIFARE DESFire Light contactless application IC

Rev. 3.3 — 5 April 2019
430733

Product data sheet
COMPANY PUBLIC

1 General description

1.1 Introduction

MIFARE DESFire Light (MF2DL(H)x0) is a versatile contactless smart card platform serving the requirements of applications managed by one single entity. Offering a powerful mix between performance, security, privacy and flexibility. It addresses the needs of limited use and simple extended use applications. Based on these parameters MIFARE DESFire Light is a trusted platform targeting the secure authentication of people with an intuitive convenient user experience.

MIFARE DESFire Light is fully compliant with the contactless proximity smart card protocol according to ISO/IEC 14443-4 and ISO/IEC 7816-4 communication frames making it compatible with the majority of existing contactless infrastructure devices and with NFC devices, such as NFC enabled mobile handsets. Its contactless performance supports superior user convenience and reading distances up to 10 cm.

MIFARE DESFire Light has a file-based memory structure compliant to ISO/IEC 7816-4 with a fixed, pre-defined configuration of six individual files (EF). The pre-defined configuration enables various use cases and allows the management of data according to best practice. Organized in one single directory (DF) and configurable access rights per file it enables different use cases of one issuing instance. MIFARE DESFire Light offers three individual standard data files with totally 544 bytes of memory for storage of application-specific data. The value file with a stored signed integer value and an upper and lower limit enables fast, flexible and secure implementation of monetary transactions, e.g. for micropayment applications. The cyclic record file with 4 entries of 16 bytes each enables an on-card logging of transactions.

As a contactless platform, MIFARE DESFire Light includes a powerful transaction management. This transaction management ensures data and transaction consistency supporting applications with the avoidance of disrupted or incomplete transactions. The optional Transaction Message Authentication (TMAC) further enables operators of, e.g., payment applications with a cryptographic checksum over the complete transaction enabling the verification of a transaction by a clearing entity.

MIFARE DESFire Light offers AES-based security features for authentication and data transfer over the contactless interface. The required level of security is defined by the needs of the application and can be done on a file basis. With 5 customer defined keys, MIFARE DESFire Light supports a key management addressing the organizational and security needs of the issuing entity.

Beside the standard AES implementation, MIFARE DESFire Light offers an alternative AES-based protocol for authentication and secure messaging using a Leakage Resilient Primitive, LRP. The LRP works as a wrapper around the AES cryptography and enhances side-channel and fault resistance.

MIFARE DESFire Light contains features like the fully encrypted communication mode enabling contactless applications to address privacy sensitive applications. With its optional support of Random ID, it enables compliance with latest user data protection regulations.

Users of MIFARE DESFire Light can change the application identifier (AID) and the file identifiers (FID) according to their needs enabling compatibility with existing data models. This further enables users to complement their use cases with an NFC forum-compliant Type 4 Tag in order to enable additional, end-user centric services, such as business card sharing or pairing with a network.

MIFARE DESFire Light is compatible with MIFARE DESFire EV2, a secure multi-application platform. Through this compatibility single application running on MIFARE DESFire Light can become part of a multi-application solution, combining applications from different entities, with minimal system impact.

MIFARE DESFire Light is designed to support standards Class 1 smart cards antenna designs with a 17 pF input capacitance as well as smaller form factors, i.e. key fobs, wristbands, by providing 50 pF input capacitance delivery forms. This ensures high user convenience throughout different form factors.

2 Features and benefits

2.1 RF Interface & Communication Protocol

- Contactless interface compliant to ISO/IEC 14443A-2/ -3/ -4, see [1] , [2], [3]
- Support of ISO/IEC 7816-4 communication frames for highest interoperability with mobile and wearables
- Low power consumption (Hmin) enabling operating distances of up to 10 cm
- Support of fast data rates: 106 kbit/s, 212 kbit/s, 424 kbit/s, and 848 kbit/s
- Support of double size (7-byte) Unique Identifiers (UID) and optionally Random ID (RID) according to ISO 14443-3 [2]
- Configurable communication frame size to support up to 128 bytes
- Fast start-up time for reliable and robust detection of MIFARE DESFire Light in legacy terminals
- Support of ISO 7816-4 wrapped commands compliant to a subset of MIFARE DESFire EV2 commands

2.2 Memory Organization

- 640 bytes user memory, equivalent to available user memory on legacy MIFARE Classic 1 kB product
- Data retention of 10 years and write endurance of minimum 200.000 cycles
- File system compliant to ISO/IEC 7816-4 with one predefined Directory File (DF) and a set of Elementary Files (EF)
  - Three standard data files, one with 32 bytes and two with 256 bytes
  - One cyclic record file of 4 records of each 16-byte record size
  - One value file for value operations including upper and lower limits
  - Optional Transaction Message Authentication Code (TMAC) file for transaction protection
- File system compliant with MIFARE DESFire EV2 file system

- User configurable file naming enabling compatibility to legacy systems and NFC Type 4 Tag compliant configurations

2.3 Security and Privacy

- Common Criteria certification: EAL4 for both Hardware and Software
- Secure messaging compliant to standard AES according to NIST Special Publication 800-38A and 800-38B [5] [6]
  - Optional enhanced side channel attack protection using LRP wrapped AES operation according to [10]
  - Secure messaging compatible with a subset of MIFARE DESFire EV2 secure messaging
- Five customer defined AES 128-bit keys including key versions
- Optional Random ID for enhanced privacy
- Individual AES 128-bit TMAC key for enhanced transaction protection
- Transaction counter limit to limit the number of transactions with the application
- 3-pass mutual authentication
- Flexible access control configurable per file (EF)
  - Individual key configuration for Read (R) / Write (W) / ReadWrite (RW) / Configuration
- Configurable secure messaging communication mode
  - Plain communication
  - CMAC protected for message integrity protection
  - Full Enciphered plus CMAC for full encryption of complete data transferred through contactless interface
- ECC-based NXP originality signature
- AES-based originality keys leveraging the LRP wrapped AES authentication

2.4 Specific Features

- Transaction-oriented automatic anti-tearing mechanism
- Configurable ATS information for card personalization
- Functional compatibility with MIFARE DESFire EV2 for easy integration of applications designed on MF2DL(H)x0 into flexible multi-application solutions
- High input capacitance (50 pF) for small form factor design available

3 Applications

MIFARE DESFire Light can be used in various contactless applications, some target applications are mentioned below.

- Access management
- Event ticketing
- Transport ticketing
- Account based services
- Electronic voucher
- Gaming
- Loyalty cards

4 Ordering information

Table 1. Ordering information

Type number	Package	Description	Version
MF2DL1001DUD/02	FFC	8 inch wafer (sawn; 120 μm thickness; Au Bumps) [1] [2] ; 640 Byte User Memory, 17 pF input capacitance	-
MF2DLH1001DUD/02	FFC	8 inch wafer (sawn; 120 μm thickness; Au Bumps) [1] [2] ; 640 Byte User Memory, 50 pF input capacitance	-
MF2DL1001DUF/02	FFC	8 inch wafer (sawn; 75 μm thickness; Au Bumps) [1] [2] ; 640 Byte User Memory, 17 pF input capacitance	-
MF2DLH1001DUF/02	FFC	8 inch wafer (sawn; 75 μm thickness; Au Bumps) [1] [2] ; 640 Byte User Memory, 50 pF input capacitance	
MF2DL1000DA8/02	MOA8	plastic leadless module carrier package [3] ; 640 Byte User Memory, 17pF input capacitance	SOT500-4
MF2DLH1000DA8/02	MOA8	plastic leadless module carrier package [3] ; 640 Byte User Memory, 50 pF input capacitance	SOT500-4
MF2DL1000DA4/02	MOA4	plastic leadless module carrier package [4] ; 640 Byte User Memory, 17 pF input capacitance	SOT500-2
MF2DLH1000DA4/02	MOA4	plastic leadless module carrier package [4] ; 640 Byte User Memory, 50 pF input capacitance	SOT500-2

[1] Delivered on film frame carrier with electronic fail die marking according to SECSII format.
[2] See [13]
[3] see for MOA8 Figure 45
[4] see for MOA4 Figure 46

5 Quick reference data

Table 2. Quick reference data

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
C_i	input capacitance		[1]	15.3 45.0	17 50	18.7 55.0	pF pF
f_i	input frequency			-	13.56	-	MHz
EEPROM characteristics							
t_{ret}	retention time	T_{amb} = 22 °C		10	-	-	year
$N_{endu}(W)$	write endurance [2]	T_{amb} = 22 °C		200.000	-	-	cycle
$t_{cy}(W)$	write cycle time	T_{amb} = 22 °C		-	1	-	ms

[1] T_{amb} = 22 °C; f_i = 13.56 MHz; 2 V RMS

[2] Write endurance of a single EEPROM cell

6 Block diagram

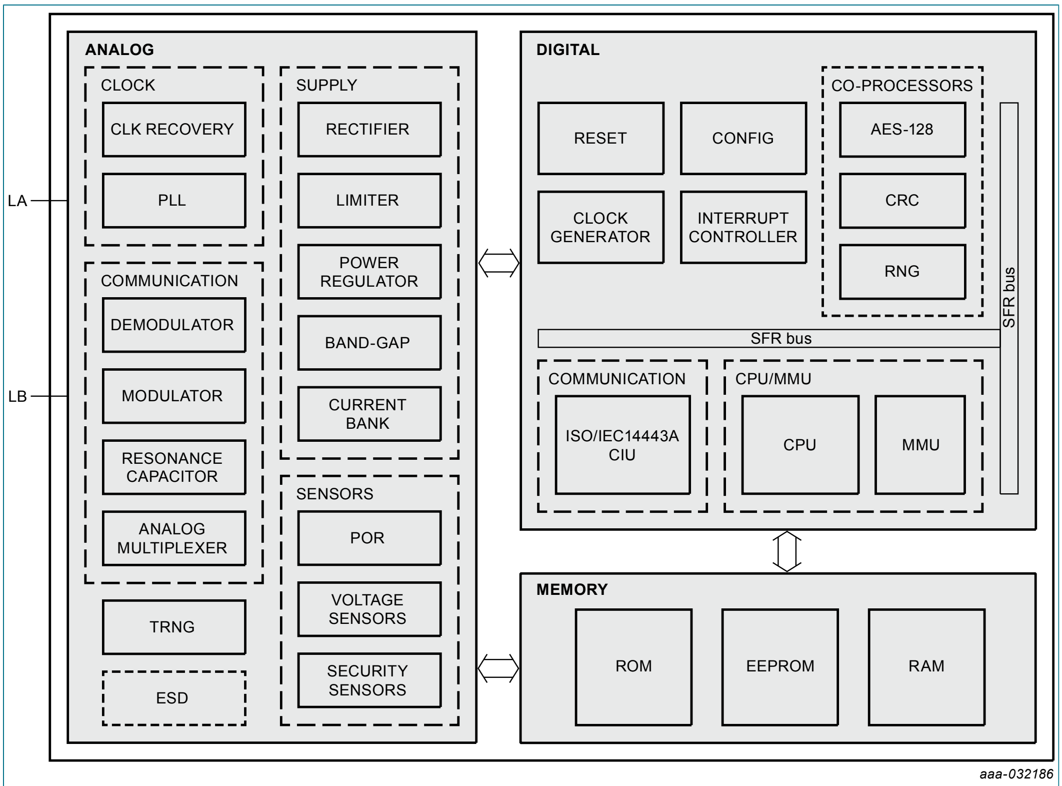


Figure 1. MIFARE DESFire Light IC blocks

7 Pinning information

7.1 Pinning

The pinning for the MF2DL(H)x0 is shown in Figure 2 for a contactless MOA8 module.

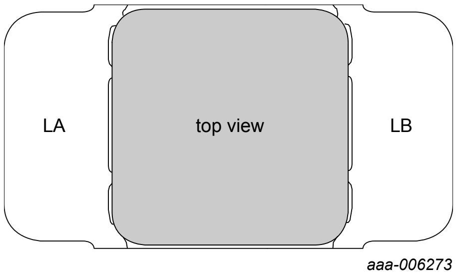


Figure 2. Pin configuration for SOT500-4 (MOA8)

The pinning for the MF2DL(H)x0 is shown in Figure 3 for a contactless MOA4 module.

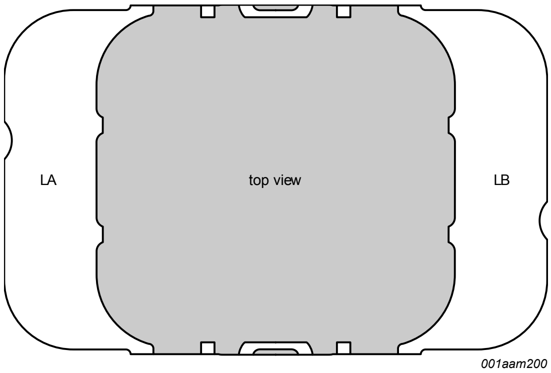


Figure 3. Pin configuration for SOT500-2 (MOA4)

Table 3. Pin allocation table

Pin	Symbol	
LA	LA	antenna coil connection LA
LB	LB	antenna coil connection LB

8 Functional description

8.1 Interface initialization and protocol

MF2DL(H)x0 is fully compliant to ISO/IEC 14443-2 [1] radio frequency power and signal interface, the initialization and anti-collision is according to ISO/IEC 14443-3 [2] and it uses transmission protocol as specified in ISO/IEC 14443-4 [3] of PICC Type A.

8.1.1 Default ISO/IEC 14443 parameter values

This section describes the default values for ISO/IEC 14443 activation and selection. Usage of Random ID and SAK, ATS values can be changed using the SetConfiguration command.

Note, that any change in the ISO/IEC 14443 parameter values through SetConfiguration requires a power cycle to make those changes effective.

ATQA

ATQA value is 0344h, which denotes double size (7-byte) UID. However, MF2DL(H)x0 offers configuration of Random ID which is single size (4-byte). If the Random ID feature is enabled, then the ATQA is changed to 0304h. According to ISO/IEC 14443-3, the ATQA bytes are transmitted as LSB first.

SAK

For double size UID, the default value of SAK1 in cascade level 1 is 04h, indicating that the UID is not complete. SAK2 in cascade level 2 is 20h, indicating UID complete and supporting ISO/IEC 14443-4. For single size UID which is used in the Random ID case, the value of SAK is 20h, indicating UID complete and supporting ISO/IEC 14443-4.

UID

The ISO/IEC 14443-3 compliant UID is programmed and locked during production. The first byte of the double size UID is fixed to 04h, indicating NXP as manufacturer.

ATS

The default value of the ATS of MF2DL(H)x0 is as follows:

Table 4. Default ATS value

ATS Parameter	Default Value	Comment
TL	06h	Length of ATS
T0	77h	TA(1), TB(1), TC(1) present in ATS and frame size is 128 bytes
TA(1)	77h	Different communication speed can be set in each direction supports communication speeds 212, 424, 848 kbps in both directions
TB(1)	71h	Max frame waiting time is 38.66 ms, start frame guard time is 604 μs
TC(1)	02h	CID supported
T1	80h	Historical byte

8.1.2 Setting of higher communication speed

After receiving an ATS, a PPS request can be sent to the MF2DL(H)x0 to set up a higher communication speed up to 848 kbit/s according to ISO/IEC 14443-4 [3].

8.1.3 Half-duplex block transmission protocol

MF2DL(H)x0 uses half-duplex block transmission protocol as specified in ISO/IEC 14443-4. It is fully compliant to block format, frame waiting time, frame waiting time extension, protocol operation, and all rules or handling as in [3].

8.2 User memory

The file system in the user memory is according to ISO/IEC 7816-4 and shown in Figure 4. In the DF (application), there are 6 EFs (files) and 5 keys.

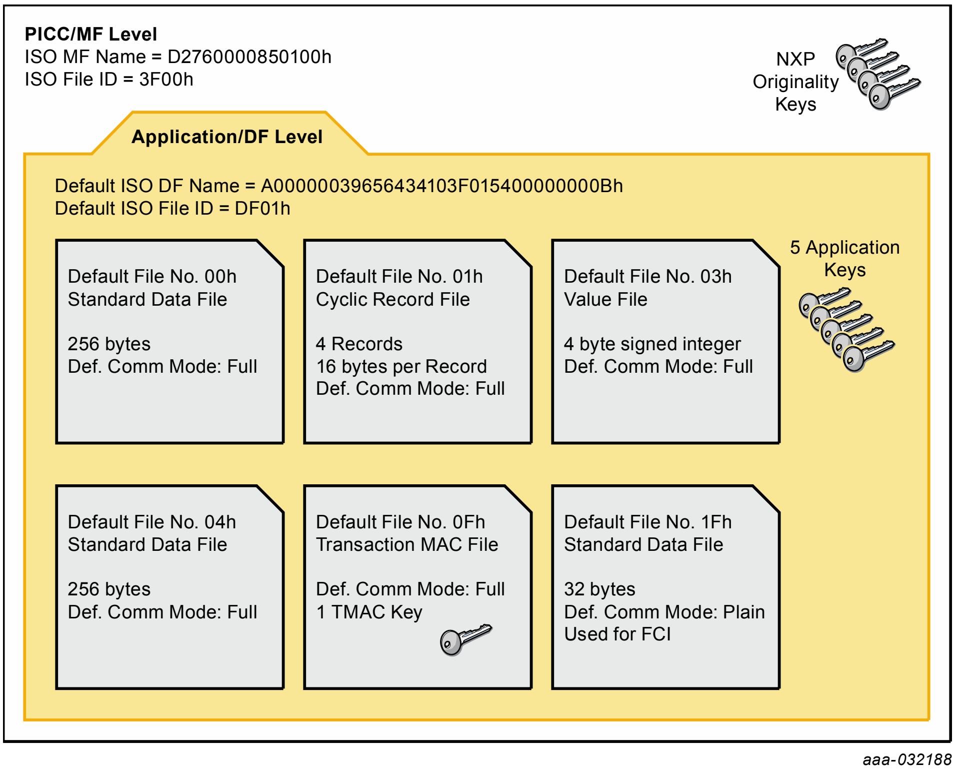


Figure 4. MIFARE DESFire Light application

8.2.1 Application and file selection

The MF (PICC Level), the DF (application) and the EFs (files) can be selected using the ISOSelectFile command specified in ISO/IEC 7816-4 [4] and described in Section 11.10.1.

The PICC level refers to the card itself and has the ISO DF Name D2760000850100h and the File ID 3F00h. The only functionality available on PICC level is an LRP mode authentication using an OriginalityKey and the retrieval of the originality signature using the Read_Sig command, see Section 11.11.1.

8.2.2 Default Application Name and ID

MF2DL(H)x0 is pre-configured with one application. The default DF name and File ID are as follows:

- DF Name: A00000039656434103F015400000000Bh
- File Identifier: DF01h

DF Name and File ID can be changed using SetConfiguration command.

8.2.3 Files

MF2DL(H)x0 stores user data into EF (files) of specific types. All files but the TransactionMAC File are statically created and cannot be deleted. However, file IDs and file settings can be changed using the SetConfiguration command. The ChangeFileSettings command can be used to change the access rights configuration. Any file number between 00h and 1Fh and any allowed File ID can be set. Note, that duplication of a file number or a file ID is not allowed.

Only the TMAC file can be deleted and re-created in order to generate a new TMAC configuration or disable the TMAC feature.

File access rights can be restricted with the keys of the application.

Table 5. File management

EF (File) Type	File type coding	Default file no.	Default file ID	File size	Example uses
StandardData file	00h	00h [1]	EF00h	256 bytes	Storage of raw data e.g. information
		04h	EF04h	256 bytes	Storage of raw data e.g. information
		1Fh[1]	EF1Fh	32 bytes	FCI template or data
Value file	02h	03h	-	-	Storage of value, points, customized counter
CyclicRecord file	04h	01h	EF01h	4 records á 16 bytes	Storage of log, activities
TransactionMAC file	05h	0Fh	-	-	Transaction counter, Transaction MAC, Reader ID, Limit of application selection.

[1] Note that this File no. needs to be explicitly selected via ISOSelectFile for ISOReadBinary or ISOUpdateBinary as it is reserved in the scope of implicit selection. The File no. can be changed in order to leverage implicit selection.

8.2.3.1 StandardData file

A StandardData file stores the data as raw data bytes. Data is accessed by chunk of byte at a certain offset in the StandardData file and with a certain byte length.

The content of file no. 1Fh is returned as FCI at Section 11.10.1 of the DF, if the read access of this file is set to "free". Note that also one of the larger files can be used to hold FCI information in case it is renamed to 1Fh.

A StandardData file can be read with the ReadData and ISOReadBinary commands. The data can be written with the WriteData and ISOUpdateBinary commands. ISOReadBinary and ISOUpdateBinary are standard ISO/IEC 7816-4 interindustry commands, see [4].

A StandardData file is not covered by the transaction mechanism and therefore does not implement a transaction-based backup mechanism. Thus data are available to read as soon as they are written in the file. The writing operations of single frames up to 128 bytes with a WriteData or ISOUpdateBinary command are also tearing protected.

8.2.3.2 Value file

The Value file stores the data as a 4 byte signed integer.

The Value file can be read with the GetValue command. The value can be manipulated with Credit, Debit and LimitedCredit commands, respectively.

The value is restricted by an upper and lower limit that the current value of the file cannot exceed.

The Value file can support a LimitedCredit feature as defined in Section 11.8.6. The free GetValue allows a user to get freely the value of a value file bypassing an authentication if necessary as long as there is one access condition associated with GetValue command that is different from "no access" condition, see Section 11.5.1

The Value file implements a backup mechanism. All operations executed on a value file within the current transaction will be effective only when a successful CommitTransaction will be executed. Meanwhile or if the transaction is aborted or not successfully committed, the previous value remains effective and is returned by a GetValue.

8.2.3.3 Record file

The record file is a CyclicRecord file which allows storing latest 4 records of 16-byte (each record) size.

The record file adds up new records when a successful WriteRecord command is executed. When the maximum number of records of the file is reached, the next new record overwrites the oldest record as defined Section 11.8.8.

The record file implements a backup mechanism. All operations executed on a record file within the current transaction are effective only when a successful CommitTransaction command is executed.

If the transaction is aborted or not successfully committed the previous file image is effective and returned by a ReadRecords command.

Within a record file, records are numbered starting with the latest record written. Meaning that the latest record written has the number 0, the second latest record written has the number 1 and so on. Its valid range is from 0 to the number of existing records

ExistingRecCount- 1 in the record file.

This indexing is used in the data management commands to address records within the Record file.

Note that when reading out records, the records are returned in the chronological order that they have been written, i.e. the oldest record is returned first.

8.2.3.4 TransactionMAC file

A TransactionMAC file is used to store Transaction MAC-related information. It consists of the following items:

- Transaction MAC Counter (TMC) 4 byte unsigned integer. The TMC is initialized to 00000000h on file creation. This value means that no Transaction MAC has been computed yet. Note that this value will never be used in a Transaction MAC computation.

- Transaction MAC Value (TMV) 8 bytes. The TMV is initialized to all zero bytes on file creation.

A TransactionMAC file is created with the CreateTransactionMACFile command, see Section 11.7.6.

A TransactionMAC file can be read with a ReadData command, protected by the Read access right, see below. The behavior is identical as reading a StandardData file of 12 bytes.

A TransactionMAC file cannot be written directly. The TMC and TMV are related with the latest successful transaction and are updated automatically on successful CommitTransaction command. In addition, the last calculated TransactionMAC values can be read out from the TransactionMAC file, in case the response to the CommitTransaction command has not been received correctly by the PCD. Therefore, the implementation of this file type supports the integrated backup mechanism for anti-tearing protected atomic update of these values together with the rest of the transaction updates.

In addition, the following content is stored in the TransactionMAC file:

- optionally the TransactionMAC ReaderID (TMRIPrev), see Section 11.9.3, as a committed value becomes valid together with the updated TMC and TMV on CommitTransaction.
- the AppTransactionMACKey, as it is specified when creating the TransactionMAC file.

However, these values are never immediately accessible on the command interface via the file (e.g. using ReadData).

8.2.3.5 File access rights management

File data is accessed with 3 different access rights Read, Write and ReadWrite.

In addition, an access right called Change is specified per file permitting ChangeFileSettings to change the file access rights.

An access right is granted if at least one condition associated to it is satisfied. Such conditions are called access conditions. Access conditions are associated with an access right and evaluated to decide whether the access right is granted or not. There are three kinds of access conditions:

- The access condition where a valid authentication with a given AppKey of the targeted application is needed to access related commands listed in Table 9. The access condition is satisfied if the current valid authentication has been performed with the given AppKey and not satisfied in all other cases. The AppKey is specified with its number.
- The free access condition meaning the related commands listed in Table 9 can be accessed without an active authentication.
- The no access condition meaning no access to related commands listed in Table 9.

The access conditions are specified on 4 bits as defined in the following table.

Table 6. Access condition

Condition value	Description
0h..4h	key number of a AppKey

Condition value	Description
Eh	free access
Fh	no access or RFU

The set of access conditions is coded on 2 bytes as shown in the following table. RFU access conditions are expected to be set to Fh (for future extensibility).

Table 7. Set of Access condition

Bit index	Description	Value
15..12	Read	access condition as in Table 6
11..8	Write	access condition as in Table 6
7..4	ReadWrite	access condition as in Table 6
3..0	Change	access condition as in Table 6

The default access conditions for the files in MF2DL(H)x0 is shown below in .

Table 8. Default file access rights

Default File No.	File Type	Read	Write	ReadWrite	Change
1Fh	StandardData File	Eh	Fh	3h	0h
00h	StandardData File	1h	Fh	3h	0h
04h	StandardData File	1h	2h	3h	0h
03h	Value File	1h	2h	3h	0h
01h	CyclicRecord File	1h	2h	3h	0h
0Fh	TransactionMAC File	1h	Fh	1h	0h

The mapping of access rights to applicable commands is shown in Table 9.

Table 9. Command list associated with access rights

Access Right	Command
Read	ReadData ISOReadBinary ReadRecords GetValue Debit
Write	WriteData ISOUpdateBinary WriteRecord GetValue Debit LimitedCredit

Access Right	Command
ReadWrite	ReadData (if not targeting the TransactionMAC file) ISOReadBinary (if not targeting the TransactionMAC file) ReadRecords WriteData ISOUpdateBinary WriteRecord UpdateRecord ClearRecordFile CommitReaderID GetValue Debit Credit LimitedCredit
Change	ChangeFileSettings

A command listed in Table 9 is accepted if at least one access condition associated with an access right granting access to it is satisfied. If authenticated and the only access conditions satisfied are the free access Eh ones, then the CommMode.Plain is to be applied.

8.2.3.6 TransactionMAC file access rights

The access rights management of a TransactionMAC file differs from the general access right management outlined in Section 8.2.3.5 for other file types. They are described in Table 10.

Note that Write is RFU and should to be set to Fh.

Table 10. TransactionMAC access rights

Access right	Bit index	Value	
Read	15-12	15-12 condition as in Table 6	
Write	11-8	RFU (i.e. Fh)	Write access is never granted to a TransactionMAC file
ReadWrite	7-4	AppCommitReaderIDKey Note that this access right has a specific meaning in the context of a TransactionMAC file as if defines the availability and configuration for the CommitReaderID command	
		0h..4h	CommitReaderID enabled, requiring authentication with the specified application key index
		Eh	CommitReaderID enabled with free access
		Fh	CommitReaderID disabled
Change	3-0	condition as in Table 6	

8.2.3.7 Communication modes

MF2DL(H)x0 supports three communication modes as defined in Table 11. As shown in the table, the different communication modes can be represented by two bits. This representation is used at several places in the document.

Table 11. Supported communication modes

Communication mode	Bit Representation	Explanation
CommMode.Plain	X0b	No protection: message is transmitted in plain text
CommMode.MAC	01b	MAC protection for integrity and authenticity
CommMode.Full	11b	Full protection for integrity, authenticity and confidentiality, also referred to as "Full Protection" mode

The communication mode defines the level of security for the communication between PCD and PICC after mutual authentication. At application level, the communication mode is defined by the command itself, as specified in Table 19. The specified communication mode is applied if there is an active authentication regardless of whether this authentication is required by the command or not.

At file level, the communication mode is defined by the file. The specified communication mode is applied if there is an active authentication. Note, if the only valid access condition for a certain access right is free access (Eh) under an active authentication, CommMode.Plain has to be applied, see also Section 8.2.3.5.

The commands for authentication have their own secure messaging rules, as indicated by N/A (not applicable) in Section 11.2. The ChangeKey command is always executed in Full Protection mode.

If there is no active authentication, the command and response are sent in plain (or the command is rejected in the case an authentication is required).

The default communication mode per file is shown in below.

Table 12. Default communication modes per file

EF (File) Type	Default file no.	Default communication mode
StandardData file	00h	CommMode.Full
	04h	CommMode.Full
	1Fh	CommMode.Plain
Value file	03h	CommMode.Full
CyclicRecord file	01h	CommMode.Full
TransactionMAC file	0Fh	CommMode.Full

8.2.4 Keys

Application keys and PICC keys and their usage are defined in Table 13 and Table 14 respectively. They are used to manage the security of the application.

Table 13. Keys at application level

Key Identifier	Key number	Change Key	Can be used for Authentication
Addressable keys:			
AppMasterKey	00h	AppMasterKey	yes
AppKey	00h..04h	AppMasterKey	yes
AppCommitReaderIDKey	00h..04h	AppMasterKey	yes
AppTransactionMACKey	N/A	N/A	no

8.2.4.1 AppMasterKey

The AppMasterKey always has the key number 00h. A successful authentication with the AppMasterKey is required to change any application key including the AppMasterKey itself with the ChangeKey command.

8.2.4.2 AppKey

The application of the MF2DL(H)x0 includes 5 AES 128-bit keys with key numbers 0, 1, 2, 3, 4. Furthermore, key number "E" (means free) and key number "F" (means never) can be assigned for access rights management.

The transport value of these 5 keys is 16 bytes of 00h, and can be changed by authentication with key number 0 in transport configuration.

Remark: It is highly recommended to change all 5 keys at personalization, even if not all keys are used in the application.

8.2.4.3 AppCommitReaderIDKey

The AppCommitReaderIDKey refers to one of the AppKey and is used for the Transaction MAC feature is described in Section 10.3. Its key number is specified in Section 8.2.3.6 and is assigned at creation time of the TransactionMAC file.

8.2.4.4 AppTransactionMACKey

The AppTransactionMACKey is not part of the application keys. Its use for the Transaction MAC feature is described in Section 10.3.

The AppTransactionMACKey is changeable through a TransactionMAC file deletion and re-creation which requires an active authentication with the AppMasterKey.

The AppTransactionMACKey is not available for authentication.

8.2.4.5 OriginalityKey

The authentication procedure for AES keys can be used to authenticate to one of the four OriginalityKey and check whether the PICC is a genuine NXP product. MF2DL(H)x0 supports targeting the OriginalityKey with the LRP authentication using AES. For details on the authentication command, see Section 9.2. The following variants can be used:

- AuthenticateLRPFirst, see Section 9.2.5
- AuthenticateLRPNonFirst, see Section 9.2.6

Table 14. Keys at PICC level

Key Identifier	Key number	Can be used for Authentication
Addressable keys:		
OriginalityKey1	01h	yes
OriginalityKey2	02h	yes
OriginalityKey3	03h	yes
OriginalityKey4	04h	yes

8.2.4.6 Key version

A 1-byte key version (00h to FFh) is assigned to each key. This key version can be used to distinguish the keys of a specific application if the same application uses different keys or key versions.

The key version is set with the ChangeKey command and at creation of a TransactionMAC file using the CreateTransactionMACFile command and can be retrieved using the GetVersion command.

8.3 Native Command Format

MF2DL(H)x0 always communicates in ISO/IEC 7816-4 wrapped mode as described in Section 8.4. Nevertheless it is important to understand the basic format of native commands as it is used in MIFARE DESFire EV2 which consist of the following parts.

A command as sent by the PCD consists of the concatenation of:

- the command code (Cmd)
- zero, one or more header fields (CmdHeader)
- zero, one or more data fields (CmdData)

The response as sent by the PICC consists of the concatenation of:

- the return code (RC)
- zero, one or more data fields (RespData)

MF2DL(H)x0 supports the APDU message structure according to ISO/IEC 7816-4 [4] for:

- wrapping of the native command format into a proprietary ISO/IEC 7816-4 APDU
- a subset of the standard ISO/IEC 7816-4 commands (ISOSelectFile, ISOReadBinary, ISOUpdateBinary)

Remark: Communication via native ISO/IEC7816-4 commands without wrapping is not supported.

On the native command interface, plain command parameters consisting of multiple bytes are represented least significant byte (LSB) first, similar as for ISO/IEC 14443 parameters during the activation, see [2]. For cryptographical parameters and keys (including the random numbers exchanged during authentication, the TI and the computed MACs), this does not hold. For these, the representation on the interface maps one-to-one to the most significant byte (MSB) first notation used in this specification.

Note that within this document, the ’Xh’ prefix indicates hexadecimal integer notation, i.e. not reflecting the byte order representation on the command interface at all.

8.4 ISO/IEC7816-4 Communication frame

MF2DL(H)x0 uses ISO/IEC 7816-4 [4] type APDUs for command-response pair for both, wrapping of native commands as outlined in Section 8.3 and standard ISO/IEC 7816-4 commands.

Note that for all parameters of standard ISO/IEC 7816-4 commands, the representation on the interface is most significant byte (MSB) first notation. As data like the 2-byte ISO/IEC 7816-4 file identifiers, are in different order for the wrapped native commands, this needs to be taken into account.

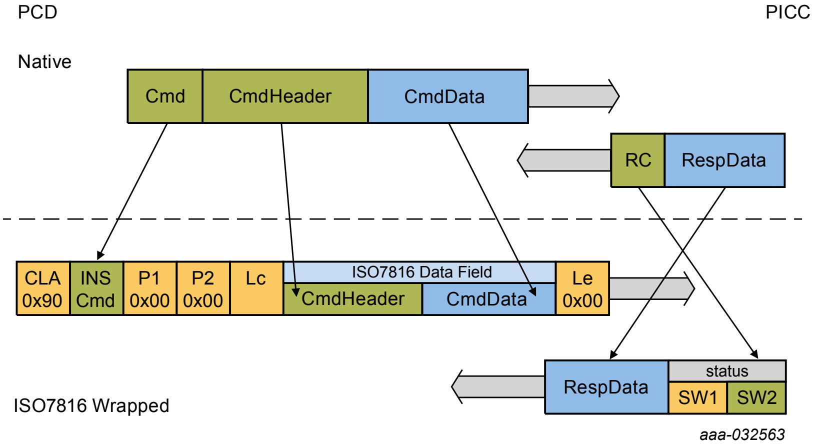


Figure 5. ISO/IEC 7816-4 command response pair

Table 15. ISO/IEC 7816-4 command fields

Field	Description	Length
Command header	Class byte (CLA)	1
	Instruction (INS)	1
	Parameters (P1,P2)	2
Lc field	Length of command data field (Lc), absent if no data field is present	1
Command data field	Absent if no data is sent in the command	Lc
Le field	Expected response length. If Le is 00h, then all available data is sent back for ISO/IEC 7816-4 standard commands. For wrapped commands, Le must always be set to 00h.	1

Note that the maximum number of bytes in the command data field, indicated by Lc, cannot exceed 255 bytes since only short length (1 byte) is supported in MF2DL(H)x0, see [4]. This includes overhead for secure messaging.

The maximum number of bytes in the response data field, indicated by Le, cannot exceed 256 bytes since only short length (1 byte) is supported in MF2DL(H)x0, see [4]. This includes overhead for secure messaging.

Table 16. ISO/IEC 7816-4 response fields

Field	Description	Length
Response data field	Response data if any, absent if no response data	up to Le
Response trailer	status byte (SW1SW2)	2

The field length and presence might vary for different commands, refer to the specific command description in Section 11.

8.5 Command Chaining

MF2DL(H)x0 supports standard ISO/IEC 14443-4 [3] command chaining in the following cases:

- the PICC supports ISO/IEC 14443-4 chaining to allow larger command or response frames than the supported buffer size for variants of the following commands:
  - Native commands wrapped into ISO/IEC 7816-4 APDU: ReadData, WriteData, ReadRecords, WriteRecord, and UpdateRecord, see Section 11.
  - Standard ISO/IEC 7816-4 commands: ISOReadBinary, ISOUpdateBinary i.e. every command where larger frame size can occur.
- the PICC will automatically split a response in several frames to fit with the FSD frame size supported by the PCD and communicated in the RATS.

When a PCD applies ISO/IEC 14443-4 chaining, see [3], it needs to assure the reassembled INF field containing the command header (i.e. ISO/IEC 7816-4 header bytes and/or (Cmd || CmdHeader)) fits within the PICC’s buffer (FSC) communicated in the ATS. If not, the PICC may respond with LENGTH_ERROR.

The ISO/IEC 14443-4 chaining does not influence the secure messaging. This means that the secure messaging mechanisms are applied as if the command or response would have been sent in a single large frame. With regards to command execution, commands are handled as if they were received in one large frame, except for write commands where the total frame size can be larger than the supported FSC (WriteData, WriteRecord, UpdateRecord and ISOUpdateBinary). In this case, command execution is started before the complete command is received.

Note that this is important for StandardData files, where the file may end up in an undefined state if the command is not completed successfully, see Section 8.6.

8.6 Backup management

MF2DL(H)x0 supports the implementation of transactions with tearing protection by its backup file management. The following file types are backup file types:

- Value
- CyclicRecord
- TMC and TMV inside the TransactionMAC file

All updates to these file types will be done only if CommitTransaction is executed successfully. If the transaction is teared, or an error occurs, before issuing CommitTransaction, none of the changes issued during the transaction so far will be applied.

If the card is teared during the CommitTransaction execution itself, the implementation will still ensure that either all, or none of the changes are applied. Note that it is possible that the reader will not receive the response to CommitTransaction despite a successful execution, i.e. when tearing on the response transmission. The chance for this to occur is limited by a short execution time of the CommitTransaction command. However, still some system level measures should be considered for this event.

Note that files of StandardData do not offer this backup mechanism. However, still a limited tearing protection is offered by ensuring that on single frame write operations the updated data is either completely written or not changed at all.

8.7 Product originality

MF2DL(H)x0 supports two types of originality check: one based on an LRP authentication with AES originality keys and another one with a static signature verification with an ECC public key. For detail of the ECC originality check, refer to originality check commands.

The AES-based originality verification using AuthenticateLRPFirst or AuthenticateLRPNonFirst is done with one of the OriginalityKey described in Section 8.2.4.

9 Secure Messaging

Prior to data transmission a mutual three-pass authentication can be done between PICC and PCD which results in the generation of session keys used in the secure messaging.

There are two secure messaging types available in the MF2DL(H)x0. One is using AES-128 and is referred to as AES mode in this data sheet. The other one is using AES-128 with a Leakage Resilient Primitive (LRP) wrapper, also referred to as LRP mode. The LRP mode can be permanently enabled using the SetConfiguration command. After this switch, it is not possible to revert back to AES mode.

Compared to AES mode, the LRP mode has the advantage that it provides strong resistance against side-channel and fault attacks. It serves as a replacement for AES as it only uses standard cryptographic constructions based on AES without any proprietary cryptography. Thus, LRP can be seen as an alternative for AES which is itself based on AES, and is provably as secure as AES, but comes with better properties w.r.t. implementation security, see also [10]. The PCD requires the same LRP mode implementation.

To improve the resistance against side-channel attacks and especially card only attacks for the AES mode, MF2DL(H)x0 provides a limit for unsuccessful authentication attempts. Every unsuccessful authentication is counted in the TotFailCtr. The parameters TotFailCtrLimit TotFailCtrDecr can be configured as described in Section 11.5.1 using the "Failed authentication counter configuration".

Each unsuccessful authentication is counted internally in the total failed authentication counter TotFailCtr. After reaching the TotFailCtrLimit, see Section 11.5.1, the related key cannot be used for authentication anymore.

In addition, after reaching a limit of *consecutive* unsuccessful authentication attempts, the MF2DL(H)x0 starts to slow down the authentication processing in order to hamper attacks. This is done by rejecting any authentication command with a AUTHENTICATION_DELAY response. The response time of a single AUTHENTICATION_DELAY response is depending on the FWT, see Section 8.1.1, and is about 65% of the maximum response time specified by FWT. The error response is sent until the total authentication delay time is reached which is equal to the sum of the frame delay times. The total authentication delay time increases with each unsuccessful authentication attempt up to a maximum value, only a successful authentication restores the full operational speed.

Changing a blocked AES key by authenticating with the AppMasterKey and using the ChangeKey command makes the referenced key accessible again. If the AppMasterKey itself is blocked, no recovery is possible.

Each successful AES authentication decrements the TotFailCtr by a value of TotFailCtrDecr.

The AES and LRP authentications are initiated by commands sharing the same command code (First Authentication and Non-First Authentication variants). These authentication protocols are both AES-based, but differ with regards to the actual protocol applied and the subsequent secure messaging mode they initiate. An overview of the different modes is given in Figure 6.

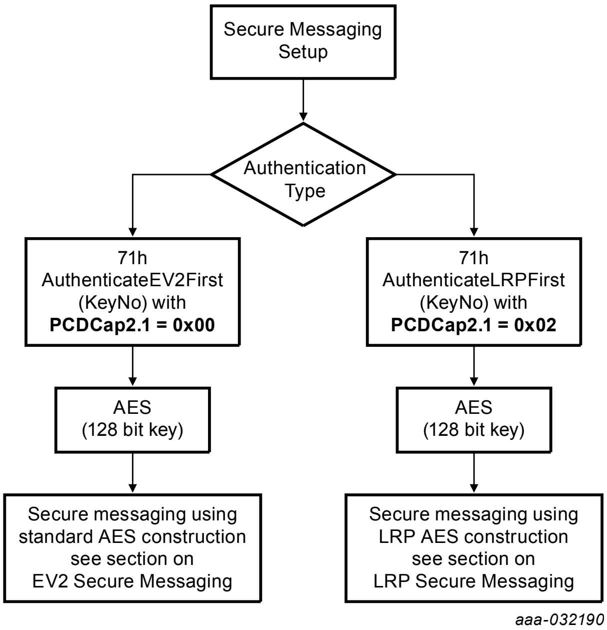


Figure 6. MIFARE DESFire Light secure messaging setup

A First Authentication can be executed at any time whether the PICC is in authenticated or not authenticated state.

The Non-First Authentication can only be executed while the card is in authenticated state after a successful First or Non-First Authentication.

Correct application of First Authentication and Non-First Authentication allows to cryptographically bind all messages within a transaction together by the transaction identifier established in a First Authentication, see Section 9.1.1, and a command counter, see Section 9.1.2, even if multiple authentications are required.

The following table specifies when to authenticate using First Authentication and when to use Non-First Authentication.

Table 17. When to use which authentication command

Purpose	First Authentication	Non-First Authentication
First authentication (i.e. when not in any authenticated state) with any key different from OriginalityKey	Allowed	Not Allowed
Subsequent authentication (i.e. when in any authenticated state) with any key different from OriginalityKey	Allowed, recommended not to use.	Allowed, recommended to use.
Any LRP authentication with OriginalityKey	Allowed	Allowed

The AuthenticateEV2First initiates a standard AES authentication and secure messaging, compatible with MIFARE DESFire EV2, see Section 9.1. The other variant AuthenticateLRPFirst initiates an AES authentication and secure messaging based on the Leakage Resilient Primitive (LRP), see Section 9.2.

The negotiation between those two variants is done using the capabilities of the First Authentication and the return message of the first part, where a PCD can distinguish between standard AES authentication and LRP authentication based on the message length.

On First Authentication, the PCD can choose between AES and LRP secure messaging by setting bit 1 of PCDCap2.1 in the issued command. The PD is configured for either AES or LRP secure messaging by the setting of bit 1 from PDCap2.1. This setting is defined with SetConfiguration, see Section 11.5.1.

If the PCD chooses for AES secure messaging, it sends PCDCap2.1 equaling 00h (or no PCDCap2 at all). A MF2DL(H)x0 will accept the authentication if its PDCap2.1 bit 1 is not set, i.e. the MF2DL(H)x0 is configured for AES secure messaging. The command is interpreted as AuthenticateEV2First, see Section 9.1.5 for detailed specification. If PDCap2.1 bit 1 is set, i.e. the MF2DL(H)x0 is configured for LRP secure messaging, the authentication request is rejected.

If the PCD chooses for LRP secure messaging, it sends PCDCap2.1 equaling 02h. MIFARE DESFire Light will accept the authentication if its PDCap2.1 bit 1 is set, i.e. the MF2DL(H)x0 is configured for LRP secure messaging. The command is interpreted as AuthenticateLRPFirst and replied with 18 bytes, i.e. ADDITIONAL_FRAME, followed by an additional AuthMode indicating LRP secure messaging, and 16 bytes of data, see Section 11.4.3 for detailed specification. If PDCap2.1 bit 1 is not set, i.e. the MF2DL(H)x0 is configured for AES secure messaging, the authentication request is also accepted, but responded with 17 bytes, i.e. the AuthenticateEV2First response composed of ADDITIONAL_FRAME, followed by 16 bytes of data, allowing the PCD to fall back to standard AES authentication as well.

With Non-First Authentication, the PCD cannot choose between standard AES and LRP. If authenticated using AES mode, AuthenticateEV2NonFirst will be applied, see Section 9.1.6. If authenticated with LRP mode, AuthenticateLRPNonFirst will be applied, see Section 11.4.4. If not authenticated at all, e.g. if targeting one of the originality keys, only AuthenticateLRPNonFirst is supported.

Below table provides possible negotiation outcomes on FirstAuthentication.

Table 18. Secure messaging mode negotiation

PCD		PD				
Mode	**PCDCap2.1**	**PDCap2. (Mode)**	**RC**	**resp PDCap2.**	**resp PCDCap**	**comment**
Requesting EV2 Secure Messaging	00h	00h (AES)	ADDITIONAL_ FRAME: 17-byte response without AuthMode	00h	00h	Matching, AES SM accepted and selected
		02h (LRP)	PERMISSION_ DENIED	N/A	N/A	No match, AES SM rejected
Requesting LRP Secure Messaging	02h	00h (AES)	ADDITIONAL_ FRAME	00h	02h	No match, PD replies with AES response, allowing a PCD to fall back.
		02h (LRP)	ADDITIONAL_ FRAME: 18-byte response with AuthMode set to 01h	02h	02h	Matching, LRP SM accepted and selected

9.1 AES Secure Messaging

The AES Secure Messaging is managed by AuthenticateEV2First and AuthenticateEV2NonFirst.

Note that AuthenticateEV2First and AuthenticateEV2NonFirst can also be used to start LRP Secure Messaging, as defined in Section 9.2. This is done with the PCDCap2 sent in First Authentication and the return code, see Section 9 and Section 9.2.5 for details.

9.1.1 Transaction Identifier

In order to avoid interleaving of transactions from multiple PCDs toward one PICC, the Transaction Identifier (TI) is included in each MAC that is calculated over commands or responses. The TI is generated by the PICC and communicated to the PCD with a successful execution of an AuthenticateEV2First command, see Section 11.4.1. The size is 4 bytes and these 4 bytes can hold any value. The TI is treated as a byte array, so there is no notion of MSB and LSB.

9.1.2 Command Counter

A command counter is included in the MAC calculation for commands and responses in order to prevent e.g. replay attacks. It is also used to construct the Initialization Vector (IV) for encryption and decryption.

Each command, besides few exceptions, see below, is counted by the command counter CmdCtr which is a 16-bit unsigned integer. Both sides count commands, so the actual value of the CmdCtr is never transmitted. The CmdCtr is reset to 0000h at PCD and PICC after a successful AuthenticateEV2First authentication and it is maintained as long as the PICC remains authenticated. In cryptographic calculations, the CmdCtr is represented LSB first. Subsequent authentications using AuthenticateEV2NonFirst do not affect the CmdCtr. Subsequent authentications using the AuthenticateEV2First will reset the CmdCtr to 0000h. The CmdCtr is increased between the command and response, for all communication modes.

When a MAC on a command is calculated at PCD side that includes the CmdCtr, it uses the current CmdCtr. The CmdCtr is afterwards incremented by 1. At PICC side, a MAC appended at received commands is checked using the current value of CmdCtr. If the MAC matches, CmdCtr is incremented by 1 after successful reception of the command, and before sending a response.

For CommMode.Full, the same holds for both the MAC and encryption IV calculation, i.e. the non-increased value is used for the command calculations while the increased value is used for the response calculations.

If the CmdCtr holds the value FFFFh and a command maintaining the active authentication arrives at the PICC, this leads to an error response and the command is handled like the MAC was wrong.

Command chaining, see Section 8.5, does not affect the counter. The chained command is considered as a single command, just as for the other aspects of secure messaging, and thus the related counter is increased only once.

9.1.3 MAC Calculation

MACs are calculated using the underlying block cipher according to the CMAC standard described in [6]. Padding is applied according to the standard.

The MAC used in MF2DL(H)x0 is truncated by using only the 8 even-numbered bytes out of the 16 bytes output as described [6] when represented in most-to-least-significant order.

Initialization Vector for MACing

The initialization vector used for the CMAC computation is the zero byte IV as prescribed [6].

9.1.4 Encryption

Encryption and decryption are calculated using AES-128 according to the CBC mode of NIST SP800-38A [5].

Padding is applied according to Padding Method 2 of ISO/IEC 9797-1 [7], i.e. by adding always 80h followed, if required, by zero bytes until a string with a length of a multiple of 16 byte is obtained. Note that if the plain data is a multiple of 16 bytes already, an additional padding block is added. The only exception is during the authentication itself (AuthenticateEV2First and AuthenticateEV2NonFirst), where no padding is applied at all.

The notation *E(key, message)* is used to denote the encryption and *D(key, message)* for decryption.

Initialization Vector for Encryption

When encryption is applied to the data sent between the PCD and the PICC, the Initialization Vector (IV) is constructed by encrypting with SesAuthENCKey according to the ECB mode of NIST SP800-38A [5] the concatenation of:

- a 2-byte label, distinguishing the purpose of the IV: A55Ah for commands and 5AA5h for responses
- Transaction Identifier TI
- Command Counter CmdCtr (LSB first)
- Padding of zeros acc. to NIST SP800-38B [6]

This results in the following IVs:

IV for CmdData = *E(SesAuthENCKey; A5h* || *5Ah* || TI || CmdCtr || *0000000000000000h)*

IV for RespData = *E(SesAuthENCKey;5Ah* || *A5h* || TI || CmdCtr || *0000000000000000h)*

When an encryption or decryption is calculated, the CmdCtr to be used in the IV are the current values. Note that this means that if CmdCtr = *n* before the reception of a command, after the validation of the command CmdCtr = *n + 1* and that value will be used in the IV for the encryption of the response.

For the encryption during authentication (both AuthenticateEV2First and AuthenticateEV2NonFirst), the IV will be 128 bits of 0.

9.1.5 AuthenticateEV2First Command

This section defines the Authentication, which is mandatory to be used first in a transaction when using Secure Messaging, see Table 17. In this procedure both, the PICC as well as the PCD show in an encrypted way that they possess the same secret, i.e. the same key. This authentication is supported with AES keys.

The authentication consists of two parts: AuthenticateEV2First - Part1 and Section 9.1.6 - Part2. Detailed command definition can be found in Section 11.4.1. The protocol cannot be interrupted by other commands. On any command different from AuthenticateEV2First - Part2 received after the successful execution of the first part, the PICC aborts the ongoing authentication.

During this authentication phase, the PICC accepts messages from the PCD that are longer than the lengths derived from this specification as long as LenCap is correct. This feature is to support the upgradability to following generations of MF2DL(H)x0. The PCD rejects answers from the PICC when they don't have the proper length. Note that if present, PCDcap2:1:Bit1 must not be set, otherwise LRP authentication is targeted, see Section 9.2.5.

Upon reception of AuthenticateEV2First, the PICC validates the targeted key. If the key does not exist, AuthenticateEV2First is rejected.

The PICC generates a random 16-byte challenge *RndB* and send this encrypted to the PCD, according to Section 9.1.4. Additionally, the PICC resets CmdCtr to zero and generate a random Transaction Identifier (TI).

Upon reception of the AuthenticateEV2First response from the PICC, the PCD also generates a random 16-byte challenge *RndA*. The PCD encrypts, on his turn, the concatenation of *RndA* with *RndB'*, which is the received challenge after decryption and rotating it left by one byte. Within AuthenticateEV2First - Part2, this is sent to the PICC. Upon reception of AuthenticateEV2First - Part2, the PICC decrypts the second message and validates the received *RndB'*. If not as expected, the command is rejected. Else it generates *RndA'* by rotating left the received *RndA* by one byte. This is returned together with the generated TI. Also, the PICC sends 12 bytes of capabilities to the PCD: 6 bytes of PICC capabilities PDcap2 and 6 bytes of PCD capabilities PCDcap2 that were received on the command (sent back for verification).

The use of those capabilities, and the negotiation process is described in Section 9. Note that part of PDCap will be configurable with SetConfiguration. PCDcap2 is used to refer both to the value sent from the PCD to the PICC and to the value used in the encrypted response message from the PICC to the PCD where in this case the PCDcap2 is the adjusted version of the originally sent PCDcap2: i.e. truncated or padded with zero bytes to a length of 6 bytes if needed.

On successful execution of the authentication protocol, the session keys SesAuthMACKey and SesAuthENCKey are generated according to Section 9.1.7. The PICC is in EV2 authenticated state and the Secure Messaging is activated. On any failure during the protocol or if one of the OriginalityKey were targeted, the PICC ends up in not authenticated state.

If there is a mismatch between the capabilities expected by the PCD and the capabilities presented by the PICC to the PCD (both the PDcap2 and the echoed/adjusted PCDcap2), it is the responsibility of the PCD to take the proper actions based on the application the PCD is running. This decision is outside the scope of this specification.

9.1.6 AuthenticateEV2NonFirst Command

This section defines the Non-First Authentication, which is recommended to be used if Secure Messaging is already active, see Table 17. In this procedure both, the PICC as well as the PCD show in an encrypted way that they possess the same secret, i.e. the same key. This authentication is supported with AES keys.

The authentication consists of two parts: AuthenticateEV2NonFirst - Part1 and AuthenticateEV2NonFirst - Part2. Detailed command definition can be found in Section 11.4.2. This command is rejected if there is no active authentication, except if the targeted key is the OriginalityKey. For the rest, the behavior is exactly the same as for AuthenticateEV2First, except for the following differences:

- No *PCDcap2* and *PDcap2* are exchanged and validated.
- Transaction Identifier TI is not reset and not exchanged.
- Command Counter CmdCtr is not reset.

After successful authentication, the PICC remains in EV2 authenticated state. On any failure during the protocol, the PICC ends up in not authenticated state.

9.1.7 Session Key Generation

At the end of a valid authentication with AuthenticateEV2First or AuthenticateEV2NonFirst, both the PICC and the PCD generate two session keys for secure messaging, as shown in Figure 7:

- SesAuthMACKey for MACing of messages
- SesAuthENCKey for encryption and decryption of messages

Note that these identifiers are also used in context of the LRP protocol, though the actual calculation of the session keys is different, see Section 9.2.7.

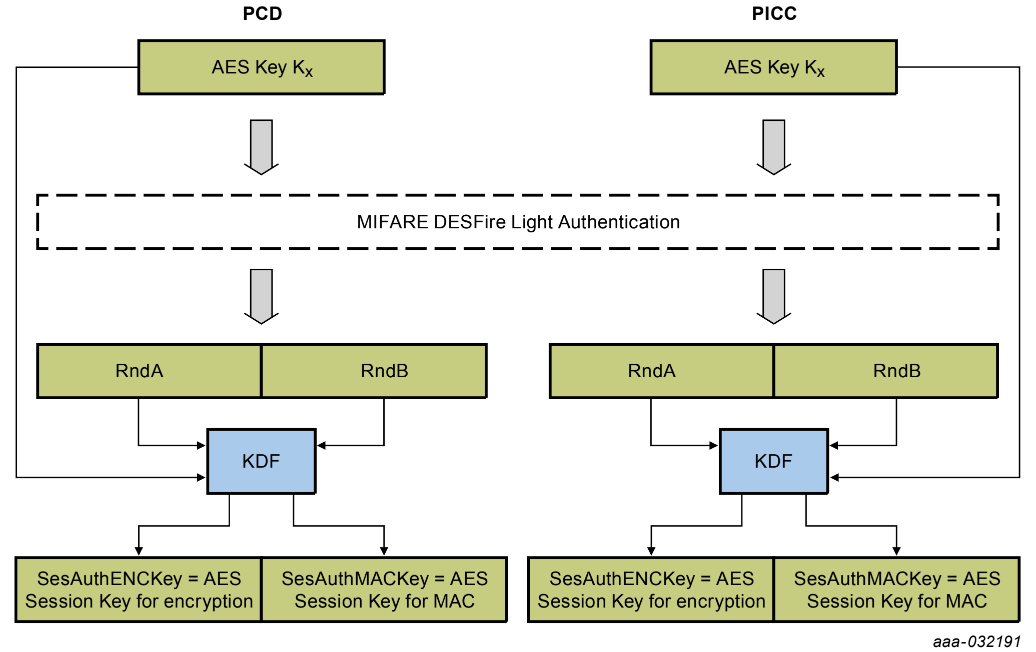


Figure 7. Session key generation for Secure Messaging

The session key generation is according to NIST SP 800-108 [8] in counter mode.

The Pseudo Random Function PRF(key; message) applied during the key generation is the CMAC algorithm described in NIST Special Publication 800-38B [6]. The key derivation key is the key Kx that was applied during authentication. As the authentications are restricted to target AES keys, the generated session keys are also of AES.

The input data is constructed using the following fields as defined by [8]. Note that NIST SP 800-108 allows defining a different order than proposed by the standard as long as it is unambiguously defined.

- a 2-byte label, distinguishing the purpose of the key: 5AA5h for MACing and A55Ah for encryption
- a 2-byte counter, fixed to 0001h as only 128-bit keys are generated.
- a 2-byte length, fixed to 0080h as only 128-bit keys are generated.
- a 26-byte context, constructed using the two random numbers exchanged, RndA and RndB

First, the 32-byte input session vectors SV_x are derived as follows:

$$SV1 = A5h||5Ah||00h||01h||00h||80h||RndA[15..14]|| \\ (RndA[13..8] \oplus RndB[15..10])||RndB[9..0]||RndA[7..0]$$

$$SV2 = 5Ah||A5h||00h||01h||00h||80h||RndA[15..14]|| \\ (RndA[13..8] \oplus RndB[15..10])||RndB[9..0]||RndA[7..0]$$

with $\oplus$ being the XOR-operator.

Then, the 16-byte session keys are constructed as follows:

$$SesAuthENCKey = PRF(K_x, SV1)$$
$$SesAuthMACKey = PRF(K_x, SV2)$$

9.1.8 Plain Communication Mode

The command and response data is not secured. The data is sent in plain, see Figure 8, i.e. as defined in the command specification tables, see Section 11.

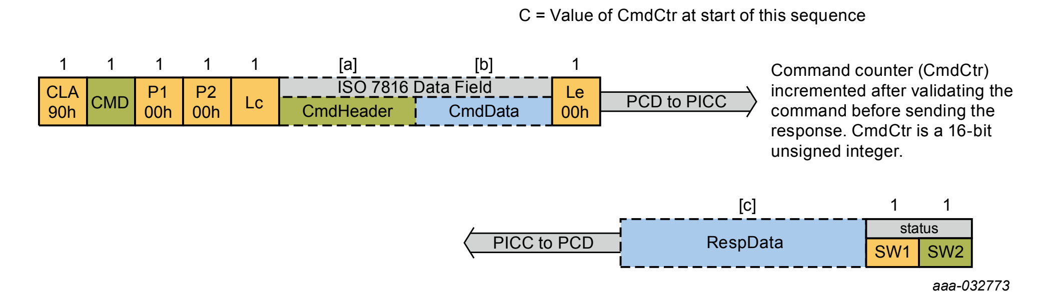


Figure 8. Plain Communication Mode

However, note that, as the PICC is in authenticated state (EV2 authenticated state or LRP authenticated state), the command counter CmdCtr is still increased as defined in Section 9.1.2.

This allows the PCD and PICC to detect any insertion and/or deletion of commands sent in CommMode.Plain on any subsequent command that is sent in CommMode.MAC (e.g. CommitTransaction) or CommMode.Full.

9.1.9 MAC Communication Mode

The Secure Messaging applies MAC to all commands listed as such in Section 11.2.

In case MAC is to be applied, the following holds. The MAC is calculated using the current session key SesAuthMACKey. MAC calculation is done as defined in Section 9.1.3.

For commands, the MAC is calculated over the following data (according to the definitions from Section 8.3) in this order:

- Command, Cmd

- Command Counter CmdCtr
- Transaction Identifier TI
- Command header - CmdHeader (if present)
- Command data - CmdData (if present)

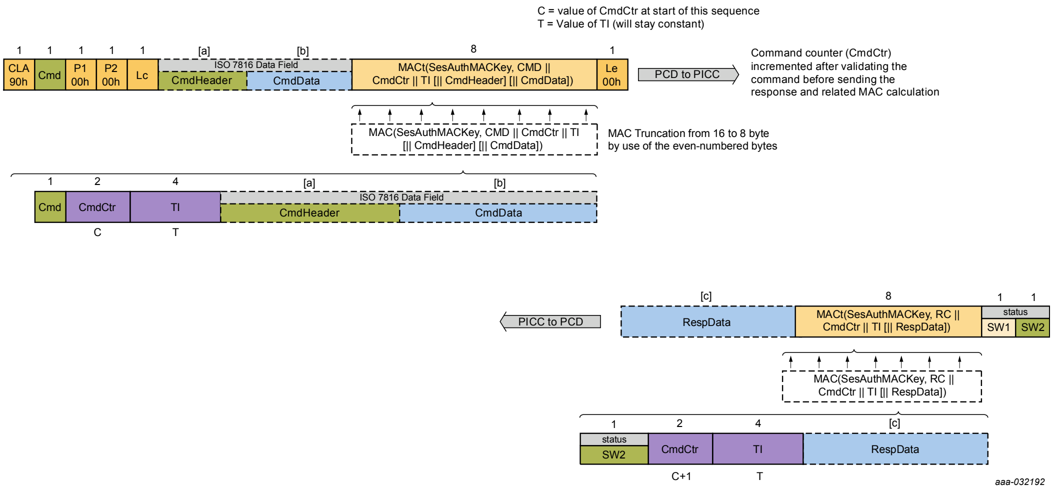
For responses, the MAC is calculated over the following data in this order:

- Return code - RC
- Command Counter - CmdCtr (The already increased value)
- Transaction Identifier - TI
- Response data - RespData (if present)

CmdCtr is the Command Counter as defined in Section 9.1.2. Note that the CmdCtr is increased between the computation of the MAC on the command and the MAC on the response. TI is the Transaction Identifier, as defined in Section 9.1.1. The other input parameters are as defined in Section 8.3. The calculation is illustrated in Figure 9.

In case of command chaining, the MAC calculation is not interrupted. The MAC is calculated over the data including the complete data field (i.e. either CmdData or RespData of all frames) at once. The MAC is always transmitted by appending to the unpadded plain command. If necessary, an additional frame is sent. If a MAC over the command is received, the PICC verifies the MAC and rejects commands that do not contain a valid MAC by returning INTEGRITY_ERROR.

In this case, the ongoing command and transaction are aborted (see also Section 11). The authentication state is immediately lost and the error return code is sent without a MAC appended. Note that any other error during the command execution has the same consequences.



CLA, P1, P2, LC, Le and SW1 not included in secure messaging calculation. SW2 is the return code (RC) and appended in the beginning for secure messaging calculation

Figure 9. Secure Messaging: MAC Communication mode

9.1.10 Full Communication Mode

The Secure Messaging applies encryption (CommMode.Full) to all commands listed as such in Section 11.2. In case CommMode.Full is to be applied, the following holds. The encryption/decryption is calculated using the current session key SesAuthENCKey. Calculation is done as defined in Section 9.1.4 over either the command or the response data field (i.e. CmdData or RespData). Note that none of the commands have a data field in both the command and the response frame.

After the encryption, the command and response frames are handled as with MAC. This means that additionally a MAC is calculated and appended for transmission using the current session key SesAuthMACKey. This is exactly done as specified for MAC in Section 9.1.9, replacing the plain CmdData or RespData by the encrypted field: *E(SesAuthENCKey; CmdData)* or *E(SesAuthENCKey; RespData)*. The complete calculation is illustrated in Figure 10. In case of command chaining, the encryption/ decryption is applied over the complete data field (i.e. of all frames). If necessary, due to the padding or the MAC added, an additional frame is sent. If encryption of the command is required, after the MAC verification as described for MAC, the PICC verifies and removes the padding bytes. Commands without a valid padding are also rejected by returning INTEGRITY_ERROR.

In this case, the ongoing command and transaction are aborted (see also Section 11). The authentication state is immediately lost and the error return code is sent without a MAC appended. Note that any other error during the command execution has the same consequences.

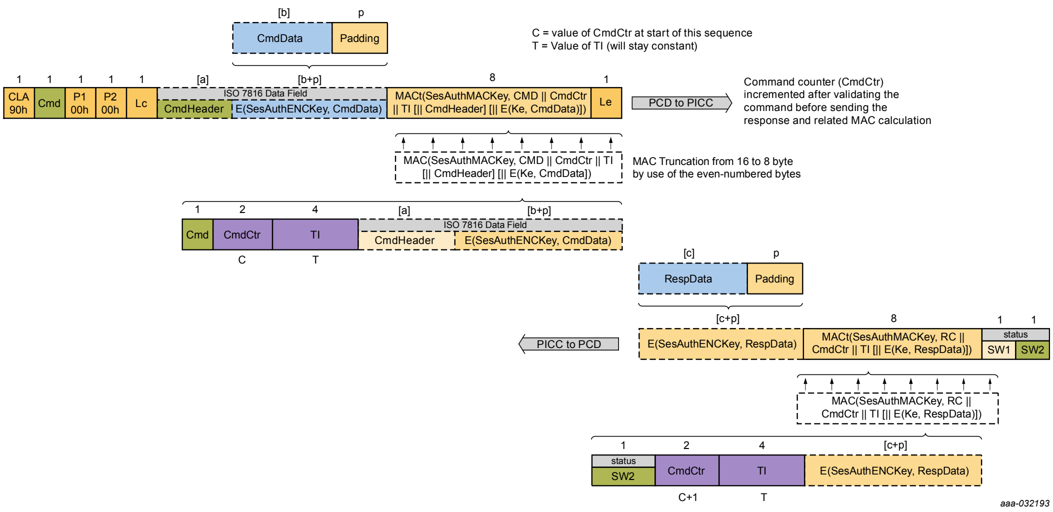


Figure 10. Secure Messaging: CommMode.Full

9.2 LRP Secure Messaging

The LRP Secure Messaging is using AES-128 to construct a Leakage Resilient Primitive. This way, it allows side-channel resistant implementation.

Like the AES secure messaging, this secure messaging mode is managed by commands with the same command code as AuthenticateEV2First and AuthenticateEV2NonFirst. To distinguish and ease the descriptions, they are renamed for the LRP case into AuthenticateLRPFirst and AuthenticateLRPNonFirst. The recommendations of Section 9 on when to use one or the other command also apply for LRP secure messaging.

9.2.1 Transaction identifier

The Transaction Identifier (TI) is treated exactly in the same way by LRP secure messaging as defined for AES secure messaging, see Section 9.1.1.

9.2.2 Command counter

The Command counter (CmdCtr) is treated exactly in the same way by LRP secure messaging as defined for AES secure messaging, see Section 9.1.2.

9.2.3 MAC calculation

MACs are computed by using a CMAC construction on top of the LRP primitive. This is specified in [10]. This document uses the following notation where the right hand refers to the notation of [10].

$MAC_{LRP}(key,message) = CMAC\_LRP(4,key, Len(message),message)$

Note that in the LRP context a key is not purely a single value, but rather consists of the associated set of plain texts, an updated key and in context of CMAC also the subkeys K1 and K2. Therefore K1 and K2 are not shown (contrary to [10]) as they can be calculated inside.

$MACt_{LRP}(key, message)$ denotes the CMAC after truncation to 8 bytes which is identical to the truncation of the AES secure messaging i.e. the even-numbered bytes are retained in most-to-least-significant order, see Section 9.1.3.

The initialization vector used for the CMAC computation is the zero byte IV as prescribed [10].

9.2.4 Encryption

Encryption and decryption are calculated using a Leakage Resilient Indexed CodeBook (LRICB) construction on top of the LRP primitive: *LRICB*of [10].

For this purpose an Encryption Counter is maintained: EncCtr is a 32-bit unsigned integer as Input Vector (IV) for encryption/decryption. The EncCtr is reset to 000000000h at PCD and PICC when starting an authentication with AuthenticateLRPFirst or AuthenticateLRPNonFirst targeting LRP. The counter is incremented during each encryption/decryption of each 16-byte block. i.e. for 64-byte encryption/decryption the EncCtr is increased by 5 due to 4 blocks of 16-byte of data plus one block of padding. Note that for AuthenticateLRPFirst the value 00000000h is already used for the response of part 2, so the actual secure messaging starts from 00000001h. For AuthenticateLRPNonFirst, secure messaging starts from 00000000h as the counter is not used during the authentication. EncCtr is further maintained as long as the PICC remains in LRP authenticated state. Note that for the key stream calculation [10], the counter is represented MSB first.

Padding is applied according to Padding Method 2 of ISO/IEC 9797-1 [7], i.e. by adding always 80h followed, if required, by zero bytes until a string with a length of a multiple of 16 bytes is obtained. Note that if the plain data is a multiple of 16 bytes already, an additional padding block is added. The only exception is during the authentication itself (AuthenticateLRPFirst and AuthenticateLRPNonFirst), where no padding is applied at all.

The notation $E_{LRP}(key, plaintext)$ is used to denote the encryption, i.e. *LRICBEnc* of [10] and $D_{LRP}(key, ciphertext)$ for the complementary decryption operation. Note that in the LRP context a key is not purely a single value, but rather consists of the associated set of plain texts and updated key. Also, as specified in [10], the EncCtr is updated as part of the operation.

Note that the EncCtr cannot overflow. Due to the supported file sizes, the CmdCtr will always expire before.

Note that the MSB representation of EncCtr is different from other counter representations in this specification, but allows saving some AES calculations in the key stream generation.

9.2.5 AuthenticateLRPFirst command

The AuthenticateLRPFirst command reuses the same command code as AuthenticateEV2First. The distinction is made via the PCDCap2.1 parameter, as explained in Section 9.

The AuthenticateLRPFirst command is fully compliant with the mutual three-pass authentication of ISO/IEC 9798-4 [7].

The authentication consists of two parts: AuthenticateLRPFirst - Part1 and AuthenticateLRPFirst Part2. Detailed command definition can be found in Section 11.4.3.

The protocol cannot be interrupted by other commands. On any command different from AuthenticateLRPFirst - Part2 received after the successful execution of the first part, the PICC aborts the ongoing authentication.

During this authentication phase, the PICC accepts messages from the PCD that are longer than the lengths derived from this specification as long as *LenCap* is correct. This feature is to support the upgradability to following generations of MIFARE DESFire Light.

Apart from bit 1 of PCDCap2.1, which need to be set to 1 for AuthenticateLRPFirst resulting into 020000000000h, the content of PCDCap2 is not interpreted by the PICC. The PCD rejects answers from the PICC when they don’t have the proper length.

Upon reception of AuthenticateLRPFirst, the PICC validates the targeted key. If the key does not exist, AuthenticateLRPFirst is rejected. At PICC level, the only available key is the OriginalityKey.

The PICC generates a random 16-byte challenge *RndB* and send this in plain to the PCD. Additionally, the PICC and PCD reset both CmdCtr and EncCtr to zero and generate a random TI.

Upon reception of the AuthenticateLRPFirst response from the PICC, the PCD also generates a random 16-byte challenge *RndA*. Now the PCD calculates the session keys SesAuthMACKey and SesAuthENCKey, as specified in Section 9.2.7. As explained there for LRP, a session key consists of a set of plain texts and an updated key.

Then the PCDResponse computes a MAC over the concatenation of *RndA* with *RndB*, applying the SesAuthMACKey with the algorithm defined in Section 9.2.3. Note that MACs are not truncated during the authentication. Within AuthenticateLRPFirst - Part2, the concatenation of *RndA* and this MAC is sent to the PICC.

Upon reception of AuthenticateLRPFirst - Part2, the PICC validates the received MAC. If not as expected, the command is rejected. Else it encrypts the generated TI concatenated with 12 bytes of capabilities to the PCD: 6 bytes of PICC capabilities *PDCap2* and 6 bytes of PCD capabilities *PCDCap2* that were received on the command (sent back for verification). Encryption is done according to Section 9.2.4, applying SesAuthENCKey.

Note that part of PDCap is configurable with SetConfiguration. *PCDCap2* is used to refer both to the value sent from the PCD to the PICC and to the value used in the encrypted response message from the PICC to the PCD where in this case the *PCDCap2* is the adjusted version of the originally sent *PCDCap2*: i.e. truncated or padded with zero bytes to a length of 6 bytes if needed. After that encryption, the PICCResponse will also compute a MAC over the concatenation of *RndB*, *RndA* and the encrypted data.

9.2.6 AuthenticateLRPNonFirst command

This section defines the LRP Non-First Authentication, which is recommended to be used if LRP Secure Messaging is already active, see Table 17.

The authentication consists of two parts: AuthenticateLRPNonFirst - Part1 and AuthenticateLRPNonFirst Part2. Detailed command definition can be found in Section 11.4.4.

This command is rejected if there is no active LRP authentication, except if the targeted key is the OriginalityKey.

For the rest, the behavior is exactly the same as for AuthenticateLRPFirst, except for the following differences:

- PCDCap2 and PDCap2 are not exchanged and validated
- TI is not reset and not exchanged
- CmdCtr is not reset

Note that EncCtr is reset to zero also on AuthenticateLRPNonFirst.

After successful authentication, the PICC remains in LRP authenticated state, except if the OriginalityKey was targeted. In that case, the PICC is in not authenticated state. On any failure during the protocol, the PICC ends up in not authenticated state.

9.2.7 Session key generation

Next to the algorithms for MAC calculation and encryption, one of the major differences between the LRP secure messaging and the AES secure messaging is that the session keys are generated and already applied during the authentication with AuthenticateLRPFirst or AuthenticateLRPNonFirst.

Also for the LRP protocol, two keys are generated:

- SesAuthMACKey for MACing of messages
- SesAuthENCKey for encryption and decryption of messages

During the authentication, the SesAuthMACKey is used for both AuthenticateLRPFirst and AuthenticateLRPNonFirst. SesAuthENCKey is only used for AuthenticateLRPFirst.

Being LRP keys, this section shows how both the plain texts and the updated key [10] related to these session keys are computed. In the remainder of the document, when the session key is applied in the LRP context the combination of those plain texts and updated key is meant.

The session key generation is according to NIST SP 800-108 [8] in counter mode.

The Pseudo Random Function *PRF(key; message)* applied during the key generation is the CMAC algorithm on top of the LRP primitive. This is specified in [10], see also Section 9.2.3. The key derivation key is the key *Kx* that was applied during authentication. Note that from this key a set of plaintexts and updated key is computed, so the static key is only used in this derivation. The generated session keys are AES keys. The input data is constructed using the following fields as defined by [8]. Note that NIST SP 800-108 allows defining a different order than proposed by the standard as long as it is unambiguously defined.

- a 2-byte counter, fixed to 0001h as only 128-bit keys are generated

- a 2-byte length, fixed to 0080h as only 128-bit keys are generated
- a 26-byte context, constructed using the two random numbers exchanged, *RndA* and *RndB*
- a 2-byte label: 9669h

Firstly, the 32-byte input session vector *SV* is derived as follows:

$$SV = 00h \,||\, 01h \,||\, 00h \,||\, 80h \,||\, RndA[15::14] \,||$$
$$(RndA[13::8] \oplus RndB[15::10]) \,||\, RndB[9::0] \,||\, RndA[7::0] \,||\, 96h \,||\, 69h$$

with ⊕ being the XOR-operator.

Then, the session key material is constructed as follows:

$AuthSPT = generatePlaintexts(4; K_x)$
$\{AuthUpdateKey\} = generateUpdatedKeys(1; K_x)$
$SesAuthMasterKey = MACLRP(K_x; SV)$
$SesAuthSPT = generatePlaintexts(4; SesAuthMasterKey)$
$\{SesAuthMACUpdateKey; SesAuthENCUpdateKey\} = generateUpdatedKeys(2; SesAuthMasterKey)$

with *generatePlaintexts* and *generateUpdatedKeys* the functions from [10]. Note that the output of *generateUpdatedKeys* is shown in the order that the keys are generated. The actual SesAuthMACKey then consists for LRP of the set of plaintexts SesAuthSPT (consisting of 16 16-byte values) and SesAuthMACUpdateKey. The SesAuthENCKey consists of the same set of plaintexts SesAuthSPT and SesAuthENCUpdateKey.

9.2.8 Plain communication mode

For CommMode.Plain, command processing in LRP authenticated state is identical to AES secure messaging in EV2 authenticated state, see Section 9.1.8.

9.2.9 MAC communication mode

For MAC, apart from using the LRP MAC algorithm, as specified in Section 9.2.3, the command processing in LRP authenticated state is identical to AES secure messaging in EV2 authenticated state, see Section 9.1.9. The calculation is illustrated in Figure 11.

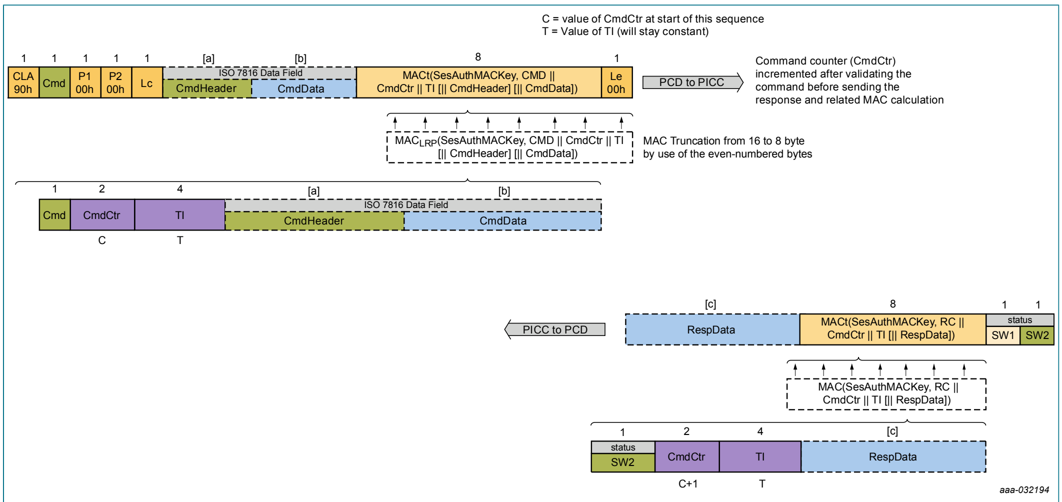


Figure 11. LRP Secure Messaging: MAC Protection Mode

9.2.10 Full communication mode

For CommMode.Full, apart from using the LRP encryption and MAC algorithm, as specified in Section 9.2.4, the command processing in LRP authenticated state is identical to AES secure messaging in EV2 authenticated state, see Section 9.1.10. This is as well illustrated in Figure 12.

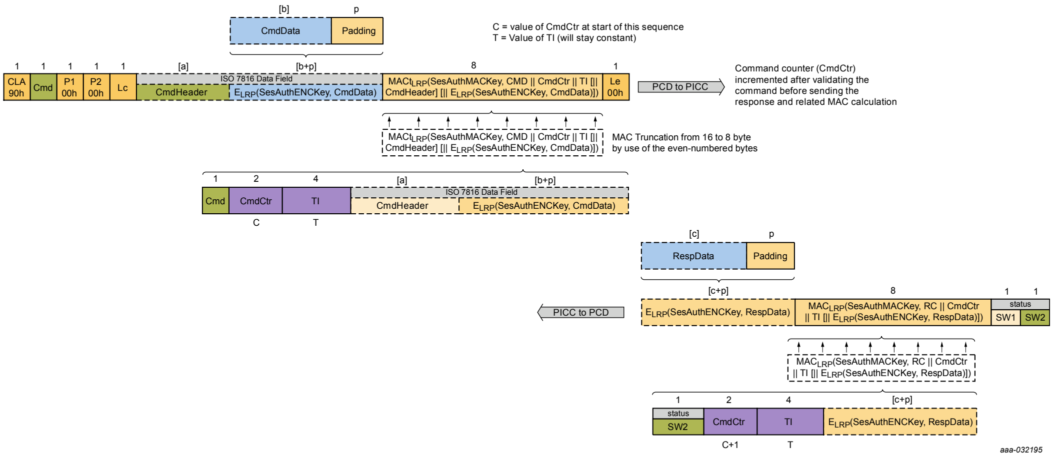


Figure 12. LRP Secure Messaging: CommMode.Full

All information provided in this document is subject to legal disclaimers.

10 Transaction Management

MIFARE DESFire Light supports the grouping of data management operations into transactions. This allows for committing or aborting all previous write access to files with integrated backup mechanisms within the selected application via a single atomic operation.

How a transaction can be initiated and concluded, is detailed in Section 10.1 and Section 10.2. On top of this, MF2DL(H)x0 supports a Transaction MAC feature which allows proofing transaction execution toward a third party. This feature can be used if, for example, a merchant needs to be reimbursed by a backend for the transaction he executed. This way the backend is protected against the fraudulent merchant scenario. The Transaction MAC feature is specified in Section 10.3.

10.1 Transaction Initiation

MF2DL(H)x0 does not provide a separate command for initiating a transaction. Instead, a transaction is started by selecting the application with a ISOSelectFile command.

Next to this, a new transaction is also automatically initiated after conclusion of a previous transaction, see Section 10.2.

Note that a transaction is not affected by the execution of an authentication. One can have several authentications during the course of a single transaction.

10.2 Transaction Conclusion

A transaction can be concluded by successfully committing all write operations on backup files, as detailed in Section 10.2.1, or by abortion which will cancel all changes, as detailed in Section 10.2.2. For both operations, MF2DL(H)x0 supports a specific command, but transaction abortion can also be triggered by other commands.

Note that some of the parameters and actions triggered by these commands are related with the Transaction MAC feature, explained in Section 10.3. For a full understanding, the description in that section should be referred to.

10.2.1 CommitTransaction

Validating all previous write access to back up files within the currently selected application is possible with the command CommitTransaction as defined in Section 11.9.1.

The CommitTransaction takes an optional parameter option, which indicates if the Transaction MAC Counter (TMC) and Transaction MAC Value (TMV) of the calculated Transaction MAC, see Section 10.3, are to be sent with the response. If this is requested while the selected application does not support Transaction MAC calculation, i.e. no TransactionMAC file is present, the command is rejected.

The CommitTransaction validates all preceding write access of the ongoing transaction to files with integrated backup mechanisms:

- Value file
- CyclicRecord file

If the Transaction MAC feature is enabled, the TransactionMAC file is updated if the Transaction MAC Input (TMI) is different from the empty string. The updated file holds the calculated TMV, as defined in Section 10.3.2.4 and the increased TMC which was used for the calculation of SesTMACKey.

In case the application is configured for LRP, this means the used actTMC and sesTMC, see Section 10.3.2.1, are written into the file. On successful CommitTransaction, the sesTMC is also reset to zero and written to the non-user-accessible NVM for future session key generation. If TMC reached the TMCLimit, see Section 10.3.2.2, the application remains selected, but with limited functionality: no data management commands, see Section 11.8 can be executed anymore, except ReadData targeting the TransactionMAC file. Also CommitReaderID will be rejected. All these file updates are processed as a single atomic operation. In case of tearing, either all or none of the changes are applied to the memory.

After execution of this command a new transaction is started.

The command is rejected if no application has been selected.

An error code is also returned if no write access has been done to any of the backup file types and, in case the application is protected by the Transaction MAC feature (see Section 10.3), no Transaction MAC calculation update has been done. This means no command that is included in the Transaction MAC calculation has been executed, i.e. the TMI is still the empty string. So note that if Transaction MAC is enabled, successful execution of CommitTransaction updating the TransactionMAC is possible without any write operation, as read operations are also included in the Transaction MAC calculation.

If CommitReaderID is required according to access right configuration of the TransactionMAC, this command will be rejected if no ReaderID has been committed during the ongoing transaction, i.e. if TMRICur is still the empty string. If a ReaderID was committed in an authenticated state (EV2 authenticated state or LRP authenticated state), this value TMRICur is stored in non-volatile memory as TMRIPrev as part of the atomic operation. If the ReaderID was committed when not authenticated, TMRIPrev is not updated.

The CommitTransaction requires MAC if there is an active authentication. Information on authentication and secure messaging-dependent structure of the command can be found in Section 9.

The CommitTransaction is typically the last command of a transaction before deselecting the card with the ISO/IEC 14443-4 deselect command [3].

Note that TMC and TMV can also be read out encrypted, if preferred. In this case, the TransactionMAC file should be configured for CommMode.Full. One can then use ReadData to retrieve this information, instead of requesting it within the response of CommitTransaction.

10.2.2 AbortTransaction

A transaction can be aborted by explicitly issuing a AbortTransaction command.

Next to this any ongoing transaction is also automatically aborted after:

- unsuccessful execution of a command
- selecting an application with the ISOSelectFile command, even if the same application is reselected.
- creating/deleting the TransactionMAC file with CreateTransactionMACFile or DeleteTransactionMACFile
- enabling LRP with SetConfiguration
- renaming files with SetConfiguration

- configuring Value file with SetConfiguration
- enabling TMCLimit with ChangeFileSettings
- enabling exclusion of unauthenticated operations from Transaction MAC calculation with ChangeFileSettings

This has the same effects as successful execution of AbortTransaction detailed below. A new transaction is started.

Invalidating all previous write access to back up files within the currently selected application is possible with the AbortTransaction command as defined in Section 11.9.2. This command takes no parameter. The AbortTransaction invalidates all preceding write access of the ongoing transaction to files with integrated backup mechanisms:

- Value file
- CyclicRecord file

If a Transaction MAC is calculated, see Section 10.3, the ongoing calculation is also aborted and a new calculation is initiated.

The AbortTransaction does not affect the authentication status. After execution of this command a new transaction is started.

The command is rejected if no application has been selected, i.e. the PICC or MF level is currently selected. An error code is also returned if no write access has been done to any of the backup file types and, in case the application is protected by the Transaction MAC feature, no Transaction MAC calculation update has been done. This means no command that is included in the Transaction MAC calculation has been executed, i.e. the TMI is still the empty string. This is similar as for CommitTransaction.

The AbortTransaction requires CommMode.MAC if there is an active authentication.

It can be useful to cancel a transaction with the AbortTransaction command because there is no need for re-authentication to the PICC or no need for a reset of the RF field and a complete new ISO/IEC 14443 activation, which would have the same effect.

10.3 Transaction MAC

The Transaction MAC feature helps preventing e.g. fraudulent merchant attacks. It allows a merchant operating a point-of-sale terminal to prove that the transactions he executed with customers are genuine toward the card issuer's or application provider's backend.

This is done by letting the card generate a Transaction MAC over the transaction with a key shared only by the card and the card issuer's (or application provider's) backend. This key is called the AppTransactionMACKey. As this key will not be present in the merchant's terminal, it is not possible for the merchant to forge MACs toward the backend. Note that this implies that (pre-)personalization of the card is done by the card issuer and not at the merchant's location.

MAC generation will also involve a Transaction MAC Counter maintained by the card, which allows the backend to detect replay by the merchant. In addition this counter allows the backend to detect missing transactions, which allows detecting fraud where the merchant has increased the balance without reporting the transaction. Note that a counter also allows for easier detection mechanisms of replay than keeping logs with fingerprints of every transaction.

Fraud detection alone might not be sufficient if one cannot point to the fraudulent merchant. Therefore, an additional configuration option requires that the merchant's terminal commits its identity during the transaction. This requires support of a SAM

(Secure Access Module) which allows a secure commit of an ID that is then linked with the merchant. The goal of this is that the merchant cannot forge the committed ID. This ID is called ReaderID. For this feature, a key is assigned which is known by both the card and the SAM: AppCommitReaderIDKey. At the merchant's site, this key should not leave the SAM.

The card supports linking ReaderIDs of merchants by storing the committed ReaderID and returning the ReaderID of the previous transaction. This way the backend can detect which merchant did not report missing transactions.

10.3.1 General Overview

The Transaction MAC can be activated by creating a TransactionMAC file. When creating this file, one also specifies the AppTransactionMACKey, see Section 11.7.6. Additionally, one can optionally also require the commitment of a ReaderID, by assigning a AppCommitReaderIDKey for this, see Section 8.2.3.6. Once the Transaction MAC feature is activated, the Transaction MAC computation consists of the following steps:

1. Initialization: before performing the first command that needs a Transaction MAC computation update, the Transaction MAC Session Keys (SesTMMACKey and SesTMENCKey) are derived from the AppTransactionMACKey using the next Transaction MAC Counter value, see also Section 10.3.2.1.
2. Computation updates: during the transaction, the Transaction MAC is computed using SesTMMACKey over the following commands:
   - ReadData, GetValue and ReadRecords
   - WriteData, Credit, Debit, LimitedCredit, WriteRecord, UpdateRecord and ClearRecordFile
   - CommitReaderID: the committed ReaderID is stored on the card and the ReaderID of the previous transaction is returned encrypted with the SesTMENCKey. This is also included in the ongoing Transaction MAC computation.

   Note that for simplicity operations all file types are included. Note that StandardData can be written without transaction finalization with CommitTransaction. Therefore, the protection of StandardData files by the TransactionMAC is limited.
   MF2DL(H)x0 does support an option that excludes unauthenticated data management operations from the Transaction MAC calculation. This can be enabled with ChangeFileSettings.
3. Finalization: on CommitTransaction, the Transaction MAC is finalized. The update of the manipulated backup files and the computed Transaction MAC, the related counter and (optionally) the committed ReaderID, are one atomic operation. The computed Transaction MAC and the related counter are either provided with the response of CommitTransaction or can be read afterwards using ReadData on the TransactionMAC file.

10.3.2 Cryptographic and Other Concepts

10.3.2.1 Transaction MAC Counter

Each Transaction MAC calculation is associated with a Transaction MAC Counter (TMC) which is a 4 byte unsigned integer. On the interface or in cryptographic calculations, the TMC is represented LSB first. The TMC is stored in the TransactionMAC file.

How the TMC is exactly processed depends on whether the application is using standard AES or LRP, as configured with PDCap2.1 via SetConfiguration, see Section 11.5.1.

Standard AES

The TMC is processed as a 4-byte integer. The TMC added by 1 serves as input for the Transaction MAC Session Key generation, see Section 10.3.2.3.

LRP

The TMC is composed by two 2-byte unsigned integer subparts:

actTMC: the actual Transaction MAC Counter subpart serves the actual role of TMC toward the backend. This means the PICC will only output a single TMV for an actTMC value, and therefore the backend can use this part for replay or missing transaction detection. At any point in time the current actTMC can be read from the TransactionMAC file and actTMC added by 1 serves as input for the Transaction MAC Session Key generation

sesTMC: the session Transaction MAC Counter subpart ensures that each session key generation results in different keys. sesTMC added by 1 serves as input for the Transaction MAC Session Key generation. On successful CommitTransaction, the current sesTMC is written to the TransactionMAC file as part of the stored TMC. On CommitTransaction, this non-user-accessible sesTMC is reset to zero. If sesTMC reached its maximum value (FFFFh), no Transaction MAC Session Key generation can be executed, as the counter would overflow. As a consequence, no data management commands, see Section 11.8 can be executed, except ReadData targeting the TransactionMAC file. Also CommitReaderID will be rejected. This means the application can still be selected, but with limited functionality. ISOSelectFile will be responded with error code 6283h, indicating that the selected file or application is deactivated. The counter can only be reset by recreating the TransactionMAC file.

For composing TMC the two subparts are concatenated as follows: actTMC || sesTMC. Both subparts are represented LSB first.

If configuring from Standard AES to LRP, see Section 11.5.1, the actTMC is set to the minimum of FFFFh and the current value of TMC. sesTMC is set to 0000h.

10.3.2.2 Transaction MAC Counter Limit

The number of transactions that can be executed within an application can be limited by setting a Transaction MAC Counter Limit (TMCLimit). This is an unsigned integer of 4 bytes (if configured for Standard AES, related with TMC) or 2 byte (if LRP, related with actTMC). On the interface or in cryptographic calculations, the TMC is represented LSB first.

At delivery, the TMCLimit is disabled. This is equivalent to holding the maximum value (FFFFFFFFh, as configured for Standard AES at that time). The TMCLimit can be enabled by setting a customized value with ChangeFileSettings. It can be retrieved with GetFileSettings.

Once the TMC (actTMC in case of LRP) equals the TMCLimit, no data management commands, see Section 11.8 can be executed, except ReadData targeting the TransactionMAC file. Also CommitReaderID will be rejected. This means the application can still be selected, but with limited functionality. ISOSelectFile will be responded with error response 6283h indicating that the selected file or application has been deactivated and provides limited functionality.

If configuring from standard AES to LRP, see Section 11.5.1, the TMCLimit is set to the minimum of FFFFh and its current value. This means it is possible that suddenly the maximum value is reached, i.e. if TMC at that point in time hold a value of FFFFh or larger. If the TMCLimit gets disabled with ChangeFileSettings, this is also equivalent to putting it to the maximum value: FFFFFFFFh if standard AES, or FFFFh if LRP. Also when the TransactionMAC file gets re-created with CreateTransactionMACFile, the feature is disabled and TMCLimit is set to the maximum value.

10.3.2.3 Transaction MAC Session Keys

Out of the AppTransactionMACKey, two session keys are generated:

- SesTMMACKey for computing the Transaction MAC Value
- SesTMENCKey for encrypting the committed ReaderID (if used)

The Transaction MAC Session Keys are derived using the following algorithms.

AES mode

The session key generation is according to NIST SP 800-108 [10] in counter mode.

The Pseudo Random Function PRF(key; message) applied during the key generation is the CMAC algorithm described in NIST Special Publication 800-38B [6]. The key derivation key is the AppTransactionMACKey, see Section 8.2.4.

The input data is constructed using the following fields as defined by [10]. Note that NIST SP 800-108 allows defining a different order than proposed by the standard as long as it is unambiguously defined.

- a 1-byte label, distinguishing the purpose of the key: 31h for MACing and 32h for encryption
- a 2-byte counter, fixed to 0001h as only 128-bit keys are generated.
- a 2-byte length, fixed to 0080h as only 128-bit keys are generated.
- a 11-byte context, constructed using the 4-byte TMC+1 and the UID

Two session vectors SV_x are derived as follows:

$$SV1 = 5Ah||00h||01h||00h||80h||(TMC+1)||UID$$
$$SV2 = A5h||00h||01h||00h||80h||(TMC+1)||UID$$

Then, the 16-byte session keys SesTMMACKey and SesTMENCKey are constructed as follows:

$$SesTMMACKey = PRF(AppTransactionMACKey, SV1)$$
$$SesTMENCKey = PRF(AppTransactionMACKey, SV2)$$

LRP mode

The session key generation is according to NIST SP 800-108 [10] in counter mode.

The Pseudo Random Function PRF(key; message) applied during the key generation is the CMAC algorithm on top of the LRP primitive. This is specified in [10], see also Section 9.2.3. The key derivation key is the AppTransactionMACKey.

The input data is constructed using the following fields as defined by [10]. Note that NIST SP 800-108 allows defining a different order than proposed by the standard as long as it is unambiguously defined.

- a 2-byte counter, fixed to 0001h as only 128-bit keys are generated.
- a 2-byte length, fixed to 0080h as only 128-bit keys are generated.
- a 11-byte context, constructed using the 2-byte actTMC+1, 2-byte sesTMC+1 and the UID, followed by zero byte padding if needed
- a 1-byte label, distinguishing the purpose of the key: 5Ah for MACing and A5h for encryption

Two session vectors SV x are derived as follows:

$$SV1 = 00h||01h||00h||80h||(actTMC+1)||(sesTMC+1)||UID||5Ah$$
$$SV2 = 00h||01h||00h||80h||(actTMC+1)||(sesTMC+1)||UID||A5h$$

No padding is done because the input using the 7 Byte UID is equal to 16 byte.

Then, the 16-byte session keys SesTMMACKey and SesTMENCKey are constructed as follows:

$SesTMMACKey = MAC_{LRP}$ (AppTransactionMACKey; $SV1$)

$SesTMENCKey = MAC_{LRP}$ (AppTransactionMACKey; $SV2$)

After the session key calculation, but before the first data operation, the non-user-accessible sesTMC is updated with the incremented value, such that a subsequent session key generation will use a new value.

10.3.2.4 Transaction MAC Value and Input

The 8-byte Transaction MAC Value (TMV) is computed over the Transaction MAC Input (TMI). This input depends on the commands executed during the transaction, see Section 10.3.4. The applied key is SesTMMACKey, defined in Section 10.3.2.3.

A different notation than the one for Secure Messaging MACs is used: $MACt_{TM}$*(key; message)* is used to denote the CMAC operation including truncation. MAC_{TM}*(key; message)* denotes the CMAC result before truncation. Note that, though similar algorithm as for Secure Messaging are used, this MAC calculation is unrelated with the secure messaging itself as a different key is applied.

The TMV is calculated as follows:

$$TMV = MACt_{TM}(SesTMMACKey, TMI)$$

using the MAC algorithm of the Secure Messaging with zero byte IV, see Section 9.1.3.

Note that even if the application is configured for LRP, standard AES CMAC is used for the TMV calculation. LRP is only used for the session key generation.

10.3.2.5 Transaction MAC Reader ID and its encryption

A Transaction MAC ReaderID is 16 bytes. If ReaderID commitment is enabled, see Section 10.3.4.3, two ReaderIDs are maintained by the card.

- TMRICur: the current ReaderID of the ongoing transaction. It is set with CommitReaderID

- TMRIPrev: the ReaderID of the latest successful transaction. On successful execution of CommitTransaction, TMRICur is stored on the PICC as TMRIPrev

During the next transaction, the TMRIPrev is returned encrypted using SesTMENCKey. This is done via the EncTMRI parameter in the response of the CommitReaderID:

$$EncTMRI = E_{TM}(\text{SesTMENCKey}, \text{TMRIPrev})$$

using the AES block cipher according to the CBC mode of NIST SP800-38A [5] without adding any padding. The zero byte IV is applied, i.e. 128 bits of 0.

The initial TMRIPrev, as configured during creation of the TransactionMAC, is set to all zero bytes, see Section 11.7.6.

Note that even if the application is configured for LRP, standard AES encryption is used for the EncTMRI calculation. LRP is only used for the session key generation. The exact specification of the TMRI is out of scope for this document, but an example can be the 7-byte SAM UID with padding.

10.3.3 Transaction MAC Enabling

For MF2DL(H)x0, the Transaction MAC feature is enabled by default due to the existence of the TransactionMAC file. The Transaction MAC feature can be disabled for an application by deleting the TransactionMAC file with DeleteTransactionMACFile. The Transaction MAC feature can be re-enabled by creating the TransactionMAC file again with CreateTransactionMACFile, see Section 11.7.6.

As soon as the Transaction MAC feature is enabled for a selected application, the Transaction MAC will be calculated, as specified in the next section, for every transaction within the currently selected application, starting with the next transaction.

MF2DL(H)x0 supports additional Transaction MAC-related configuration options which can be set via ChangeFileSettings:

- enabling a Transaction MAC counter limit TMCLimit.
- excluding unauthenticated operations from Transaction MAC calculation.

The AppTransactionMACKey is not an application key. Instead it is fully specified and set by the CreateTransactionMACFile. Therefore, if this key needs to be changed, the TransactionMAC file needs to be deleted and re-created. Notice that by doing this, the TMC is also reset to zero.

Note that for MF2DL(H)x0 it is highly recommended to delete the initial TransactionMAC file and re-create a new TransactionMAC file using a confidential AppTransactionMACKey.

10.3.4 Transaction MAC Calculation

10.3.4.1 Transaction MAC Initiation

A Transaction MAC calculation is initiated on transaction start, if the TransactionMAC file is present. Next to this, a new Transaction MAC calculation is initiated each time a transaction with ongoing Transaction MAC calculation is committed successfully with CommitTransaction, or aborted, as described in Section 10.2.

Initiating a Transaction MAC calculation consists of the following steps:

- Set TMI to the empty byte string.
- Set TMRICur to the empty byte string.

Note that AbortTransaction can be used to exclude data read (ReadData, ReadRecords or GetValue) from the Transaction MAC calculation if this data does not need to be authenticated via the Transaction MAC toward the backend.

10.3.4.2 Transaction MAC Update

If the Transaction MAC Input TMI is still empty, the Transaction MAC Session Keys (SesTMMACKey and SesTMENCKey), as defined in Section 10.3.2.3, are calculated. Note that the calculation of SesTMENCKey may be delayed until CommitReaderID.

Once a Transaction MAC calculation is ongoing, the Transaction MAC Input TMI gets updated on each following data manipulation command targeting a file of any file type within the application, except TransactionMAC file itself.

The affected commands are listed below including the exact TMI updates. The following holds for all commands:

ZeroPadding is the minimal number of zero bytes added such that the length of the TMI up to and including the *ZeroPadding* is a multiple of 16 bytes. Note that this padding is also added if this TMI update is not the last one before CommitTransaction.

Note that if executed while in not authenticated state (in case the access rights allow), the command can be excluded from Transaction MAC processing, according to the TransactionMAC configuration as can be done with ChangeFileSettings. In that case, TMI is not updated at all.

Note that for each of these commands always the plain data are added, independently from the actual communication settings and secure messaging (i.e. plain data without any MAC, CRC, padding,. . .). In case of command chaining, the chaining overhead is ignored for the Transaction MAC computation. *Data* is the complete response data of all frames with a total byte length of *Length*.

Except otherwise noted in the commands, all parameters are exactly as they appear on the command interface.

If TMCLimit was reached, see Section 10.3.2.2, the command is rejected.

ReadData command TMI update

TMI = TMI || *Cmd* || *FileNo* || *Offset* || *Length* || *ZeroPadding* || *Data* || *ZeroPadding*

If *Length* is set to 000000h in the command, meaning that the whole file is read, the actual length value is filled in with the number of bytes read for the Transaction MAC calculation.

The Transaction MAC is not updated when the TransactionMAC file is targeted with the ReadData command.

WriteData command TMI update

TMI = TMI || *Cmd* || *FileNo* || *Offset* || *Length* || *ZeroPadding* || *Data* || *ZeroPadding*

Note that the first ZeroPadding for the WriteData command is actually adding 8 zero bytes after the command parameter fields so that those and the padding add up to 16 bytes.

GetValue command TMI update

TMI = TMI || *Cmd* || *FileNo* || *Value* || *ZeroPadding*

Credit command TMI update

TMI = TMI || *Cmd* || *FileNo* || *Value* || *ZeroPadding*

Debit command TMI update

TMI = TMI || *Cmd* || *FileNo* || *Value* || *ZeroPadding*

LimitedCredit command TMI update

TMI = TMI || *Cmd* || *FileNo* || *Value* || *ZeroPadding*

ReadRecords command TMI update

TMI = TMI || *Cmd* || *FileNo* || *RecNo* || *RecCount* || *ZeroPadding* || *Data*

If *RecCount* is set to 000000h in the command, meaning that all records are read (starting from the *RecNo*), the actual length value is filled in with the number of records read for the TM computation.

Note that ZeroPadding for the ReadRecords command is actually adding 8 zero bytes after the command parameter fields so that those and the padding add up to 16 bytes. As the data is always a multiple of 16 bytes, no padding is needed at the end of the TMI.

WriteRecord command TMI update

TMI = TMI || *Cmd* || *FileNo* || *Offset* || *Length* || *ZeroPadding* || *Data*

Note that ZeroPadding for the WriteRecord command is actually adding 8 zero bytes after the command parameter fields so that those and the padding add up to 16 bytes. As the data is always a multiple of 16 bytes, no padding is needed at the end of the TMI.

UpdateRecord command TMI update

TMI = TMI || *Cmd* || *FileNo* || *RecNo* || *Offset* || *Length* || *ZeroPadding* || *Data*

Note that ZeroPadding for the UpdateRecord command is actually adding 5 zero bytes after the command parameter fields so that those and the padding add up to 16 bytes. As the data is always a multiple of 16 bytes, no padding is needed at the end of the TMI.

ClearRecordFile command TMI update

TMI = TMI || *Cmd* || *FileNo* || *ZeroPadding*

10.3.4.3 CommitReaderID Command

The application can be configured to require a reader to securely commit to a ReaderID during transaction. Typically this requires the support of a SAM inside the reader such that the ReaderID cannot be forged by a fraudulent attacker (and also the key required to commit the ReaderID cannot be known). For example, the ReaderID can be the SAM UID. The SAM command API will need to ensure that the ReaderID value itself cannot be manipulated by an attacker. In this case ReadWrite of the TransactionMAC file will be configured to require authentication with a specific key.

As specified in Section 8.2.3.6, ReadWrite of the TransactionMAC file can also be configured to allow free CommitReaderID access. This allows for use cases where this command can be used to add reader data (ID, time, GPS location, etc.) to the Transaction MAC calculation. However, it should not be used for the use case where this command is used to be able to track unreported and missing transactions.

Committing a ReaderID is possible with the command CommitReaderID as defined in Section 11.9.3. Note that CommitTransaction will be rejected if CommitReaderID is required and has not been executed during the ongoing transaction. This is even the case if ReadWrite of the TransactionMAC file is configured for free access.

Depending on the configuration of ReadWrite of the TransactionMAC file, an authentication is required with AppCommitReaderIDKey, see Section 8.2.3.6. If the selected application does not hold such a file or committing the ReaderID is disabled according to this access right, the command is rejected.

The command is also rejected if a ReaderID has already been committed during the ongoing transaction.

The parameter TMRI is the committed ReaderID and is assigned to TMRICur. If authenticated, this value is stored as TMRIPrev on a successful CommitTransaction. If not authenticated, the value is only used for the Transaction MAC computation.

The Transaction MAC Input (TMI) is updated as follows including the received ReaderID, i.e. TMRICur, and if authenticated, the encrypted ReaderID of the latest successful transaction, i.e. EncTMRI:

if authenticated: TMI = *TMI || Cmd || TMRICur || EncTMRI || ZeroPadding*
if not authenticated: TMI = *TMI || Cmd || TMRICur || ZeroPadding*

ZeroPadding is the minimal number of zero bytes added such that the length of the TMI up to and including the ZeroPadding is a multiple of 16 byte. Note that this padding is also added if this TMI update is not the last one before CommitTransaction. If authenticated, CommitReaderID will provide in its response the encrypted ReaderID of the previous transaction, as defined in Section 10.3.2.5. Note that this encryption is independent of the secure messaging as it is with a key derived from AppTransactionMACKey. If not authenticated, no data is returned and the given TMRICur is discarded. Note that even if ReadWrite is configured for free access, it is possible to execute the command authenticated, which will trigger the TMRIPrev processing, i.e. updating the value and replying the encrypted ReaderID. The CommitReaderID requires CommMode.MAC if there is an active authentication. If TMCLimit was reached, see Section 10.3.2.2, the command is rejected.

10.3.4.4 Transaction MAC Finalization

The Transaction MAC computation is successfully finalized on CommitTransaction as described in Section 10.2.

The Transaction MAC is computed as defined in Section 10.3.2.4.

If a transaction is aborted, either with AbortTransaction or implicitly by some other event, the ongoing Transaction MAC calculation is also aborted. This is described in Section 10.2.2.

11 Command set

11.1 Introduction

This section contains the full command set of MIFARE DESFire Light.

The command set is, where applicable, a compatible subset of MIFARE DESFire EV2 commands. Therefore, some parameters like the Option parameter for the SetConfiguration command may seem to have arbitrary values assigned.

Remark: In the figures and tables, always CommMode.Plain is presented and the field length is valid for the plain data length. For the CommMode.MAC and CommMode.Full, the cryptogram needs to be calculated according to the secure messaging Section 9, then data field needs to fill with the cryptogram (Plain; CMAC; encrypted data with CMAC). Communication mode and condition are mentioned in the command description.

11.2 Supported commands and APDUs

Table 19. APDUs

Command	C-APDU (hex)							R-APDU (hex)		Communication mode
INS	CLA	INS	P1	P2	Lc	Data	Le	Data	SW1SW2 Succ essful	
AbortTransaction	90	A7	00	00	XX	-	00	-	9100	MAC
AuthenticateEV2First - Part1	90	71	00	00	XX	Data	00	Data	91AF	N/A (command specific)
AuthenticateEV2First - Part2	90	AF	00	00	20	Data	00	Data	9100	
AuthenticateEV2NonFirst - Part1	90	77	00	00	XX	Data	00	Data	91AF	N/A (command specific)
AuthenticateEV2NonFirst - Part2	90	AF	00	00	20	Data	00	Data	9100	
AuthenticateLRPFirst - Part1	90	71	00	00	XX	Data	00	Data	91AF	N/A (command specific)
AuthenticateLRPFirst - Part2	90	AF	00	00	20	Data	00	Data	9100	
AuthenticateLRPNonFirst - Part1	90	77	00	00	XX	Data	00	Data	91AF	N/A (command specific)
AuthenticateLRPNonFirst - Part2	90	AF	00	00	20	Data	00	Data	9100	
ChangeFileSettings	90	5F	00	00	XX	Data	00	-	9100	CommMode.Full
ChangeKey	90	C4	00	00	XX	Data	00	-	9100	CommMode.Full
ClearRecordFile	90	EB	00	00	XX	Data	00	-	9100	CommMode.MAC
CommitReaderID	90	C8	00	00	XX	Data	00	Data	9100	CommMode.MAC
CommitTransaction	90	C7	00	00	XX	XX	00	Data	9100	CommMode.MAC
CreateTransactionMACFile	90	CE	00	00	XX	Data	00	-	9100	CommMode.Full
Credit	90	0C	00	00	XX	Data	00	-	9100	Comm. mode of targeted file
Debit	90	DC	00	00	XX	Data	00	-	9100	Comm. mode of targeted file
DeleteTransactionMACFile	90	DF	00	00	01	File ID	00	-	9100	CommMode.MAC
GetCardUID	90	51	00	00	-	-	00	Data	9100	CommMode.Full
GetFileIDs	90	6F	00	00	-	-	00	Data	9100	CommMode.MAC
GetFileSettings	90	F5	00	00	01	File number	00	Data	9100	CommMode.MAC

Command	C-APDU (hex)							R-APDU (hex)		Communication mode
GetISOFileIDs	90	61	00	00	-	-	00	Data	9100	CommMode.MAC
GetKeyVersion	90	64	00	00	01	Key number	00	Data	9100	CommMode.MAC
GetValue	90	6C	00	00	01	File number	00	Data	9100	Comm. mode of targeted file
GetVersion - Part1	90	60	00	00	-	-	00	Data	91AF	CommMode.MAC [1]
GetVersion - Part2	90	AF	00	00	-	-	00	Data	91AF	CommMode.MAC
GetVersion - Part3	90	AF	00	00	-	-	00	Data	9100	CommMode.MAC
LimitedCredit	90	1C	00	00	01	Data	00	-	9100	Comm. mode of targeted file
ISOReadBinary	00	B0	XX	XX	-	-	XX	Data	9000	CommMode.Plain
ReadData	90	AD	00	00	XX	Reference	00	Data	9100	Comm. mode of targeted file
ReadRecords	90	AB	00	00	XX	Data	00	Data	9100	Comm. mode of targeted file
Read_Sig	90	3C	00	00	01	00	00	Data	9100	CommMode.Full
ISOSelectFile	00	A4	XX	XX	XX	Data to send	XX	FCI	9000	CommMode.Plain
SetConfiguration	90	5C	00	00	XX	Data	00	-	9100	CommMode.Full
ISOUpdateBinary	00	D6	XX	XX	XX	Data to write	-	-	9000	CommMode.Plain
UpdateRecord	90	BA	00	00	XX	Data	00	-	9100	Comm. mode of targeted file
WriteData	90	8D	00	00	XX	Data	00	-	9100	Comm. mode of targeted file
WriteRecord	90	8B	00	00	XX	Data	00	-	9100	Comm. mode of targeted file

[1] MAC on command and returned with the last response, calculated over all 3 responses

11.3 Status word

Table 20. SW1 SW2 for CLA byte 0x90

SW1 SW2	Name	Description
0x9100	OPERATION_OK	Successful operation.
0x910C	NO_CHANGES	No changes done to backup files, CommitTransaction / AbortTransaction not necessary.
0x911C	ILLEGAL_COMMAND_CODE	Command code not supported.
0x911E	INTEGRITY_ERROR	CRC or MAC does not match data. Padding bytes not valid.
0x9140	NO_SUCH_KEY	Invalid key number specified.
0x917E	LENGTH_ERROR	Length of command string invalid.
0x919D	PERMISSION_DENIED	Current configuration / status does not allow the re- quested command.
0x919E	PARAMETER_ERROR	Value of the parameter(s) invalid.
0x91AD	AUTHENTICATION_DELAY	Currently not allowed to authenticate. Keep trying until full delay is spent.
0x91AE	AUTHENTICATION_ERROR	Current authentication status does not allow the re- quested command.
0x91AF	ADDITIONAL_FRAME	Additionaldata frame is expected to be sent.

SW1 SW2	Name	Description
0x91BE	BOUNDARY_ERROR	Attempt toread/write data from/to beyond the file's/record's limits. Attempt to exceed the limits of a value file.
0x91CA	COMMAND_ABORTED	Previous Command was not fully completed. Not all Frames were requested or provided by the PCD.
0x91DE	DUPLICATE_ERROR	Creation of file/application failed because file/ application with same number already exists.
0x91F0	FILE_NOT_FOUND	Specified file number does not exist.

Table 21. SW1 SW2 for CLA byte 0x00

SW1 SW2	Description
0x6283	Selected file or application deactivated: selected with limited functionality
0x6700	Wrong length; no further indication
0x6982	Security status not satisfied
0x6985	Conditions of use not satisfied
0x6A80	Incorrect parameters in the command data field
0x6A82	File or application not found
0x6A86	Incorrect parameters P1-P2
0x6A87	Lc inconsistent with parameters P1-P2
0x6C00	Wrong Le field
0x6CXX	Wrong Le field; SW2 encodes the exact number of avail- able data bytes.
0x6D00	Instruction code not supported or invalid
0x6E00	Class not supported
0x9000	Normal processing (no further qualification)

11.4 Authentication commands

Authentication with the defined key is required to access the protected file according to access rights. Based on successful authentication session keys are generated, which are used for secure messaging between the terminal and MF2DL(H)x0.

Remark: Default FWI settings for AES-based protocol for authentication and secure messaging without LRP are set according to GSMA specification v2.0 to the value 7h. This value is stored in the User ATS and is configurable. With the default setting of FWI value 7h and use of Leakage Resilient Primitive, LRP commands with the need of interrupt handling could abort. To avoid an abort of commands with the need of interrupt handling, the recommendation is to change the FWI value from 7h to 8h. The change can be applied by changing the configurable ATS within the Configuration Settings.

11.4.1 AuthenticateEV2First

This command initiates an authentication based on standard AES. After this authentication, AES secure messaging is applied. This authentication command is used to authenticate for the first time in a transaction and can always be used within a transaction. AuthenticateEV2First starts a transaction with a Transaction Identifier (TI) and AuthenticateEV2NonFirst continues the transaction with that TI. This 3-pass challenge-response-based mutual authentication command is completed in two parts:

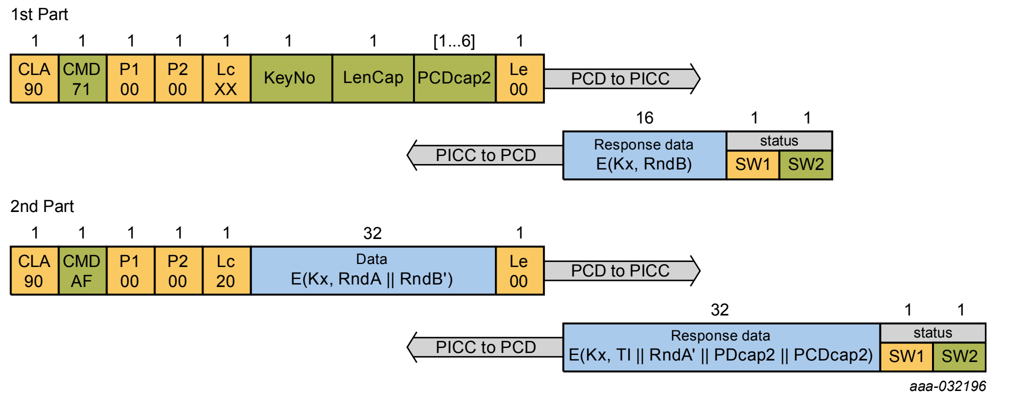


Figure 13. AuthenticateEV2First command protocol

Table 22. Command parameters description - AuthenticateEV2First - Part1

Name	Length	Value	Description
Command Header Parameters			
CMD	1	71h	Command code.
KeyNo	1		Targeted authentication key
	Bit 7-6	00b	RFU
	Bit 5-0	0h to 4h	Key number
LenCap	1	00h to 06h	Length of the PCD Capabilities. [This value should be set to 00h].
PCDcap2.1	[1]	-	Capability vector of the PCD.
	Bit 7-2	Full range	RFU, can hold any value
	Bit 1	0b	EV2 secure messaging

Name	Length	Value	Description
	Bit 0	Full range	RFU, can hold any value
PCDcap2.2-6	[1..5]	Full range	Capability vector of the PCD. All other bytes but PCDcap2.1 are optional, RFU and can hold any value. [If LenCap set to 00h, no PCDcap2 present]
Command Data Parameters			
-	-	-	No data parameters

Table 23. Response data parameters description - AuthenticateEV2First - Part1

Name	Length	Value	Description
E(Kx, RndB)	16	Full range	Encrypted PICC challenge The following data, encrypted with the key Kx referenced by KeyNo: - RndB: 16 byte random from PICC
SW1SW2	2	91AFh 91XXh	successful execution Refer to Table 23

Table 24. Return code description - AuthenticateEV2First - Part1

Status	Value	Description
COMMAND_ABORTED	CAh	Chained command or multiple pass command ongoing.
LENGTH_ERROR	7Eh	Command size not allowed.
PARAMETER_ERROR	9Eh	Parameter value not allowed.
NO_SUCH_KEY	40h	Targeted key does not exist
PERMISSION_DENIED	9Dh	Targeted key not available for authentication.
		Targeted key not enabled.
		Targeting EV2 authentication and secure messaging, while not allowed by configuration (PDCap2.1.Bit1 is ’1’).
AUTHENTICATION_DELAY	ADh	Currently not allowed to authenticate. Keep trying until full delay is spent.

Table 25. Command parameters description - AuthenticateEV2First - Part2

Name	Length	Value	Description
CMD	1	AFh	Additional frame
E(Kx, RndA || RndB')	32	Full range	Encrypted PCD challenge and response
			The following data, encrypted with the key Kx referenced by KeyNo: - RndA: 16 byte random from PCD. - RndB': 16 byte RndB rotated left by 1 byte

Table 26. Response data parameters description - AuthenticateEV2First - Part2

Name	Length	Value	Description
E(Kx, TI || RndA' || PDcap2 || PCDcap2)	32	Full range	Encrypted PICC response The following data encrypted with the key referenced by KeyNo: - TI: 4 byte Transaction Identifier - RndA': 16 byte RndA rotated left by 1 byte. - PDcap2: 6 byte PD capabilities - PCDcap2: 6 byte PCD capabilities
SW1SW2	2	9100h 91XXh	successful execution Refer to Table 27

Table 27. Return code description - AuthenticateEV2First - Part2

Status	Value	Description
LENGTH_ERROR	7Eh	Command size not allowed.
AUTHENTICATION_ERROR	AEh	Wrong RndB'
MEMORY_ERROR	EEh	Failure when reading or writing to non-volatile memory.

11.4.2 AuthenticateEV2NonFirst

The AuthenticateEV2NonFirst command can be used only if there is a valid authentication with AuthenticateEV2First. It continues the transaction with the transaction started by previous AuthenticateEV2First command. It starts a new session. The scheme of transaction and sessions within the transaction have been designed to protect any possible sophisticated replay attacks

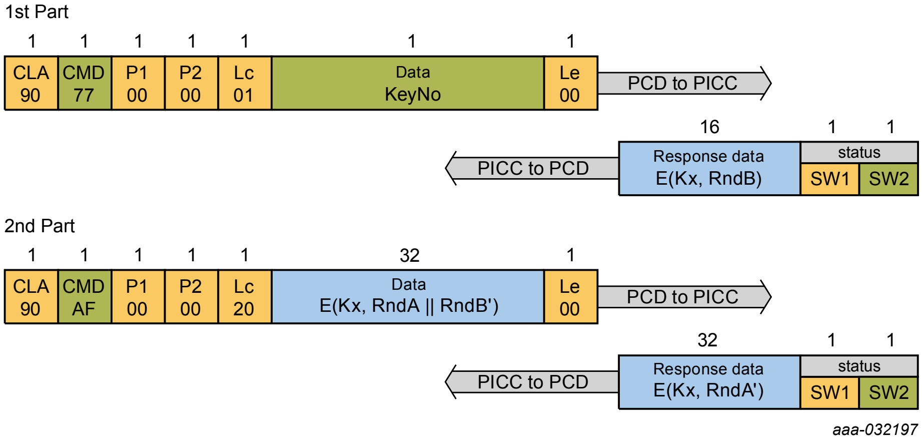


Figure 14. AuthenticateEV2NonFirst command protocol

Table 28. Command parameters description - AuthenticateEV2NonFirst - Part1

Name	Length	Value	Description
Command Header Parameters			
CMD	1	77h	Command code.
KeyNo	1		Targeted authentication key
	Bit 7-6	0	RFU
	Bit 5-0	0h to 04h	Key number
Command Data Parameters			
-	-	-	No data parameters

Table 29. Response data parameters description - AuthenticateEV2NonFirst - Part1

Name	Length	Value	Description
E(Kx, RndB)	16	Full range	Encrypted PICC challenge The following data, encrypted with the key Kx referenced by KeyNo: - RndB (16 byte): Random number from the PICC.
SW1SW2	2	91AFh 91XXh	successful execution Refer to Table 30

Table 30. Return code description - AuthenticateEV2NonFirst - Part1

Status	Value	Description
COMMAND_ABORTED	CAh	Chained command or multiple pass command ongoing.
LENGTH_ERROR	7Eh	Command size not allowed.
PARAMETER_ERROR	9Eh	Parameter value not allowed.
NO_SUCH_KEY	40h	Targeted key does not exist
PERMISSION_DENIED	9Dh	In not authenticated state and not targeting OriginalityKey
		Targeted key not available for authentication.
		Targeted key not enabled.
AUTHENTICATION_DELAY	ADh	Currently not allowed to authenticate. Keep trying until full delay is spent.
MEMORY_ERROR	EEh	Failure when reading or writing to non-volatile memory.

Table 31. Command parameters description - AuthenticateEV2NonFirst - Part2

Name	Length	Value	Description
CMD	1	AFh	Additional frame
E(Kx, RndA || RndB')	32	Full range	Encrypted PCD challenge and response
			The following data, encrypted with the key Kx referenced by KeyNo: - RndA: 16 byte random from PCD. - RndB': 16 byte RndB rotated left over 1 byte.

Table 32. Response data parameters description - AuthenticateEV2NonFirst - Part2

Name	Length	Value	Description
E(Kx, RndA')	16	Full range	Encrypted PICC challenge and response The following data, encrypted with the key Kx referenced by KeyNo: - RndA: 16 byte random from PCD. - RndB': 16 byte RndB rotated left over 1 byte.
SW1SW2	2	9100h 91XXh	successful execution Refer to Table 33

Table 33. Return code description - AuthenticateEV2NonFirst - Part2

Status	Value	Description
LENGTH_ERROR	7Eh	Command size not allowed.
AUTHENTICATION_ERROR	AEh	Wrong RndB'
MEMORY_ERROR	EEh	Failure when reading or writing to non-volatile memory.

11.4.3 AuthenticateLRPFirst

Authentication for LRP secure messaging. The AuthenticationLRPFirst is intended to be the first in a transaction and recommended. LRP secure messaging allows side-channel resistant implementations.

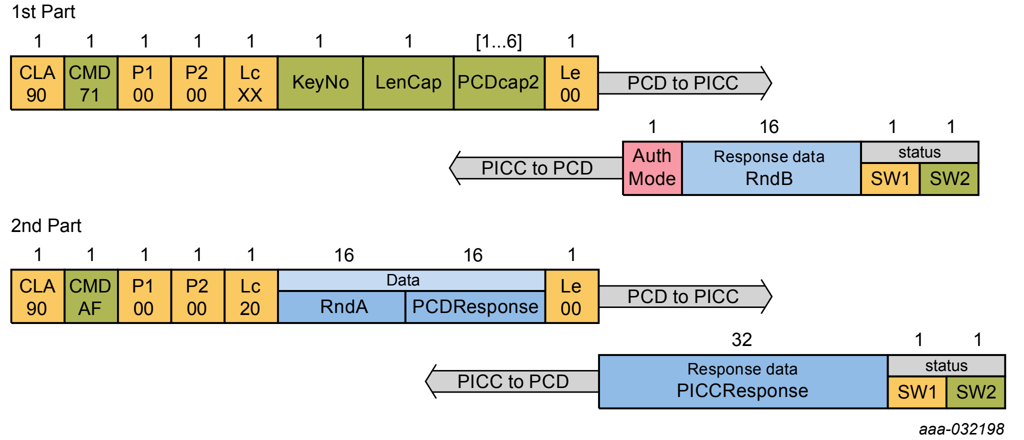


Figure 15. AuthenticateLRPFirst command protocol

Table 34. Command parameters description - AuthenticateLRPFirst - Part1

Name	Length	Value	Description
Command Header Parameters			
CMD	1	71h	Command code.
KeyNo	1		Targeted authentication key
	Bit 7-6	00b	RFU
	Bit 5-0	0h..4h	Key number
LenCap	1	1h..6h	Length of the PCD Capabilities.
PCDcap2.1	1	-	Capability vector of the PCD.
	Bit 7-2	Full range	RFU, can hold any value
	Bit 1	1b	LRP secure messaging
	Bit 0	Full range	RFU, can hold any value
PCDcap2.2-6	[1..5]	Full range	Capability vector of the PCD. All other bytes but PCDcap2.1 are optional, RFU and can hold any value.
Command Data Parameters			
-	-	-	No data parameters

Table 35. Response data parameters description - AuthenticateLRPFirst - Part1

Name	Length	Value	Description
AuthMode	1	01h	LRP Mode
RndB	16	Full range	PICC challenge RndB

Name	Length	Value	Description
SW1SW2	2	91AFh 91XXh	successful execution Refer to Table 36

Table 36. Return code description - AuthenticateLRPFirst - Part1

Status	Value	Description
COMMAND_ABORTED	CAh	Chained command or multiple pass command ongoing.
LENGTH_ERROR	7Eh	Command size not allowed.
PARAMETER_ERROR	9Eh	Parameter value not allowed.
NO_SUCH_KEY	40h	Targeted key does not exist.
PERMISSION_DENIED	9Dh	Targeted key is locked as related TotFailCtr is equal to or bigger than the TotFailCtrLimit.
AUTHENTICATION_DELAY	ADh	Currently not allowed to authenticate. Keep trying until full delay is spent.
MEMORY_ERROR	EEh	Failure when reading or writing to non-volatile memory.

Table 37. Command parameters description - AuthenticateLRPFirst - Part2

Name	Length	Value	Description
CMD	1	AFh	Additional frame
RndA	16	Full range	PCD challenge
PCDResponse	16	Full range	PCD response MAC_{LRP} *(SesAuthMACKey;RndA || RndB)*

Table 38. Response data parameters description - AuthenticateLRPFirst - Part2

Name	Length	Value	Description
PICCData	16	Full range	Encrypted PICC data, E_{LRP} (SesAuthENCKey; TI; PDCap2; PCDCap2)
PICCRespons e	16	Full range	PICC response to the challenge MAC_{LRP} *(SesAuthMACKey; RndB || RndA || PICCData)*
SW1SW2	2	9100h 91XXh	successful execution Refer to Table 39

Table 39. Return code description - AuthenticateLRPFirstPart2

Status	Value	Description
LENGTH_ERROR	7Eh	Command size not allowed.
AUTHENTICATION_ERROR	AEh	Wrong PCDResp
MEMORY_ERROR	EEh	Failure when reading or writing to non-volatile memory.

11.4.4 AuthenticateLRPNonFirst

Consecutive authentication for LRP secure messaging. After this authentication, LRP secure messaging is used. This authentication is intended to be the following authentication in a transaction.

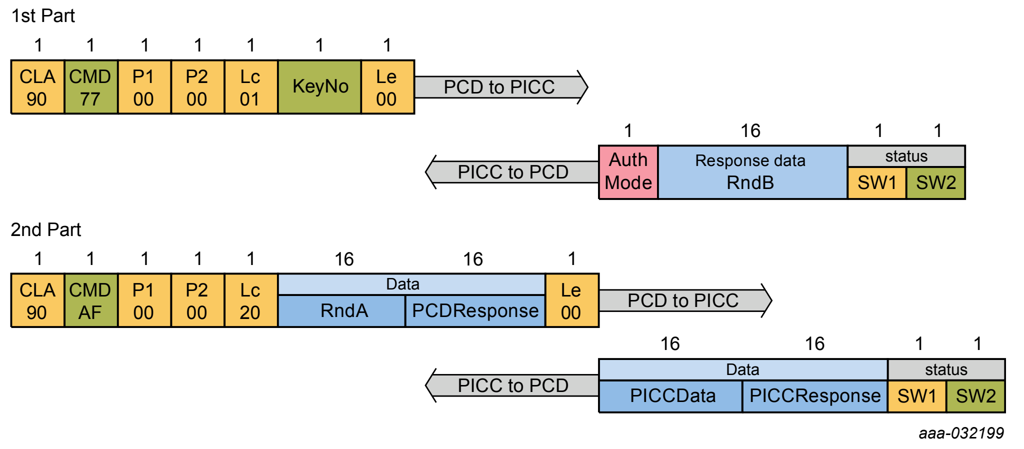


Figure 16. AuthenticationLRPNonFirst command protocol

Table 40. Command parameters description - AuthenticateLRPNonFirst - Part1

Name	Length	Value	Description
Command Header Parameters			
CMD	1	77h	Command code.
KeyNo	1		Targeted authentication key
	Bit 7-6	00b	RFU
	Bit 5-0	00h to 04h	Key Number
Command Data Parameters			
-	-	-	No data parameters

Table 41. Response data parameters description - AuthenticateLRPNonFirst - Part1

Name	Length	Value	Description
AuthMode	1	01h	LRP
RndB	16	Full range	PICC random RndB
SW1SW2	2	91AFh 91XXh	successful execution Refer to Table 42

Table 42. Return code description - AuthenticateLRPNonFirst - Part1

Status	Value	Description
COMMAND_ABORTED	CAh	Chained command or multiple pass command ongoing.
LENGTH_ERROR	7Eh	Command size not allowed.
PARAMETER_ERROR	9Eh	Parameter value not allowed.

Status	Value	Description
NO_SUCH_KEY	40h	Targeted key does not exist
PERMISSION_DENIED	9Dh	In not authenticated state and not targeting OriginalityKeys
		Targeted key not available for authentication.
		Targeted key not enabled.
AUTHENTICATION_DELAY	ADh	Currently not allowed to authenticate. Keep trying until full delay is spent.
MEMORY_ERROR	EEh	Failure when reading or writing to non-volatile memory.

Table 43. Command parameters description - AuthenticateLRPNonFirst - Part2

Name	Length	Value	Description
CMD	1	AFh	Additional frame
RndA	16	Full range	PCD challenge RndA
PCDResp	16	Full range	PCD response to the challenge MAC_{LRP} *(SesAuthMACKey;RndA || RndB)*

Table 44. Response data parameters description - AuthenticateLRPNonFirst - Part2

Name	Length	Value	Description
PICCResp	16	Full range	PICC response to the challenge MAC_{LRP} *(SesAuthMACKey;RndB || RndA)*
SW1SW2	2	9100h 91XXh	successful execution Refer to Table 45

Table 45. Return code description - AuthenticateLRPNonFirst - Part2

Status	Value	Description
LENGTH_ERROR	7Eh	Command size not allowed.
AUTHENTICATION_ERROR	AEh	Wrong PCDResp
MEMORY_ERROR	EEh	Failure when reading or writing to non-volatile memory.

11.5 Memory and configuration commands

11.5.1 SetConfiguration

With the SetConfiguration command, the application attributes can be configured. It requires an authentication with the AppMasterKey and CommMode.Full.

The command consists of an option byte and a data field with a size depending on the option. The option byte specifies the nature of the data field content.

In the below table “No change” references are used with configurations that are persistent. This means that the associated configuration is left as it is already in the card and its value is not changed.

Note that options 06h, 08h and 09h are one-time configurations which can only be issued once and are locked afterwards.

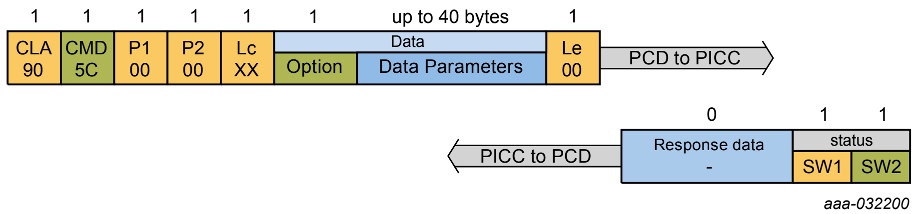


Figure 17. SetConfiguration command protocol

Table 46. Command parameters description - SetConfiguration

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	5Ch	Command code.
Option	1	-	Configuration Option. It defines the length and content of the Data parameter. The Option byte is transmitted in plain text, whereas the Data is always transmitted in CommMode.Full.
		00h	PICC configuration.
		02h	ATS update.
		03h	SAK update
		04h	Secure Messaging Configuration.
		05h	Capability data.
		06h	DF Name renaming
		08h	File renaming
		09h	Value file configuration
		0Ah	Failed authentication counter setting
		0Bh	HW configuration
		Other values	RFU
Command Data Parameters			
Data	Up to 20 bytes	-	Data content depends on option values.

Name	Length	Value	Description
		Full range	Data content depends on option value as defined in setConfigOptionsList Table.

Table 47. SetConfigOptionList

Option	Data Length	Field	Length/bitindex	Description
00h	1 byte	PICC Configuration		
		PICCConfig	Bit 7-6	RFU
			Bit 5	UID0 configuration for Random ID. If bit 1 is set to 0b, this bit should be 0b. 0b: [if Bit1 = 1b] Enable ISO-compliant random ID (UID0 = 08h). 1b: [if Bit1 = 1b] Enable legacy random ID compatible with MIFARE DESFire EV1 (UID0 = 80h).
			Bit 4 -2	RFU
			Bit 1	UseRID configuration for Random. Random ID is disabled at delivery time. 1b: Enable Random ID 0b: No change
			Bit 0	RFU
02h	5 to 20 bytes	User ATS	5 to 20 bytes	User-defined ATS. TL byte of the ATS should be as 0 < TL< 20. Please refer to ISO/IEC 14443-4 for ATS definition. TL: Length byte: 5 <TL <20 T0: Supported values: 75h or 77h. This means interface bytes always present. FSC of 64 byte or 128 byte. TA(1): Any value TB(1): 71h, 81h. FWI is set by default to 7h. In use of LRP, FWI value shall be changed to 8h. SFGI should be 1h. TC(1): 02h. CID supported, NAD not supported. Tk: Historical bytes: can hold any value
03h	2 bytes	User SAK	2 bytes	User-defined SAK1 and SAK2 each of one byte long formatted as SAK1 and SAK2. Refer to ISO/IEC 14443-3 for SAK1 and SAK2 definition.

Option	Data Length	Field	Length/bitindex	Description
04h	2 bytes	Secure Messaging Configuration		
		SMConfig	Bit 15 to 3	RFU
			Bit 2	Secure messaging configuration for StandardData file 0b: No Change 1b: disable chained writing with WriteData command in CommMode.MAC and CommMode.Full
			Bit 1-0	RFU
05h	10 bytes	Capability data, consisting of PDCap2		
			4 bytes	RFU
			1 byte	User configured PDCap2.1 Bit 7 to 2: RFU Bit 1: 1b means enable LRP mode. This change is permanent, LRP mode cannot be disabled afterwards. Bit 1: 0b means no change
			3 bytes	RFU
			1 byte	User configured PDCap2.5
			1 byte	User configured PDCap2.6
06h	17,19 bytes	Application ID renaming (one-time configuration)		
		AppNameOptions	1 byte	Bit 7 must be set to 1b and means update of the 2-byte ISOFile ID Bit 6 to 5: RFU Bit 4 to 0: Length of DF Name, must be in the range of 1 to 16
		DF Name	16 bytes	DF Name - up to 16 byte, padded with 00h bytes if needed
		ISO FileID	2 bytes	ISO File ID, LSB first Excluding the following values reserved by ISO: 0000h, 3F00h, 3FFFh, FFFFh
08h	3,5,9 bytes	File renaming (one-time configuration)		
		File Name Option	1 byte	Bit 7 to 2: RFU Bit 1: Set to 1b means update 2 files Bit 1: Set to 0b means update 1 file Bit 0: Set to 1b means update ISO File ID Bit 0: Set to 0b means no update of ISO File ID
		OldFileNo	1 byte	Current FileNo of the first file
		NewFileNo	1 byte	New FileNo for the first file

Option	Data Length	Field	Length/bitindex	Description
		NewFileID	2 bytes	[Optional, present if FileNameOptions Bit 0 = 1b] New ISOFileID for the first file. LSB first. Excluding the following values reserved by ISO: 0000h, 3F00h, 3FFFh, FFFFh
		OldFileNo2	1 byte	[Optional, present if FileNameOptions Bit 1 = 1b] Current FileNo of the second file
		NewFileNo2	1 byte	[Optional, present if FileNameOptions Bit 1 = 1b] New FileNo for the second file
		NewFileID2	2 bytes	[Optional, present if FileNameOptions Bit 1 = 1b AND File- NameOptions Bit 0 = 1b] New ISOFileID for the second file. LSB first. Excluding the following values reserved by ISO: 0000h, 3F00h, 3FFFh, FFFFh
09h	14 bytes	Value file configuration (one-time configuration)		
		File Nr	1 byte	FileNo of the targeted file
		Lower Limit	4 bytes	Lower Limit of the file encoded as a 4 byte signed integer, with LowerLimit less than or equal to UpperLimit
		UpperLimit	4 bytes	Upper Limit of the file encoded as a 4 byte signed integer, with LowerLimit less than or equal to UpperLimit.
		Value	4 bytes	Current Value with Value greater than or equal to Lower- Limit and less than or equal to UpperLimit.
		ValueOption	1byte	Bit 7 to 2: RFU Bit1: 0b means No free access to GetValue Bit1: 1b means free access to GetValue Bit 0: 0b means LimitedCredit command disabled Bit 0: 1b means LimitedCredit command enabled
0Ah	5 bytes	Failed authentication counter configuration		
		FailedCtrOption	1 byte	Bit 7 to 1: RFU Bit 0: Set to 0b for disabling Bit 0: Set to 1b for enabling [default]

Option	Data Length	Field	Length/bitindex	Description
		TotFailCtrLimit	2 bytes	configurable limit, encoded as 2-byte unsigned integer (LSB first), must be bigger than 0000h. Default value: 1000. When disabling, this value is ignored
		TotFailCtrDecr	2 bytes	configurable decrement value, encoded as 2-byte unsigned integer (LSB first). Default value: 10. When disabling, this value is ignored.
0Bh	1 byte	HW configuration		
		HW Option	1 byte	Bit 7 to 1: RFU Bit 0: Set to 0b for Standard back modulation Bit 0: Set to 1b for Strong back modulation (default)

Table 48. Response data parameters description - SetConfiguration

Name	Length	Value	Description
Response data	0	-	No response data
SW1SW2	2	9100h 91XXh	successful execution Refer to Table 49

Table 49. Return code description - SetConfiguration

Status	Value	Description
COMMAND_ABORTED	CAh	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	1Eh	Invalid cryptogram (padding or CRC). Invalid secure messaging MAC.
LENGTH_ERROR	7Eh	Command size not allowed. Option 00h: Data length is not 1 Option 02h: Data length is not in the range [5..20]. Option 03h: Data length is not 2 Option 04h: Data length is not 2 Option 05h: Data length is not 10 Option 06h: Data inconsistent length Option 08h: Data inconsistent length Option 09h: Data length is not 14 Option 0Ah: Data length is not 5 Option 0Bh: Data length is not 1

Status	Value	Description
PARAMETER_ERROR	9Eh	Parameter value not allowed. Option 00h: Data bit 7-6 or bit 4-2 or bit 0 not set to 0b. Option 02h: TL inconsistent with length of received ATS string or unsupported T0, TB(1) or TC(1) value Unsupported option (i.e. Reserved).
PERMISSION_DENIED	9Dh	Option 00h, 02h, 03h, 04h, 05h, 06h, 08h, 09h, 0Ah, 0Bh: not supported / allowed at PICC level Option 06h: application renaming not allowed anymore as already done once Option 08h: file renaming not allowed anymore as (for both files if updating two) it was already done once. Option 08h: trying to update ISOFileID while the targeted file does not have an ISOFileID. Option 09h: file configuration not allowed anymore as already done once or targeted file is not a Value file
FILE_NOT_FOUND	F0h	Option 08h: File with targeted OldFileNo1 or OldFileNo2 does not exist within the targeted application. Option 09h: File with targeted FileNo does not exist within the targeted application.
AUTHENTICATION_ERROR	AEh	Option 00h, 02h, 03h, 04h, 05h, 06h, 08h, 09h, 0Ah, 0Bh: No active authentication with AppMasterKey.
DUPLICATE_ERROR	DEh	Option 06h: executing application renaming would result in duplicate DFNames. Option 08h: executing file renaming would result in duplicate FileNos or ISOFileIDs.
MEMORY_ERROR	EEh	Failure when reading or writing to non-volatile memory.

11.5.2 GetVersion

The GetVersion command returns manufacturing related data of MIFARE DESFire Light (MF2DL(H)x0). No parameters are required for this command.

Remark: This command is only available after ISO/IEC 14443-4 activation.

The version data is return over three frames. Part1 returns the hardware-related information, Part2 returns the software-related information and Part3 and last frame returns the production-related information. This command is freely accessible without secure messaging as soon as the PD is selected and there is no active authentication.

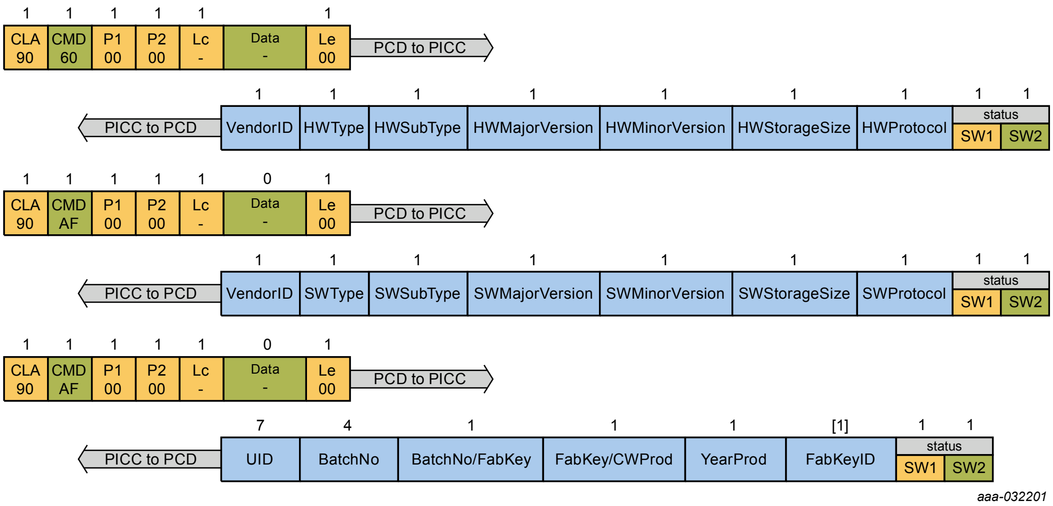


Figure 18. GetVersion command protocol

Part 1

Table 50. Command parameters description - GetVersion - Part1

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	60h	Command code.
Command Data Parameters			
-	-	-	No data parameters

Table 51. Response data parameters description - GetVersion - Part1

Name	Length	Value	Description
VendorID	1	04h	Vendor ID
HWType	1	08h	HW type for MIFARE DESFire Light
HWSubType	1	-	HW subtype
		X1h	17 pF
		X2h	50 pF
		0Xh	Strong back modulation
		8Xh	Standard back modulation

Name	Length	Value	Description
HWMajorVersion	1	30h	HW major version number
HWMinorVersion	1	00h	HW minor version number
HWStorageSize	1	-	HW storage size
		13h	512 B<storage size< 1 kB
		other values	RFU
HWProtocol	1	05h	HW communication protocol type
SW1SW2	2	91AFh 91XXh	successful execution Refer to Table 56

Part 2

Table 52. Command parameters description - GetVersion - Part2

Name	Length	Value	Description
CMD	1	AFh	Additional frame request.
Data	0	-	No data parameters:

Table 53. Response data parameters description - GetVersion - Part2

Name	Length	Value	Description
VendorID	1	04h	Vendor ID
SWType	1	08h	SW type for MIFARE DESFire Light
SWSubType	1	01h	SW subtype
SWMajorVersion	1	00h	SW major version number
SWMinorVersion	1	02h	SW minor version number
SWStorageSize	1	-	SW storage size
		13h	512 B<storage size< 1 kB
		other values	RFU
SWProtocol	1	05h	SW communication protocol type
SW1SW2	2	91AFh 91XXh	successful execution Refer to Table 56

Part 3

Table 54. Command parameters description - GetVersion - Part3

Name	Length	Value	Description
CMD	1	AFh	Additional frame request.
Data	0	-	No data parameters:

Table 55. Response data parameters description - GetVersion - Part3

Name	Length	Value	Description
UID	7	-	UID
		All zero	if configured for RandomID
		Full range	UID if not configured for RandomID
BatchNo	4	Full range	Production batch number
BatchNo/FabKey	1		
	Bit 7-4	Full range	Production batch number
	Bit 3-0	0h	Default FabKey, other values RFU
FabKey/CWProd	1		
	Bit 7	0b	Default FabKey, other values RFU
	Bit 6-0	01h..52h	Calendar week of production
YearProd	1	Full range	Year of production
FabKeyID	[1]	1Fh..FFh	Optional, present for customized configurations when FabKey = 1Fh
SW1SW2	2	9100h 91XXh	successful execution Refer to Table 56

Table 56. Return code description - GetVersion

Status	Value	Description
COMMAND_ABORTED	CAh	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	1Eh	Invalid secure messaging MAC (only).
LENGTH_ERROR	7Eh	Command size not allowed.
MEMORY_ERROR	EEh	Failure when reading or writing to non-volatile memory.

11.5.3 GetCardUID

GetCardUID command is required to get the 7-byte UID from the card. In case "Random ID" at activation is configured, encrypted secure messaging is applied for this command and response. 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.

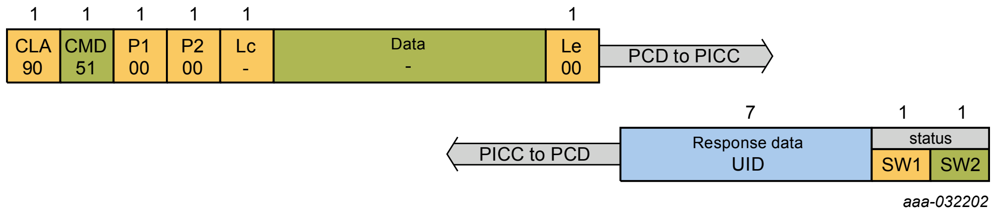


Figure 19. GetCardUID command protocol

Table 57. Command parameters description - GetCardUID

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	51h	Command code.
Command Data Parameters			
-	-	-	No data parameters

Table 58. Response data parameters description - GetCardUID

Name	Length	Value	Description
UID	7	Full range	UID of the MF2DL(H)x0
SW1SW2	2	9100h 91XXh	successful execution Refer to Table 59

Table 59. Return code description - GetCardUID

Status	Value	Description
COMMAND_ABORTED	CAh	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	1Eh	Invalid secure messaging MAC (only).
LENGTH_ERROR	7Eh	Command size not allowed.
AUTHENTICATION_ERROR	AEh	No active authentication
MEMORY_ERROR	EEh	Failure when reading or writing to non-volatile memory.

11.6 Key management commands

MF2DL(H)x0 provides the following command set for Key Management.

11.6.1 ChangeKey

The ChangeKey command is used to change the application keys. Authentication with application key number 0 is required to change the key. CommMode.Full is applied for this command. Note that the cryptogram calculations for changing key number 0 and other keys are different.

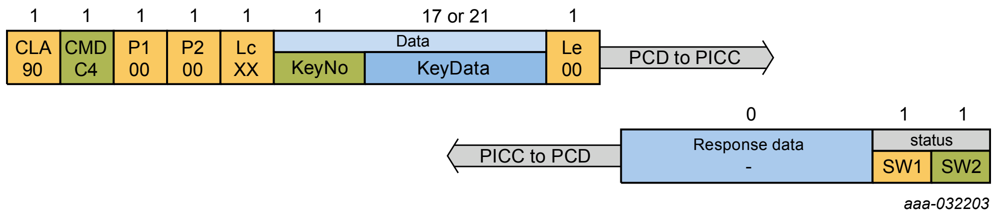


Figure 20. ChangeKey command protocol

Table 60. Command parameters description - ChangeKey

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	C4h	Command code.
KeyNo	1	-	Key number of the key to be changed.
	Bit 7-6	00b	RFU
	Bit 5-0		Key Number
		0h..4h	The application key number
Command Data Parameters			
KeyData	17 or 21		New key data.
		full range (17-byte length)	if key 0 is to be changed NewKey || KeyVer
		full range (21-byte length)	if key 1 to 4 are to be changed (NewKey XOR OldKey) || KeyVer || CRC32NK [1]

[1] The CRC32NK is the 4-byte CRC value computed according to IEEE Std 802.3-2008 (FCS Field) over NewKey [9]

Table 61. Response data parameters description - ChangeKey

Name	Length	Value	Description
Response data	0	-	No response data
SW1SW2	2	9100h 91XXh	successful execution Refer to Table 62

Table 62. Return code description - ChangeKey

Status	Value	Description
COMMAND_ABORTED	CAh	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	1Eh	Integrity error in cryptogram or invalid secure messaging MAC (Secure Messaging).
LENGTH_ERROR	7Eh	Command size not allowed.
PARAMETER_ERROR	9Eh	Parameter value not allowed.
NO_SUCH_KEY	40h	Targeted key does not exist
PERMISSION_DENIED	9Dh	At PICC level, targeting any OriginalityKey which cannot be changed
AUTHENTICATION_ERROR	AEh	At application level, missing active authentication with AppMasterKey while targeting any AppKey.
MEMORY_ERROR	EEh	Failure when reading or writing to non-volatile memory.

11.6.2 GetKeyVersion

The GetKeyVersion command retrieves the current key version of any key. Key version can be changed with the ChangeKey command together with the key.

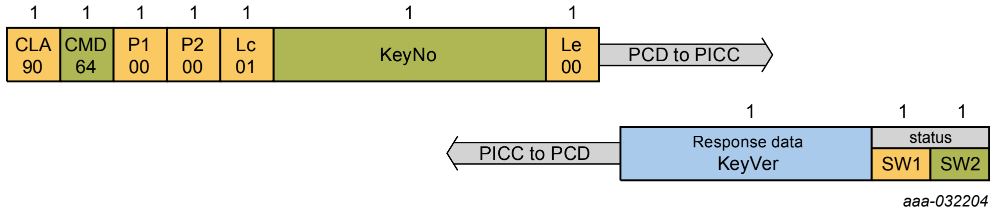


Figure 21. GetKeyVersion command protocol

Table 63. Command parameters description - GetKeyVersion

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	64h	Command code.
KeyNo	1	-	Key number of the targeted key
	Bit 7-4	00h	RFU
	3 to 0	00h to 04h	Application key number
Command Data Parameters			
-	-	-	No data parameters

Table 64. Response data parameters description - GetKeyVersion

Name	Length	Value	Description
KeyVer	1	-	Key version of the targeted key
		00h	[if targeting disabled keys]
		00h	[if targeting OriginalityKey]
		Full range	[else]
SW1SW2	2	9100h 91XXh	successful execution Refer to Table 65

Table 65. Return code description - GetKeyVersion

Status	Value	Description
COMMAND_ABORTED	CAh	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	1Eh	Invalid secure messaging MAC (only).
LENGTH_ERROR	7Eh	Command size not allowed.
PARAMETER_ERROR	9Eh	Parameter value not allowed.
NO_SUCH_KEY	40h	Targeted key does not exist.
MEMORY_ERROR	EEh	Failure when reading or writing to non-volatile memory.

11.7 File management commands

The MF2DL(H)x0 provides the following command set for File Management functions.

11.7.1 ChangeFileSettings

The ChangeFileSettings command changes the access parameters of an existing file. The communication mode can be either CommMode.Plain or CommMode.Full based on current access right of the file.

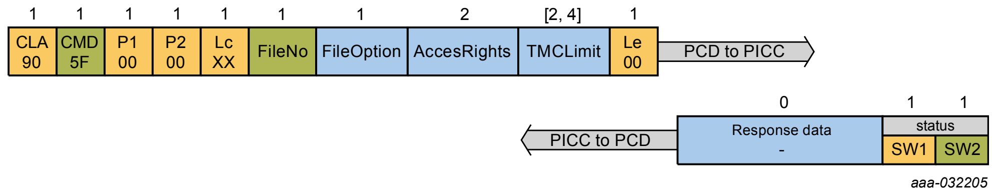


Figure 22. ChangeFileSettings command protocol

Table 66. Command parameters description - ChangeFileSettings

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	5Fh	Command code.
FileNo	1	-	File number of the targeted file.
	Bit 7-5		RFU
	Bit 4-0		File number
Command Data Parameters			
FileOption	1	-	Options for the targeted file.
	Bit 7-6	00b	RFU
	Bit 5		if targeting a TransactionMAC file: TMCLimit configuration
		0b	disabled
		1b	enabled
	Bit 5	0b	RFU for all other file types
	Bit 4		if targeting a TransactionMAC file: exclude unauthenticated operations from TMI
		0b	disabled
		1b	enabled
	Bit 4	0b	RFU for all other file types
	Bit 3-2	00b	RFU
	Bit 1-0		CommMode (see Table CommunicationModes)
AccessRights	2	-	Set of access conditions for the first set in the file (see Section 8.2.3.5).
TMCLimit	[2,4]	-	[Optional, present if FileOption[Bit 5] = 1b] TMCLimit

Name	Length	Value	Description
		00000001h .. FFFFFFFFh	[if AES mode] TMCLimit value (4 byte)
		0001h .. FFFFh	[if LRP mode] TMCLimit value (2 byte)

Table 67. Response data parameters description - ChangeFileSettings

Name	Length	Value	Description
Response data	0	-	No response data
SW1SW2	2	9100h 91XXh	successful execution Refer to Table 68

Table 68. Return code description - ChangeFileSettings

Status	Value	Description
COMMAND_ABORTED	CAh	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	1Eh	Integrity error in cryptogram. Invalid Secure Messaging MAC (only).
LENGTH_ERROR	7Eh	Command size not allowed.
PARAMETER_ERROR	9Eh	Parameter value not allowed.
		Targeted key for one of the access conditions in AccessRights does not exist.
		Trying to set a TMCLimit which is smaller or equal to the current TMC.
		Trying to exclude unauthenticated operations from TMI (FileOption Bit 4 is set) while ReadWrite is set to Eh, i.e. enabling free CommitReaderID.
PERMISSION_DENIED	9Dh	PICC level (MF) is selected.
		File access right Change of targeted file has access conditions set to Fh.
FILE_NOT_FOUND	F0h	File with targeted FileNo does not exist for the targeted application.
AUTHENTICATION_ERROR	AEh	File access right Change of targeted file not granted as there is no active authentication with the required key while the access conditions is different from Fh.
MEMORY_ERROR	EEh	Failure when reading or writing to non-volatile memory.

11.7.2 GetFileSettings

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

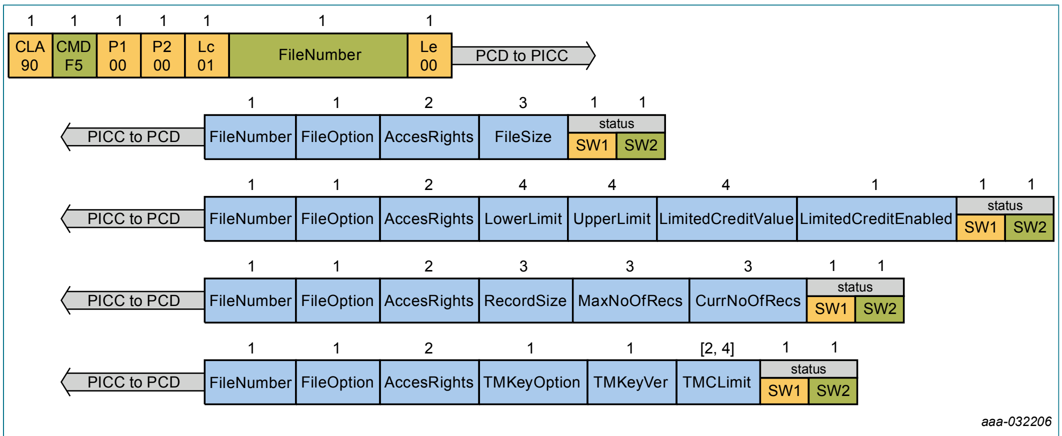


Figure 23. GetFileSettings command protocol

Table 69. Command parameters description - GetFileSettings

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	F5h	Command code.
FileNo	1	-	File number of the targeted file.
	Bit 7-5		RFU
	Bit 4-0		File number
Command Data Parameters			
-	-	-	No data parameters

Table 70. Response data parameters description - GetFileSettings - Targeting Data File

Name	Length	Value	Description
FileType	1	-	File Type of the targeted file.
		00h	StandardData File
		Other values	RFU
FileOption	1	-	Options for the targeted file.
	Bit 7-2		RFU
	Bit 1-0		CommMode (see Table CommunicationModes)
AccessRights	2	-	Set of access conditions for the 1st set in the file (see Section 8.2.3.5).
FileSize	3	-	File size of the targeted file.
SW1SW2	2	9100h 91XXh	successful execution Refer to Table 74

Table 71. Response data parameters description - GetFileSettings - Targeting Value File.

Name	Length	Value	Description
FileType	1	02h	File Type of the targeted file.

Name	Length	Value	Description
FileOption	1	-	Options for the targeted file.
	Bit 7-2		RFU
	Bit 1-0		CommMode (see Table CommunicationModes)
AccessRights	2	-	Set of access conditions for the 1st set in the file (see Section 8.2.3.5).
LowerLimit	4	-	Lower Limit of the file (as defined at file creation).
UpperLimit	4	-	Upper Limit of the file (as defined at file creation).
LimitedCreditValue	4	-	Current maximum Limited Credit Value (all zero if disabled).
LimitedCreditEnabled	1	-	Encodes if LimitedCredit and Free GetValue is allowed for this file.
	Bit 7-2		RFU
	Bit 1		Free GetValue
		0b	No free access to GetValue
		1b	Free access to GetValue
	Bit 0		LimitedCredit support
		0b	LimitedCredit disabled
		1b	LimitedCredit enabled
SW1SW2	2	9100h 91XXh	successful execution Refer to Table 74

Table 72. Response data parameters description - GetFileSettings - Targeting CyclicRecord File.

Name	Length	Value	Description
FileType	1	-	File Type of the targeted file.
		04h	CyclicRecord
FileOption	1	-	Options for the targeted file.
	Bit 7-2		RFU
	Bit 1-0		CommMode (see Table CommunicationModes)
AccessRights	2	-	Set of access conditions for the 1st set in the file (see Section 8.2.3.5).
RecordSize	3	-	Size of a single record in the file (as defined at file creation).
MaxNoOfRecs	3	-	Maximum number of records within the file (as defined at file creation).
CurrNoOfRecs	3	-	Current number of records within the file.

Name	Length	Value	Description
SW1SW2	2	9100h 91XXh	successful execution Refer to Table 74

Table 73. Response data parameters description - GetFileSettings - Targeting TransactionMAC File.

Name	Length	Value	Description
FileType	1	05h	File Type of the targeted file.
FileOption	1	-	Options for the targeted file.
	Bit 7-6		RFU
	Bit 5		TMCLimit configuration
		0b	disabled
		1b	enabled
	Bit 4		Exclude unauthenticated operations from TMI
		0b	disabled
		1b	enabled
	Bit 3-2	00b	RFU
	Bit 1-0		CommMode (see Table CommunicationModes)
AccessRights	2	-	Access conditions of the targeted file (see Section 8.2.3.6)
TMKeyOption	1	-	Options for AppTransactionMACKey
	Bit 7-2		RFU
	Bit 1-0		KeyType of the AppTransactionMACKey
		10b	AES
		Other values	RFU
TMKeyVer	1	-	Key version of AppTransactionMACKey
TMCLimit	[2,4]	-	[Optional, present if Bit5 of FileOption set] TMCLimit
		00000001h .. FFFFFFFFh	[if AES mode] TMCLimit value (4 byte)
		0001h .. FFFFh	[if LRP mode] TMCLimit value (2 byte)
SW1SW2	2	9100h 91XXh	successful execution Refer to Table 74

Table 74. Return code description - GetFileSettings

Status	Value	Description
COMMAND_ABORTED	CAh	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	1Eh	Invalid secure messaging MAC (only).

Status	Value	Description
LENGTH_ERROR	7Eh	Command size not allowed.
PARAMETER_ERROR	9Eh	Parameter value not allowed.
PERMISSION_DENIED	9Dh	PICC level (MF) is selected.
FILE_NOT_FOUND	F0h	File with targeted FileNo does not exist for the targeted application.
MEMORY_ERROR	EEh	Failure when reading or writing to non-volatile memory.

11.7.3 GetFileIDs

The GetFileID command returns the File IDentifiers of all active files within the MIFARE DESFire Light application. Communication mode is CommMode.MAC.

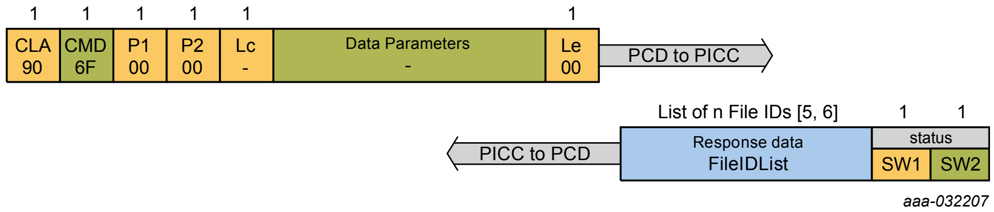


Figure 24. GetFileIDs command protocol

Table 75. Command parameters description - GetFileIDs

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	6Fh	Command code.
Command Data Parameters			
-	-	-	No data parameters

Table 76. Response data parameters description - GetFileIDs - OPERATION_OK

Name	Length	Value	Description
FileIDList	5, 6	-	List of n File IDs, if the TransactionMAC file is present, the 6 file IDs will be returned, otherwise 5
		00h..1Fh	for each byte
SW1SW2	2	9100h 91XXh	successful execution Refer to Table 77

Table 77. Return code description - GetFileIDs

Status	Value	Description
COMMAND_ABORTED	CAh	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	1Eh	Invalid secure messaging MAC (only).
LENGTH_ERROR	7Eh	Command size not allowed.
PERMISSION_DENIED	9Dh	PICC level (MF) is selected.
MEMORY_ERROR	EEh	Failure when reading or writing to non-volatile memory.

11.7.4 GetISOFileIDs

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

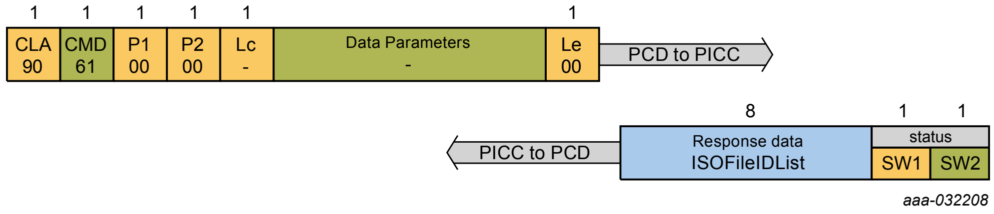


Figure 25. GetISOFileIDs command protocol

Table 78. Command parameters description - GetISOFileIDs

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	61h	Command code.
Command Data Parameters			
-	-	-	No data parameters

Table 79. Response data parameters description - GetISOFileIDs

Name	Length	Value	Description
ISOFileIDList	8	-	List of n ISO File IDs of the 3 StandardData files and the CyclicRecord file, 2 bytes per file.
SW1SW2	2	9100h 91XXh	successful execution Refer to Table 80

Table 80. Return code description - GetISOFileIDs

Status	Value	Description
COMMAND_ABORTED	CAh	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	1Eh	Invalid secure messaging MAC (only).
LENGTH_ERROR	7Eh	Command size not allowed.
PERMISSION_DENIED	9Dh	PICC level (MF) is selected.
MEMORY_ERROR	EEh	Failure when reading or writing to non-volatile memory.

11.7.5 DeleteTransactionMACFile

The TransactionMAC file can be deleted using the DeleteTransactionMACFile command. DeleteTransactionMACFile command needs an active authentication with the Application Master Key, AppMasterKey.

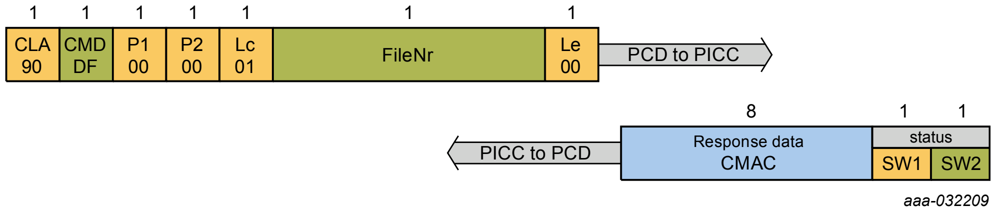


Figure 26. DeleteTransactionMACFile command protocol

Table 81. Command parameters description - DeleteTransactionMACFile

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	DFh	Command code.
FileNr	1	-	File number of the file to be deleted.
	Bit 7-5		RFU
	Bit 4-0		File number
		Full Range	
Command Data Parameters			
-	-	-	No data parameters

Table 82. Response data parameters description - DeleteTransactionMACFile

Name	Length	Value	Description
Response data	8	CMAC	8-byte CMAC
SW1SW2	2	9100h 91XXh	successful execution Refer to Table 83

Table 83. Return code description - DeleteTransactionMACFile

Status	Value	Description
COMMAND_ABORTED	CAh	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	1Eh	Invalid secure messaging MAC (only).
LENGTH_ERROR	7Eh	Command size not allowed.
PARAMETER_ERROR	9Eh	Parameter value not allowed.
PERMISSION_DENIED	9Dh	PICC level (MF) is selected.
		Targeted file is not TransactionMAC file.
FILE_NOT_FOUND	F0h	Targeted file does not exist in the targeted application.
AUTHENTICATION_ERROR	AEh	No active authentication with AppMasterKey.

Status	Value	Description
MEMORY_ERROR	EEh	Failure when reading or writing to non-volatile memory.

11.7.6 CreateTransactionMACFile

The CreateTransactionMACFile is to create the TransactionMAC file. As the TransactionMACKey, AppTransactionMACKey is updated at this command, it shall be executed only in a secure environment.

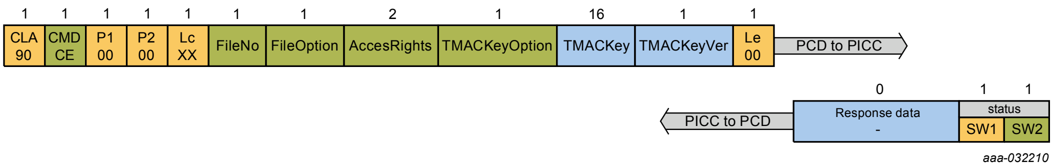


Figure 27. CreateTransactionMACFile command protocol

Table 84. Command parameters description - CreateTransactionMACFile

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	CEh	Command code.
FileNo	1	-	File number of the file to be created.
	Bit 7-5		RFU
	Bit 4-0	00h to 1Fh	File number
FileOption	1	-	Options for the targeted file.
	Bit 7-2		RFU
	Bit 1-0		CommMode (see Table CommunicationModes)
		X0b	CommMode.Plain
		01b	CommMode.MAC
		11b	CommMode.Full
AccessRights	2	-	Set of access conditions (see Table TransactionMACAccessRights)
TMACKeyOption	1	-	Options for AppTransactionMACKey
	Bit 7-2		RFU
	Bit 1-0		KeyType of the AppTransactionMACKey
		00b	RFU
		01b	RFU
		10b	AES
		11b	RFU
Command Data Parameters			
TMACKey	16	Full Range	Key value of AppTransactionMACKey
TMACKeyVer	1	Full Range	Key version of AppTransactionMACKey

Table 85. Response data parameters description - CreateTransactionMACFile

Name	Length	Value	Description
No response data			
SW1SW2	2	9100h 91XXh	successful execution Refer to Table 86

Table 86. Return code description - CreateTransactionMACFile

Status	Value	Description
COMMAND_ABORTED	CAh	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	1Eh	Invalid secure messaging MAC (only).
LENGTH_ERROR	7Eh	Command size not allowed.
PARAMETER_ERROR	9Eh	Parameter value not allowed.
		Targeted key for one of the access conditions in AccessRights does not exist.
PERMISSION_DENIED	9Dh	PICC level (MF) is selected.
		Selected application already holds a TransactionMAC file.
AUTHENTICATION_ERROR	AEh	No active authentication with AppMasterKey.
DUPLICATE_ERROR	DEh	File with the targeted FileNo already exists.
MEMORY_ERROR	EEh	Failure when reading or writing to non-volatile memory.

11.8 Data management commands

The MF2DL(H)x0 provides the following command set for Data Management functions.

11.8.1 ReadData

The ReadData command allows reading data from StandardData Files. The Read command requires a preceding authentication either with the key specified for Read or ReadWrite access, see the access rights section Section 8.2.3.5. Depending on the communication mode settings of the file secure messaging is applied, see Section 8.2.3.7.

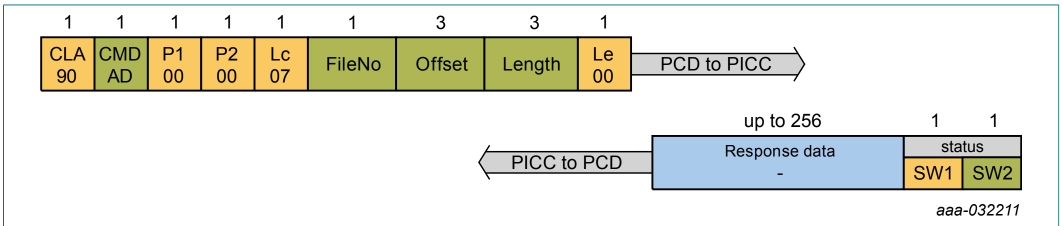


Figure 28. ReadData command protocol

Table 87. Command parameters description - ReadData

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	ADh	Command code.
FileNo	1	-	File number of the targeted file.
	Bit 7-5	000b	RFU
	Bit 4-0		File number
		Full Range	
Offset	3	000000h .. (FileSize - 1)	Starting position for the read operation.
Length	3	-	Number of bytes to be read.
		000000h	Read the entire StandardData file, starting from the position specified in the offset value. Note that the short length Le limits response data to 256 byte including secure messaging (if applicable).
		000001h .. (FileSize - Offset)	
Command Data Parameters			
-	-	-	No data parameters

Table 88. Response data parameters description - ReadData

Name	Length	Value	Description
Response data	up to 256 byte including secure messaging	Full Range	Data read from the file
SW1SW2	2	9100h 91XXh	successful execution Refer to Table 89

Table 89. Return code description - ReadData

Status	Value	Description
COMMAND_ABORTED	CAh	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	1Eh	Invalid secure messaging MAC (only)
LENGTH_ERROR	7Eh	Command size not allowed
PARAMETER_ERROR	9Eh	Parameter value not allowed
PERMISSION_DENIED	9Dh	PICC level (MF) is selected.
		Targeted file is not of StandardData or TransactionMAC.
		Read and ReadWrite of targeted StandardData file only have access conditions set to Fh.
		Read of targeted TransactionMAC file only have access conditions set to Fh.
		Targeted file is not a TransactionMAC file and sesTMC reached its maximal value.
		Targeted file is not a TransactionMAC file and TMCLimit was reached.
FILE_NOT_FOUND	F0h	Targeted file does not exist in the targeted application
AUTHENTICATION_ERROR	AEh	Read and ReadWrite of targeted StandardData file not granted while at least one of the access conditions is different from Fh.
		Read of targeted TransactionMAC file not granted while different from Fh.
BOUNDARY_ERROR	BEh	If targeting StandardData, attempt to read beyond the file boundary as set during creation
		If targeting TransactionMAC, attempt to read beyond the file boundary of 12 byte
MEMORY_ERROR	EEh	Failure when reading or writing to non-volatile memory

11.8.2 WriteData

The WriteData command allows writing data to StandardData Files. MF2DL(H)x0 supports tearing protection for data that is sent within one communication frame to the file. Consequently, when using ISO/IEC 14443-4 chaining to write to a StandardData file, each frame itself is tearing protected but an interrupted chaining can lead to inconsistent files. Using single-frame WriteData commands instead of using the chaining can enable better control of the overall write process.

Depending on the communication mode settings of the Data file, data needs to be sent with either CommMode.Plain, CommMode.MAC or CommMode.Full. All cryptographic operations are done in CBC mode. In case of CommMode.MAC or CommMode.Full, the validity of data is verified by the PICC by checking the MAC. If the verification fails, the PICC stops further user memory programming and returns an Integrity Error to the PCD. As a consequence of the Integrity Error, any transaction, which might have begun, is automatically aborted. This can lead to the same situation as described above for an interrupted WriteData using chained communication.

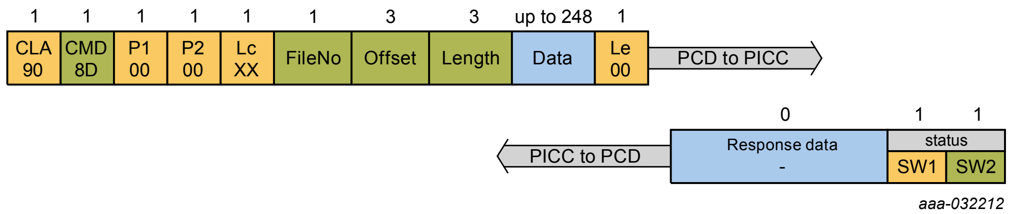


Figure 29. WriteData command protocol

Table 90. Command parameters description - WriteData

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	8Dh	Command code.
FileNo	1	-	File number of the targeted file.
	Bit 7-5	000b	RFU
	Bit 4-0		File number
		Full range	
Offset	3	000000h .. (FileSize - 1)	Starting position for the write operation.
Length	3	000001h .. (FileSize - Offset)	Number of bytes to be written.
Command Data Parameters			
Data	up to 248 byte including secure messaging	Full range	Data to be written.

Table 91. Response data parameters description - WriteData

Name	Length	Value	Description
No response data parameters defined for this command			
Response data	0	-	No response data
SW1SW2	2	9100h 91XXh	successful execution Refer to Table 92

Table 92. Return code description - WriteData

Status	Value	Description
COMMAND_ABORTED	CAh	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	1Eh	Invalid secure messaging MAC or encryption padding.
LENGTH_ERROR	7Eh	Command size not allowed.
PARAMETER_ERROR	9Eh	Parameter value not allowed.
PERMISSION_DENIED	9Dh	PICC level (MF) is selected.
		Targeted file is not of StandardData.
		Write and ReadWrite of targeted file only have access conditions set to Fh.
		Targeting a StandardData file with a chained command in MAC or Full while this is not allowed.
		sesTMC reached its maximal value.
		TMCLimit was reached.
FILE_NOT_FOUND	F0h	Targeted file does not exist in the targeted application.
AUTHENTICATION_ERROR	AEh	Write and ReadWrite of targeted file not granted while at least one of the access conditions is different from Fh.
BOUNDARY_ERROR	BEh	Attempt to write beyond the file boundary as set during creation.
MEMORY_ERROR	EEh	Failure when reading or writing to non-volatile memory.

11.8.3 GetValue

The GetValue command allows reading the currently stored value from the Value file. The value is always represented LSB first. The GetValue command requires a preceding authentication with the key specified for Read, Write or ReadWrite access. Secure messaging applies to the command according to the file communication mode setting. If free GetValue is configured for the Value file, the GetValue command does not need any prior authentication and CommMode.Plain is applied. After updating a value file value but before issuing the CommitTransaction command, the GetValue command will always retrieve the old, unchanged value which is still the valid one.

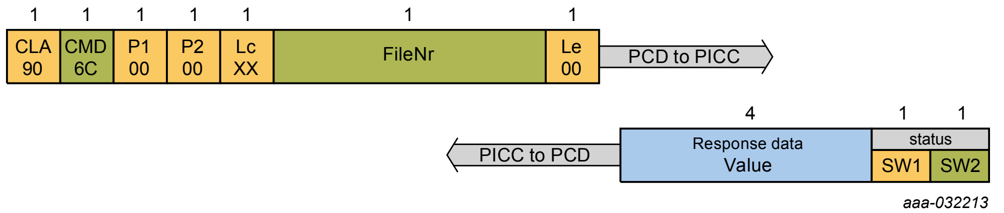


Figure 30. GetValue command protocol

Table 93. Command parameters description - GetValue

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	6Ch	Command code.
Data	1	FileNr	File number of the targeted file.
	Bit 7-5	000b	RFU
	Bit 4-0	00h to 1Fh	File number which is targeted
Command Data Parameters			
-	-	-	No data parameters

Table 94. Response data parameters description - GetValue

Name	Length	Value	Description
Response data	4	Value	Current Value.
SW1SW2	2	9100h 91XXh	successful execution Refer to Table 95

Table 95. Return code description - GetValue

Status	Value	Description
COMMAND_ABORTED	CAh	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	1Eh	Invalid secure messaging MAC (only).
LENGTH_ERROR	7Eh	Command size not allowed.
PARAMETER_ERROR	9Eh	Parameter value not allowed.
PERMISSION_DENIED	9Dh	PICC level (MF) is selected.

Status	Value	Description
		Targeted file is not of Value.
		Read, Write and ReadWrite of targeted file only have access conditions set to Fh.
		sesTMC reached its maximal value.
		TMCLimit was reached.
FILE_NOT_FOUND	F0h	Targeted file does not exist in the targeted application.
AUTHENTICATION_ERROR	AEh	Read, Write and ReadWrite of targeted file not granted while at least one of the access conditions is different from Fh and no free access was granted at file creation.
MEMORY_ERROR	EEh	Failure when reading or writing to non-volatile memory.

11.8.4 Credit

The Credit command allows increasing a value stored in a Value File. 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. The value is always represented LSB first. It is necessary to validate the updated value with a CommitTransaction command. An AbortTransaction command invalidates all changes. 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.

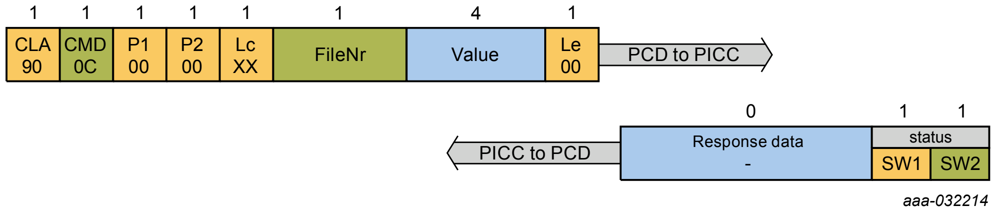


Figure 31. Credit command protocol

Table 96. Command parameters description - Credit

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0Ch	Command code.
FileNr	1	-	File number of the targeted file.
	Bit 7-5	000b	RFU
	Bit 4-0	00h to 1Fh	FileNr
Command Data Parameters			
Value	4	00000000h .. 7FFFFFFFh	Value to be credited. LSB first. should be positive.

Table 97. Response data parameters description - Credit

Name	Length	Value	Description
Response data	0	-	No response data
SW1SW2	2	9100h 91XXh	successful execution Refer to Table 98

Table 98. Return code description - Credit

Status	Value	Description
COMMAND_ABORTED	CAh	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	1Eh	Invalid secure messaging MAC or encryption padding.
LENGTH_ERROR	7Eh	Command size not allowed.
PARAMETER_ERROR	9Eh	Parameter value not allowed.

Status	Value	Description
PERMISSION_DENIED	9Dh	PICC level (MF) is selected.
		Targeted file is not of Value.
		ReadWrite of targeted file only have access conditions set to Fh
		sesTMC reached its maximal value.
		TMCLimit was reached.
FILE_NOT_FOUND	F0h	Targeted file does not exist in the targeted application.
AUTHENTICATION_ERROR	AEh	ReadWrite of targeted file not granted while at least one of the access conditions is different from Fh.
BOUNDARY_ERROR	BEh	Data added to the current value of the value file is bigger than the value file UpperLimit or create an integer arithmetic overflow.
MEMORY_ERROR	EEh	Failure when reading or writing to non-volatile memory.

11.8.5 Debit

The Debit command allows decreasing a value stored in a Value File. The current value will be subtracted by the 4-byte signed integer value. Only positive values are allowed for the Debit command. The value is always represented LSB first. It is necessary to validate the updated value with a CommitTransaction command. An AbortTransaction command invalidates all changes. The value modifications of Credit, Debit and LimitedCredit commands are cumulated until a CommitTransaction command is issued. 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 cannot rebook more values than a debiting transaction deducted before.

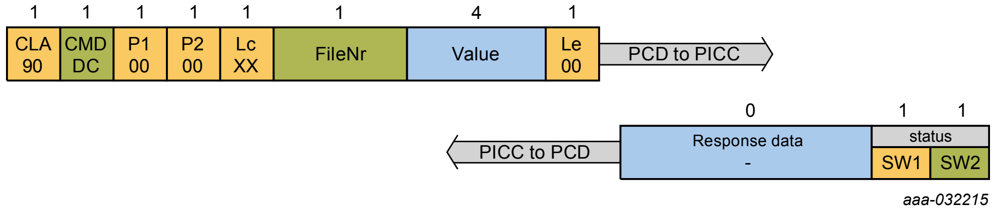


Figure 32. Debit command protocol

Table 99. Command parameters description - Debit

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	DCh	Command code.
FileNr	1	-	File number of the targeted file.
	Bit 7-5	000b	RFU
	Bit 4-0	00h to 1Fh	File number
Command Data Parameters			
Value	4	00000000h .. 7FFFFFFFh	Value to be debited. LSB first. Should be positive.

Table 100. Response data parameters description - Debit

Name	Length	Value	Description
Response data	0	-	No response data
SW1SW2	2	9100h 91XXh	successful execution Refer to Table 101

Table 101. Return code description - Debit

Status	Value	Description
COMMAND_ABORTED	CAh	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	1Eh	Invalid secure messaging MAC or encryption padding.
LENGTH_ERROR	7Eh	Command size not allowed.

Status	Value	Description
PARAMETER_ERROR	9Eh	Parameter value not allowed.
PERMISSION_DENIED	9Dh	PICC level (MF) is selected.
		Targeted file is not of Value.
		Read, Write and ReadWrite of targeted file only have access conditions set to Fh
		sesTMC reached its maximal value.
		TMCLimit was reached.
FILE_NOT_FOUND	F0h	Targeted file does not exist in the targeted application.
AUTHENTICATION_ERROR	AEh	Read, Write and ReadWrite of targeted file not granted while at least one of the access conditions is different from Fh.
BOUNDARY_ERROR	BEh	Data subtracted to the current value of the value file is smaller than the value file LowerLimit or create an integer arithmetic underflow.
MEMORY_ERROR	EEh	Failure when reading or writing to non-volatile memory.

11.8.6 LimitedCredit

The LimitedCredit command allows a limited increase of a value stored in a Value File without having full Credit permissions to the file. This feature can be enabled or disabled through the SetConfiguration command. It is necessary to validate the updated value with a CommitTransaction command. An AbortTransaction command invalidates all changes. 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 AppKey specified for “Write” or "Read&Write" access. 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 rebooked. Therefore, the LimitedCredit command can only be used once after a Debit transaction. Secure messaging applies to this command according to file communication mode setting.

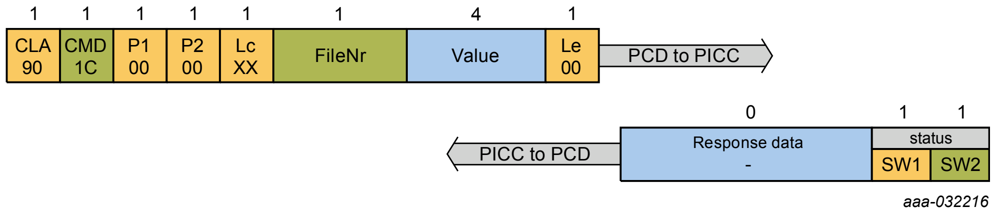


Figure 33. LimitedCredit command protocol

Table 102. Command parameters description - LimitedCredit

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	1Ch	Command code.
FileNr	1	-	File number of the targeted file.
	Bit 7-5	000b	RFU
	Bit 4-0	00h to 1Fh	File number
Command Data Parameters			
Value	4	00000000h .. 7FFFFFFFh	Value to be Ltd. credited. LSB first. Should be positive.

Table 103. Response data parameters description - LimitedCredit

Name	Length	Value	Description
Response Data	0	-	No response data
SW1SW2	2	9100h 91XXh	successful execution Refer to Table 104

Table 104. Return code description - LimitedCredit

Status	Value	Description
COMMAND_ABORTED	CAh	Chained command or multiple pass command ongoing.

Status	Value	Description
INTEGRITY_ERROR	1Eh	Invalid secure messaging MAC or encryption padding.
LENGTH_ERROR	7Eh	Command size not allowed.
PARAMETER_ERROR	9Eh	Parameter value not allowed.
PERMISSION_DENIED	9Dh	PICC level (MF) is selected.
		Targeted file is not of Value.
		Write and ReadWrite of targeted file only have access conditions set to Fh
		Limited credit not enabled for file with FileNo.
		sesTMC reached its maximal value.
		TMCLimit was reached.
FILE_NOT_FOUND	F0h	Targeted file does not exist in the targeted application.
AUTHENTICATION_ERROR	AEh	Write and ReadWrite of targeted file not granted while at least one of the access conditions is different from Fh.
BOUNDARY_ERROR	BEh	Data is exceeding the limited credit limit.
		Already successfully executed a LimitedCredit command during the current ongoing transaction.
		Already successfully executed and committed a L imitedCredit command since the latest successful transaction containing at least one Debit
MEMORY_ERROR	EEh	Failure when reading or writing to non-volatile memory.

11.8.7 ReadRecords

The ReadRecords command allows reading out a set of complete records from the CyclicRecord file. A ReadRecords command on an empty record file results in an error. The ReadRecords command requires a preceding authentication either with the AppKey specified for "Read" or "Read&Write" access, Section 8.2.3.5. Depending on the communication mode setting, Section 8.2.3.7, and authentication linked to the file, data will be sent by the PICC with either CommMode.Plain, CommMode.MAC or CommMode.Full.

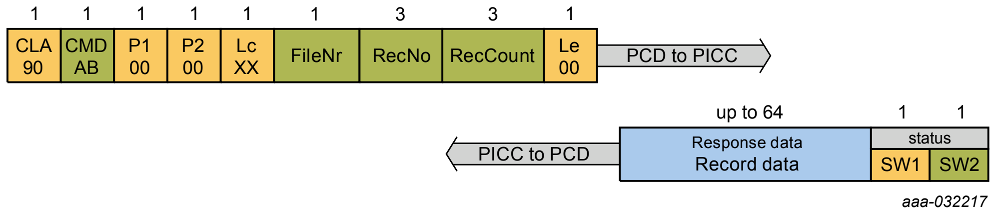


Figure 34. ReadRecords command protocol

Table 105. Command parameters description - ReadRecords

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	ABh	Command code.
FileNr	1	-	File number of the targeted file.
	Bit 7-5	000b	RFU
	Bit 4-0	00h to 1Fh	File number
RecNo	3	-	Record number. Records are numbered starting with the latest record written.
		000000h .. (ExistingRecCount - 1)	000000h meaning the latest record written. The number is limited by the number of existing records (ExistingRecCount).
RecCount	3	-	Number of records to be read. Records are always transmitted by the PICC in chronological order (= starting with the oldest, which is ReadRecords.RecCount-1 before the one addressed by the given ReadRecords.RecNo).
		000000h	Read all records, from the oldest record up to and including the newest record (given by ReadRecords.RecNo) are read
		000001h .. (ExistingRecCount - RecNo)	The number is limited by the number of existing records (ExistingRecCount) and the offset given by ReadRecords.RecNo.
Command Data Parameters			
-	-	-	No data parameters

Table 106. Response data parameters description - ReadRecords

Name	Length	Value	Description
Response data	Up to 64	Full range	Record data [Number of record read x 16 bytes]
SW1SW2	2	9100h 91XXh	successful execution Refer to Table 107

Table 107. Return code description - ReadRecords

Status	Value	Description
COMMAND_ABORTED	CAh	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	1Eh	Invalid secure messaging MAC (only).
LENGTH_ERROR	7Eh	Command size not allowed.
PARAMETER_ERROR	9Eh	Parameter value not allowed.
PERMISSION_DENIED	9Dh	PICC level (MF) is selected.
		Targeted file is not of CyclicRecord.
		Read and ReadWrite of targeted file only have access conditions set to Fh
		sesTMC reached its maximal value.
		TMCLimit was reached.
FILE_NOT_FOUND	F0h	Targeted file does not exist in the targeted application.
AUTHENTICATION_ERROR	AEh	Read and ReadWrite of targeted file not granted while at least one of the access conditions is different from Fh.
BOUNDARY_ERROR	BEh	Attempt to read more records than have been written so far.
MEMORY_ERROR	EEh	Failure when reading or writing to non-volatile memory.

11.8.8 WriteRecord

The WriteRecord command allows writing data to a record in the Cyclic File. The WriteRecord command appends one record at the end of the file, it erases and overwrites the oldest record in case if it is already full. The entire new record is cleared before data is written to it. If no CommitTransaction command 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 invalidates all changes. 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 mode settings, linked to the file, data needs to be sent by the PCD with either CommMode.Plain, CommMode.MAC or CommMode.Full.

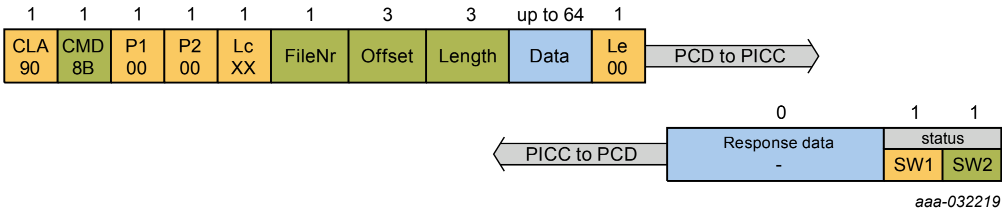


Figure 35. WriteRecord command protocol

Table 108. Command parameters description - WriteRecord

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	8Bh	Command code.
FileNr	1	-	File number of the targeted file.
	Bit 7-5	000b	RFU
	Bit 4-0	00h to 1Fh	File number
Offset	3	000000h .. 00000Fh	Starting position for the write operation within the targeted record.
Length	3	000001h .. (000010h - Offset)	Number of bytes to be written.
Command Data Parameters			
Data	up to 64	Full range	Data to be written.

Table 109. Response data parameters description - WriteRecord

Name	Length	Value	Description
Response data	0	-	No response data
SW1SW2	2	9100h 91XXh	successful execution Refer to Table 110

Table 110. Return code description - WriteRecord

Status	Value	Description
COMMAND_ABORTED	CAh	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	1Eh	Invalid secure messaging MAC or encryption padding.
LENGTH_ERROR	7Eh	Command size not allowed.
PARAMETER_ERROR	9Eh	Parameter value not allowed.
PERMISSION_DENIED	9Dh	PICC level (MF) is selected.
		Targeted file is not of CyclicRecord.
		Write and ReadWrite of targeted file only have access conditions set to Fh
		A UpdateRecord targeting the currently targeted file has already been performed within the current transaction. A ClearRecordFile targeting the currently targeted file has already been performed within the current transaction.
		sesTMC reached its maximal value.
		TMCLimit was reached.
FILE_NOT_FOUND	F0h	Targeted file does not exist in the targeted application.
AUTHENTICATION_ERROR	AEh	Write and ReadWrite of targeted file not granted while at least one of the access conditions is different from Fh.
BOUNDARY_ERROR	BEh	Offset + 1 is bigger than 16 bytes.
		Offset + Length + 1 is bigger than 16 bytes.
MEMORY_ERROR	EEh	Failure when reading or writing to non-volatile memory.

11.8.9 UpdateRecord

The UpdateRecord command updates data of an existing record in the Cyclic Record File.

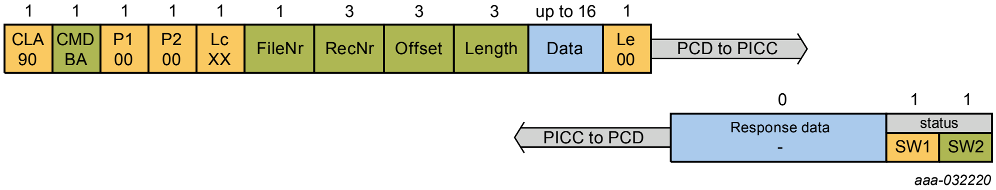


Figure 36. UpdateRecord command protocol

Table 111. Command parameters description - UpdateRecord

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	BAh	Command code.
FileNr	1	-	File number of the targeted file.
	Bit 7-5	000b	RFU
	Bit 4-0	00h to 1Fh	File number
RecNr	3	-	Record number. Records are numbered starting with the latest record written.
		000000h .. (ExistingRecCount - 1)	000000h meaning the latest record written. The number is limited by the number of existing records (ExistingRecCount).
Offset	3	000000h .. (00000Fh)	Starting position for the update operation within the targeted record.
Length	3	000001h .. (000010h - Offset)	Number of bytes to be updated.
Command Data Parameters			
Data	up to 16	Full range	Data to be written in the targeted record

Table 112. Response data parameters description - UpdateRecord

Name	Length	Value	Description
Response data	0	-	No response data
SW1SW2	2	9100h 91XXh	successful execution Refer to Table 113

Table 113. Return code description - UpdateRecord

Status	Value	Description
COMMAND_ABORTED	CAh	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	1Eh	Invalid secure messaging MAC or encryption padding.
LENGTH_ERROR	7Eh	Command size not allowed.

Status	Value	Description
PARAMETER_ERROR	9Eh	Parameter value not allowed.
PERMISSION_DENIED	9Dh	PICC level (MF) is selected.
		Targeted file is not of CyclicRecord.
		ReadWrite of targeted file only have access conditions set to Fh
		A ClearRecordFile targeting the currently targeted file has already been performed within the current transaction.
		A WriteRecord targeting the currently targeted file has already been performed within the current transaction.
		An UpdateRecord targeting a record different from the currently targeted record has already been performed within the current transaction.
		sesTMC reached its maximal value.
		TMCLimit was reached.
FILE_NOT_FOUND	F0h	Targeted file does not exist in the targeted application.
AUTHENTICATION_ERROR	AEh	ReadWrite of targeted file not granted while at least one of the access conditions is different from Fh.
BOUNDARY_ERROR	BEh	RecNo is strictly bigger than the number of existing record in the file.
		Offset + 1 is bigger than 16 bytes.
		Offset + Length + 1 is bigger than 16 bytes.
MEMORY_ERROR	EEh	Failure when reading or writing to non-volatile memory.

11.8.10 ClearRecordFile

The command clears the Cyclic Record File. After successful execution of the ClearRecordFile command the Cyclic Record File becomes empty. After executing the ClearRecordFile command but before CommitTransaction, all subsequent WriteRecord commands will fail. The ReadRecords command returns 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) invalidates the clearance.

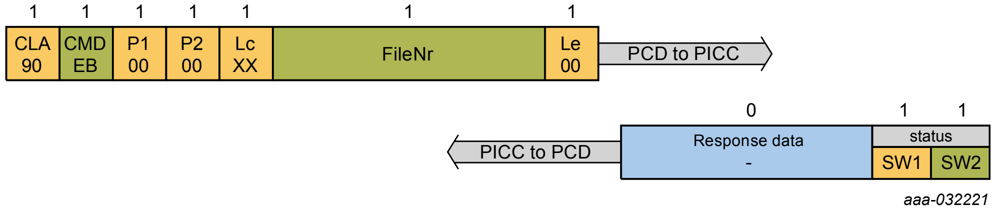


Figure 37. ClearRecordFile command protocol

Table 114. Command parameters description - ClearRecordFile

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	EBh	Command code.
FileNr	1	-	File number of the targeted file.
	Bit 7-5		RFU
	Bit 4-0	00h to 1Fh	File number
Command Data Parameters			
-	-	-	No data parameters

Table 115. Response data parameters description - ClearRecordFile

Name	Length	Value	Description
Response data	0	-	No response data
SW1SW2	2	9100h 91XXh	successful execution Refer to Table 116

Table 116. Return code description - ClearRecordFile

Status	Value	Description
COMMAND_ABORTED	CAh	
INTEGRITY_ERROR	1Eh	Invalid secure messaging MAC.
LENGTH_ERROR	7Eh	Command size not allowed.
PARAMETER_ERROR	9Eh	Parameter value not allowed.
PERMISSION_DENIED	9Dh	PICC level (MF) is selected.
		Targeted file is not of CyclicRecord.

Status	Value	Description
		ReadWrite of targeted file only have access conditions set to Fh
		sesTMC reached its maximal value.
		TMCLimit was reached.
FILE_NOT_FOUND	F0h	Targeted file does not exist in the targeted application.
AUTHENTICATION_ERROR	AEh	ReadWrite of targeted file not granted while at least one of the access conditions is different from Fh.
MEMORY_ERROR	EEh	Failure when reading or writing to non-volatile memory.

11.9 Transaction management commands

The MF2DL(H)x0 provides the following command set for Transaction Management functions.

11.9.1 CommitTransaction

The CommitTransaction command is required to commit the changes made in the backed-up files, e.g., Value and the Record File. The CommitTransaction is typically the last command of a transaction before the ISO/IEC 14443-4 deselect command.

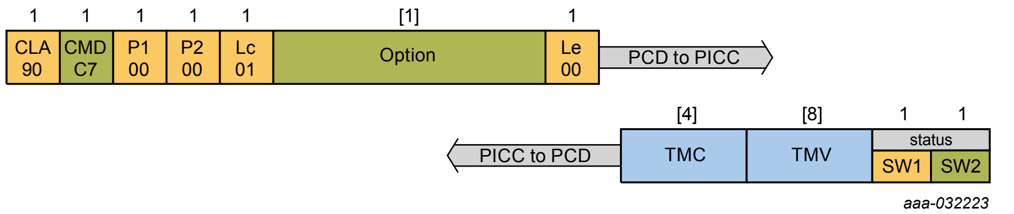


Figure 38. CommitTransaction command protocol

Table 117. Command parameters description - CommitTransaction

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	C7h	Command code.
Option	[1]	-	[Optional] Option parameter, If not used then Lc should be absent
	Bit 7-1	0000000b	RFU
	Bit 0		Calculated Transaction MAC requested on response
		0b	No TMC and TMV returned
		1b	TMC and TMV returned
Command Data Parameters			
-	-	-	No data parameters

Table 118. Response data parameters description - CommitTransaction

Name	Length	Value	Description
TMC	[4]	00000001h .. FFFFFFFFh	[Optional, only present if Bit0 of Option is set] Transaction MAC Counter (TMC)
TMV	[8]	Full Range	[Optional, only present if Bit0 of Option is set] Transaction MAC Value (TMV)
SW1SW2	2	9100h 91XXh	successful execution Refer to Table 119

Table 119. Return code description - CommitTransaction

Status	Value	Description
COMMAND_ABORTED	CAh	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	1Eh	Invalid secure messaging MAC (only).
LENGTH_ERROR	7Eh	Command size not allowed.
PARAMETER_ERROR	9Eh	Parameter value not allowed.
		TMC and TMV are requested while no Transaction MAC calculation is ongoing.
PERMISSION_DENIED	9Dh	PICC level (MF) is selected.
		CommitReaderID is required but was not performed, i.e. TMRICur is the empty string.
NO_CHANGES	0Ch	No backup files (Value or CyclicRecord) updated and (if Transaction MAC active) TMI is the empty string.
MEMORY_ERROR	EEh	Failure when reading or writing to non-volatile memory.

11.9.2 AbortTransaction

The AbortTransaction command allows invalidating all previous write access on the Value File and the Record File within one application. This command is useful to cancel a transaction without losing the current authenticated status. Hence, reducing the need for re-selection and re-authentication of the application. The AbortTransaction command invalidates all write access to files with integrated backup mechanisms without changing the authentication status.

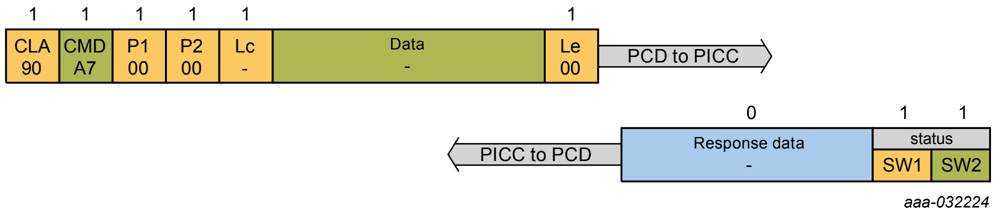


Figure 39. AbortTransaction command protocol

Table 120. Command parameters description - AbortTransaction

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	A7h	Command code.
Command Data Parameters			
-	-	-	No data parameters

Table 121. Response data parameters description - AbortTransaction

Name	Length	Value	Description
Response data	0	-	No response data
SW1SW2	2	9100h 91XXh	successful execution Refer to Table 122

Table 122. Return code description - AbortTransaction

Status	Value	Description
COMMAND_ABORTED	CAh	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	1Eh	Invalid secure messaging MAC (only).
LENGTH_ERROR	7Eh	Command size not allowed.
PERMISSION_DENIED	9Dh	PICC level (MF) is selected.
NO_CHANGES	0Ch	No backup files (Value or CyclicRecord) updated and (if Transaction MAC active) TMI is the empty string.
MEMORY_ERROR	EEh	Failure when reading or writing to non-volatile memory.

11.9.3 CommitReaderID

Using the CommitReaderID command any 16-byte data can be sent to the card, to be part of the TransactionMAC calculation. Using terminal and or transaction tracking data allow a backend to identify any inconsistency in use of the card.

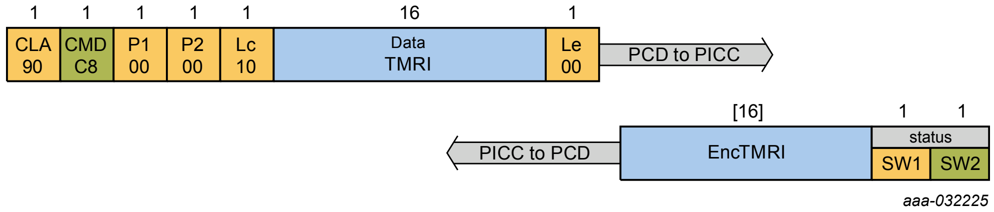


Figure 40. CommitReaderID command protocol

Table 123. Command parameters description - CommitReaderID

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	C8h	Command code.
Command Data Parameters			
TMRI	16	Full Range	Transaction MAC ReaderID, can be any arbitrary data

Table 124. Response data parameters description - CommitReaderID

Name	Length	Value	Description
EncTMRI	[16]	Full Range	Optional, present if authenticated. Encrypted Transaction MAC ReaderID of the latest successful transaction. E_{TM}*(SesTMENCKey; TMRIPrev)*
SW1SW2	2	9100h 91XXh	successful execution Refer to Table 125

Table 125. Return code description - CommitReaderID

Status	Value	Description
COMMAND_ABORTED	CAh	Chained command or multiple pass command ongoing.
INTEGRITY_ERROR	1Eh	Invalid secure messaging MAC.
LENGTH_ERROR	7Eh	Command size not allowed.
PERMISSION_DENIED	9Dh	PICC level (MF) is selected.
		The application does not hold a TransactionMAC file.
		No ReaderID required according to TransactionMAC file's access rights.
		ReaderID has already been committed during the ongoing transaction, i.e. TMRICur is not the empty string anymore.
		sesTMC reached its maximal value.

Status	Value	Description
		TMCLimit was reached.
AUTHENTICATION_ERROR	AEh	No active authentication with AppCommitReaderIDKey.
MEMORY_ERROR	EEh	Failure when reading or writing to non-volatile memory.

11.10 Inter-industry standard commands

MF2DL(H)x0 provides the following ISO/IEC 7816-4 wrapped commands.

11.10.1 ISOSelectFile

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

If P1 is set to 00h, 01h or 02h, selection is done by a 2-byte ISO file identifier. For PICC level / MF selection, 3F00h or empty data has to be used. For Deticated application File (DF) and Elementary File (EF) selection, data holds the 2-byte ISO/IEC 7816-4 file identifier.

If P1 is set to 04h, selection is done by Deticated File (DF) name which can be up to 16 bytes. The registered ISO DF name of MIFARE DESFire is D2760000850100h. When selecting this DF name, the PICC level (or MF) is selected. For selecting the application immediately, the default ISO/IEC 7816-4 DF name A00000039656434103F015400000000Bh or the customized DF name personalized with SetConfiguration can be used.

P2 indicates whether or not File Control Information (FCI) is to be returned in case of application selection. The number of bytes requested by Le up to the complete file data will be returned in plain. There is no specific FCI template format checked, i.e. the data stored in the file will be sent back as is. In case of PICC level or file selection, FCI data is never returned.

When trying to select the application (DF) and the TMCLimit was reached, see Section 10.3.2.2, the application will still be selected, but an ISO6283 error is returned to indicate limited functionality: no data management commands, see Section 11.8 can be executed, except ReadData targeting the TransactionMAC file. Also CommitReaderID will be rejected.

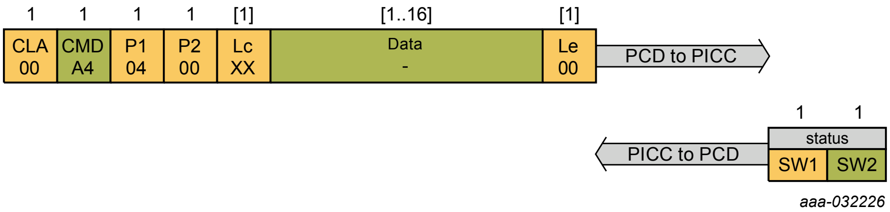


Figure 41. ISOSelectFile command protocol

Table 126. Command parameters description - ISOSelectFile

Name	Length	Value	Description
CLA	1	00h	
INS	1	A4h	
P1	1	-	Selection Control
		00h	Select MF, DF or EF, by file identifier
		01h	Select child DF
		02h	Select EF under the current DF, by file identifier
		03h	Select parent DF of the current DF
		04h	Select by DF name, see [3]

Name	Length	Value	Description
P2	1	-	Option
		00h	Return FCI template: data stored in the file with ID 1Fh should be returned
		0Ch	No response data: no FCI should be returned
Lc	[1]	00h .. 10h	Length of subsequent data field
Data	[1..16]	-	Reference
		Empty	[if P1 == 00h OR P1 == 03h] Select MF
		Full range	[if P1 == 00h OR P1 == 01h OR P1== 02h] Select with the given file identifier
		Full range	[if P1 == 04h] Select DF with the given DF name
Le	[1]	Full range	Empty or length of expected response

Table 127. Response data parameters description - ISOSelectFile

Name	Length	Value	Description
Data	[X]	-	[Optional] FCI stored in file ID 1Fh of the DF.
SW1SW2	2	9000h 6283h XXXXh	successful execution application selected with limited functionality Refer to Table 128

Table 128. Return code description - ISOSelectFile

SW1|SW2	Value	Description
ISO6283	6283h	TMCLimit reached, see Section 10.3.2.2. application selected with limited functionality
		sesTMC reached maximal value. application selected with limited functionality.
ISO6700	6700h	Wrong or inconsistent APDU length.
ISO6985	6985h	Wrapped chained command or multiple pass command ongoing.
ISO6A82	6A82h	Application or file not found, currently selected application remains selected.
ISO6A86	6A86h	Wrong parameter P1 and/or P2
ISO6A87	6A87h	Wrong parameter Lc inconsistent with P1-P2
ISO6E00	6E00h	Wrong CLA

11.10.2 ISOReadBinary

The ISOReadBinary is a standard ISO/IEC 7816-4 command. It can be used to read data from the Standard Data File. This command does not support any secure messaging, it is always in plain. For executing ISOReadBinary command either "Read" or "Read&Write", access right must be set to free access rights.

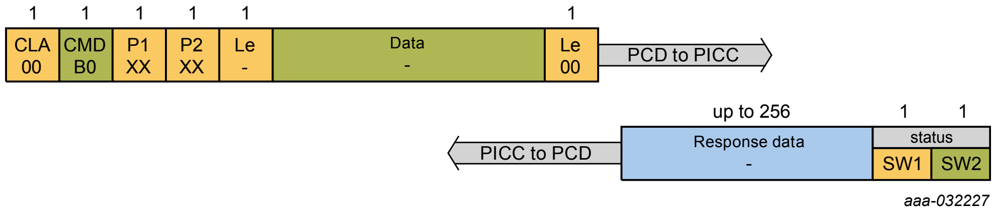


Figure 42. ISOReadBinary command protocol

Table 129. Command parameters description - ISOReadBinary

Name	Length	Value	Description
CLA	1	00h	
INS	1	B0h	
P1	1		ShortFile ID/Offset
	Bit 7		Encoding
		1b	P1[Bit 6..5] are RFU. P1[Bit 4..0] encode a short ISO FileID. P2[Bit 7..0] encode an offset from zero to 255.
		0b	P1 - P2 (15 bits) encode an offset from zero to 32767.
	Bit 6-5	00b	[if P1[7] == 1b] RFU
	Bit 4-0		[if P1[7] == 1b] short ISO FileID
		00h	Targeting currently selected file.
		01h .. 1Eh	Targeting and selecting file referenced by the given short ISO FileID.
		1Fh	RFU
	Bit 6-0	(see P2)	[if P1[7] == 0b] Most significant bits of Offset
P2	1	000000h .. (FileSize - 1)	Offset (see above)
Le	1	-	The number of bytes to be read from the file. The length of a secure messaging MAC (depending on communication mode settings) should be included in this value.
		00h	Read the entire StandardData file, starting from the position specified in the offset value. Note that the short length Le limits response data to 256 byte.
		01h .. FFh	If bigger than (FileSize - Offset), after subtracting MAC length if MAC is to be returned, the entire StandardData file starting from the offset position is returned.

Name	Length	Value	Description
		Full range	

Table 130. Response data parameters description - ISOReadBinary

Name	Length	Value	Description
Data	up to 256	-	Data read.
SW1SW2	2	9000h XXXXh	successful execution Refer to Table 131

Table 131. Return code description - ISOReadBinary

SW1|SW2	Value	Description
ISO6581	6581h	Memory failure
ISO6700	6700h	Wrong or inconsistent APDU length.
ISO6982	6982h	Security status not satisfied: no access allowed as Read and ReadWrite is set to Fh.
		Security status not satisfied: only free read with Read or ReadWrite equal to Eh is allowed.
		Security status not satisfied: AuthenticatedEV2 not allowed.
		Security status not satisfied: AuthenticatedLRP not allowed.
ISO6985	6985h	Wrapped chained command or multiple pass command ongoing. No file selected. Targeted file is not of StandardData. Application of targeted file holds a TransactionMAC file.
ISO6A82	6A82h	File not found
ISO6A86	6A86h	Wrong parameter P1 and/or P2
ISO6E00	6E00h	Wrong CLA

11.10.3 ISOUpdateBinary

The ISOUpdateBinary command is implemented in compliance with ISO/IEC 7816-4, the command is only possible with CommMode.Plain for a Standard Data File. MF2DL(H)x0 supports tearing protection for data that is sent within one communication frame to the file. Consequently, when using ISO/IEC 14443-4 chaining to write to a Standard Data File, each frame itself is tearing protected but an interrupted chaining can lead to inconsistent files. Using single-frame ISOUpdateBinary commands instead of using the chaining can enable better control of the overall write process.

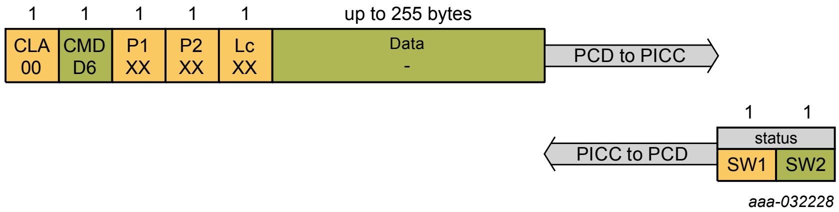


Figure 43. ISOUpdateBinary command protocol

Table 132. Command parameters description - ISOUpdateBinary

Name	Length	Value	Description
CLA	1	00h	
INS	1	D6h	
P1	1		ShortFile ID/Offset
	Bit 7		RFU
		1b	P1[Bit 6..5] are RFU. P1[Bit 4..0] encode a short ISO FileID. P2[Bit 7..0] encode an offset from zero to 255.
		0b	P1 - P2 (15 bits) encode an offset from zero to 32767.
	Bit 6-5	00b	[if P1[7] == 1b] RFU
	Bit 4-0		[if P1[7] == 1b] short ISO FileID
		00h	Targeting currently selected file.
		01h .. 1Eh	Targeting and selecting file referenced by the given short ISO FileID.
		1Fh	RFU
	Bit 6-0	(see P2)	[if P1[7] == 0b] Most significant bits of Offset
P2	1	000000h .. (FileSize - 1)	Offset (see above)
Lc	1	01h .. (FileSize - Offset)	Length of subsequent data field
Data	up to 255 byte	Full range	Data to be written

Table 133. Response data parameters description - ISOUpdateBinary

Name	Length	Value	Description
No response data parameters defined for this command			

Name	Length	Value	Description
SW1SW2	2	9000h XXXXh	successful execution Refer to Table 134

Table 134. Return code description - ISOUpdateBinary

SW1|SW2	Value	Description
ISO6581	6581h	Memory failure
ISO6700	6700h	Wrong or inconsistent APDU length.
ISO6982	6982h	Security status not satisfied: only free write with Write or ReadWrite equal to Eh is allowed. Security status not satisfied: AuthenticatedEV2 not allowed. Security status not satisfied: AuthenticatedLRP not allowed.
ISO6985	6985h	Wrapped chained command or multiple pass command ongoing. No file selected. Targeted file is not of StandardData. Attempt to write beyond the file boundary as set during creation. Application of targeted file holds a TransactionMAC file.
ISO6A82	6A82h	File not found
ISO6A86	6A86h	Wrong parameter P1 and/or P2
ISO6E00	6E00h	Wrong CLA

11.11 Originality check commands

The originality check allows verification of the genuineness of MIFARE DESFire Light (MF2DL(H)x0). Two ways are offered to check the originality of the PICC: the first is based on a symmetric authentication, the second works on the verification of an asymmetric signature retrieved from the card.

The authentication procedure for AES keys can be used to authenticate to one of the four OriginalityKey and check whether the PICC is a genuine NXP product. MF2DL(H)x0 supports targeting the OriginalityKey with the LRP authentication using AES. For details on the authentication command, see Section 9.2. The following variants can be used:

- AuthenticateLRPFirst, see Section 9.2.5
- AuthenticateLRPNonFirst, see Section 9.2.6

The asymmetric originality signature is based on ECC and only requires a public key for the verification, which is done outside the card. The Read_Sig command can be used in both ISO/IEC 14443-3 and ISO/IEC 14443-4 protocols to retrieve the signature. If the PICC is not configured for Random ID, the command is freely available. There is no authentication required. If the PICC is configured for Random ID, an authentication is required.

11.11.1 Read_Sig

The Read_Sig retrieves the asymmetric originality signature based on an asymmetric cryptographic algorithm Elliptic Curve Cryptography Digital Signature Algorithm (ECDSA), see [14] and can be used in both ISO/IEC 14443-3 and ISO/IEC 14443-4 protocol. The purpose of originality check signature is to protect from mass copying of non NXP originated ICs. The purpose of originality check signature is not to completely prevent HW copy or emulation of individual ICs.

A public key is required for the verification, which is done outside the card. The NXPOriginalitySignature is computed over the UID and written during manufacturing. If the PICC is not configured for Random ID, the command is freely available. There is no authentication required. If the PICC is configured for Random ID, an authentication with any authentication key is required. If there is an active authentication, the command requires encrypted secure messaging.

Remark: The originality function is provided to prove that the IC has been manufactured by NXP Semiconductors.

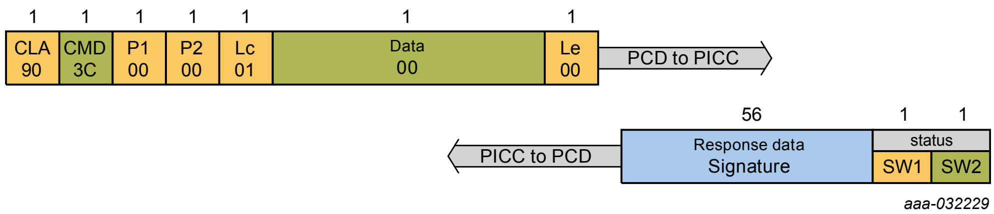


Figure 44. Read_Sig command protocol

Table 135. Command parameters description - Read_Sig

Name	Length	Value	Description
Command Header Parameters			
CMD	1	3Ch	Command code
Command Data Parameters			

Name	Length	Value	Description
Command Header Parameters			
Data	1	-	Targeted ECC originality check signature
		00h	Targeting NXPOriginalitySignature
		01h .. FFh	RFU

Table 136. Response data parameters description - Read_Sig

Name	Length	Value	Description
Data	56	Full range	NXPOriginalitySignature
SW1SW2	2	9100h 91XXh	successful execution Table 137

Table 137. Return code description - Read_Sig

Status	Value	Description
COMMAND_NOT_FOUND	0Bh	Not allowed without valid authentication due to Random ID configuration.
		Missing or incorrect MAC when authenticated.
COMMAND_ABORTED	CAh	Previous Command was not fully completed. Not all Frames were requested or provided by the PCD.
COMMAND_FORMAT_ERROR	0Ch	Unexpected command length.
		Unsupported Address

The NXPOriginalitySignature is computed over the UID with the use of asymmetric cryptographic algorithm Elliptic Curve Cryptography Digital Signature Algorithm (ECDSA), see [14]. No hash is computed: M is directly used as H. The NXP Originality Signature calculation uses curve secp224r1. NXP Originality signature verification together with example is explained in MF2DL(H)x0 - Feature and Hints application note.

12 Limiting values

Stresses exceeding one or more of the limiting values, can cause permanent damage to the device. Exposure to limiting values for extended periods can affect device reliability.

Table 138. Limiting values

In accordance with the Absolute Maximum Rating System (IEC 60134).

Symbol	Parameter	Conditions		Min	Max	Unit
I_I	input current	on LA/LB		-	40	mA
P_{tot}	total power dissipation			-	120	mW
T_{stg}	storage temperature			-55	125	°C
T_{amb}	ambient temperature			-25	70	°C
V_{ESD}	electrostatic discharge voltage	on LA/LB	[1]	-	2	kV

[1] ANSI/ESDA/JEDEC JS-001; Human body model: C = 100 pF, R = 1.5 kΩ

CAUTION

This device is sensitive to ElectroStatic Discharge (ESD). Observe precautions for handling electrostatic sensitive devices.

Such precautions are described in the *ANSI/ESD S20.20, IEC/ST 61340-5, JESD625-A* or equivalent standards.

13 Characteristics

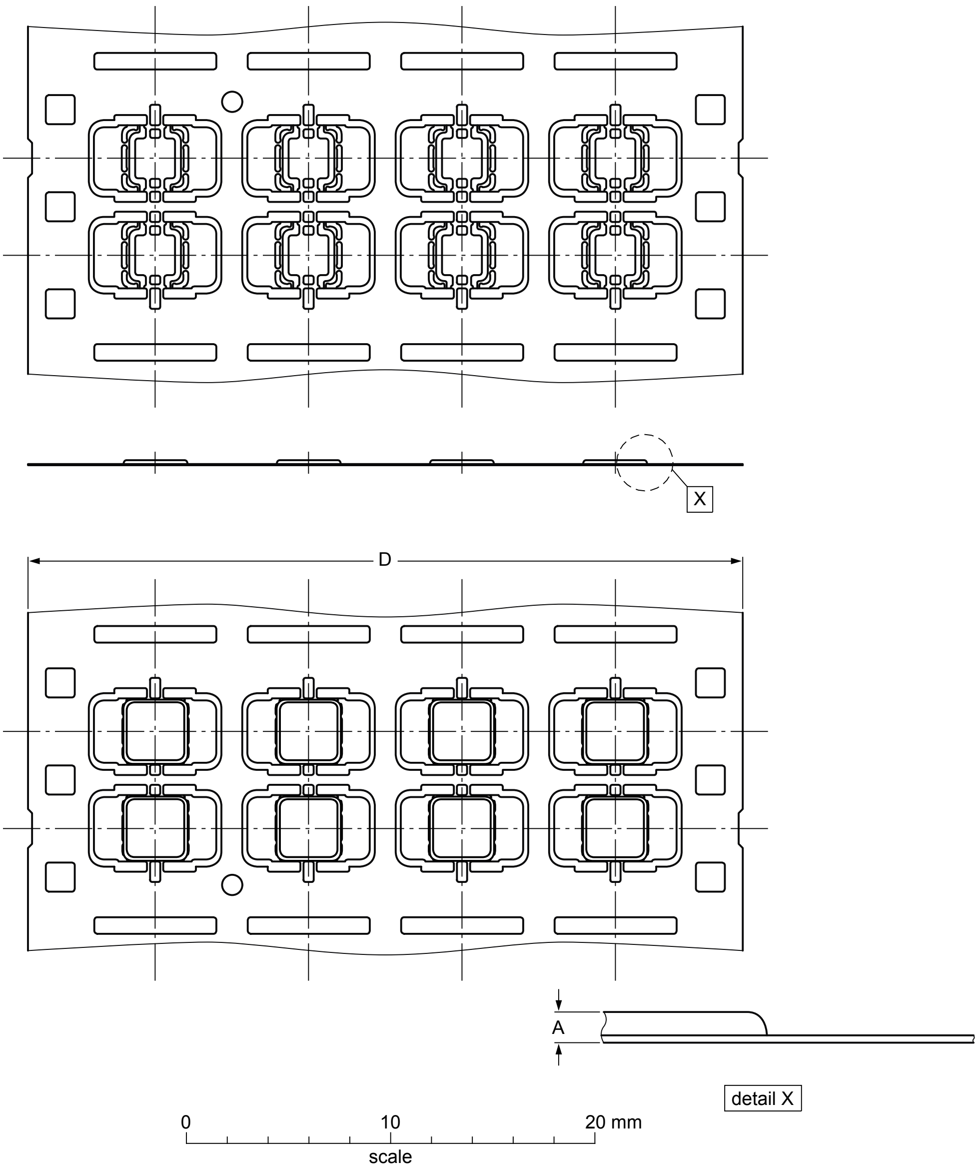
Table 139. Characteristics

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
C_i	input capacitance MF2DLx		[1]	15.3	17.0	18.7	pF
C_i	input capacitance MF2DLHx		[1]	45.0	50.0	55.0	pF
f_i	input frequency			-	13.56	-	MHz
EEPROM characteristics							
t_{ret}	retention time	T_{amb} = 22 °C		10	-	-	year
$N_{endu(W)}$	write endurance	T_{amb} = 22 °C		200.000	-	-	cycle

[1] T_{amb} = 22 °C, f = 13.56 MHz, V_{LaLb} = 2 V RMS

14 Package outline

PLLMC: plastic leadless module carrier package; 35 mm wide tape **SOT500-4**



Dimensions

Unit		A(1)	D
mm	max	0.26	35.05
	nom		35.00
	min		34.95

For unspecified dimensions see PLLMC-drawing given in the subpackage code.

Note

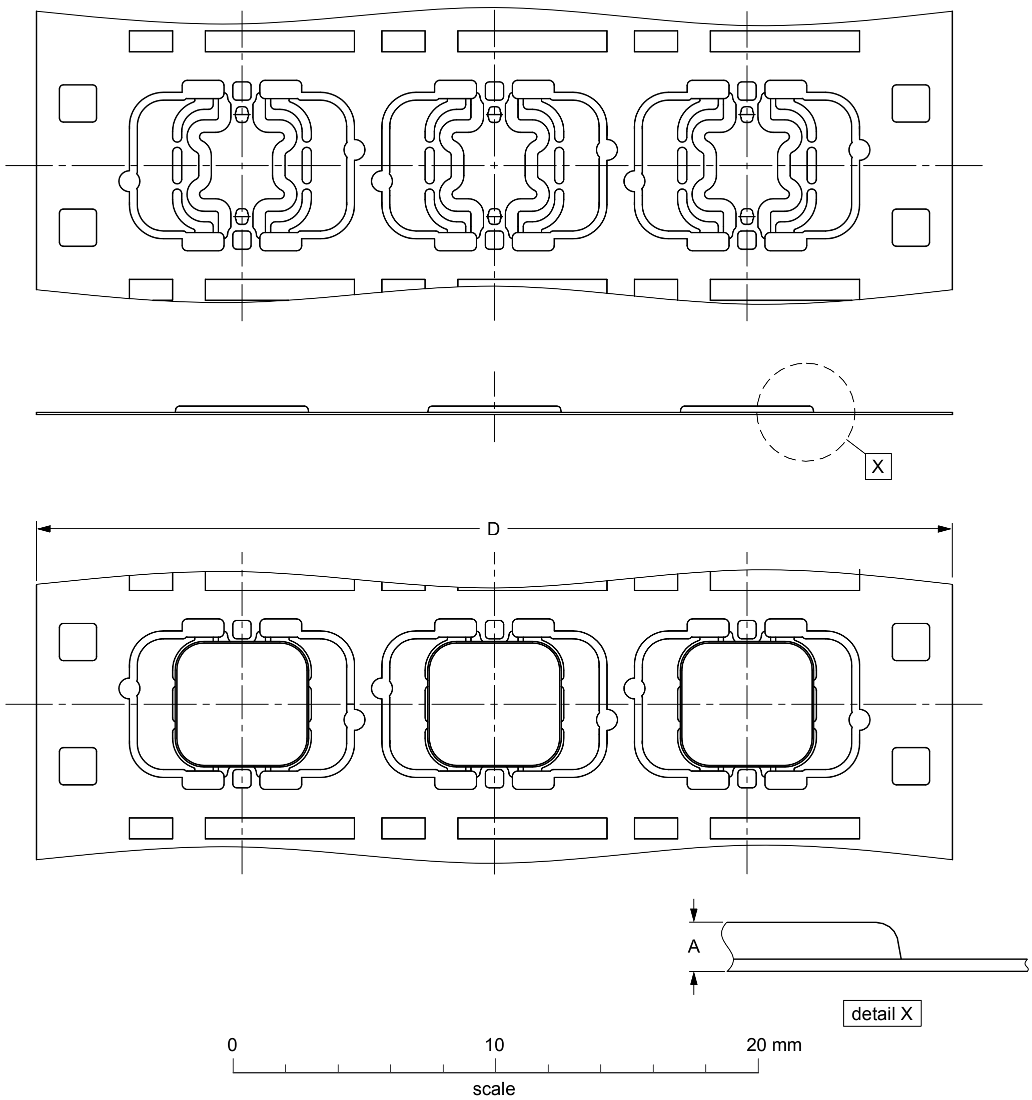
1. Total package thickness, exclusive punching burr.

sot500-4_po

Outline version	References				European projection	Issue date
	IEC	JEDEC	JEITA			
SOT500-4	- - -	- - -	- - -			11-02-18

Figure 45. Package outline SOT500-4 (MOA8)

PLLMC: plastic leadless module carrier package; 35 mm wide tape **SOT500-2**



DIMENSIONS (mm are the original dimensions)

UNIT	A(1) max.	D	
mm	0.33	35.05 34.95	For unspecified dimensions see PLLMC-drawing given in the subpackage code.

Note

1. Total package thickness, exclusive punching burr.

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	JEITA			
SOT500-2	- - -	- - -	- - -			~~03-09-17~~ 06-05-22

Figure 46. Package outline SOT500-2 (MOA4)

For details on the contactless modules MOA4 and MOA8 refer to [11] and [12].

15 Abbreviations

Table 140. Abbreviations

Acronym	Description
AES	Advanced Encryption Standard
AID	Application Identifier
APDU	Application Protocol Data Unit
AppKey	Application Key
AppMasterKey	Application Master Key
AppCommitReaderIDKey	Key to be used to commit a Reader ID for Transaction MAC calculations
AppTransactionMACKey	Transaction MAC Key, refers to one of the AppKeys
ATQA	Answer to Request A
ATS	Answer to Select
C-APDU	Command APDU
CBC	Cipher Block Chain, one mode of operation for block ciphers
CID	Channel Identifier
CLA	Class
CMAC	Cipher-based Message Authentication Code
CmdCtr	Command Counter
DF	Dedicated File (Application)
EAL	Evaluation Assurance Level
ECC	Elliptic Curve Cryptography
EF	Elementary File (File)
FCI	File Control Information
FFC	Film Frame Carrier
FID	File Identifier
FSC	Frame Size for proximity Card
FWI	Frame Waiting Time Integer
INS	Instructions
IV	Initialization Vector
LRP	Leakage Resilient Primitive
LSB	Least Significant Byte
MAC	Message Authentication Code
MF	Master File (PICC Level)
MSB	Most Significant Byte
NFC	Near Field Communication
NVM	Non-Volatile Memory
PCD	Proximity Coupling Device (Contactless Reader)

Acronym	Description
PCDCap	Proximity Coupling Device Capabilities
PD	Proximity Device, used as synonym for the PICC
PDCap	Proximity Device Capabilities
PICC	Proximity IC Card
PPS	Protocol Parameter Select
RC	Return Code
R-APDU	Response APDU
RFU	Reserved for future use
RID	Random ID
SAK	Select Acknowledge
SAM	Secure Access Module
SesAuthENCKey	Session key for encryption
SesAuthMACKey	Session key for MACing
SesTMENCKey	Transaction MAC Session Key for Encryption
SesTMMACKey	Transaction MAC Session Key for MACing
SM	Secure Messaging
SV	Session Vector, input for session key calculation
SW	Status Word
TI	Transaction Identifier
TMAC	Transaction MAC
TMC	Transaction MAC Counter
TMCLimit	Transaction MAC Counter Limit
TMI	Transaction MAC Input
TMRICur	Current Transaction MAC Reader ID
TMRIPrev	Previous Transaction MAC Reader ID
TMV	Transaction MAC Value
TotFailCtr	Total Failed Authentication Counter
TotFailCtrDecr	Total Failed Authentication Counter Decrement
TotFailCtrLimit	Total Failed Authentication Counter Limit
UID	Unique IDentifier

16 References

[1] **ISO/IEC 14443-2:2016**
Identification cards -- Contactless integrated circuit cards -- Proximity cards -- Part 2: Radio frequency power and signal interface

[2] **ISO/IEC 14443-3:2016**

Identification cards -- Contactless integrated circuit cards -- Proximity cards -- Part 3: Initialization and anti-collision

[3] **ISO/IEC 14443-4:2016**
Identification cards -- Contactless integrated circuit cards -- Proximity cards -- Part 4: Transmission protocol

[4] **ISO/IEC 7816-4:2005**
Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange

[5] **NIST Special Publication 800-38A**
National Institute of Standards and Technology (NIST). Recommendation for BlockCipher Modes of Operation.
http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf

[6] **NIST Special Publication 800-38B**
National Institute of Standards and Technology (NIST). Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication.
http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf

[7] **ISO/IEC 9797-1:1999**
Information technology – Security techniques – Message Authentication Codes (MACs) – Part 1: Mechanisms using a block cipher.

[8] **NIST Special Publication 800-108**
National Institute of Standards and Technology (NIST). Recommendation for key derivation using pseudorandom functions.

[9] **IEEE Std 802.3-2008**
IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements Part 3: Carrier sense multiple access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications.

[10] **LRP**
Leakage Resilient Primitive (LRP) Specification, Document number 4660** [1]

[11] **Contactless smart card module specification MOA4**
Delivery Type Description, Document number 0823**[1]

[12] **Contactless smart card module specification MOA8**
Delivery Type Description, Document number 1636**[1]

[13] **Data sheet addendum**
MF2DL(H)x0 - Wafer specification, Document number 4475**[1]

[14] **Certicom Research. Sec 1**
Elliptic curve cryptography. Version 2.0, May 2009.

17 Revision history

Table 141. Revision history

Document ID	Release date	Data sheet status	Change notice	Supersedes
MF2DL_H_x0 v.3.3	20190405	Product data sheet	-	MF2DL_H_x0 v.3.2
Modifications:	• Table 8, Table 9 and Table 119 updated			
MF2DL_H_x0 v. 3.2	20190212	Product data sheet	-	MF2DL_H_x0 v. 3.1
Modifications:	• Authentication command updated			

1 ** ... document version number

Document ID	Release date	Data sheet status	Change notice	Supersedes
MF2DL_H_x0 v. 3.1	20190207	Product data sheet	-	MF2DL_H_x0 v. 3.0
Modifications:	• Editorial update			
MF2DL_H_x0 v. 3.0	20190131	Product data sheet	-	430712
Modifications:	• Data sheet status changed into Product data sheet • Security status changed into "Company public"			
430712	20181105	Objective data sheet		430711
Modifications:	• added *response codes* in Section 11.3 • clarified generation of *SDMMetaReadUpdateKey* in LRP mode encryption			
430711	20180321	Objective data sheet	-	430710
Modifications:	• added TMCLimit to ChangeFileSettings and GetFileSettings • added note about the limit of data within a single ISO/IEC 7816-4 frame in Section 8.4 • added clarification about the authentication delay after unsuccessful AES mode authentications in Section 9 • added default file parameters to the individual sections in chapter Section 8.2.3 • added details for PCDcap2 in authentication commands, see Section 11.4.1 and Section 11.4.3 • corrected GetVersion response • clarified Application ID renaming option in Section 11.5.1 • clarifications added across the data sheet • editorial changes			
430710	20180111	Objective data sheet	-	-

18 Legal information

18.1 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.

18.2 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.

Product specification — The information and data provided in a Product data sheet shall define the specification of the product as agreed between NXP Semiconductors and its customer, unless NXP Semiconductors and customer have explicitly agreed otherwise in writing. In no event however, shall an agreement be valid in which the NXP Semiconductors product is deemed to offer functions and qualities beyond those described in the Product data sheet.

18.3 Disclaimers

Limited warranty and liability — 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. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors. In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory. Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

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 life support, life-critical or safety-critical systems or 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 and its suppliers accept 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. Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products. NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Limiting values — Stress above one or more limiting values (as defined in the Absolute Maximum Ratings System of IEC 60134) will cause permanent damage to the device. Limiting values are stress ratings only and (proper) operation of the device at these or any other conditions above those given in the Recommended operating conditions section (if present) or the Characteristics sections of this document is not warranted. Constant or repeated exposure to limiting values will permanently and irreversibly affect the quality and reliability of the device.

Terms and conditions of commercial 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, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

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.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications. In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

18.4 Licenses

ICs with DPA Countermeasures functionality	
LICENSED DPA COUNTERMEASURES™	NXP ICs containing functionality implementing countermeasures to Differential Power Analysis and Simple Power Analysis are produced and sold under applicable license from Cryptography Research, Inc.

18.5 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.

Tables

Figures

Contents

Date of release: 5 April 2019
Document identifier: MF2DL_H_x0
Document number: 430733