



# DUALi Protocol Specification

## DUALi Inc.

Document Version: 2.50

Last Revised Date: 19 NOV. 2019

---

Copyright © 2013-2014 DUALi Inc. All rights reserved. You are strictly prohibited to copy, disclose, distribute, or use this document in part or as a whole for any purposes other than those for which this document is disclosed. This document is copyrighted and contains confidential information and other intellectual property rights of DUALi Inc. Any unauthorized use, copy, disclosure or distribution constitutes infringement of DUALi's intellectual property rights.

DUALi Inc. reserves the right to make changes to its applications or services or to discontinue any application or service at any time without notice. DUALi provides customer assistance in various technical areas, but does not have full access to data concerning the use and applications of customer's products.

Therefore, DUALi assumes no liability and is not responsible for customer applications or software design or performance relating to systems or applications incorporating DUALi products. In addition, DUALi assumes no liability and is not responsible for infringement of patents and/or any other intellectual or industrial property rights of third parties, which may result from assistance provided by DUALi.

Composition of the information in this manual has been done to the best of our knowledge. DUALi does not guarantee the correctness and completeness of the details given in this manual and may not be held liable for damages ensuing from incorrect or incomplete information. Since, despite all our efforts, errors may not be completely avoided, we are always grateful for your useful tips.

We have our development center in South Korea to provide technical support. For any technical assistance can contact our technical support team as below;

Tel: +82 31 213 0074

e-mail : [sales@duali.com](mailto:sales@duali.com)

## **Revision History**

- 2009.02.17(Ver. 1.0) : First Release
- 2009.03.10(Ver. 1.01) : Timeout table update
- 2009.03.16(Ver. 1.02): Timeout extension command(0x1B,0x1C),ISO7816 T=0 parity test command(0xCA)
- 2009.04.29(Ver. 1.03): ISO 15693 commands added (only valid for 632 version, DE-620R)
- 2009.07.15(Ver. 1.04): Anti-tearing, BitMode command added
- 2009.08.08(Ver. 1.05): ISO 15693 commands modified
- 2009.08.15(Ver. 1.05): Inventory function modified and \_DED\_FindCard(0x79) function added  
PC/SC function added (still under going), \_DED\_Inventory, \_DED\_FindCard modified  
\_DEA\_CIM added, Add FeliCa batch command
- 2009.10.30(Ver. 1.06): Delete \_DED\_FIND\_ISO15693(0x79) command  
Update \_DED\_INVENTORY(0x70) command  
Add ICODE1 command \_DED\_TRANSPARENT\_I(0x76)
- 2009.11.02(Ver. 1.10): Delete functions for chip developers
- 2009.12.02(Ver. 1.11): Update timeout value in <TIMEOUT TABLE>
- 2009.12.06(Ver. 1.12): Change AFI in '15693 Find Card' command as optional.
- 2009.12.18(Ver. 1.13): Change the explain of "IC Card Speed Set" command
- 2010.01.19(Ver. 1.14): Change response of "Select Level" command.
- 2010.03.12(Ver. 1.15): '15693 Next Slot' command(3.7.7) return includes CRC.
- 2010.06.25(Ver. 1.16): Anti Tearing Command insert.
- 2010.08.23(Ver. 1.17): Modify IC Card Power On and add IC Card Bypass.
- 2010.09.13(Ver. 1.18): Add NFC commands
- 2010.12.01(Ver. 1.19):Update Flash Read/Write protocol
- 2011.06.03(Ver. 1.20):Add RS-232 Protocol
- 2011.06.13(Ver. 1.21):Modify \_NFC\_ISTART(0x94), \_NFC\_TSTART(0x95), \_NFC\_STOP command.  
Add new \_NFC\_GET\_TARGET(0x99) & \_NFC\_SET\_TARGET(0x9A) commands.  
Add new LLC Protocol related commands.
- 2011.09.26(Ver. 1.22):Add NFC Tags-related commands
- 2011.10.27(Ver. 1.23):Add Card Existence check command, Add Buzzer Control with tone(HTY)
- 2012.01.11(Ver. 1.24):Add LED control to Buzzer Control command(HTY)
- 2012.01.26(Ver. 1.25):Add DCLB mode at Mode Change(HTY)
- 2012.01.31(Ver. 1.26):Add LCD and RTC command for DE-630, RF TRX speed mistyping correction(HTY)

- 2012.02.01 : Add NFC T1 Tag Transparent(HTY)
- 2012.02.24(Ver. 2.0): Add Reader Information command(HTY)
  - Add FeliCa card batch functions
- 2012.04.03(Ver. 2.01): Add explanation for Response Code
- 2012.04.06(Ver. 2.02): Save MIFARE keys to nonvolatile memory(HTY)
- 2012.04.13(Ver. 2.03): Open UID get command(0x4D) which had been not opened yet(HTY)
  - Add SAK to response of *Request to Select command*. (Document correction)
- 2012.06.11(Ver. 2.10): Update for VHBR functions(HTY)
  - Add VHBR speed at 'RF Find Card (Command = 0x4C)' function.
  - Add VHBR speed at 'RF Speed Set (Command = 0x1A)'
  - Add 'SPARAMETER Activate (Command = 0x68)'
  - Modify 'Mode Change (Command = 0x15)' to support FSDI update
- 2012.08.17(Ver. 2.1x): SIO-15693 card, 8 byte write function added
- 2012.10.07(Ver. 2.12): Add Set RF Para (Command = 0x1D) options to set CID, FWI and FSDI
- 2013.01.07(Ver. 2.13): Add LED control in title with BUZZER
- 2013.01.08(Ver. 2.13): DE-930 and DE-950 Serial communication speed is 9600bps
- 2013.01.11(Ver. 2.14): Set UID command(0x4E) was added
- 2013.03.22(Ver. 2.15): Saving MIFARE key to volatile memory was added
- 2013.07.01(Ver. 2.16): Add Reading NFC BardCode feature & Card Emulation feature
  - Modify LLCP Command format to use easily.
  - Merge all SNEP & NPP batch commands into one command.
- 2013.08.26(Ver. 2.17): Fix several command descriptions in LLCP
- 2014.02.14(Ver. 2.20): Add LCD image related functions and RTC time display function
- 2014.02.25(Ver. 2.21): Add explanation to transmit EOF after write command to ISO15693 card.
- 2014.07.30(Ver. 2.22): Add MIFARE read multi block command(0x42, chapter3.2.23).
- 2014.08.27(Ver. 2.23): Add MIFARE \_DEA\_ANTICOLL\_LEVEL(0x3E) and \_DEA\_SELECT\_LEVEL(0x3F) example for 7UID.
- 2014.09.15(Ver. 2.24): Add MIFARE write multi block command(0x43, chapter3.2.24).
- 2014.12.09(Ver. 2.25): Add Buzzer Control with time command(0x13, chapter3.1.4).
- 2015.01.26(Ver. 2.26): Add RF selection command(0x69, chapter3.1.20).
- 2015.03.13(Ver. 2.27): Add Explanation for PPS(0xC0, chapter3.7.1).
- 2015.06.25(Ver. 2.28): Find Tags (Command = 0x83, chapter 3.6.5).
- 2015.09.30(Ver. 2.29): Reader Number Write, Response data add(Command = 0x7A, chapter 3.1.12).
- 2015.12.15(Ver. 2.30): Add commands to Mode change (3.1.5).
- 2016.06.01(Ver. 2.40): Add commands for NDEF tags (3.9.14).
- 2016.07.18(Ver. 2.41): Add description for IC Power On without IFS request (3.7.1).
- 2016.07.18(Ver. 2.41): Add description for IC Power On without IFS request (3.7.1).

- 2016.11.28(Ver. 2.42): Add 'RF Find Card(0x4C)' without sending RATS(3.6.1)  
Add Rats\_PPS command (0x4B)
- 2017.01.20(Ver. 2.43): LCD command(0x7C) example for DE-632 correction(3.1.14)
- 2017.08.07(Ver. 2.44): LED mode command(0x13) add (3.1.4)
- 2018.04.25(Ver. 2.45): MIFARE read multi block command(0x42, chapter3.2.23) option byte add.
- 2019.11.19(Ver. 2.50): Barcode(QR) data get command(0x0A, chapter3.2.23) add.

## **CONTENTS**

1.	INTRODUCTION.....	9
2.	INTERFACE SPECIFICATION .....	10
2.1	COMMUNICATION PROTOCOL FRAME FORMAT.....	10
2.1.1	Command frame format.....	10
2.1.2	Response frame format .....	10
2.1.3	RS-232 Protocol (option) .....	10
3.	COMMAND DEFINITION .....	11
3.1	COMMON FUNCTIONS .....	11
3.1.1	RF ON (Command = 0x10, _DE_RFON) .....	11
3.1.2	RF OFF (Command = 0x11, _DE_RFOFF) .....	11
3.1.3	Terminal Reset (Command = 0x12, _DE_RESET).....	11
3.1.4	Buzzer/LED Control (Command = 0x13, _DE_BUZZER).....	12
3.1.5	Mode Change (Command = 0x15, _DE_DEV_CHANGE) .....	13
3.1.6	Get Version (Command = 0x16, _DE_VERSION).....	14
3.1.7	RF Speed Set (Command = 0x1A, _DE_TRXSPEED) .....	15
3.1.8	RF WTX Extend (Command = 0x1B, _DE_RF_WTX).....	15
3.1.9	Contact WTX Extend (Command = 0x1C, _DE_CONTACT_WTX) .....	16
3.1.10	Flash Read/Write (Command = 0x1F, _DE_FLASH).....	16
3.1.11	RF Reset (Command = 0x20, _DE_RF_RESET).....	17
3.1.12	Reader Number Write (Command = 0x7A, _DE_RW_WRITE).....	17
3.1.13	Reader Number Read (Command = 0x7B, _DE_RW_READ) .....	18
3.1.14	LCD Commands(Command = 0x7C, _DE_RW_LCD, optional).....	18
3.1.15	RTC Write/Read/Display (Command = 0x7D, _DE_RW_RTC, optional) .....	19
3.1.16	Reader Information (Command = 0x09, _DE_INF_GET).....	21
3.1.17	Get UID (Command = 0x4D, _DE_UID_GET).....	21
3.1.18	Set UID (Command = 0x4E, _DE_UID_SET).....	21
3.1.19	Set RF Para (Command = 0x1D, _DE_PARA_SET).....	22
3.1.20	Select RF Antenna (Command = 0x69, DE-AFCMI only).....	22
3.2	MIFARE AND A-TYPE CARD FUNCTIONS.....	23
3.2.1	Idle Request (Command = 0x21, _DEA_IDLE_REQ) .....	23
3.2.2	Wakeup Request (Command = 0x22, _DEA_WAKEUP_REQ).....	23

3.2.3	Anti-Collision (Command = 0x23, _DEA_ANTICOLL) .....	24
3.2.4	Select (Command = 0x24, _DEA_SELECT) .....	24
3.2.5	Authentication Mifare (Command = 0x25, _DEA_AUTH) .....	24
3.2.6	Halt (Command = 0x26, _DEA_HALT) .....	25
3.2.7	Read Mifare (Command = 0x27, _DEA_READ) .....	25
3.2.8	Write Mifare (Command = 0x28, _DEA_WRITE) .....	26
3.2.9	Increment Mifare (Command = 0x29, _DEA_INCREMENT) .....	26
3.2.10	Decrement Mifare (Command = 0x2A, _DEA_DECREMENT) .....	27
3.2.11	Increment and Transfer (Command = 0x2B, _DEA_INC_TRANS) .....	27
3.2.12	Decrement and Transfer (Command = 0x2C, _DEA_DEC_TRANS) .....	28
3.2.13	Restore Mifare (Command = 0x2D, _DEA_RESTORE) .....	28
3.2.14	Transfer Mifare (Command = 0x2E, _DEA_TRANSFER) .....	29
3.2.15	Load Key Mifare (Command = 0x2F, _DEA_LOADKEY) .....	29
3.2.16	Auth Key Mifare (Command = 0x30, _DEA_AUTHKEY) .....	30
3.2.17	Request to Auth (Command = 0x31, _DEA_REQ_ANTI_AUTH) .....	30
3.2.18	Request to Auth Key (Command = 0x32, _DEA_REQ_ANTI_AUTHKEY) .....	31
3.2.19	Increment and Transfer2 (Command = 0x33, _DEA_INC_TRANS2) .....	31
3.2.20	Decrement and Transfer2 (Command = 0x34, _DEA_DEC_TRANS2) .....	32
3.2.21	Request to Read A (Command = 0x35, _DEA_REQ_ANTI_AUTH_READ) .....	33
3.2.22	Request to Read K (Command = 0x36, _DEA_REQ_ANTI_AUTHKEY_READ) .....	33
3.2.23	Request to Read Multi (Command = 0x42, _DEA_REQ_AUTHKEY_READ_M) .....	34
3.2.24	Request to Write Multi (Command = 0x43, _DEA_REQ_AUTHKEY_WRITE_M) .....	35
3.2.25	Request to Write A (Command = 0x37, _DEA_REQ_ANTI_AUTH_WRITE) .....	36
3.2.26	Request to Write K (Command = 0x38, _DEA_REQ_ANTI_AUTHKEY_WRITE) .....	36
3.2.27	Request to Select (Command = 0x39, _DEA_REQ_ANTI_SEL) .....	37
3.2.28	Write Ultra Light (Command = 0x3B, _DEA_UWRITE) .....	38
3.2.29	Full Anti-Collision and Select (Command = 0x3D, _DEA_ANTI_SEL_LEVEL) .....	38
3.2.30	Anti-Collision Level (Command = 0x3E, _DEA_ANTICOLL_LEVEL) .....	39
3.2.31	Select Level (Command = 0x3F, _DEA_SELECT_LEVEL) .....	39
3.2.32	Device Information (Command = 0x40, _DEA_DEVINFO) .....	40
3.2.33	Type-A Transparent (Command = 0x41, _DEA_TRANSPARENT) .....	40
3.2.34	Type-A Transparent2 (Command = 0x47, _DEA_TRANSPARENT2) .....	41
3.2.35	NFC T1 Tag Transparent (Command = 0x6B) .....	42
3.2.36	RATS and PPS (Command = 0x4B, _DEA_RATS_PPS) .....	42
3.3	FELICA CARD FUNCTIONS .....	44
3.3.1	Type-C Transparent (Command = 0x50, _DEC_TRANSPARENT) .....	44
3.3.2	Type-C Polling (Command=0x51, _DEC_POLLING_NOENC) .....	44
3.3.3	Type-C Read (Command=0x52, _DEC_READ_NOENC) .....	45

3.3.4	Type-C Write (Command=0x53, _DEC_WRITE_NOENC).....	45
3.4	FELICA CARD BATCH FUNCTIONS.....	46
3.4.1	FeliCa SAM Authentication (Command = 0x56) .....	46
3.4.2	FeliCa Mutual Authentication (Command=0x57) .....	46
3.4.3	FeliCa Mutual Authentication RWSAM (Command=0x58) .....	47
3.4.4	FeliCa Command (Command=0x59) .....	47
3.5	B-TYPE CARD FUNCTIONS .....	48
3.5.1	Type-B Transparent (Command = 0x60, _DEB_TRANSPARENT) .....	48
3.5.2	Type-B Transparent2 (Command = 0x6E, _DEB_TRANSPARENT2).....	48
3.6	A/B-TYPE COMMON FUNCTIONS.....	50
3.6.1	RF Find Card (Command = 0x4C, _DE_FIND_CARD) .....	50
3.6.2	RF APDU (Command = 0x61, _DE_APDU) .....	51
3.6.3	RF CARD STATUS (Command = 0x62, _DE_EXIST_CHK) .....	51
3.6.4	SPARAMETER Activate (Command = 0x68).....	52
3.6.5	RF Find Tags (Command = 0x83, _DE_FIND_TAGS) .....	54
3.7	IC-CARD FUNCTIONS.....	55
3.7.1	IC Card Power On (Command = 0xC0, _DE_CARD_PON).....	55
3.7.2	IC Card Case1 (Command = 0xC1, _DE_CARD_CASE1) .....	56
3.7.3	IC Card Case2 (Command = 0xC2, _DE_CARD_CASE2) .....	57
3.7.4	IC Card Case3 (Command = 0xC3, _DE_CARD_CASE3) .....	57
3.7.5	IC Card Case4 (Command = 0xC4, _DE_CARD_CASE4) .....	58
3.7.6	IC Card Power Off (Command = 0xC5, _DE_CARD_POFF) .....	58
3.7.7	IC Card T1 Bypass (Command = 0xC7, _DE_CARD_T1BYPASS) .....	59
3.7.8	IC Card Speed Set (Command = 0xC8, _DE_CARD_SPEED).....	60
3.7.9	IC Card APDU (Command = 0xC9, _DE_CARD_APDU).....	60
3.7.10	IC Card BYPASS (Command = 0xCB, _DE_CARD_BYPASS).....	61
3.7.11	IC CARD STATUS (Command = 0xCA, _DE_CARD_EXIST).....	61
3.8	ISO-15693 COMMANDS (OPTIONAL FUNCTIONS).....	62
3.8.1	15693 Find Card (Command = 0x70, _DED_INVENTORY) .....	62
3.8.2	15693 Select (Command = 0x71, _DED_SELECT) .....	62
3.8.3	15693 Read (Command = 0x72, _DED_READ).....	63
3.8.4	15693 Write (Command = 0x73, _DED_WRITE) .....	63
3.8.5	15693 Write (Command = 0x7E, _DED_WRITE8).....	64
3.8.6	15693 Transparent (Command = 0x74, _DED_TRANSPARENT) .....	64
3.8.7	15693 Transparent2 (Command = 0x75, _DED_TRANSPARENT2) .....	65
3.8.8	15693 Next Slot (Command = 0x78, _DED_NEXTSLOT).....	65
3.8.9	I-CODE Transparent (Command = 0x76, _DED_TRANSPARENT_I).....	66
3.9	NFC COMMANDS (OPTIONAL FUNCTIONS) .....	67



3.9.1	Device Initialization (Command = 0x90, _NFC_INIT).....	67
3.9.2	Initiator Transparent (Command = 0x91, _NFC_INITIATOR_COMM) .....	68
3.9.3	Target Transparent (Command = 0x92, _NFC_TARGET_COMM) .....	68
3.9.4	Device (Command = 0x93, _NFC_CARD_MODE) .....	69
3.9.5	Start Initiator Mode (Command = 0x94, _NFC_ISTART).....	69
3.9.6	Start Target Mode (Command = 0x95, _NFC_TSTART) .....	71
3.9.7	Initiator DEP Data(Command = 0x96, _NFC_SEND_DATAI) .....	72
3.9.8	Target DEP Data(Command = 0x97, _NFC_SEND_DATAT) .....	72
3.9.9	Deselect/Release/Wakeup/Stop (Command = 0x98, _NFC_STOP) .....	73
3.9.10	Get Target Data(Command = 0x99, _NFC_GET_TARGET) .....	73
3.9.11	Set Target Data (Command = 0x9A, _NFC_SET_TARGET).....	74
3.9.12	Get Target State (Command = 0x9B, _NFC_TARGET_STATE).....	74
3.9.13	Set Target Data (Command = 0x9A, _NFC_SET_TARGET).....	75
3.9.14	NDEF Tag Batch Commands (Command = 0x9D, _NFC_NDEF_TAG) .....	75
3.9.15	NFC_Tag_Command (Command = 0x9E, _NFC_TAG_COMMAND).....	76
3.10	LLC COMMANDS .....	90
3.10.1	MAC Activation (Command = 0xBE, _LLC_MAC_ACTIVE) .....	90
3.10.2	MAC Deactivation (Command = 0xBF, _LLC_MAC_DEACTIVE) .....	92
3.10.3	LLC Transceive PDU (Command = 0xB0, _LLC_Transceive_PDU) .....	92
3.10.4	LLC Get PDU (Command = 0xB5, _LLC_GET_PDU).....	94
3.10.5	LLC Set PDU (Command = 0xB6, _LLC_SET_PDU).....	94
3.10.6	Start Tag Emulation (Command = 0xBB, _LLC_TAG_EMU) .....	95
3.10.6	Connection-oriented LLC Batch (Command = 0xBC, _LLC_CO_BATCH) .....	96
3.11	GET DEVICE DATA COMMANDS (0x0A) .....	98
3.11.1	Get QR Data (Command = 0x0A00, _DE_GET_DATA) .....	98
4.	RESPONSE CODE DEFINITION .....	99

## 1. Introduction

This document defines the USB and serial communication protocol between DUALi's readers and a host computer (Some readers support USB only, but some other readers and modules support RS-232 with same protocol. So this document includes the RS-232 also).

DUALi's readers and modules support ISO7816, ISO14443 type A/B, ISO18092, my-d™, Mifare®, DESFire® FeliCa™, ISO15693 and I-CODE cards and high communication speed depending on the chip used in the reader.

**DE-620** supports ISO7816, ISO14443 type A/B, ISO18092, my-d™, Mifare®, DESFire® and FeliCa™ cards. (**DE-ABCM, DE-ABM5, Dragon, DE-ABCM6**)

**DE-620R** supports ISO7816, ISO14443 type A/B, my-d™, Mifare®, DESFire®, ISO15693 and I-CODE cards. (**DE-ABM4**)

**DE-620L** supports ISO7816, ISO14443 type A/B, my-d™, Mifare®, DESFire® cards. (**DE-ABM2, DE-ABM4SIC**)

This document is dedicated for all readers and modules. So, when you use some reader product of DUALi, those readers have a possibility to return code UNKNOWN COMMAND ERROR(23, 0x17), it means your reader or module don't support this command.

my-d™ is registered trademarks of Infineon Technologies AG  
FeliCa™ is registered trademark of SONY Corporation  
Mifare® and DesFire® are registered trademarks of NXP Semiconductors

## 2. Interface Specification

### 2.1 Communication Protocol Frame Format

#### 2.1.1 Command frame format

(To show a Hexadecimal number, "0x" is appended to the first of the number)

Name	STX	LEN-H	LEN-L	CMD	Data[n]	LRC
Values	0x02	0xHH	0xHH	0xHH	Data[n]	0xHH
Length.	1-byte	1-byte	1-byte	1-byte	n-byte	1-byte

**(STX and LRC are for RS-232. In case of USB, these elements should be removed)**

- STX : 0x02
- Data length = command (It must be 1) + Length of data.
  - LEN-H : Higher byte of data length
  - LEN-L : Lower byte of data length
- CMD : Command byte
- Data[n] : Data bytes
- LRC : XORing from LEN-H to Data[n]

#### 2.1.2 Response frame format

Name	STX	LEN-H	LEN-L	Resp	Data[n]	LRC
Values	0x02	0xHH	0xHH	0xHH	Data[n]	0xHH
Length.	1-byte	1-byte	1-byte	1-byte	n-byte	1-byte

**(STX and LRC are for RS-232. In case of USB, these elements should be removed)**

- STX : 0x02
- Data length = response (it must be 1) + Length of data.
  - LEN-H : Higher byte of data length
  - LEN-L : Lower byte of data length
- Resp : Response code from reader
- Data[n] : Data Bytes
- LRC : XORing from LEN-H to Data[n]

#### 2.1.3 RS-232 Protocol (option)

- Speed : 115200bps (except DE-ABM4 : 57600bps, DE-930/DE-950 : 9600bps )
- Data: 8 bit
- Parity : No
- Stop : 1 bit

### 3. Command Definition

#### 3.1 Common Functions

##### 3.1.1 RF ON (Command = 0x10, \_DE\_RFON)

This command activates RF field.

■ Command frame

LEN-H	LEN-L	CMD
0x0001		0x10

■ Response frame

LEN-H	LEN-L	Resp
0x0001		OK(0x00) , Err(pls refer to the response code).

##### 3.1.2 RF OFF (Command = 0x11, \_DE\_RFOFF)

This command deactivates RF field.

■ Command frame

LEN-H	LEN-L	CMD
0x0001		0x11

■ Response frame

LEN-H	LEN-L	Resp
0x0001		OK(0x00) , Err(pls refer to the response code).

##### 3.1.3 Terminal Reset (Command = 0x12, \_DE\_RESET)

This command resets reader. Some or all parameters are initialized. If there is no data after CMD code, the USB is not initialized . The reader can be used without re-connect. If there are any data after CMD code, the reader will be reset in hardware. The reader will be re-connected.

■ Command frame

LEN-H	LEN-L	CMD
0x0001		0x12

■ Response frame

LEN-H	LEN-L	Resp
0x0001		OK(0x00) , Err(pls refer to the response code).

### 3.1.4 Buzzer/LED Control (Command = 0x13, \_DE\_BUZZER)

This command controls buzzer in the reader.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1,2,3]
0x0002		0x13	Buzzer Con	When Data[0] is 2 or 3, Data[1]:Octave Data[2]:Frequency Data[3]:time

#### - Data[0]

Data[0]	Buzzer/LED Control
0x00	Buzzer Turn On
0x01	Buzzer Turn Off
0x02	Buzzer Turn On with Tone(optional) Data[1] : Octave : 0~7 Data[2] : Frequency : 0~11(Do~Si)
0x03	Buzzer Turn On with Tone and time Data[1] : Octave : 0~7 Data[2] : Frequency : 0~11(Do~Si) Data[3] : time in scale of 10millisecond
0x10	LED Control (optional) Data[1] : LED position Data[2] : 0(off) or 1(on)
0x11	LED Blink Data[1] : LED position Data[2..3] : Blink time in millisecond
0x12	LED mode Set (added at 2017.08.07) Data[1] : PC/SC LED mode 0: automatic LED (when card is detected and there is communication) (default) 3: all automatic LED control disable. LED is controllable only with command(0x02:communication LED disable, 0x01:PCSC connection LED disable)

#### ■ Response frame

LEN-H	LEN-L	Resp
0x0001		OK(0x00) , Err(pls refer to the response code).

### 3.1.5 Mode Change (Command = 0x15, \_DE\_DEV\_CHANGE)

This command changes operating mode of the reader. DE-620 supports 4 modes like below :

- Vendor : This mode changes DE-620 as dummy(transparent) reader. User can execute all commands in this document and could access all cards which are referred in the DE-620 specification by using proper protocol for each card.
- PC/SC : This mode changes DE-620 as PC/SC reader. It can supports PC/SC Contact reader , PC/SC Contactless reader, PC/SC SAM reader simultaneously or separately. In this mode, user shall use IOCTLs command to access Contact or SAM. User can execute restricted commands than vendor mode because PC/SC mode only supports cards which have conformity with ISO7816 in case of contact or ISO14443 A/B part4 in case of contactless.  
In order to use vendor command in this mode, IOCTLs also give easy way to use some vendor command. (DUALi IOCTLs.txt shows how to use this method in PC/SC mode)
- FSDI : Reader's default FSDI(Frame Size of Device Integer) is 8(256bytes long). This value could be change using this command.

#### ■ Command frame (changing mode )

LEN-H	LEN-L	CMD	Data[0]	Data[1]	Data[2..]
0x0003~0x000D		0x15	Flag	Mode	CLF

##### - Data[0]

Data[0]	Description
0x00	Inquiry of reader's current mode Data[1] must be omitted
0x01	Mode set
0x02	<b>FSDI(Frame Size of Device Integer) set</b>
0x03	<b>Get FSDI(Frame Size of Device Integer)</b>
0x04	<b>ATR Mode set</b>
0x05	<b>Set checking card types</b>

<b>0x06</b>	<b>Product serial number fix(0x00) or variable(0x01)</b>
-------------	--

- Data[1]

Data[1]	Mode when Data[0] is 0x01
0x00	Vendor mode
0x01	PC/SC mode, RF Only
0x02	PC/SC mode, RF + Contact
0x03	PC/SC mode, RF + Contact + SAM
0x04	PC/SC mode, DCLB(Digital Contact Less Bridge), <b>DE-620V only</b>

Data[1]	Mode when Data[0] is 0x02
0x08	256bytes
0x09	512bytes
0x0A	1024bytes
0x0B	2048bytes, Not support yet
0x0C	4096bytes, Not support yet

- Data[2..] : CLF only for DCLB mode, Max 10 bytes of CLF

#### ■ Response frame for set mode

LEN-H	LEN-L	Resp
0x0001		OK(0x00) , Err(pls refer to the response code).

#### ■ Response frame for inquiry mode

LEN-H	LEN-L	Resp	Data[0]
0x0002		OK	Mode

- Data[0] : 1 byte reader mode **or FSDI**

### 3.1.6 Get Version (Command = 0x16, \_DE\_VERSION)

This command inquires the version of reader's firmware.

#### ■ Command frame

LEN-H	LEN-L	CMD
0x0001		0x16

#### ■ Response frame if response is OK

LEN-H	LEN-L	Resp	Data[0...N-1]
N+1		OK	VER[0...N-1]

- Data[0...N-1]: Version of reader.  
e.g. : 0x44452D3632305F4150505F30393031313900  
→ ASCII : DE-620\_APP\_090119
- Response: OK(0x00) Err(pls refer to the response code).

### 3.1.7 RF Speed Set (Command = 0x1A, \_DE\_TRXSPEED)

This command changes communication speed between reader and contactless card. The reader can control TX and RX speed separately and support speed up to 847kbps.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]
0x0002		0x1A	Trxspd

- Data[0] : Trxspd : 0xH<sub>1</sub> H<sub>2</sub> ( H<sub>1</sub> : RX Speed nibble, H<sub>2</sub> : TX Speed nibble)  
: H – 0(106K), 1(212K), 2(424K), 3(847K)  
**4(1.7Mbps) , 5(3.4Mbps) , 6(6.8Mbps) only for DE-620V**

Ex) 0x12(RX : 212K, TX : 424K)

**Ex) 0x63(RX : 6.8M, TX : 847K)**

#### ■ Response frame

LEN-H	LEN-L	Resp
0x0001		OK(0x00) , Err(pls refer to the response code).

### 3.1.8 RF WTX Extend (Command = 0x1B, \_DE\_RF\_WTX)

This command extends contactless waiting time more than maximum FWT(9.897s at FWI=15) During card activation sequence, this command will be initialized. This command is valid after card activation sequence.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]
0x0002		0x1B	WTXI

- Data[0]: WTXI, waiting time extension integer : waiting time = TOUT \* (WTXI + 1)
- TOUT: See \_DEA\_TRANSPARENT, \_DEA\_TRANSPARENT2,  
\_DEB\_TRANSPARENT, \_DEB\_TRANSPARENT2



#### ■ Response frame

LEN-H	LEN-L	Resp
0x0001		OK(0x00) , Err(pls refer to the response code).

### 3.1.9 Contact WTX Extend (Command = 0x1C, \_DE\_CONTACT\_WTX)

This command extends contact waiting time more than maximum WT/BWT.

During card activation sequence, this command will be initialized. This command is valid after card activation sequence.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1]
0x0003		0x1C	SLOT	WTXI

- Data[0]: WTXI, waiting time extension integer : waiting time = WTXI \* 2sec

#### ■ Response frame

LEN-H	LEN-L	Resp
0x0001		OK(0x00) , Err(pls refer to the response code).

### 3.1.10 Flash Read/Write (Command = 0x1F, \_DE\_FLASH)

This command reads/writes user data from/to internal flash memory of the reader. The size of memory is 128bytes.

#### ■ Command frame of writing data

LEN-H	LEN-L	CMD	Data[0]	Data[1]	Data[1...128]
0x82		0x1F	0xFF	Offset	128 byte data

- Data[0] : 0xFF – write data
- Data[1] : Offset (0~127)
- Data[n] : n-byte data(Maximum : 128byte)

#### ■ Response frame if response is OK

LEN-H	LEN-L	Resp
0x0001		OK(0x00) , Err(pls refer to the response code).

#### ■ Command frame of reading data

LEN-H	LEN-L	CMD	Data[0]	Data[1]	Data[2]
0x0002		0x1F	0x00	Offset	Length to read

- Data[0] : 0x00 – read data

- Data[1] : Offset(0~127)
- Data[2] : Length to read(1~128)

■ Response frame if response is OK

LEN-H	LEN-L	Resp	Data[0...127]
0x081		0x00	128byte data

- Data[0...127]: 128 byte data in FLASH memory.
- Response: OK(0x00) Err(pls refer to the response code).

### 3.1.11 RF Reset (Command = 0x20, \_DE\_RF\_RESET)

This command reset RF field during the delay time.

■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1]
0x0003		0x20	Delay[0]	Delay[1]

- Data[0]..Data[1] : reset duration of RF field in millisecond. [LSB First]

■ Response frame

LEN-H	LEN-L	Resp
0x0001		OK(0x00) , Err(pls refer to the response code).

### 3.1.12 Reader Number Write (Command = 0x7A, \_DE\_RW\_WRITE)

This command writes reader's unique number to flash memory of the reader. This number can be used to distinguish readers in PC/SC mode or vendor mode.

■ Command frame

LEN-H	LEN-L	CMD	Data[0]
0x0002		0x7A	data

- Data[0] : 1 byte reader number

■ Response frame

LEN-H	LEN-L	Resp	Data[0]
0x0002		OK	data

- Response : OK(0x00), Err(pls refer to the response code).

### 3.1.13 Reader Number Read (Command = 0x7B, \_DE\_RW\_READ)

This command reads reader's unique number from reader's flash memory. This number can be used to distinguish readers in PC/SC mode or vendor mode.

#### ■ Command frame

LEN-H	LEN-L	CMD
0x0002		0x7B

#### ■ Response frame

LEN-H	LEN-L	Resp	Data[0]
0x0002		OK	data

- Data[0] : 1 byte reader number

### 3.1.14 LCD Commands(Command = 0x7C, \_DE\_RW\_LCD, optional)

This command displays message or image to LCD. It displays message until it meets null (0x00) character when data[0] means message. It displays full screen or full line when data[0] means logo or image message.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1]	Data[2..]
0x0003 ~		0x7C	Position(0~63 or 95) 0x80+Position IMG Line(0xC0 + line) Full Logo(0xFF) Clear LCD(0xF0) Big Size Message(0xFD)	MSG[0] MSG[0] Toggle IMG[0] Position	MSG[1..] MSG[1..] IMG[0..] IMG[1..] MSG[0..]

- Position : 0~63(DE-630/DE-631), 0~95(DE-632)

0x80 means toggle black and white

0: First character of the first line

0x80: First character of the first line with toggled color

**<DE-630/DE-631>**

0x10(16): First character of the second line

0xA0(0x80+32) : First character of the third line with toggled color

**<DE-632>**

0x0C(12): First character of the second line

0x98(0x80+24) : First character of the third line with toggled color

- IMG Line : Line number to display message image.

- 0~3 for DE-630 and DE-631
- 0~5 for DE-632 horizontal type
- 0~7 for DE-632 vertical type
- MSG : Message String (ASCII for string )
  - When display “ DUALi ” at the third line,
  - Position is 0x20(32) and
  - MSG[...] is 0x20204455414C69202020202020202000
- Toggle : means toggle black and white
- IMG : Image data from DUALi's image tool application (MakeIMGMessage.exe or MakeNFCPADLOGO.exe)
  - Logo image size : 128x8 bytes for DE-630( and DE-631)
  - Logo image size : 128x12 bytes for DE-632
  - Message image size : 256 bytes for DE-630/631 and DE-632 horizontal type
  - Message image size : 192 bytes for DE-632 vertical type

#### ■ Response frame

LEN-H	LEN-L	Resp
0x0001		OK

### 3.1.15 RTC Write/Read/Display (Command = 0x7D, \_DE\_RW\_ RTC, optional)

This command reads or writes reader's Real Time Clock.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1]	Data[2..9]
0x0002		0x7D	Set(0x00) Read(0x01) Display(0x02)	BCD/INT  Line	Data and Time[8]

- Data[0] : 0 – Write, 1- Read, 2 - Display
- Data[1]: BCD/INT option, 0-integer, 1-BCD, BCD is recommended
  - Line – Line number to display time(0~7), 0xFF means stop display time
- Data[2..9]: when BCD(1), Data[2] - year high(century), Data[3] - year low, Data[4]-month, Data[5]-date, Data[6]-hour, Data[7]-minute, Data[8]-Second, Data[9]-day of week  
e.g: 0x20, 0x12, 0x01,0x31, 0x17,0x31,0x58, 0x02 : 2012-01-31 PM 05:31:58, Tuesday

#### ■ Response frame

LEN-H	LEN-L	Resp	Data[0..6]
0x0002		OK	Date and Time

Response data is integer format.

Data[0..1]-Year, Data[2]-month, Data[3]-date, Data[4]-hour, Data[5]-minute, Data[6]-Second

e.g: 0x07DB090B040413 : 0x07DB=2011, 0x09=September, 0x0B=11<sup>th</sup>, 04:04:19

### 3.1.16 Reader Information (Command = 0x09, \_DE\_INF\_GET)

This command reads reader's unique ID generated internally or memory size. The reader ID can't be modified.

■ Command frame

LEN-H	LEN-L	CMD	Data[0]
0x0002		0x09	Option

- Data[0] : 0 – Reader Unique ID, 1- Flash(Program) Memory Size

■ Response frame

LEN-H	LEN-L	Resp	Data[0..n]
0x000D or 0x0003		OK	UID[12] or LEN[2]

UID : 12 bytes unique ID string(ASCII)

LEN : 2 byte flash memory size, MSB first, {0x00, 0x80}=128KBytes, {0x00,0x40}=64KBytes

### 3.1.17 Get UID (Command = 0x4D, \_DE\_UID\_GET)

This command reads card's unique ID. This command is only valid after reading card.

■ Command frame

LEN-H	LEN-L	CMD
0x0001		0x4D

■ Response frame

LEN-H	LEN-L	Resp	Data[0..n]
0x0005 ~0x000B		OK	UID[4~10]

UID : 4, 7, 8 or 10 bytes unique ID of card.

### 3.1.18 Set UID (Command = 0x4E, \_DE\_UID\_SET)

This command sets card's unique ID. This command must be executed when Select Level (Command = 0x3F) was used for selection. Authentication will be failed if card's UID was not set using this function.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0..n]
0x05,0x08, 0x09,0x0B		0x4E	UID[4~10]

UID : 4, 7, 8 or 10 bytes unique ID of card.

#### ■ Response frame

LEN-H	LEN-L	Resp
0x0001		OK

### 3.1.19 Set RF Para (Command = 0x1D, \_DE\_PARAM\_SET)

This command enables users to set CID, NAD, FWI and FSDI. Users can use RF APDU (Command = 0x61, \_DE\_APDU) command after set these data. This data must be set every time when reader is booted.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1..]
0x0004		0x1D	0x20	Data[1]=CID usage(0-not use, 1-use), Data[2]=CID FWI, refer to <Table1> at 3.2.31 FSDI(0x08=256byte, 0x0A=1024byte)
0x0003			0x21	
0x0003			0x22	

#### ■ Response frame

LEN-H	LEN-L	Resp	Data[0..n]
0x0001		OK	

### 3.1.20 Select RF Antenna (Command = 0x69, DE-AFCMI only)

This command changes active antenna. This command is only for DE-AFCMI.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1..]
0x0003		0x69	0x00	Internal Antenna External Antenna
			0x01	

#### ■ Response frame

LEN-H	LEN-L	Resp
0x0001		OK

## 3.2 Mifare and A-Type Card Functions

### 3.2.1 Idle Request (Command = 0x21, \_DEA\_IDLE\_REQ)

This command sends the REQA command to probe the field for cards of Type A.

■ Command frame

LEN-H	LEN-L	CMD
0x0001		0x21

■ Response frame if response is OK

LEN-H	LEN-L	Resp	Data[0]	Data[1]
0x0003		OK	ATQA[0]	ATQA[1]

- Data[0] : ATQA (lower byte)
- Data[1] : ATQA (higher byte)  
e.g. : Mifare returns 0x00, 0x04.
- Response : OK(0x00) Err(pls refer to the response code).

### 3.2.2 Wakeup Request (Command = 0x22, \_DEA\_WAKEUP\_REQ)

This command sends the WUPA command to probe the field for cards of Type A. Particularly the WUPA Command is sent by the PCD to put PICCs which have entered the HALT State back into the READY\* State. The WUPA function is necessary at the beginning of a new selection of a card.

■ Command frame

LEN-H	LEN-L	CMD
0x0001		0x22

■ Response frame if response is OK

LEN-H	LEN-L	Resp	Data[0]	Data[1]
0x0003		OK	ATQA[0]	ATQA[1]

- Data[0] : ATQA (lower byte)
- Data[1] : ATQA (higher byte)
- Response: OK(0x00) Err(pls refer to the response code).



### 3.2.3 Anti-Collision (Command = 0x23, \_DEA\_ANTICOLL)

This command starts the anti-collision loop and returns UID of a card in the operating field of the antenna. It must be called after REQA or WUPA. This command supports cascade level 1.

■ Command frame

LEN-H	LEN-L	CMD
0x0001		0x23

■ Response frame if response is OK

LEN-H	LEN-L	Resp	Data[0]	Data[1]	Data[2]	Data[3]
0x0005		OK	UID[0]	UID[1]	UID[2]	UID[3]

- Data[0...3] : UID[0] is lowest byte and UID[3] is highest byte
- Response: OK(0x00) Err(pls refer to the response code).

### 3.2.4 Select (Command = 0x24, \_DEA\_SELECT)

This command selects the card with UID of card. It returns SAK.

■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1]	Data[2]	Data[3]
0x0005		0x24	UID[0]	UID[1]	UID[2]	UID[3]

- Data[0...3]: Card UID[0] ..... UID[3]

■ Response frame

LEN-H	LEN-L	Resp	Data[0]
0x0002		OK	SAK

- Data[0]: SAK, Select acknowledge, The expected SAK for Mifare is 0x08, 0x28.
- Response: OK(0x00) Err(pls refer to the response code).

### 3.2.5 Authentication Mifare (Command = 0x25, \_DEA\_AUTH)

This command starts the authentication sequence. \_DEA\_LOADKEY command (0x2F) should be executed before this command, because keys shall be stored into internal SRAM of reader. We recommend using \_DEA\_AUTHKEY command (0x30) rather than this command.

■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1]	Data[2]
0x0004		0x25	Mode	KeyNo	BlockNo

- Data[0] :

Data[0]	Mode
0x00	Authentication with key A.
0x04	Authentication with key B.

- Data[1]: Key number (0-15) in which key is stored.
- Data[2]: Block number of the card.

#### ■ Response frame

LEN-H	LEN-L	Resp
0x0001		OK(0x00) , Err(pls refer to the response code).

### 3.2.6 Halt (Command = 0x26, \_DEA\_HALT)

This command makes selected card into HALT State.

#### ■ Command frame

LEN-H	LEN-L	CMD
0x0001		0x26

#### ■ Response frame

LEN-H	LEN-L	Resp
0x0001		OK(0x00) , Err(pls refer to the response code).

### 3.2.7 Read Mifare (Command = 0x27, \_DEA\_READ)

This command reads one block(16 bytes) from the authenticated block of the selected card.

#### ■ Command frame

LEN-L	LEN-L	CMD	Data[0]
0x0002		0x27	BlockNo

- Data[0]: Block number.

#### ■ Response frame

LEN-L	LEN-L	Resp	Data[0 ... 15]
-------	-------	------	----------------

0x0011	OK	Bdata[0] ... Bdata[15]
--------	----	------------------------

- Response: OK(0x00) , Err(pls refer to the response code).
- Data[0 ... 15]: 16 byte data from selected block .

### 3.2.8 Write Mifare (Command = 0x28, \_DEA\_WRITE)

This command writes one block(16 bytes) to the authenticated block of the selected card.

#### ■ Command frame

LEN-L	LEN-L	CMD	Data[0]	Data[1 ... 16]
0x0012		0x28	BlockNo	Bdata[0] ... Bdata[15]

- Data[0] : Block number of the card (0 block is not available) .
- Data[1... 16] : Data to be written into the selected block.

#### ■ Response frame

LEN-H	LEN-L	Resp
0x0001		OK(0x00) , Err(pls refer to the response code).

### 3.2.9 Increment Mifare (Command = 0x29, \_DEA\_INCREMENT)

This command reads the accessed value-block, checks the data structure, increases the contents of the value block by the transmitted value (using the 31 least significant bits, not all 32 bits) and stores the result in the card's internal register.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]
0x0006		0x29	BlockNo	Value[0]	Value[1]	Value[2]	Value[3]

- Data[0]: Block number of the card.
- Data[1...4]: The increasing value. This variable is interpreted as an unsigned long integer(4bytes)  
e.g. : if 1, 0x01 0x00 0x00 0x00  
if 255, 0xFF 0x00 0x00 0x00

#### ■ Response frame

LEN-H	LEN-L	Resp
0x0001		OK(0x00) , Err(pls refer to the response code).

### 3.2.10 Decrement Mifare (Command = 0x2A, \_DEA\_DECREMENT)

This command reads the accessed value block, checks the data structure, decreases the contents of the value block by the transmitted value (using the 31 least significant bits, not all 32 bits) and stores the result in the card's internal register.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]
0x0006		0x2A	BlockNo	Value[0]	Value[1]	Value[2]	Value[3]

- Data[0] : Block number of the card.
  - Data[1...4]: The decreasing value. This variable is interpreted as an unsigned long integer (4bytes).
- e.g. : if 1, 0x01 0x00 0x00 0x00  
if 255, 0xFF 0x00 0x00 0x00

#### ■ Response frame

LEN-H	LEN-L	Resp
0x0001		OK(0x00) , Err(pls refer to the response code).

### 3.2.11 Increment and Transfer (Command = 0x2B, \_DEA\_INC\_TRANS)

This command reads the accessed value block, checks the data structure, increases the contents of the value block by the transmitted value (using the 31 least significant bits, not all 32 bits) and stores the result in the card's selected block([Trans BN] of frame format).

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1...4]	Data[5]
0x0007		0x2B	BlockNo	Value[0]~Value[4]	Trans BN

- Data[0] : Block number of the card.
  - Data[1...4]: The increasing value. This variable is interpreted as an unsigned long integer (4bytes)
- e.g. : if 1, 0x01 0x00 0x00 0x00  
if 255, 0xFF 0x00 0x00 0x00
- Data[5] : Block number of card which the increased value is going to transfer

#### ■ Response frame

LEN-H	LEN-L	Resp
-------	-------	------

0x0001	OK(0x00) , Err(pls refer to the response code).
--------	---

### 3.2.12 Decrement and Transfer (Command = 0x2C, \_DEA\_DEC\_TRANS)

This command reads the accessed value block, checks the data structure, decreases the contents of the value block by the transmitted value (using the 31 least significant bits, not all 32 bits) and stores the result in the card's selected block([Trans BN] of frame format).

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1...4]	Data[5]
0x0007		0x2C	BlockNo	Value[0]~Value[4]	Trans BN

- Data[0]: Block number of the card.
- Data[1...4] : The decreasing value. This variable is interpreted as an unsigned long integer(4bytes)  
e.g. : if 1, 0x01 0x00 0x00 0x00  
if 255, 0xFF 0x00 0x00 0x00
- Data[5] : Block number of card which the decreased value is going to transfer

#### ■ Response frame

LEN-H	LEN-L	Resp
0x0001		OK(0x00) , Err(pls refer to the response code).

### 3.2.13 Restore Mifare (Command = 0x2D, \_DEA\_RESTORE)

This command reads the accessed value block, checks the data structure and stores the result in the card's internal register. Value is stored in volatile memory only. In order to store this value into value block, \_DEA\_TRANSFER command should be executed after this command.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]
0x0002		0x2D	BlockNo

- Data[0]: Block number of the card .

#### ■ Response frame

LEN-H	LEN-L	Resp
0x0001		OK(0x00) , Err(pls refer to the response code).

### 3.2.14 Transfer Mifare (Command = 0x2E, \_DEA\_TRANSFER)

This command transfers the contents of the internal register of card to the selected address. The sector must be authenticated for this operation. The transfer function can be called directly after Increment, Decrement or Restore!

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]
0x0002		0x2E	BlockNo

- Data[0]: Block number of the card.

#### ■ Response frame

LEN-H	LEN-L	Resp
0x0001		OK(0x00) , Err(pls refer to the response code).

### 3.2.15 Load Key Mifare (Command = 0x2F, \_DEA\_LOADKEY)

This command writes a master key into internal memory(FLASH) of reader and puts into the master key buffer. Reader can store 16 keys in internal nonvolatile or volatile memory for authentication procedure of accessed card. Although power is off, keys saved at nonvolatile memory (0~15) are not lost. This command is used for authentication procedure.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1]	Data[2] ... Data[7]
0x0008		0x2F	Mode	KeyNo	Key[0] ... Key[5]

- Data[0]: Mode

Data[0]	Load Key Mode
0x00	Loading Key A
0x04	Loading Key B

- Data[1]: The key number or index for authentication.  
0~15(0x0F) : save to nonvolatile memory  
16(0x10) : save to volatile memory
- Data[2...7]: Key[0]..[5], The Key data to be stored into internal memory of Reader.

#### ■ Response : OK(0x00), Err(pls refer to the response code).

### 3.2.16 Auth Key Mifare (Command = 0x30, \_DEA\_AUTHKEY)

This command writes a secret key directly into the secret key buffer and authenticates selected sector with submitted key.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1 ... 6]	Data[7]
0x0008		0x30	Mode	Key[0] ... Key[5]	Block No

- Data[0]: Mode

Data[0]	Load Key Mode
0x00	Loading Key A
0x04	Loading Key B

- Data[1...6]: Key[0]..[5], The key data to be stored into the secret key buffer.
- Data[7]: The block number for Authentication

#### ■ Response frame

LEN-H	LEN-L	Resp
0x0001		OK(0x00) , Err(pls refer to the response code).

### 3.2.17 Request to Auth (Command = 0x31, \_DEA\_REQ\_ANTI\_AUTH)

This command performs Request, Anti-collision, Authentication operations at once and returns the selected card's UID. This command use internal key data stored into internal memory. Before using this command, the \_DEA\_LOADKEY(0x2F) command should be executed.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1]	Data[2]	Data[3]
0x0005		0x31	R-mode	A-Mode	Key No	BlockNo

- Data[0] : Request Mode

Data[0]	Request Mode
0x00	Request Idle
0x01	Request All, Wakeup Request

- Data[1] : Authentication Mode

Data[1]	Authentication Mode
0x00	Authentication with Key A
0x04	Authentication with Key B

- Data[2]: The key number or index(0~15) for authentication.
- Data[3]: Block Number of the card

#### ■ Response frame

LEN-H	LEN-L	Resp	Data[0]	Data[1]	Data[2]	Data[3]
0x0005		OK	UID[0]	UID[1]	UID[2]	UID[3]

- Data[0...3]: UID of card.
- Response: OK(0x00), Err(pls refer to the response code).

### 3.2.18 Request to Auth Key (Command = 0x32, \_DEA\_REQ\_ANTI\_AUTHKEY)

This command performs Request, Anti-collision, Authentication operations with Key at once and returns the selected card UID.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1]	Data[2]	Data[3...8]
0x000A		0x32	R-Mode	A-Mode	BlockNo	Key[0..5]

- Data[0] : Request Mode

Data[0]	Request Mode
0x00	Request Idle
0x01	Request All, Wakeup Request

- Data[1] : Authentication Mode

Data[1]	Authentication Mode
0x00	Authentication with Key A
0x04	Authentication with Key B

- Data[2] : Block Number (0 ~ 63) of the card
- Data[3...8] : Key data for authentication of selected block

#### ■ Response frame

LEN-H	LEN-L	Resp	Data[0]	Data[1]	Data[2]	Data[3]
0x0005		OK	UID[0]	UID[1]	UID[2]	UID[3]

- Data[0...3] : UID of card.
- response : OK(0x00), Err(pls refer to the response code).

### 3.2.19 Increment and Transfer2 (Command = 0x33, \_DEA\_INC\_TRANS2)

This command reads the accessed value block, checks the data structure, increases the contents of the value block by the transmitted value (using the 31 least significant bits, not all 32 bits), and stores the result in the card's selected block and verify correct increment.



#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1...4]	Data[5]
0x0007		0x33	BlockNo	Value[0]~Value[3]	Trans BN

- Data[0]: Block number of the card.
- Data[1...4] : The increasing value. This variable is interpreted as an unsigned long integer (4bytes)  
e.g. : if 1, 0x01 0x00 0x00 0x00  
if 255, 0xFF 0x00 0x00 0x00
- Data[5]: Block number of card which the increased value is going to transfer

#### ■ Response frame

LEN-H	LEN-L	Resp
0x0001		OK(0x00) , Err(pls refer to the response code).

### 3.2.20 Decrement and Transfer2 (Command = 0x34, \_DEA\_DEC\_TRANS2)

This command reads the accessed value block, checks the data structure and shortage of balance, decreases the contents of the value block by the transmitted value (using the 31 least significant bits, not all 32 bits) and stores the result in the card's selected block and verify correct decrement

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1...4]	Data[5]
0x0007		0x34	BlockNo	Value[0]~Value[4]	Trans BN

- Data[0]: The address on the card from which the value block is read.
- Data[1...4]: The decreasing value. This variable is interpreted as an unsigned long integer (4bytes) with the lower byte located at the lower address  
e.g. : if 1, 0x01 0x00 0x00 0x00  
if 255, 0xFF 0x00 0x00 0x00
- Data[5]: Block number of card which the decreased value is going to transfer

#### ■ Response frame

LEN-H	LEN-L	Resp
0x0001		OK(0x00) , Err(pls refer to the response code).

### 3.2.21 Request to Read A (Command = 0x35, \_DEA\_REQ\_ANTI\_AUTH\_READ)

This command performs Request, Anti-collision, Authentication, read operations at once and returns the selected card UID, block data. This command use internal key data stored into internal memory. Before using this command, the \_DEA\_LOADKEY(0x2F) command should be executed.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1]	Data[2]	Data[3]
0x0005		0x35	R-mode	A-Mode	Key No	BlockNo

- Data[0] : Request Mode

Data[0]	Request Mode
0x00	Request Idle
0x01	Request All, Wakeup Request

- Data[1] : Authentication Mode

Data[1]	Authentication Mode
0x00	Authentication with Key A
0x04	Authentication with Key B

- Data[2] : The key number or index(0~15) for authentication
- Data[3] : Block Numbers of the card.

#### ■ Response frame

LEN-H	LEN-L	Resp	Data[0...3]	Data[4...19]
0x0015		OK	UID[0...3]	BlockData[0...15]

- Data[0...3] : Card serial number of accessed card.
- Data[4...19] : Data to be reading from selected block
- Response: OK(0x00), Err(pls refer to the response code).

### 3.2.22 Request to Read K (Command = 0x36, \_DEA\_REQ\_ANTI\_AUTHKEY\_READ)

This command performs Request, Anti-collision, Authentication, read operations with Key at once and returns the selected card UID and block data of accessed block.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1]	Data[2]	Data[3..8]
0x000A		0x36	R-Mode	A-Mode	BlockNo	Key[0..5]

- Data[0]: Request Mode

Data[0]	Request Mode
---------	--------------

0x00	Request Idle
0x01	Request All, Wakeup Request

- Data[1]: Authentication Mode

Data[1]	Authentication Mode
0x00	Authentication with Key A
0x04	Authentication with Key B

- Data[2] : Block Number of the card
- Data[3...8] : External key data for authentication of selected block

#### Response frame

LEN-H	LEN-L	Resp	Data[0...3]	Data[4...19]
0x0015		OK	UID[0...3]	BlockData[0...15]

- Data[0...3] : Card serial number of accessed card.
- Data[4...19] : Data to be reading from selected block
- Response: OK(0x00), Err(pls refer to the response code).

### 3.2.23 Request to Read Multi (Command = 0x42, \_DEA\_REQ\_AUTHKEY\_READ\_M)

This command performs Request, Anti-collision, Authentication, read operations with Key at once and returns data of accessed blocks. Key of all sectors must be same.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1..6]	Data[7]	Data[8]	Data[9] (Option)
0x000A or 0x000B		0x42	A-Mode	Key[0..5]	BlockNo	Block Count	0: No UID 1:UID add

- Data[0]: Authentication Mode

Data[0]	Authentication Mode
0x00	Authentication with Key A
0x04	Authentication with Key B

- Data[1...6] : External key data for authentication of selected block
- Data[7] : Block Number of the card
- Data[8] : Block Count of the card.
- Data[9] : Optional byte, SAK and UID option. Add Data[9] (= 0x01) to get SAK and UID with Block data.

Response frame

(SAK, UID absent)

LEN-H	LEN-L	Resp	Data[0...]
0x00xx		OK	BlockData[0...]

(SAK, UID exist, supports from 25-APR-2018)

LEN-H	LEN-L	Resp	Data[0...]
0x00xx		OK	SAK[1]+UID[10]+BlockData[0...]

- UID[10] : Bytes after effective UID is filled with 0x00.
- Data[0... ] : Data to be reading from selected block  
Block Data, Length = Block Size \* Block Length(16)
- Response: OK(0x00), Err(pls refer to the response code).

### 3.2.24 Request to Write Multi (Command = 0x43, \_DEA\_REQ\_AUTHKEY\_WRITE\_M)

This command performs Request, Anti-collision, Authentication, write operations with Key and Data at selected blocks. Key of all sectors must be same.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1..6]	Data[7]	Data[8]	Data[9..]
0x000A + Block Count*16		0x43	A-Mode	Key[0..5]	BlockNo	Block Count	Block Data (Block Count*16)

- Data[0]: Authentication Mode

Data[0]	Authentication Mode
0x00	Authentication with Key A
0x04	Authentication with Key B

- Data[1...6] : External key data for authentication of selected block
- Data[7] : Block Number of the card
- Data[8] : Block Count of the card.
- Data[9..] : Data to write to Blocks.

Response frame

LEN-H	LEN-L	Resp
0x0001		OK

- Response: OK(0x00), Err(pls refer to the response code).

### 3.2.25 Request to Write A (Command = 0x37, \_DEA\_REQ\_ANTI\_AUTH\_WRITE)

This command performs Request, Anti-collision, Authentication, Write operations at once and returns the selected card UID. This command use internal key data stored into internal memory. Before using this command, the \_DEA\_LOADKEY(0x2F) command should be executed.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1]	Data[2]	Data[3]	Data[4...19]
0x00	15	0x37	R-mode	A-Mode	Key No	BlockNo	BlockData[0...15]

- Data[0] : Request Mode

Data[0]	Request Mode
0x00	Request Idle
0x01	Request All, Wakeup Request

- Data[1] : Authentication Mode

Data[1]	Authentication Mode
0x00	Authentication with Key A
0x04	Authentication with Key B

- Data[2]: The key number or index(0~15) for authentication
- Data[3] : Block Number of the card
- Data[4...19]: Data to be written into the selected block

#### ■ Response frame

LEN-H	LEN-L	Resp	Data[0...3]
0x00	05	OK	UID[0...3]

- Data[0...3] : Card UID of selected card
- Response: OK(0x00), Err(pls refer to the response code).

### 3.2.26 Request to Write K (Command = 0x38, \_DEA\_REQ\_ANTI\_AUTHKEY\_WRITE)

This command performs Request, Anti-collision, Authentication, Write operations at once and returns the selected card UID.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1]	Data[2]	Data [3...8]	Data[9~24]
-------	-------	-----	---------	---------	---------	--------------	------------

0x001A	0x38	R-mode	A-Mode	BlockNo	Key [0...5]	BlockData[0...15]
--------	------	--------	--------	---------	-------------	-------------------

- Data[0]: Request Mode

Data[0]	Request Mode
0x00	Request Idle
0x01	Request All, Wakeup Request

- Data[1] : Authentication Mode

Data[1]	Authentication Mode
0x00	Authentication with Key A
0x04	Authentication with Key B

- Data[2] : Block Number of the accessed card
- Data[3...8] : External key data(6 byte) for authentication of selected block
- Data[9...24]: Data to be written into the selected block

#### ■ Response frame

LEN-H	LEN-L	Resp	Data[0...3]
0x0005		OK	UID[0...3]

- Data[0...3] : Card UID of selected card
- Response: OK(0x00), Err(pls refer to the response code).

### 3.2.27 Request to Select (Command = 0x39, \_DEA\_REQ\_ANTI\_SEL)

This command performs Request, Anti-collision, Select operations at once and returns the SAK and UID of selected card. It supports cascade level up to 3.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]
0x0002		0x39	RMode

- Data[0]: Request Mode

Data[0]	Request Mode
0x00	Request Idle
0x01	Request All, Wakeup Request

#### ■ Response frame

LEN-H	LEN-L	Resp	Data[0...]
0x0005		OK	SAK[1]+UID[0...]

- SAK: 1Byte SAK
- Data[0...]: 4,7 or 10 Bytes UID of selected card

- Response: OK(0x00), Err(pls refer to the response code).

### 3.2.28 Write Ultra Light (Command = 0x3B, \_DEA\_UWRITE)

This command writes 4 bytes to a selected address of Mifare ultra light card.

#### ■ Command frame

LEN-L	LEN-L	CMD	Data[0]	Data[1...4]
0x0006		0x3B	Address	Bdata[0] ... Bdata[3]

- Data[0]: address of Mifare ultra light card.
- Data[1...4] : data(4byte) to be written into the selected address.

#### ■ Response frame

LEN-H	LEN-L	Resp
0x0001		OK(0x00) , Err(pls refer to the response code).

### 3.2.29 Full Anti-Collision and Select (Command = 0x3D, \_DEA\_ANTI\_SEL\_LEVEL)

This command starts the anti-collision and select sequence in cascade mode to get complete UID when UID size is more than single size in type A. Request command shall be executed before this command.

#### ■ Command frame

LEN-H	LEN-L	CMD
0x0001		0x3D

#### ■ Response frame if response is OK

LEN-H	LEN-L	Resp	Data[0]	Data[1...N-1]
N+1		OK	SAK	UID[0...N-2]

- Data[0] : SAK, Select Acknowledge
- Data[1...N-1] Card UID, The length of UID is 4(Single size) or 7(Double size) or 10(Triple size).
- Response: OK(0x00), Err(pls refer to the response code).

Ex)	Commands	Responses
	REQA : 0xA1 0x00 0x07 0x26	ATQA : 0x04 0x00
	ANTI_SEL : 0x3D	SAK + UID : 0x08 0xB2 0xFA 0x26 0x1F 0x71

### 3.2.30 Anti-Collision Level (Command = 0x3E, \_DEA\_ANTICOLL\_LEVEL)

This command performs anti-collision sequence defined by each cascade level in bit mode. It will be used to know the UID data (4-byte serial number + 1-byte LRC) during anti-collision function at each level. It returns just 5-byte UID of defined level in case of no collision. Select sequence shall be performed using \_DEA\_TRANSPARENT command containing SEL\_CODE data and 5-byte UID after this command. Please refer to ISO 14443 Part 3 specification to know the detailed anti-collision sequence in cascade mode.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1]	Data[2..5]
0x0007		0x3E	Cascade Level	Bit count	Valid UID

- Data[0]: Cascade Level

Data[0]	Cascade Level
0x93	Cascade Level 1 (Equal or More than single UID)
0x95	Cascade Level 2 (Equal or More than Double UID)
0x97	Cascade Level 3 (Triple UID)

- Data[1] : Bit count(Valid UID bit count, 0~32)
- Data[2..5] : N-bytes valid UID

#### ■ Response frame if response is OK

LEN-H	LEN-L	Resp	Data[0..3]
0x0005		OK	UID[0..3] or CT + UID[0..2]

- Data[0...3]: CT is cascade tag, UID is Card UID.
- Response: OK(0x00), Err(pls refer to the response code).

Ex)	Commands	Responses
REQA	: 0x21	ATQA : 0x04 0x00
Level 1 ANTI	: 0x3E 0x93 0x00	UID : 0xB2 0xFA 0x26 0x1F
Level 1 SEL	: 0x3F 0x93 0xB2 0xFA 0x26 0x1F	SAK : 0x08

### 3.2.31 Select Level (Command = 0x3F, \_DEA\_SELECT\_LEVEL)

This command starts the select sequence. It will be used when UID size is more than single to get complete UID. **UID must be loaded using 'Set UID (Command = 0x4D)' when select card.** Authentication will be failed if UID was not set. (Not implemented in ISO15693 version)

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1..4]
-------	-------	-----	---------	------------



0x0005	0x3F	Cascade Level	UID
--------	------	---------------	-----

- Data[0]( Cascade Level) :0x93 = Cascade Level 1 (Equal or More than single UID)  
0x95 = Cascade Level 2 (Equal or More than Double UID)  
0x97 = Cascade Level 3 (Triple UID)
- Data[1..4] : UID

#### ■ Response frame if response is OK

LEN-H	LEN-L	Resp	Data[0]
0x0005		OK	SAK

- Data[0] : SAK, Select Acknowledge
- Response: OK(0x00), Err(pls refer to the response code).

Ex)	Commands	Responses
	REQA : 0x21	ATQA : 0x44 0x00
	Level 1 ANTI : 0x3E 0x93 0x00 0x00 0x00 0x00 0x00	UID : 0x88 0xFA 0x26 0x1F
	Level 1 SEL : 0x3F 0x93 0x88 0xFA 0x26 0x1F	SAK : 0x0C
	Level 2 ANTI : 0x3E 0x95 0x00 0x88 0xFA 0x26 0x1F	UID : 0x00 0x11 0x22 0x33
	Level 1 SEL : 0x3F 0x95 0x00 0x11 0x22 0x33	SAK : 0x08

### 3.2.32 Device Information (Command = 0x40, \_DEA\_DEVINFO)

This command returns Product Type Identification and Product Serial Number of reader.

#### ■ Command frame

LEN-H	LEN-L	CMD
0x01		0x40

#### ■ Response frame if response is OK

LEN-H	LEN-L	Resp	Data[0...N-1]
N+1		OK	PSNR[0...N-1]

- Data [0...N-1] : Product Serial Number (In case of DE-620, 12 byte)
- Response: OK(0x00), Err(pls refer to the response code).

### 3.2.33 Type-A Transparent (Command = 0x41, \_DEA\_TRANSPARENT)

This command is used to send data with computed CRC from a host to the card. Host doesn't need to compute 2-byte CRC. Reader extracts 2-byte CRC data from card response data.

#### ■ Command Frame

LEN-H	LEN-L	CMD	Data[0...N-2]	Data[N-1]
N+1		0x41	APDU	TOUT

- Data[0...N-2] : APDU to be send to card. Reader computes CRC and attaches it to APDU.
- Data[N-1]: TOUT, Timeout value - reader waits for card response.

**Table 1 < Time Out Table >**

TOUT Value in hexadecimal	Time Value	FWI
0x02	1.208 msec	2
0x03	2.416 msec	3
0x05	4.832 msec	4
0x0A	9.664 msec	5
0x14	19.32 msec	6
0x27	38.66 msec	7
0x4E	77.3 msec	8
0x9B	154.6 msec	9
0xB0 ~	309.3 msec	10
0xC0 ~	618.6 msec	11
0xD0 ~	1.2371 sec	12
0xE0 ~	2.4742 sec	13
0xF0 ~	4.9485 sec	14
0xFF	9.897 sec	15

#### ■ Response frame

LEN-H	LEN-L	Resp	Data[0...N-1]
N+1		OK	APDU

- Data[0...N-1] : APDU received from Card except 2-byte CRC.
- Response: OK(0x00), Err(pls refer to the response code).

### 3.2.34 Type-A Transparent2 (Command = 0x47, \_DEA\_TRANSPARENT2)

This command is used to bypasses data from a host to the card. Host must compute 2-byte CRC data and send computed 2-byte CRC data with APDU to RF module. Reader receives response data from card and bypass received data to the Host.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0 ... N-2]	Data[N-1]
N+1		0x47	APDU+CRC	TOUT

- Data[0...N-2] : APDU to be send to Card. A HOST computes CRC and should attach it to APDU.
- Data[N-1]: TOUT, Timeout value - Reader waits for card response.

Refer to Table1 at page 30.

#### ■ Response frame

LEN-H	LEN-L	Resp	Data[0 .. N-1]
N+1		OK	APDU+CRC

- Data[0...N-1] : APDU received from Card
- Response: OK(0x00)

### 3.2.35 NFC T1 Tag Transparent (Command = 0x6B)

This command is used to send data with computed CRC to the NFC T1 Tag(TOPAZ). Host doesn't need to compute 2-byte CRC. Reader extracts 2-byte CRC data from card response data.

#### ■ Command Frame

LEN-H	LEN-L	CMD	Data[0...N-2]	Data[N-1]
N+1		0x6B	CMD+DATA	TOUT

- Data[0...N-2] : APDU to be send to card. Reader computes CRC and attaches it to APDU.
- Data[N-1]: TOUT, Timeout value - reader waits for card response.

### 3.2.36 RATS and PPS (Command = 0x4B, \_DEA\_RATS\_PPS)

This command performs RATS and PPS (when it is needed) command to card. This command must be executed to communicate with A-type card (except MIFARE) when card is detected with 'RF Find Card' and its option byte is 0x40.

#### ■ Command frame

LEN-H	LEN-L	CMD
0x0002		0x4B

#### ■ Response frame

LEN-H	LEN-L	Resp	Data[0...]
0x000x		OK	ATS

- Response: OK(0x00), Err(pls refer to the response code).



### 3.3 FeliCa Card Functions

#### 3.3.1 Type-C Transparent (Command = 0x50, \_DEC\_TRANSPARENT)

This command is used to send data with computed CRC from a host to the card. The host doesn't need to compute 2-byte CRC. Reader extracts 2-byte CRC data from card response data.

##### ■ Command frame

LEN-H	LEN-L	CMD	Data[0...N-2]	Data[N-1]
N+1		0x50	APDU	TOUT

- Data[0...N-2] : APDU to be send to card. Reader computes CRC and attaches it to APDU.
- Data[N-1]: TOUT, Timeout value - Reader waits for card response.

Refer to Table1 at page 30

##### ■ Response frame

LEN-H	LEN-L	Resp	Data[0...N-1]
N+1		OK	APDU

- Data[0...N-1] : APDU received from Card except 2-byte CRC.
- Response: OK(0x00), Err(pls refer to the response code).

#### 3.3.2 Type-C Polling (Command=0x51, \_DEC\_POLLING\_NOENC)

This command is used to send Polling command to FeliCa card without data encryption.

##### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]...[3]	Data[4]
0x0006		0x51	Data	TOUT

- Data[0]...[1] : system code. 0xffff indicate all system code.
- Data[2] : 0x01(system code is requested) Others(system code is not requested)
- Data[3] : Time slot. (0x00, 0x01, 0x03, 0x07, 0x0f)

##### ■ Response frame

LEN-H	LEN-L	Resp	Data[0...N-1]
N+1		OK	Data

- Data[0]...[7] : IDm of FeliCa card.
- Data[8]...[15] : PMm of FeliCa card.
- Data[16]...[17] : System code(when the system code is requested)
- Response: OK(0x00), Err(pls refer to the response code).

### 3.3.3 Type-C Read (Command=0x52, \_DEC\_READ\_NOENC)

This command is used to send Read command to FeliCa card without data encryption.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0...N-2]	Data[N-1]
N+1		0x52	Data	TOUT

- Data[0]...[7] : IDm of FeliCa card.
- Data[8]...[9] : service code(little endian)
- Data[10] : block

#### ■ Response frame

LEN-H	LEN-L	Resp	Data[0...N-1]
N+1		OK	Data

- Data[0...N-1] : block data, Only when the Status flag1 is OK
- Response: OK(0x00), Err(pls refer to the response code).

### 3.3.4 Type-C Write (Command=0x53, \_DEC\_WRITE\_NOENC)

This command is used to send Write command to FeliCa card without data encryption.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0...N-2]	Data[N-1]
N+1		0x53	Data	TOUT

- Data[0]...[7] : IDm of FeliCa card.
- Data[8]...[9] : service code(little endian)
- Data[10] : block
- Data[11]...[26] : block data

#### ■ Response frame

LEN-H	LEN-L	Resp
0x0001		OK

- Response: OK(0x00), Err(pls refer to the response code).

### 3.4 FeliCa Card Batch Functions

#### 3.4.1 FeliCa SAM Authentication (Command = 0x56)

This is an authentication between the FeliCa USER and RC-S251(SAM) based on a 3-way mutual authentication mechanism using shared symmetrical keys. User must submit this symmetrical a 24-byte key to RC-S251. The default CBC is all 0x00s. The method of the authentication is not disclosed and executed inside the reader.

##### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1] ~ Data[24]	Data[25]~[32]
0x0022		0x56	ENC Mode	Default Key[24]	CBC[8]

- ENC Mode: this mode is maintained until finish all transactions.

0: authentication for unencrypted communication

1: authentication for encrypted communication

- Default Key : symmetrical a 24-byte TDES key

- CBC : default CBC is {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}

##### ■ Response frame

LEN-H	LEN-L	Resp	Data[0] ~ [7]	Data[8] ~ [9]
0x000B		OK	IDM[8]	IDt[2]

- IDM[8] : IDM of FeliCa card

- IDt[2]

#### 3.4.2 FeliCa Mutual Authentication (Command=0x57)

This is an authentication between the FeliCa card and RC-S251(SAM). User must submit global key and user key. The method of the mutual authentication is not disclosed and executed inside the reader.

##### ■ Command frame

LEN-H	LEN-L	CMD	Data[0...1]	Data[2 ... 3]	Data[4 ...11]	Data[12 ... 19]
0x0015		0x57	Area Code[2]	Service Code[2]	Global Key[8]	User Key[8]

##### ■ Response frame

LEN-H	LEN-L	Resp
1		OK

### 3.4.3 FeliCa Mutual Authentication RWSAM (Command=0x58)

This is an authentication between the FeliCa card and RC-S251(SAM) using the keys stored in SAM. User doesn't need to submit keys but need to submit key codes and versions. The method of the mutual authentication is not disclosed and executed inside the reader.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0...1]	Data[2...5]	Data[6...9]
0x000B		0x58	System Code[2]	GSK Code and Version[4]	USK Code and Version[4]

#### ■ Response frame

LEN-H	LEN-L	Resp
1		OK

### 3.4.4 FeliCa Command (Command=0x59)

This command is used to exchange commands with the FeliCa card after authentication. IDtr is controlled inside the reader. User must submit all other data. User can read or write each block using this command. The method of the process is not disclosed and executed inside the reader.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0...3]	Data[4...n-1]
N+1		0x59	Header[4]	Input Data[0..k]

- Header[4]: Dispatcher[0], Reserved[2], Command Code[1]
- Input Data: all other data except IDtr

#### ■ Response frame

LEN-H	LEN-L	Resp	Data[0...n]
N+1		OK	Response

- Response: OK(0x00), Err(pls refer to the response code).



### 3.5 B-Type Card Functions

#### 3.5.1 Type-B Transparent (Command = 0x60, \_DEB\_TRANSPARENT)

Reader receives data from Host and sends data with RF Module-computed CRC to the card. Host doesn't need compute 2-byte CRC. Reader extracts 2-byte CRC data from card response data.

##### ■ Command frame

LEN-H	LEN-L	CMD	Data[0...N-2]	Data[N-1]
N+1		0x60	APDU	TOUT

- Data[0...N-2] : APDU to be send to Card. Reader computes CRC and attach it to APDU.
- Data[N-1] : Timeout value - Reader waits for card response.

Refer to Table1 at page 30.

##### ■ Response frame

LEN-H	LEN-L	Resp	Data[0...N-1]
N+1		OK	APDU

- Data[0...N-1] : APDU received from card except 2-byte CRC.
- Response: OK(0x00), Err(pls refer to the response code).

#### 3.5.2 Type-B Transparent2 (Command = 0x6E, \_DEB\_TRANSPARENT2)

Reader receives data from Host and bypasses data to the card. A Host must computes 2-byte CRC data and sends computed 2-byte CRC data with APDU to reader. Reader receives response data from card and bypass received data to the Host.

##### ■ Command frame

LEN-H	LEN-L	CMD	Data[0 ... N-2]	Data[N-1]
N+1		0x6E	APDU+CRC	TOUT

- Data[0...N-2] : APDU to be send to card. A Host computes CRC and attaches it to APDU.
- Data[N-1] : Timeout value - Reader waits for card response.

Refer to Table1 at page 30.

##### ■ Response frame

LEN-H	LEN-L	Resp	Data[0 .. N-1]
N+1		OK	APDU+CRC

- Response: OK(0x00), Err(pls refer to the response code).

### 3.6 A/B-Type Common Functions

#### 3.6.1 RF Find Card (Command = 0x4C, \_DE\_FIND\_CARD)

This command detects type-A and type-B cards in the RF field.

##### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1]	Data[2]	Data[3]
0x0005		0x4C	BRate	CID	NAD	Option

- Data[0]

Data[0]	Max Card Baud Rate Use same speed for TX and RX.
0x00	0x00 = 106 Kbps
0x01	0x01 = 212 Kbps
0x02	0x02 = 424 Kbps
0x03	0x03 = 848 Kbps
<b>0x04</b>	<b>VHBR, 1.6 ~ 6.8Mbps</b>

- Data[1] : CID

- Data[2] : NAD

- Data[3] : Option ('A': only A-type, 'B': only B-type, 'K': NFC Barcode, other: All  
0x40:Do not send RATS when check A-type card – 2016.11.28 applied)

##### ■ Response frame

LEN-H	LEN-L	Resp	Data[0]	Data[1]	Data[2...N-1]
N+1		OK	Type	CSPD	CD[..],UID[0]...UID[3]

- Data[0]

Data[0]	RF Type
'M', 0x4D, 77	Mifare
'A', 0x41, 65	A type
'B', 0x42, 66	B type

- Data[1]: Card communication speed or SAK when option byte is 0x40

- Data[2]: Card Data and Card UID. It is different each card.

- Response: OK(0x00), Err(pls refer to the response code).

##### ■ Response frame for NFC Barcode

LEN-H	LEN-L	Resp	Data[0...N-1]
N+1		OK	16bytes or 32bytes UID

### 3.6.2 RF APDU (Command = 0x61, \_DE\_APDU)

This command transfers APDU command to contactless card. RF Find Card (Command = 0x4C, \_DE\_FIND\_CARD) command must be executed first to use this command. If you don't use \_DE\_FIND\_CARD command, communication with card could be unstable because default communication parameter with card is applied.

#### ■ Command Frame

LEN-H	LEN-L	CMD	Data[0...N-1]
N+1		0x61	APDU and data

- Data[0...N-1] : APDU and data to be send to card

If N is 1, this command deselects card.

#### ■ Response frame

LEN-H	LEN-L	Resp	Data[0..N-1]
N+1		OK	Receive buffer

- Receive buffer: received data from card.

- Response: OK(0x00), Err(pls refer to the response code).

### 3.6.3 RF CARD STATUS (Command = 0x62, \_DE\_EXIST\_CHK)

This command transfers NAK to contactless card to check card existence.

#### ■ Command Frame

LEN-H	LEN-L	CMD
N+1		0x62

#### ■ Response frame

LEN-H	LEN-L	Resp
N+1		OK

- Response: OK(0x00-Card Exists), Others(Does not exist).

### 3.6.4 SPARAMETER Activate (Command = 0x68)

Reader reads s-parameter and applies highest speed among the speeds which card supports. This command must be executed after ATTRIBUTE(Type-B) and RATS(Type-A).

Attribute with speed 106Kbps must be executed for type-B card. You don't need to execute this command if you used 'RF Find Card' (Command, 0x4C) because this command was automatically executed inside 'RF Find Card' command.

- Command frame

LEN-H	LEN-L	CMD
N+1		0x68

- Response frame

LEN-H	LEN-L	Resp	Data[0]
N+1		OK	DATA

- Response: OK(0x00)
- Data[0] : Applied speed.
  - : 0xH<sub>1</sub> H<sub>2</sub> ( H<sub>1</sub> : RX Speed nibble, H<sub>2</sub> : TX Speed nibble)
  - : H – 0(106Kbps), 1(212Kbps), 2(424Kbps), 3(847Kbps),  
4(1.7Mbps) , 5(3.4Mbps) , 6(6.8Mbps)
  - Ex) 0x63(RX : 6.8M, TX : 847K)

<This log shows all data include length, command and data >

1 =>6005000050 (REQB)

2 <=00500123456711000011778183

3 =>601D012345670008010050 (ARRTIBUTE)

4 <= 0000

5 =>68 (SPARAMETER SET)

6 <= 0064 (applied Rx speed is 6.8Mbps and Tx Speed is 1.7Mbps)

7 =>6002C0B000000F80 (APDU with PCB)

8 <=00020000000000000000000000000000009000

9 =>6003C0B000000F80 (APDU with PCB)

10 <= 000300000000000000000000000000000000009000

<This log shows all data include length, command and data >

1 =>4C04000100 (Find Card, VHBR speed and CID=0)

2 <= 00420050012345671100001177818301234567

3 =>61C0B000000F (APDU)

4 <=00000000000000000000000000000000009000



### 3.6.5 RF Find Tags (Command = 0x83, \_DE\_FIND\_TAGS)

This command detects tags in the RF field.

#### ■ Command frame

LEN-H	LEN-L	CMD
0x0001		0x83

#### ■ Response frame

LEN-H	LEN-L	Resp	Data[0...N-1]
N+1		OK	ATR

- ATR

Byte	Value(HEX)	Designation	Description
0	3B	TS	
1	8n 4 < n < 16	T0	Higher nibble 8 means : only TD1 is following Lower nibble n means : length of historical bytes(Tk)
2	80	TD1	Higher nibble 8 means : only TD2 is following Lower nibble 0 means : T=1 supported
3	01	TD2	Higher nibble 0 means : no data is following Lower nibble 1 means : T=1 supported
4 ~ 3+n	0xHH	Tk	Byte for card type <b>0xF0 or 0x01 - Mifare card</b> <b>0xFD or 0x02 – ISO15693 card</b> <b>0xFC or 0x03 – FeliCa card</b> <b>0xF1 - Topaz, NFC Type1 Tag</b> <b>0xFB – NFC Barcode(KOVIO)</b>
	UID[5~ ]		UID for MiFare : 4byte or 7byte (or 10byte) UID for FeilCa & ISO15693 : 8byte UID for NFC Barcode : 14byte
	0xHH		MIFARE Type 0x30 : MIFARE 1K,      0x31 : MIFARE Ultra Light 0x32 : MIFARE 4K,      0x33 : MIFARE Mini 0x34 : MIFARE PLUS 2K 0x35 : MIFARE PLUS 4K
4+n	0xHH	TCK	Exclusive-oring of all the bytes T0 to TK

-  
-  
-

- Data[1]: Card communication speed
- Data[2]: Card Data and Card UID. It is different each card.
- Response: OK(0x00), Err(pls refer to the response code).

### 3.7 IC-Card Functions

#### 3.7.1 IC Card Power On (Command = 0xC0, \_DE\_CARD\_PON)

Reader resets the card slot or sends PPS command to the active slot.

##### ■ Command frame of reset

LENG-H	LENG-L	CMD	Data[0]	Data[1]
0x0003		0xC0	SLOT	PPS

- Data[0]: SLOT

Data[0]	SLOT
0x00	contact card slot or first SAM slot
0x01 ~	SAM slot

- Data[1] = 0xF0 : Reader executes PPS exchange automatically.
- = 0xD0 : Reader receives ATR without analyzing ATR format.
- = 0xA (lower nibble is 0x0A) : Warm Reset.
- = 0x8 (lower nibble is 0x08) : Do not send IFS request although T=1 protocol
- = others : Reader does not execute PPS exchange.

##### ■ Response frame of reset

LENG-H	LENG-L	Resp	Data[0..N-1]
N+1		OK	

- Data[0..N-1] : ATR
- Response: OK(0x00), Err(pls refer to the response code).

##### ■ Command frame of PPS

LENG-H	LENG-L	CMD	Data[0]	Data[1...4]
0x0006		0xC0	SLOT	0xFF(PPSS)+PPS0~[2...4]

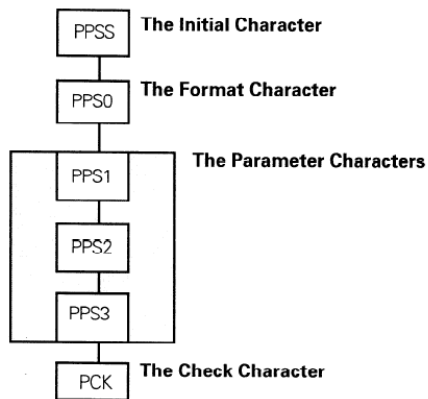
- Data[0]: SLOT

Data[0]	SLOT
0x00	contact card slot or first SAM slot



0x01 ~	SAM slot
--------	----------

- Data[1] = 0xFF : it means PPSS
- 0xHH(except 0xFF) : it doesn't PPS.



<Fig. 1 : Structure of PPS>

PPS example :      0xff 0x10 0x11 0xfe  
                          0xff 0x10 0x12 0xfd  
                          0xff 0x10 0x13 0xfc

#### ■ Response frame of PPS

LENG-H	LENG-L	Resp	Data[0..N-1]
N+1		OK	

- Data[0..N-1] : PPS response
- Response: OK(0x00), Err(pls refer to the response code).

### 3.7.2 IC Card Case1 (Command = 0xC1, \_DE\_CARD\_CASE1)

Reader sends 5 bytes APDU and receives 2 bytes(status words).

This command support case 1 APDU in ISO/IEC 7816-3.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1..5]
0x0007		0xC1	SLOT	APDU

- Data[0]: SLOT, Refer to "3.6.1 IC Card Power On command".
- Data[1..5]: APDU, CLA(1byte) + INS(1byte) + P1(1byte) + P2(1byte) + 0x00

#### ■ Response frame

LEN-H	LEN-L	Resp	Data[0..1]
0x0003		OK	SW

- Response: OK(0x00), Err(pls refer to the response code).

### 3.7.3 IC Card Case2 (Command = 0xC2, \_DE\_CARD\_CASE2)

Reader sends APDU(5 bytes) and receives N bytes(data and status words).

This command support case 2 APDU in ISO/IEC 7816-3.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1..5]
0x0007		0xC2	SLOT	APDU[0..4]

- Data[0]: SLOT, Refer to “3.6.1 IC Card Power On command”.
- Data[1...5]: APDU, CLA(1byte) + INS(1byte) + P1(1byte) + P2(1byte) + Le

#### ■ Response frame

LEN-H	LEN-L	Resp	Data[0..N-1]
N+1		OK	Data + SW

- Data[0...N-1] : Response data from IC card and length is sum of APDU[4](Le) and SW(2-byte).
- Response: OK(0x00), Err(pls refer to the response code).

### 3.7.4 IC Card Case3 (Command = 0xC3, \_DE\_CARD\_CASE3)

Reader sends APDU+data(n)(5+n bytes) and receives status words.

This command support case 3 APDU in ISO/IEC 7816-3.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1..n+5]
0x0007+n		0xC3	SLOT	APDU[0..4]+data[0..n]

- Data[0]: SLOT, Refer to “3.6.1 IC Card Power On command”.
- Data[1...n+5]: APDU, CLA + INS + P1 + P2 + Lc + n-byte data.

#### ■ Response frame

LEN-H	LEN-L	Resp	Data[0..1]
0x0003		OK	SW

- Response: OK(0x00), Err(pls refer to the response code).

### 3.7.5 IC Card Case4 (Command = 0xC4, \_DE\_CARD\_CASE4)

Reader sends APDU+data(n)+Le(1)(6+n bytes) and receives N bytes(data and status words).  
This command support case 4 APDU in ISO/IEC 7816-3.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1..n+6]
0x0008+n		0xC4	SLOT	APDU[0..4]+data[0..n-1]+Le[0]

- Data[0]: SLOT, Refer to “3.6.1 IC Card Power On command”.

- Data[1...n+6]: APDU,

CLA(1byte) + INS(1byte) + P1(1byte) + P2(1byte) + Lc(1byte)

data[0]...[n-1] : n-byte data.

Le[0] : length of expected response data.

#### ■ Response frame

LEN-H	LEN-L	Resp	Data[0..N-1]
N+1		OK	Data + SW

- Data[0...N-1] : Response data from IC card and length is sum of APDU[4](Le) and SW(2-byte).

- Response: OK(0x00), Err(pls refer to the response code).

### 3.7.6 IC Card Power Off (Command = 0xC5, \_DE\_CARD\_POFF)

Reader disables contact card in the slot.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]
0x0002		0xC5	SLOT

- Data[0]: SLOT, Refer to “3.6.1 IC Card Power On command”.

#### ■ Response frame

LEN-H	LEN-L	Resp
0x0001		OK

- Response: OK(0x00), Err(pls refer to the response code).

### 3.7.7 IC Card T1 Bypass (Command = 0xC7, \_DE\_CARD\_T1BYPASS)

On receiving this command, reader bypasses data from pc application to the selected card or SAM slot but the data format should be T1 format. Reader bypasses data from card to PC application.

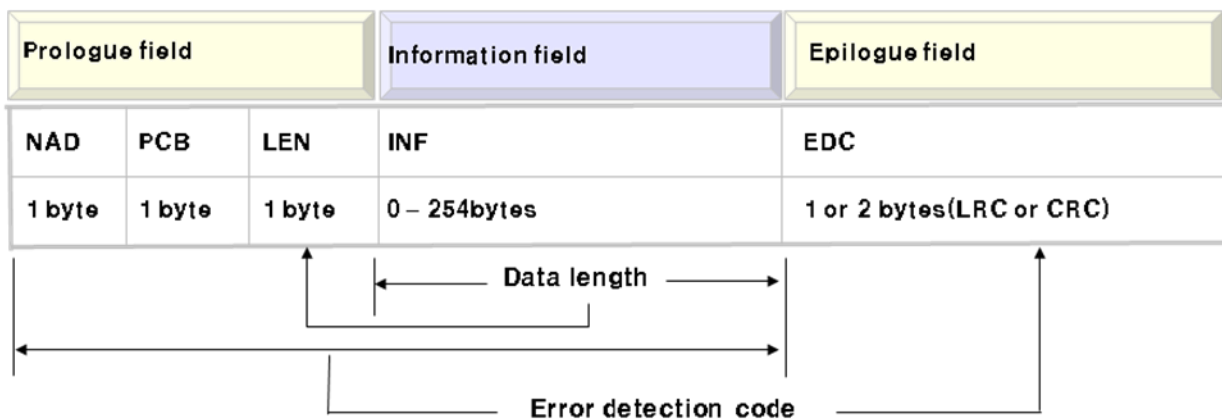
#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1..N-1]
N+1		0xC7	Slot	

- Data[0]: SLOT, Refer to “3.6.1 IC Card Power On command”.

- Data[1...N-1]:

Data[1]	NAD
Data[2]	PCB
Data[3]	LEN
Data[4..N-1]	INF
Data[N]	EDC



\* NAD : Node Address, PCB : Protocol Control Byte, INF : Information, EDC : Error Detection Code

○ : Mandatory field      ○ : Optional field

<figure.2 : T1 Block frame – ISO 7816-3>

#### ■ Response frame

LEN-H	LEN-L	Resp	Data[0..N-1]
N+1		OK	Data[0...N-1]

- Data[0...N-1] : receiving data from card

- Response: OK(0x00), Err(pls refer to the response code).

### 3.7.8 IC Card Speed Set (Command = 0xC8, \_DE\_CARD\_SPEED)

Reader changes the communication speed between reader and contact card. This command does not include PPS(PTS) command to card. This command could be used after changing card's communication speed using special APDU(it's depend on card). The speed defined by ISO 7816's FIDI value.

#### ■ Command frame

LEN-H	LEN-L	CMD	Data[0]	Data[1]
0x0003		0xC8	SLOT	FIDI

- Data[0]: SLOT, Refer to "3.6.1 IC Card Power On command".
- Data[1]: FIDI, FI and DI value of ISO 7816-3

#### ■ Response frame

LEN-H	LEN-L	Reps
0x0001		OK

- Response: OK(0x00), Err(pls refer to the response code).

### 3.7.9 IC Card APDU (Command = 0xC9, \_DE\_CARD\_APDU)

Reader sends APDU to card and receive data from card.

In case of T0, reader analyzes case1 to 3 and support get-response procedure.(delete)

In case of T1, it makes T1 frame, sends frame to card and receive data from card.

#### ■ Command Frame

LEN-H	LEN-L	CMD	Data[0]	Data[1..N-1]
N+1		0xC9	SLOT	APDU[0..4]+data[5..N-2]

- Data[0]: SLOT, Refer to "3.6.1 IC Card Power On command".

#### ■ Response Frame

LEN-H	LEN-L	Resp	Data[0..N-1]
N+1		OK	Data + SW

- Response: OK(0x00), Err(pls refer to the response code).

### 3.7.10 IC Card BYPASS (Command = 0xCB, \_DE\_CARD\_BYPASS)

Reader sends APDU or DATA to card and receive data from card. It doesn't process any PB or Error recovery. This API is for COS or Chip developers.

#### ■ Command Frame

LEN-H	LEN-L	CMD	Data[0]	Data[1...N-1]
N+1		0xCB	SLOT	APDU or any data

- Data[0]: SLOT, Refer to "3.6.1 IC Card Power On command".

#### ■ Response Frame

LEN-H	LEN-L	Resp	Data[0..N-1]
N+1		OK	PB or Data + SW

- Response: OK(0x00), Err(pls refer to the response code).

### 3.7.11 IC CARD STATUS (Command = 0xCA, \_DE\_CARD\_EXIST)

This command checks card existence at selected slot.

#### ■ Command Frame

LEN-H	LEN-L	CMD	Data[0]
N+1		0xCA	SLOT

#### ■ Response frame

LEN-H	LEN-L	Resp
N+1		OK

- Response: OK(0x00-Card Exists), Others(Does not exist).

### 3.8 ISO-15693 Commands (optional functions)

These commands are only valid for the reader with chip supporting ISO 15693 functions. The configuration parameters used by reader are as followings: 100% ASK, Fast Mode. \_DED\_INVENTORY, \_DED\_SELECT, \_DED\_READ, \_DED\_WRITE functions are just for showing the functionalities of ISO15693. If you want implement full functions of those commands, please use \_DED\_TRANSPARENT command to send your own data to the card.

#### 3.8.1 15693 Find Card (Command = 0x70, \_DED\_INVENTORY)

This command sends the Inventory request and checks whether a card is in the operating field of the antenna. The Inventory function is necessary at the beginning of a new selection of a card. User can assign AFI value optionally in this command.

***This command finds all cards in the field and returns the number of found cards and their UIDs. This command performs anti-collision to find cards and fall them into quiet mode. So, developers must execute RF\_OFF and RF\_ON command to communicate with some card.***

##### ■ Command Frame

LENG-H	LENG-L	CMD	Data[0]	Data[1]
0x0002 or 0x0003		0x70	Flag	Afi(optional)

- Data[0]: Flag, 0x00 or 0x01 = 1 time slot  
0x10 = 16 time slots

- Data[1]: Afi, optional data byte to assign AFI in Inventory command.

**Terminal doesn't send AFI when this byte is not present.**

- Please refer to ISO15693 standards for detailed information of Anti-collision.

##### ■ Response Frame

LENG-H	LENG-L	Resp	Data[0]	Data[1..8]	Data[9...16]	...	Data[...8*n]
8xn + 2		OK	Card No	UID#1	UID#2		UID#n

- Data[0] : Found Card Count

- Data[1..8\*n]: UIDs, Unique Identifier, each UID is 8Bytes

- Response: OK(0x00), Err(pls refer to the response code).

#### 3.8.2 15693 Select (Command = 0x71, \_DED\_SELECT)

This command selects the card with the specified serial number during inventory request .

#### ■ Command Frame

LENG-H	LENG-L	CMD	Data[0...7]
0x0001 or 0x0009		0x71	UID(optional)

- **When UID is absent, reader selects the card which was found last.**

#### ■ Response Frame

LENG-H	LENG-L	Resp	Data[0]
0x0002		OK	Flags

- Data[0]: Flags, Response status byte

- Response: OK(0x00), Err(pls refer to the response code).

### 3.8.3 15693 Read (Command = 0x72, \_DED\_READ)

This command reads specified block of the selected card.

#### ■ Command Frame

LENG-H	LENG-L	CMD	Data[0]
0x0001		0x72	BN

- BN : Block number to be read

#### ■ Response Frame

LENG-H	LENG-L	Resp	Data[0]	Data[1]	Data[2...5]
0x0007		OK	Flags	BSS	Read Data

- Received Data Format if Response is OK :

Data[0]: Flags, Response status byte

Data[1]: BSS, Block security status

Data[2...5]: Read Data, stored data in block

- Response: OK(0x00), Err(pls refer to the response code).

### 3.8.4 15693 Write (Command = 0x73, \_DED\_WRITE)

This command writes 4bytes data into the specified block of the selected card.

#### ■ Command Frame

LENG-H	LENG-L	CMD	Data[0]	Data[1...4]
0x0006		0x73	BN	Write data

- Data[0]: Block number to be read

Reader transmits EOF after write command if MSB of block number is 1.



(It means Data[0] to be '0x80 | BN').

#### ■ Response Frame

LENG-H	LENG-L	Resp	Data[0]
0x0002		OK	Flags

- Received Data Format if Response is OK :

Data[0]: Flags, Response status byte

- Response: OK(0x00), Err(pls refer to the response code).

### 3.8.5 15693 Write (Command = 0x7E, \_DED\_WRITE8)

This command writes 8bytes data into the specified block of the selected card.

#### ■ Command Frame

LENG-H	LENG-L	CMD	Data[0]	Data[1...8]
0x0006		0x7E	BN	Write data

- Data[0]: Block number to be read

#### ■ Response Frame

LENG-H	LENG-L	Resp	Data[0]
0x0002		OK	Flags

- Received Data Format if Response is OK :

Data[0]: Flags, Response status byte

- Response: OK(0x00), Err(pls refer to the response code).

### 3.8.6 15693 Transparent (Command = 0x74, \_DED\_TRANSPARENT)

The reader receives data from Host and sends this data to the accessed card. ISO15693 CRC data is computed in the reader. It is the best way to use this command to implement ISO commands set excluding EOF.

#### ■ Command Frame

LENG-H	LENG-L	CMD	Data[0...N-1]
N+1		0x74	Command data

- Data[0...N-1] : Command Data to be send to the Card. DE-620 computes CRC and attach 2-byte CRC to the frame internally.

- For example, Inventory request is coded as 0x36 0x01 0x00 0x00 in Command Data.

#### ■ Response Frame

LENG-H	LENG-L	Resp	Data[0...N-1]
N+1		OK	Response data

- Received Data Format if Response is OK :

Data[0...N-1] : Response Data received from Card excluding 2-byte CRC.

- Response: OK(0x00), Err(pls refer to the response code).

### 3.8.7 15693 Transparent2 (Command = 0x75, \_DED\_TRANSPARENT2)

The reader receives data from Host and sends data to the accessed card. ISO15693 CRC data must be computed in the host side.

#### ■ Command Frame

LENG-H	LENG-L	CMD	Data[0...N-1]
N+1		0x75	Command data

- Data[0...N-1] : Command Data to be send to the Card. DE-620 does not computes CRC internally.

#### ■ Response Frame

LENG-H	LENG-L	Resp	Data[0...N-1]
N+1		OK	Response data

- Received Data Format if Response is OK :

Data[0...N-1] : Response Data received from Card.

- Response: OK(0x00), Err(pls refer to the response code).

### 3.8.8 15693 Next Slot (Command = 0x78, \_DED\_NEXTSLOT)

This command is used to change time slot to the next. This function must be executed after Inventory function with multiple slots. This command can be used after any kinds of commands.

#### ■ Command Frame

LENG-H	LENG-L	CMD
0x0001		0x78

#### ■ Response Frame if the previous command is Inventory

LENG-H	LENG-L	Resp	Data[0]	Data[1]	Data[2...11]
0x000B		OK	Flags	DSFID	UID + CRC

- Received Data Format if Response is OK :

Flags : Response status byte

DSFID : Data Storage Format Identifier

UID+CRC :Unique Identifier + CRC 2 byte

- Response: OK(0x00), Err(pls refer to the response code).

### 3.8.9 I-CODE Transparent (Command = 0x76, \_DED\_TRANSPARENT\_I)

DE-620 receives data from Host and sends data to the original ICODE card. It is the best way to use this command to implement ICODE commands set excluding QIUT.

#### ■ Command Frame

LENG-H	LENG-L	CMD	Data[0...N-1]
N+1		0x76	Command data

#### ■ Response Frame

LENG-H	LENG-L	Resp	Data[0...n-1]
N+1		OK	Response data

- Received Data Format if Response is OK :

Data[0...N-1] : Response Data received from Card.

- Response: OK(0x00), Err(pls refer to the response code).

### 3.9 NFC Commands (optional functions)

These commands are only valid for the reader with the chip supporting ISO 18092(NFC) functions. Please refer to ISO18092 and NFC Forum specifications for the detailed information.

#### 3.9.1 Device Initialization (Command = 0x90, \_NFC\_INIT)

This command sets the initial communication mode of NFC devices. Initiator or target mode with communication speed in passive and active communication modes can be established using this command. When this command is used in NFC\_A initiator, SENS\_REQ, SDD\_REQ, SEL\_REQ commands are executed in sequence internally to detect a target in the RF field. After this command, \_NFC\_INITIATOR\_COMM(0x91) command shall be followed to send ATR\_REQ with proper parameters. When in NFC\_F initiator, NFC\_F Polling command is executed.

##### ■ Command frame

LENG-H	LENG-L	CMD	Data[0]	Data[1]	Data[2]	Data[3]
0x0006		0x90	CMD	Para1	Para2	Para3

-Data[0] : CMD, 0x41('A') = Active initiator

0x50('P') = Passive initiator

0x54('T') = Target

0x4E('N') = Normal

-In case of CMD = 0x41('A') or 0x50('P') (Initiator mode)

Data[1] : communication speed, Para1 = 00(106kbps), 01(212kbps), 02(424kbps)

Data[2], Data[3] : not used.(default = 0x00)

-In case of CMD = 0x54('T') (Target mode)

Data[1] : Para1 = 0x00(Normal target), 0x01(Passive target), 0x02(Active target)

Data[2] : Para2 = 0x00(normal), 0x01(106kbps), 0x02(212kbps), 0x03(424kbps)

Data[3] : Para3 = Timeout value (Target will wait the data from Initiator by this value.)

##### ■ Response frame

LENG-H	LENG-L	Resp	Data[0...n-1]
N+1		OK	Response data

- Received Data Format if Response is OK :

Ex ) When the device is set as initiator in 106kps Passive mode.

Command frame : 90500000

(Passive Initiator at 106kbps)

Response frame : 0004034008123456

(Status code(1) + SENS\_RES(2) + SEL\_RES(1) + NFCID1(4))

Command frame : 9111D400112233445566778899AA00000000B0  
(ATR\_REQ : NFCID3(10) + DID(1) + BS(1) + BR(1) + PP(1) + TimeOut(1))

When the device is set as initiator in 212kps Passive mode.

Command frame : 90500100

(Passive Initiator at 212kbps)

Response frame : 00120101FEA2A3A4A5A6A7C0C1C2C3C4C5C6C7

(Status code(1) + LEN(1) + RES\_CODE(1) + NFCID2(8) + PAD(8))

Command frame : 9111D40001FEA2A3A4A5A6A7000000000000B0

(ATR\_REQ : NFCID3(10) + DID(1) + BS(1) + BR(1) + PP(1) + TimeOut(1))

### 3.9.2 Initiator Transparent (Command = 0x91, \_NFC\_INITIATOR\_COMM)

This command just sends the received data from host to the target at initiator device without any data processing. All data processing and handling shall be done at host side.

#### ■ Command frame

LENG-H	LENG-L	CMD	Data[0]	Data[1]	Data[2]	Data[3] ~ Data[n-2]	Data[n-1]
N+1		0x91	LEN	CMD1	CMD2	Payload	TO

-Data[0] : LEN = N bytes

-Data[1] : CMD1 = 0xD4

-Data[2] : CMD2 = 0x00/0x02/0x04/0x06/0x08/0x0A

-Data[3] ~ Data[n-2] : Payload to be sent

-Data[n-1] : Timeout Value

#### ■ Response frame

LENG-H	LENG-L	Resp	Data[0...n-1]
N+1		OK	Response data

- Received Data Format if Response is OK : LEN + RES1 + RES2 + Payload  
LEN = N bytes, RES1 = 0xD5, RES2 = 0x01/0x03/0x05/0x07/0x09/0x0B

### 3.9.3 Target Transparent (Command = 0x92, \_NFC\_TARGET\_COMM)

This command just sends the received data from host to the initiator at target device without data processing. All data processing and handling shall be done at host side.

#### ■ Command frame

LENG-H	LENG-L	CMD	Data[0]	Data[1]	Data[2]	Data[3]	Data[4] ~ Data[n-2]	Data[n-1]
N+1		0x92	0x01	LEN	RES1	RES2	Payload	TO

- Data[0] : Mode = 0x01
- Data[1] : LEN = N-1 bytes
- Data[2] : RES1 = 0xD5
- Data[2] : RES2 = 0x01/0x03/0x05/0x07/0x09/0x0B
- Data[3] ~ Data[n-2] : Payload to be sent
- Data[n-1] : Timeout Value

#### ■ Response frame

LENG-H	LENG-L	Resp	Data[0...n-1]
N+1		OK	Response data

- Received Data Format if Response is OK : LEN + CMD1 + CMD2 + Payload  
LEN = N bytes, CMD1 = 0xD4, CMD2 = 0x00/0x02/0x04/0x06/0x08/0x0A

### 3.9.4 Device (Command = 0x93, \_NFC\_CARD\_MODE)

To be determined.

### 3.9.5 Start Initiator Mode (Command = 0x94, \_NFC\_ISTART)

This command sets the NFC device to initiator mode and performs RF Collision Avoidance. And initiator in passive mode sends NFC Polling command to check the existence of the target device. If target device is detected, ATR\_REQ command will be sent. Initiator in active mode sends only ATR\_REQ command. NFCID1 + ATR\_RES in NFC\_A or NFCID2 + ATR\_RES in NFC\_F will be received by the Initiator.

#### ■ Command frame

LENG-H	LENG-L	CMD	Data[0]	Data[1]	Data[2]	Data[3] ~ Data[n-1]
0x0003 or 0x0004 or N+1		0x94	Mode	Para1	[DID]	[GB]

- Data[0] : Mode, 0x41('A') = Active initiator  
0x50('P') = Passive initiator  
0x70('p') = Passive initiator
- Data[1] : Communication speed, Para1 = 00(106kbps), 01(212kbs), 02(424kbps)
- Data[2] : DID of Initiator. This is optional. When DID equals to zero, it means that DID is not used during communication. If this field doesn't present, it means DID = 0.

## ■ Response frame

LENG-H	LENG-L	Resp	Data[0...n-1]
N+1		OK	Response data

- Received Data Format if Response is OK :
- Response Data :   ATQA + SEL + NFCID1 + ATR\_RES in passive mode(106kbps)  
                          NFCID2 + PAD1 + MRTI + ATR\_RES in passive mode(212/424kbps)  
                          ATR\_RES in active mode  
                          In case of 0x70('p') mode, only ATR\_RES data is returned.

Ex) Example of start the reader as a Initiator

1 => 945001 (Start Initiator in 212kbps NFC-F mode)

2 <= 00+01FEC4727737CF9E0000000000000000  
      +26D50101FEC4727737CF9E00000000000732  
      +46666D010111020207FF03020013040164070103

ATR\_REQ = 1ED40001FEC4727737CF9E00000000003246666D01011103020003040164

ATR\_RES = 26D50101FEC4727737CF9E00000000000732  
          +46666D010111020207FF0302001304 0164070103

Ex) Example of start the reader as a Initiator (There is no NFCID field with 0x70 mode)

1 => 947001(Start Initiator in 212kbps NFC-F mode)

2 <= 00+26D50101FEC4727737CF9E00000000000732  
      +46666D010111020207FF03020013040164070103

ATR\_REQ = 1ED40001FEC4727737CF9E00000000003246666D01011103020003040164

ATR\_RES = 26D50101FEC4727737CF9E00000000000732  
          +46666D010111020207FF03020013040164070103

Ex) Example of start the reader as an Initiator with user defined General Bytes

1 => 947001001**12233**

2 <= 00+26D50101FEC4727737CF9E00000000000732  
      +46666D010111020207FF03020013040164070103

ATR\_REQ = 14D40001FEC4727737CF9E000000000032**112233**

ATR\_RES = 26D50101FEC4727737CF9E00000000000732  
          +46666D010111020207FF03020013040164070103

### 3.9.6 Start Target Mode (Command = 0x95, \_NFC\_TSTART)

This command sets the NFC device to target mode and the target will be answered automatically for the Polling and ATR\_REQ command. \_NFC\_GET\_TARGET(0x99) command or \_NFC\_SEND\_DATAT(0x97) shall be followed after this command to enable data reception from the Initiator.

#### ■ Command frame

LENG-H	LENG-L	CMD	Data[0]	Data[1]	Data[2]	Data[3]
0x0004		0x95	Proto-Sel	Mode	Speed	TO

-Data[0] : Protocol Selection,

0x54('T') = Use \_NFC\_SEND\_DATAT(0x97) command to receive/send data from/to Initiator.

0x74('t') = Use \_NFC\_GET\_TARGET(0x99) command to receive data from Initiator and \_NFC\_SET\_TARGET(0x9A) command to send data to Initiator.

-Data[1] : Mode, 0x41('A') = Active target

0x50('P') = Passive target

-Data[2] = Speed

-Data[3] : Time-out value (Target will wait the data from Initiator by this value)

\*\_NFC\_SEND\_DATAT(0x97) command is useful when data to be sent at target is prepared before receiving data from initiator. This command is divided into two commands (\_NFC\_GET\_TARGET(0x99), \_NFC\_SET\_TARGET(0x9A)) for those cases preparing response data at target after analyzing data from initiator. There is time gap between two commands to prepare data to be sent at host side.

#### ■ Response frame

LENG-H	LENG-L	Resp	Data[0...n-1]
N+1		OK	Response data

- Received Data Format if Response is OK :

- Response data : ATR\_REQ

Ex) Example of start the reader as an initiator

1 => 95740000F0 (Target start)

2 <= 00F01ED4007728FD0238078921392600000032+46666D+01011103020013040196  
(ATR + Magic Number + LLC Parameters list)

ATR\_REQ = F01ED4007728FD023807892139260000003246666D01011103020013040196

ATR\_RES = F01FD50101FE0102030405060708000000073246666D01011103020003040164



Ex) Example of start the reader as a target with user defined General Bytes

1 => 95740000F0112233

2 <= 00F01ED400393346B360BF9BA333130000003246666D01011103020013040196

ATR\_REQ = F01ED400393346B360BF9BA333130000003246666D01011103020013040196

ATR\_RES = F015D50101FE01020304050607080000000732112233

### 3.9.7 Initiator DEP Data(Command = 0x96, \_NFC\_SEND\_DATAI)

This command can be used only when the reader is configured as initiator. After initiator establishes the connection with the target, initiator can send user data to the target using NFC DEP command (DEP\_REQ) and receive response data from the target. Error recovery, chaining and CRC generation/check operations are done internally by Initiator based on ISO18092. User can handle Information PDU only. ACK/NACK PDU and Supervisory PDU will be generated internally according to protocol procedures with the target.

#### ■ Command frame

LENG-H	LENG-L	CMD	Data[0] ~ Data[n-1]
N+1		0x96	Transport Data

-Data[0] ~ Data[n-1] : Data to be sent to Target(Max. Length : 500 bytes)

#### ■ Response frame

LENG-H	LENG-L	Resp	Data[0...n-1]
N+1		OK	Response data

- Received Data from Target if Response is OK.

### 3.9.8 Target DEP Data(Command = 0x97, \_NFC\_SEND\_DATAT)

This command can be used only when the reader is configured as target. After establishment of connection, target checks command data (DEP\_REQ) from the initiator and sends response data(DER\_RES) to the initiator. Error recovery, chaining and CRC generation/check operations are done internally by Target based on ISO18092. First of all, this command shall be executed to receive DEP\_REQ command data from the initiator after \_NFC\_TSTART(0x95) command. ACK PDU and Supervisory PDU will be generated internally according to protocol procedures with the initiator.

#### ■ Command frame

LENG-H	LENG-L	CMD	Data[0] ~ Data[n-1]
N+1		0x97	Transport Data

-Data[0] ~ Data[n-1] : Data to be sent to Initiator (Max. Length : 500 bytes)

#### ■ Response frame

LENG-H	LENG-L	Resp	Data[0...n-1]
N+1		OK	Response data

- Received Data from Initiator if Response is OK.

### 3.9.9 Deselect/Release/Wakeup/Stop (Command = 0x98, \_NFC\_STOP)

This command is used to send Deselect/Release/Wakeup Request in initiator mode to the target and end the operation of target in abnormal cases of the target.

#### ■ Command frame

LENG-H	LENG-L	CMD	Data[0]
N+1		0x98	CODE

-Data[0] : CODE = 0 Initiator sends DSL\_REQ.

CODE = 1 Initiator sends RLS\_REQ.

CODE = 2 It force Target to stay Initial\_State of the device.

Target device can't receive the Initiator command any more.

CODE = 3 Initiator sends WUP\_REQ used only in active mode communication.

#### ■ Response frame

LENG-H	LENG-L	Resp
0x0001		OK

### 3.9.10 Get Target Data(Command = 0x99, \_NFC\_GET\_TARGET)

This command can be used only when the reader is configured as target. After connecting with initiator, the reader as target receives data frame from the initiator and manages protocol management and error handling. The reader as target sends supervisory PDU(SPDU, RTOX\_REQ), to cover the time gap between when data is received from initiator and when the reader as target is ready to send data prepared by host. The timeout value in case of not receiving data from initiator is about 300ms. To receive data from initiator, host shall send this command to the reader. ACK PDU and Supervisory PDU will be generated internally according to protocol procedures with the initiator.

\_NFC\_SEND\_DATAT(0x97) command is divided into two commands, \_NFC\_GET\_TARGET(0x99) which is used to receive data from initiator and \_NFC\_SET\_TARGET(0x9A) used to send data to initiator, to make flexible communication between initiator and the reader as target. \_NFC\_SEND\_DATAT(0x97) commands need data to be sent prepared before receiving data from Initiator. It is impossible to know data to be sent before getting data from Initiator in some cases.

#### ■ Command frame

LENG-H	LENG-L	CMD
N+1		0x99

#### ■ Response frame

LENG-H	LENG-L	Resp	Data[0...n-1]
N+1		OK	Response data

- Received Data from Initiator if Response is OK.

### 3.9.11 Set Target Data (Command = 0x9A, \_NFC\_SET\_TARGET)

This command can be used only when the reader is configured as target. After connecting with initiator, the reader as target sends data frame received from host to the initiator and manages protocol management and error handling.

#### ■ Command frame

LENG-H	LENG-L	CMD	Data[0] ~ Data[n-1]
N+1		0x9A	Data to be sent

#### ■ Response frame

LENG-H	LENG-L	Resp
0x0001		OK

### 3.9.12 Get Target State (Command = 0x9B, \_NFC\_TARGET\_STATE)

This command is used to know what the current state of the target is.

#### ■ Command frame

LENG-H	LENG-L	CMD
0x0001		0x9B

#### ■ Response frame

LENG-H	LENG-L	Resp	Data[0]	Data[1]	Data[2]
0x0001		OK	State	Mode	Speed

-Data[0] : State, 0=Idle State, 1=Active State, 2=Deselect State, 4=Release State

-Data[1] : Mode, Passive Initiator=0x20, Active Target=0x21

-Data[2] : Speed, 0x00 = 106kbps, 0x11=212kbps, 0x22=424kbps

### 3.9.13 Set Target Data (Command = 0x9A, \_NFC\_SET\_TARGET)

This command can be used only when the reader is configured as target. After connecting with initiator, the reader as target sends data frame received from host to the initiator and manages protocol management and error handling.

#### ■ Command frame

LENG-H	LENG-L	CMD	Data[0] ~ Data[n-1]
N+1		0x9A	Data to be sent

#### ■ Response frame

LENG-H	LENG-L	Resp
0x0001		OK

### 3.9.14 NDEF Tag Batch Commands (Command = 0x9D, \_NFC\_NDEF\_TAG)

This command is used to batch read and write NDEF tags.

#### ■ Command frame

LENG-H	LENG-L	CMD	Data[0]	Data[1...]
-		0x9D	CMD Type	CC or NDEF URI Data

-Data[0] : CMD type, 0x00=CC Read, 0x01=NDEF Read,

0x10=CC Write, 0x11=NDEF URI Write

-Data[1..] : CC or NDEF URI data

#### ■ Response frame

LENG-H	LENG-L	Resp	Data[0...]
0x00xx		OK	CC or NDEF data

### 3.9.15 NFC\_Tag\_Command (Command = 0x9E, \_NFC\_TAG\_COMMAND)

This command is used to handle NFC tags defined by NFC Forum. There are 4 types of tags. User can use batch commands or can make own commands using transparent mode.

#### ■ Command frame

LENG-H	LENG-L	CMD	Data[0]	Data[1]	Data[2] ...
-		0x9E	Tag Type	Tag CMD	Optional Data

Data[0] : NFC Tag Type, 0x10 = Type 1, 0x20 = Type 2, 0x30 = Type 3, 0x40 = Type 4

Data[1] ; Tag CMD, command code to be executed,

Tag CMD=0xFF :Transparent Mode, All input data will be sent to the tag without any data processing

Data[2] .... : Optional Data according to command format

For Type 1 Tag (Tag Type = 0x10),

-Read Identifier (RID), Tag CMD=0x00

CMD	Data[0]	Data[1]
0x9E	0x10	0x00

-Read All, Tag CMD=0x01

Read all blocks from 0x00 to 0x0E in a static memory including Header ROM bytes  
(Total 130bytes = 128bytes user data + 2bytes HR)

CMD	Data[0]	Data[1]
0x9E	0x10	0x01

-Read a byte, Tag CMD=0x02

Read single byte in a static memory

CMD	Data[0]	Data[1]	Data[2]	Data[3]
0x9E	0x10	0x02	Block Addr	Byte Offset

Data[2] : Block address (0x00 ~ 0x0E),

Data[3] : Byte offset (0x00 ~ 0x07)

-WriteE, Tag CMD=0x03

Write single byte with erase in a static memory

CMD	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]
0x9E	0x10	0x03	Block Addr	Byte Offset	DATA

Data[2] : Block address (0x00 ~ 0x0E),

Data[3] : Byte offset (0x00 ~ 0x07)

Data[4] : Data to be written

-WriteNE, Tag CMD=0x04

Write single byte without erase in a static memory

CMD	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]
0x9E	0x10	0x04	Block Addr	Byte Offset	DATA

Data[2] : Block address (0x00 ~ 0x0E),

Data[3] : Byte offset (0x00 ~ 0x07)

Data[4] : Data to be written

For Type 2 Tag (Tag Type = 0x20),

-Read Block, Tag CMD=0x00

CMD	Data[0]	Data[1]	Data[2]
0x9E	0x20	0x00	BN

Data[2] : Block Number to be accessed. 1 block consists of 4bytes but reading 1 block returns 4 block data (16bytes)

-Write Block, Tag CMD=0x01

CMD	Data[0]	Data[1]	Data[2]	Data[3] ... Data[6]
0x9E	0x20	0x01	BN	DATA (4bytes)

Data[2] : Block Number to be accessed. 1 block consists of 4bytes but reading 1 block returns 4 block data (16bytes)

Data[3] ... Data[6] : 4 bytes data to be written for the specified block.

-Sector Select, Tag CMD=0x02

CMD	Data[0]	Data[1]	Data[2]
0x9E	0x20	0x02	SN

Data[2] : Sector Number to be accessed.

For Type 3 Tag (Tag Type = 0x30)

-Polling, Tag CMD=0x00

Send Polling command to the T3T.

CMD	Data[0]	Data[1]	Data[2].. Data[3]	Data[4]	Data[5]
0x9E	0x30	0x00	System Code	Time Slot	Request Flag

Data[2]...Data[3] : System Code (0xFFFF, 0x12FC)

Data[4] : Time Slot

Data[5] : System Code Request Flag, 0x00=no request, 0x01=request

-Check, Tag CMD=0x01

Read data from the specified Service Code and Block Lists.

CMD	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	...	Data[j]	Data[j+1]
0x9E	0x30	0x01	NS	NB	SC1L	SC1H	SC2L	SC2H	...	BL1H	BL1L

Data[2] : NS = Number of Services,

Data[3] : NB = the total Number of Blocks to be accessed

SCxL = Lower Byte of Service Code x,

SCxH = Higher Byte of Service Code x

Service Code consists of a Service Number(MSB 10bit of the 2bytes) and an Access Attribute(LSB 6bit of the 2bytes). Service Code should be sent in Little Endian format.

Access Attribute = 001001b (Read/Write)

= 001011b (Read Only)

BLxH : Higher Byte of Block x indicates Service Code List Order from 0x00 to 0x0F

The Service the block belongs to. The first BL1H should start from 0x00.

BLxL : Lower Byte of Block x indicates the Block Number of the Service Code

1 Block consists of 16 bytes of data.

To read data from Block 0, 1 of Service Code 0009h and Block 2,3 of Service Code 000Bh,

-> 9E 30 01 02 04 09 00 0B 00 00 00 00 01 01 02 01 03 should be sent to the reader

-Update, Tag CMD=0x02

Write data to the specified Service Code and Block Lists.

CMD	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	...	Data[j]	Data[j+1]
0x9E	0x30	0x02	NS	NB	SC1L	SC1H	SC2L	SC2H	...	BL1H	BL1L

...	Data[k] ...
...	16 * NB

Data[2] : NS = Number of Services,

Data[3] : NB = the total Number of Blocks to be accessed

SCxL = Lower Byte of Service Code x,

SCxH = Higher Byte of Service Code x

Service Code consists of a Service Number(MSB 10bit of the 2bytes) and an Access Attribute(LSB 6bit of the 2bytes). Service Code should be sent in Little Endian format.

Access Attribute = 001001b (Read/Write)

= 001011b (Read Only)

BLxH : Higher Byte of Block x indicates Service Code List Order from 0x00 to 0x0F

The Service the block belongs to. The first BL1H should start from 0x00.

BLxL : Lower Byte of Block x indicates the Block Number of the Service Code

To write data to Block 0 of Service Code 0009h and Block 2 of Service Code 000Bh,

-> 9E 30 02 02 02 09 00 0B 00 00 00 01 02 xx xx xx xx xx xx xx xx xx xx xx xx xx  
yy yy yy yy yy yy yy yy yy yy yy yy yy yy yy yy yy should be sent to the reader

For Type 4 Tag (Tag Type = 0x40)

-Select File, Tag CMD=0x00

CMD	Data[0]	Data[1]	Data[2]
0x9E	0x40	0x00	SFlag

Data[2] : SFlag = 0x00 Select NDEF Application  
= 0x01 Select CC File  
= 0x02 Select NDEF File

-Read Binary, Tag CMD=0x01

CMD	Data[0]	Data[1]	Data[2]
0x9E	0x40	0x01	LEN

Data[2] : LEN = length of data to be read,

-Update Binary, Tag CMD=0x02

CMD	Data[0]	Data[1]	Data[2]	Data[3] ... Data[n-1]
0x9E	0x40	0x01	LEN	DATA (LEN bytes)

Data[2] : LEN = length of data to be written

Data[3] ... Data[n-1] : Data to be written,

#### ■ Response frame

LENG-H	LENG-L	Resp	Data[0] .. Data[n-1]
n		OK	Response Data

For Type 1 Tag (Tag Type = 0x10),

-Read Identifier, Tag CMD = 0x00

Resp	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
OK	HR0	HR1	UID0	UID1	UID2	UID3

-HRx : Header ROM Format

-UIDx : Unique Identifier

-Read All, Tag CMD = 0x01

Resp	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	....	Data[128]	Data[129]
OK	HR0	HR1	UID0	UID1	UID2	....	xx	xx

-HRx : Header ROM Format



-UIDx : Unique Identifier

-Read a byte, Tag CMD = 0x02

Resp	Data[0]	Data[1]
OK	Byte Address	DATA

-Data[0] : Byte Address of the reading DATA

-Data[1] : DATA reading from a tag

-Write a byte, Tag CMD = 0x03 or 0x04

Resp	Data[0]	Data[1]
OK	Byte Address	DATA

-Data[0] : Byte Address of the written DATA

-Data[1] : DATA reading from a tag

For Type 2 Tag (Tag Type = 0x20),

-Read Block, Tag CMD=0x00

Resp	Data[0] ... Data[15]
OK	DATA(16bytes from the Block)

-Write Block, Tag CMD=0x01

Resp	Data[0]
OK	ACK(0x0A) or NACK

-Sector Select, Tag CMD=0x02

Resp	Data[0]
OK	ACK(0x0A) or NACK

For Type 3 Tag (Tag Type = 0x30)

-Polling, Tag CMD=0x00

Resp	Data[0]...Data[7]	Data[8]...Data[15]	Data[16]...Data[17]
OK	IDm	PMm	[System Code]

Optional Data : System Code when System Code is requested in the Polling Frame.

-Check, Tag CMD=0x01

Normal response from the Tag :

Resp	Data[0]	Data[1] ~
OK	NB	16 x BN

Data[0] : NB=Number of Blocks, it indicates the total block number from reading.

Data[1]~.: Data from the Tag, Byte number of data after reading should be multiple of 16.

Error response from the Tag :

Resp	Data[0]	Data[1]
OK	Status1 Flag	Status2 Flag

Data[0] : Status1 Flag = other value except 0x00

Data[1] : Status2 Flag = other value except 0x00

-Update, Tag CMD=0x02

Normal response from the Tag :

Resp	Data[0]	Data[1]
OK	Status1 Flag	Status2 Flag

Data[0] : Status1 Flag = 0x00

Data[1] : Status2 Flag = 0x00

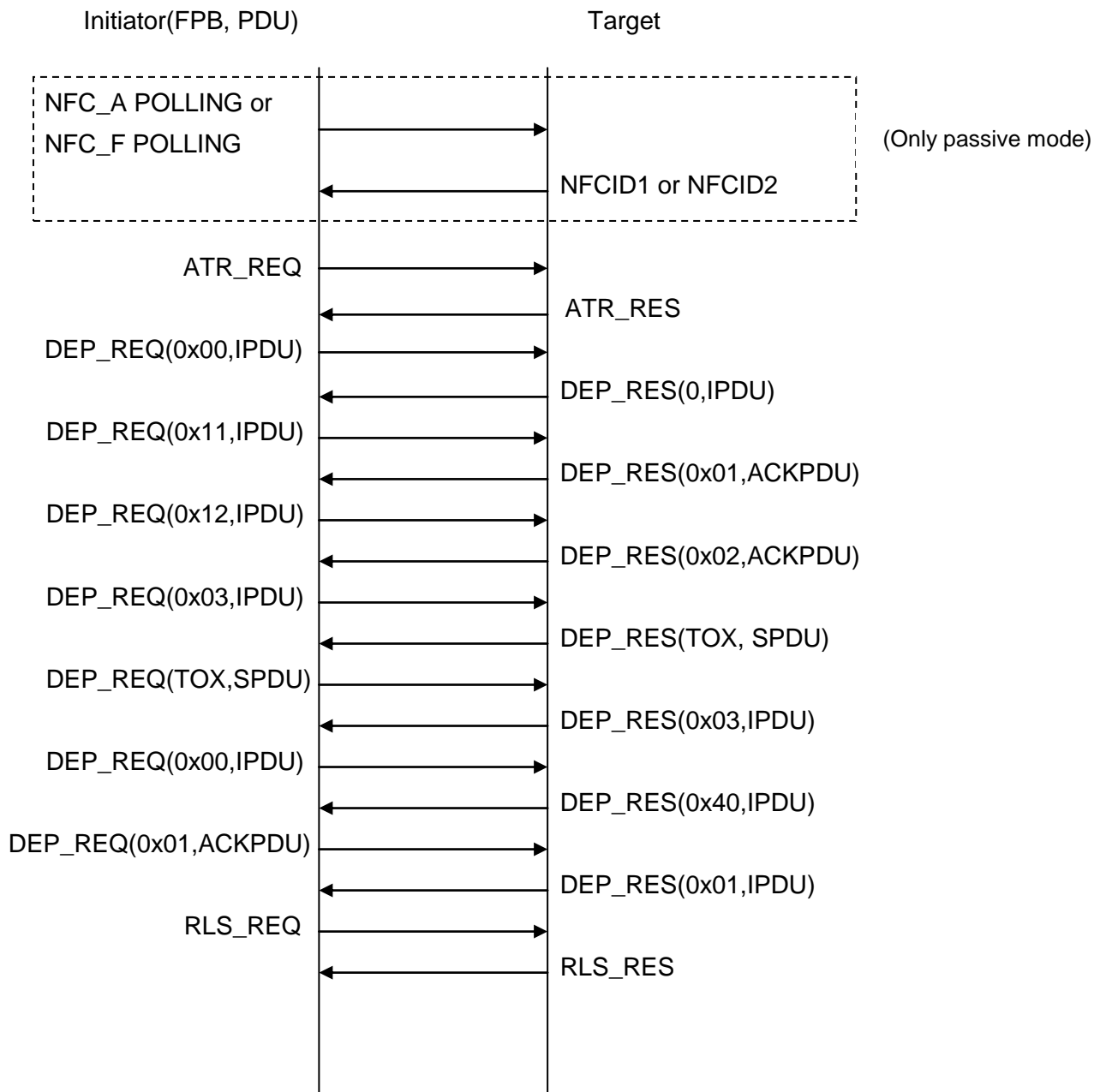
Error response from the Tag :

Resp	Data[0]	Data[1]
OK	Status1 Flag	Status2 Flag

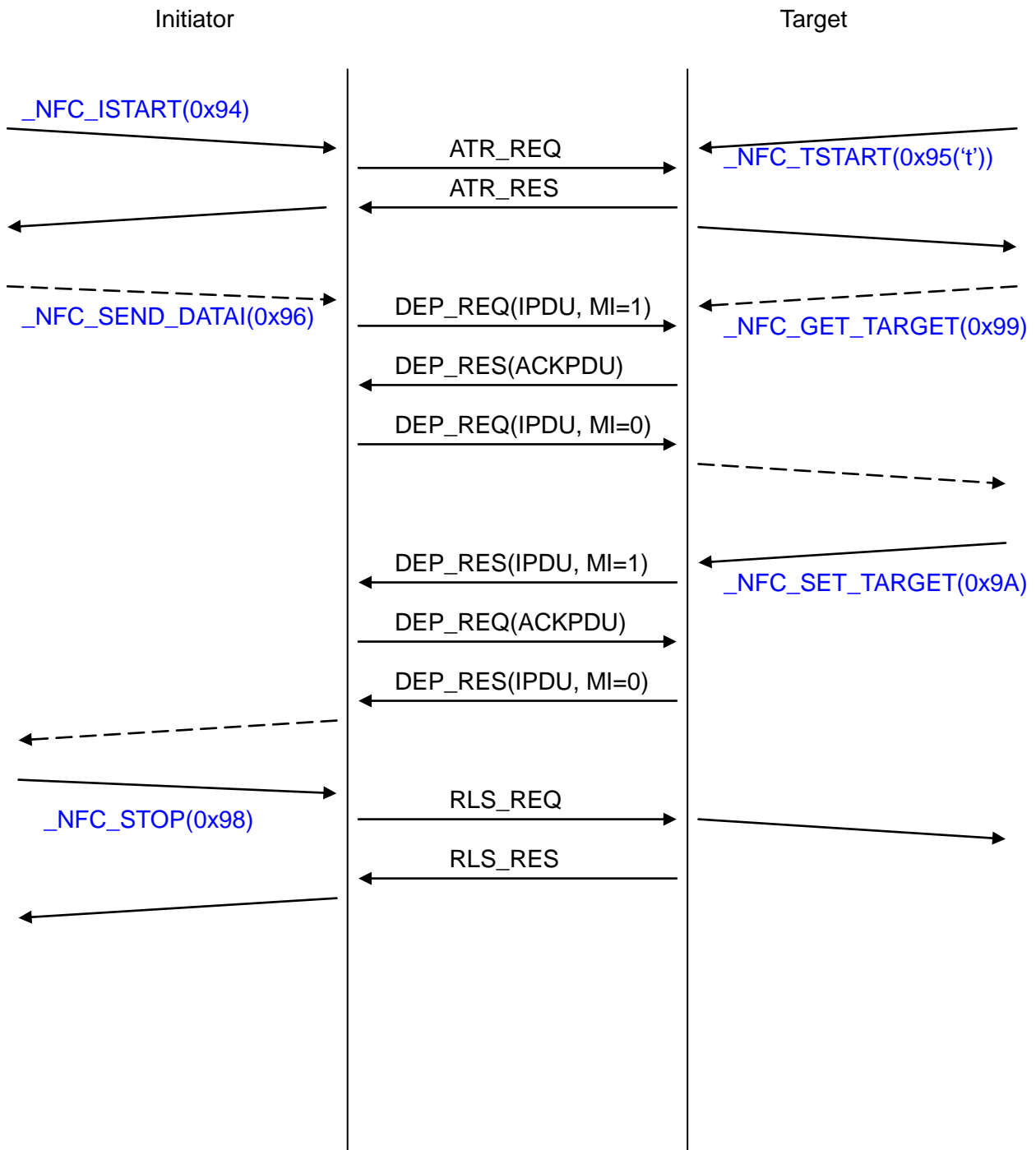
Data[0] : Status1 Flag = other value except 0x00

Data[1] : Status2 Flag = other value except 0x00

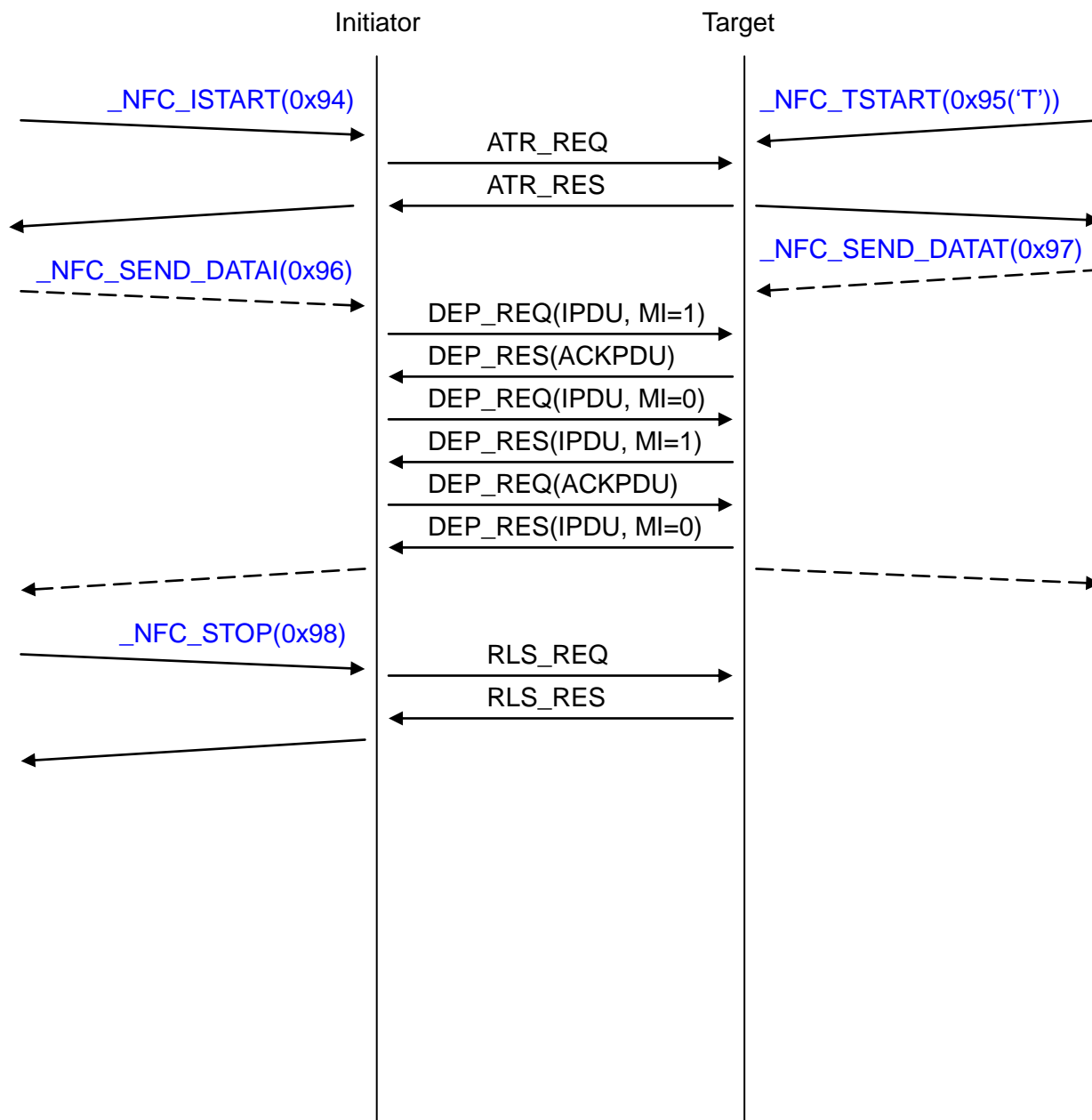
# 1) NFC Protocol Examples based on ISO18092



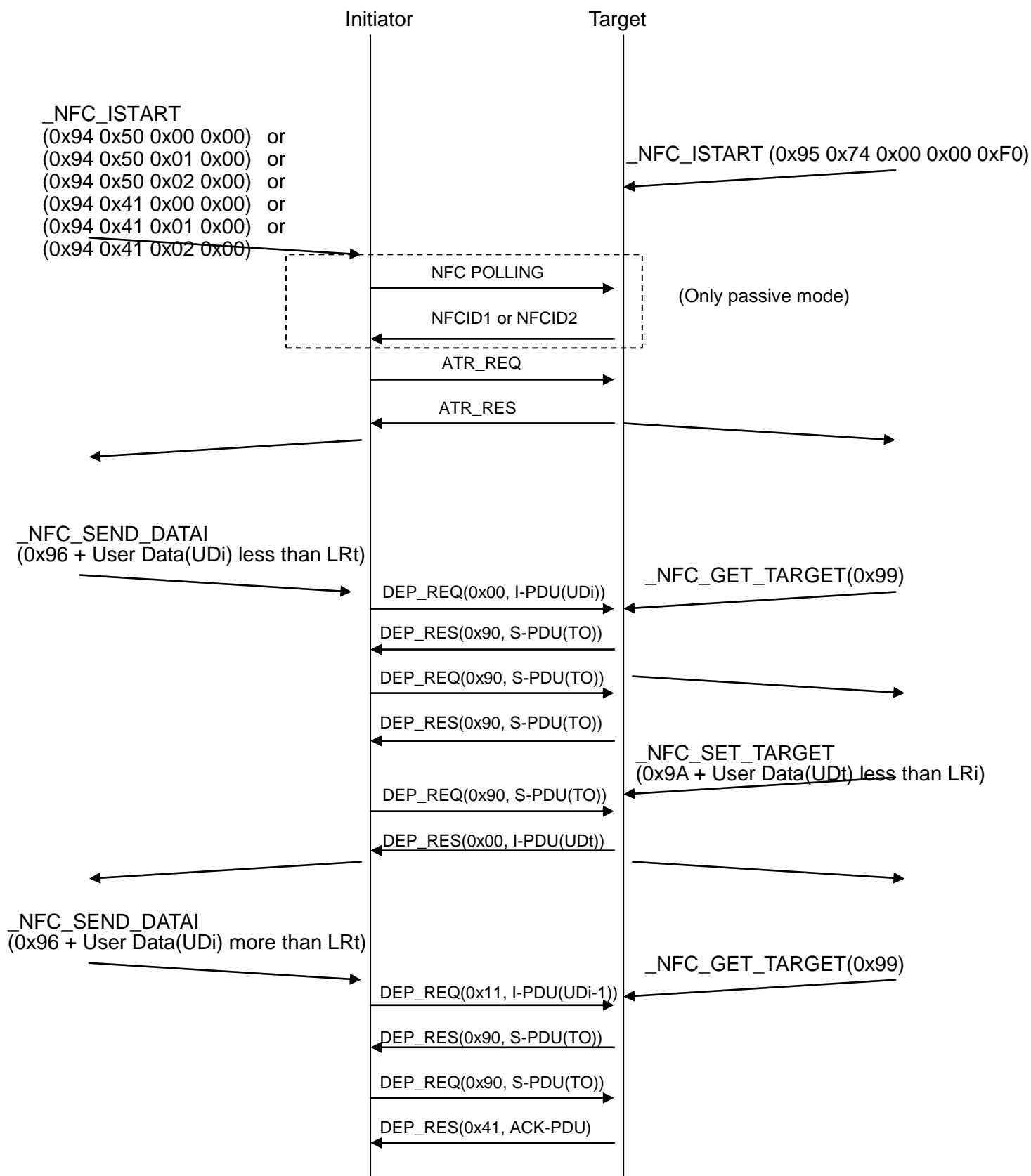
2) Example with defined commands (\_NFC\_GET\_TARGET, NFC\_SET\_TARGET) for NFC operation using two readers

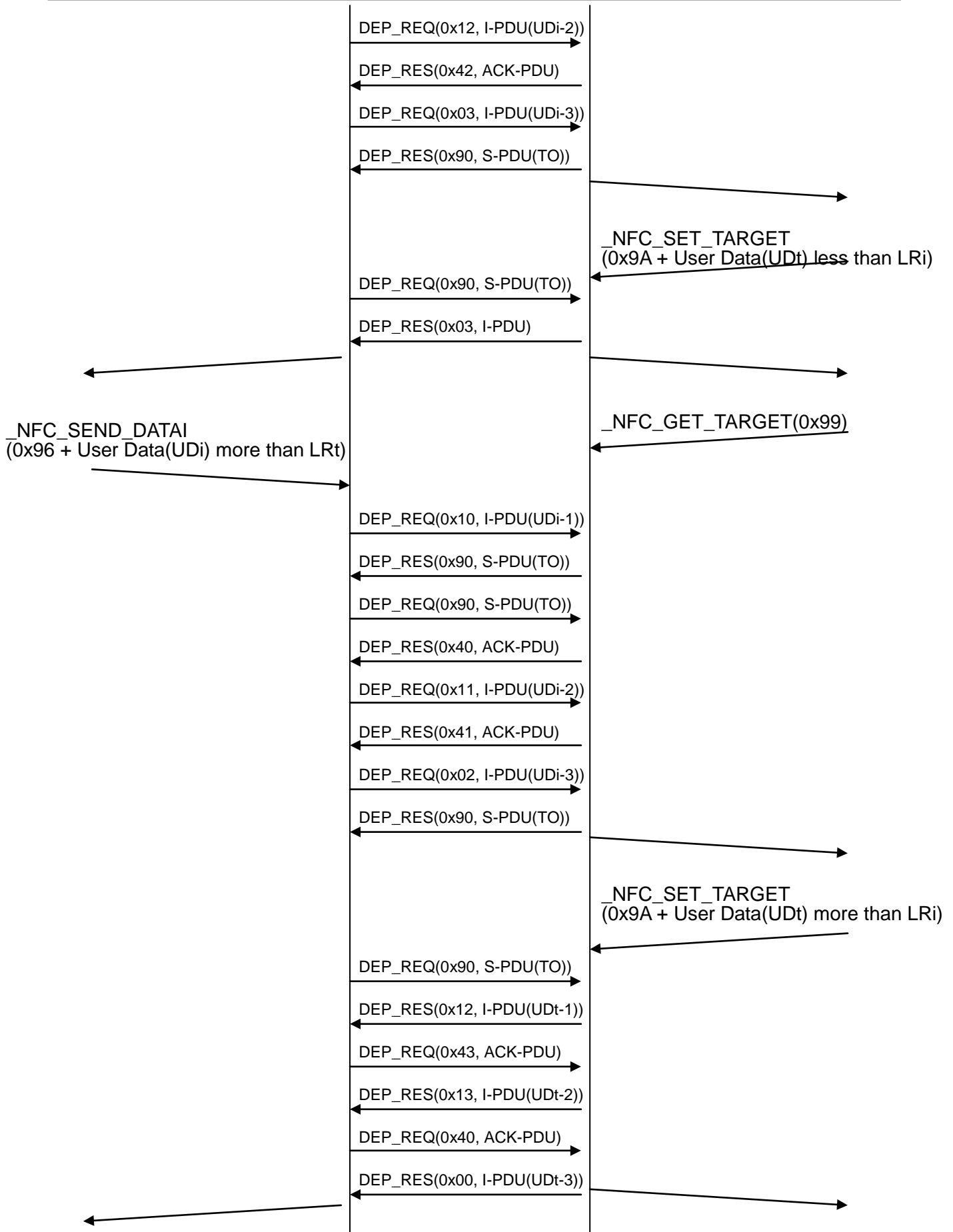


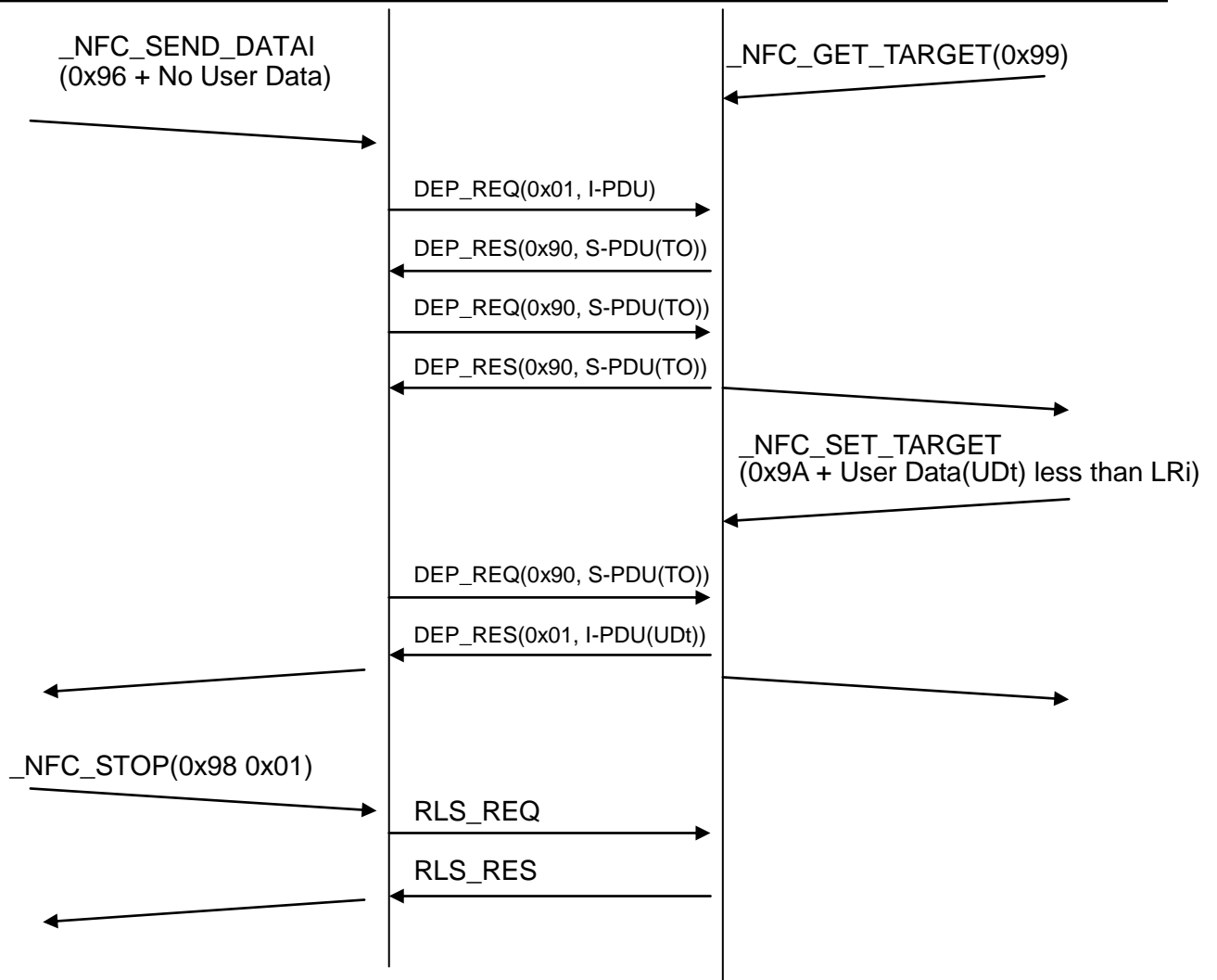
### 3) Example with defined commands (\_NFC\_SEND\_DATAT) for NFC operation using two readers



- 4) Example of detailed frame using defined commands (`_NFC_GET_TARGET`, `_NFC_SET_TARGET`) for NFC operation using two readers



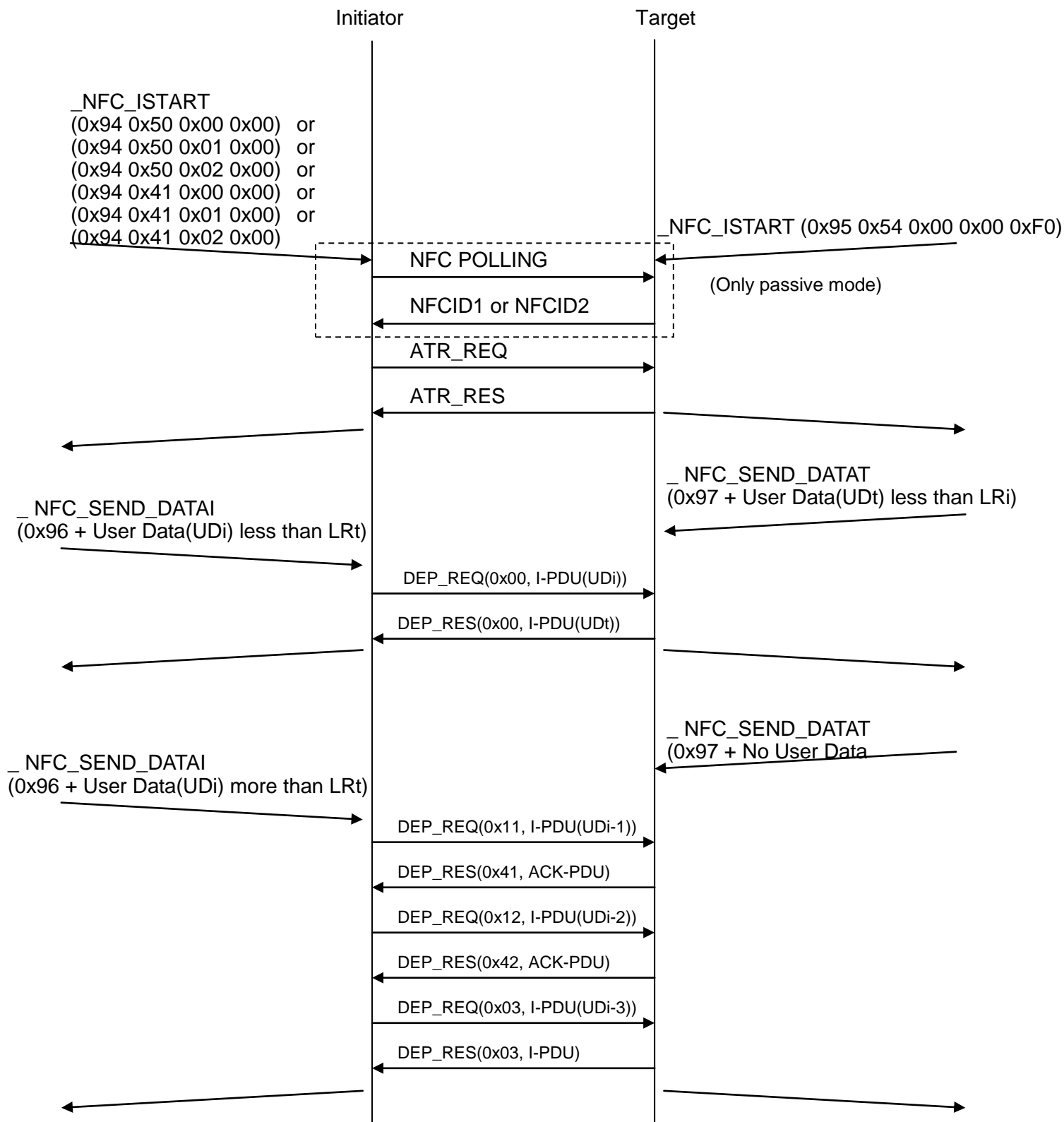


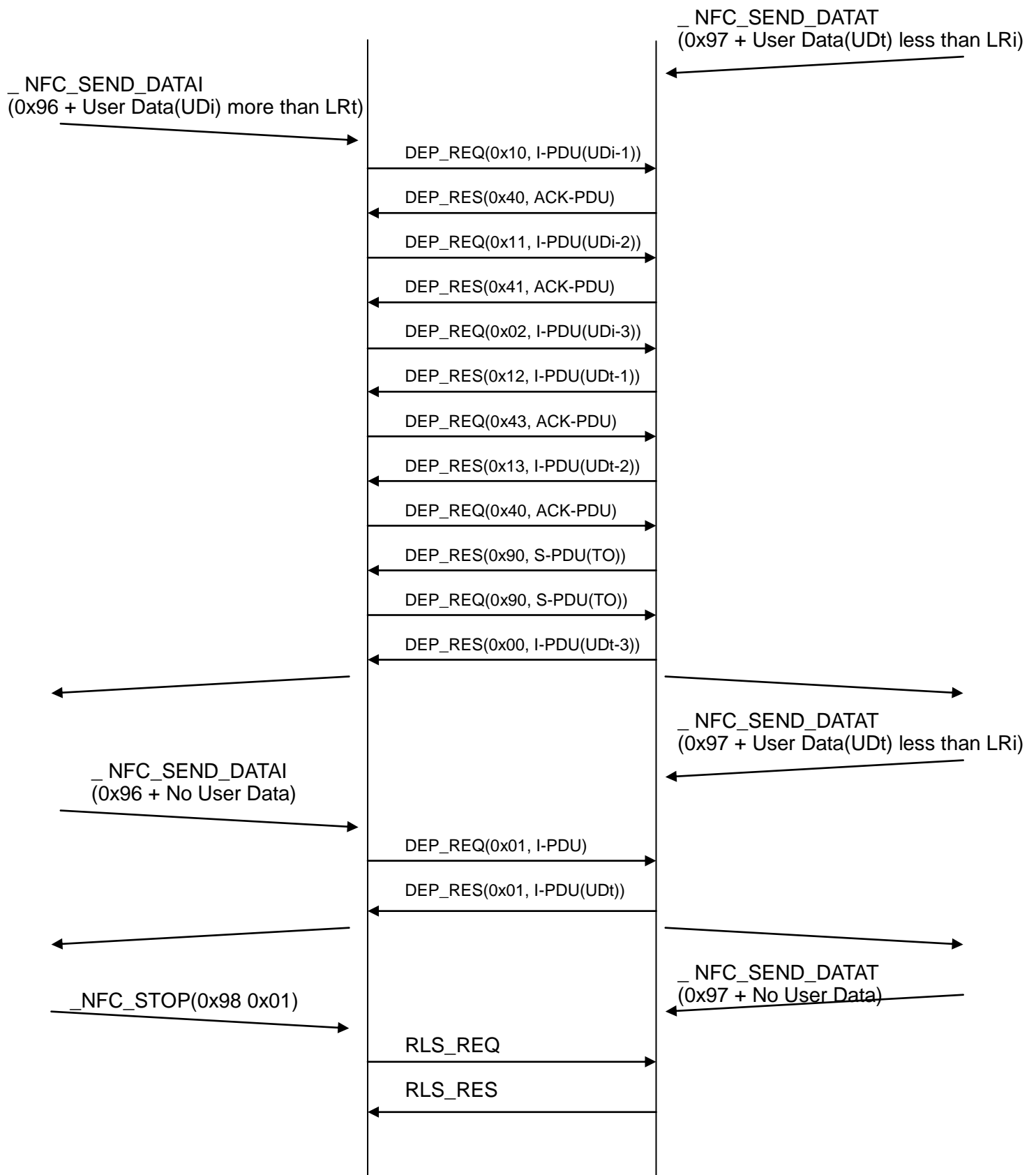


\*WTX feature can not be used in real communication.



5) Example of detailed frame using defined commands (\_NFC\_SEND\_DATAT) for NFC operation using two readers





\*WTX can not be used in real communication

### 3.10 LLC Commands

These commands are only valid for only the reader supporting NFC LLC features.

LLC(Logical Link Control) protocol is used during P2P(peer-to-peer) communication between NFC devices. LLCP Commands are used to communicate with remote LLC as an initiator or target.

User can also implement these LLC functionalities using ISO18092 commands explained above.

When it is impossible to implement LLC functionalities, please use LLC batch commands. Internal memory size of LLC batch commands to handle data is maximum 1Kbytes.

Please refer to LLCP specification from NFC Forum for detail information.

#### 3.10.1 MAC Activation (Command = 0xBE, \_LLC\_MAC\_ACTIVE)

This command can be used only when service access point (or service user) want to activate a MAC link between NFC devices according to LLCP specification. To check compatibility with the requirements of LLC specification, the use of the Magic Number shall be checked during the activation process. The NFC device can be configured as initiator or target to perform P2P communication. DID value is zero by default. It means DID is not used in ISO18092 protocol.

##### ■ Command frame

[Activation as Initiator]

LENG-H	LENG-L	CMD	Data[0]	Data[1]	Data[2]	Data[3] ~ Data[n-1]
0x0004 or N+1		0xBE	Mode	Speed	GB_Flag	[GB]

-Data[0] : Mode, 0x41('A') = Active initiator

0x70('p') = Passive initiator (Only ATR\_RES is included in response)

-Data[1] : Communication speed, speed = 00(106kbps), 01(212kbs), 02(424kbps)

-Data[2] : 0x00 = General bytes of ATR\_REQ are fixed as internal values.

0xFF = All data after Data[2] are used in General bytes of ATR\_REQ.

User can send own general bytes but shall keep the rule to make GB.

[Activation as Target]

LENG-H	LENG-L	CMD	Data[0]	Data[1]	Data[2]	Data[3]	Data[4] ... Data[n-1]
0x0005 or n		0xBE	Mode	0x00	GB_Flag	Para1	[GB_Data]

-Data[0] : Mode, 0x74('t') = Target

-Data[1] : always 0x00

-Data[2] : 0x00 = General bytes of ATR\_REQ are fixed as internal values.

0xFF = All data after Data[3] are used in General bytes of ATR\_RES.

User can send own general bytes but shall keep the rule to make GB.

-Data[3] : Para1=Timeout value. (Please use 0xF0 value.)

-Data[4] ... Data[n-1] : GB\_Data, Optional data of General Bytes

## ■ Response frame

LENG-H	LENG-L	Resp	Data[0] ~Data[9]	Data[10] ~ Data[n-1]
N+1		OK	[NFCID3]	LLC Parameters List

-Data[0] ~ Data[9] : NFCID3t(10bytes) in link activation as initiator,  
NFCID3i(10bytes) in link activation as target

-Data[10] ~ Data[n-1] : TLV formatted LLC parameters lists like VERSION, MIUX, WKS, LTO,  
RW, SN, OPT

Ex1) MAC Activation as a initiator in NFC-F(212k) using DualCard

1 => BE700100

2 <= 00+CEBAE73CA4BC52D340C3+01011003020001040196 (Status+NFCID3+LLC Para as  
TLV format)

ATR\_REQ = 1ED40001FEBA1560BD9FA80000000000324666D01011103020003040164

ATR\_RES = 1FD501CE9ABF4BC97903D3E96C0000000E324666D01011003020001040196

Ex2) MAC Activation as a initiator in NFC-F(212k) using DualCard

1 => BE7001FF**112233**

2 <= 00+FB3D2538E6326F77C34A+01011003020001040196 (Status+NFCID3+LLC Para as  
TLV format)

ATR\_REQ = 14D40001FE508D73DA0628000000000032**112233**

ATR\_RES = 1FD501FB3D2538E6326F77C34A0000000E324666D01011003020001040196

Ex3) MAC Activation as a target

1 => BE740000F0

2 <= 00E2665BC6EC65D460A21E01011003020001040196

ATR\_REQ = F01ED400E2665BC6EC65D460A21E000000324666D01011003020001040196

ATR\_RES = F01FD50101FE010203040506070800000007324666D01011103020003040164

Ex4) MAC Activation as a target with user defined General Bytes

1 => BE740000**FFF0112233**

2 <= 00+110951B89A85FD07D313+01011003020001040196

ATR\_REQ = F01ED400BB2185E6CE2ED9EFD08000000032+46666D+01011003020001040196

ATR\_RES = F015D50101FE01020304050607080000000732+**112233**

### 3.10.2 MAC Deactivation (Command = 0xBF, \_LLC\_MAC\_DEACTIVE)

. This command can be used only when service access point (or service user) want to deactivate a MAC link between NFC devices.

#### ■ Command frame

LENG-H	LENG-L	CMD	Data[0]
0x0003 or 0x0004		0xBF	TYPE

-Data[0] : TYPE = 0x00, Deselect

TYPE = 0x01, Release

#### ■ Response frame

LENG-H	LENG-L	Resp
N+1		OK

### 3.10.3 LLC Transceive PDU (Command = 0xB0, \_LLC\_Transceive\_PDU)

This command is only used to send and receive LLC PDU by local LLC configured as a initiator to remote LLC as a target after MAC activation. Detailed descriptions of each LLC PDU are following.

*Please use \_LLC\_GET\_PDU and \_LLC\_SET\_PDU commands when local LLC is configured as a target during MAC activation.*

#### ■ Command frame

LENG-H	LENG-L	CMD	Data[0]	Data[1]	Data[2] ~ Data[n-1]
N+1		0xB0	DIR/MODE	PTYPE	[Data to be sent]

-Data[0] : DIR/MODE = Direction or Mode of the LLC PDU.

This byte is used as MODE(SDP or SNEP) in case of LLC CONNECT

PDU and used to indicate direction (DIR = Inbound or Outbound) in other LLC PDUs

DIR : 0x01 = INBOUND, 0x00 = OUTBOUND

MODE : 0x01 = SDP, 0x04 = SNEP

-Data[1] : PTYPE = Type of LLC PDU

: 0x00 = SYMM PDU

0x01 = PAX PDU

0x02 = AGF PDU

0x03 = UI PDU

0x04 = CONNECT PDU

0x05 = DISC PDU (In case of forced DISC, please use values except 0x00, 0x01 as DIR)

0x06 = CC PDU

0x07 = DM PDU

0x08 = FRMR PDU

0x09 = SNL PDU

0x0C = I PDU

0x0D = RR PDU

0x0E = RNR PDU

-Data[2] ~ Data[n-1] : Optional LLC parameters lists with the LLC PDU to be used or Data to be sent.

#### ■ Response frame

LENG-H	LENG-L	Resp	Data[0]	Data[1]	Data[2]	Data[3] ~ Data[n-1]
N+1		OK	PTYPE	DSAP	SSAP	LLC Response Data

-Data[0] : PTYPE = Type of LLC PDU received from remote LLC (Only bit6~bit0).

Bit7 of PTYPE is the direction of the link connection. (0 = outbound, 1 = inbound)

‘Inbound’ means a link arriving at local LLC from remote LLC,

‘Outbound’ means a link departing from local LLC to remote LLC

-Data[1] : DSAP

-Data[2] : SSAP

-Data[3] ~ Data[n-1] : LLC Response data from remote LLC

Ex) Example for PUSH Information to remote LLC(as target) at local LLC(as initiator) using DualCard

1 => BE700100 (MAC activation as passive initiator,212K)

2 <= 0001FE83FE89BF56446CF301011103020013040196 (NFCID3t + LLC Parameters)

3 => B00404 (CONNECT CMD\_PDU for SNEP outbound)

4 <= 00000000 (SYMM RES\_PDU)

5 => B00000 (SYMM CMD\_PDU)

6 <= 0006200402020078 (CC RES\_PDU with LLC Parameters List)

7 => B00000 (SYMM CMD\_PDU)

8 <= 00000000 (SYMM RES\_PDU)

9 => B0000C100200000012D1010E54026B6F0A31323334353637383930

(I CMD\_PDU for SNEP PUT Request)

10 <= 000D200401 (RR RES\_PDU)

11 => B00000 (SYMM CMD\_PDU)

- 12 <= 000C2004108100000000 (I RES\_CMD for SNEP SUCCESS)  
13 => B0000D (RR CMD\_PDU)  
14 <= 00000000 (SYMM RES\_PDU)  
15 => B00005 (DISC CMD\_PDU for outbound link)  
16 <= 0007200400 (DM RES\_PDU with reason code)  
17 => B00000 (SYMM CMD\_PDU)  
18 <= 00000000 (SYMM RES\_PDU)  
19 => BF01 (MAC Deactivation, Release)  
20 <= 00

### 3.10.4 LLC Get PDU (Command = 0xB5, \_LLC\_GET\_PDU)

This command is used to receive LLC PDU from remote LLC configured as initiator when the local LLC is configured as target. Service user can know the link direction by comparing DSAP and SSAP whether the direction of logical connection is inbound or outbound.

#### ■ Command frame

LENG-H	LENG-L	CMD
0x0001		0xB5

#### ■ Response frame

LENG-H	LENG-L	Resp	Data[0]	Data[1]	Data[2]	Data[3] ...Data[n-1]
N+1		OK	PTYPE	DSAP	SSAP	PDU

-Data[0] : PTYPE = Type of LLC PDU received from remote LLC(Only bit6~bit0).

Bit7 of PTYPE has no meaning for this case. (0 = inbound, 1 = outbound)

-Data[1] : DSAP

-Data[2] : SSAP

-Data[3] ... Data[n-1] : PDU = Information field of LLC PDU

### 3.10.5 LLC Set PDU (Command = 0xB6, \_LLC\_SET\_PDU)

This command is used to send LLC PDU to remote LLC when the local LLC is configured as target.

#### ■ Command frame

LENG-H	LENG-L	CMD	Data[0]	Data[1]	Data[2]	Data[3] ...Data[n-1]
0x0002		0xB6	MODE	DIR	PTYPE	PDU

-Data[0] :

In case of CONNECT PDU : MODE = 0x01(SDP), 0x04(SNEP)

- Data[1] : DIR = 0, Outbound (Outgoing link at local LLC)  
DIR = 1, Inbound (Ingoing link at local LLC)  
DIR = 2, Forced Disconnect in case of DISC PDU
- Data[2] : PTYPE = Type of LLC PDU being sent to remote LLC
- Data[3] ... Data[n-1] : PDU = Information field of LLC PDU

#### ■ Response frame

LENG-H	LENG-L	Resp
0x0001		OK

Ex) Example for PUSH Information to remote LLC(as initiator) at local LLC(as target) using DualCard.

- 1 =>BE740000F0 (MAC activation as target)
- 2 <=00D621275BCBB98727AE8001011003020001040196 (NFCID3i + LLC Parameters)
- 3 =>B5 (Get LLC PDU)
- 4 <=00840121060F636F6D2E616E64726F69642E6E7070 (Receive LLC CONNECT)
- 5 =>B60004 (Send LLC CONNECT PDU )
- 6 <=00
- 7 =>B5 (Get LLC PDU)
- 8 <=00062F1002020078 (Receive LLC CC PDU with parameter)
- 9 =>B6000C01000000010100000010D1010C54026B6FED858CEC8AA4ED8AB8  
(Send LLC I PDU to push text data to remote LLC based on NDEF message)
- 10 <=00
- 11 =>B5 (Get LLC PDU)
- 12 <=000D2F1001 (Receive LLC RR PDU)
- 13 =>B60005 (Send LLC DISC PDU)
- 14 <=00
- 15 =>B5 (Get LLC PDU)
- 16 <=00072F1000 (Receive LLC DM PDU with reason byte)

### 3.10.6 Start Tag Emulation (Command = 0xBB, \_LLC\_TAG\_EMU)

This command is used to emulate NFC type 2 tag with maximum 1408 memory size. The reader acts as NFC type 2 tag. User can read data from tag or write data to tag. Only READ, WRITE, SELECT SECTOR commands are supported in this mode.

#### ■ Command frame



LENG-H	LENG-L	CMD
N+1		0xBB

#### ■ Response frame

LENG-H	LENG-L	Resp
N+1		OK

### 3.10.6 Connection-oriented LLC Batch (Command = 0xBC, \_LLC\_CO\_BATCH)

This command is connection-oriented batch command to send Text-type or URI-type NDEF message or NDEF formatted data to remote LLC configured as target or initiator when the local LLC is configured as initiator or target. All LLC-related commands are automatically executed in the reader. The received frames are NDEF formatted data from the remote LLC without any data processing. It is necessary to know NDEF format message to understand the received data. To know more information NDEF message, please refer to NFC Forum specifications. When RTD TYPE is 15(0Fh), NDEF frame shall be done at host side. In this case, the device will just add SNEP or NPP header to NDEF formatted data. The device can support SNEP and SDP protocols automatically.

#### ■ Command frame

LENG-H	LENG-L	CMD	Data[0]	Data[1] ...Data[n-1]
N+1		0xBC	TYPE	Text or URI or NDEF formatted data

- Bit Definitions in TYPE as Initiator, Data[0]

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0	0	DeviceMode	0	RTD TYPE			

Bit7=0

Bit6=0

Bit5= Device Mode : Set to 0, NFC device will be configured as a Initiator.

Set to 1, NFC device will be configured as a Target.

Bit4=0

Bit3~Bit0 : RTD TYPE : 0=Text Type Data

1=URI Type Data (01h format only, example data = "duali.com")

2=Smart Poster

3=vCard

15(0Fh) = Transparent Mode (NDEF data must be done at host side)

#### ■ Response frame

LENG-H	LENG-L	Resp	Data[0] ... Data[n-1]
N+1		OK	NDEF-formatted data (optional)

Ex) Example for sending text data ('0' ~ '9') to remote LLC

1 => BC0030313233343536373839 (Send data as Initiator)

2 <= 00

3 => BC2030313233343536373839 (Send data as Target)

4 <= 00

### 3.11 Get Device Data Commands (0x0A)

This command can be used to get data of Barcode or Mag Stripe reader. This command is supportable only when QR reader is included in the reader.

#### 3.11.1 Get QR Data (Command = 0x0A00, \_DE\_GET\_DATA)

This command can be used to get Barcode or QR data .

##### ■ Command frame

LENG-H	LENG-L	CMD	Data[0]	Data[1]
N+1		0x0A	0x00	Time in 100mSecond

##### ■ Response frame

LENG-H	LENG-L	Resp	Data[0] ... Data[n-1]
N+1		OK	Barcode or QR data

## 4. Response Code Definition

Received correct response from card	: 0 (0x00)
No response from card	: 2 (0x02)
Wrong CRC was transmitted from card	: 3 (0x03)
Smart card is not inserted	: 4 (0x04)
MIFARE card key authentication error	: 5 (0x05)
Smart card is in turned off state	: 5 (0x05)
Wrong parity bit was transmitted from type-A card	: 6 (0x06)
Wrong command code was transmitted from host	: 7 (0x07)
Check byte of UID is wrong	: 8 (0x08)
This command is not authenticated	: 10 (0x0A)
Bit count of received data is incorrect	: 11 (0x0B)
More or less data from protocol was sent from card	: 12 (0x0C)
Error occurred when it write data to MIFARE card	: 15 (0x0F)
Error occurred when it increment data to MIFARE card	: 16 (0x10)
Error occurred when it decrement data to MIFARE card	: 17 (0x11)
Error occurred when it read FeliCa card	: 18 (0x12)
FIFO was overflowed when receive data from card	: 19 (0x13)
Received data is out of correct frame (protocol)	: 21 (0x15)
Unsupported command was sent from host	: 23 (0x17)
Collision was detected when receive data from card	: 24 (0x18)
Communication with RF chip is unable, RF chip error	: 26 (0x19)
Chaining retry overflowed limited count	: 33 (0x21)
ACK was received for deselect command	: 34 (0x22)
Retry count overflowed maximum limit	: 35 (0x23)
Receive buffer is smaller than expected data length from card	: 49 (0x31)
More data than receive buffer was transmitted from card	: 50 (0x32)
Wrong data from protocol was received or transmitted	: 52 (0x34)
RF command is not supported when contact card exists	: 64 (0x40)
Other response was received when ACK is supposed from card	: 65 (0x41)
NAK was received when waiting for other response from card	: 66 (0x42)
Temperature error	: 78 (0x4C)
Action for this command is not implemented yet	: 100(0x64)
Error occurred when write data to FIFO	: 109(0x6D)
Data beyond limit was sent from host	: 123(0x7B)
Read value from card is different from written value	: 123(0x7B)
Card responded value error for the command	: 124(0x7C)

## (NFC Related Codes)

RF is not in ready state to communicate with card (NFC)	: 51 (0x33)
Invalid data from protocol format was sent from host (NFC)	: 55 (0x37)
Wrong parameter from protocol format was sent from host (NFC)	: 60 (0x3C)
Invalid parameter from protocol format was sent from host (NFC)	: 61 (0x3D)
Unsupported NFC command was sent from host (NFC)	: 63 (0x3F)
User mode values and register mode values are different	: 0x90
User speed and register speed are different	: 0x91
Protocols between two nfc devices are abnormal	: 0x92
There is no response until time-out value	: 0x93
Transmission failed until retry number is over 3 times	: 0x94
Scope data value is out of ranges defined in the specifications	: 0x95
Trying to connect each other during disconnected mode	: 0x96
Two NFC devices are using different DID settings	: 0x97
Target didn't get Information PDU during Get_Target Command	: 0x98
MAC activation error when connect LLC communication	: 0xA0
There is no response to LLC connect command	: 0xA1
There is no response to LLC symmetry command	: 0xA2
There is no response to LLC information command	: 0xA3
There is no response to LLC receive ready command	: 0xA4
There is no response to LLC disconnect command	: 0xA5
MAC deactivation error when close LLC	: 0xA6
Access Conditions, Data Size and Version are not different from defined value	: 0xA7
Checksum values in Attribute Block of Type 3 tag are wrong	: 0xA8
There is no data from remote LLC in a target mode	: 0xA9
There is error during sending PDU to remote LLC	: 0xAA
Magic Number Error	: 0xAB
Length error in LLC PDU format	: 0xAC