



NDNF Tool V1.0

SYSTEM NDNF FORMAT

FUNCTION

ACTIVE CHECK NDNF FORMAT NDNF ERASE NDNF

URL WIFI BLUETOOTH TEXT

URL Identifier Code

URL

READ&WRITE

NDNF

DATA(ASCII)

NDNF

PROTOCOL SCREEN

```
>> 50 00 00 C7 97
<< 50 00 00 C7 97 -success
>> 50 00 14 C5 00 89 04 69 64 74 72 6F 6E 69 63 2D 72 66 69 64 2E 63 6F 6D 77
<< 50 00 16 C3 00 1B D1 01 12 55 00 69 64 74 72 6F 6E 69 63 2D 72 66 69 64 2E 63 6F 6D 77 -success
>> 50 00 16 C3 D1 01 12 55 04 69 64 74 72 6F 6E 69 63 2D 72 66 69 64 2E 63 6F 6D 6D
<< 50 00 00 C3 93 -success
>> 50 00 01 C2 00 99
<< 50 00 16 C2 D1 01 12 55 04 69 64 74 72 6F 6E 69 63 2D 72 66 69 64 2E 63 6F 6D 6C -success
```

NEO2 ISO14443A/B, NFC NDEF 13.56 MHz OEM RFID Module Communication Protocol

iDTRONIC GmbH
Ludwig-Reichling-Straße 4
67059 Ludwigshafen
Germany/Deutschland

Issue 0.3
– 23. November 2023 –

Phone: +49 621 6690094-0
Fax: +49 621 6690094-9
E-Mail: info@idtronic.de
Web: idtronic.de

Subject to alteration without prior notice.
© Copyright iDTRONIC GmbH 2023
Printed in Germany

Contents

1	General.....	5
1.1	Glossary.....	5
1.2	Telegram Format	5
1.3	General dialog structure.....	5
1.3.1	Successful command.....	6
1.3.2	Unsuccessful command	6
1.3.3	Answer with an acknowledge: (power_off, idle_mode, power_down_mode)	6
1.4	UART Interface	6
2	Command Set.....	7
3	Error Code List.....	8
3.1	List of possible error code (Reader, System command)	8
3.2	Other Error Code	8
4	COMMANDS DESCRIPTION	10
4.1	System Commands	10
4.1.1	SET_UR_BAUDRATE (0x01)	10
4.1.2	SET_BUZZER (0x02)	10
4.1.3	GetSoftwareVS (0x04).....	10
4.1.4	GetReaderUID (0x05).....	11
4.2	ISO14443A Commands	11
4.2.1	PICCSELECT(0x13)	11
4.2.2	PICCAUTHKEY (0x16).....	11
4.2.3	PICCREAD_A (0x17).....	12
4.2.4	PICCWRITE_A (0x18)	12
4.2.5	PICCWRITE_UL (MIFARE Ultralight) (0x19)	13
4.2.6	PICCRESET (0x21)	13
4.2.7	PICCACTIVATE (0x22)	13
4.2.8	AUTOLISTCARD(CMD = 0x23)	14
4.2.8.1	Automatic Reporting UART Message Format.....	15
4.2.8.2	List of Data Fields.....	15
4.2.9	PICCRMULREAD_A (CMD=0x27)	16
4.2.10	PICCRMULWRITE_A(CMD = 0x28)	18
4.2.11	PICCRATS (0x2A)	19
4.2.12	PICCAPDU (0x2C)	19
4.2.13	PICCTransfer (0x2E)	20
4.3	ISO14443B Commands	21
4.3.1	PICCACTIVATE_B (0x41)	21
4.4	NDEF Commands.....	22
4.4.1	phalTop Custom Error Codes	22
4.4.2	Configuration types	22
4.4.2.1	Use these Macro in phalTop_SetConfig to configure Tags.....	22
4.4.2.2	Set the Card Type	23
4.4.3	TOP_CHECK_NDEF (CMD = 0xC1)	23
4.4.4	TOP_READ_NDEF(CMD = 0xC2)	24
4.4.5	TOP_WRITE_NDEF (CMD = 0xC3)	25
4.4.6	TOP_RESET (CMD = 0xC4)	25
4.4.7	TOP_SET_CONFIG (CMD = 0xC5)	26

4.4.8 TOP_ERASE_NDEF(CMD = 0xC7) 28

4.4.9 TOP_FORMAT_NDEF (CMD = 0xC8) 28

1 General

1.1 Glossary

ATQ	Answer To request
MFC	Mifare Classic
MFP	Mifare Plus
MLE	Maximum R-APDU data size.
MLC	Maximum C-APDU data size.
NFC	Near Field Communication
NDEF	NFC Data Exchange Format
SAK	Select Acknowledge
T1T	Type 1 Tag
T2T	Type 2 Tag
T4T	Type 4 Tag
T5T	Type 5 Tag
TLV	Tag Length Value
TMS	Tag Memory Size
UID	Unique IDentifier

1.2 Telegram Format

The communication between the host controller and the reader obeys to a protocol named PARA. This protocol encapsulates the useful data of a message in an invariant frame structure and defines a dialog structure of messages exchanges.

Frame structure

Data is exchanged between the host controller and the reader in blocks, each made up of binary characters on one byte:

4 bytes	0 to 506 bytes	1 byte
Header characters	Data	XOR
	Information field	Checksum

4 bytes header byte includes:*

1 st byte	2 nd byte	3 rd byte	4 th byte
A 1 A 1 0 0 0 0			
	Data length to be transmitted excluding header and XOR	Command byte	

Remark: A = 0, ACKnowledge of the frame (1st byte = 50)
 A = 1, NACK of the frame (message with a status error, 1st byte = F0)

XOR byte: is such that the exclusive-oring of all bytes including XOR is null.

1.3 General dialog structure

The host controller is the master for the transmission; each command from the master is followed by an answer from the reader including the same command byte as the input command.

However, in some cases (card insertion or extraction, time out detection on Rx line or an automatic emergency deactivation of the card) the reader is able to initiate an exchange.

1.3.1 Successful command

System to Reader

50	XX XX	YY	Nnnnnnnnnnnnnnnnnnnnn	ZZ
ACK	Length	CMD	Data	XOR

Reader to System:

50	UU UU	YY	mmmmmmmmmmmmmmmmmmmm	ZZ
ACK	Length	CMD	Data	XOR

The same command byte YY is returned in the answer from the reader.

1.3.2 Unsuccessful command

System to Reader

50	XX XX	YY	nnnnnnnnnnnnnnnnnnnn	ZZ
ACK	Length	CMD	Data	XOR

Reader to System

F0	UU UU	YY	SS	TT
ACK	Length	CMD	Status	XOR

In that case, the status contains the error code information (see error list).

1.3.3 Answer with an acknowledge: (power_off, idle_mode, power_down_mode)

System to Reader (example: PiccHalt)

50	00 00	14	44
ACK	Length	CMD	XOR

Reader to System:

50	00 00	14	44
ACK	Length	CMD	XOR

In the case where the answer is an acknowledge of the command, the reader sends back a frame with the same content of the command.

1.4 UART Interface

The serial interface between the Reader and the host controller is a full duplex interface using the two lines RX and TX.

RX is used to receive data from the host controller;

TX is used to send data to the host controller.

No flow control or supplementary line is used (no hand check).

The serial data format used is:

1	Start bit
8	Data bit
1	Stop bit, no parity
Default baudrate	115200 bps

2 Command Set

The following command bytes are available (listed in numerical order):

Command	Code
System	
SET_UR_BAUDRATE	0x01
SET_BUZZER	0x02
ISO 14443A (MIFARE Classic&Ultralight&NTAG)	
PICCREAD_A	0x17
PICCWRITE_UL	0x19
PICCRMULREAD_A	0x27
PICCRMULWRITE_A	0x28
PICCACTIVATE	0x22
PICCRATS	0x2A
PICCAPDU	0x2C
PICCTransfer	0x2E
ISO 14443B	
PICCACTIVATE_B	0x41
ISO 15693	
I2_INVENTORY	0xA1
I2_READ_BLOCK	0xA3
I2_WRITE_BLOCK	0xA4
NDEF Commands	
TOP_CHECK_NDEF	0xC1
TOP_READ_NDEF	0xC2
TOP_WRITE_NDEF	0xC3
TOP_RESET	0xC4
TOP_SET_CONFIG	0xC5
TOP_ERASE_NDEF	0xC7
TOP_FORMAT_NDEF	0xC8

3 Error Code List

3.1 List of possible error code (Reader, System command)

Status code	Description
0xF1	LRC error
0xF2	NO THIS CMD
0xF3	SET_ERROR
0xF4	PARA_ERROR
0xB1	NO_CARD
0xB2	ANTICOLL_ERROR
0xB3	SELECT_ERROR
0xB4	HALT_ERROR
0xB6	AUTH_ERROR
0xB7	READ_ERROR
0xB8	WRITE_ERROR
0xB9	VALUEOPER_ERROR
0xBA	VALUEBAK_ERROR
0xBC	RATS_ERROR
0xBE	TPCL_ERROR
0xD1	POWERUP_ERROR
0xD2	POWEROFF_ERROR
0xD3	APDU_ERROR
0xD4	PTS_ERROR
0xD5	NO_SLOT
0xD6	CHACK_ERROR

3.2 Other Error Code

Board IF Err		
0x10	TIMEOUT_RECEIVE	No input data is received within given time, time defined
0x11	LRC_ERR	Verification of input Package checksum is failed
0x12	RX_BUFFER_FULL	Interface buffer is already full
Board Function Err		
0x30	WRITE_HARDWARE_PARAM_FAIL	Writing hardware parameter in IC's EEPROM Fail
0x31	CHECKSUM_HARDWARE_PARAM_FAIL	Checksum hardware parameter failed
0x32	HARDWARE_PARAM	
Communication Protocol Err		
0x20	UNKNOWN_CMD_TYPE	Input command category is undefined
0x21	UNKNOWN_CMD	Input command is undefined
0x22	PARAMETER_NOT_CORRECT	Parameter is incomplete or invalid
ISO14443A Err		
0xA0	A_HALT_ERR	Error if there is a response after sending Halt command
0xA1	AUTHENT_ERR	Error if Crytol bit in Control register(Reg 0x09)is not set after preforming AUTHENT command
0xA2	NOT_AUTHENT	Error from Operating MIFARE command, i.e. Increment when cryptol bit is not set
0xA3	MIFARE_ERR	NACK (0x04 or 0x05) from MIFARE card is received
FELICA Err		

0xC0	FELICA_RESP_CODE_ERR	Response code mismatched
RF Communication Err		
0xE0	NO_RESPONSE	No card response within given time indicating by timeout from ASIC Timer
0xE1	FRAMING_ERR	Format of receive frame errors indicating by FramingErr bit in SIC9xx's ErrorFlag register (Reg 0x0A)
0xE2	COLLISION_ERR	Bit collision is detected indicating by CollErr bit in IC's ErrorFlag register (Reg 0x0A)
0xE3	PARITY_ERR	Parity Bit Check is invalid indicating by ParityErr bit in IC's ErrorFlag register (Reg 0x0A)
0xE4	CRC_ERR	CRC Check is invalid indicating by CRCErr bit in IC's ErrorFlag register (Reg 0x0A)
0xE5	INVALID_RESP	Response is invalid or unexpected from operation protocol
0xE6	SUBC_DET_ERR	Subcarrier from card is detected indicating by SubC_Det bit in IC's Status register (Reg 0x05); but cannot recognized following standard(available only x410)

4 COMMANDS DESCRIPTION

4.1 System Commands

4.1.1 SET_UR_BAUDRATE (0x01)

Note: The new Baud rate will only be taken over after a cold boot.

```
int SetUARTBaudRate(    unsigned char ucRates);
```

-----DLL Explanation-----

ucRates:	Parameter only	
	Parameter	Baud rate (Baud)
	04	9600
	03	19200
	02	38400
	01	57600
	00	115200

Return: 0(OK) or Error Code

-----Protocol Example-----

>> 50 00 01 01 01 51 (Set to 57600 Baud)

<< 50 00 01 01 01 51 (Return in old Baud rate, and then the new one will be initialized)

4.1.2 SET_BUZZER (0x02)

```
int SetBuzzer(    unsigned char ucRates,
                  unsigned char ucTimes);
```

-----DLL Explanation-----

ucRates:	beep keeping times will be $ucRates * 50$ ms and silence $(500 - ucRates * 50)$ ms
ucTimes:	beep ucTimes times.

Return: 0(OK) or Error Code

-----Protocol Example-----

>> 50 00 02 02 03 04 57 (beep 4 times, every beep keep sound 150ms and silence 350ms)

<< 50 00 00 02 52

4.1.3 GetSoftwareVS (0x04)

```
int GetSoftwareVS(unsigned char *relen,unsigned char *reVS);
```

-----DLL Explanation-----

*relen: Version length

*reuid: Version return

Return: 0(OK) or Error Code

-----Protocol Example-----

>> 50 00 00 04 54

<< 50 00 04 04 **72 18 07 24** 19

4.1.4 GetReaderUID (0x05)

```
int __stdcall GetReaderUID(    unsigned char *relen,
                             unsigned char *reuid)
```

-----DLL Explanation-----

Return: 0(OK) or Error Code

-----Protocol Example-----

>> 50 00 00 05 55

<< 50 00 0C 05 4F 7A 6A 16 68 98 0D 5A 74 12 75 77 59

Remark: 4F 7A 6A 16 68 98 0D 5A 74 12 75 77 is the reader UID

4.2 ISO14443A Commands

4.2.1 PICCSELECT(0x13)

This command is not needed in most situation.

unsigned char PiccSelect (unsigned char mode)

-----DLL Explanation-----

```
Mode:          #define INTF_UNDETERMINED  0x0
                #define INTF_FRAME        0x1 //
                #define INTF_ISODEP       0x2
                #define INTF_NFCDEP      0x3
                #define INTF_NFCDEP      0x3
                #define INTF_TAGCMD      0x80 //mifare
```

Return: 0(OK) or Error Code

This command is used to select the RFID tag for further operation.

-----Protocol Example-----

>> 50 00 01 13 80 C2 (Select mifare)

<<50 00 00 13 43

4.2.2 PICCAUTHKEY (0x16)

```
int PiccAuthKey(    unsigned char auth_mode,
                   unsigned char addr,
                   unsigned char *pSnr,
                   unsigned char *pKey)
```

-----DLL Explanation-----

```
auth_mode:      0x60 --KeyA
                0x61 --KeyB
addr:           block number (1 byte)
                S50:           0~63
                S70:           0~255
                PLUS CPU (2K):  0~127
                PLUS CPU (4K):  0~255
```

pSnr: card serial number (4 bytes)
if card Serial No more than 4 bytes, only 4 MSB needed

*pKey: 6 bytes key

Return: 0(OK) or Error Code

-----Protocol Example-----

```
>>50 00 0C 16 60 04 1D B7 60 57 FF FF FF FF FF FF B3
    (authenticate the card(SN=1D B7 60 57) of 0x04 block(0x01 sector) using keyA(0x60)
    with 6bytes key(0xFF,0xFF,0xFF,0xFF,0xFF,0xFF))
<<50 00 00 16 46
```

4.2.3 PICCREAD_A (0x17)

int PiccRead(unsigned char ucBlock,
 unsigned char *pBuf)

-----DLL Explanation -----

ucBlock: Block number (1byte)

S50:	0~63
S70:	0~255
PLUS CPU (2K):	0~127
PLUS CPU (4K):	0~255

*pBuf: Block data (16 bytes)

Return: 0(OK) or Error Code

-----Protocol Example-----

```
>>50 00 01 17 04 42(Read 0x04 block)
<<50 00 10 17 05 05 05 05 05 05 05 05 05 05 05 05 05 05 57
```

4.2.4 PICCWRITE_A (0x18)

int PiccWrite(unsigned char ucBlock,
 unsigned char *pBuf)

-----DLL Explanation -----

ucBlock: Block number (1 byte)

S50:	0~63
S70:	0~255
PLUS CPU (2K):	0~127
PLUS CPU (4K):	0~255

*pBuf: Data to write (16 bytes)

After one block have been authenticated successfully, the other block in the same sector need no authentication

Return: 0(OK) or Error Code

-----Protocol Example-----

```
>>50 00 11 18 04 05 05 05 05 05 05 05 05 05 05 05 05 05 5D (Write 0x04 block to 16 bytes 0x05)
<<50 00 00 18 48
```

4.2.5 PICCWRITE_UL (MIFARE Ultralight) (0x19)

```
int PiccULWrite(      unsigned char ucBlock,
                     unsigned char *pBuf)
```

-----DLL Explanation-----

```
ucBlock:            Block number (1 byte)
                    S50:                0~63
                    S70:                0~255
                    PLUS CPU (2K):      0~127
                    PLUS CPU (4K):      0~255
*pBuf:              Data to write (4 bytes)
```

Return: 0(OK) or Error Code

-----Protocol Example-----

```
>> 50 00 05 19 04 05 05 05 48   (Write 0x04 block to 4bytes 0x05)
<< 50 00 00 19 49
```

4.2.6 PICCRESET (0x21)

```
Int PiccReset(      unsigned char _ms)
```

-----DLL Explanation-----

Input parameter:

_1ms: reset the antenna (1 byte) in X ms

This means the antenna will be closed in X ms, then, and then reopened
0 is to keep antenna closed

Return: 0(OK) or Error Code

-----Protocol Example-----

```
>> 50 00 01 21 05 75
<< 50 00 00 21 71
```

NOTE: this command is used to save power or deactivate the card in field.

4.2.7 PICCACTIVATE (0x22)

```
int PiccActivate(    unsigned char ucRst_1ms,
                    unsigned char ucReqCode,
                    unsigned char *pATQ,
                    unsigned char *pSAK,
                    unsigned char *pUIDLen,
                    unsigned char *pUID)
```

-----DLL Explanation-----

Input parameter:

ucRst_1ms:

Refer to PiccReset command, if set, the antenna will close for ucRst_1ms(ms) first and then Open

ucReqCode:0x26 IDLE,0x52 ALL

Output variables:

*pATQ:Answer to request (ATQ) (2bytes)

*pSAK:Select acknowledge (SAK) (1byte)

*pUIDLen:UID length (1byte)

*pUID:UID (4 bytes or 7bytes most)

Return: 0(OK) or Error Code

-----Protocol Example-----

>> 50 00 02 22 10 52 32

<< 50 00 08 22 04 00 08 04 1D B7 60 57 EF (ATQ:0400; SAK:0x08; 0x04 bytes UID: 1D B7 60 57)

NOTE: use this command to activate the card before any other command, PICCACTIVATE will run REQA, Anti-collision and Select sequence as defined in ISO/IEC 14443_3 document.

4.2.8 AUTOLISTCARD(CMD = 0x23)

```
int PiccAutoListCard(unsigned char ucType,
                    unsigned char ucMode,
                    unsigned char ucNFC,
                    unsigned char ucNotice,
                    unsigned char ucWait)
```

-----DLL Explanation -----

Input Parameters:

```
Tag Type:      #define TYPE_BIT_MASK_A(ISO14443-A)      0x01
               #define TYPE_BIT_MASK_B(ISO14443-B)      0x02
               #define TYPE_BIT_MASK_V(ISO15693)         0x04
               #define TYPE_BIT_MASK_F(SONY FELICA)      0x08
               #define TYPE_BIT_MASK_C(CHINESE ID)       0x10
               #define TYPE_BIT_MASK_M(FUDAN FM1208)     0x20
```

Card type: (8 bit)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		FM1208	Chinese ID	SONY Felica	ISO15693	ISO14443B	ISO14443B

Examples

- TYPE 0x01 = 0b0000.0001: ISO14443 A card only
- TYPE 0x04 = 0b0000.0100: ISO15693 Only
- TYPE 0x05 = 0b0000.0101: ISO15693 + ISO14443 A
- TYPE 0xFF: All card types the module supports

Notes

This module does not support ISO15693.

4.2.8.1 Automatic Reporting UART Message Format

SOF	LEN0	LEN1	CMD	TYPE	MODE	NFC	NOTICE	WAIT	INFOR	EOF
50	00	XX	23	00...FF	00...FF	00...FF	01...04	00	XX...	XOR

4.2.8.2 List of Data Fields

MODE: default enable Read/Write mode and Card EMU mode, set 0x05

/*

* Flag definition used for NFC library configuration

*/

```
#define NXPNCI_MODE_CARDEMU      (1<<0)
```

```
#define NXPNCI_MODE_P2P          (1<<1)
```

```
#define NXPNCI_MODE_RW           (1<<2)
```

NFC: (default enable LISTEN_PASSIVE_NFCA, set 0x04)

```
#define POLL_ACTIVE_NFCA        0x01
```

```
#define POLL_ACTIVE_NFCF        0x02
```

```
#define LISTEN_PASSIVE_NFCA      0x04  (default setting)
```

```
#define LISTEN_PASSIVE_NFCB      0x08
```

```
#define LISTEN_PASSIVE_NFCF      0x10
```

```
#define LISTEN_ACTIVE_NFCA       0x20
```

```
#define LISTEN_ACTIVE_NFCF       0x40
```

NOTICE: (default support NDEF R/W and UART output protocol like 50/F0... data bytes, default HEX value 0xD0)

bit7 and bit6:(when this module in EMU mode, will support NDEF read and write)

```
#define AR_NDEF_RxEnBitMask      0x80
```

```
#define AR_NDEF_TxEnBitMask      0x40
```

Bit5 and bit4 defined UART output mode (example:01- Protocol output but not law UID only)

```
#define AR_UartOUTModeRawUID      0x00
```

```
#define AR_UartOUTModeProtocol    0x10
```

```
#define AR_UartOUTModeRegistUID   0x20
```

```
#define AR_UartOUTModeRegistCMD   0x30
```

When bit5 and bit4 set to 01,then Bit3 and bit2 executed like this:

```
#define AR_UartProtocal50F0      0x00  (default)
```

```
#define AR_UartProtocalAABB      0x01
```

```
#define AR_UartProtocal0203      0x10
```

Bit1 and bit0: RFU

(The function is to register 4 groups of commands into the module and let the module automatically execute these commands when conditions are appropriate)

WAIT: (default 0x01, 1 second)

The reader stop autolist-card function in this time, and waits for another UART command

Output variables

NULL

Return: 0(OK) or Error Code

Protocol Example

```
>> 50 00 05 23 01 05 04 D0 01 A7 //EmuA, read typeA, out protocol data
<< 50 00 00 23 73 (ACK of setting, but not the card reporting message)
```

After this command, if an ISO14443A card can be detected, the RFID device will automatically output this telegram:

```
<< 50 00 10 23 01 01 00 D0 01 44 03 20 07 04 28 69 9A 4F 22 80 E0
```

The information in detail

01 01 00 D0 01	The first 5 bytes are mirrored from the setup of the auto-list cards operation mode
4403	ATQ (Answer To request)
20	SAK (Select Acknowledge)
07	UID length
04 28 69 9A 4F 22 80	UID (Unique Identifier)

4.2.9 PICCRMULREAD_A (CMD=0x27)

This combines the command PICCACTIVATE, PICCAUTHKEY and PICCREAD_A into a powerful macro-command. It will only operate with Mifare Classic tags or emulations.

```
int PiccITGRead(
    unsigned char rst_ms,
    unsigned char req_mode,
    unsigned char sector_begin,
    unsigned char sector_much,
    unsigned char block_each,
    unsigned char auth_mode,
    unsigned char *pKey,
    unsigned char *pBuf);
```

DLL Explanation

rst_ms:	time (1 byte) in ms, 0 will keep antenna close
req_mode:	0x26 IDLE 0x52 ALL
sector_begin:	sector number (1 byte): S20: 0...3 S50: 0...15 S70: 0...32
sector_much:	how much sector to read begin from sector_begin
block_each:	how much blocks will read out of each selected sector
auth_mode:	0x60 --KeyA 0x61 --KeyB
*pKey:	6 bytes key
pBuf:	data read out, Will depend on sector_begin / sector_much / block_each

Return: 0(OK) or Error Code

-----Protocol Example-----

Example1: Start at Sector 02, read from 4 subsequent sectors all 4 Blocks

>> 50 00 0C 27 0A 52 02 04 04 60 FF FF FF FF FF FF 41

<< 50 01 00 27 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF
 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 06 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF
 07 00 00 10 00 00 00 00 00 00 00 00 00 00 00 00
 08 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 09 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF
 0A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF

6A

Example2: Start at Sector 02, read from 4 subsequent sectors 3 Blocks (without trailing block)

>> 50 00 0C 27 0A 52 02 04 03 60 FF FF FF FF FF FF 46

<< 50 00 C0 27 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 11 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 12 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 13 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 23 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 24 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 25 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

85

4.2.10 PICCRMULWRITE_A(CMD = 0x28)**(PICCACTIVATE +PICCAUTHKEY+ PICCWRITE_A)**

```
int PiccITGWrite(
    unsigned char rst_ms,
    unsigned char req_mode,
    unsigned char sector_begin,
    unsigned char sector_much,
    unsigned char block_each,
    unsigned char auth_mode,
    unsigned char *pKey,
    unsigned char *pBuf);
```

-----DLL Explanation -----

rst_ms: time (1 byte) in ms, 0 will keep antenna close

req_mode: 0x26 IDLE
0x52 ALL

sector_begin: sector number (1 byte): S20: 0...4
S50: 0...15
S70: 0...32

sector_much,: how many sectors to write begin from sector_begin

block_each: how much block will write in of each selected sector

auth_mode: 0x60 --KeyA
0x61 --KeyB

*pKey: 6 bytes key

*pBuf: data to write in, Must according to sector_begin / sector_much / block_each

Return: 0(OK) or Error Code

-----Protocol Example-----

Write 4 sectors (3 block each, without sector trailer)

```
>>50 00 CC 28      0A 52 02 04 03 60 FF FF FF FF FF FF
                   11 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                   22 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                   33 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                   44 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                   55 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                   66 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                   77 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                   88 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                   99 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                   AA 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                   BB 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                   CC 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

45

<< 50 00 00 28 78

4.2.11 PICCRATS (0x2A)

```
int PiccRequestATS(      unsigned char ucRFU,
                        unsigned char *pATSLen,
                        unsigned char *pATS )
```

-----DLL Explanation-----

Input parameter:

ucRFU: Set to 0x00

Output variables:

* pATSLen:Length of the ATS from the card

*pATS:ATS (answer to select)

Return: 0(OK) or Error Code

-----Protocol Example-----

Send >>50 00 00 2A 7A

Return <<50 00 10 2A 10 78 80 90 02 20 90 00 00 00 00 00 F6 D0 B1 26 11

NOTE:

Set ISO14443-3 card into ISO14443-4 mode

4.2.12 PICCAPDU (0x2C)

```
int PiccAPDU(           unsigned int usSendLen,
                        unsigned char *pSendBuf,
                        unsigned int *pRcvLen,
                        unsigned char *pRcvBuf)
```

-----DLL Explanation-----

Input parameter:

usSendLen:length of data to be sent to the card

*pSendBuf:Data to be sent to the card

Output variables:

pRcvLen:length of data received from the card

* pRcvBuf:Data received from the card

Return: 0(OK) or Error Code

-----Protocol Example-----

Send >>50 00 05 2C 00 84 00 00 08 F5

Return <<50 00 0A 2C F2 EB 10 97 2D A5 54 3B 90 00 9F

NOTE:

For ISO14443A Card:

The Card have been RATS and go into ISO14443-4 mode may use this 0x2c command

For ISO14443B Card:

RATS is not need for Type B Card, so after 0x41 command, you may use this 0x2c command.

4.2.13 PICCTransfer (0x2E)

```
int PiccTransfer(          unsigned int usSendLen,  
                        unsigned char *pSendBuf,  
                        unsigned int *pRcvLen,  
                        unsigned char *pRcvBuf)
```

-----DLL Explanation-----

Input parameter:

usSendLen:length of data to be sent to the card

*pSendBuf:Data to be sent to the card

Output variables:

pRcvLen:length of data received from the card

* pRcvBuf:Data received from the card

Return: 0(OK) or Error Code

-----Protocol Example-----

Send >>50 00 07 2E 0A 00 00 84 00 00 08 FF

Return <<50 00 0C 2E 0A 00 F2 EB 10 97 2D A5 54 3B 90 00 91

NOTE:**For ISO14443A Card:**

The Card is activated into ISO14443-3 (have not RATS) mode may use this 0x2E command.

4.3 ISO14443B Commands

4.3.1 PICCACTIVATE_B (0x41)

```
int PiccActivateB(    unsigned char ucRst_1ms,
                    unsigned char ucAFI,
                    unsigned char ucMethod,
                    unsigned char *pUIDLen,
                    unsigned char *pUID);
```

-----DLL Explanation-----

Input parameters:

ucRst_1ms:
Refer to PiccReset command,
if set, the antenna will close for ucRst_1ms(ms) first and then Open

ucAFI:
Application Family Identifier

ucMethod:
1: probabilistic
Others: slot-marker

pUIDLen:
UID length (1 byte)

pUID: will be 0x50 as a preamble, and then 4bytes UID and others

Return: 0(OK) or Error Code

Output variables:

*pUIDLen:UID length (1 byte)

*pUID:UID: 0x50 as a preamble, and then 4bytes UID and others Card information

Return:

0(OK) or Error Code

-----Protocol Example-----

>> 50 00 01 41 00 10

<< 50 00 0C 41 50 BB EE 44 11 11 22 33 11 77 83 C3 6B (UID: BB EE 44 11)

4.4 NDEF Commands

4.4.1 phalTop Custom Error Codes

#define PH_ERR_SUCCESS	0x00U	
#define TOP_ERR_READONLY_TAG	0x81U	/**< Tag is Read Only */
#define TOP_ERR_INVALID_STATE	0x82U	/**< Tag state is invalid */
#define TOP_ERR_FORMATTED_TAG	0x83U	/**< Tag already in NDEF formatted state. */
#define TOP_ERR_UNSUPPORTED_VERSION	0x84U	/**< Unsupported NDEF version. */
#define TOP_ERR_MISCONFIGURED_TAG	0x85U	/**< Tag not configured as per NDEF specification. */
#define TOP_ERR_UNSUPPORTED_TAG	0x86U	/**< Tag with unsupported structure/format. */
#define TOP_ERR_EMPTY_NDEF	0x87U	/**< NDEF message length is zero (i.e. Tag in initialized state). */
#define TOP_ERR_NON_NDEF_TAG	0x88U	/**< Tag is not NDEF formatted. */

4.4.2 Configuration types

4.4.2.1 Use these Macro in phalTop_SetConfig to configure Tags.

#define TOP_CONFIG_TAG_TYPE	0x51U	/**< Set/Get Tag type. Should be configured before calling CheckNdef. */
#define TOP_CONFIG_TAG_STATE	0x52U	/**< Get tag state. This shall be also used to set tag to read-only state. */
#define TOP_CONFIG_NDEF_LENGTH	0x53U	/**< Get current NDEF message Length. */
#define TOP_CONFIG_MAX_NDEF_LENGTH	0x54U	/**< Get Max support NDEF Length by tag. */
#define TOP_CONFIG_NDEF_VERSION	0x55U	/**< Get NDEF Version Number. */
#define TOP_CONFIG_T1T_TMS	0x03U	/**< Set tag memory size. Set before format operation. */
#define TOP_CONFIG_T1T_TERMINATOR_TLV	0x05U	/**< Set Terminator TLV presence. Set before format/write operation to enable writing terminator TLV at end of NDEF TLV. */
#define TOP_CONFIG_T2T_TMS	0x09U	/**< Set tag memory size. Set before format operation. */
#define TOP_CONFIG_T4T_NDEF_FILE_ID	0x15U	/**< Set NDEF file ID. Set before format operation. */
#define TOP_CONFIG_T4T_NDEF_FILE_SIZE	0x18U	/**< Set Max NDEF length. Set before format operation. */
#define TOP_CONFIG_T4T_MLE	0x19U	/**< Set MLe. Set before format operation. */
#define TOP_CONFIG_T4T_MLC	0x1AU	/**< Set MLc. Set before format operation. */
#define TOP_CONFIG_T5T_MLEN	0x1DU	/** < Set T5T NDEF data area. Set before format operation. */
#define TOP_CONFIG_T5T_MBREAD	0x20U	/**< Set T5T multiple block read support. Set before format operation. */
#define TOP_CONFIG_T5T_LOCKBLOCK	0x21U	/**< Set T5T Lock block command support. Set before format operation. */
#define TOP_CONFIG_T5T_SPL_FRM	0x22U	/**< Set T5T special frame support. Set before format operation. */
#define TOP_CONFIG_T5T_OPTION_FLAG	0x26U	/**< Set T5T option Flag. Set before format operation. */
#define TOP_CONFIG_T5T_TERMINATOR_TLV	0x1FU	/**< Set Terminator TLV presence. Set before format/write operation to enable writing terminator TLV at end of NDEF TLV. */
#define TOP_CONFIG_MFCTOP_CARD_TYPE	0x60U	

4.4.2.2 Set the Card Type

Mifare Classic Card Types:

```
#define TOP_NO_MFC           0x00U    /**< No Card is selected */
#define TOP_MFC_1K          0x01U    /**< MFC 1K is selected*/
#define TOP_MFC_4K          0x02U    /**< MFC 4K is selected*/
#define TOP_MFP_2K          0x03U    /**< MFP 2K is selected*/
#define TOP_MFP_4K          0x04U    /**< MFP 4K is selected*/
```

Tag types: Tag types are used to set Tag used using phalTop_SetConfig with TOP_CONFIG_TAG_TYPE Type

```
#define TOP_TAG_TYPE_T1T_TAG 0x01U    /**< Type 1 Tag. */
#define TOP_TAG_TYPE_T2T_TAG 0x02U    /**< Type 2 Tag. */
#define TOP_TAG_TYPE_T3T_TAG 0x03U    /**< Type 3 Tag. */
#define TOP_TAG_TYPE_T4T_TAG 0x04U    /**< Type 4 Tag. */
#define TOP_TAG_TYPE_T5T_TAG 0x05U    /**< Type 5 Tag. */
#define TOP_TAG_TYPE_MFC_TOP 0x06U    /**< MFC as Tag Type */
/*@}*/
```

Tag States:

```
#define TOP_STATE_NONE      0x00U    /**< Default initial state. */
#define TOP_STATE_INITIALIZED 0x01U    /**< Initialized state. */
#define TOP_STATE_READONLY  0x02U    /**< Read Only state. */
#define TOP_STATE_READWRITE 0x04U    /**< Read/Write state. */
```

NOTE: This module support T2T (Type 2 Tag) only

4.4.3 TOP_CHECK_NDEF (CMD = 0xC1)

Does the NDEF detection procedure as per NFC Tag Operation specifications for each tag type.

-----DLL Explanation -----

The caller has to ensure that Tag activation is done before calling this API. Also phalTop_SetConfig should be called before to configure the Tag type.

#TOP_CONFIG_TAG_TYPE. For Mifare Classic as tag type phalTop_SetConfig needs to be called to configure the Mifare Classic card type #TOP_CONFIG_MFCTOP_CARD_TYPE e.g. #TOP_MFC_1K.

Only after phalTop_CheckNdef is called any other NDEF operation on Tag can be performed.

Return Status codes

#PH_ERR_SUCCESS	Operation successful.
#TOP_ERR_NON_NDEF_TAG	Tag don't indicate NDEF presence
#TOP_ERR_UNSUPPORTED_VERSION	Tag indicates NDEF presence but NDEF version mentioned in tag is not supported by reader library.
#TOP_ERR_MISCONFIGURED_TAG	Tag indicates NDEF presence but the NDEF CC or NDEF attribute Information is wrongly configured
#TOP_ERR_UNSUPPORTED_TAG	Tag uses some proprietary or RFU or unsupported configuration

Other Depending on implementation and underlying component.

-----DLL-----

```
int Top_CheckNdef(unsigned char * pTagState)
```

-----Protocol Example-----

```
>> 50 00 00 C1 91
<< F0 00 01 C1 21 11 //ERROR:not config yet
<< F0 00 01 C1 88 B8 //ERROR:TOP_ERR_NON_NDEF_TAG
<< 50 00 01 C1 04 94 //OK:TOP_STATE_READWRITE OK
```

//If return OK, TagState will be:

```
#define TOP_STATE_NONE           0x00U    /**< Default initial state. */
#define TOP_STATE_INITIALIZED    0x01U    /**< Initialized state. */
#define TOP_STATE_READONLY      0x02U    /**< Read Only state. */
#define TOP_STATE_READWRITE     0x04U    /**< Read/Write state. */
```

4.4.4 TOP_READ_NDEF(CMD = 0xC2)

Read NDEF message from Tag.

-----DLL Explanation-----

ReadNdef shall only be called after tag detected returned success. If it is an empty NDEF message(i.e. initialized state) then this will return empty NDEF error.

Return Status Codes

#PH_ERR_SUCCESS	Operation successful.
#TOP_ERR_INVALID_STATE	Tag is not is any valid state (i.e. when check NDEF failed or has not been called before.)
#TOP_ERR_EMPTY_NDEF	Tag is in initialized state (i.e. no NDEF / empty NDEF)

Other Depending on implementation and underlying component.

-----DLL-----

```
Int Top_ReadNdef(
    uint8_t ucDataParams,    /**< [In] Mode,      1: get the stored NDEF data in the reader which have been read,
                                0: will be a new read launch. */
    uint8_t * pData,         /**< [Out] NDEF data from the Tag. User has to allocate memory accordingly */
    uint16_t * pLength       /**< [Out] NDEF data length. */
);
```

-----Protocol Example-----

```
>> 50 00 01 C2 01 92 (get the ndef data from reader RAM)
>> 50 00 00 C2 92 (launch a new read routine)
<< 50 00 13 C2 D1 01 0F 55 00 77 77 77 2E 74 61 6F 62 61 6F 2E 63 6F 6D 0B
(URL:www.taobao.com,14bytes)
(Record HEX: D1 01 10 55 00 77 77 77 2E 74 61 6F 62 61 6F 2E 63 6F 6D)

>> 50 00 00 C2 92
<< 50 00 24 C2 D1 02 1F 53 70 91 01 0E 54 02 65 6E 68 65 6C 6C 6F 20 77 6F 72 6C 64 51 01 09 55 01 73 69 6E 61 2E 63 6F
6D ED
(Text record "hello world" Mix URL record "sina.com")
```


4.4.5 TOP_WRITE_NDEF (CMD = 0xC3)

Write NDEF message into Tag. The Tag should have been formatted before.

-----DLL Explanation-----

WriteNdef shall be used to write a new NDEF message to tag if check NDEF returned success. If tag is in read only state, write NDEF will return error. The tag is expected to be a formatted NDEF tag for this to succeed.

WriteNDEF will update the NDEF message TLV or the capability container with the length of the data written to the tag.

Return Status Codes

#PH_ERR_SUCCESS	Operation successful.
#PH_ERR_INVALID_PARAMETER	wLength is 0 or wLength is more than supported max length.
#TOP_ERR_INVALID_STATE	Tag is not in any valid state (i.e. when check NDEF failed).
#TOP_ERR_READONLY_TAG	Tag is in read only state.

Other Depending on implementation and underlying component.

-----DLL-----

```
Int Top_WriteNdef(
    uint8_t * pData,      /**< [In] NDEF data to be written to tag. User has to allocate memory accordingly */
    uint8_t wLength      /**< [In] Length of NDEF data to be written. */
);
```

-----Protocol Example-----

```
>> 50 00 24 C3 D1 02 1F 53 70 91 01 0E 54 02 65 6E 68 65 6C 6C 6F 20 77 6F 72 6C 64 51 01 09 55 01 73 69 6E 61 2E 63 6F
6D EC
<< 50 00 00 C3 93
```

4.4.6 TOP_RESET (CMD = 0xC4)

Reset Tag Operation parameters

-----DLL Explanation-----

User has to call phalTop_Reset to reset all the Software parameters. This shall be called after performing all NDEF operations if needed.

Return Status Code

#PH_ERR_SUCCESS	Operation successful.
-----------------	-----------------------

-----DLL-----

```
int Top_Reset(void)
```

-----Protocol Example-----

```
>> 50 00 00 C4 94
<< 50 00 00 C4 94
```

4.4.7 TOP_SET_CONFIG (CMD = 0xC5)

Set configuration parameter.

-----DLL Explanation-----

Refer to #TOP_CONFIG_TAG_TYPE from where the configurable parameters are listed.

Return Status Codes

#PH_ERR_SUCCESS	Operation successful.
#PH_ERR_INVALID_PARAMETER	Parameter value is invalid.

-----DLL-----

```
int Top_SetConfig(
    uint16_t wConfig,    /**< [In] Configuration Identifier. */
    Uint8_t *wValue      /**< [In] Configuration Value. */
);
```

-----Protocol Examples-----

Example 1: Set Tag Type

```
>> 50 00 04 C5 00 51 00 02 C2      //Config 0x0051, set tag type 0x0002(ultralight or NTAG2xx)
```

Example 2: Set Tag UID

```
>> 50 00 05 C5 00 79 11 22 33 E9    //Config 0x0079, set tag UID low 3 bytes to 11 22 33 when the module emulates
                                     a Type2 tag, and then the module UID will be 08 11 22 33
                                     ( the first byte will lock to 0x08)
```

Example 3: Set Tag UID

```
>> 50 00 05 C5 00 79 00 00 00 E9    //Config 0x0079, if set tag UID low 3 bytes to 00 00 00,
                                     then the module UID will be in Random mode
```

Example 4: RAW NDEF IN RAM

```
>>50 00 26 C5 00 84 D1 02 1F 53 70 91 01 0E 54 02 65 6E 68 65 6C 6C 6F 20 77 6F 72 6C 64 51 01 09 55 01 73 69 6E 61 2E
63 6F 6D 6C
```

wConfig = 0x0084 is use to set the NDEF file store in the Reader RAM, which will be use in card emulation and P2P communicate to push this NDEF out to the mobile phone-

Example 5: RAW NDEF IN RAM

Write a wifi connecting information in the the RAM of the reader.

```
>>50 00 60 C5 00 84 DA 17 42 01 61 70 70 6C 69 63 61 74 69 6F 6E 2F 76 6E 64 2E 77 66 61 2E 77 73 63 30 10 0E 00 34 10
45 00 0B 42 52 44 2D 52 46 49 44 5F 35 47 10 20 00 06 34 96 72 FF 43 8D 10 27 00 0B 31 33 38 30 30 31 33 38 30 30 30 10
03 00 02 00 20 10 0F 00 02 00 08 10 49 00 06 00 37 2A 00 01 20 54
```

Example 6: RAW NDEF IN FLASH

```
>>50 00 26 C5 00 85 D1 02 1F 53 70 91 01 0E 54 02 65 6E 68 65 6C 6C 6F 20 77 6F 72 6C 64 51 01 09 55 01 73 69 6E 61 2E
63 6F 6D 6D
```

wConfig = 0x0085 is use to set the NDEF file store in the Reader Flash, which will be use in card emulation and P2P communicate to push this NDEF out to the mobile phone.

Example 7: AAR

```
>>50 00 16 C5 00 86 63 6F 6D 2E 61 75 74 6F 6E 61 76 69 2E 6D 69 6E 69 6D 61 70 04
```

wConfig = 0x0086 is use to set a AAR record to module Flash

Android PKG name:com.autonavi.minimap,

HEX:63 6F 6D 2E 61 75 74 6F 6E 61 76 69 2E 6D 69 6E 69 6D 61 70 (20 bytes)

Example 8: BT

```
>>50 00 18 C5 00 87 00 27 FD 8F 4C EF 40 0E 09 41 49 52 50 55 4C 53 45 20 41 32 30 30 ED
```

wConfig = 0x0087 is use to set a BT(Bluetooth) record to module Flash

//6bytes MAC,and then Eir buffer

//BL type:0-BrEdr,1-Le,2-SecureBrEdr,3-SecureLe

//MAC address:40:EF:4C:8F:FD:27

//EIR[0] length:0E

//EIR[0] type:09 = Complete local name

//Complete local name:"AIRPULSE A200":41 49 52 50 55 4C 53 45 20 41 32 30 30

Example 9: BT

wConfig = 0x0087 is use to set a BT(Bluetooth) record to module Flash

//BL type:0-BrEdr,1-Le,2-SecureBrEdr,3-SecureLe

//MAC address:

//EIR[0] length:0E

//EIR[0] type:09 = Complete local name

//Complete local name:"DOPASS 0000":41 49 52 50 55 4C 53 45 20 41 32 30 30

Example 10: WIFI

```
>>50 00 20 C5 00 88 20 08 0B 31 38 36 38 38 39 38 30 30 30 30 0F 54 50 2D 4C 49 4E 4B 2D 30 37 35 35 54 5A 43 61
```

wConfig = 0x0088 is use to set a WIFI record to module Flash

//KEY:18688980000 (11bytes:31 38 36 38 38 39 38 36 30 34 35)

//SSID:TP-LINK-0755TZC:54 50 2D 4C 49 4E 4B 2D 30 37 35 35 54 5A 43 (15bytes)

Command:

Example 11: URL

```
>>50 00 18 C5 00 89 00 68 74 74 70 3A 2F 2F 77 77 77 2E 74 61 6F 62 61 6F 2E 63 6F 6D 26
```

wConfig = 0x0089 is use to set a HTTP to module Flash

https://www.taobao.com/ : 68 74 74 70 3A 2F 2F 77 77 77 2E 74 61 6F 62 61 6F 2E 63 6F 6D

Example 12: TEXT

wConfig = 0x008A is use to set a TEXT record to module Flash

```
>>50 00 0B C5 00 8A 00 31 31 32 32 33 33 34 34 14 //11223344 is stored as ASCII characters
```

```
>>50 00 1A C5 00 8A 00 68 74 74 70 73 3A 2F 2F 77 77 77 2E 62 72 64 72 66 69 64 2E 63 6F 6D 2F
```

// 68 74 74 70 73 3A 2F 2F 77 77 77 2E 62 72 64 72 66 69 64 2E 63 6F 6D = https://www.brdrfid.com

Command Return

```
<<50 00 00 C5 95
```

4.4.8 TOP_ERASE_NDEF(CMD = 0xC7)

Erase a valid NDEF message with an empty NDEF.

-----DLL Explanation-----

EraseNdef shall be used to erase a valid NDEF message by writing an empty NDEF (i.e. NDEF length as 0) to tag. This will change the tag from read/write state to initialized state. If tag is already in initialized state this API will return error.

Return Status Codes

#PH_ERR_SUCCESS	Operation successful.
#TOP_ERR_INVALID_STATE	Tag is not in any valid state (i.e. when check NDEF failed)
#TOP_ERR_READONLY_TAG	Tag is in read only state.
#TOP_ERR_EMPTY_NDEF	Tag is in initialized state (i.e. no NDEF / empty NDEF)

Other Depending on implementation and underlying component.

-----DLL-----

```
int Top_EraseNdef(void);
```

-----Protocol Example-----

```
>> 50 00 00 C7 97
```

```
<< 50 00 00 C7 97
```

4.4.9 TOP_FORMAT_NDEF (CMD = 0xC8)

Format a new non-NDEF tag as NDEF tag.

-----DLL Explanation-----

FormatNdef shall be used for formatting a non-NDEF tag as NDEF tag if needed when CheckNdef returns error. Before formatting, tag parameters like max. NDEF size etc., shall be specified by the application. If not specified tag will be formatted with default values. FormatNdef shall be called only once for a tag. Once formatted, tag will become a NDEF tag. To remove NDEF format if needed, application needs to overwrite / delete NDEF format using tag specific read/write commands.

This is only a utility function not specified by NFC Forum.

Please note that for T4T, this will attempt to use MIFARE DESFire proprietary commands to format the tag. If T4T is not MIFARE DESFire tag then format will not succeed.

Return Status Codes

#PH_ERR_SUCCESS	Operation successful.
#TOP_ERR_UNSUPPORTED_TAG	Tag does not support NDEF formatting.
#TOP_ERR_FORMATTED_TAG	Tag is already NDEF formatted.

Other Depending on implementation and underlying component.

-----DLL-----

```
int Top_FormatNdef(void)
```

-----Protocol Example-----

```
>> 50 00 00 C8 98
```

```
<< 50 00 00 C8 98 //OK
```

```
<< F0 00 01 C8 83 BA //ERROR:TOP_ERR_FORMATTED_TAG | 0x80
```