J4210N API Documentation

Table of Contents

Introduction

There is JAVA and C API. JAVA project contains example code to read card.

Go to bin directory, you will find J4210N.dll. This is the main DLL for accessing the reader. There are other DLL, which will also be needed. This is a 64-bit DLL. 32-bit is no longer supported.

This DLL can also be loaded in C# and Python using standard procedure.

Auto Detection Cards

ULTRALIGHT, ULTRALIGHT_EV1, ULTRALIGHT_C, NTAG203 ,NTAG213, NTAG215, NTAG216, MIFARE CLASSIC 1K, MIFARE CLASSIC 4K, MIFARE DESFIRE 2K, MIFARE DESFIRE 4K, MIFARE DESFIRE 8K, SONY FELICA LITE-S. Auto detection enables internal knowledge of the card, therefore, programming a card become easier. This reader is designed to read only one card at a time.

Demo

There is a demo folder which contains an application program. The program is written in Java and the source code is provided with detailed comment. The demo will run on Windows PC. In order to run on Linux and Mac OS X, open the program with Eclipse on respective platform and run from Eclipse.

DLL Functions

Following DLL functions are available.

unsigned char AvailablePorts(char ports[256][8]);

Auto detects avialable serial ports and copies them into the two dimensional array.

Parameters:
ports: com port name array. Must be of size 256 x 8 or a single dimension array of 2048. Example: ports[0] = “COM4”, ports[1] = “COM7”, etc

Returns: 1 on success.

unsigned char OpenPort(unsigned char* port);

Opens a COM port.

Parameters:
port: com port name. Example “COM4”, etch

Returns: 1 on success.

void ClosePort();

Closes COM port that was opened. Otherwise does nothing.

unsigned char Scan(unsigned char* uid, unsigned char *size, unsigned char retries)

Scans for Card. If found, the UID of the card is saved into the supplied uid parameter and its size is stored in size parameter. The size of uid array should be sufficiently large, for example, 16 bytes.

Parameters:
uid: saves UID of the card.
size: saves array of UID.

Returns: 1 on success.

unsigned char Keys(unsigned char* keyA, unsigned char* keyB)

Set access keys for the card.

Parameters:
keyA: access key A. For MIFARE 1k, this is usually 6 bytes.
keyB: access key B. For MIFARE 1k, this is usually 6 bytes.

Returns: 1 on success.

unsigned char Read(int block, unsigned char *data, int *size, unsigned char keyB)

Reads a block. Allocate sufficiently large array to hold the read data. The data read must equal the block or page size of the card.

Parameters:
data: stores the read data into the array.
size: stores the number of bytes read. This is equal to the block size.

Returns: 1 on success.

unsigned char Write(int block, unsigned char* data, int size, unsigned char keyB)

Writes a block of data into Card. Data size must equal or bigger than the block or page size. If array passed is smaller, random values would be written. Zero fill the array if data is small. All data may not be written if size > block/page size.

Parameters:
data: array of data to be written. Array size must equal block or page size.
size: size in bytes of data to be written.

Returns: 1 on success.

unsigned char CardType()

Reads the card type.

Parameters: none

Returns: card type as integer.

char *CardName()

Returns card name, e.g, MIFARE, NTAG, etc.

Parameters: none

Returns: card name as string. This string corresponds to the card type.

unsigned char NdefFormat()

NDEF format the card. A card must be detected before calling this method.

Parameters: none. Operates on the currently detected card.

Returns: 1 if NDEF format was successful. If no card found, will return 0.

unsigned char NdefAddUri(char* uri)

Adds NDEF URI record.

Parameters: URI to be added.

Returns: 1 if command was successful. If no card found, will return 0.

unsigned char NdefAddText(char* text)

Adds NDEF Text record.

Parameters: Text to be added.

Returns: 1 if command was successful. If no card found, will return 0.

unsigned char NdefErase()

Erases record on the card.

Parameters: none.

Returns: 1 if command was successful. If no card found, will return 0.

int NdefRead()

Reads record on the card.

Parameters: none.

Returns: Number of records read, if command was successful. If no card found or no record found, will return -1.

unsigned char NdefGetRecord(int i, char* type, char* id, char* encoding, char* payload, int* size)

Reads the i-th record. i must be 0 to (number of records – 1). Record type is stored in type.

Parameters:
i – zero based record index. Must be valid. The should be to 0 to the number returned by NdefRead() - 1.
type – record type. Pass at lease a character array of 64 byte.
id – record id. Pass at lease a character array of 64 byte.
encoding – the encoding used in the payload. Pass at lease a character array of 64 byte.
payload – record payload. Pass sufficiently large character array. 1024 byte recommended.
size – payload size. The actual size of the payload is stored here.

Returns: 1 if command was successful. If no card found, will return 0.

int BlockCount()

Returns number of blocks in the card memory. This method must be called after scan operation.

Parameters: none.

Returns: Number of blocks in card memory as integer.

int BlockSize()

Returns block size of the card memory. This method must be called after scan operation. Blocks size indicates number of bytes that has to be written into or read from the card in a single write or read operation. Individual bytes could not be read or written, but only blocks can be read or written.

Parameters: none.

Returns: Block size of the card memory as integer.

unsigned char Format()

Formats and attempts to reset the tag to factory setting. This operation will reset all data to ZEROs. If the tag is password protected, reset to default password before calling this method.

Parameters: none.

Returns: Resets all data in the card to ZEROs.

unsigned char IsNdef()

Call this method to find out if the tag is NDEF formatted. A brand new card is not NDEF formatted. Calling this method after a scan operation would return 0 on a brand new card. This method should return 1 after the card is NDEF formatted using NdefFormat() method.

Parameters: none.

Returns: 1 if the tag is NDEF formatted.

unsigned char NdefReadBlock(int block, unsigned char *data, int *size)

This is a helper function to read raw NDEF blocks directly from memory for debugging purposes. Those developers who are planning to use Java NDEF library instead of the library provided by the driver may use this method to read raw blocks and display them.

Parameters:
block: zero based block number to read.
data: an array of unsigned char of at least size returned by BlockSize() method.
size: Actually number of bytes copied. Usually equal to BlockSize() value.

Returns: 1, if successful.

unsigned char UserMemory(int *start, int *end)

For the card under scan, retrieves the user memory range.

Parameters:
start: pointer to integer that will store the beginning of the block.
end: pointer to integer that will store the ending block.

Returns: 1, if successful.

unsigned char Sync()

If the card under scan gets out of sync, call this method to sync the card. A card may go out of sync if any read or write operation fails. After any of the failure, this method must be called to make the card in sync with the software.

Parameters: none.

Returns: 1, if successful.

unsigned char EmulateInit(unsigned char *uid, int buffsize, unsigned char writeable)

Setup device for card emulation but does not enters into emulation mode. This method must be called before you can begin card emulation. This methods allocates space for UID and sets up internal buffer size for card emulation.

Parameters:
uid: a three byte array with non zero UID. For example: {0x3A, 0x46, 0xf2}.
buffsize: card memory size in bytes. Maximum allowed is 128 bytes.
writeable: If cards need to be writeable, pass one. Otherwise cards will be emulated as read-only.

Returns: 1, if successful.

unsigned char EmulateStart(int timeout)

This is a blocking operation, so choose timeout in milliseconds during which time the device will emulate as card. Do not use a large timeout. This may hang the software.

Parameters:
timeout: time out in milli seconds.

Returns: 1, if a card received a write. For read only operation, the return value will always be zero.

unsigned char EmulateStop()

Stops emulation mode. Once stopped, to restart emulation, first call EmulateInit method to initialize then call EmulateStart. This method de-allocates memory that was initialized by EmulateInit.

Parameters: none

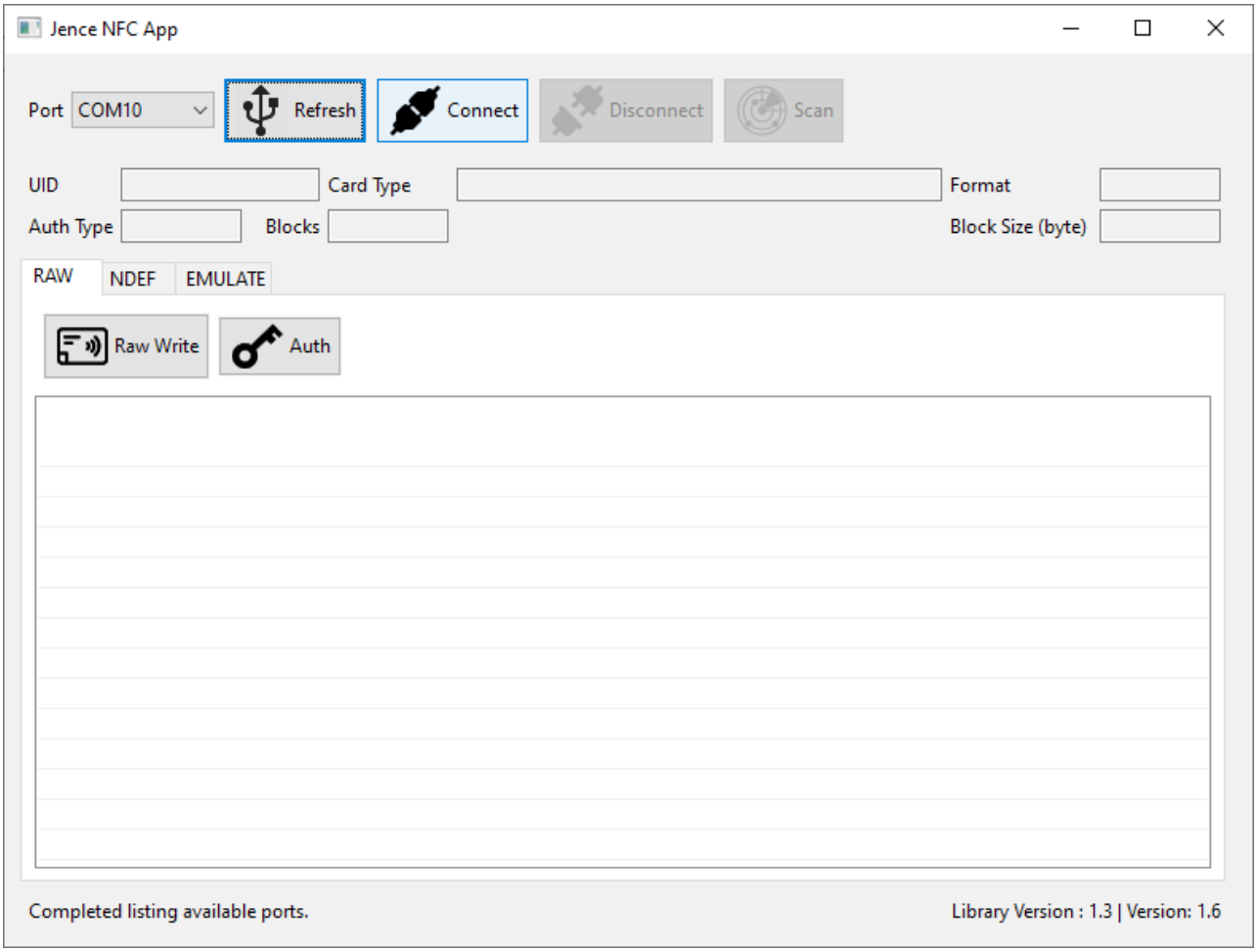
Returns: 1, if successful.

Jence Nfc App

How to download

Please download the software from our official website: www.jence.com. In the search option, type NFC, then you will see **NFC Desktop Reader Writer Model 4210N USB Powered 13.56MHz NFC Reader Writer** in the suggestion. Select it to go to the product page. Then after expanding the show more and scrolling down, you will see the SDK download option. Click here to download the zip folder of the software tool.

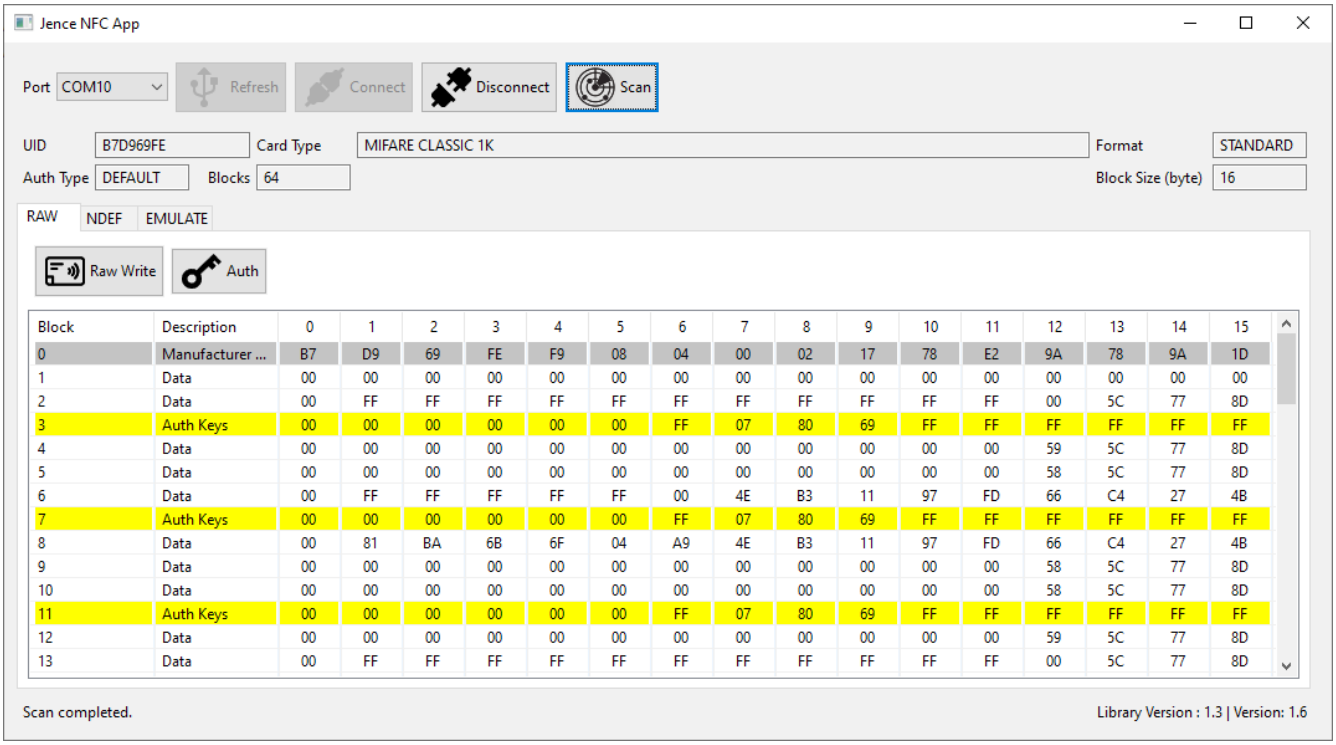
How to use the software application



Connect the Software: After the download, please unzip the folder. Inside the folder, please go to the demo folder and then click on the application file which is **j4210n.exe**. Two command prompt window will appear. Now connect the NFC hardware with the computer through USB cable and find the com port inside the device manager. Inside the Jence NFC app window, click on the **Refresh** button and select the com port of the NFC hardware. Finally, click on the **Connect** button.

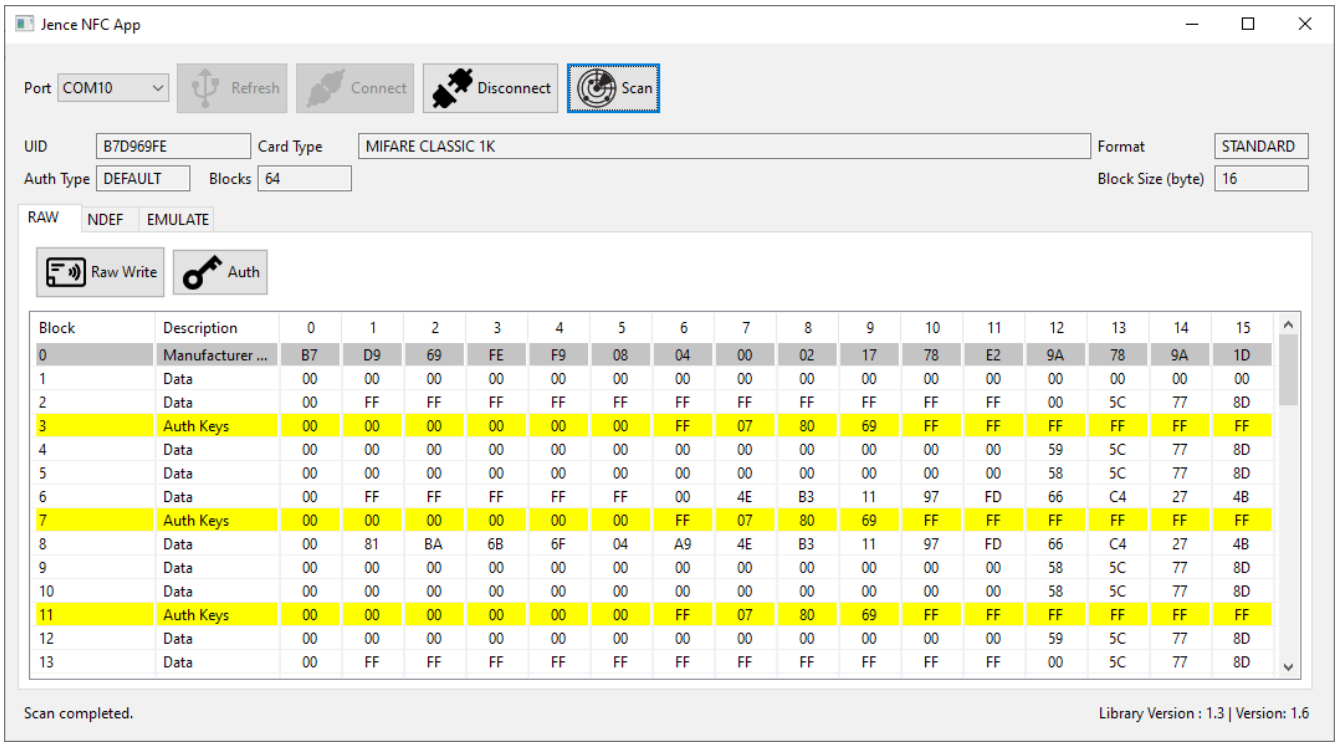
Scan The Card

This software tool can scan different types of cards, such as NXP Mifare Classic 1k, NXP Ultralight, NFC NTAG. All these cards are brand new, unprogrammed and clean. If we place any of these card on the NFC hardware and click on the **Scan** button inside the software, it will scan the card and we will see the inside contents of that card, such as its type, size, no of block, whether it is NDEF or non NDEF (Standard) format.



Block	Description	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	Manufacturer ...	B7	D9	69	FE	F9	08	04	00	02	17	78	E2	9A	78	9A	1D
1	Data	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
2	Data	00	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	00	5C	77	8D
3	Auth Keys	00	00	00	00	00	00	FF	07	80	69	FF	FF	FF	FF	FF	FF
4	Data	00	00	00	00	00	00	00	00	00	00	00	00	59	5C	77	8D
5	Data	00	00	00	00	00	00	00	00	00	00	00	00	58	5C	77	8D
6	Data	00	FF	FF	FF	FF	FF	00	4E	B3	11	97	FD	66	C4	27	4B
7	Auth Keys	00	00	00	00	00	00	FF	07	80	69	FF	FF	FF	FF	FF	FF
8	Data	00	81	BA	6B	6F	04	A9	4E	B3	11	97	FD	66	C4	27	4B
9	Data	00	00	00	00	00	00	00	00	00	00	00	00	58	5C	77	8D
10	Data	00	00	00	00	00	00	00	00	00	00	00	00	58	5C	77	8D
11	Auth Keys	00	00	00	00	00	00	FF	07	80	69	FF	FF	FF	FF	FF	FF
12	Data	00	00	00	00	00	00	00	00	00	00	00	00	59	5C	77	8D
13	Data	00	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	00	5C	77	8D

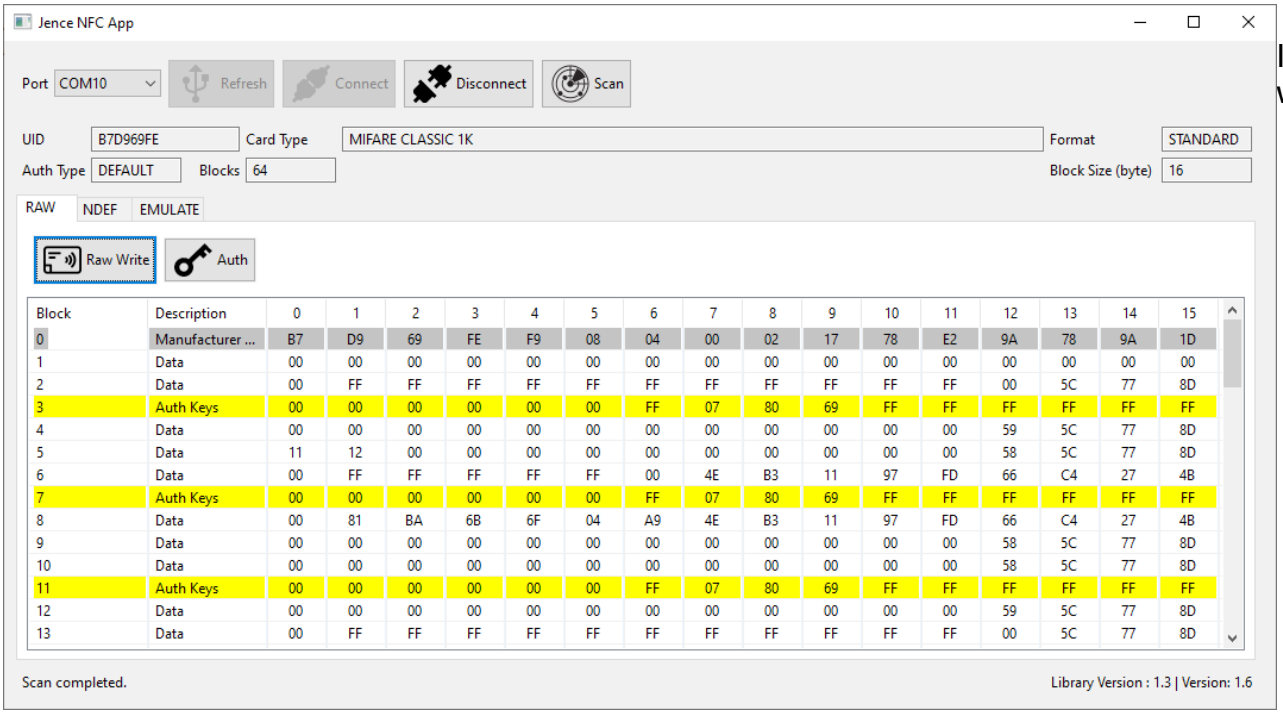
Raw Data Input



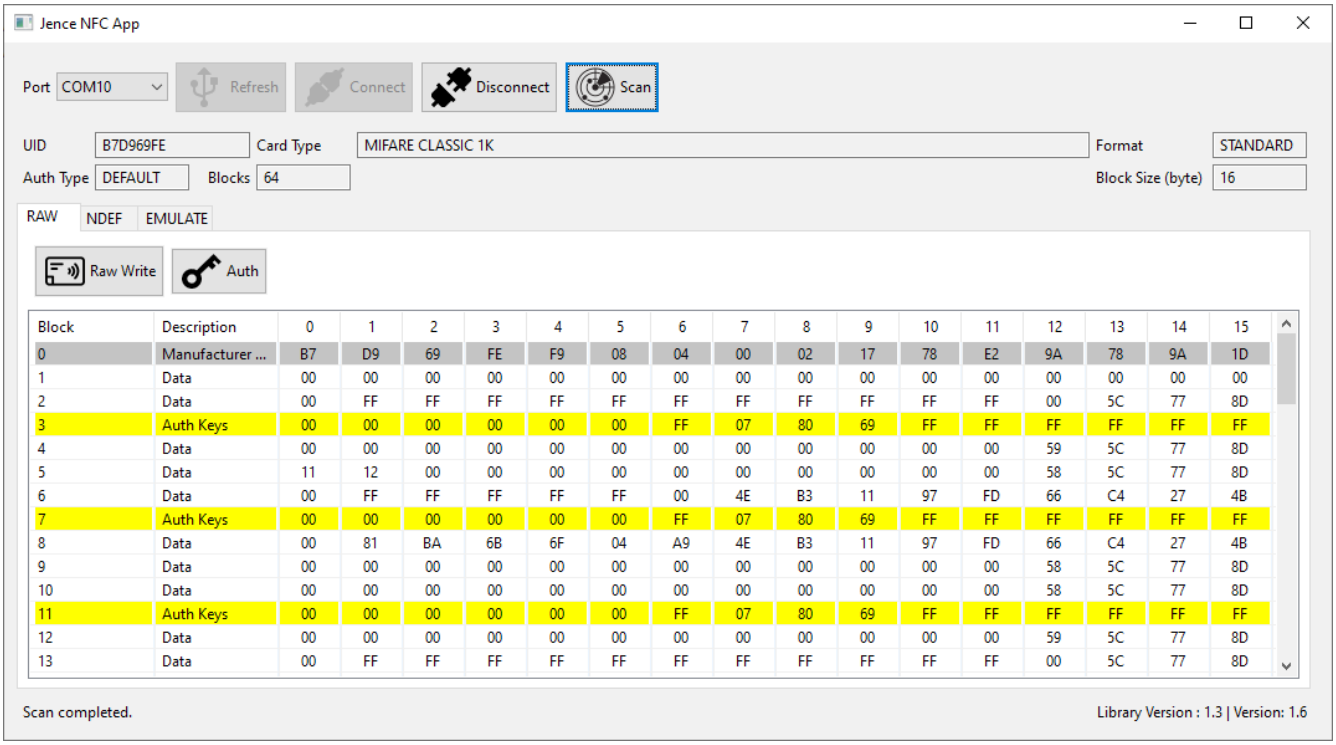
Block	Description	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	Manufacturer ...	B7	D9	69	FE	F9	08	04	00	02	17	78	E2	9A	78	9A	1D
1	Data	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
2	Data	00	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	00	5C	77	8D
3	Auth Keys	00	00	00	00	00	00	FF	07	80	69	FF	FF	FF	FF	FF	FF
4	Data	00	00	00	00	00	00	00	00	00	00	00	00	59	5C	77	8D
5	Data	00	00	00	00	00	00	00	00	00	00	00	00	58	5C	77	8D
6	Data	00	FF	FF	FF	FF	FF	00	4E	B3	11	97	FD	66	C4	27	4B
7	Auth Keys	00	00	00	00	00	00	FF	07	80	69	FF	FF	FF	FF	FF	FF
8	Data	00	81	BA	6B	6F	04	A9	4E	B3	11	97	FD	66	C4	27	4B
9	Data	00	00	00	00	00	00	00	00	00	00	00	00	58	5C	77	8D
10	Data	00	00	00	00	00	00	00	00	00	00	00	00	58	5C	77	8D
11	Auth Keys	00	00	00	00	00	00	FF	07	80	69	FF	FF	FF	FF	FF	FF
12	Data	00	00	00	00	00	00	00	00	00	00	00	00	59	5C	77	8D
13	Data	00	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	00	5C	77	8D

If we want to give any raw data input, we can give it by clicking on **Raw Write** Button. Then that raw data will be saved in the card. For demonstration, here we have altered data in the 5th

raw in column 3 from 00 to 11 and in column 4 from 00 to 12 and clicked on the **Raw Write** button. Now these data have been saved in the card.

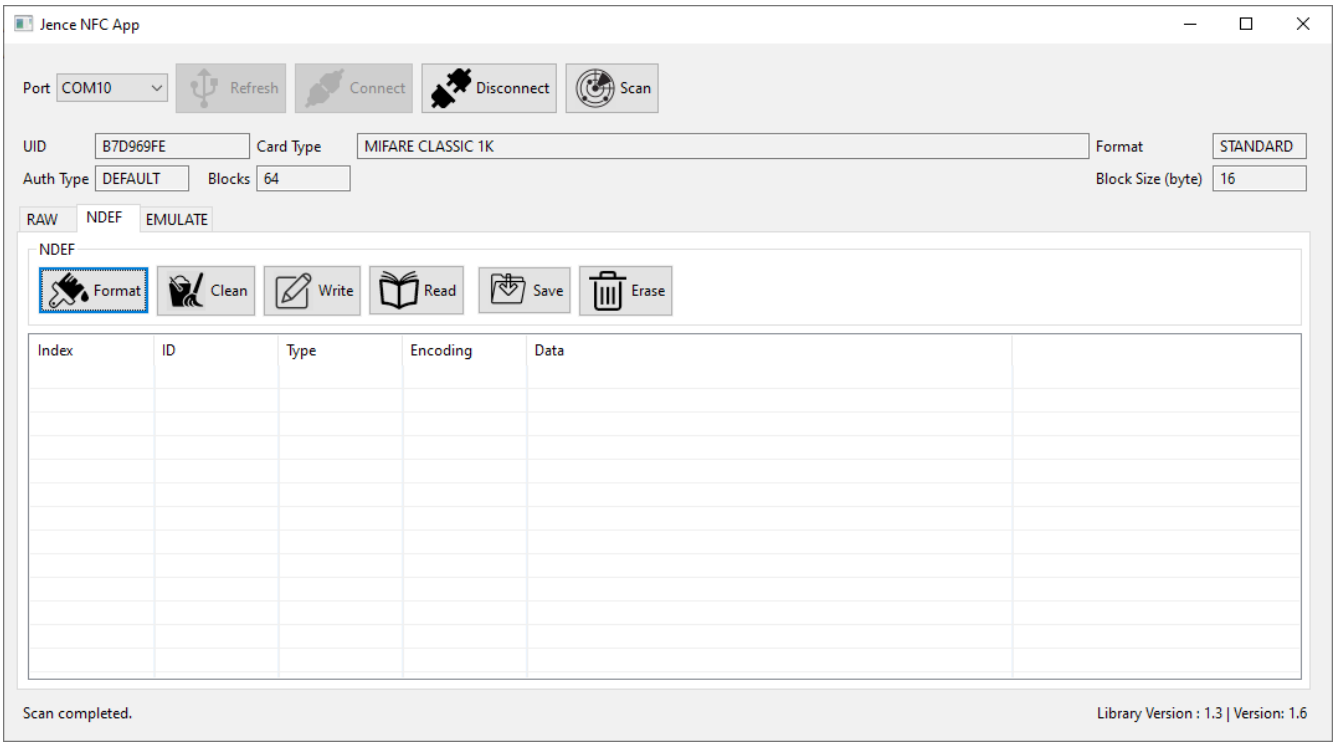


If we scan the card now, we will see the card with the altered data(11 and 12) in it.

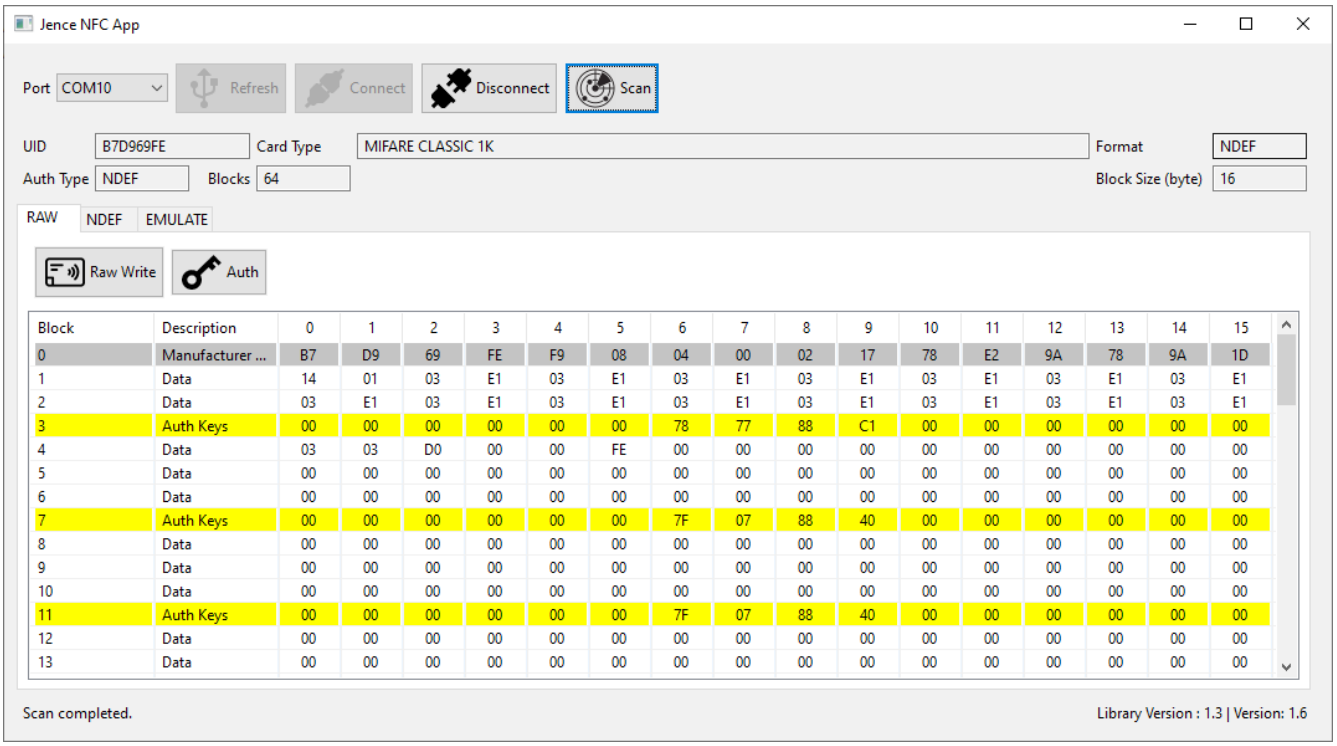


Format the Card

If we want to change the card format from Standard to NDEF, first we have to go to the NDEF tab and click on the **Format** button to format the card.

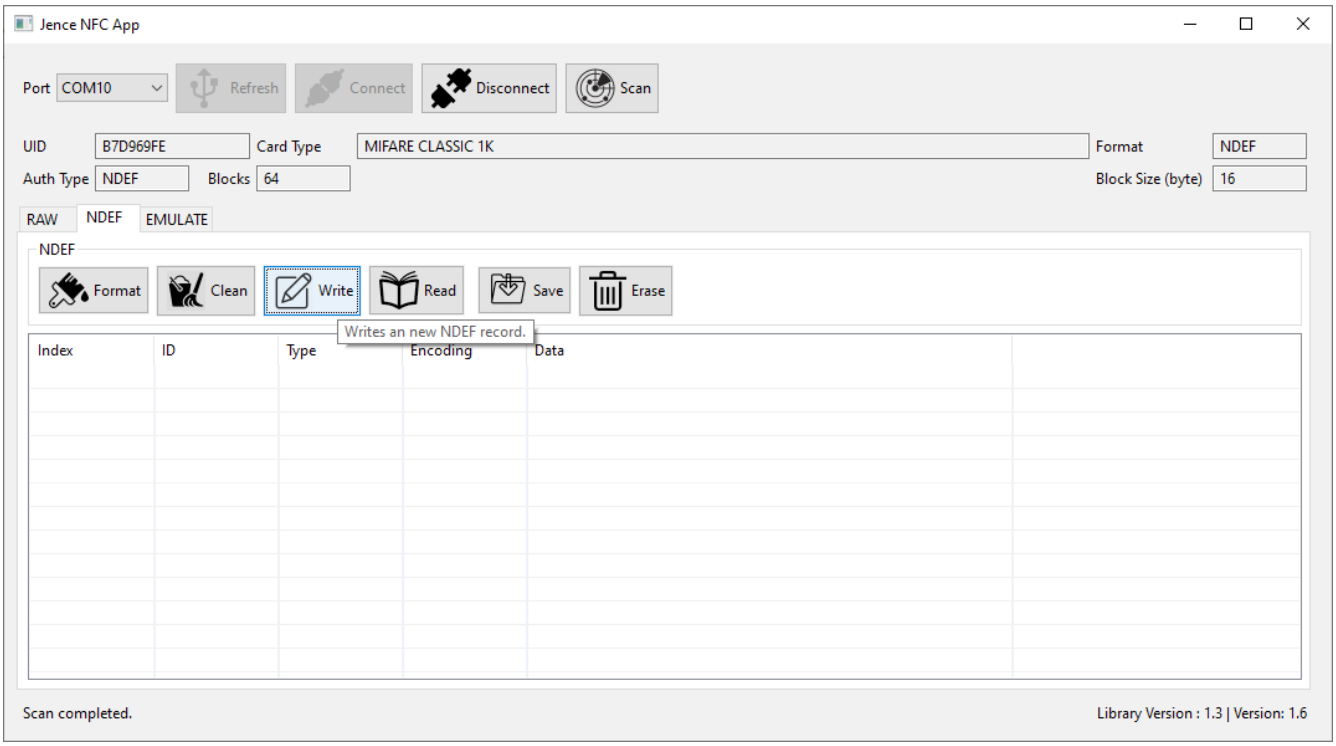


If we scan the card after formatting, we will see the card is in NDEF format.

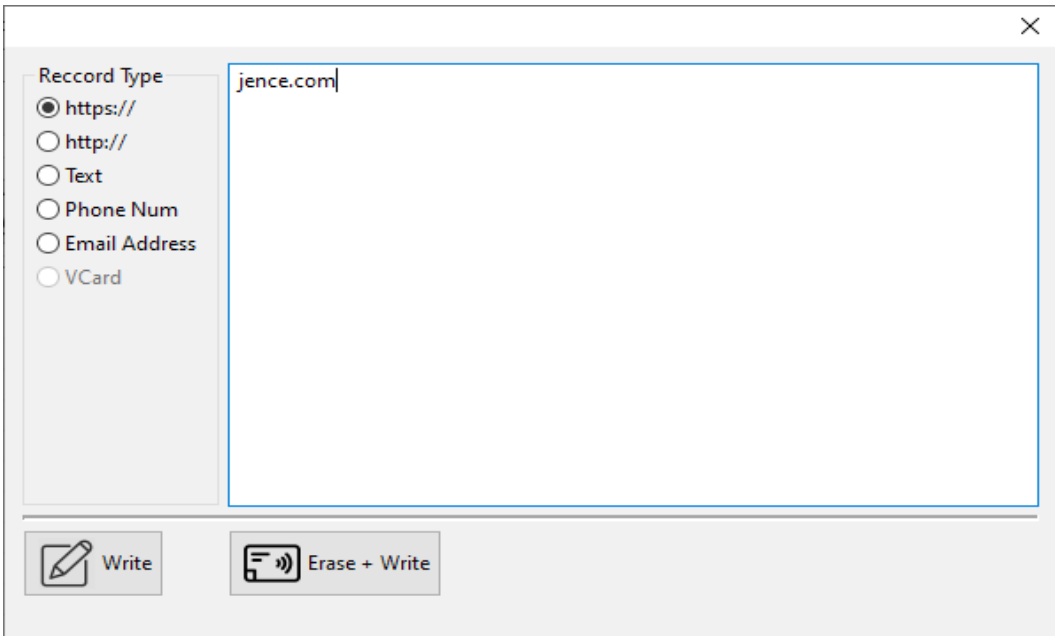


Write and Read Data

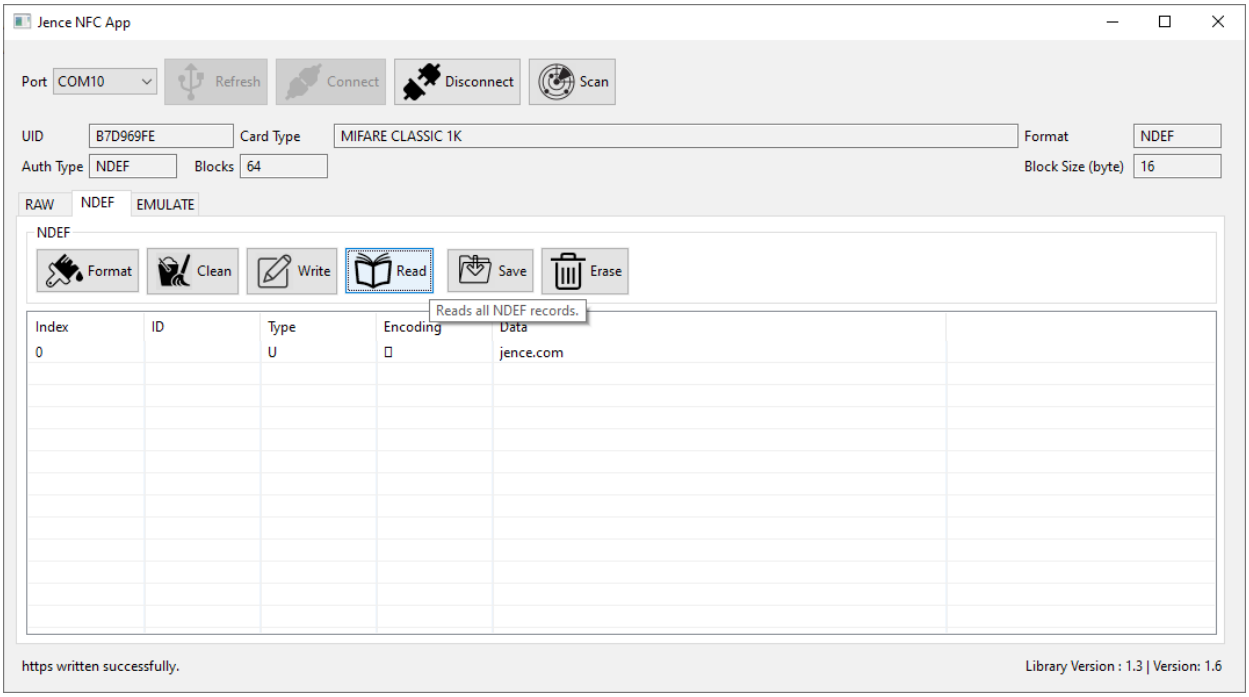
Now if we want to write any text or URL, we can do it by clicking on the **Write** button inside the NDEF tab.



Here, we have selected the URL (https://) option and written a web address. Similarly, we can choose the Text, Email Address or other options and write any data according to the chosen option. There are two buttons in the bottom. **Write** button will show the new record but will not remove the old record. Conversely, **Erase+Write** button will erase the past data record while showing the new data.



We can read the written data by clicking on the **Read** button.



Read data on the NFC Enabled Mobile Phone

After writing any data in the aforementioned way, if we place the card on the back of the NFC enabled android phone or iPhone, we can see that data on the phone.

Troubleshooting

1. Could not connect to PC.

A. Make sure that the COM port number (for Windows) or TTY (for Linux and Mac OSX) appear when the device is connected. If the serial port is not recognized, then uninstall and then reinstall the driver. In some cases, you may need to reboot the PC after driver installation.

2. Could not detect type of card.

A. If card type is not detected, then the card may be password protected. In this case, the user may pro programmatically set the card type.

3. Could not NDEF format a card.

A. Card may have unknown password. To format the card, all password on the card should be reset to the default factory password.

Version History

Version 1.4

- NDEF formatting on Ultralight bug fix.

Version 1.3

- NDEF to clean formatting bug fix.

Version 1.2

- NDEF support added: NDEF formatting, NDEF URL, Text, Phone number addition. NDEF read and write. NDEF format and erase.
- Auto detection support added to NTAG203, ULTRALIGHT.
- Card Emulation.
- Additional Functions: Format, BlockSize, BlockCount, UserMemory, Sync, EmulateInit, EmulateStart and EmulateStop implemented.

Version 1.1

More card type added. Auto detection support added to ULTRALIGHT_C, MIFARE CLASSIC 1K, MIFARE CLASSIC 4K.

Version 1.0

Initial version. Auto detection support added to ULTRALIGHT_EV1, NTAG213, NTAG215, NTAG216.

For questions, contact Jence.

JENCE
http://www.jence.com
Email: jence@jence.com