

---

# **USB Mifare/Memory Card API**

**Edition 1.7**

## Contents

Common API.....	4
<b>int HY_UsbGetFirmware(char *sVer).....</b>	<b>4</b>
Mifare Card API.....	5
<b>int HY_UsbPowerUpMifareReader().....</b>	<b>5</b>
<b>int HY_UsbPowerDownMifareReader().....</b>	<b>6</b>
<b>int HY_UsbMifareSelect(byte *serial, byte *sak).....</b>	<b>7</b>
<b>int HY_UsbMifareLogin(byte sector, byte type, byte *key).....</b>	<b>8</b>
<b>int HY_UsbMifareReadBlock(int block, byte *data).....</b>	<b>9</b>
<b>int HY_UsbMifareWriteBlock(int block, byte *data).....</b>	<b>11</b>
<b>int HY_UsbMifareReadValue(int block, byte *data).....</b>	<b>12</b>
<b>int HY_UsbMifareWriteValue(byte block, byte *data).....</b>	<b>14</b>
<b>int HY_UsbMifareIncrement(int block, byte *data).....</b>	<b>15</b>
<b>int HY_UsbMifareDecrement(int block, byte *data).....</b>	<b>16</b>
<b>int HY_UsbMifareWriteKeyA(int sector, byte *key).....</b>	<b>17</b>
<b>int HY_UsbMifareHalt().....</b>	<b>18</b>
<b>int HY_UsbMifareRequest(byte *atq).....</b>	<b>19</b>
<b>int HY_UsbMifareGetATS(byte *ats, byte *ats_len).....</b>	<b>20</b>
<b>int HY_UsbMifareExchangeAPDU(byte *apdu, byte apdu_len, byte *response, byte     *response_len).....</b>	<b>21</b>
<b>int HY_UsbMifareExchangeAPDU2(const char *apdu, char *response).....</b>	<b>23</b>
<b>int HY_UsbMifareDeselect().....</b>	<b>25</b>
<b>int HY_UsbMifareBeep(int ms).....</b>	<b>26</b>
<b>int HY_UsbMifareRepeatBeep(byte repeat, int on_duration, int off_duration).....</b>	<b>27</b>
Memory Card API.....	28
<b>int HY_UsbMCardIsInserted ().....</b>	<b>28</b>

<b>int HY_UsbMCardSetType(byte byType).....</b>	<b>29</b>
<b>int HY_UsbMCardPowerUp(byte *byATR).....</b>	<b>30</b>
<b>int HY_UsbMCardPowerDown ().....</b>	<b>31</b>
<b>int HY_UsbMCardVerifyPSC(byte *pbPSC, int nLength).....</b>	<b>32</b>
<b>int HY_UsbMCardReadCounter(byte *pbCounter).....</b>	<b>33</b>
<b>int HY_UsbMCardUpdateCounter(byte byCounter).....</b>	<b>34</b>
<b>int HY_UsbMCardReadPSC(byte *pbPSC, byte *pbLength).....</b>	<b>35</b>
<b>int HY_UsbMCardUpdatePSC (byte *pbPSC, int nLength).....</b>	<b>36</b>
<b>int HY_UsbMCardReadMemory(int nOffset, byte *pbReadBuffer, int nReadLen).....</b>	<b>37</b>
<b>int HY_UsbMCardWriteMemory(int nOffset, byte *pbWritedBuffer, int nWriteLen).....</b>	<b>38</b>
<b>int HY_UsbMCardSetWriteProtection(int nOffset, int nWriteLen).....</b>	<b>40</b>
<b>int HY_UsbMCardGetWriteProtection(byte *pbProtected, byte *pbLength).....</b>	<b>41</b>
<b>int HY_UsbSetLed(bool isOn, int nDuration).....</b>	<b>43</b>
<b>Summary of Return Codes.....</b>	<b>44</b>

# Common API

## **int HY\_UsbGetFirmware(char \*sVer)**

### **Function**

Get the firmware version of the device

### **Parameter**

*sVer* [output] the firmware version will be copied to this character pointer

Note: 25 bytes should be enough to get the version from the pinpad.

### **Return**

If the function succeeds, the return value is zero (HY\_SUCCESS).

If the function fails, the return value is non-zero.

Example:

```
char sVersion[25];
int ret = HY_UsbGetFirmware(sVersion);
CString s;
if (ret == 0)
{
    s.Format("Firmware Version: %s", sVersion);
    AfxMessageBox(s);
}
else
{
    s.Format("Status: %02X", ret);
    AfxMessageBox(s);
}
```

# Mifare Card API

## **int HY\_UsbPowerUpMifareReader()**

### **Function**

Power up the reader to enable Mifare related functions. By default, the reader is powered up automatically.

### **Parameter**

*None*

### **Return**

If the function succeeds, the return value is zero.

Example:

```
int ret = HY_UsbPowerUpMifareReader();
CString s;

if (ret == 0)
{
    s.Format("Power up successfully");
    AfxMessageBox(s);
}
else
{
    s.Format("Status: %02X", ret);
    AfxMessageBox(s);
}
```

## int HY\_UsbPowerDownMifareReader()

### Function

Power down the reader to reduce power consumption. When the reader is powered down, if a user performs any Mifare related functions, the reader will be powered up automatically.

### Parameter

*None*

### Return

If the function succeeds, the return value is zero.

Example:

```
int ret = HY_UsbPowerDownMifareReader();
CString s;

if (ret == 0)
{
    s.Format("Power down successfully");
    AfxMessageBox(s);
}
else
{
    s.Format("Status: %02X", ret);
    AfxMessageBox(s);
}
```

## **int HY\_UsbMifareSelect(byte \*serial, byte \*sak)**

### **Function**

Select Mifare card (Supported Card: Mifare 1 Tpye A and Mifare Pro)

### **Parameter**

*serial* [output] Serial number of the card detected if the operation is success, 4 bytes.

*sak* [output] SAK of the card detected if the operation is success, 1 byte.

### **Return**

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero.

0x1E => Collision

0x1F => No SAK

0x21 => No Tag

Example:

```
byte serial[25], sak;
int ret = HY_UsbMifareSelect(serial, &sak);
CString s;

if (ret == 0)
{
    s.Format("Serial number: 0x%02X 0x%02X 0x%02X 0x%02X", serial[0],
serial[1], serial[2], serial[3]);
    AfxMessageBox(s);
}
else
{
    s.Format("Status: %02X", ret);
    AfxMessageBox(s);
}
```

## **int HY\_UsbMifareLogin(byte sector, byte type, byte \*key)**

### **Function**

Login to a sector (Supported Card: Mifare 1 Tpye A)

### **Parameter**

*sector*    Sector need to login (0x00 to 0x0F)  
*type*       Key type  
            0xAA:    authenticate with key type A  
            0xBB:    authenticate with key type B  
*key*        Authentication key, 6 bytes.

### **Return**

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero.

0x20 => Login Fail

### **Note:**

You must select the card first before login.

### **Example**

```
byte key[] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
           // Please ask the key from the card manufacturer
byte type = 0xAA;
byte sector = 0;
CString s;

Int ret = HY_UsbMifareLogin(sector, type, key);
if (ret == 0)
    AfxMessageBox("Login success");
else
{
    s.Format("Status: %02X", ret);
    AfxMessageBox(s);
}
```



## **int HY\_UsbMifareReadBlock(int block, byte \*data)**

### **Function**

Read a data block (Supported Card: Mifare 1 Type A)

### **Parameter**

*block* The block index to be read, 1 byte.

*data* [output] Data block returned if operation is success, MUST be at least 16 bytes.

### **Return**

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero.

0x21 => No Tag

0x22 => Not Authenticated

0x23 => Read Fail

### **Note:**

The block index is determined by the Sector you have login.

There are 4 Blocks in each Sector.

The 1st Sector contains Block 0x00, 0x01, 0x02 and 0x03.

The 2nd Sector contains Block 0x04, 0x05, 0x06 and 0x07.

The 3rd Sector contains Block 0x08, 0x09, 0x0A and 0x0B.

.....

The last Block for each Sector contains the key information to the Sector.

It is better not to write anything to the last Block of each Sector unless you read the Mifare card specifications clearly.

### **Example**

```
byte data[25];
int block = 0;
CString s, s1;

int ret = HY_UsbMifareReadBlock(block, data);
if (ret == 0) {
    s = "";
    for (int i=0; i<16; i++)
```

```
        s.Format("%s%02X ", s, data[i]);

        s1.Format("Read Data: %s", s);
        AfxMessageBox(s1);
    }
    else {
        s1.Format("Status: %02X", ret);
        AfxMessageBox(s1);
    }
```

## **int HY\_UsbMifareWriteBlock(int block, byte \*data)**

### **Function**

Write a data block (Supported Card: Mifare 1 Tpye A)

### **Parameter**

*block*      The block index to be read, 1 byte.

*data*        The data to write, MUST be 16 bytes.

### **Return**

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero.

0x21 => No Tag

0x22 => Not Authenticated

0x24 => Write Fail

### **Note:**

The block index is determined by the Sector you have login.

There are 4 Blocks in each Sector.

The 1st Sector contains Block 0x00, 0x01, 0x02 and 0x03.

The 2nd Sector contains Block 0x04, 0x05, 0x06 and 0x07.

The 3rd Sector contains Block 0x08, 0x09, 0x0A and 0x0B.

.....

### **Example**

```
byte byData[16] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09,
0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F};
int block = 0;
CString s;

int ret = HY_UsbMifareWriteBlock(block, byData);
if (ret == 0)
    AfxMessageBox("Write success");
else {
    s.Format("Status: %02X", ret);
    AfxMessageBox(s);
}
```

## **int HY\_UsbMifareReadValue(int block, byte \*data)**

### **Function**

Read a value from a given block (Supported Card: Mifare 1 Tpye A)

### **Parameter**

*Block*      The block number to be read, 1 byte.

*data*        [out] Block value returned if operation is success, MUST be 4 bytes.

### **Return**

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero.

0x21 => No Tag

0x22 => Not Authenticated

0x23 => Read Fail

### **Note:**

The block index is determined by the Sector you have login.

There are 4 Blocks in each Sector.

The 1st Sector contains Block 0x00, 0x01, 0x02 and 0x03.

The 2nd Sector contains Block 0x04, 0x05, 0x06 and 0x07.

The 3rd Sector contains Block 0x08, 0x09, 0x0A and 0x0B.

.....

### **Example**

```
byte data[4];
int block = 5;
CString s = "", s1 = "";

int ret = HY_UsbMifareReadValue(block, data);
if (ret == 0)
{
    for (int i=0; i<4; i++)
        s.Format("%s%02X ", s, data[i]);

    s1.Format("Read Value: %s", s);
    AfxMessageBox(s1);
}
```

```
}  
else  
{  
    s1.Format("Status: %02X", ret);  
    AfxMessageBox(s1);  
}
```

## **int HY\_UsbMifareWriteValue(byte block, byte \*data)**

### **Function**

Write a value to a given block (Supported Card: Mifare 1 Tpye A)

### **Parameter**

*block*      The block number to be read, 1 byte.

*data*        The value to write, MUST be 4 bytes.

### **Return**

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero.

0x21 => No Tag

0x22 => Not Authenticated

0x24 => Write Fail

### **Note:**

The block index is determined by the Sector you have login.

There are 4 Blocks in each Sector.

The 1st Sector contains Block 0x00, 0x01, 0x02 and 0x03.

The 2nd Sector contains Block 0x04, 0x05, 0x06 and 0x07.

The 3rd Sector contains Block 0x08, 0x09, 0x0A and 0x0B.

.....

### **Example**

```
CString s1;
byte byData[4] = {0x12, 0x34, 0x56, 0x78};
int ret=0, block = 10;

ret = HY_UsbMifareWriteValue(block, byData);
if (ret == 0)
    AfxMessageBox("Write success");
else {
    s1.Format("Status: %02X", ret);
    AfxMessageBox(s1);
}
```

## **int HY\_UsbMifareIncrement(int block, byte \*data)**

### **Function**

Add a value to a given value block (Supported Card: Mifare 1 Tpye A)

### **Parameter**

block     The block number to be read,1 byte.  
data     The value to be added, MUST be 4 bytes, Little Endian  
          e.g. if add 1, data should be [0x01, 0x00, 0x00, 0x00]

### **Return**

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero.

0x21 => No Tag

0x22 => Not Authenticated

0x24 => Write Fail

### **Note:**

The block index is determined by the Sector you have login.

There are 4 Blocks in each Sector.

The 1st Sector contains Block 0x00, 0x01, 0x02 and 0x03.

The 2nd Sector contains Block 0x04, 0x05, 0x06 and 0x07.

The 3rd Sector contains Block 0x08, 0x09, 0x0A and 0x0B.

.....

### **Example**

```
byte byData[4] = {0x01, 0x00, 0x00, 0x00}; // add 1
int ret = 0, block = 10;
ret = HY_UsbMifareIncrement(block, byData);
if (ret == 0)
    AfxMessageBox("Increment success");
else {
    CString s1;
    s1.Format("Status: %02X", ret);
    AfxMessageBox(s1);
}
```

## **int HY\_UsbMifareDecrement(int block, byte \*data)**

### **Function**

Subtract a value from a given value block (Supported Card: Mifare 1 Tpye A)

### **Parameter**

*block*      The block number to be read,1 byte.

*data*      The value to be subtracted, MUST be 4 bytes, Little Endian  
            e.g. if subtract 1, the data becomes [0x01, 0x00, 0x00, 0x00]

### **Return**

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero.

0x21 => No Tag

0x22 => Not Authenticated

0x24 => Write Fail

### **Note:**

The block index is determined by the Sector you have login.

There are 4 Blocks in each Sector.

The 1st Sector contains Block 0x00, 0x01, 0x02 and 0x03.

The 2nd Sector contains Block 0x04, 0x05, 0x06 and 0x07.

The 3rd Sector contains Block 0x08, 0x09, 0x0A and 0x0B.

.....

### **Example**

```
byte byData[4] = {0x01, 0x00, 0x00, 0x00}; // add 1
int ret = 0, block = 10;
ret = HY_UsbMifareDecrement (block, byData);
if (ret == 0)
    AfxMessageBox("Decrement success");
Else {
    CString s1;
    s1.Format("Status: %02X", ret);
    AfxMessageBox(s1);
}
```



## **int HY\_UsbMifareWriteKeyA(int sector, byte \*key)**

### **Function**

Change the key of type A (Supported Card: Mifare 1 Tpye A)

### **Parameter**

*sector*    The key of the sector to be changed, 1 byte.

*key*        Authentication key, 6 bytes.

### **Return**

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero.

0x21 => No Tag

0x22 => Not Authenticated

0x24 => Write Fail

0x27 => Access Denied

### **Note**

You must login to the sector before change the key of the same sector.

If the return value is zero, you need not login to the sector with the new key unless you remove the card.

### **Example**

```
byte byKey[6] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
int  sector=3, ret = 0;
ret = HY_UsbMifareWriteKeyA(sector, byKey);
if (ret == 0)
    AfxMessageBox("Change success");
else
{
    CString s1;
    s1.Format("Status: %02X", ret);
    AfxMessageBox(s1);
}
```

## int HY\_UsbMifareHalt()

### Function

Halt the mifare card (Supported Card: Mifare 1 Tpye A, Mifare Pro)

### Parameter

None

### Return

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero.

0x28 => Unable to halt

### Note

After select the card, you can call this function.

### Example

```
int ret = HY_UsbMifareHalt();
if (ret == 0)
    AfxMessageBox("Halt success");
else
{
    CString s1;
    s1.Format("Status: %02X", ret);
    AfxMessageBox(s1);
}
```

## **int HY\_UsbMifareRequest(byte \*atq)**

### **Function**

Make a request to the mifare card (Supported Card: Mifare 1 Tpye A, Mifare Pro)

### **Parameter**

*atq* [out ] ATQ of the card detected if the operation is success, 2 bytes.

### **Return**

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero.

0x1E => Collision

0x21 => No Tag

### **Note**

As this version supports detecting only one card at the antenna, this function is actually useless.

You can call this function after select the card.

### **Example**

```
byte atq[2];
int ret = HY_UsbMifareRequest(atq);
CString s;

if (ret == 0)
    s.Format("ATQ: 0x%02X 0x%02X", atq[0], atq[1]);
else
    s.Format("Status: %02X", ret);

AfxMessageBox(s);
```

## **int HY\_UsbMifareGetATS(byte \*ats, byte \*ats\_len)**

### **Function**

Get ATS from Mifare Pro card.

### **Parameter**

*ats* [out] ATS if the operation is success, at most 252 bytes.

*ats\_len* [out] the length of the ats, 1 byte

Different cards return different ATS with different lengths.

### **Return**

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero.

0x21 => No Tag

### **Note**

You can call this function after select the card.

### **Example**

```
byte ats[256], ats_len;
int ret = HY_UsbMifareGetATS(ats, &ats_len);
CString s="ATS: ";
if (ret == 0) {
    for (int i=0; i<ats_len; i++)
    {
        s1.Format("%02X ", ats[i]);
        s += s1;
    }
    s1.Format("\nLength: %d bytes", ats_len);
    s += s1;
    AfxMessageBox(s);
}
else
    s.Format("Status: %02X", ret);

AfxMessageBox(s);
```

## **int HY\_UsbMifareExchangeAPDU(byte \*apdu, byte apdu\_len, byte \*response, byte \*response\_len)**

### **Function**

Exchange APDU with the Mifare Pro card.

### **Parameter**

*apdu* APDU command sent to the card, at most 253 bytes.  
(Please ask your mifare card vendor for the details.)  
*apdu\_len* the length of the APDU command. The max. value is 253.  
*response* [out] the response from the card, at most 252 bytes  
*response\_len* [out] the length of the response.

### **Return**

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero.

0x21 => No Tag

### **Note**

You can call this function after HY\_UsbMifareGetATS returns success.

### **Example**

```
byte apdu[]={ 0x 00, 0x84,0x00, 0x00,0x04}, apdu_len=5;
byte response[253], response_len;
int ret = HY_UsbMifareExchangeAPDU(apdu, apdu_len, response, &response_len);
CString s="Response: ", s1="";
if (ret == 0)
{
    for (int i=0; i<response_len; i++)
    {
        s1.Format("%02X ", response[i]);
        s += s1;
    }
    s1.Format("\nLength: %d bytes", response_len);
    s += s1;
}
```

```
else
    s.Format("Status: %02X", ret);

AfxMessageBox(s);
```

## **int HY\_UsbMifareExchangeAPDU2(const char \*apdu, char \*response)**

### **Function**

Exchange APDU with the Mifare Pro card.

Same as HY\_UsbMifareExhcangeAPDU function, except the input and out parameters are string type.

### **Parameter**

<i>apdu</i>	APDU command sent to the card, an array of char, null-terminated string. It can only contain {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F} characters and the null character at the end. The max. length is 506 and the length of the string must be the multiple of 2. (Please ask your mifare card vendor for the details.)
<i>response</i>	[out] the response from the card, an array of char, null-terminated string. The max. length is 504 and the length of the string is the multiple of 2.

### *Note:*

*You must allocate enough memory for "response" parameter.*

### **Return**

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero.

0x21 => No Tag

0xE6 => Invalid Parameter

### **Note**

You can call this function after HY\_UsbMifareGetATS returns success.

### **Example**

```
char apdu[] = "0084000004";  
char response[504];  
int response_len;  
int ret = HY_UsbMifareExchangeAPDU2(apdu, response);
```

```
CString s;  
if (ret == 0)  
    s.Format("Response: %s", response);  
else  
    s.Format("Status: %02X", ret);  
  
AfxMessageBox(s);
```



## int HY\_UsbMifareDeselect()

### Function

Deselect the Mifare Pro card

### Parameter

None

### Return

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero.

0x21 => No Tag

### Example

```
int ret = HY_UsbMifareDeselect();
if (ret == 0)
    AfxMessageBox("Deselect success");
else
{
    CString s1;
    s1.Format("Status: %02X", ret);
    AfxMessageBox(s1);
}
```

## **int HY\_UsbMifareBeep(int ms)**

### **Function**

Play a beep sound

### **Parameter**

*ms*            the duration for a beep sound, unit: ms

### **Return**

If the function succeeds, the return value is zero.

0x50 => the duration is out of range. max < 10000ms

### **Note**

It depends on what model of the device connected.

### **Example**

```
int ms = 500;
int ret = HY_UsbMifareBeep(ms); // beep 0.5 sec
if (ret == 0)
    AfxMessageBox("Beep success");
else
{
    CString s1;
    s1.Format("Status: %02X", ret);
    AfxMessageBox(s1);
}
```

## **int HY\_UsbMifareRepeatBeep(byte repeat, int on\_duration, int off\_duration)**

### **Function**

Make the beep sound repeatedly

### **Parameter**

<i>Repeat</i>	number of repeat beep sound
<i>on_duration</i>	the duration of the beep sound on. The unit is 0.1 second.
<i>off_duration</i>	the duration of the beep sound off. The unit is 0.1 second.

### **Return**

If the function succeeds, the return value is zero (HY\_SUCCESS) after finishing the beep sound.

If  $(on\_duration + off\_duration) * repeat > 8000$  (ms), the return value is 0xE6.

If the function fails, the return value is non-zero.

### **Note**

It depends on what model of the device connected.

### **Example**

```
byte repeat = 2;
int on_duration = 5, off_duration = 5;

int ret = HY_UsbMifareRepeatBeep(repeat, on_duration, off_duration);
if (ret == 0)
    AfxMessageBox("Beep success");
else
{
    CString s1;
    s1.Format("Status: %02X", ret);
    AfxMessageBox(s1);
}
```

# Memory Card API

## **int HY\_UsbMCardIsInserted ()**

### **Function**

Check if the card is inserted. (But it cannot check if the card is SLE4442 or not)

### **Parameter**

None

### **Return**

If the function succeeds, the return value is 0x01 for card inserted or 0x00 for card not inserted.

If the function fails, the return value is nonzero.

### **Note**

After select the card, you can call this function.

### **Example**

```
int ret = HY_UsbMCardIsInserted();
if (ret == 1)
    AfxMessageBox("A memory card is inserted");
else if (ret == 0)
    AfxMessageBox("A memory card is not inserted");
else
{
    CString s1;
    s1.Format("Status: %02X", ret);
    AfxMessageBox(s1);
}
```

## **int HY\_UsbMCardSetType(byte byType)**

### **Function**

Set the memory card type. (But currently only SLE4442 card is supported.)

### **Parameter**

*byType*            type of memory card  
                    0x00:    SLE4441

### **Return**

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero.

                    0x31    Incorrect card type

### **Note**

It should be called before calling HY\_UsbMCardPowerUp function.

### **Example**

```
int nType = 0;
int ret = HY_UsbMCardSetType(nType); // Set SLE4441 card
if (ret == 0)
    AfxMessageBox("Set the card type successfully");
else
{
    CString s1;
    s1.Format("Status: %02X", ret);
    AfxMessageBox(s1);
}
```

## **int HY\_UsbMCardPowerUp(byte \*byATR)**

### **Function**

Power up the memory card and get the ATR (Answer to Reset) if power up successfully

### **Parameter**

*byATR* [out ] The buffer supplied by the application layer where the ATR is to be stored. For SLL444E, the buffer size must be 4 bytes

### **Return**

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero.

0x14 Not card is detected

### **Note**

It should be called before calling any memory card functions other than HY\_UsbMCardIsInserted and HY\_UsbMCardSetType.

### **Example**

```
byte atr[4];
int ret = HY_UsbMCardPowerUp(atr);
CString s;

if (ret == 0)
    s.Format("Power up successfully.\nATR: %02X %02X %02X %02X", atr[0],
atr[1], atr[2], atr[3]);
else
    s.Format("Status: %02X", ret);

AfxMessageBox(s);
```

## **int HY\_UsbMCardPowerDown ()**

### **Function**

Power down the memory card

### **Parameter**

None

### **Return**

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero.

### **Note**

It should be called before the card is removed from the reader slot.

### **Example**

```
int ret = HY_UsbMCardPowerDown();
if (ret == 1)
    AfxMessageBox("Power down successfully");
else
{
    CString s1;
    s1.Format("Status: %02X", ret);
    AfxMessageBox(s1);
}
```

## **int HY\_UsbMCardVerifyPSC(byte \*pbPSC, int nLength)**

### **Function**

Verify the PSC of SLE4442card.

### **Parameter**

*pbPSC*            [in] The buffer containing the bytes of the PSC  
*nLength*          The number of bytes of the PSC

### **Return**

If the function succeeds, the return value is zero and the value of the Error Counter will become maximized automatically.

If the function fails, the return value is nonzero.

0x16      Incorrect PSC, the value of the Error Counter will be deducted by one automatically.

0x14      No card is detected

0x17      No power up yet

### **Note**

You must call this function before writing/updating any data to the card.

If the card is removed or powered down, the card will become non-verified.

### **Example**

```
byte pbPSC[] = {0xFF, 0xFF, 0xFF};
int nLen = 1;
CString s;
int ret = HY_UsbMCardVerifyPSC(pbPSC, nLen);
if (ret == 0)
    s.Format("Verify successfully");
else if (ret == HY_INCORRECT_PSC) // 0x16
    s.Format("Incorrect PSC ");
else
    s.Format("Status: %02X", ret);

AfxMessageBox(s);
```



## **int HY\_UsbMCardReadCounter(byte \*pbCounter)**

### **Function**

Read the Error Counter of the security memory area in SLE4442 card

### **Parameter**

*pbCounter* [out ] The buffer supplied by the application layer where the Error Counter is to be stored.

For SLL444E, the buffer size must be 1 byte

### **Return**

If the function succeeds, the return value is zero and pbCounter will be updated.

If the function fails, the return value is nonzero.

0x14 Not card is detected

0x17 No power up yet

### **Note**

If Error Counter is 0, the data stored in the SLE4442 card will no longer be updated.

### **Example**

```
byte bCounter;  
int ret = HY_UsbMCardReadCounter(&bCounter);  
CString s;  
  
if (ret == 0)  
    s.Format("Error Counter (Remaining Trials): %d", bCounter);  
else  
    s.Format("Status: %02X", ret);  
  
AfxMessageBox(s);
```

## **int HY\_UsbMCardUpdateCounter(byte byCounter)**

### **Function**

Update the Error Counter of SLE4442 card

### **Parameter**

*byCounter*      The value of the Error Counter to be updated.

### **Return**

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero.

0x14	No card is detected
0x15	Fail to update because it is protected
0x17	No power up yet

### **Note**

It can be updated if byCounter is greater than the current Error Counter value only when the PSC has been verified.

It can be updated if byCounter is less than the current Error Counter value when the PSC has not been verified.

### **Example**

```
int nCounter = 3;
int ret = HY_UsbMCardUpdateCounter(nCounter);
if (ret == 0)
    AfxMessageBox("Set counter successfully.");
else
{
    CString s1;
    s1.Format("Status: %02X", ret);
    AfxMessageBox(s1);
}
```

## **int HY\_UsbMCardReadPSC(byte \*pbPSC, byte \*pbLength)**

### **Function**

Read the PSC from the SLE4442 card

### **Parameter**

*pbPSC* [out] The buffer supplied by the application layer where the PSC is to be stored. At least 3 bytes..

*pbLength* [out] the buffer supplied by the application layer where the length of the PSC is to be stored, 1 byte

### **Return**

If the function succeeds, the return value is zero and pbPSC and pbLength will be updated.

If the function fails, the return value is nonzero.

0x14 No card is detected

0x17 No power up yet

### **Note**

If the card has not been PSC-verified, PSC will be returned as {0x00, 0x00, 0x00}

### **Example**

```
byte byPSC [3], byLength;  
int ret = HY_UsbMCardReadPSC(byPSC, &byLength);  
CString s="", s1="";  
if (ret == 0) {  
    for (int i=0; i< byLength; i++)  
        s1.Format("%s%02X ", s1, byPSC[i]);  
    s.Format("PSC: %s", s1);  
}  
else  
    s.Format("Status: %02X", ret);  
  
AfxMessageBox(s);
```

## **int HY\_UsbMCardUpdatePSC (byte \*pbPSC, int nLength)**

### **Function**

Update the current PSC stored in the SLE4442 card.

### **Parameter**

*pbPSC*            [in] The buffer containing the bytes of the PSC to be updated  
*nLength*           The number of bytes of the PSC

### **Return**

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero.

0x14	No card is detected
0x16	PSC not verified
0x17	No power up yet

### **Note**

The new PSC must be kept if updated successfully.

### **Example**

```
byte pbPSC[] = {0xEE, 0xEE, 0xEE};
int nLen = 3;
CString s;
int ret = HY_UsbMCardUpdatePSC(pbPSC, nLen);
if (ret == 0)
    s.Format("Update successfully");
else
    s.Format("Status: %02X", ret);

AfxMessageBox(s);
```

## **int HY\_UsbMCardReadMemory(int nOffset, byte \*pbReadBuffer, int nReadLen)**

### **Function**

Read the requested bytes from the SLE4442 card

### **Parameter**

<i>pbReadBuffer</i>	[output] The buffer supplied by the application layer where the bytes read are to be stored
<i>nOffset</i>	<i>The offset from which the reading has to take place</i>
<i>nReadLen</i>	<i>The number of bytes to be read</i>

### **Return**

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero.

0xE6	Incorrect parameter, e.g. nReadLen > 128, nOffset < 0, nReadLen <= 0
0xE8	nOffset + nReadLen greater than size of the card
0x14	No card is detected
0x17	No power up yet

### **Note:**

It is only allowed to read at most 128 bytes in one function call.

### **Example**

```
byte pbReadData[128];
CString s="", s1="";
int ret = HY_UsbMCardReadMemory(m_nReadOffset, pbReadData,
m_nReadLength);
if (ret == 0) {
    for (int i=0; i<m_nReadLength; i++)
        s1.Format("%s%02X ", s1, pbReadData[i]);
    s.Format("Read Data: %s", s1);
} else
    s.Format("Status: %02X", ret);
AfxMessageBox(s);
```

## **int HY\_UsbMCardWriteMemory(int nOffset, byte \*pbWritedBuffer, int nWriteLen)**

### **Function**

Read the requested bytes from the SLE4442 card

### **Parameter**

<i>pbWriteBuffer</i>	[input] The buffer supplied by the application layer where the bytes to be written are stored
<i>nOffset</i>	<i>The offset from which the writing has to take place</i>
<i>nWriteLen</i>	<i>The number of bytes to be written</i>

### **Return**

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero.

0x09	Memory fail
0x15	Fail to write because the address is protected
0xE6	Incorrect parameter, e.g. nWriteLen > 128, nOffset < 0, nWriteLen <= 0
0xE8	nOffset + nWriteLen greater than size of the card
0x14	No card is detected
0x16	PSC not verified
0x17	No power up yet

### **Note:**

It is only allowed to write at most 128 bytes in one function call.

The DLL does not internally perform a read back fore very write to verify whether the bytes have actually been written.

### **Example**

```
byte pbWriteData[5] = {0xAA, 0xBB, 0xCC, 0xDD, 0xEE};
CString s="";

int ret = HY_UsbMCardWriteMemory(60, pbWriteData, 5);
if (ret == 0)
    s.Format("Write successfully");
else
```

```
s.Format("Status: %02X", ret);
```

```
AfxMessageBox(s);
```

## **int HY\_UsbMCardSetWriteProtection(int nOffset, int nWriteLen)**

### **Function**

Set the protection level for the first 32 bytes of the SLE4442 card

### **Parameter**

*nOffset*            the offset from which the protection has to begin  
*nWriteLen*        the number of bytes to be protected

### **Return**

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero.

0x09	Memory fail
0x15	Fail to protect because the address is protected
0xE6	Incorrect parameter, e.g. nWriteLen > 32, nOffset < 0, nWriteLen <= 0
0xE8	nOffset + nWriteLen greater 32
0x14	No card is detected
0x16	PSC not verified
0x17	No power up yet

### **Note**

The write protection feature can permanently protect the data from being altered again. So application developers must be sure enough before calling this API function.

### **Example**

```
int offset=5, len=2;
CString s;
int ret = HY_UsbMCardSetWriteProtection(offset, len);
if (ret == 0)
    s.Format("Write successfully");
else
    s.Format("Status: %02X", ret);
AfxMessageBox(s);
```



## **int HY\_UsbMCardGetWriteProtection(byte \*pbProtected, byte \*pbLength)**

### **Function**

Get the protection level for the first 32 bytes of the SLE4442 card

### **Parameter**

*pbProtected* [out] The buffer supplied by the application layer where the protection level are to be stored, at least 4 bytes  
The 1st bit (counting from the LSB) of the 1st byte of the buffer shows the protection level of the 1st byte of the card.  
The 2nd bit of the 1st byte of the buffer shows the protection level of the 2nd byte of the card.  
...  
The 8th bit of the 1st byte of the buffer shows the protection level of the 8th byte of the card.  
The 1st bit (counting from the LSB) of the 2nd byte of the buffer shows the protection level of the 9th byte of the card.  
The 2nd bit of the 2nd byte of the buffer shows the protection level of the 10th byte of the card.  
...  
The 8th bit of the 4th byte of the buffer shows the protection level of the 32th byte of the card.  
0 means protected.  
1 means unprotected.  
For example, if pbProtected is assigned with {0xF0, 0xFF, 0x1F, 0xFE}, it means only the 1st,2nd,3rd,4th,13th,22nd,23rd,24th and 25th of the memory are protected.

*pbLength* [out] the buffer supplied by the application layer where the size of pbProtected to be stored, 1 byte

### **Return**

If the function succeeds, the return value is zero and pbProtected and pbLength will be updated.

If the function fails, the return value is nonzero.

0x14 No card is detected

0x17      No power up yet

### Example

```
byte byProtected[4], byLength=0, byTmp=0;
CString s;

int ret = HY_UsbMCardGetWriteProtection(byProtected, &byLength);
if (ret == 0) {
    for (int i=0; i< byLength; i++) {
        byTmp = byProtected[i];
        for (int j=0; j<8; j++) {
            if ((byTmp & 0x01) == 0) {
                // the (j+i*8+1)th byte is protected
            }
            else {
                // the (j+i*8+1)th byte is unprotected
            }
            byTmp = byTmp >> 1;
        }
    }
}
else
    s.Format("Status: %02X", ret);

AfxMessageBox(s);
```

## **int HY\_UsbSetLed(bool isOn, int nDuration)**

### **Function**

Set the LED on/off only if the LED is supported.

### **Parameter**

*isOn*                true when turning LED on and false when turning LED off  
*nDuration*        the duration of LED to be turned on. [unit: ms]  
                      if it is zero, the LED will be turned on forever.  
                      If isOn is false, it can be any value.

### **Return**

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero.

### **Note**

Other API functions are not allowed to be called before the LED turns off.

### **Example**

```
int offset=5, len=2;
CString s;
int ret = HY_UsbSetLed(true, 300); // turn the LED on for 300 ms
if (ret == 0)
    s.Format("Set LED successfully");
else
    s.Format("Status: %02X", ret);
AfxMessageBox(s);
```

## Summary of Return Codes

	Values	Definitions
HY_SUCCESS	0x00	Success
HY_CHECKSUM_ERR	0xF0	Checksum error
HY_COM_LINE_ERR	0xF2	Communication error
HY_MEM_FAIL	0x09	Memory fail
HY_EEPROM_MISSING	0x0A	EEPROM is missing
HY_MIFARE_COLLISION	0x1E	Mifare collision
HY_MIFARE_NO_SAK	0x1F	No SAK
HY_MIFARE_LOGIN_FAIL	0x20	Login fail
HY_MIFARE_NO_TAG	0x21	No tag
HY_MIFARE_NO_AUTH	0x22	No authorization
HY_MIFARE_READ_FAIL	0x23	Fail to read
HY_MIFARE_WRITE_FAIL	0x24	Fail to write
HY_INIT_FAIL	0xE1	Fail to initialize the usb device
HY_SEND_FAIL	0xE2	Fail to send the command to the device
HY_TIMEOUT_ERR	0xE3	Timeout to receive the response from the device
HY_FEATURE1_ERR	0xE4	USB communication error
HY_FEATURE2_ERR	0xE5	USB communication error
HY_INVALID_PARA	0xE6	Invalid parameter
HY_RECV_FAIL	0xE7	Fail to receive any response from the device
HY_OVERLIMIT	0xE8	Exceeds the memory limit
HY_UNKNOWN_ERR	0xEF	Other error
HY_NO_CARD	0x14	No memory card is detected
HY_INCORRECT_PSC	0x16	Incorrect PSC
HY_NO_POWER_UP	0x17	The memory card is not powered up yet
HY_INCORRECT_CARD_TYPE	0x31	Invalid memory card type