IT-100MU/(R)
IT-101MU NFC Reader

軟體介面開發文件

WinNfc API

Version 1.7.5

訊想科技股份有限公司

2014 年 12 月 15 日

文件制/修訂紀錄

版次	日期	制/修訂說明	作者
V 1.00	200/11/03	First Version 只支援 32 位元 Windows	Roman
V1.40	2011/04/25	採用 libUSB 可支援 32/64 位元 Windows 作業系統	Roman
V1.41	2012/01/02	修正部分 Bug, 增加底層 Error Code 說明	Roman
V1.6	2012/08/16	支援新 Model IT-100MU(R) 增加 IC 卡和 SAM 卡的 API	Roman
V1.6.1	2012/10/22	增加 IT-100MU(R)底層 Error Code 說明	Roman
V.1.7.2	2013/06/13	支援 Mode: IT-101MU 增加 NfcGetParamete Function	Roman
V.1.7.5	2014/12/15	支援 Mifare Ultra Light card	Roman

目 錄

1.WinNfc API 函式清單

1.1 建立/釋放函式庫

NfcEstablishContext

建立並初始化 WinNfc 函式庫。

```
WINNFC_API LONG WINAPI
NfcEstablishContext(
    IN   DWORD dwScope,
    IN   LPCVOID pvReserved1,
    IN   LPCVOID pvReserved2,
    OUT LPNFCCONTEXT phContext);
```

參數:

dwScope:個人或系統模式, 此參數暫時無用途。

pvReserved1:保留參數。

pvReserved2: 保留參數。

phContext:回傳 WinNfc Handle，後續操作其他函式使用。

回覆值:

成功: NFC_S_SUCCESS

失敗: 錯誤代碼(詳見 Error Code 清單)

範例：

```
NFCCONTEXT phCtx = NULL;
LONG rc = 0;

//initialize NFC functions
rc = NfcEstablishContext(0, 0, 0, &phCtx);
if (rc) {
    ShowError(phCtx, rc);
    return 0;
}

printf("initialize NFC functions successfully\n", rc);
```


NfcReleaseContext

結束並釋放 WinNfc 函式庫,

```
WINNFC_API LONG WINAPI
NfcReleaseContext(
    IN        NFCCONTEXT hContext);
```

參數:

phContext: WinNfc Handle。

回覆值:

成功: NFC_S_SUCCESS

失敗: 錯誤代碼(詳見 Error Code 清單)

範例:

```
NfcReleaseContext(ctx);
printf("release NFC functions successfully\n", rc);
```


1.2 讀卡機相關函式

NfcConnect

連接讀卡機。

```
WINNFC_API LONG WINAPI
NfcConnect(
    IN      NFCCONTEXT hContext,
    IN      LPCSTR szReader,
    IN      DWORD dwShareMode);
```

參數:

hContext: WinNfc Handle。

szReader: 讀卡機名稱，輸入 NULL 自動尋找讀卡機；輸入"InfoThink IT-100MU 0"代表連接第一台讀卡機；輸入"InfoThink IT-100MU 1" 代表連接第二台讀卡機，以此內推。

dwShareMode:連接模式，此參數暫時無用途。

回覆值:

成功: 0

失敗: 錯誤代碼(詳見 Error Code 清單)

注意：若出現**讀卡機被占用**錯誤訊息，請檢查你是否有安裝 NfcCode 應用程式，此程式會常駐在系統，建議您移除該程式以方便系統開發。

範例：

```
//Auto connect to IT-100MU reader
rc = NfcConnect(phCtx, NULL, 0);

if (rc) {
    ShowError(phCtx, rc);
    return 0;
}

printf("auto connect NFC reader successfully\n");
```


NfcDisConnect

中斷讀卡機連線。

```
WINNFC_API LONG WINAPI
NfcDisconnect(
    IN      NFCCONTEXT hContext,
    IN      DWORD dwDisposition);
```

參數:

hContext: WinNfc Handle。

dwDisposition: 中斷模式，此參數暫時無用途。

回覆值:

成功: 0

失敗: 錯誤代碼(詳見 Error Code 清單)

範例：

```
//disconnect IT-100MU reader
rc = NfcDisconnect(phCtx, 0);

if (rc) {
    ShowError(phCtx, rc);
    return 0;
}

printf("disconnect successfully\n);
```


NfcGerParameter

讀取相關參數。

```
WINNFC_API LONG WINAPI
NfcGetParameter(
    IN        NFCCONTEXT hContext,
    IN        DWORD dwType,
    IN OUT    void* value,
    IN OUT    DWORD* dwLength);
```

參數:

hContext: WinNfc Handle。

dwType: 參數編號，請參考參數表。

Value: 參數值，需注意型態，請參考參數表

dwLength: 參數

回覆值:

成功: 0

失敗: 錯誤代碼(詳見 Error Code 清單)

範例：

```
char szReaderName[64] = {0};
dwReaderName = 64;
rc = NfcGetParameter(phCtx, NFC_PARAMETER_READER_NAME,
                             szReaderName, &dwReaderName);

printf("get reader name = %s\n", szReaderName);
```

參數表：

dwType	參數型態	說明
NFC_PARAMETER_READER_NAME	char	取得讀卡機名稱
NFC_PARAMETER_ESCAPE_COMMAND	BOOL	取得 Enable Escape Command

NfcLEDControl

開關 IT-100MU 讀卡機左上方藍色和紅色 LED 燈 。

PS: IT-100MU(R)需先呼叫 NfcSelectCard 成功後才能控制 LED

```
WINNFC_API LONG WINAPI
NfcLEDControl(
    IN NFCCONTEXT hContext,
    IN BYTE bState);
```

參數:

hContext: WinNfc Handle。

bState:開關 LED 燈參數，bit-1 控制紅燈 ON 或 OFF, bit-2 控制藍燈 ON 或 OFF。

回覆值:

成功: 0

失敗: 錯誤代碼(詳見 Error Code 清單)

範例：

```
//LED test
printf("LED BLUE ON\n");
rc = NfcLEDControl(phCtx, LED_BLUE);
Sleep(500);
printf("LED RED ON\n");
rc = NfcLEDControl(phCtx, LED_RED);
Sleep(500);
printf("LED ALL ON\n");
rc = NfcLEDControl(phCtx, LED_ALL);
Sleep(500);
printf("LED ALL OFF\n");

rc = NfcLEDControl(phCtx, LED_OFF);
```


NfcBuzzerControl

開關 IT-100MU 讀卡機蜂鳴器。

PS: IT-100MU(R)需先呼叫 NfcSelectCard 成功後才能控制 Buzzer

```
WINNFC_API LONG WINAPI
NfcBuzzerControl(
    IN NFCCONTEXT hContext,
    IN BYTE bState);
```

參數:

hContext: WinNfc Handle。

bState:開關 Buzzer 參數，0x00 打開 Buzzer 或 0x08 關閉 Buzzer。

回覆值:

成功: 0

失敗: 錯誤代碼(詳見 Error Code 清單)

範例：

```
//BUZZER test
NfcBuzzerControl(phCtx, BUZZER_ON);
Sleep(50);

NfcBuzzerControl(phCtx, BUZZER_OFF);
```


1.3 感應卡片相關函式

NfcSelectCard

由於天線的訊號控制在一定的範圍，當卡片距離讀卡機 4 公分內，呼叫此函式搜尋卡片，找到卡後即可透過 NfcTransmit 函式與卡片交換資料。若天線範圍內有多張卡片，IT-100MU 同時間只能存取 1 張卡。

```
WINNFC_API LONG WINAPI
NfcSelectCard(
    IN      NFCCONTEXT hContext,
    IN      DWORD dwCardTypes,
    IN      PBYTE pbInitData,
    IN      DWORD dwInitDataLen);
```

參數:

hContext: WinNfc Handle。

dwCardTypes: 卡別。

pbInitData: 輸入資料(保留參數)。

dwInitDataLen: 輸入資料長度(保留參數)。

回覆值:

成功: 0

失敗: 錯誤代碼(詳見 Error Code 清單)

範例：

```
//Find card and select the first card
rc = NfcSelectCard(phCtx, NFC_CARDTYPE_MIFARE, NULL, 0);

if (rc) {
    ShowError(phCtx, rc);
    return 0;
}

printf("select card successfully\n");
```


NfcDeSelectCard

完成卡片資料存取後，可以呼叫此函式釋放此卡片。

```
WINNFC_API LONG WINAPI
NfcDeSelectCard(
    IN      NFCCONTEXT hContext);
```

參數:

hContext: WinNfc Handle。

回覆值:

成功: 0

失敗: 錯誤代碼(詳見 Error Code 清單)

範例：

```
//deselect the card if you don't use it anymore
rc = NfcDeSelectCard(phCtx);
if (rc) {
    ShowError(phCtx, rc);
    return 0;
}

printf("DeSelect card successfully\n");
```


以下函式是使用 WinScard 來連接感應式卡片，四個函式為 WinScard 的簡化版本，您也可以直接使用 WinScard 函式集。

(R)NFC_ConnectCard

連接卡片。此函式僅可適用 IT-100MU(R)讀卡機。

若您對 WinScard 很熟可以參考 SCardConnect 函式說明。

```
WINNFC_API LONG WINAPI
NFC_ConnectCard(
    IN NFCCONTEXT hContext,
    IN DWORD dwShareMode,
    DWORD dwPreferredProtocols);
```

參數:

hContext: WinNfc Handle。

dwShareMode: 輸入 0 採用預設參數 SCARD_SHARE_SHARED。

dwPreferredProtocols: 輸入 0 採用預設參數 SCARD_PROTOCOL_Tx

回覆值:

成功: 0

失敗: 錯誤代碼(詳見 WinsCard Error Code 清單)

範例：

```
//Connect card
rc = NFC_ConnectCard(phCtx, 0, 0);

if (rc) {
    ShowError(phCtx, rc);
    return 0;
}

printf("connect card successfully\n");
```


(R)NFC_GetATR

取得卡片 Answer to Rest(ATR)資料。此函式僅可適用 IT-100MU(R) 讀卡機。

若您對 WinScard 很熟可以參考 SCardStatus 函式說明。

```
WINNFC_API LONG WINAPI
NFC_GetATR(
    IN NFCCONTEXT hContext,
    OUT PBYTE pAtr,
    OUT LPDWORD pLength,
    OUT LPDWORD pActiveProtocol);
```

參數:

hContext: WinNfc Handle。

pAtr: 卡片 ATR 資料。

pLength: 卡片 ATR 資料長度。

pActiveProtocol: 卡片啟用的通訊協定通常為 T=0 或 T=1。

回覆值:

成功: 0

失敗: 錯誤代碼(詳見 WinsCard Error Code 清單)

範例:

```
//get ATR
BYTE byATR[64] = {0};
DWORD dwLen;
DWORD dwActiveProtocol;
rc = NFC_GetATR(phCtx, byATR, &dwLen, &dwActiveProtocol);

if (rc) {
    ShowError(phCtx, rc);
    return 0;
}

printf("get ATR successfully\n");
```


(R)NFC_SendAPDU

符合 ISO7816 規格的卡片，可透過 T=0 或 T=1 的通訊協定將 APDU 指令傳送至卡片。此函式僅可適用 IT-100MU(R)讀卡機。

若您對 WinScard 很熟可以參考 SCardTransmit 函式說明。

```
WINNFC_API LONG WINAPI
NFC_SendAPDU(
    IN NFCCONTEXT hContext,
    IN LPCBYTE pbSendBuffer,
    IN DWORD cbSendLength,
    OUT LPBYTE pbRecvBuffer,
    IN OUT LPDWORD pcbRecvLength);
```

參數:

hContext: WinNfc Handle。

pbSendBuffer:送出資料。

cbSendLength:送出資料長度。

pbRecvBuffer:接收資料。

pcbRecvLength: 接收資料長度。

回覆值:

成功: 0

失敗: 錯誤代碼(詳見 WinsCard Error Code 清單)

範例：

```
//Seelect AID APDU
BYTE SelectAID[] = {0x00,0xA4,0x04,0x00,0x07,0xA0,0x00,0x00,0x00,0x03,0x00,0x00,0x00};

BYTE Rec[262] = {0};
DWORD dwLen = 262;

printf("Select AID \n");
ShowData(SelectAID, 13);
```



```
//send APDU command
rc = NFC_SendAPDU(phCtx, (LPCBYTE)SelectAID, 13, Rec, &dwLen);
if (rc) {
     ShowError(phCtx, rc);
     return 0;
}

printf("get response data successfully\n");
ShowData(Rec, dwLen);
```


(R)NFC_DisconnectCard

關閉卡片連線。此函式僅可適用 IT-100MU(R)讀卡機。

若您對 WinScard 很熟可以參考 SCardDisconnect 函式說明。

```
WINNFC_API LONG WINAPI
NFC_DisconnectCard(
    IN NFCCONTEXT hContext,
    IN DWORD dwDisposition);
```

參數:

hContext: WinNfc Handle。

dwDisposition: 輸入 0 採用預設參數 SCARD_LEAVE_CARD。

回覆值:

成功: 0

失敗: 錯誤代碼(詳見 WinsCard Error Code 清單)

範例:

```
//disconnect card
rc = NFC_DisconnectCard(phCtx, 0);
if (rc) {
    ShowError(phCtx, rc);
    return 0;
}

printf("disconnect card successfully\n");
```


1.4 IC 卡片相關函式

以下函式是 WinScard 的簡化版本，您也可以直接使用 WinScard 函式集。

ICC_ConnectCard

連接卡片。若您對 WinScard 很熟可以參考 SCardConnect 函式說明。

```
WINNFC_API LONG WINAPI
ICC_ConnectCard(
    IN NFCCONTEXT hContext,
    IN DWORD dwShareMode,
    DWORD dwPreferredProtocols);
```

參數:

hContext: WinNfc Handle。

dwShareMode: 輸入 0 採用預設參數 SCARD_SHARE_SHARED。

dwPreferredProtocols: 輸入 0 採用預設參數 SCARD_PROTOCOL_Tx

回覆值:

成功: 0

失敗: 錯誤代碼(詳見 WinsCard Error Code 清單)

範例：

```
//Connect card
rc = ICC_ConnectCard(phCtx, 0, 0);

if (rc) {
    ShowError(phCtx, rc);
    return 0;
}

printf("connect card successfully\n");
```


ICC_GetATR

取得卡片 Answer to Rest(ATR)資料。若您對 WinScard 很熟可以參考 SCardStatus 函式說明。

```
WINNFC_API LONG WINAPI
ICC_GetATR(
    IN NFCCONTEXT hContext,
    OUT PBYTE pAtr,
    OUT LPDWORD pLength,
    OUT LPDWORD pActiveProtocol);
```

參數:

hContext: WinNfc Handle。

pAtr: 卡片 ATR 資料。

pLength: 卡片 ATR 資料長度。

pActiveProtocol: 卡片啟用的通訊協定通常為 T=0 或 T=1。

回覆值:

成功: 0

失敗: 錯誤代碼(詳見 WinsCard Error Code 清單)

範例:

```
//get ATR
BYTE byATR[64] = {0};
DWORD dwLen;
DWORD dwActiveProtocol;
rc = ICC_GetATR(phCtx, byATR, &dwLen, &dwActiveProtocol);

if (rc) {
    ShowError(phCtx, rc);
    return 0;
}

printf("get ATR successfully\n");
```


ICC_SendAPDU

符合 ISO7816 規格的卡片，可透過 T=0 或 T=1 的通訊協定將 APDU 指令傳送至卡片。

若您對 WinScard 很熟可以參考 SCardTransmit 函式說明。

```
WINNFC_API LONG WINAPI
ICC_SendAPDU(
    IN NFCCONTEXT hContext,
    IN LPCBYTE pbSendBuffer,
    IN DWORD cbSendLength,
    OUT LPBYTE pbRecvBuffer,
    IN OUT LPDWORD pcbRecvLength);
```

參數:

hContext: WinNfc Handle。

pbSendBuffer:送出資料。

cbSendLength:送出資料長度。

pbRecvBuffer:接收資料。

pcbRecvLength: 接收資料長度。

回覆值:

成功: 0

失敗: 錯誤代碼(詳見 WinsCard Error Code 清單)

範例：

```
//Seelect AID APDU
BYTE SelectAID[] = {0x00,0xA4,0x04,0x00,0x07,0xA0,0x00,0x00,0x00,0x03,0x00,0x00,0x00};

BYTE Rec[262] = {0};
DWORD dwLen = 262;

printf("Select AID \n");
ShowData(SelectAID, 13);
```



```
//send APDU command
rc = ICC_SendAPDU(phCtx, (LPCBYTE)SelectAID, 13, Rec, &dwLen);
if (rc) {
     ShowError(phCtx, rc);
     return 0;
}

printf("get response data successfully\n");
ShowData(Rec, dwLen);
```


ICC_DisconnectCard

關閉卡片連線。。

若您對 WinScard 很熟可以參考 SCardDisconnect 函式說明。

```
WINNFC_API LONG WINAPI
ICC_DisconnectCard(
    IN NFCCONTEXT hContext,
    IN DWORD dwDisposition);
```

參數:

hContext: WinNfc Handle。

dwDisposition: 輸入 0 採用預設參數 SCARD_LEAVE_CARD。

回覆值:

成功: 0

失敗: 錯誤代碼(詳見 WinsCard Error Code 清單)

範例:

```
//disconnect card
rc = ICC_DisconnectCard(phCtx, 0);
if (rc) {
    ShowError(phCtx, rc);
    return 0;
}

printf("disconnect card successfully\n");
```


1.5 SAM 卡片相關函式

以下函式是 WinScard 的簡化版本，您也可以直接使用 WinScard 函式集。
注意：您必須安裝 SAM 讀卡機的驅動程式(SAM driver)，才能使用 SAM 模組。

SAM_ConnectCard

連接卡片。若您對 WinScard 很熟可以參考 SCardConnect 函式說明。

```
WINNFC_API LONG WINAPI
SAM_ConnectCard(
    IN NFCCONTEXT hContext,
    IN DWORD dwShareMode,
    DWORD dwPreferredProtocols);
```

參數:

hContext: WinNfc Handle。

dwShareMode: 輸入 0 採用預設參數 SCARD_SHARE_SHARED。

dwPreferredProtocols: 輸入 0 採用預設參數 SCARD_PROTOCOL_Tx

回覆值:

成功: 0

失敗: 錯誤代碼(詳見 WinsCard Error Code 清單)

範例：

```
//Connect card
rc = SAM_ConnectCard(phCtx, 0, 0);

if (rc) {
    ShowError(phCtx, rc);
    return 0;
}

printf("connect card successfully\n");
```


SAM_GetATR

取得卡片 Answer to Rest(ATR)資料。若您對 WinScard 很熟可以參考 SCardStatus 函式說明。

```
WINNFC_API LONG WINAPI
SAM_GetATR(
    IN NFCCONTEXT hContext,
    OUT PBYTE pAtr,
    OUT LPDWORD pLength,
    OUT LPDWORD pActiveProtocol);
```

參數:

hContext: WinNfc Handle。

pAtr: 卡片 ATR 資料。

pLength: 卡片 ATR 資料長度。

pActiveProtocol: 卡片啟用的通訊協定通常為 T=0 或 T=1。

回覆值:

成功: 0

失敗: 錯誤代碼(詳見 WinsCard Error Code 清單)

範例:

```
//get ATR
BYTE byATR[64] = {0};
DWORD dwLen;
DWORD dwActiveProtocol;
rc = SAM_GetATR(phCtx, byATR, &dwLen, &dwActiveProtocol);

if (rc) {
    ShowError(phCtx, rc);
    return 0;
}

printf("get ATR successfully\n");
```


SAM_SendAPDU

符合 ISO7816 規格的卡片，可透過 T=0 或 T=1 的通訊協定將 APDU 指令傳送至卡片。

若您對 WinScard 很熟可以參考 SCardTransmit 函式說明。

```
WINNFC_API LONG WINAPI
SAM_SendAPDU(
    IN NFCCONTEXT hContext,
    IN LPCBYTE pbSendBuffer,
    IN DWORD cbSendLength,
    OUT LPBYTE pbRecvBuffer,
    IN OUT LPDWORD pcbRecvLength);
```

參數:

hContext: WinNfc Handle。

pbSendBuffer:送出資料。

cbSendLength:送出資料長度。

pbRecvBuffer:接收資料。

pcbRecvLength: 接收資料長度。

回覆值:

成功: 0

失敗: 錯誤代碼(詳見 WinsCard Error Code 清單)

範例：

```
//Seelect AID APDU
BYTE SelectAID[] = {0x00,0xA4,0x04,0x00,0x07,0xA0,0x00,0x00,0x00,0x03,0x00,0x00,0x00};

BYTE Rec[262] = {0};
DWORD dwLen = 262;

printf("Select AID \n");
ShowData(SelectAID, 13);

//send APDU command
```



```
    rc = SAM_SendAPDU(phCtx, (LPCBYTE)SelectAID, 13, Rec, &dwLen);
    if (rc) {
        ShowError(phCtx, rc);
        return 0;
    }

    printf("get response data successfully\n");
    ShowData(Rec, dwLen);
```


SAM_DisconnectCard

關閉卡片連線。。

若您對 WinScard 很熟可以參考 SCardDisconnect 函式說明。

```
WINNFC_API LONG WINAPI
SAM_DisconnectCard(
    IN NFCCONTEXT hContext,
    IN DWORD dwDisposition);
```

參數:

hContext: WinNfc Handle。

dwDisposition: 輸入 0 採用預設參數 SCARD_LEAVE_CARD。

回覆值:

成功: 0

失敗: 錯誤代碼(詳見 WinsCard Error Code 清單)

範例:

```
//disconnect card
rc = SAM_DisconnectCard(phCtx, 0);
if (rc) {
    ShowError(phCtx, rc);
    return 0;
}

printf("disconnect card successfully\n");
```


1.6 MIFARE 卡相關函式

MIFARE 卡格式的相關文件可以參考：

(1) MIFARE Classic 1K / 4K: NXP MIFARE 1K/4K 卡規格文件

(2) MIFARE Ultralight.pdf: NXP MIFARE Ultra-Light 卡規格文件

Mifare_GetCardID

取得 Mifare 卡號。

```
WINNFC_API LONG WINAPI
Mifare_GetCardID(
    IN   NFCCONTEXT hContext,
    IN   LPSTR mszID);
```

參數:

hContext: WinNfc Handle。

mszID: 卡號字串(8~14 個 bytes)。

回覆值:

成功: 0

失敗: 錯誤代碼(詳見 Error Code 清單)

範例：

```
//get Mifare Card ID
char szCardID[9] = {0};
rc = Mifare_GetCardID(phCtx, szCardID);
if (rc) {
     ShowError(phCtx, rc);
     return 0;
}


printf("get MIFARE Card ID: %s \n", szCardID);
```


Mifare_KeyAuthority

Mifare 1K/4K 卡每個 Sector 有 4 個 Block, 由兩把金鑰分別為 KeyA 和 KeyB 控管權限，此函式可用來認證 KeyA 或 KeyB，驗證成功後即會擁有特定權限(請參考 Mifare 規格)。

```
WINNFC_API LONG WINAPI
Mifare_KeyAuthority(
    IN   NFCCONTEXT hContext,
    IN BYTE bBlock,
    IN BYTE bKeyType,
    IN LPCBYTE pbKey);
```

參數:

hContext: WinNfc Handle。

bBlock:卡片 Block 編號位置(每個 Sector 有 4 個 Block, 0~3 為 Sector1, 4~7 為 Sector2 以此類推)。

bKeyType: KeyA 或 KeyB。

pbKey:金鑰資料(6 個 bytes, 預設值為 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF)。

回覆值:

成功: 0

失敗: 錯誤代碼(詳見 Error Code 清單)

範例：

```
//authenticate key A on Block N given
rc = Mifare_KeyAuthority(phCtx, Block, KEY_TYPE_A, KeyA);
if (rc) {
    ShowError(phCtx, rc);
    return 0;
}

printf("authenticate key A successfully\n");
```


Mifare_ReadBlock

讀取第 N 個 Block 裡的 16 個 Byte 資料，例如 1K 的 Mifare 卡片有 16 個 Sector，每個 Sector 有 4 個 Block，所有共有 0~63 個 Block;4K 的 Mifare 卡片有 0~255 個 Block;Ultra-Light 卡片有 0~15 個 Page。

```
WINNFC_API LONG WINAPI
Mifare_ReadBlock(
    IN   NFCCONTEXT hContext,
    IN BYTE bBlock,
    IN OUT LPBYTE pbData,
    IN OUT LPDWORD pcbLength);
```

參數:

hContext: WinNfc Handle。

bBlock:卡片 Block 編號位置;Ultra-Light 卡為 Page 編號位置。

pbData: 讀取資料的 Buff。

pcbLength:讀取資料長度。**Ultra-Light 卡一次讀 4 個 page 資料。**

回覆值:

成功: 0

失敗: 錯誤代碼(詳見 Error Code 清單)

範例：

```
//read data from Block N / Page N
BYTE data[16] = {0};
DWORD dwLen = 16;
//for Ultra-Light Card this will read 16 bytes data of 4 pages
rc = Mifare_ReadBlock(phCtx, Block, data, &dwLen);

if (rc) {
    ShowError(phCtx, rc);
    return 0;
}
printf("read data from block %d successfully\n", Block);
ShowData(data, 16);
```


Mifare_WriteBlock

寫入 16 個 Byte 資料到第 N 個 Block，例如 1K 的 Mifare 卡片有 16 個 Sector，每個 Sector 有 4 個 Block，所有共有 0~63 個 Block;4K 的 Mifare 卡片有 0~255 個 Block; Ultra-Light 卡片有 0~15 個 Page，每個 Page 只能寫入 4 個 Byte 資料。

```
WINNFC_API LONG WINAPI
Mifare_WriteBlock(
    IN   NFCCONTEXT hContext,
    IN BYTE bBlock,
    IN LPCBYTE pbData,
    IN DWORD cbLength);
```

參數:

hContext: WinNfc Handle。

bBlock:卡片 Block 編號位置;Ultra-Light 卡為 Page 編號位置。

pbData: 寫入資料的 Buff。

cbLength:寫入資料長度。**Ultra-Light 卡寫入長度必須是 4 個 Byte。**

回覆值:

成功: 0

失敗: 錯誤代碼(詳見 Error Code 清單)

範例：

```
//write data to Block N / Page N
PBYTE pData = (PBYTE)"\x01\x02\x03\x04\x05\x06\x07\x08\x01\x02\x03\x04\x05\x06\x07\x08";
rc = Mifare_WriteBlock(phCtx, Block, pData, 16);

//change to 4 byte if you are writing data to Ultra-Light Card
//rc = Mifare_WriteBlock(phCtx, Block, pData, 4);

if (rc) {
    ShowError(phCtx, rc);
    return 0;
}

printf("write data to block %d successfully\n", Block);
```


Mifare_ReadValue

Mifare 定義一種 16 bytes 的 Value Block，可用來儲存數值(通常為數位錢包，這種 Value Block 可以透過特定指令來加值(Increase Value)或扣款(Decrease Value)，此函式可以用來讀取 Value Block 的數值。

```
WINNFC_API LONG WINAPI
Mifare_ReadValue(
    IN   NFCCONTEXT hContext,
    IN BYTE bBlock,
    IN LPDWORD pcbValue);
```

參數:

hContext: WinNfc Handle。

bBlock:卡片 Block 編號位置。

pcbValue: Block 裡所存放的數值。

回覆值:

成功: 0

失敗: 錯誤代碼(詳見 Error Code 清單)

範例：

```
dwBalance = 0;
rc = Mifare_ReadValue(phCtx, Block, &dwBalance);
if (rc) {
    ShowError(phCtx, rc);
    return 0;
}

printf("now block %d balance is %d\n", Block, dwBalance);
```


Mifare_WriteValue

寫入數值到第 N 個 Value Block，例如 1K 的 Mifare 卡片有 16 個 Sector，每個 Sector 有 4 個 Block，所有共有 0~63 個 Block。

```
WINNFC_API LONG WINAPI
Mifare_WriteValue(
    IN   NFCCONTEXT hContext,
    IN BYTE bBlock,
    IN LPDWORD pcbValue);
```

參數:

hContext: WinNfc Handle。

bBlock:卡片 Block 編號位置。

pcbValue: 寫入的數值。

回覆值:

成功: 0

失敗: 錯誤代碼(詳見 Error Code 清單)

範例：

```
//write value on Block N given
rc = Mifare_WriteValue(phCtx, Block, Value);
if (rc) {
     ShowError(phCtx, rc);
     return 0;
}

printf("write value %d successfully\n", Value);
```


Mifare_IncreaseValue

針對第 N 個 Value Block 增加數值(加值)，此函式會加值的結果暫時放在記憶體空間，等呼叫 Mifare_TransferValue 後才會把結果存放到指定的 Block。

```
WINNFC_API LONG WINAPI
Mifare_IncreaseValue(
    IN   NFCCONTEXT hContext,
    IN BYTE bBlock,
    IN DWORD pcbValue);
```

參數:

hContext: WinNfc Handle。

bBlock:卡片 Block 編號位置。

pcbValue: 欲增加的數值。

回覆值:

成功: 0

失敗: 錯誤代碼(詳見 Error Code 清單)

範例：

```
/* 加值100 元/點*/
rc = Mifare_IncreaseValue(phCtx, Block, 200);
if (rc) {
    ShowError(phCtx, rc);
    return 0;
}
rc = Mifare_TransferValue(phCtx, Block);
if (rc) {
    ShowError(phCtx, rc);
    return 0;
}

printf("add 200 to block %d\n", Block);
```


Mifare_DecreaseValue

針對第 N 個 Value Block 減少數值(扣款)，此函式會扣款的結果暫時放在記憶體空間，等呼叫 Mifare_TransferValue 後才會把結果存放到指定的 Block。

```
WINNFC_API LONG WINAPI
Mifare_DecreaseValue(
    IN   NFCCONTEXT hContext,
    IN BYTE bBlock,
    IN DWORD pcbValue);
```

參數:

hContext: WinNfc Handle。

bBlock:卡片 Block 編號位置。

pcbValue: 欲減少的數值。

回覆值:

成功: 0

失敗: 錯誤代碼(詳見 Error Code 清單)

範例：

```
/* 扣款100 元/點*/
rc = Mifare_DecreaseValue(phCtx, Block, 100);
if (rc) {
    ShowError(phCtx, rc);
    return 0;
}
rc = Mifare_TransferValue(phCtx, Block);
if (rc) {
    ShowError(phCtx, rc);
    return 0;
}
printf("remove 100 from block %d\n", Block);
```


Mifare_TransferValue

將執行 Mifare_IncreaseValue 和 Mifare_DecreaseValue 的結果(暫存的記憶體)存放到指定的 Block。 。

```
WINNFC_API LONG WINAPI
Mifare_TransferValue(
    IN   NFCCONTEXT hContext,
    IN BYTE bBlock);
```

參數:

hContext: WinNfc Handle。

bBlock:卡片 Block 編號位置。

回覆值:

成功: 0

失敗: 錯誤代碼(詳見 Error Code 清單)

範例：

```
/* 扣款100 元/點*/
rc = Mifare_DecreaseValue(phCtx, Block, 100);
if (rc) {
    ShowError(phCtx, rc);
    return 0;
}
rc = Mifare_TransferValue(phCtx, Block);
if (rc) {
    ShowError(phCtx, rc);
    return 0;
}

printf("remove 100 from block %d\n", Block);
```


Mifare_RestoreValue

在 MIFARE 電子錢包的設計上，為了安全起見會採用兩個 Block 來存放同一個數值，另一個數值作為備份用途，同時若有交易不明確的情況下，可以取消交易並將資料從備份的 Block 復原回去。

```
WINNFC_API LONG WINAPI
Mifare_RestoreValue(
    IN   NFCCONTEXT hContext,
    IN BYTE bBlock);
```

參數:

hContext: WinNfc Handle。

bBlock:卡片 Block 編號位置。

回覆值:

成功: 0

失敗: 錯誤代碼(詳見 Error Code 清單)

範例：

```
rc = Mifare_RestoreValue(phCtx, Block + 1);
if (rc) {
    ShowError(phCtx, rc);
    return 0;
}
rc = Mifare_TransferValue(phCtx, Block);
if (rc) {
    ShowError(phCtx, rc);
    return 0;

}
```


1.7 ISO14443-4A/B

ISO14443_SendAPDU

符合 ISO1443-4A/B 規格的卡片，可透過 T=CL 的通訊協定將 APDU 指令傳送至卡片。

ISO1443-4 詳細規格可參考 ISOIEC 14443-4.pdf

```
WINNFC_API LONG WINAPI
ISO14443_SendAPDU(
    IN   NFCCONTEXT hContext,
    IN LPCBYTE pbSendBuffer,
    IN DWORD cbSendLength,
    OUT LPBYTE pbRecvBuffer,
    IN OUT LPDWORD pcbRecvLength);
```

參數:

hContext: WinNfc Handle。

pbSendBuffer:送出資料。

cbSendLength:送出資料長度。

pbRecvBuffer:接收資料。

pcbRecvLength: 接收資料長度。

回覆值:

成功: 0

失敗: 錯誤代碼(詳見 Error Code 清單)

範例：

```
//Seelect AID APDU
BYTE SelectAID[] = {0x00,0xA4,0x04,0x00,0x07,0xA0,0x00,0x00,0x00,0x03,0x00,0x00,0x00};

BYTE Rec[262] = {0};
DWORD dwLen = 262;
```



```
    printf("Select AID \n");
    ShowData(SelectAID, 13);

    //transmit C=TL APDU command
    rc = ISO14443_SendAPDU(phCtx, (LPCBYTE)SelectAID, 13, Rec, &dwLen);
    if (rc) {
        ShowError(phCtx, rc);
        return 0;
    }

    printf("get response data successfully\n");
    ShowData(Rec, dwLen);
```


2.錯誤代碼(Error Code)

2.1 WinNfc Error Code

說明：這是 WinNfc.dll 應用程式層所定義的錯誤代碼。
版本：適用 1.3/1.4 版本以上。

NFC_S_SUCCESS	**0**	函式呼叫正確
NFC_F_INTERNAL_ERROR	**0x80100001L**	內部錯誤
NFC_E_INVALID_HANDLE	**0x80100003L**	傳入的 Handle 值錯誤
NFC_E_INVALID_PARAMETER	**0x80100004L**	傳入參數錯誤
NFC_E_NO_MEMORY	**0x80100006L**	記憶體不足
NFC_E_UNKNOWN_READER	**0x80100009L**	無法辨識的讀卡機
NFC_E_TIMEOUT	**0x8010000AL**	函式逾時
NFC_E_SHARING_VIOLATION	**0x8010000BL**	讀卡機被占用
NFC_E_NO_SMARTCARD	**0x8010000CL**	找不到卡片
NFC_E_UNKNOWN_CARD	**0x8010000DL**	無法辨識的卡片
NFC_E_INVALID_VALUE	**0x80100011L**	傳入參數值錯誤
NFC_F_UNKNOWN_ERROR	**0x80100014L**	未知型錯誤
NFC_E_READER_UNAVAILABLE	**0x80100017L**	目前無法存取讀卡機
NFC_E_READER_UNSUPPORTED	**0x8010001AL**	不支援的讀卡機
NFC_E_CARD_UNSUPPORTED	**0x8010001CL**	不支援的卡片
NFC_E_NO_READERS_AVAILABLE	**0x8010002EL**	找不到讀卡機

注意：若出現讀卡機被占用的錯誤訊息，請檢查你是否有安裝 NfcCode 應用程式，此程式會常駐在系統，你可以從電腦桌面右下角圖示找到該應用程式並將它關閉。建議您移除該程式以方便您進行系統開發。

2.2 nfc Error Code

說明：這是 nfc.dll 底層所定義的錯誤代碼。

版本：適用 1.4 版本以上。

/* Chip-level errors */		
ETIMEOUT	**0x80000001L**	Timeout
ECRC	**0x80000002L**	CRC Error
EPARITY	**0x80000003L**	Parity Error
EBITCOUNT	**0x80000004L**	Erroneous Bit Count
EFRAMING	**0x80000005L**	Framing Error
EBITCOLL	**0x80000006L**	Bit-collision
ESMALLBUF	**0x80000007L**	Communication Buffer Too Small
EBUFOVF	**0x80000009L**	Buffer Overflow
ERFTIMEOUT	**0x8000000aL**	RF Timeout
ERFPROTO	**0x8000000bL**	RF Protocol Error
EOVHEAT	**0x8000000dL**	Chip Overheating
EINBUFOVF	**0x8000000eL**	Internal Buffer overflow
EINVPARAM	**0x80000010L**	Invalid Paramete
EDEPUNKCMD	**0x80000012L**	Unknown DEP Command
EINVRXFRAM	**0x80000013L**	Invalid Received Frame
/* MIFARE ISO14443 communication error*/		
EMFAUTH	**0x80000014L**	Mifare Authentication Error
ENSECNOTSUPP	**0x80000018L**	NFC Secure not supported
EBCC	**0x80000023L**	Wrong UID Check Byte (BCC)
EDEPINVSTATE	**0x80000025L**	Invalid DEP State
EOPNOTALL	**0x80000026L**	Operation Not Allowed
ECMD	**0x80000027L**	Command Not Acceptable
ETGREL	**0x80000029L**	Target Released
ECID	**0x8000002aL**	Card ID Mismatch
ECDISCARDED	**0x8000002bL**	Card Discarded
ENFCID3	**0x8000002cL**	NFCID3 Mismatch
EOVCURRENT	**0x8000002dL**	Over Current
ENAD	**0x8000002eL**	NAD Missing in DEP Frame
/* Software level errors */		

ETGUIDNOTSUP	**0x80000100L**	Target UID not supported
DENOTSUP	**0x80000400L**	Operation not supported
/* Common device-level errors */		
DEIO	**0x80001000L**	Input/output error
DEINVAL	**0x80002000L**	Invalid argument
DETIMEOUT	**0x80003000L**	Operation timed-out
DEABORT	**0x80004000L**	Operation aborted

2.3 IT-100MU(R) WinScard Error Code

說明：IT-100MU(R)是 CCID 相容的讀卡機，底層使用 WinScard 開發，這是 WinNfc.dll 應用程式層所定義的錯誤代碼。
版本：適用 1.6 版本以上。

Value	Description
ERROR_BROKEN_PIPE **0x00000109**	The client attempted a smart card operation in a remote session, such as a client session running on a terminal server, and the operating system in use does not support smart card redirection.
SCARD_E_BAD_SEEK **0x80100029**	An error occurred in setting the smart card file object pointer.
SCARD_E_CANCELLED **0x80100002**	The action was canceled by an **SCardCancel** request.
SCARD_E_CANT_DISPOSE **0x8010000E**	The system could not dispose of the media in the requested manner.
SCARD_E_CARD_UNSUPPORTED **0x8010001C**	The smart card does not meet minimal requirements for support.
SCARD_E_CERTIFICATE_UNAVAILABLE **0x8010002D**	The requested certificate could not be obtained.
SCARD_E_COMM_DATA_LOST	A communications error with the smart card has been

0x8010002F	detected.
SCARD_E_DIR_NOT_FOUND **0x80100023**	The specified directory does not exist in the smart card.
SCARD_E_DUPLICATE_READER **0x8010001B**	The *reader* driver did not produce a unique reader name.
SCARD_E_FILE_NOT_FOUND **0x80100024**	The specified file does not exist in the smart card.
SCARD_E_ICC_CREATEORDER **0x80100021**	The requested order of object creation is not supported.
SCARD_E_ICC_INSTALLATION **0x80100020**	No primary provider can be found for the smart card.
SCARD_E_INSUFFICIENT_BUFFER **0x80100008**	The data buffer for returned data is too small for the returned data.
SCARD_E_INVALID_ATR **0x80100015**	An *ATR string* obtained from the registry is not a valid ATR string.
SCARD_E_INVALID_CHV **0x8010002A**	The supplied PIN is incorrect.
SCARD_E_INVALID_HANDLE **0x80100003**	The supplied handle was not valid.
SCARD_E_INVALID_PARAMETER **0x80100004**	One or more of the supplied parameters could not be properly interpreted.
SCARD_E_INVALID_TARGET **0x80100005**	Registry startup information is missing or not valid.
SCARD_E_INVALID_VALUE **0x80100011**	One or more of the supplied parameter values could not be properly interpreted.

SCARD_E_NO_ACCESS **0x80100027**	Access is denied to the file.
SCARD_E_NO_DIR **0x80100025**	The supplied path does not represent a smart card directory.
SCARD_E_NO_FILE **0x80100026**	The supplied path does not represent a smart card file.
SCARD_E_NO_KEY_CONTAINER **0x80100030**	The requested key container does not exist on the smart card.
SCARD_E_NO_MEMORY **0x80100006**	Not enough memory available to complete this command.
SCARD_E_NO_PIN_CACHE **0x80100033**	The smart card PIN cannot be cached. **Windows Server 2008, Windows Vista, Windows Server 2003, and Windows XP:** This error code is not available.
SCARD_E_NO_READERS_AVAILABLE **0x8010002E**	No smart card reader is available.
SCARD_E_NO_SERVICE **0x8010001D**	The smart card *resource manager* is not running.
SCARD_E_NO_SMARTCARD **0x8010000C**	The operation requires a smart card, but no smart card is currently in the device.
SCARD_E_NO_SUCH_CERTIFICATE **0x8010002C**	The requested certificate does not exist.
SCARD_E_NOT_READY **0x80100010**	The reader or card is not ready to accept commands.
SCARD_E_NOT_TRANSACTED **0x80100016**	An attempt was made to end a nonexistent transaction.

SCARD_E_PCI_TOO_SMALL **0x80100019**	The PCI receive buffer was too small.
SCARD_E_PIN_CACHE_EXPIRED **0x80100032**	The smart card PIN cache has expired. **Windows Server 2008, Windows Vista, Windows Server 2003, and Windows XP:** This error code is not available.
SCARD_E_PROTO_MISMATCH **0x8010000F**	The requested protocols are incompatible with the protocol currently in use with the card.
SCARD_E_READ_ONLY_CARD **0x80100034**	The smart card is read-only and cannot be written to. **Windows Server 2008, Windows Vista, Windows Server 2003, and Windows XP:** This error code is not available.
SCARD_E_READER_UNAVAILABLE **0x80100017**	The specified reader is not currently available for use.
SCARD_E_READER_UNSUPPORTED **0x8010001A**	The reader driver does not meet minimal requirements for support.
SCARD_E_SERVER_TOO_BUSY **0x80100031**	The smart card resource manager is too busy to complete this operation.
SCARD_E_SERVICE_STOPPED **0x8010001E**	The smart card resource manager has shut down.
SCARD_E_SHARING_VIOLATION **0x8010000B**	The smart card cannot be accessed because of other outstanding connections.
SCARD_E_SYSTEM_CANCELLED **0x80100012**	The action was canceled by the system, presumably to log off or shut down.
SCARD_E_TIMEOUT **0x8010000A**	The user-specified time-out value has expired.
SCARD_E_UNEXPECTED	An unexpected card error has occurred.

0x8010001F	
SCARD_E_UNKNOWN_CARD **0x8010000D**	The specified smart card name is not recognized.
SCARD_E_UNKNOWN_READER **0x80100009**	The specified reader name is not recognized.
SCARD_E_UNKNOWN_RES_MNG **0x8010002B**	An unrecognized error code was returned.
SCARD_E_UNSUPPORTED_FEATURE **0x80100022**	This smart card does not support the requested feature.
SCARD_E_WRITE_TOO_MANY **0x80100028**	An attempt was made to write more data than would fit in the target object.
SCARD_F_COMM_ERROR **0x80100013**	An internal communications error has been detected.
SCARD_F_INTERNAL_ERROR **0x80100001**	An internal consistency check failed.
SCARD_F_UNKNOWN_ERROR **0x80100014**	An internal error has been detected, but the source is unknown.
SCARD_F_WAITED_TOO_LONG **0x80100007**	An internal consistency timer has expired.
SCARD_P_SHUTDOWN **0x80100018**	The operation has been aborted to allow the server application to exit.
SCARD_S_SUCCESS	No error was encountered.
SCARD_W_CANCELLED_BY_USER **0x8010006E**	The action was canceled by the user.

SCARD_W_CACHE_ITEM_NOT_FOUND **0x80100070**	The requested item could not be found in the cache.
SCARD_W_CACHE_ITEM_STALE **0x80100071**	The requested cache item is too old and was deleted from the cache.
SCARD_W_CACHE_ITEM_TOO_BIG **0x80100072**	The new cache item exceeds the maximum per-item size defined for the cache.
SCARD_W_CARD_NOT_AUTHENTICATED **0x8010006F**	No PIN was presented to the smart card.
SCARD_W_CHV_BLOCKED **0x8010006C**	The card cannot be accessed because the maximum number of PIN entry attempts has been reached.
SCARD_W_EOF **0x8010006D**	The end of the smart card file has been reached.
SCARD_W_REMOVED_CARD **0x80100069**	The smart card has been removed, so further communication is not possible.
SCARD_W_RESET_CARD **0x80100068**	The smart card was reset.
SCARD_W_SECURITY_VIOLATION **0x8010006A**	Access was denied because of a security violation.
SCARD_W_UNPOWERED_CARD **0x80100067**	Power has been removed from the smart card, so that further communication is not possible.
SCARD_W_UNRESPONSIVE_CARD **0x80100066**	The smart card is not responding to a reset.
SCARD_W_UNSUPPORTED_CARD **0x80100065**	The reader cannot communicate with the card, due to ATR string configuration conflicts.
SCARD_W_WRONG_CHV **0x8010006B**	The card cannot be accessed because the wrong PIN was presented.

2.4 IT-100MU(R) 特殊 Error Code

說明：這部分存取 NFC 卡片時會碰到的 Error。
版本：適用 1.6 版本以上。

Common Error Codes

0x6282	End of data reached before Le bytes (Le is greater than data length)
0x6300	No information is given
0x6700	Wrong length
0x6800	Class byte is not correct

Get Data

0x6A81	Function not supported
0x6B00	Wrong parameter P1-P2
0x6C??	Wrong length (wrong number Le; '??' encodes the exact number) if Le is less than the available UID length)

Load Keys

0x6982	Card key not supported
0x6883	Reader key not supported
0x6984	Plain transmission not supported
0x6985	Secured transmission not supported
0x6986	Volatile memory is not available
0x6987	Non volatile memory is not available
0x6988	Key number not valid
0x6989	Key length is not correct

Authenticate

0x6300	No information is given
0x6581	Memory failure, addressed by P1-P2 is does not exist
0x6982	Security status not satisfied
0x6983	Authentication cannot be done
0x6984	Reference key not useable
0x6986	Key type not known
0x6988	Key number not valid

Read/Update Binary

0x6281	Part of returned data may be corrupted.
0x6282	End of file reached before reading expected number of bytes.
0x6581	Memory failure (unsuccessful writing).
0x6981	Command incompatible.
0x6982	Security status not satisfied.
0x6986	Command not allowed.
0x6A81	Function not supported.
0x6A82	File not found / Addressed block or byte does not exist.
0x6C??	Wrong length (wrong number Le; '??' is the exact number).

3.API 技術詢問窗口

若您有採購合法的 SDK 軟體開發工具，有任何使用本 API 的問題，均可來信訊問，詢問窗口為 roman@ittec.com.tw

4.如何更新 API 版本

最新版本：http://www.ittec.com.tw/download/nfcsdk/setup.exe
版本說明：http://www.ittec.com.tw/download/nfcsdk/release.txt

