macally™

PCSC NFC Reader User Manual

Ver. 1.2

09/30/2008

RFCyber Corporation
http://www.rfcyber.com

Table of Content

Macally Product Information

Registration

Please register this product at www.macally.com/register.html.

Technical Support

Please call 1(909)230-6778 (M-F 8:30AM - 5:30PM, Pacific Standard Time, USA) or E-mail us at techsupport@macally.com

Warranty

Macally peripherals warrants that this product will be free from defects in title, materials and manufacturing workmanship for one year from the date of purchase. If the product is found to be defective then, as your sole remedy and as the manufacturer's only obligation, Macally will repair or replace the product. This warranty shall not apply to products that have been subject to abuse, misuse, abnormal electrical or environmental conditions, or any condition other than what can be considered as normal use.

Limitation of Liability

The liability of Macally Peripherals arising from this warranty and sale shall be limited to a refund of the purchase price. In no event shall Macally Peripherals be liable for costs of procurement of substitute products or services, or for any lost profit, or for any consequential, incidental, direct or indirect damages, however caused and on any theory of liability, arising from this warranty and sale. These limitations shall apply not with standing any failure of essential purpose of any limited remedy.

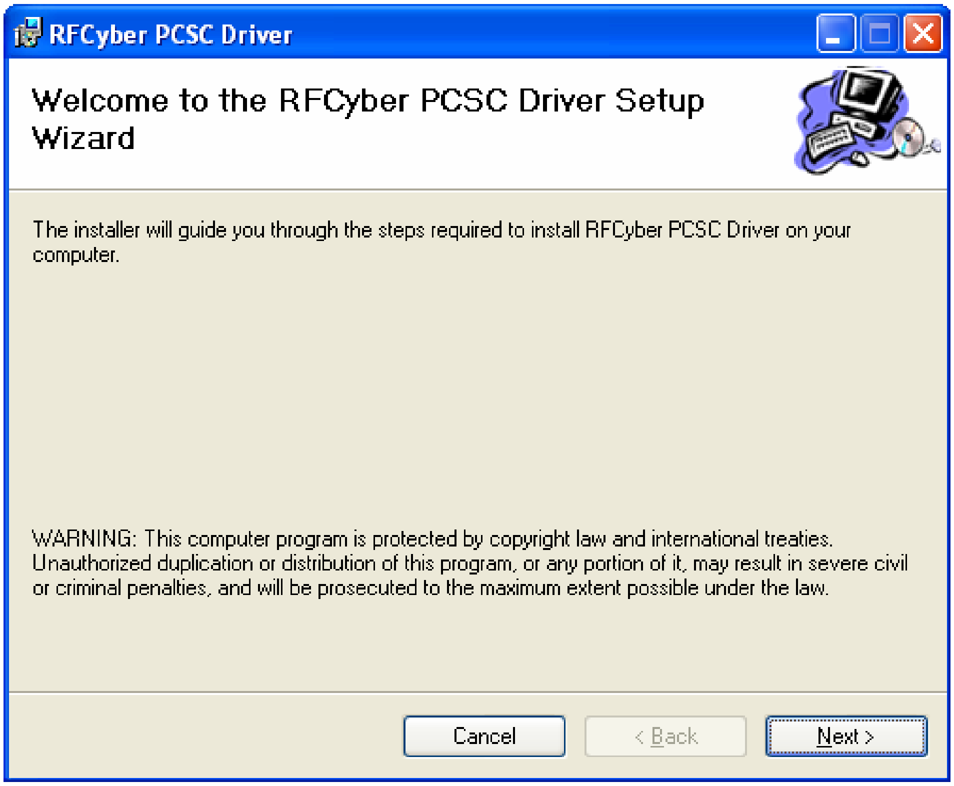
1 Introduction

This document is a guide for software developers who want to integrate contactless card (Mifare, Felica, 14443B, and smartcard) with PCSC eNetMouse/eNetPad reader. In order to support the PCSC feature, the user has to install the driver that described in chapter 2. The chapter 3 describes the PSCS 2.01 interface to communicate 14443-4 and Mifare tag. The chapter 4 will describes the other card (Felica, 14443-3B) support. The chapter 5 describes the proprietary RFCyber Mifare API. The chapter 6 describes the proprietary RFCyber NFC API.

2 Driver Installation

The driver is mandatory for all systems except the RS232 eNetPad user by using RFCyber NFC API (described in chapter 7)

The driver setup file (pcsc_driver_setup.msi) will copy the driver setup file to your hard driver and also configure a default polling card type. Please follow the setup wizard procedures that showing as below:

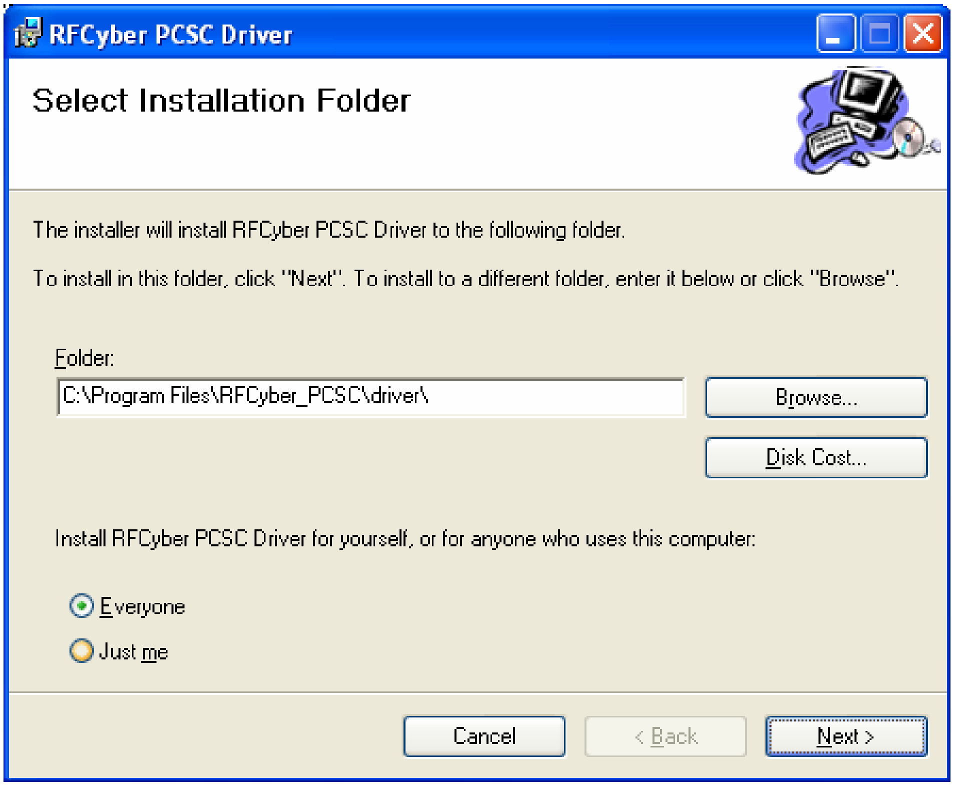


RFCyber PCSC Driver

Welcome to the RFCyber PCSC Driver Setup Wizard

The installer will guide you through the steps required to install RFCyber PCSC Driver on your computer.

WARNING: This computer program is protected by copyright law and international treaties. Unauthorized duplication or distribution of this program, or any portion of it, may result in severe civil or criminal penalties, and will be prosecuted to the maximum extent possible under the law.



Select Installation Folder

The installer will install RFCyber PCSC Driver to the following folder.

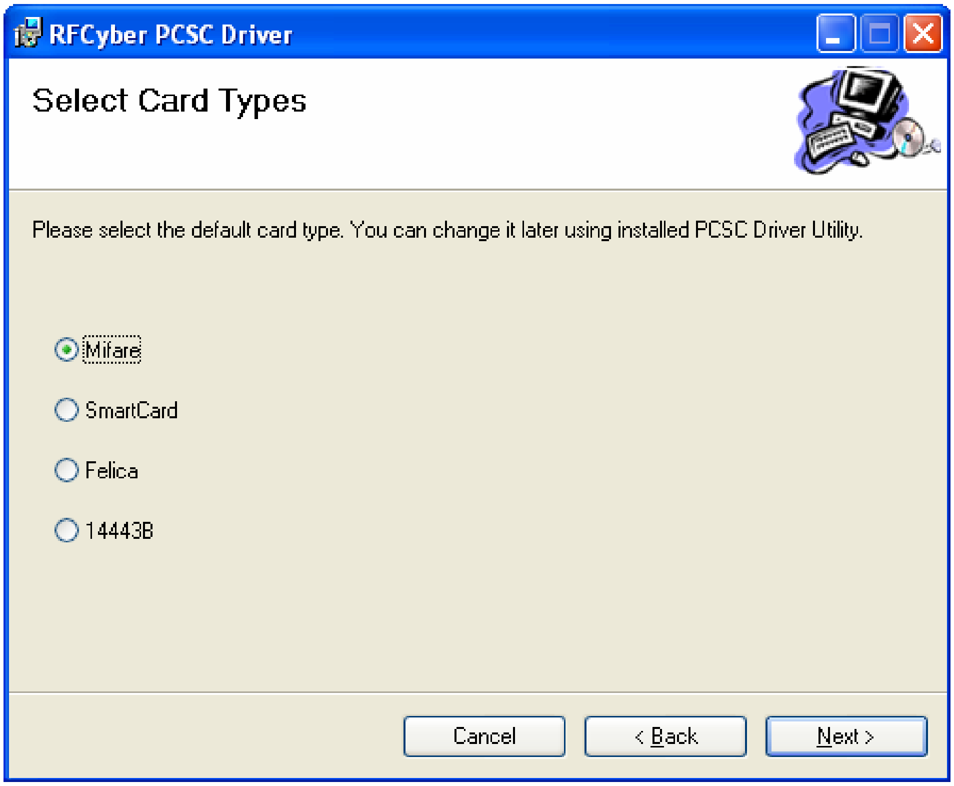
To install in this folder, click "Next". To install to a different folder, enter it below or click "Browse".

Folder: C:\Program Files\RFCyber_PCSC\driver\

Install RFCyber PCSC Driver for yourself, or for anyone who uses this computer:

● Everyone

○ Just me



Select Card Types

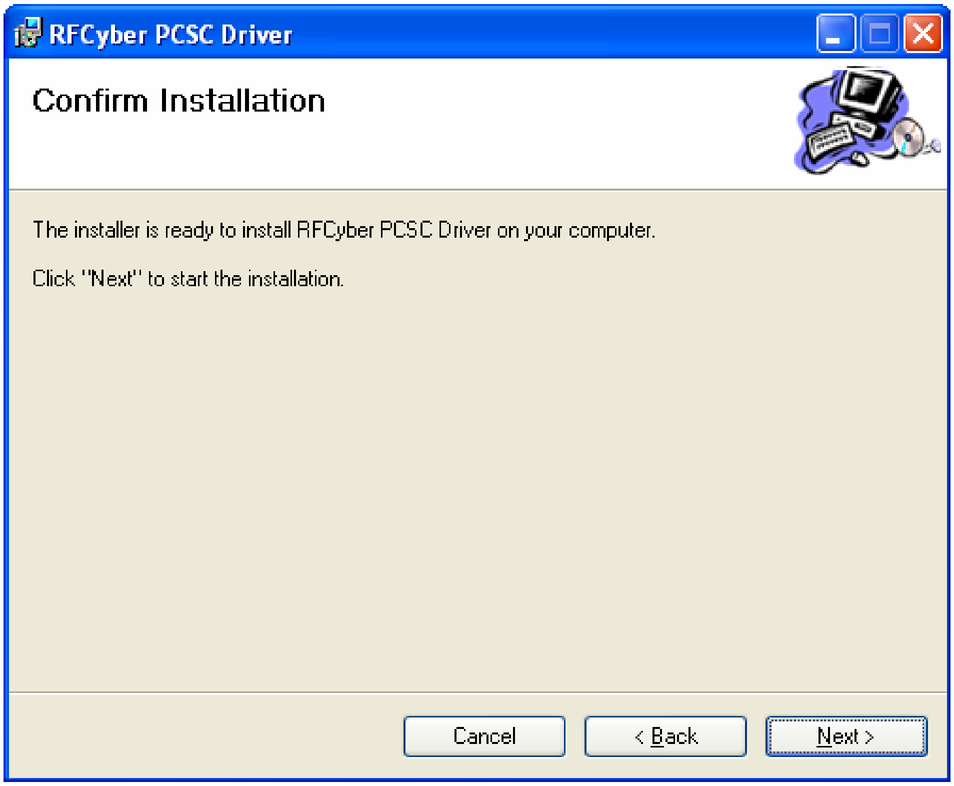
Please select the default card type. You can change it later using installed PCSC Driver Utility.

● Mifare

○ SmartCard

○ Felica

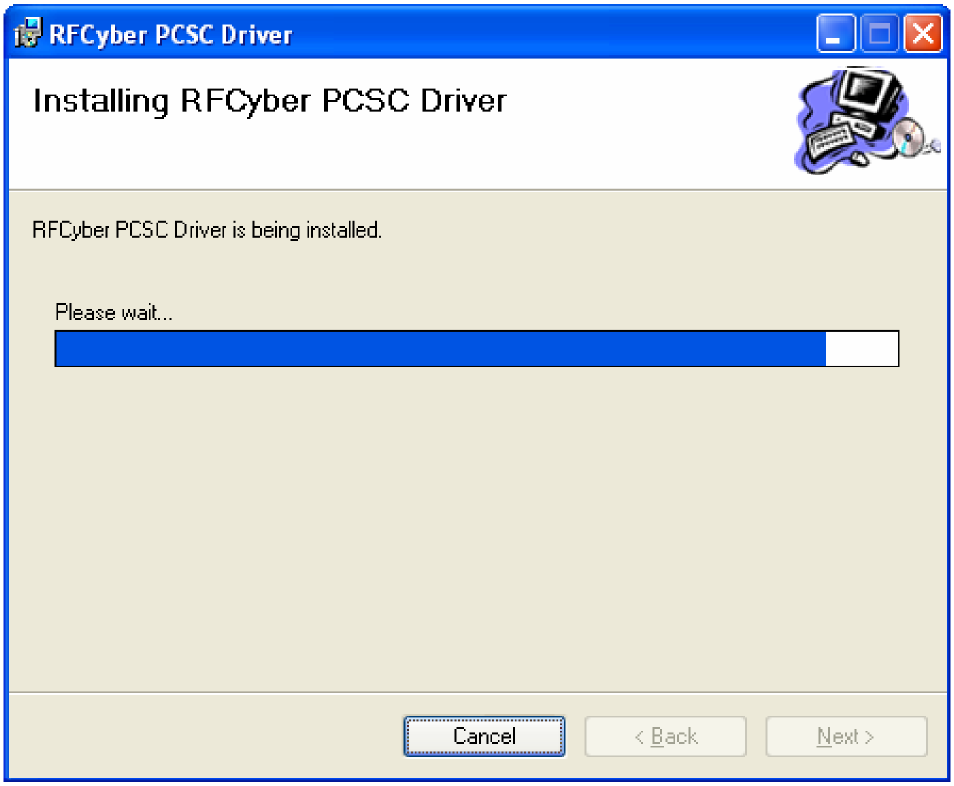
○ 14443B



Confirm Installation

The installer is ready to install RFCyber PCSC Driver on your computer.

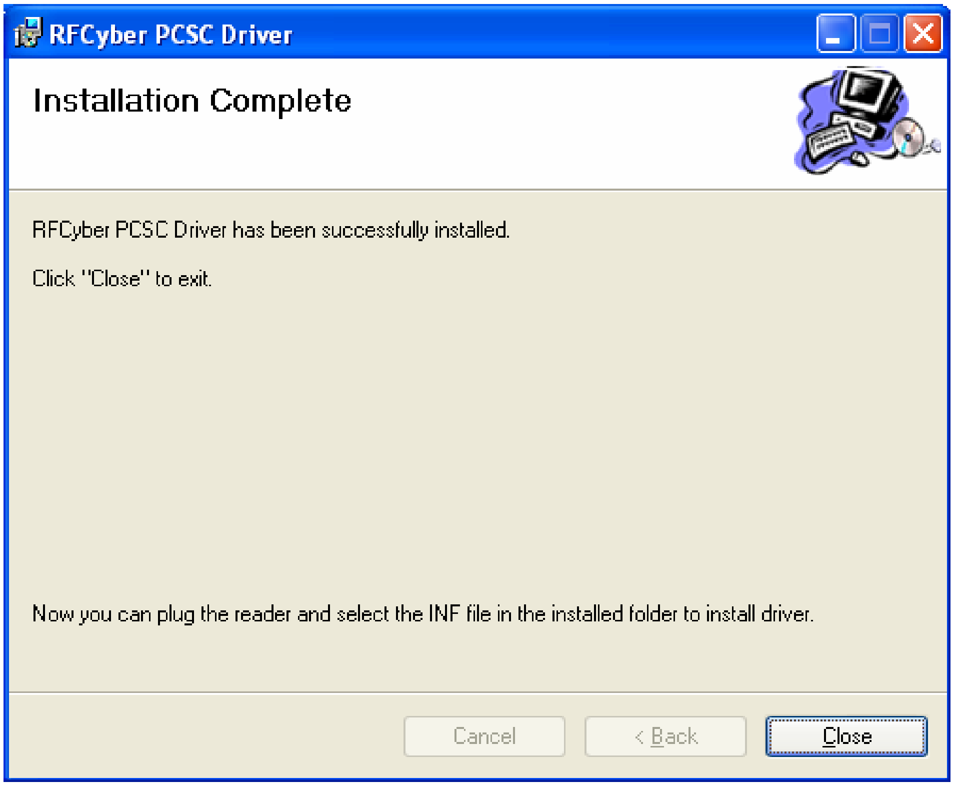
Click "Next" to start the installation.



Installing RFCyber PCSC Driver

RFCyber PCSC Driver is being installed.

Please wait...



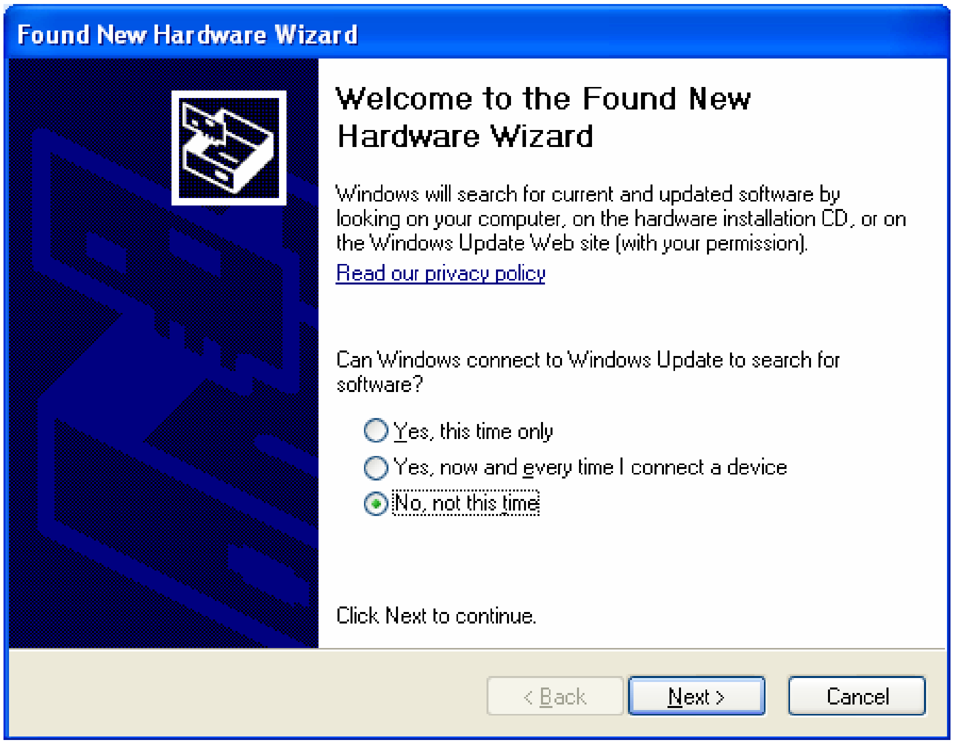
RFCyber PCSC Driver has been successfully installed.

Click "Close" to exit.

Now you can plug the reader and select the INF file in the installed folder to install driver.

After you push the Close button showing on top dialog, the plug and play driver should be installed. If not, the “Found New Hardware Wizard” dialog will pop up and follow the following procedure to continue the driver setup. If no any dialog pop up and the reader’s LED are off, please plug in your reader to the USB port, the “Found New Hardware Wizard” will guide you the installation procedure. If reader is already plug in to the USB port, please unplug and plug it again.

The following steps describe how to install the driver:
1. Connect the reader to a USB port of your computer.
2. The “Found New Hardware Wizard” will popup and chose “No, not this time” showing as below and push the “Next” button.



Found New Hardware Wizard

Welcome to the Found New Hardware Wizard

Windows will search for current and updated software by looking on your computer, on the hardware installation CD, or on the Windows Update Web site (with your permission).

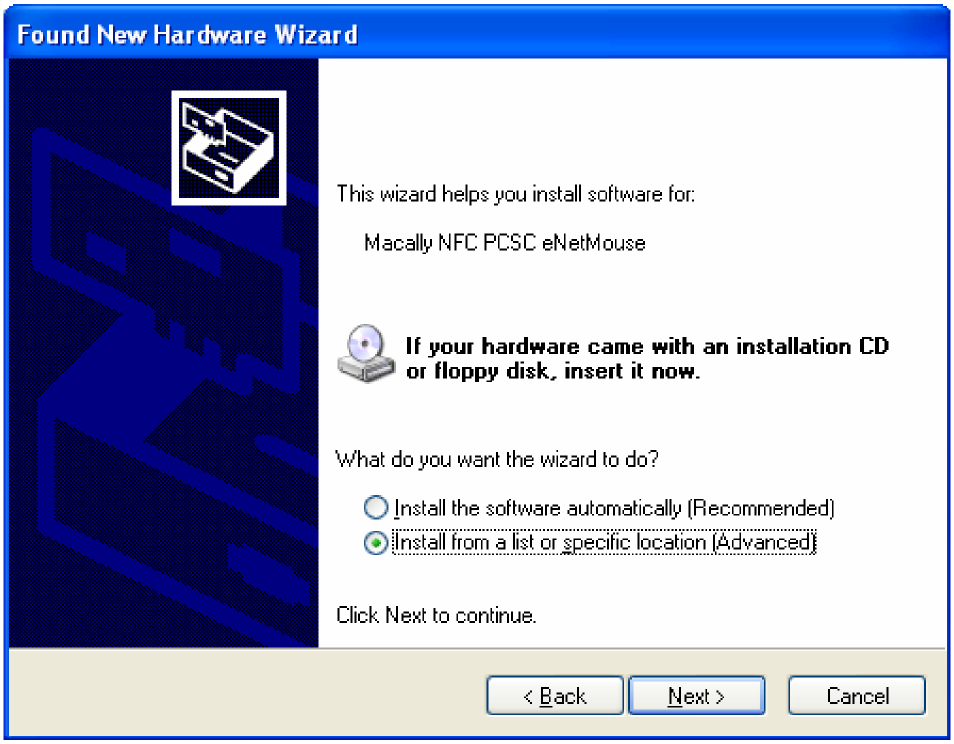
Read our privacy policy

Can Windows connect to Windows Update to search for software?

○ Yes, this time only
○ Yes, now and every time I connect a device
● No, not this time

Click Next to continue.

3. Select the "Install from a list or specific location (Advanced)" and push the "Next" button.



Found New Hardware Wizard

This wizard helps you install software for:

Macally NFC PCSC eNetMouse

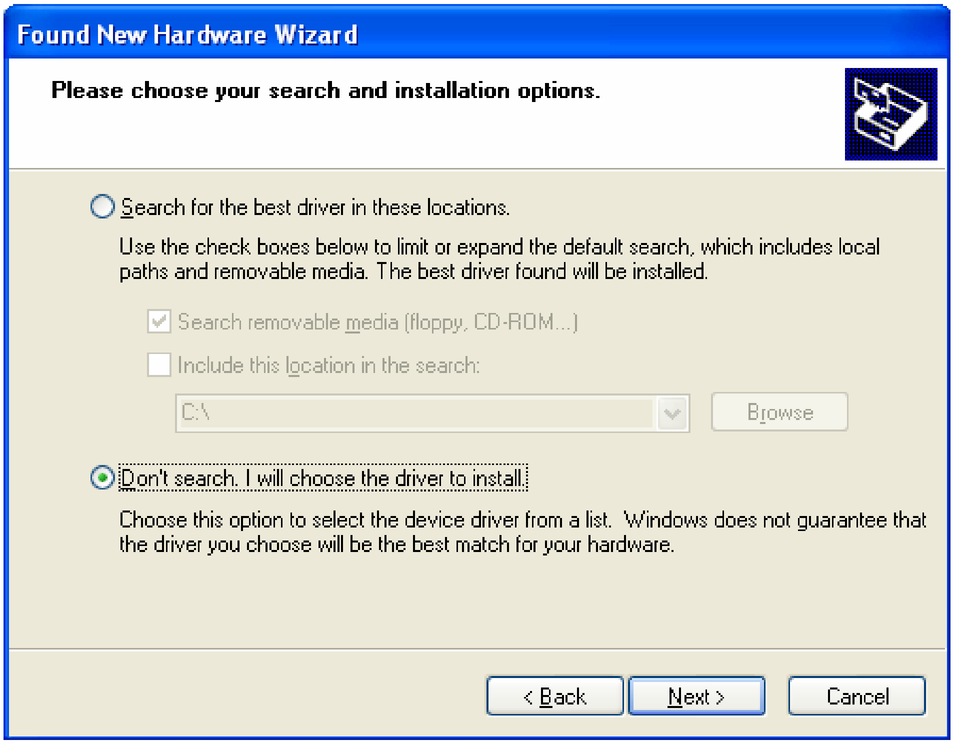
If your hardware came with an installation CD or floppy disk, insert it now.

What do you want the wizard to do?

○ Install the software automatically (Recommended)
● Install from a list or specific location (Advanced)

Click Next to continue.

4. Select "Don't search. I will choose the driver to install" and push the "Next" button.



Found New Hardware Wizard

Please choose your search and installation options.

○ Search for the best driver in these locations.

Use the check boxes below to limit or expand the default search, which includes local paths and removable media. The best driver found will be installed.

☑ Search removable media (floppy, CD-ROM...)

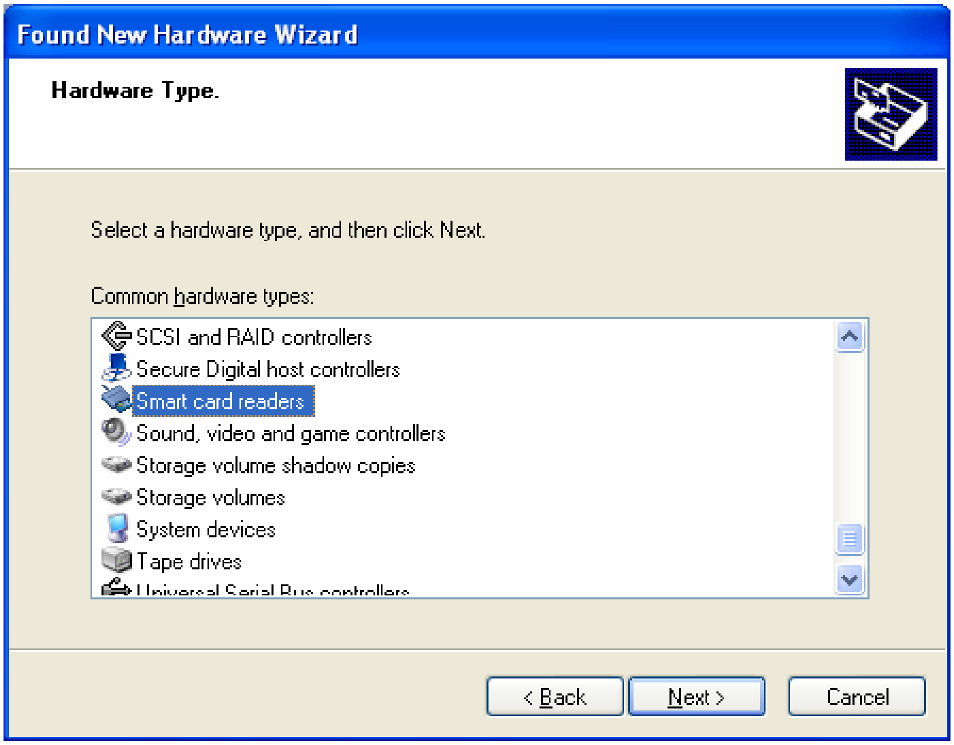
□ Include this location in the search:

C:\

● Don't search. I will choose the driver to install.

Choose this option to select the device driver from a list. Windows does not guarantee that the driver you choose will be the best match for your hardware.

5. The following dialog popup and push the “Next” button.



Found New Hardware Wizard

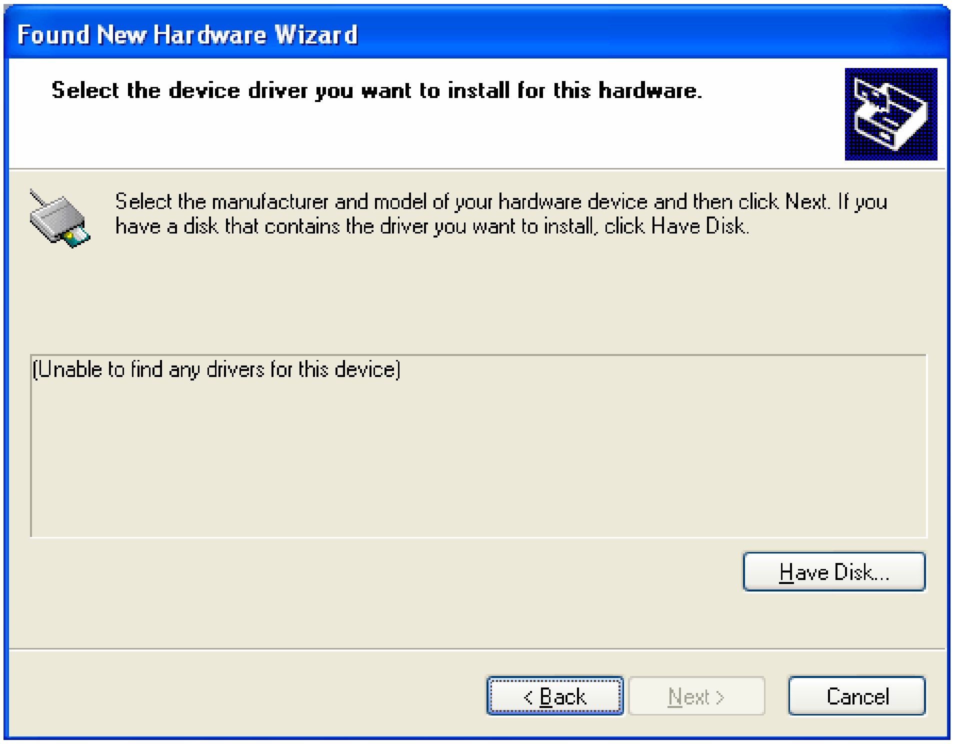
Hardware Type.

Select a hardware type, and then click Next.

Common hardware types:

- SCSI and RAID controllers
- Secure Digital host controllers
- Smart card readers
- Sound, video and game controllers
- Storage volume shadow copies
- Storage volumes
- System devices
- Tape drives
- Universal Serial Bus controllers

One of the following driver selection dialogs will popup:

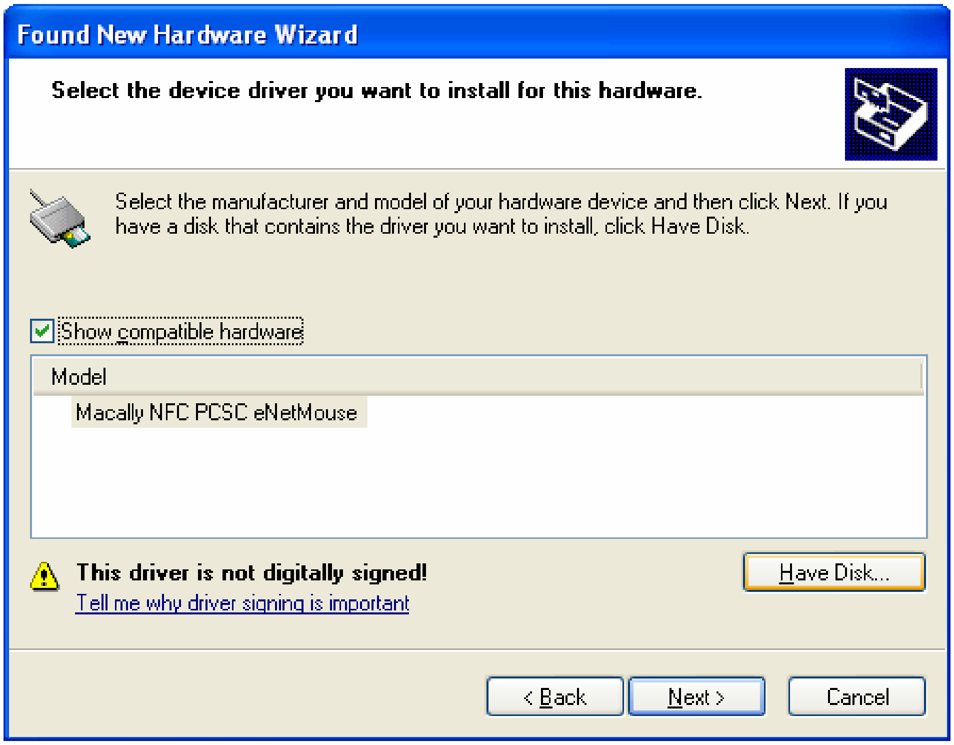


Found New Hardware Wizard

Select the device driver you want to install for this hardware.

Select the manufacturer and model of your hardware device and then click Next. If you have a disk that contains the driver you want to install, click Have Disk.

(Unable to find any drivers for this device)



Found New Hardware Wizard

Select the device driver you want to install for this hardware.

Select the manufacturer and model of your hardware device and then click Next. If you have a disk that contains the driver you want to install, click Have Disk.

☑ Show compatible hardware

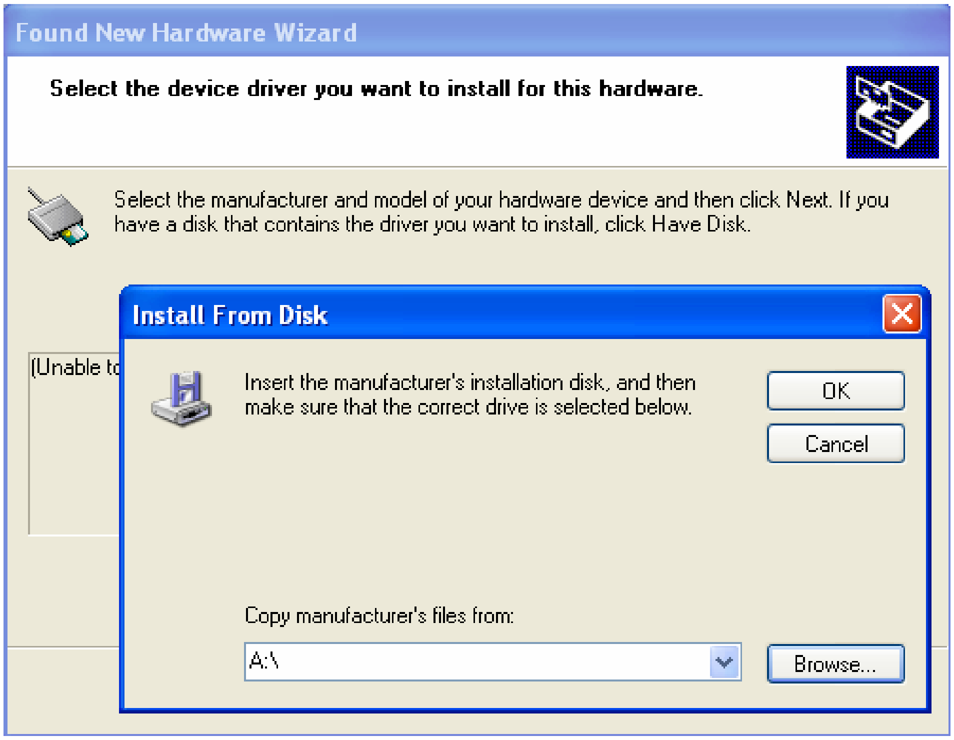
Model
Macally NFC PCSC eNetMouse

This driver is not digitally signed!

Tell me why driver signing is important

Select “Have Disk” and the “Install From Disk” dialog will popup as below.

6. Click the “Browse” button and select the driver folder. (C:\Program Files\RFCyber_PCSC\driver)


Found New Hardware Wizard

Select the device driver you want to install for this hardware.

Select the manufacturer and model of your hardware device and then click Next. If you have a disk that contains the driver you want to install, click Have Disk.

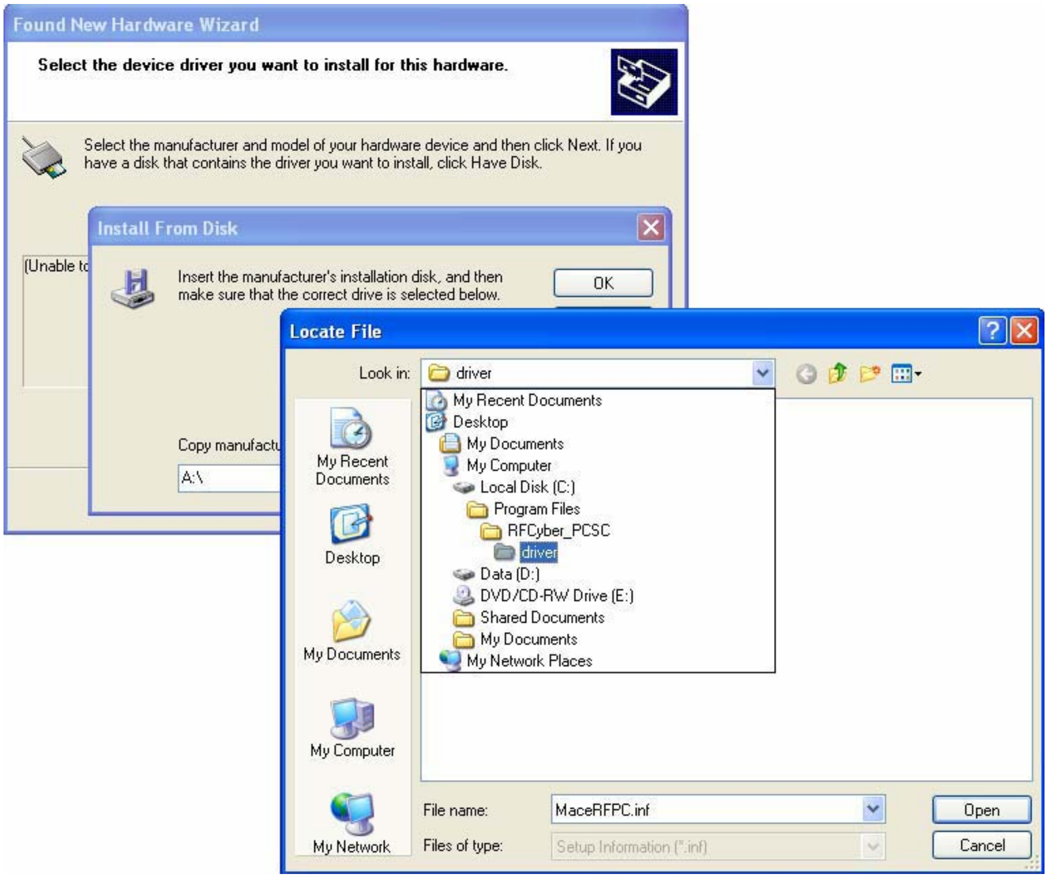
(Unable to

Install From Disk

Insert the manufacturer's installation disk, and then make sure that the correct drive is selected below.

Copy manufacturer's files from:

A:\


Found New Hardware Wizard

Select the device driver you want to install for this hardware.

Select the manufacturer and model of your hardware device and then click Next. If you have a disk that contains the driver you want to install, click Have Disk.

(Unable to

Install From Disk

Insert the manufacturer's installation disk, and then make sure that the correct drive is selected below.

Copy manufactu

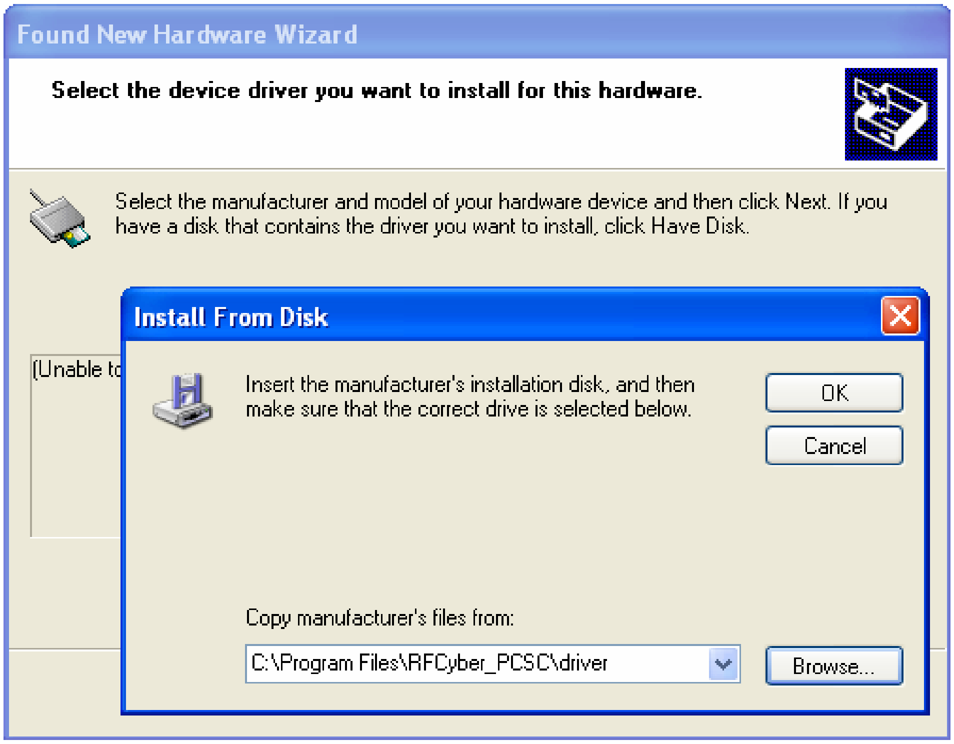
A:\

Locate File

Look in: driver

File name: MaceRFPC.inf

Files of type: Setup Information (*.inf)

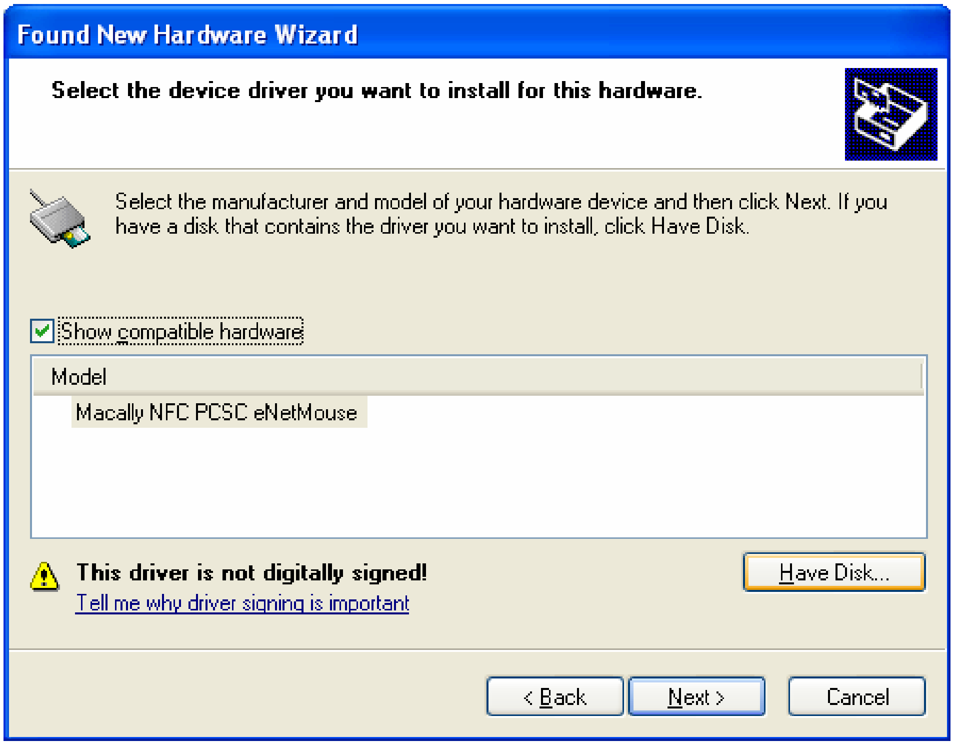


Select the device driver you want to install for this hardware.

Select the manufacturer and model of your hardware device and then click Next. If you have a disk that contains the driver you want to install, click Have Disk.

Insert the manufacturer's installation disk, and then make sure that the correct drive is selected below.

Copy manufacturer's files from: C:\Program Files\RFCyber_PCSC\driver



Select the device driver you want to install for this hardware.

Select the manufacturer and model of your hardware device and then click Next. If you have a disk that contains the driver you want to install, click Have Disk.

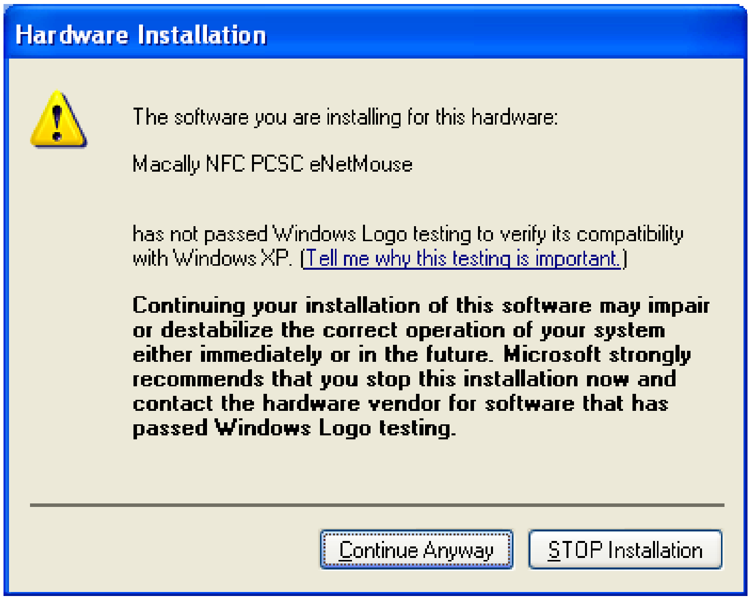
☑ Show compatible hardware

Model
Macally NFC PCSC eNetMouse

This driver is not digitally signed!

Tell me why driver signing is important

7. Push the Next and the following dialog will popup, Choose the "Continue Anyway" and it will starting the driver installation.



Hardware Installation

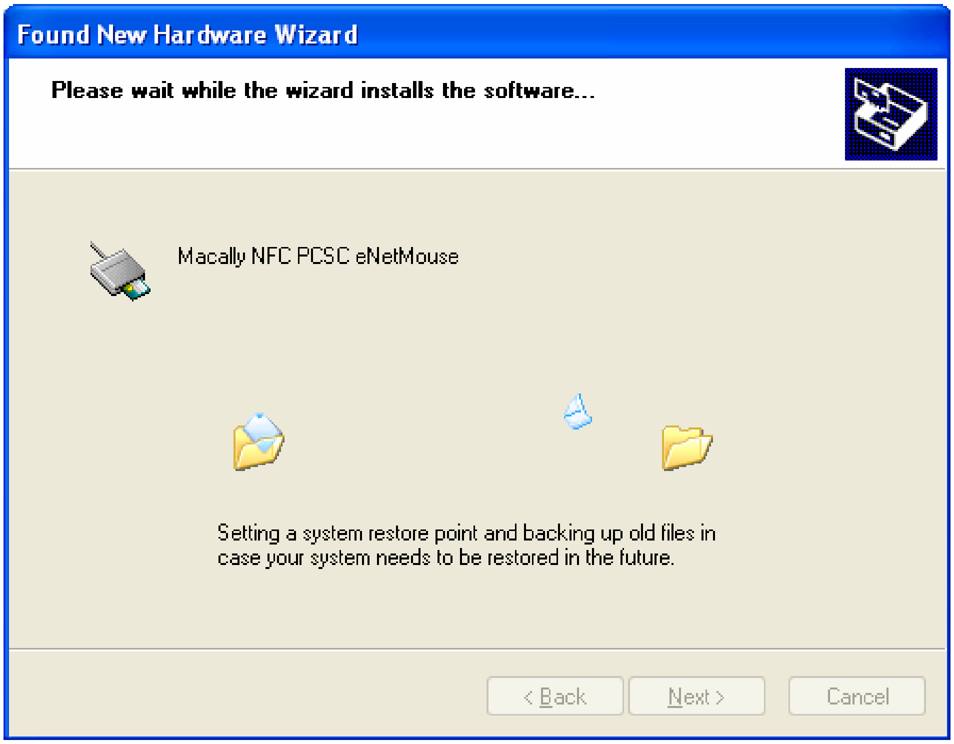
The software you are installing for this hardware:

Macally NFC PCSC eNetMouse

has not passed Windows Logo testing to verify its compatibility with Windows XP. (Tell me why this testing is important.)

Continuing your installation of this software may impair or destabilize the correct operation of your system either immediately or in the future. Microsoft strongly recommends that you stop this installation now and contact the hardware vendor for software that has passed Windows Logo testing.

Following dialogs are showing the driver installation status. In the last step just push the “Finish”button.

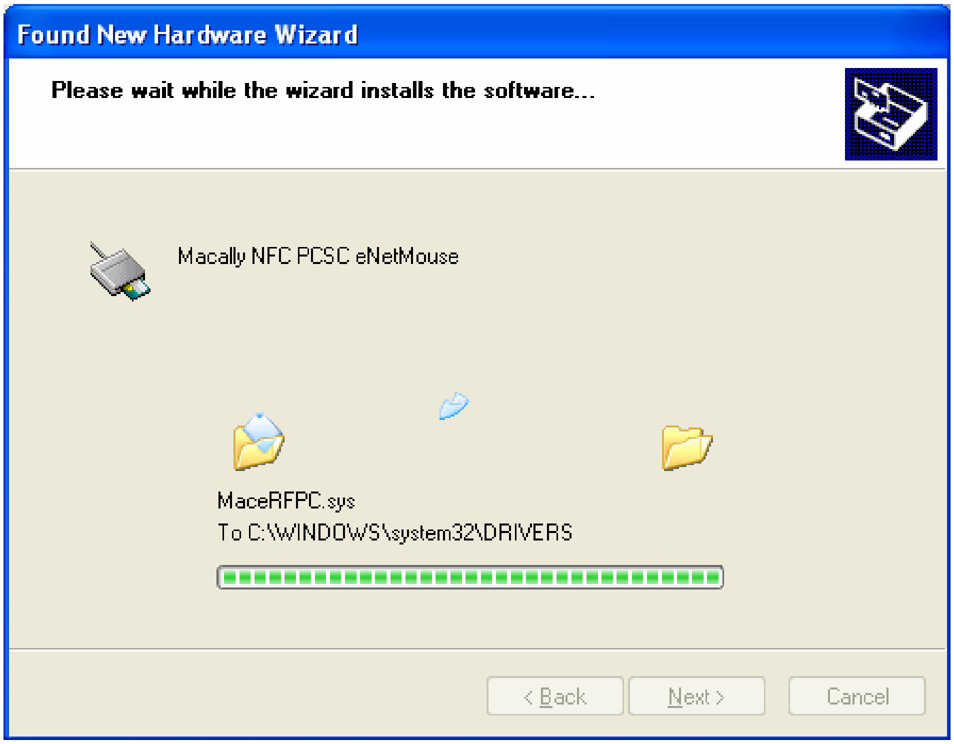


Found New Hardware Wizard

Please wait while the wizard installs the software...

Macally NFC PCSC eNetMouse

Setting a system restore point and backing up old files in case your system needs to be restored in the future.

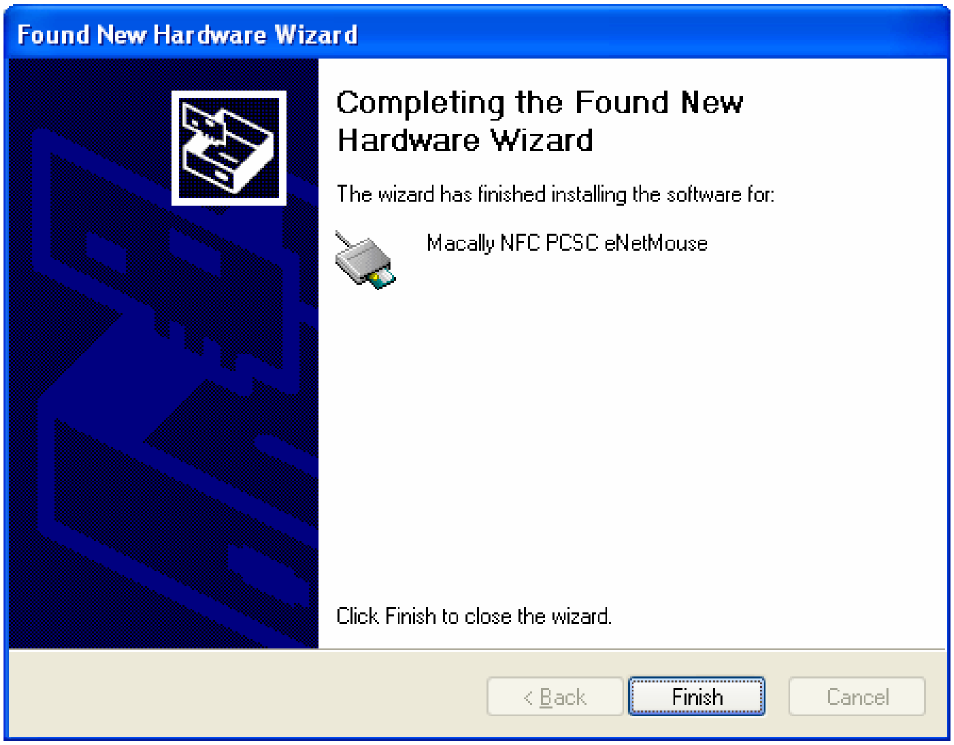


Found New Hardware Wizard

Please wait while the wizard installs the software...

Macally NFC PCSC eNetMouse

MaceRFPC.sys
To C:\WINDOWS\system32\DRIVERS



Found New Hardware Wizard

Completing the Found New Hardware Wizard

The wizard has finished installing the software for:

Macally NFC PCSC eNetMouse

Click Finish to close the wizard.

If the driver was successfully installed, the blue LED on the eNetMouse (or green LED on the eNetPad) will light up and the PCSC resource manager will show this reader name as “Macally NFC PCSC Reader #”.

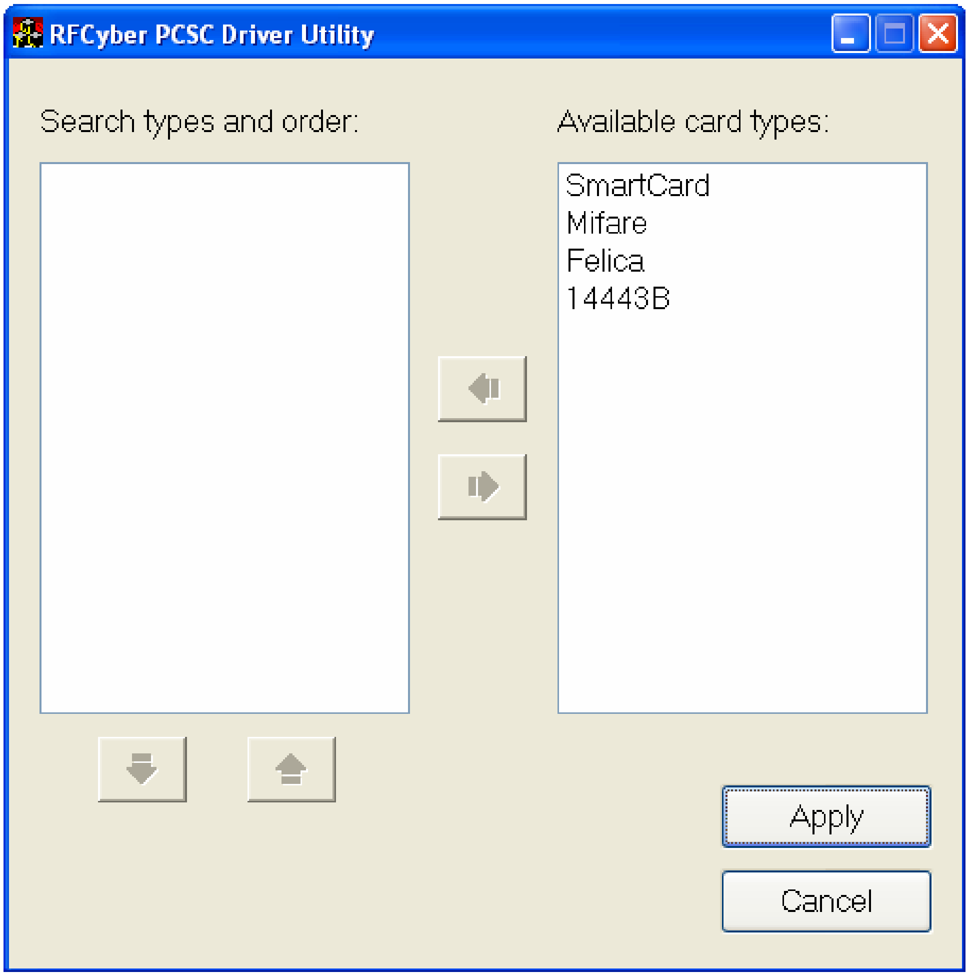
3 RFID Configuration

3.1 RFID Polling Configuration Tool

In order to support the contactless tags in the PC/SC environments, the reader needs to keep trying to do a polling to detect the RF field whether there is a contactless tag arrived. As soon as the reader polled a tag, it will create the communication handler for that tag.

The PCSC Driver Utility is the tool to configure the polling mechanism to let the reader to be able to decide what kind of card type will be needed and what is the respective searching order. User can choose the card type to put to the left hand side box with correct order. After push the “Apply” button, the polling order and card types will be changed. User need to unplug the reader and plug it back to the computer to get this change effect.

The utility can be found under C:\Program Files\RFCyber_PCSC\driver\PcscDrvUtil.exe



RFCyber PCSC Driver Utility

Search types and order:

Available card types:

SmartCard
Mifare
Felica
14443B

3.2 RFID Polling Configuration API

Application developer can use the following proprietary SCardControl API to configure the polling order.

Following is the PC/SC SCardControl API definition:

```
LONG SCardControl(
__in SCARDHANDLE hCard,
__in DWORD dwControlCode,
__in LPCVOID lpInBuffer,
__in DWORD nInBufferSize,
__out LPVOID lpOutBuffer,
__in DWORD nOutBufferSize,
__out LPDWORD lpBytesReturned
);
```

3.2.1 Query Card Info

Application can use this API to verify the card type and UID for the Mifare card. If the card type is not the one application expect, the next API could be used to change the polling card type.

dwControlCode:

SCARD_CTL_CODE(2300)

lpInBuffer and nInBufferSize:

N/A

lpOutBuffer, nOutBufferSize, and lpBytesReturned:

1 byte Card Type + 4 byte UID if it is a Mifare card

The CARD_TYPE will be one of following enum:

```
enum CARD_TYPE {
CARD_TYPE_UNKNOWN,
CARD_TYPE_SMARTCARD,
CARD_TYPE_MIFARE,
CARD_TYPE_FELICA,
CARD_TYPE_14443B
};
```

3.2.2 Setup the Polling Card Type

Application can use this API to setup the card type for the polling mechanism.

After this API call, the PCSC card handler will be disconnected and a new polling will be restarted with the specified polling card type(s). Application need to listen the status changes again (SCardGetStatusChange) to get the new card handler. The specified polling card type(s) in this function will be valid until the SCardDisconnect got called or application exit and the polling will be go back to the user setup polling mode after that.

dwControlCode:

SCARD_CTL_CODE(2530)

lpInBuffer and nInBufferSize:

A byte array. Each byte's value is a CARD_TYPE. It is polled from first byte

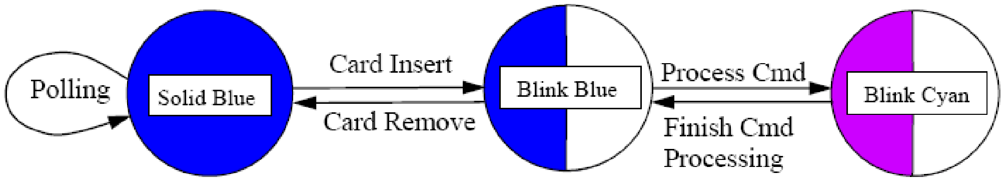
lpOutBuffer, nOutBufferSize, and lpBytesReturned:

N/A

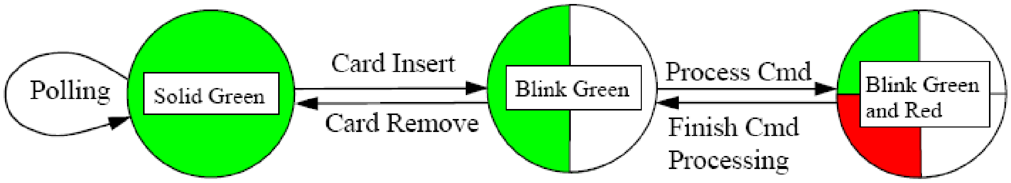
3.3 LED indication

There are two LED on both eNetPad and eNetMouse reader. One is the red LED, the other is the blue LED on eNetMouse and green LED on eNetPad. Following is the state diagram for the LED indication.

If only the red LED is on or blinking, means the error.
Following are the normal behavior LED indication:
1. eNetMouse



2. eNetPad



4 PC/SC 2.0 Smart Card Interface

To communicate with the ISO7816 contactless cards and the reader is go trough the PC/SC framework. The Microsoft Developer Network (MSDN) Library contains all the detail information. Also there is a documentation in the

MSDN Platform SDK: “Security” section for the SCard API.

The following steps are the guideline to create the PC/SC application. For other detail description, please refer to the MSDNLIB.

1. Establish Context

```
LONG SCardEstablishContext ( IN DWORD dsScope,
                             IN LPCVOID pvReserved1,
                             IN LPCVOID pvReserved2,
                             OUT LPSCARDCONTEXT phContext);
```

2. Get Status Change

```
LONG SCardGetStatusChange (IN SCARDCONTEXT hContext,
                           IN DWORD dwTimeout,
                           IN OUT LPSCARD_READERSTATE rgReaderStates,
                           IN DWORD cReaders);
```

3. List Readers

```
LONG SCardListReaders (     IN SCARDCONTEXT hContext,
                            IN LPCTSTR,mszGroups,
                            OUT LPTSTR mszReaders,
                            IN OUT LPDWORD pcchReaders);
```

4. Connect

```
LONG SCardConnect (         IN SCARDCONTEXT hContext,
                            IN LPCTSTR,szReader,
                            IN DWORD dwShareMode,
                            IN DWORD dwPreferredProtocols,
                            OUT LPSCARDHANDLE phCard,
                            OUT LPDWORD pdwActiveProtocol);
```

5. Exchange Data

```
LONG SCardTransmit (        IN SCARDHANDLE hCard,
                            IN LPCSCARD_IO_REQUEST pioSendPci,
                            IN LPCBYTE pbSendBuffer,
                            IN DWORD cbSendLength,
                            IN OUT LPSCARD_IO_REQUEST pioRecvPci,
                            OUT LPBYTE pbRecvBuffer,
                            IN OUT LPDWORD pcbRecvLength);
```

The 14443-4 Chaining mechanism (Chaining, Waiting time Extension, and Error handling) has been implemented in this API. Please refer to the ISO/IEC 14443-4 spec for the detail. (This chaining mechanism is only apply to the ISO 14443-4 Smart Card)

6. Disconnect

```
LONG SCardDisconnect (      IN SCARDHANDLE hCard,
                            IN DWORD dwDisposition);
```

7. Release

```
LONG SCardReleaseContext (  IN SCARDHANDLE hCard);
```

5 PC/SC 2.01 Mifare Card Interface

The PCSC eNetMouse/eNetPad supports Mifare 1K, Mifare 4K, and Mifare Ultra Light cards by using PC/SC 2.01 interface. Please refer to the PC/SC Workgroup Specifications 2.01 and MIFARE Data Sheets for the Mifare card commands. Following are the commands that defined in PCSC 2.01 and the command description are described in section 5.1 to section 5.5.

- getUID (PCSC 2.01)
- LoadKey (PCSC 2.01)
- Authenticate (PCSC 2.01)
- Read Binary (PCSC 2.01)
- Update Binary (PCSC 2.01)

There are four proprietary API to help the user easily to access the Mifare's value block data. The following commands are supported and the detail command will be described in the section 5.6 to section 5.9.

- Read Value (proprietary API)
- Update Value (proprietary API)
- Increment Value (proprietary API)
- Decrement Value (proprietary API)

Following are the Common Error Code

Command	**SW1**	**SW2**	**Meaning**
Error	***67***	***00***	Wrong length
	68	***00***	Class byte is not correct
	6A	***81***	Function not supported
	6B	***00***	Wrong parameter P1-P2
	60	***00***	Unexpected fail

5.1 Mifare Get UID API

Get UID Command APDU

Command	**Class**	**INS**	**P1**	**P2**	**Lc**	**Data In**	**Le**
Get UID	0xFF	0xCA	0x00	0x00	--	--	XX

Le: 0x00 means: Return full length of the UID. For ISO 14443A single 4 bytes, double 7 bytes, triple 10 bytes. (Mifare 1K and 4K is 4 bytes. Mifare UltraLight is 7 bytes).

For eNetMouse, the UltraLight will return 8 bytes, the 1st byte in the UID field could be ignored. This is a known bug from the reader chip.

Get UID Command Output

Data Output
UID + SW1 SW2

Get UID Error Codes

Command	SW1	SW2	Meaning
Warning	***62***	***82***	End of UID reached before Le bytes (Le is greater than UID Length) In this case the IFD subsystem has to insert zero-value padding bytes up until the length of the UID expected by the caller.
Error	***6C***	***XX***	Wrong Length (wrong number Le; 'XX' encodes the exact number) if Le is less than the available UID length)

5.2 Mifare Load Keys API

Load Keys Command APDU

Command	Class	INS	P1	P2	Lc	Data In	Le
Load Keys	0xFF	0x82	Key Structure	Key number	Key Length	Key	-

P1 (Key Structure): 0x00. The eNetPad and eNetMouse only support 0x00 for this release. For the detail Key Structure, please refer to the PCSC specification.
P2 (Key Number): Application can store 80 different keys by specify this key location.
Key Length: 0x06 for Mifare Key length.

Load Keys Command Output

Data Output
SW1 SW2

Load Keys Error Codes

Command	SW1	SW2	Meaning
Warning	***63***	***00***	No information is given
Error		***83***	Reader key not supported
		85	Secured transmission not supported
		87	Non volatile memory is not available
		88	Key number not valid
		89	Key length is not correct

5.3 Mifare Authenticate API

Authenticate Command APDU

Command	Class	INS	P1	P2	Lc	Data In	Le
Authenticate	0xFF	0x88	Address MSB	Address LSB	Key Type	Key Nr.	-

P1: 0x00: Mifare Block Number MSB, for mifare it is always 0x00
P2: Mifare Block Number LSB
P3: Key Type. For Mifare KEY_A (0x60) or KEY_B (0x61).
Data In (Key Nr.): The card key number, which will be used for this authentication. (One of the Key Nr. has been previously used by the Load Keys API)

Authenticate Command Output

Data Output
SW1 SW2

Authenticate Error Codes

Command	SW1	SW2	Meaning
Warning	***63***	***00***	No information is given
Error	***65***	***81***	Memory failure, addressed by P1-P2 does not exist
	69	***83***	Authentication cannot be done
		85	Key type not known
		88	Key number not valid

5.4 Mifare Read Binary API

Read Binary Command APDU

Command	Class	INS	P1	P2	Lc	Data In	Le
Read Binary	0xFF	0xB0	Address MSB	Address LSB	-	-	XX

P1: 0x00: Mifare Block Number MSB, for Mifare it is always 0x00
P2: Mifare Block Number LSB (or the Page Number for UltraLigt)
Le: Mifare 1K and 4K is 0x10 and Mifare UltraLight is 0x04
.

Read Binary Command Output

Data Output
Data + SW1 SW2

Read Binary Error Codes

Command	SW1	SW2	Meaning

Warning	***62***	***81***	Part of returned data may be corrupted
Error	***6A***	***82***	File not found/ Addressed block or byte dose not exit
	6C	***XX***	Wrong length (wrong number le; 'XX' is the exact number).

5.5 Mifare Update Binary API

Update Binary Command APDU

Command	Class	INS	P1	P2	Lc	Data In	Le
Update Binary	0xFF	0xD6	Address MSB	Address LSB	XX	Data	-

P1: 0x00: Mifare Block Number MSB, for Mifare it is always 0x00
P2: Mifare Block Number LSB (or the Page Number for UltraLigt)
Lc: The data length. For Mifare 1K and 4K is 0x10 and Mifare UltraLight is 0x04.

Update Binary Command Output

Data Output
SW1 SW2

Update Binary Error Codes

Command	SW1	SW2	Meaning
Warning	***65***	***81***	Memory failure (unsuccessful writing).
Error	***6A***	***82***	File not found/ Addressed block or byte dose not exit

5.6 Mifare Read Value API

This command reads the block has been formatted to a value block and return a four byte (integer) value in the little endian format (C convention in the PC), the least significant byte first (at the lowest address in the memory).

Read Value Command APDU

Command	Class	INS	P1	P2	Lc	Data In	Le
Read Value	0xFF	0xB2	Address MSB	Address LSB	--	--	--

The Le will be ignored in the API.

Read Value Command Output

Data Output
four byte value data + SW1 SW2

The output value data must be a four byte (integer) value in the little endian format (C convention in the PC), the least significant byte first (at the lowest address in the memory).

Read Value Error Codes

Command	SW1	SW2	Meaning
Error	***6A***	***82***	Invalid block address. MSB can not bigger than 0
	62	***81***	Currupted data. (The block may have an invalid value block format)

5.7 Mifare Update Value API

This command updates the value on the specified block that has been formatted to a value block. The input data must be a four byte (integer) value in the little endian format (C convention in the PC), the least significant byte first (at the lowest address in the memory).

Update Value Command APDU

Command	Class	INS	P1	P2	Lc	Data In	Le
Read Value	0xFF	0xD0	Address MSB	Address LSB	4	four bye value	--

The Le will be ignored in this API.

Update Value Command Output

Data Output
SW1 SW2

Update Value Error Codes

Command	SW1	SW2	Meaning
Error	***6A***	***82***	Invalid block address. MSB can not bigger than 0

5.8 Mifare Increment Value API

This command increases the value on the specified block that has been formatted to a value block. The input data must be a four byte (integer) value in the little endian format (C convention in the PC), the least significant byte first (at the lowest address in the memory).

Increment Value Command APDU

Command	Class	INS	P1	P2	Lc	Data In	Le
Increment Value	0xFF	0xD2	Address MSB	Address LSB	4	four bye value	--

The Le will be ignored in this API

Increment Value Command Output

Data Output
SW1 SW2

Increment Value Error Codes

Command	SW1	SW2	Meaning
Error	***6A***	***82***	Invalid block address. MSB can not bigger than 0

5.9 Mifare Decrement Value API

This command decrements the value on the specified block that has been formatted to a value block. The input data must be a four byte (integer) value in the little endian format (C convention in the PC), the least significant byte first (at the lowest address in the memory).

Decrement Value Command APDU

Command	Class	INS	P1	P2	Lc	Data In	Le
Decrement Value	0xFF	0xD4	Address MSB	Address LSB	4	four bye value	--

The Le will be ignored in this API

Decrement Value Command Output

Data Output
SW1 SW2

Decrement Value Error Codes

Command	SW1	SW2	Meaning
Error	***6A***	***82***	Invalid block address. MSB can not bigger than 0

6 Other Interface

The eNetMouse/eNetPad supports other contactless card, like DESFire, Felica, and 14443B. The card should be detected by the reader as long as the polling mechanism has been setting up correctly. The section 6.1 and 6.2 are the APIs to talk to the DESFire, Felica, and 14443B cards.

For those user prefer to use NFC Tama commands, the API will be described in section 6.3. Due to the PCSC limitation, the API can only be used after the card connection has been established. Therefore, for those Tama commands to drive the NFC reader (not the card), only be sent after the card handler has been detected by the PCSC resource manager.

6.1 DESFire Card Interface

The DESFire cards can be accessed via ISO7816-4 compliant framed APDU commands.

ISO 7816-4 Framed APDU to Write Card Data

Command	**Class**	**INS**	**P1**	**P2**	**Lc**	**File No.**	**Offset**	**Length**	**Data**	**Le**
Write Card Data	0x90	0x3D	00	00	XX	XX	XXXXXX	XXXXXX	XX…XX	00

Lc= 7+DataLength; Le=0

Command Output

Data Output
Response Data + SW1 SW2

Status Codes

Command	**SW1**	**SW2**	**Meaning**
Return Status	***90***	***00***	Success
	91	***XX***	Error (Please refer to ESFire Data Sheet)

6.2 Felica and 14443B Data Exchange Interface

After the reader polled the Felica or 14443B card, the application can use the SCardTransmit API (described in section 4) or the following SCardControl API to exchange data with the card. Please refer to section 3.2 about the PC/SC SCardControl API definition and following is the SCardControl data exchange API definition:

dwControlCode:

SCARD_CTL_CODE(2110)

lpInBuffer and nInBufferSize:

The input data sending to the card

lpOutBuffer, nOutBufferSize, and lpBytesReturned:

The output data return from the card

6.3 NFC Tama Interface

This API allows the application to send NFC PN531/PN532 command through the PCSC SCardControl API:

dwControlCode:

SCARD_CTL_CODE(2200)

lpInBuffer and nInBufferSize:

The input data sending to the reader

lpOutBuffer, nOutBufferSize, and lpBytesReturned:

The output data return from the reader

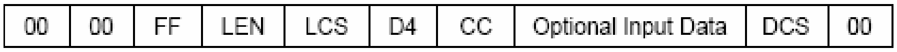
7 Proprietary APIs for RS232 eNetPads

This section is a proprietary API Sets to support the SmartCard, Mifare, Felica, and 14443B tags. This section is only used for RS232 eNetPad.

7.1 Command Format

The input command and response (output) command are in binary format. The following descriptions for the commands are showing in HEX (hexadecimal) mode.

7.1.1 Input Command



00	00	FF	LEN	LCS	D4	CC	Optional Input Data	DCS	00

LEN is the length of bytes from D4 till the end of optional input data, inclusively.

LCS is 1 packet length checksum. The LCS byte satisfies the relation: Lower byte of [LEN+LCS]=0x00.

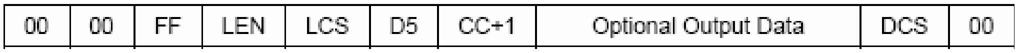
D4 means this packet is an input command. The value is 0xD4.

CC is the individual ISO 14443-A/Mifare command. It is described in next section. (0xD0, 0xD2, 0xD4,…etc.).

DCS is the packet checksum that satisfies the relation:
Lower byte of [D4 + CC + ... + DCS] = 0x00

Optional Input Data It is described in the following sections.

7.1.2 Response (Output) Command



00	00	FF	LEN	LCS	D5	CC+1	Optional Output Data	DCS	00

LEN is the length bytes starting from D5 till the end of optional output data, inclusively.

LCS is 1 packet length checksum. The LCS byte satisfies the relation: Lower byte of [LEN+LCS]=0x00.

D5 means this packet is an output command. The value is 0xD5.

CC+1 is the individual ISO 14443-A/Mifare response command. It is described in next section. (0xD1, 0xD3, 0xD5, etc.).

DCS is the packet checksum that satisfies the relation:

Lower byte of [D5 + CC+1 + … + DCS] = 0x00

Optional Output Data It is described in the following sections.

7.2 ISO 14443-A/Mifare Command

This section describes the ISO 14443-A / Mifare command sets. The Mifare Command Sets are the command sets to allow the user to access ISO 14443 type A tag.

All of the commands described in this section are based on the previous section's format. There are two parts for each command. For the input commands, this section is described from the "LEN" field until the end of the "Optional Input Data" field. For the output commands, it is described from the "D5" field until the end of the "Optional Output Data" field.

Every output has a result code. The result code is a short integer. In the case of success, the result code is 0 and the other return parameters followed after this result code. In the case of failure, only a non-zero result code is returned. Please refer to the Result Code section.

The Integer and Short data type showing on the following commands are stored as little endian (C convention in the PC), the least significant byte first (at the lowest address in the memory).

7.2.1 Select Single Tag (0xD0)

This command turns on the RF field first. And then try to select a single tag from the RF field. It returns failed right away if nothing can be detected. In case of success the command returns the tag ID and the type of the selected tag.

Input Frame

Parameter	Length (Bytes)	Data Type	Value
LEN	1	byte	02
LCS	1	byte	FE
D4	1	byte	D4
Command Code	1	byte	D0

Output Frame

Parameter	Length (Bytes)	Data Type	Value
D5	1	byte	D5
Command Code	1	byte	D1

Result Code	2	short	
SENS_RES	2	byte	
SEL_RES (Tag Type)	1	byte	Mifare 1K=08
Length of Tag ID	1	byte	Mifare 1K=4
Tag ID	Length of Tag ID	binary	

7.2.2 Select Single Tag with Polling Command (0xD2)

This command turns on the RF field first. And then try to select a single tag with a short a period of time. It returns failed if nothing can be detected after the polling duration. In case of success the command returns the tag ID and the type of the selected tag.

Input Frame

Parameter Length (Bytes) Data Type Value

Parameter	Length (Bytes)	Data Type	Value
LEN	1	byte	03
LCS	1	byte	FD
D4	1	byte	D4
Command Code	1	byte	D2
Polling Duration	1	byte	(Second)

If the Polling Duration Value is ‘0x00”, it sets the polling duration as default value, 15 seconds. The Polling Duration is the duration that reader will wait for the RFID tag (card) showing on the Reader’s RF field.

Output Frame

Parameter	Length (Bytes)	Data Type	Value
D5	1	byte	D5
Command Code	1	byte	D3
Result Code	2	short	
SENS_RES	2	byte	
SEL_RES (Tag Type)	1	byte	Mifare 1K=08
Length of Tag ID	1	byte	Mifare 1K=4
Tag ID	Length of Tag ID	binary	

7.2.3 Authentication Command (0xD4)

This command performs an authentication to access one sector of a tag. Only one sector can be accessed at the same time. For example, to access sector 0, the authentication needs to be successfully done with the block address 00 or 01 or 02 or 03.

Input Frame

Parameter	Length (Bytes)	Data Type	Value
LEN	1	byte	09
LCS	1	byte	F7
D4	1	byte	D4
Command Code	1	byte	D4
Key type	1	byte	60 or 61
Block Address	1	byte	00 ~ FF
Authentication Key	6	byte	

Key type:

60: authentication using key type A
61: authentication using key type B

Output Frame

Parameter	Length (Bytes)	Data Type	Value
D5	1	byte	D5
Command Code	1	byte	D5
Result Code	2	short	

7.2.4 Halt Command (0xD6)

This command sets a selected tag in the field into halt state. If the tag is unknown, a specific error code is returned. If the tag is already deselected, no action is performed and Status OK is returned.

Input Frame

Parameter	Length (Bytes)	Data Type	Value
LEN	1	byte	02
LCS	1	byte	FE
D4	1	byte	D4
Command Code	1	byte	D6

Output Frame

Parameter	Length (Bytes)	Data Type	Value
D5	1	byte	D5
Command Code	1	byte	D7
Result Code	2	short	

7.2.5 RF Off Command (0xD8)

This command can be used to switch off the RF field. The antenna is automatically switched on during a select command.

Input Frame

Parameter	Length (Bytes)	Data Type	Value
LEN	1	byte	02
LCS	1	byte	FE
D4	1	byte	D4
Command Code	1	byte	D8

Output Frame

Parameter	Length (Bytes)	Data Type	Value
D5	1	byte	D5
Command Code	1	byte	D9
Result Code	2	short	

7.2.6 Read Block Command (0xE0)

This command reads a data block on a tag (or pages for UltraLight tag). The size of returned valid data depends on the used tag. The block/page address range depends on the present tag as well. The reading requires a successful authentication for the Mifare tags.

Input Frame

Parameter	Length (Bytes)	Data Type	Value
LEN	1	byte	03
LCS	1	byte	FD
D4	1	byte	D4
Command Code	1	byte	E0
Block Address	1	byte	00 ~ FF

Output Frame

Parameter	Length (Bytes)	Data Type	Value
D5	1	byte	D5
Command Code	1	byte	E1
Result Code	2	short	
Block Data	16	byte	

Block Data:

Mifare Tag: 16 bytes block data.

UltraLight Tag: 4 pages data with 4 bytes each. 16 bytes total.

7.2.7 Write Block Command (0xE2)

This command writes data to a block. The writing requires a successful authentication for Mifare tags.

Input Frame

Parameter	Length (Bytes)	Data Type	Value
LEN	1	byte	13
LCS	1	byte	ED
D4	1	byte	D4
Command Code	1	byte	E2
Block Address	1	byte	00 ~ FF
Block Data	16	byte	

Output Frame

Parameter	Length (Bytes)	Data Type	Value
D5	1	byte	D5
Command Code	1	byte	E3
Result Code	2	short	

7.2.8 Read Value Command (0xE4)

This command reads the value from a value block on a tag. The reading value block requires a successful authentication and a correct formatted value block as source and correct settings of the access condition bits.

Input Frame

Parameter	Length (Bytes)	Data Type	Value
LEN	1	byte	03
LCS	1	byte	FD
D4	1	byte	D4
Command Code	1	byte	E4
Block Address	1	byte	00 ~ FF

Output Frame

Parameter	Length (Bytes)	Data Type	Value
D5	1	byte	D5
Command Code	1	byte	E3
Result Code	2	short	
Value Block Data	4	int	

7.2.9 Write Value Command (0xE6)

This command writes the value to a value block on a tag. The writing value block requires a successful authentication and a correct formatted value block as source and correct settings of the access condition bits.

Input Frame

Parameter	Length (Bytes)	Data Type	Value

LEN	1	byte	07
LCS	1	byte	F9
D4	1	byte	D4
Command Code	1	byte	E6
Block Address	1	byte	00 ~ FF
Value Block Data	4	int	

Output Frame

Parameter	Length (Bytes)	Data Type	Value
D5	1	byte	D5
Command Code	1	byte	E7
Result Code	2	short	

7.2.10 Adjust Value Command (0xE8)

The increments or decrements a value block with a defined value. This command requires a successful authentication and a correct formatted value block as source and correct settings of the access condition bits.

Input Frame

Parameter	Length (Bytes)	Data Type	Value
LEN	1	byte	08
LCS	1	byte	F8
D4	1	byte	D4
Command Code	1	byte	E8
Adjustment Type	1	byte	C0 or C1
Block Address	1	byte	00 ~ FF
Value Block Data	4	int	

Only two “Adjustment Type” are accepted:

C0: Decrement Value
C1: Increment Value

Output Frame

Parameter	Length (Bytes)	Data Type	Value
D5	1	byte	D5
Command Code	1	byte	E9
Result Code	2	short	

7.2.11 Write 4Byte Block Command (0xEA)

This command is for Mifare UltraLite tag only. It writes 4 bytes data to a Mifare ultralight page. It can be used for programming of the Mifare ultralight OTP (One time Programming) pages as well.

Input Frame

Parameter	Length (Bytes)	Data Type	Value
LEN	1	byte	07
LCS	1	byte	F9
D4	1	byte	D4
Command Code	1	byte	EA
Page Address	1	byte	00 ~ FF
Page Data	4	byte	

Output Frame

Parameter	Length (Bytes)	Data Type	Value
D5	1	byte	D5
Command Code	1	byte	EB
Result Code	2	short	

7.3 Felica, 14443-3B, and TCL Command

This section describes the Felica, 14443-3B, TCL, and data exchange command sets. For Felica tags, users will first use Section 3.1 command to poll Felica tags in the field, then using Section 3.4 command to exchange data. For 14443-3B tags, users will first use Section 3.2 command to poll 14443-3B tags in the field, then Section 3.4 command to exchange data. For ISO/IEC14443-4 compliant tags, users will first use Section 3.3 command to poll tags in the field, then using Section 3.4 command to exchange data. It is users' responsibility to compose corresponding command stored in the "Input Data Exchange" parameter field, where the command handles Felica, 14443-3B, or ISO/IEC14443-4 compliant tag.

The commands described in Sections 3.1 to 3.3 are based on the previous section's format. There are two parts for each command. For the input commands, this section is described from the "LEN" field until the end of the "Optional Input Data" field. For the output commands, it is described from the "D5" field until the end of the "Optional Output Data" field.

Every output has a result code. The result code is a short integer. In the case of success, the result code is 0 and the other return parameters followed after this result code. In the case of failure, only a non-zero result code is returned. Please refer to the Result Code section.

The Integer and Short data type showing on the following commands are stored as little endian (C convention in the PC), the least significant byte first (at the lowest address in the memory).

7.3.1 Felica Polling Command (0xDA)

This command turns on the RF field first, then tries to poll a single Felica tag with a short period of time. It returns failed if nothing can be detected after the polling duration. In case of success the command returns the POL_RES of the selected tags.

Input Frame

Parameter	Length (Bytes)	Data Type	Value
LEN	1	byte	03
LCS	1	byte	FD
D4	1	byte	D4
Command Code	1	byte	DA
Polling Duration	1	byte	(Second)

If the Polling Duration Value is '0x00", it sets the polling duration as default value, 15 seconds. The Polling Duration is the duration that reader will wait for the RFID tag (card) showing on the Reader's RF field.

Output Frame

Parameter	Length (Bytes)	Data Type	Value
D5	1	byte	D5
Command Code	1	byte	DB
Result Code	2	short	
NbTg	1	byte	
POL_RES		byte	

The return may contain more than one tag. The NbTg is the number of initialized tags with minimum 0 and maximum 2. The POL_RES may contain two tag data. The 1st byte of the tag data (Tg) indicate the tag position.

Each tag data will have following format:

Parameter	Length (Bytes)	Data Type
Tg	1	byte
POL_RES Length	1	byte
0x01	1	byte
NFCID2t	8	byte
Pad	8	byte
SYST_CODE (optional)	2	byte

7.3.2 14443-3B Polling Command (0xDC)

This command turns on the RF field first, then tries to select a single 14443-3B tag with a short period of time. It returns failed if nothing can be detected after the polling duration. In case of success the command returns the ATTRIB_RES of the selected tag.

Input Frame

Parameter	Length (Bytes)	Data Type	Value
LEN	1	byte	03
LCS	1	byte	FD
D4	1	byte	D4
Command Code	1	byte	DC
Polling Duration	1	byte	(Second)

If the Polling Duration Value is ‘0x00”, it sets the polling duration as default value, 15 seconds. The Polling Duration is the duration that reader will wait for the RFID tag (card) showing on the Reader’s RF field.

Output Frame

Parameter	Length (Bytes)	Data Type	Value
D5	1	byte	D5
Command Code	1	byte	DD
Result Code	2	short	
NbTg	1	byte	
POL_RES		byte	

The return may contain more than one tag. The NbTg is the number of initialized tags with minimum 0 and maximum 2. The ATTRIB_RES may contain two tag data. The 1st byte of the tag data (Tg) indicate the tag position.

Each tag data will have following format:

Parameter	Length (Bytes)	Data Type
Tg	1	byte
ATQB Response	12	byte
ATTRIB_RES Length	1	byte
ATTRIB_RES[]	ATTRIB_RES Length	byte

7.3.3 TCL Polling Command (WaitForCard) (0xDE)

This command turns on the RF field first, then tries to select a single 14443-4 tag with a short period of time. It returns failed if nothing can be detected after the polling duration. In case of success the command returns the ATTRIB_RES of the selected tag.

Input Frame

Parameter	Length (Bytes)	Data Type	Value
LEN	1	byte	03
LCS	1	byte	FD

D4	1	byte	D4
Command Code	1	byte	DE
Polling Duration	1	byte	(Second)

If the Polling Duration Value is ‘0x00”, it sets the polling duration as default value, 15 seconds. The Polling Duration is the duration that reader will wait for the RFID tag (card) showing on the Reader’s RF field.

Output Frame

Parameter	Length (Bytes)	Data Type	Value
D5	1	byte	D5
Command Code	1	byte	DF
Result Code	2	short	
NbTg	1	byte	
ATTRIB_RES		byte	

The return may contain more than one tag. The NbTg is the number of initialized tags with minimum 0 and maximum 2. The ATTRIB_RES may contain two tag data. The 1st byte of the tag data (Tg) indicate the tag position.

Each tag data will have following format:

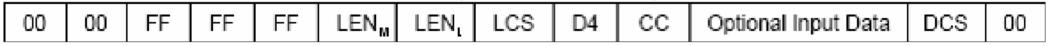
Parameter	Length (Bytes)	Data Type
Tg	1	byte
SENS_RES	2	byte
SEL_RES	1	byte
NFCIDLength	1	byte
NFCID1[]	NFCIDLength	byte
ATSLength	1	byte
ATS[]	ATSLength-1	byte

7.3.4 Data Exchange Command (0xEC)

The purpose of this command is for all the tag application to do the data exchange with tag. This command has an extended definition allowing exchanging more data.

Input Frame

In order to support more data to be able to send to the reader, the input command format need to be changed to following structure.



00	00	FF	FF	FF	LEN_M	LEN_L	LCS	D4	CC	Optional Input Data	DCS	00

The real length is coded in the two following bytes **LEN_M** (MSByte) and **LEN_L** (LSByte) with:

LENGTH = LEN_M x 256 + LEN_L coding the number of bytes in the data field (from D4 till the end of optional input data).

LCS is one byte Packet Length Checksum that satisfies the relation: **LEN_M +**

LEN$_L$ + LCS = 0x00

In the firmware implementation of the reader, the maximum length of the packet data (the Optional Input Data) is limited to 263 bytes.

User need to put one byte tag position (1 or 2) and following the exchange data.

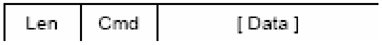
Parameter	Length (Bytes)	Data Type	Value
D4	1	byte	D4
Command Code	1	byte	EC
External 14443-4 Chaining	1	byte	0x00 or 0x01
Tg	1	byte	0x01 or 0x02
Input Exchange Data	LENGTH-4	byte	

The “External 14443-4 Chaining” parameter can only apply to ISO/IEC 14443-4 cards. If this parameter set to 0x01, the 14443-4 chaining will be implemented by the caller. If the parameter set to 0x00, the 14443-4 chaining will be taken care by the reader. For other cards, this parameter will be ignored and we strongly recommend set this to 0x00.

The “Tg” is the tag position, usually will be 0x01 if you have only one tag on the reader. If the “External 14443-4 Chaining” is set to 0x01, the “Tg” parameter will be ignored and we recommend set the “Tg” to 0x01.

The “Input Exchange Data” length could be from 0 to 262 bytes. It is users’ responsibility to compose command stored in the “Input Exchange Data” parameter.

• FeliCa Card
When the tag is a FeliCa card, the reader just transfers the data contained in the InputExchangeData buffer as they are. The Len and Cmd bytes of the FeliCa protocol must be present in this buffer.



Len	Cmd	[Data]

o **Len** is the length of the total InputExchangeData buffer.
o **Cmd** is the FeliCa specific command byte.
o **Data** is an optional array of data bytes depending on the command used.

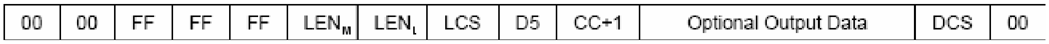
• ISO/IEC14443-4 Card
When the tag is ISO/IEC14443-4 compliant card and the “External 14443-4 Chaining”parameter set to 0x01, the ISO 14443-4 protocol mechanisms (including the Chaining, Error handling, and Waiting time Extension, etc.) must be implemented by the caller. Please refer to the ISO/IEC 14443-4 spec for the detail.

If the “External 14443-4 Chaining” parameter set to 0x00, the ISO 14443-4 protocol mechanisms are implemented (Chaining, Waiting time Extension, and Error handling) inside the reader. The InputExchangeData are interpreted by the reader to execute an ISO/IEC14443-4 exchange.

The C-APDU command length can be up to 261 bytes
(CLA-INS-P1-P2-P3-255 data bytes-Le) and the R-APDU returned from the reader can have a length of 258 bytes (256 data bytes-SW1-SW2)

Output Frame

The output command format also needs to be changed with following structure.



00	00	FF	FF	FF	LEN$_M$	LEN$_L$	LCS	D5	CC+1	Optional Output Data	DCS	00

The real length calculation is the same as the Input Frame described in the previous section (3.4.1)

Parameter	Length (Bytes)	Data Type	Value
D5	1	byte	D5
Command Code	1	byte	ED
Result Code	2	byte	
NFC_Status	1	byte	
Output Exchange Data	LENGTH-5	byte	

The “Output Exchange Data” length could be from 0 to 262 bytes.

The NFC_Status is the returned code from NXP NFC chip, where 0x00 is the successful result. Please refer to NXP NFC user manual for all other error codes.

7.4 Result Codes

Code(decimal)	**Description**
0	Operation succeeded
1	Insufficient balance
2	Failed to read card (Tag)
3	Failed to verify checksum.
7	Failed to write card (Tag)
9	Operation Partially succeeded
13	Top up over the limit
15	Not a value block.
30	Failed to init reader.
31	Selecting Tag/Pooling failed: No tag in RF Field.
33	Failed to authenticate card (Tag).
34	Deselect Failed

7.5 Baud Rate Supports for RS232 Communication

This is the baud rate change API, the default communication baud rate is 9600. The baud rate will be changed after the reader sending the successful response (00 00 FF 03 FD D5 29 00 02 00).

Input Frame

Parameter	Length (Bytes)	Data Type	Value
LEN	1	byte	03
LCS	1	byte	FD
D4	1	byte	D4
Command Code	1	byte	28
BR	1	byte	

BR is a byte indicating the baud rate:

- 0x00 9.6 kbaud
- 0x01 19.2 kbaud
- 0x02 38.4 kbaud
- 0x03 57.6 kbaud
- 0x04 115.2 kbaud
- 0x05 230.4 kbaud
- 0x06 460.8 kbaud
- 0x07 921.6 kbaud
- 0x08 1.288 Mbaud

Output Frame

Parameter	Length (Bytes)	Data Type	Value
D5	1	byte	D5
Command Code	1	byte	29
NFC_Status	1	byte	23 or 00

NFC_Status: 23 means failed and the baud rate is not changed.
00 means successfully process the baud rate change request and the new baud rate will be changed after this message.

8 Revision History

Ver.	**Date**	**Name**	**Description**
1.0	2008/05/12	Kenneth Cho	Initial draft
1.2	2008/09/29	Kenneth Cho	review