



SDK for PC/SC – Readme First

Overview

PC/SC is the de-facto standard to interface Personal Computers (PC) with Smart Card (SC), and -of course- with Smart Card Readers & Writers.

PC/SC is available on Windows and most Unix systems, including Linux and Mac OS X (through the PCSC-Lite open source stack).

This SDK provides samples for the Windows platform.

Overview (cont.)

Most samples provided within this **SpringCard SDK for PC/SC** are also available as ready-to-use binaries.

Just visit **SpringCard QuickStart for PC/SC** to download, install and run these binaries in a nutshell

www.springcard.com/solutions/pcsc-quickstart.html

Content

- ✓ Compatible products
- ✓ Links to related documentations
- ✓ How to install the SDK
- ✓ License
- ✓ Directory structure
- ✓ Focus on the key examples
- ✓ Other examples provided in the SDK
- ✓ Going further
- ✓ Contacting support

Compatible products



Prox'N'Roll PCSC



CSB6 / CSB6-HSP



CrazyWriter / CrazyWriter HSP



NFC'Roll



H663/H512

Reference documentation you'll need

- ✓ PC/SC on Windows:

<http://msdn.microsoft.com/>

(enter “winscard” or “ScardTransmit” in the search box)

- ✓ Java PC/SC API (javax.smartcardio):

<http://doc.java.sun.com/DocWeb/api/javax.smartcardio>

- ✓ SpringCard's Simplified documentation of the PC/SC API

<http://www.springcard.com/en/download/find/file/pmdz061>

Developer's Reference Manuals

- ✓ H663, CrazyWriter HSP, CSBHSB:
<http://www.springcard.com/en/download/find/file/pmd2271>
- ✓ H512, NFC'Roll:
<http://www.springcard.com/en/download/find/file/pmd2176>
- ✓ CSB6, Prox'N'Roll, CrazyWriter:
<http://www.springcard.com/en/download/find/file/pmd841p>

How to install the PC/SC SDK

- ✓ To install the complete SpringCard PC/SC SDK :
 - Just unzip the archive on your hard drive
 - Recommended location is C:\DEV\SPRINGCARD\PCSC

License

SpringCard's SDK are available free of charge.

The license allows you to use the featured software (binary or source) freely, **provided that the software or any derivative works is used only in link with genuine SpringCard products.**

Please read **LICENSE.TXT** for details.

Directory structure

- ✓ **SAMPLES/C**
 - Sample programs written in ANSI C, and portable to virtually any OS supporting PC/SC
- ✓ **SAMPLES/DOTNET**
 - Sample programs written in C# and VB, targetting the .NET framework
- ✓ **SAMPLES/JAVA**
 - Sample programs written in Java, using the javax.smarcardio class available on some systems
- ✓ **SAMPLES/WIN32**
 - Samples programs written in either C or C++, that targets the Windows OS

Directory structure

✓ **BINARIES**

- Pre-compiled binaries for Windows. Some binaries rely on the .NET framework (v4, client profile). Please install the framework beforehand

✓ **DOCS**

- Contains the documentation of the libraries provided by SpringCard to ease working with some particular cards on top of PC/SC

✓ **LIBRARY/DOTNET**

- Source code of the libraries provided by SpringCard to ease working with PC/SC, and with some particular cards on top of PC/SC, from C# or VB projects running in the .NET framework

Focus on the key examples

- ✓ Memory Card Tool
- ✓ PC/SC Scriptor
- ✓ NFC Tag Tool

Memory Cards Tool

Memory Card Tool

A unique tool that:

- ✓ displays the content of a memory card
- ✓ allows to write a card content

The screenshot shows the MemoryCardTool application window. The title bar reads "MemoryCardTool". The menu bar contains "File" and "Help". The main area features a decorative orange wave graphic and the "springcard" logo. Below the logo, it states "Recognized card: Innovision Topaz". There are two input fields for "Specify maximum values: P1" (set to 0) and "P2" (set to 128), with a refresh button to the right. A table displays the card's content in hexadecimal and ASCII. The ASCII column shows a VCard format. At the bottom, there is a "Write Modifications to the Card" button and a status bar with the following information: "Reader: SpringCard Prox'N'Roll Contactle: Status: PRESENT,INUSE Card ATR: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 02 00 30 00 00 00 5A".

P1	P2	Hex.	ASCII
00	00	3D 72 3D 00 00 02 25 00 E1 10 0E 00 03 51 D2 0C	=x=...%.á....Q0.
00	10	42 74 65 78 74 2F 78 2D 76 43 61 72 64 42 45 47	Btext/x-vCardBEG
00	20	49 4E 3A 56 43 41 52 44 0D 0A 56 45 52 53 49 4F	IN:VCARD..VERSIO
00	30	4E 3A 33 2E 30 0D 0A 46 4E 3A 4A 6F 68 6E 20 44	N:3.0..FN:John D
00	40	6F 65 0D 0A 4F 52 47 3A 53 70 72 69 6E 67 43 61	oe..ORG:SpringCa
00	50	72 64 0D 0A 45 4E 44 3A 56 43 41 52 44 0D 0A FE	rd..END:VCARD..p
00	60	44 0D 0A FE FE 0D 0A 00 55 55 AA AA 00 00 00 00	D..pb...UU*^.....
00	70	01 60 00 00 00 00 00 00 00 00 00 00 00 00 00

Memory Card Tool

- ✓ In **SAMPLES/DOTNET/MEMORYCARDTOOL**
- ✓ Language = C#
- ✓ Target = .NET 4
- ✓ The project opens and builds using #Develop 4, the open-source IDE for .NET :
<http://www.icsharpcode.net/OpenSource/SD/Default.aspx>
- ✓ Porting to Microsoft Visual C# Express 2010 is straightforward

Example: Mifare Classic

ASCII translation of each sector

Recognized card

MemoryCardTool

File Help

Recognized card: Mifare Classic 1k

Number of sectors: 16 P1 0 P2 63

Sector	Hex	ASCII
0	00 1B 39 DD ED 12 08 04 00 62 63 64 65 66 67 68 69	.9Ÿi....bcdefghi
0	01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0	02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1	00 83 65 63 74 6F 72 20 31 3A 20 62 6C 6F 63 6B 30	Sector 1: block0
1	01 53 65 63 74 6F 72 20 31 3A 20 62 6C 6F 63 6B 31	Sector 1: block1
1	02 53 65 63 74 6F 72 20 31 3A 20 62 6C 6F 63 6B 32	Sector 1: block2
2		
3	00 62 6C 6B 2E 30 20 6F 66 20 73 65 63 74 6F 72 33	blk.0 of sector3
3	01 62 6C 6B 2E 31 20 6F 66 20 73 65 63 74 6F 72 33	blk.1 of sector3
3	02 62 6C 6B 2E 32 20 6F 66 20 73 65 63 74 6F 72 33	blk.2 of sector3
4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
4	01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
4	02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Reader: SpringCard Prox'N'Roll Contactless Status: PRESENT,INUSE Card ATR: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 01 00 00 00 6A

Change the number of sectors

Read card again, with specified number of sectors

Hexadecimal content of each sector

Update sector's content (if allowed)

Read sector with specified keys

Change sector's keys and access conditions

Example: Inside Contactless PicoTAG

Recognized card

The screenshot shows the MemoryCardTool interface. At the top, it says "Recognized card: Inside Contactless PICOTAG/PICOPASS". Below this, there are input fields for "Specify maximum values: P1" (set to 0) and "P2" (set to 31). A table displays data for pages 00 to 15, with columns for P1, P2, Hex, and ASCII. The ASCII column shows a mix of characters and symbols. At the bottom, there is a "Write Modifications to the Card" button and a status bar showing "Reader: SpringCard Prox'N'Roll Contactles Status: PRESENT,INUSE" and "Card ATR: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 06 FF B7 00 00 00 26".

P1	P2	Hex	ASCII
00	00	44 AF 6E 00 F9 FF 12 E0	D n . ù ý . à
00	01	12 FF FF FF FF E9 1F FF 3C	. ý ý ý é . ý <
00	02	FC FF FF FF FF FF FF FF	ù ý ý ý ý ý ý ý
00	03	FF FF FF FF FF FF FF FF	ý ý ý ý ý ý ý ý
00	04	FF FF FF FF FF FF FF FF	ý ý ý ý ý ý ý ý
00	05	FF FF FF FF FF FF FF FF	ý ý ý ý ý ý ý ý
00	06	FF FF FF FF FF FF FF FF	ý ý ý ý ý ý ý ý
00	07	FF FF FF FF FF FF FF FF	ý ý ý ý ý ý ý ý
00	08	FF FF FF FF FF FF FF FF	ý ý ý ý ý ý ý ý
00	09	FF FF FF FF FF FF FF FF	ý ý ý ý ý ý ý ý
00	0A	FF FF FF FF FF FF FF FF	ý ý ý ý ý ý ý ý
00	0B	FF FF FF FF FF FF FF FF	ý ý ý ý ý ý ý ý
00	0C	FF FF FF FF FF FF FF FF	ý ý ý ý ý ý ý ý
00	0D	FF FF FF FF FF FF FF FF	ý ý ý ý ý ý ý ý
00	0E	FF FF FF FF FF FF FF FF	ý ý ý ý ý ý ý ý
00	0F	FF FF FF FF FF FF FF FF	ý ý ý ý ý ý ý ý
00	10	FF FF FF FF FF FF FF FF	ý ý ý ý ý ý ý ý
00	11	FF FF FF FF FF FF FF FF	ý ý ý ý ý ý ý ý
00	12	FF FF FF FF FF FF FF FF	ý ý ý ý ý ý ý ý
00	13	FF FF FF FF FF FF FF FF	ý ý ý ý ý ý ý ý
00	14	FF FF FF FF FF FF FF FF	ý ý ý ý ý ý ý ý
00	15	FF FF FF FF FF FF FF FF	ý ý ý ý ý ý ý ý

Read card again, with specified P1 and P2

Specify Max values for P1 and P2

ASCII translation

Address of each page or block

Hexadecimal content

Update card's content (if allowed)



How to read / write a Memory Card

APDUs for a Mifare Classic

- ✓ To Read: FF F3 00 P2 Le, where P2 is the address of the block and Le is the number of bytes to read
- ✓ To Write: FF F4 00 P2 Lc Data, where Lc is the length of data to write and Data is the data itself
- ✓ Please refer to the Developer's guide to specify the keys, if the default keys don't work

APDUs for another Memory Card

- ✓ To Read : FF B0 P1 P2 Le, where P1 and P2 are the two address bytes (Most Significant Byte First), and Le the number of bytes to read
- ✓ To Write : FF D6 P1 P2 Lc Data, where P1 and P2 are the two address bytes (Most Significant Byte First), Lc is the number of bytes to write, and Data is the data to write
- ✓ Please refer to the Developer's guide to know the different allowed values for P1, P2, Le and Lc, for each supported Memory Card

Source code for reading

The SpringCardPCSC.cs class is used.

First, create an ScardChannel object (“channel”), from the reader name :

- ✓ `ScardReader reader = new ScardReader(readerName);`
- ✓ `ScardChannel channel = new SCardChannel(reader);`

Then, to read “length” bytes at address “address”:

- ✓ `CAPDU capdu = new CAPDU(0xFF, 0xB0, (byte) (address / 0x0100), (byte) (address % 0x0100), length);`
- ✓ `RAPDU rapdu = channel.Transmit(capdu);`
- ✓ `byte[] bytes_read = rapdu.data.GetBytes();`

Source code for reading

The SpringCardPCSC.cs class is used.

First, create an ScardChannel object (“channel”), from the reader name :

- ✓ `ScardReader reader = new ScardReader(readerName);`
- ✓ `ScardChannel channel = new SCardChannel(reader);`

Then, to read “length” bytes at address “address”:

- ✓ `CAPDU capdu = new CAPDU(0xFF, 0xB0, (byte) (address / 0x0100), (byte) (address % 0x0100), length);`
- ✓ `RAPDU rapdu = channel.Transmit(capdu);`
- ✓ `byte[] bytes_read = rapdu.data.GetBytes();`

Source code for writing

Once the channel is created, we only need to send the writing APDU to write “data” at address “address”:

- ✓ CAPDU capdu = new CAPDU(0xFF, 0xD6, (byte) (address / 0x0100), (byte) (address % 0x0100), data);
- ✓ RAPDU rapdu = channel.Transmit(capdu);
- ✓ if (rapdu.SW != 0x9000)
 - → Error !

How to recognize the card ?

- ✓ The ATR is used to recognize the card and differentiate between cards with sectors (Mifare Classic) and cards without sectors.
 - Check <http://smartcard-atr.appspot.com/> for information about ATRs
- ✓ The ATR is further analyzed with cards without sectors, to identify precisely the card and deduce the number of pages or blocks, and the number of bytes per page or block.
- ✓ For some ATRs (those from the Mifare UltraLight family), further identification is performed in reading the pages until we find a duplication (same data again) or until an error occurs.
- ✓ To obtain the ATR, use the previously defined channel :
 - `string atr = channel.CardAtr.AsString("");`

How to detect when a card is inserted ?

We have already created an ScardReader object, from the reader's name :

- ✓ `ScardReader reader = new ScardReader(readerName);`

Once it is created, we can track all the changes on this reader, via the `StartMonitor()` method, in a background thread:

- ✓ `reader.StartMonitor(new ScardReader.StatusChangeCallback (ReaderStatusChanged));`

`ReaderStatusChanged` is the callback, ie: the method called each time the background thread detects any change on the reader. This method analyses the reader state and the ATR of the card.

- ✓ `void ReaderStatusChanged(uint ReaderState, CardBuffer CardAtr) { ... }`

If a card is effectively inserted in the reader, it will then be read.

Advanced: reading in a background thread

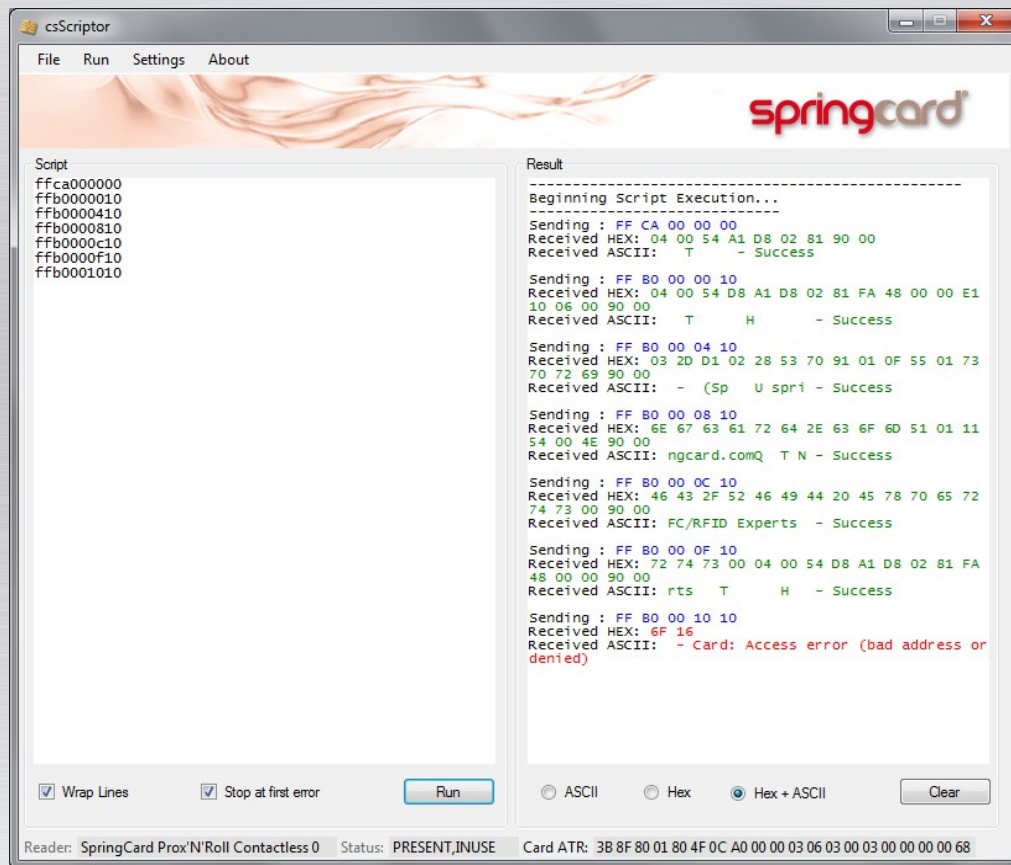
- ✓ When a card is read, the main screen might freeze during the process.
- ✓ To avoid this behavior, reading is performed in a background thread.
 - `Thread cardthread = new Thread(card_read_proc);`
 - Or: `Thread cardthread = new Thread(read_card_again);`
 - `cardthread.Start();`
- ✓ Once the card is read, the thread exits in the `onError()`, or in the `onCardRead()` callback method.

PC/SC Scriptor

(csScriptor)

csScriptor

- ✓ This tool allows to send several APDUs to a card in a row.
- ✓ For example, first ask the serial number and then read the card
- ✓ It is ideal to work with SmartCards, where several APDUs are needed to first select an application and then read is content



The screenshot shows the csScriptor application window. The title bar reads "csScriptor". The menu bar includes "File", "Run", "Settings", and "About". The main window has a decorative header with the "springcard" logo. The interface is split into two main panes: "Script" on the left and "Result" on the right. The "Script" pane contains a list of hexadecimal APDUs: ffc000000, ffb000010, ffb0000410, ffb0000810, ffb0000c10, ffb0000f10, and ffb0001010. The "Result" pane shows the execution output, starting with "Beginning Script Execution...". It displays four successful transactions, each with "Sending" (hex), "Received HEX", and "Received ASCII" information. The first three transactions are successful, while the fourth one fails with the message "Card: Access error (bad address or denied)". At the bottom of the window, there are checkboxes for "Wrap Lines" and "Stop at first error", a "Run" button, radio buttons for "ASCII", "Hex", and "Hex + ASCII" (which is selected), and a "Clear" button. The status bar at the very bottom shows "Reader: SpringCard Prox'N'Roll Contactless 0", "Status: PRESENT_INUSE", and "Card ATR: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 03 00 00 00 68".

```
Script
ffc000000
ffb000010
ffb0000410
ffb0000810
ffb0000c10
ffb0000f10
ffb0001010

Result
-----
Beginning Script Execution...
Sending : FF CA 00 00 00
Received HEX: 04 00 54 A1 D8 02 81 90 00
Received ASCII: T - Success

Sending : FF 80 00 00 10
Received HEX: 04 00 54 D8 A1 D8 02 81 FA 48 00 00 E1
10 06 00 90 00
Received ASCII: T H - Success

Sending : FF 80 00 04 10
Received HEX: 03 2D D1 02 28 53 70 91 01 0F 55 01 73
70 72 69 90 00
Received ASCII: - (sp U spr) - Success

Sending : FF 80 00 08 10
Received HEX: 6E 67 63 61 72 64 2E 63 6F 6D 51 01 11
54 00 4E 90 00
Received ASCII: ngcard.comQ T N - Success

Sending : FF 80 00 0C 10
Received HEX: 46 43 2F 52 46 49 44 20 45 78 70 65 72
74 73 00 90 00
Received ASCII: FC/RFID Experts - Success

Sending : FF 80 00 0F 10
Received HEX: 72 74 73 00 04 00 54 D8 A1 D8 02 81 FA
48 00 00 90 00
Received ASCII: rts T H - Success

Sending : FF 80 00 10 10
Received HEX: 6F 16
Received ASCII: - Card: Access error (bad address or
denied)

Wrap Lines [checked] Stop at first error [checked] Run [button]
ASCII [radio] Hex [radio] Hex + ASCII [radio] Clear [button]

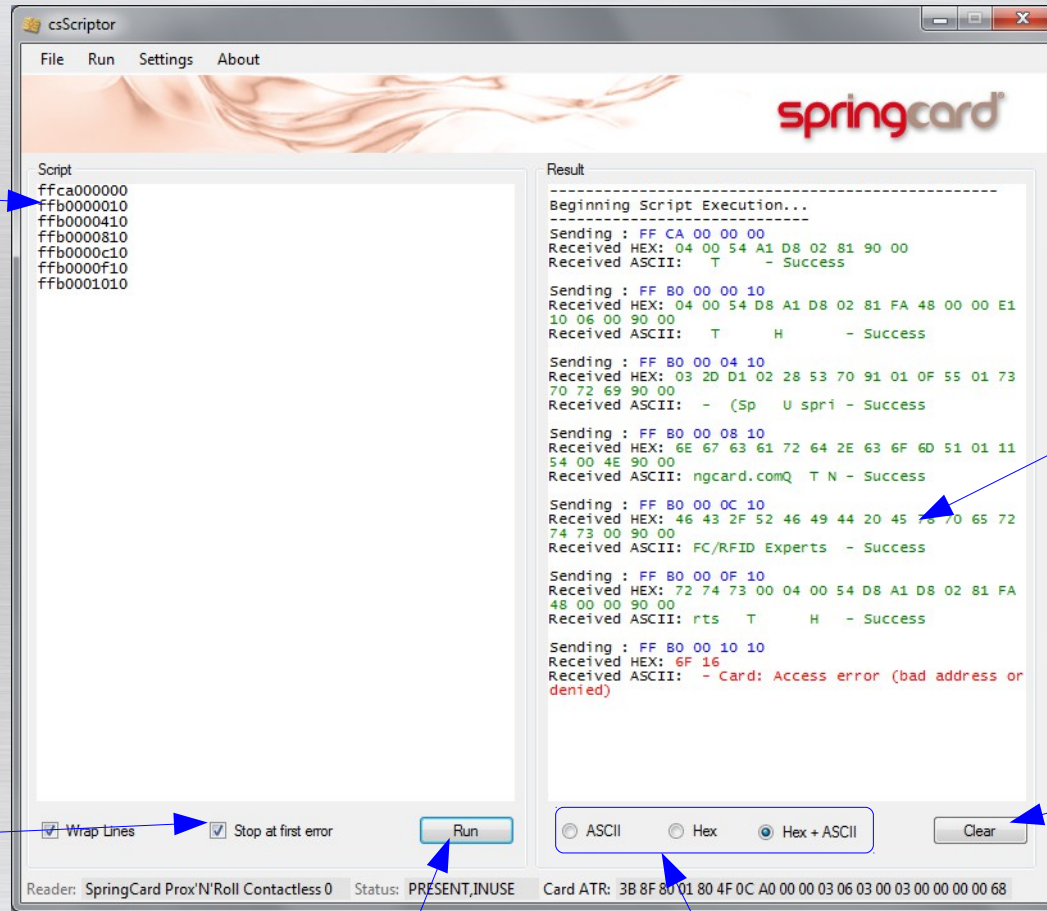
Reader: SpringCard Prox'N'Roll Contactless 0 Status: PRESENT_INUSE Card ATR: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 03 00 00 00 68
```

csScriptor

- ✓ In **SAMPLES/DOTNET/CSCRIPTOR**
- ✓ Language = C#
- ✓ Target = .NET 4
- ✓ The project opens and builds using #Develop 4, the open-source IDE for .NET :
<http://www.icsharpcode.net/OpenSource/SD/Default.aspx>
- ✓ Porting to Microsoft Visual C# Express 2010 is straightforward

Overview

Write all the APDUs in this box



The card answers are given in this box

Uncheck this box if you want the script to continue even if errors are encountered

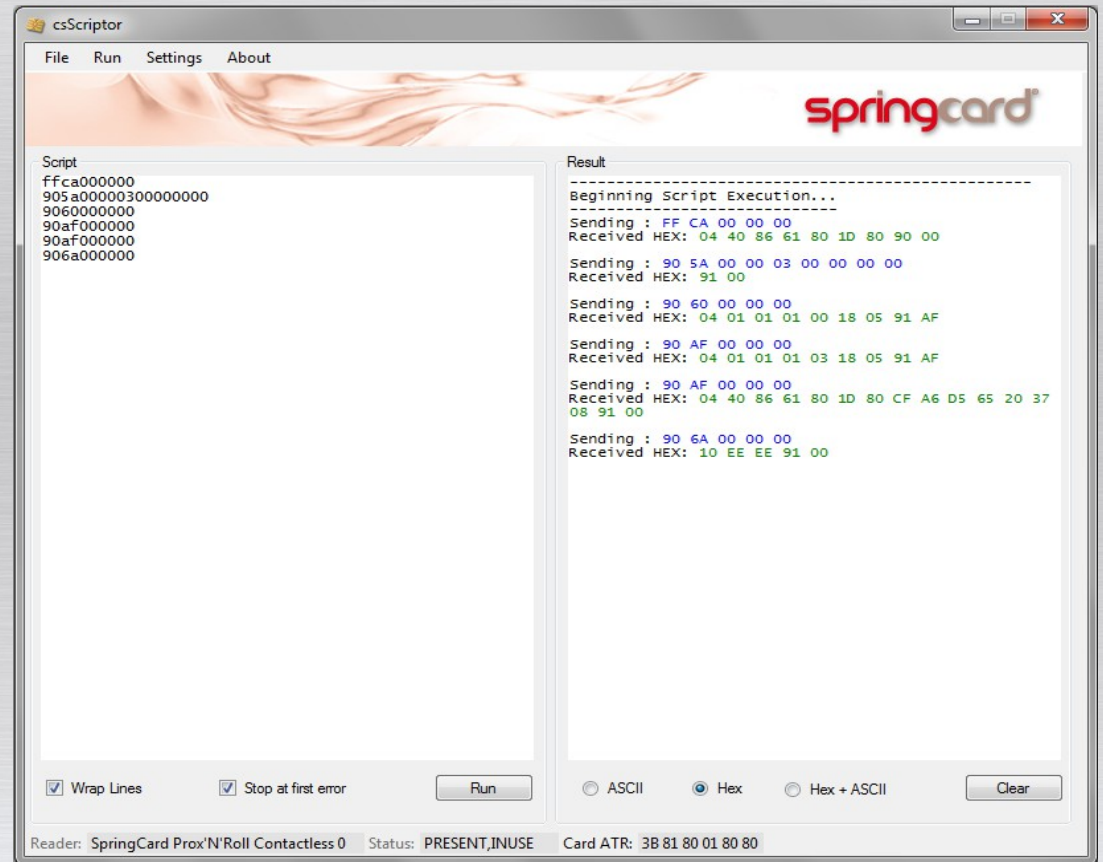
Click on "Clear" to clear the result screen

Click on "Run" to send the APDUs to the card

Choose output format

Example: a DESFire card

- ✓ Get Serial Number
- ✓ Select Application '00 00 00'
- ✓ GetVersion (3 APDUs)
- ✓ GetApplicationIDs



The screenshot shows the csScriptor application window. The title bar reads 'csScriptor'. The menu bar includes 'File', 'Run', 'Settings', and 'About'. The main area is divided into two panes: 'Script' on the left and 'Result' on the right. The 'Script' pane contains the following text:

```
ffca000000
905a0000300000000
9060000000
90af000000
90af000000
906a000000
```

The 'Result' pane shows the output of the script execution, starting with 'Beginning Script Execution...'. It displays several send/receive cycles in hexadecimal:

```
-----
Sending : FF CA 00 00 00
Received HEX: 04 40 86 61 80 1D 80 90 00

Sending : 90 5A 00 00 03 00 00 00 00
Received HEX: 91 00

Sending : 90 60 00 00 00
Received HEX: 04 01 01 01 00 18 05 91 AF

Sending : 90 AF 00 00 00
Received HEX: 04 01 01 01 03 18 05 91 AF

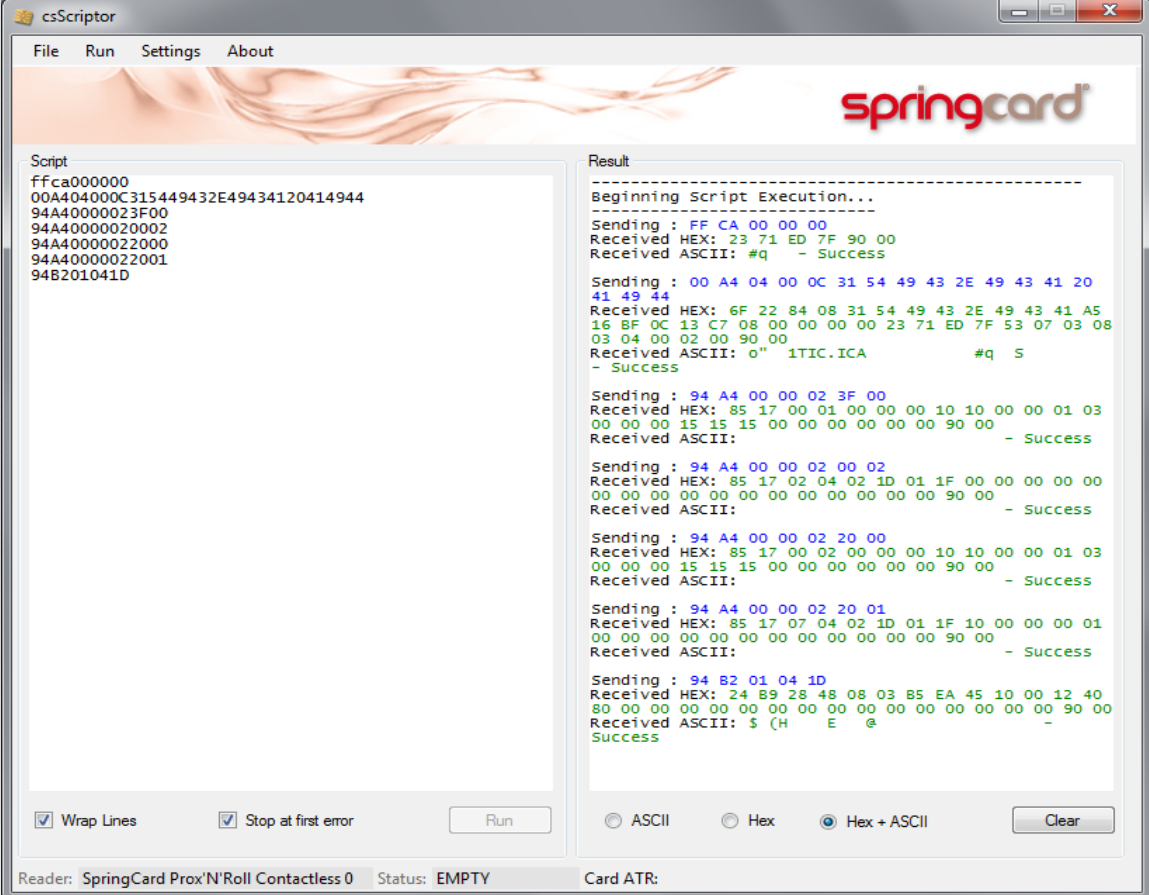
Sending : 90 AF 00 00 00
Received HEX: 04 40 86 61 80 1D 80 CF A6 D5 65 20 37
08 91 00

Sending : 90 6A 00 00 00
Received HEX: 10 EE EE 91 00
```

At the bottom of the window, there are checkboxes for 'Wrap Lines' (checked) and 'Stop at first error' (checked), a 'Run' button, and radio buttons for 'ASCII', 'Hex' (selected), and 'Hex + ASCII', along with a 'Clear' button. The status bar at the very bottom displays: 'Reader: SpringCard Prox'N'Roll Contactless 0 Status: PRESENT,INUSE Card ATR: 3B 81 80 01 80 80'. The SpringCard logo is visible in the top right corner of the application window.

Example: a Calypso card

- ✓ Get Serial Number
- ✓ Select 1TIC.ICA Application
- ✓ Select MF
- ✓ Select EF_ICC
- ✓ Select DF_Calypso
- ✓ Select EF_Enr
- ✓ Read ENR, Record#1



```
csScripTor
File Run Settings About
springcard

Script
FFca000000
00A404000C315449432E49434120414944
94A40000023F00
94A40000022002
94A40000022000
94A40000022001
94B201041D

Result
-----
Beginning Script Execution...
-----
Sending : FF CA 00 00 00
Received HEX: 23 71 ED 7F 90 00
Received ASCII: #q - Success

Sending : 00 A4 04 00 0C 31 54 49 43 2E 49 43 41 20
41 49 44
Received HEX: 6F 22 84 08 31 54 49 43 2E 49 43 41 A5
16 BF 0C 13 C7 08 00 00 00 00 23 71 ED 7F 53 07 03 08
03 04 00 02 00 90 00
Received ASCII: 0" 1TIC.ICA #q S
- Success

Sending : 94 A4 00 00 02 3F 00
Received HEX: 85 17 00 01 00 00 00 10 10 00 00 01 03
00 00 00 15 15 15 00 00 00 00 00 90 00
Received ASCII:
- Success

Sending : 94 A4 00 00 02 00 02
Received HEX: 85 17 02 04 02 1D 01 1F 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 90 00
Received ASCII:
- Success

Sending : 94 A4 00 00 02 20 00
Received HEX: 85 17 00 02 00 00 00 10 10 00 00 01 03
00 00 00 15 15 15 00 00 00 00 00 90 00
Received ASCII:
- Success

Sending : 94 A4 00 00 02 20 01
Received HEX: 85 17 07 04 02 1D 01 1F 10 00 00 00 01
00 00 00 00 00 00 00 00 00 00 90 00
Received ASCII:
- Success

Sending : 94 B2 01 04 1D
Received HEX: 24 B9 28 48 08 03 85 EA 45 10 00 12 40
80 00 00 00 00 00 00 00 00 00 00 00 00 00 90 00
Received ASCII: $ (H E @
Success

 Wrap Lines  Stop at first error Run
 ASCII  Hex  Hex + ASCII Clear

Reader: SpringCard Prox'N'Roll Contactless 0 Status: EMPTY Card ATR:
```

Example: a payment card

- ✓ Select Payment Applications
 - Try MasterCard and Visa
- ✓ Read all potential records

```
csScriptor
File Run Settings About
springcard
Script
00 A4 04 00 0E 31 50 41 59 2E 53 59 53 2E 44 44 46
30 31
00 a4 04 00 0e 32 50 41 59 2e 53 59 53 2e 44 44 46
30 31 00
00 A4 04 00 07 A0 00 00 00 04 10 10 00
00 A4 04 00 07 A0 00 00 00 03 10 10 00
00 B2 01 0C 00
00 B2 02 0C 00
00 B2 03 0C 00
00 B2 04 0C 00
00 B2 05 0C 00
00 B2 06 0C 00
00 B2 07 0C 00
00 B2 08 0C 00
00 B2 09 0C 00
00 B2 0A 0C 00
00 B2 0B 0C 00
00 B2 0C 0C 00
00 B2 0D 0C 00
00 B2 0E 0C 00
00 B2 0F 0C 00
00 B2 10 0C 00
00 B2 01 14 00
00 B2 02 14 00
00 B2 03 14 00
00 B2 04 14 00
00 B2 05 14 00
00 B2 06 14 00
00 B2 07 14 00
00 B2 08 14 00
00 B2 09 14 00
00 B2 0A 14 00
00 B2 0B 14 00
00 B2 0C 14 00
00 B2 0D 14 00
00 B2 0E 14 00
00 B2 0F 14 00
00 B2 10 14 00
00 B2 01 1C 00
00 B2 07 1C 00
Result
-----
Beginning Script Execution...
-----
Sending : 00 A4 04 00 0E 31 50 41 59 2E 53 59 53
2E 44 44 46 30 31
Received HEX: 90 00
Received ASCII: - Success

Sending : 00 A4 04 00 0E 32 50 41 59 2E 53 59 53
2E 44 44 46 30 31 00
Received HEX:
Received ASCII: 2PAY.SYS.DDF01 90 00
VISA - Success

Sending : 00 A4 04 00 07 A0 00 00 00 04 10 10 00
Received HEX: 6A 82
Received ASCII: - Check error : file not found

Sending : 00 A4 04 00 07 A0 00 00 00 03 10 10 00
Received HEX:
Received ASCII: BP VISA DEBIT
Success

Sending : 00 B2 01 0C 00
Received HEX: 90 00
Received ASCII: - Success

Sending : 00 B2 02 0C 00
Received HEX:
Wrap Lines [x] Stop at first error [ ] Run
Reader: SpringCard Prox'N'Roll Contactless 0 Status: EMPTY Card ATR:
ASCII [ ] Hex [ ] Hex + ASCII [x] Clear
```

NFC Tags Tool

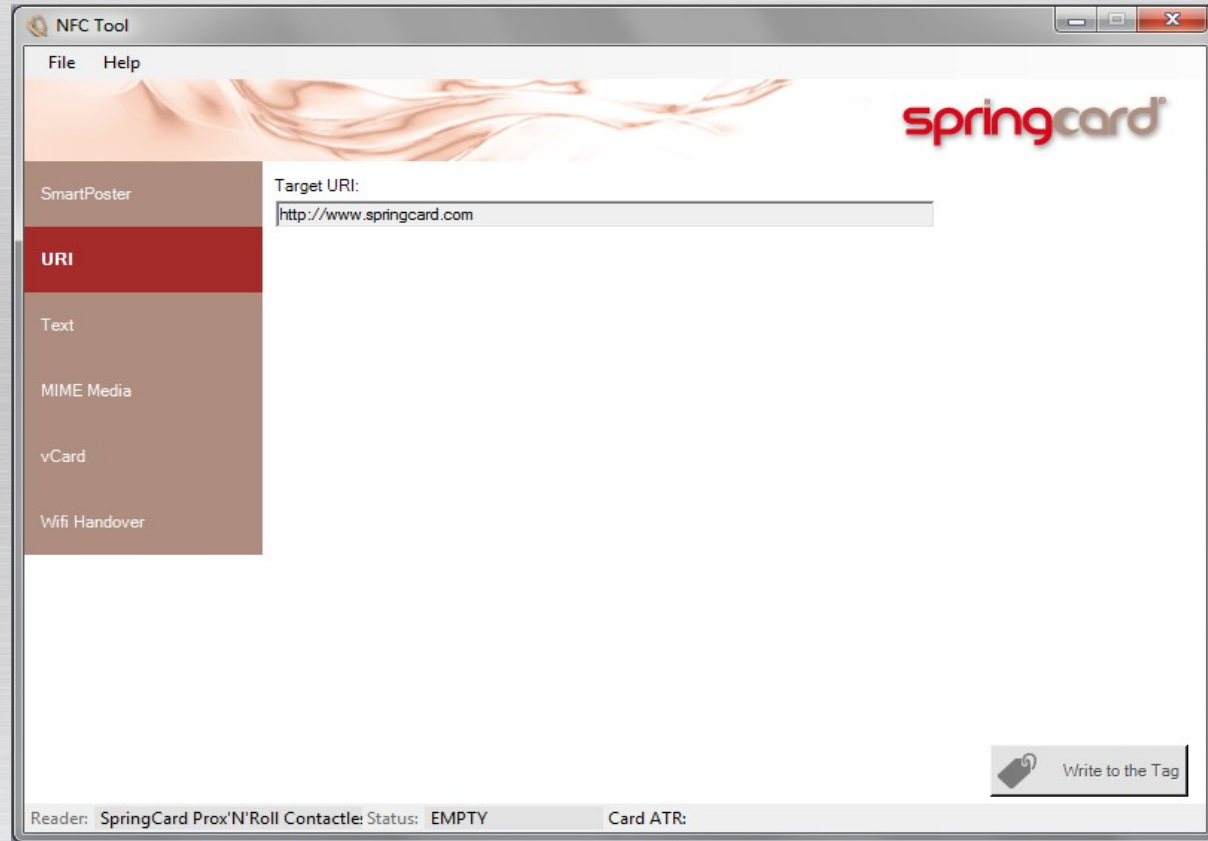
(NFCTool)

What is an NFC Forum tag ?

- ✓ An NFC Forum tag is a card, which content is valid in relation to the requirements of the NFC Forum
- ✓ 4 “types” are described :
 - NFC Forum Type 1
 - NFC Forum Type 2
 - NFC Forum Type 3
 - NFC Forum Type 4
- ✓ For more information :
 - <http://www.nfc-forum.org/home/>

NfcTool

- ✓ Nfc Tool enables to:
 - create NFC tags
 - write their content
 - read their content
- ✓ Supported contents:
 - SmartPoster
 - URI
 - Text
 - MIME Media
 - vCard
 - Wifi Handover
- ✓ Supported tags:
 - Type 2
 - Type 4



NDEF ? RTD ?

NDEF stands for “NFC Data Exchange Format”

- ✓ It contains a Type and a Payload
- ✓ The type defines the NDEF
- ✓ The payload contains the data

RTD stands for “Record Type Definition”.

An RTD is an NDEF, that has an NFC-specific type, which can be an:

- NFC Forum Well Known Type
- NFC External Type

NDEFs supported by NfcTool

Here is the list of all the NDEFs supported by NfcTool:

- ✓ RtdAlternativeCarrier
- ✓ RtdHandoverSelector
- ✓ RtdMedia
- ✓ RtdSmartPoster
- ✓ RtdText
- ✓ RtdUri
- ✓ RtdVCard

For more information on those objects, please visit our website, where online information is available:

RAJOUTER URL

NfcTool

- ✓ In **SAMPLES/DOTNET/NFCTOOL**
- ✓ Language = C#
- ✓ Target = .NET 4
- ✓ The project opens and builds using #Develop4, the open-source IDE for .NET :
<http://www.icsharpcode.net/OpenSource/SD/Default.aspx>
- ✓ Porting to Microsoft Visual C# Express 2010 is straightforward

How to create a Smartposter ?

1. Place a tag on your selected reader

2. Fill in the fields

The screenshot shows the 'NFC Tool' application window. The title bar reads 'NFC Tool'. The menu bar contains 'File' and 'Help'. The 'springcard' logo is in the top right corner. On the left, a sidebar lists options: 'SmartPoster' (highlighted in red), 'URI', 'Text', 'MIME Media', 'vCard', and 'Wifi Handover'. The main area contains the following fields:

- 'Target URI:' with a text box containing 'http://www.springcard.com'
- 'Text:' with a text box containing 'NFC/RFID Experts' and a 'Lang:' dropdown menu.
- 'Action to perform:' with a dropdown menu.
- 'Estimated size of the target (in bytes):' with a text box.
- 'MIME type of the target:' with a dropdown menu.

At the bottom right, there is a button with a tag icon and the text 'Write to the Tag'. The status bar at the bottom displays: 'Reader: SpringCard Prox'N'Roll Contactle: Status: PRESENT,INUSE Card ATR: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 03 00 00 00 68'.

3. Click on "Write to the Tag"

Writing a Type 2 Tag

A Type 2 Tag is memory card.

To write, the APDU is “FF D6 P1 P2 Lc Data”, where P1 and P2 are the two address bytes (Most Significant Byte First), Lc is the number of bytes to write, and Data is the data to write.

The `NfcTagType2.WriteBinary()` method is used, where the APDU is transmitted to the card through the `ScardChannel` object (the same as in `MemoryCardTool`):

- ✓ `ScardReader reader = new ScardReader(readerName);`
- ✓ `ScardChannel cardchannel = new SCardChannel(reader)`
- ✓ `CAPDU capdu = new CAPDU(0xFF, 0xD6, (byte) (address / 0x0100), (byte) (address % 0x0100), data)`
- ✓ `RAPDU rapdu = channel.Transmit(capdu);`
- ✓ `if (rapdu.SW != 0x9000)`
 - → Error !

Writing a Type 4 Tag

Assuming an already formatted Type 4 Tag, we first need to select the NDEF File: APDU=00 A4 00 0C 02 E1 04 .

We use the `NfcTagType4.SelectFile(ushort file_id)` method, where `file_id=0xE104`:

- ✓ `CAPDU capdu = new CAPDU(0x00, 0xA4, 0x00, 0x0C, (new CardBuffer(file_id)).GetBytes());`
- ✓ `RAPDU rapdu = channel.Transmit(capdu);`
- ✓ `if (rapdu.SW != 0x9000)`
 - → Error !

Then, use the APDU “FF D6 P1 P2 Lc Data”, where P1 and P2 are the two address bytes (Most Significant Byte First), Lc is the number of bytes to write, and Data is the data to write.

We use the `NfcTagType4.WriteBinary(SCardChannel channel, ushort offset, byte[] buffer)` method:

- ✓ `CAPDU capdu = new CAPDU(0x00, 0xD6, (byte) (offset / 0x0100), (byte) (offset % 0x0100), buffer);`
- ✓ `RAPDU rapdu = channel.Transmit(capdu);`
- ✓ `if (rapdu.SW != 0x9000)`
 - → Error !

Formatting a DESFire EV1 into a Type 4 Tag

We use a Command Line Application to format a DESFire EV1 into a Type 4 tag : NfcDesfire.exe

This application is launched twice from the “DesfireFormatForm” form.

- ✓ `ProcessStartInfo info = new ProcessStartInfo("NFCDesfire.exe", parameters);`
 - The first call erases the card, provided the given keys are correct
 - The second call creates the CC File and the NDEF File

The main functions used by NfcDesfire.exe come from the pcsc_desfire.dll dll:

- ✓ `FormatPICC`
- ✓ `CreateIsoApplication`
- ✓ `SelectApplication`
- ✓ `CreateIsoStdDataFile`

How to read an NFC Tag

First thing to do: recognize the type of card

- ✓ This is done in the `NfcTag.Recognize(...)` method
 - Check the ATR of the card to determine if it can be a Type 2
 - If not, check if it is a Type 4
 - If not, check if it is a DESFire EV1 that can be formatted

Once the ATR is analyzed, the `NfcTag` object is created

- ✓ It is entirely read (override method “`Read()`” in `NfcTagType2` and `NfcTagType4`)
- ✓ Then, the content is parsed to determine the different NDEFs
 - For Type2 tags, the `ParseUserData(...)` method parses the content into TLVs
 - Then, the `Ndef.Parse(byte[] buffer)` static method parses the content into Ndef objects

At the end, the first valid Ndef object found is printed in the corresponding screen (SmartPoster, Vcard, URI, etc...)

Other examples provided in the SDK

Unit. tests

✓ **SAMPLES/C/REFERENCE**

- Various utilities, written in ANSI C, to perform the unitary tests of our products / libraries
- Use Microsoft Visual C++ 6 (Visual Studio 98) to build them

NFC Tags in command line

✓ **SAMPLES/C/NFCTOOLS**

- Creates NFC Forum Tags (only type 2 and type 4 on Desfire EV1 supported) from the command line
- Use Microsoft Visual C++ Express 2010 to open and build the project

PC/SC Monitor

✓ SAMPLES/C/PCSCMON

- **pcscmon** tracks every PC/SC reader connected to the computer, and traces the insertion/removal of cards
- This is a derivative work from pcsc_scan
<http://ludovic.rousseau.free.fr/software/pcsc-tools/>
and as though distributed under the GPL license. **SpringCard has no link with the writer of this project. Please observe the specific license policy.**

SmartCard APDU from the command line

✓ **SAMPLES/C/SMACADU**

- Same idea as csSriptor but in pure C
- This is an open-source project, provided for convenience only.
SpringCard has no link with the writer of this project. Please observe the specific license policy.

PC/SC Diagnostic for .NET

✓ **SAMPLES/DOTNET/PCSCDIAG2**

- Handy tool to check the installation of the readers, and to perform 'quick and dirty' tests in no time: send APDUs to a card, send Control commands to a reader.
- Use #Develop 4 to open and build the project

Get UID

- ✓ **SAMPLES/DOTNET/VBGETUID**
 - Show how to communicate with PC/SC readers and cards from VB.NET
 - Use Microsoft Visual Basic Express 2010 to open and build the project

vCard printing and encoding

✓ **SAMPLES/DOTNET/ZENIUSVCARD**

- Creates your electronic business cards (vCard on NFC Forum Tags) using an Evolis Zenius printer and the integrated SpringCard CrazyWriter or CrazyWriter HSP
- Demonstrates how to synchronize the contactless encoding with the printing and the moves of the card in the printer's path
- Use #Develop 4 to open and build the project

PC/SC Diagnostic for Win32

✓ **SAMPLES/WIN32/PCSCDIAG**

- Handy tool to check the installation of the readers, and to perform 'quick and dirty' tests in no time: send APDUs to a card, send Control commands to a reader.
- Use Microsoft Visual C++ 6 (Visual Studio 98) to open and build the project (needs MFC and VS 6 runtime)

Java PC/SC applet

✓ SAMPLES/JAVA/JPCSCAPPLET

- This applet acts as a 'bridge' between JavaScript and PC/SC. This makes it possible for a web page to communicate with the readers and cards (see www.nfcwizard.com for a live demo of an advanced version of this applet!)
- No IDE – use java compiler from the command line
- The applet must be signed to be allowed to access the readers from a web page running in the browser (look for *Verisign Code Signing Certificate for Java* on the web)

Java PC/SC monitor

- ✓ **SAMPLES/JAVA/JPCSCMON**
 - Same as PC/SC Monitor but in Java
 - No IDE – use java compiler from the command line

Going further

Interesting articles on CodeProjects

<http://www.codeproject.com/Articles/23018/How-to-access-SmartCards-simply-and-effectively>

<http://www.codeproject.com/Articles/16653/A-Smart-Card-Framework-for-NET>

<http://www.codeproject.com/Articles/17013/Smart-Card-Framework-for-NET>

springcard®

www.springcard.com

DISCLAIMER

This document is provided for informational purposes only and shall not be construed as a commercial offer, a license, an advisory, fiduciary or professional relationship between Pro-Active and you. No information provided in this document shall be considered a substitute for your independent investigation.

The information provided in document may be related to products or services that are not available in your country.

This document is provided "as is" and without warranty of any kind to the extent allowed by the applicable law. While PRO ACTIVE will use reasonable efforts to provide reliable information, we don't warrant that this document is free of inaccuracies, errors and/or omissions, or that its content is appropriate for your particular use or up to date. PRO ACTIVE reserves the right to change the information at any time without notice.

PRO ACTIVE does not warrant any results derived from the use of the products described in this document. PRO ACTIVE will not be liable for any indirect, consequential or incidental damages, including but not limited to lost profits or revenues, business interruption, loss of data arising out of or in connection with the use, inability to use or reliance on any product (either hardware or software) described in this document.

These products are not designed for use in life support appliances, devices, or systems where malfunction of these product may result in personal injury. PRO ACTIVE customers using or selling these products for use in such applications do so on their own risk and agree to fully indemnify PRO ACTIVE for any damages resulting from such improper use or sale.

COPYRIGHT NOTICE

SPRINGCARD, the SPRINGCARD logo, PRO ACTIVE and the PRO ACTIVE logo are registered trademarks of PRO ACTIVE SAS.

All other trademarks are property of their respective owners.

Information in this document is subject to change without notice. Reproduction without written permission of PRO ACTIVE is forbidden.

All information in this document is either public information or is the intellectual property of PRO ACTIVE and/or its suppliers or partners.

You are free to view and print this document for your own use only. Those rights granted to you constitute a license and not a transfer of title : you may not remove this copyright notice nor the proprietary notices contained in this documents, and you are not allowed to publish or reproduce this document, either on the web or by any mean, without written permission of PRO ACTIVE.

Copyright © PRO ACTIVE SAS 2013, all rights reserved.

EDITOR'S INFORMATION

Published by **PRO ACTIVE SAS** company with a capital of 227 000 €

RCS EVRY B 429 665 482

NAF 722C

VAT# : FR 27 429 665 482