Ludovic Rousseau's blog

My activities related to smart card and Free Software (as in free speech).

Friday, November 19, 2010

PCSC sample in C#

Here is the PCSC sample in C# language I promised in PC/SC sample in different languages.

Available wrappers

After searching I found different projects to wrapper PC/SC from C#.

- pcsc-sharp (update 5 Nov 2015: the project moved to github project pcsc-sharp)

  Please note that my C# wrappers classes are not well tested yet. There are more mature projects available (e.g at SmartcardFmwk) - I am not sure if they work with Linux though.
  (Windows uses LLP64 -> sizeof(long) = 4bytes, Linux uses LP64 -> sizeof(long) = 8bytes. This is a problem if you handle with P/Invoke to access external/native libraries and call functions having 'long' data types as parameters).

- Smart Card Framework for .NET
- MonoPcsc
- pcsc-sharp referenced from the Mono library collection.

So much projects for the same service gives different messages:

- the service is needed by "many" users
- the C# community is fragmented and do not have a central point of discussion (forum, mailing list, web site, etc.) to setup on just one implementation?
- the authors suffer from the NIH syndrome?

Installation

Prerequisite

Install the Mono c# compiler:

```
sudo aptitude install mono-gmcs
```

I recommend you to use the nice working development IDE, monodevelop:

```
sudo aptitude install monodevelop
```

Build pcsc-sharp

1. Download pscs-sharp, current version is 2010-11-10 (update 5 Nov 2015: the project moved to github project pcsc-sharp and the current version is 3.3.0.0 from Oct 2015)
2. Unpack the source.
3. Go into the `pcsc-sharp/` directory and simply run `make`.
   It will compile the PC/SC classes within a second. By default, a "Release" version without debug information will be build. You can change the configuration target by editing the `Makefile` file. If you do not like the command line, you can use 'monodevelop', the graphical IDE as well. Monodevelop uses Visual Studio's file format for its solution and project files.

Create the HelloWorld application with Monodevelop

1. Start monodevelop, click at File -> New -> Solution
2. Select C#" as programming language, use the "Console Project" template and name it "HelloWorld". You can skip the package feature dialog.
3. You need to add a reference to the `pcsc-sharp.dll` file.
   To do this right-click at "References" in the Solution panel and choose "Edit References". Click at ".Net Assembly" and browse to the path where the `pcsc-sharp.dll` file is located. Double click the dll and it will be added to the project.
4. Use the `HelloWorld.cs` code listed below.

Google+ Badge

Ludovic Rousseau blog
G+ Follow

Blog Archive

- ► 2017 (33)
- ► 2016 (49)
- ► 2015 (51)
- ► 2014 (61)
- ► 2013 (38)
- ► 2012 (27)
- ► 2011 (46)
- ▼ 2010 (55)
  - ► December (5)
  - ▼ November (5)
    - PCSC sample in C#
    - PSSI: SIM card phone book listing (part 2)
    - PC/SC client and server on two different hosts
    - SIM card phone book listing
    - Debian Miniconf Paris/2010
  - ► October (9)
  - ► September (1)
  - ► August (8)
  - ► July (1)
  - ► June (10)
  - ► May (6)
  - ► April (10)

Search This Blog

Search

Subscribe To

Posts

Comments

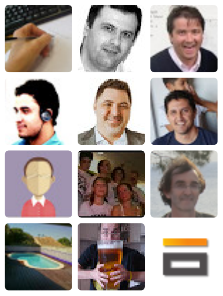
Google+ Followers

Source code

```
using System;
using System.Text;

using PCSC;

namespace HelloWorld
{
    class Program
    {
        static void CheckErr(SCardError err)
        {
            if (err != SCardError.Success)
                throw new PCSCException(err,
                    SCardHelper.StringifyError(err));
        }
        static void Main(string[] args)
        {
            try
            {
                // Establish SCard context
                SCardContext hContext = new SCardContext();
                hContext.Establish(SCardScope.System);

                // Retrieve the list of Smartcard readers
                string[] szReaders = hContext.GetReaders();
                if (szReaders.Length <= 0)
                    throw new PCSCException(SCardError.NoReadersAvailable,
                        "Could not find any Smartcard reader.");

                Console.WriteLine("reader name: " + szReaders[0]);

                // Create a reader object using the existing context
                SCardReader reader = new SCardReader(hContext);

                // Connect to the card
                SCardError err = reader.Connect(szReaders[0],
                    SCardShareMode.Shared,
                    SCardProtocol.T0 | SCardProtocol.T1);
                CheckErr(err);

                long pioSendPci;
                switch (reader.ActiveProtocol)
                {
                    case SCardProtocol.T0:
                        pioSendPci = SCardPCI.T0;
                        break;
                    case SCardProtocol.T1:
                        pioSendPci = SCardPCI.T1;
                        break;
                    default:
                        throw new PCSCException(SCardError.ProtocolMismatch,
                            "Protocol not supported: "
                            + reader.ActiveProtocol.ToString());
                }

                byte[] pbRecvBuffer = new byte[256];

                // Send SELECT command
                byte[] cmd1 = new byte[] { 0x00, 0xA4, 0x04, 0x00, 0x0A, 0xA0,
                    0x00, 0x00, 0x00, 0x62, 0x03, 0x01, 0x0C, 0x06, 0x01 };
                err = reader.Transmit(pioSendPci, cmd1, ref pbRecvBuffer);
                CheckErr(err);

                Console.Write("response: ");
                for (int i = 0; i < pbRecvBuffer.Length; i++)
                    Console.Write("{0:X2} ", pbRecvBuffer[i]);
                Console.WriteLine();
```

Ludovic Rousseau b...

Follow



336 have us in circles

View all

```
                pbRecvBuffer = new byte[256];

                // Send test command
                byte[] cmd2 = new byte[] { 0x00, 0x00, 0x00, 0x00 };
                err = reader.Transmit(pioSendPci, cmd2, ref pbRecvBuffer);
                CheckErr(err);

                Console.Write("response: ");
                for (int i = 0; i < pbRecvBuffer.Length; i++)
                    Console.Write("{0:X2} ", pbRecvBuffer[i]);
                Console.WriteLine();

                hContext.Release();
            }
            catch (PCSCException ex)
            {
                Console.WriteLine("Ouch: "
                    + ex.Message
                    + " (" + ex.SCardError.ToString() + ")");
            }
        }
    }
}
```

Output

```
$ ./HelloWorld.exe
reader name: Gemalto GemPC Twin 00 00
response: 90 00
response: 48 65 6C 6C 6F 20 77 6F 72 6C 64 21 90 00
```

I don't know how to convert a byte array of ASCII characters to a string. I search a bit for a "%c" equivalent in C# but have not found it. So, exercise for next time: display the "string" returned by the card.

Lessons learned

Monodevelop

Monodevelop is a nice tool.

I got caught by a strange (for me) behavior of monodevelop. The HelloWorld project embark/copy its own version of `pcsc-sharp.dll` the PCSC wrapper in the `bin/Debug/` directory. So if you modify the wrapper you need to **rebuild** the HelloWorld project, not just rerun it.

Maybe it is possible to install the DLL in a system directory or something like that. So that different applications can share the same file.
The `HelloWorld.exe` file is 5120 bytes, or 5 kiB.
The `pcsc-sharp.dll` file is 83456 bytes or 81 kiB.

C#

A C# program can be executed directly from the shell on my GNU/Linux system. It is surprising since it is recognised as a Windows binary:

```
$ file HelloWorld.exe
HelloWorld.exe: PE32 executable for MS Windows (console) Intel 80386 32-bit Mono/.Net assembly
```

I also tried to execute on Windows XP the binary generated on Gnu/Linux. And it works! No change needed.

Thanks

Thanks to Daniel Mueller, author of pcsc-sharp, for writing the sample code and a large part of the documentation included in this article.

Update 5 Nov 2015

The project moved to github project pcsc-sharp. It has been active in 2015.

I have not checked my sample code still work with the latest version of the wrapper.

G+

Labels: code, pcsc-lite

Newer Post Home Older Post

Bitcoin

License: by-nc-sa