PMD15240-AA
DRAFT - PUBLIC

USING SPRINGCARD PC/SC COUPLERS WITH ANDROID

Getting Started and Development Guide

www.springcard.com

DOCUMENT IDENTIFICATION

Category	Developer's manual		
Family/Customer	PC/SC Couplers		
Reference	PMD15240	Version	AA
Status	Draft	Classification	Public
Keywords	Android, APK, PC/SC, Prox'N'Roll, H663, CrazyWriter, TwistyWriter, CSB		
Abstract			

File name	V:\Dossiers\SpringCard\A-Etudes Soft\PCSC for Android\Documentation\[PMD15240-AA] Using SpringCard PCSC Readers in Android.odt		
Date saved	16/06/15	Date printed	15/06/15

REVISION HISTORY

Ver.	Date	Author	Valid. by		Approv. by	Details
			Tech.	Qual.		
AA	29/05/15	MBA				Creation

Contents

1. INTRODUCTION

1.1. ABSTRACT

SpringCard offers a free software solution to support its **USB PC/SC Couplers** on an Android tablet.

The solution is made of

- **SpringCard PC/SC Service for Android**, a freeware application that installs from Google Play,
- **SpringCard PC/SC Library for Android**, an open-source library that makes it easy for application developers to communicate through the Service with contactless cards and NFC/RFID tags activated by a **SpringCard PC/SC coupler**.

This explains the software architecture and shows how-to get started developing with the Service and the Library.

Furthermore, SpringCard provides also a PC/SC Demo application. Tips and hints are provided to re-compile the Demo application, and/or retrieve useful pieces of software code and include them into your own project.

1.2. SUPPORTED PRODUCTS

At the time of writing, this document refers to all **SpringCard USB PC/SC Couplers** in the CSB6, and H663 groups, namely:

- **SpringCard** Prox'N'Roll PC/SC,
- **SpringCard** Prox'N'Roll PC/SC HSP, TwistyWriter HSP, CrazyWriter HSP, CSB HSP.

Please note that the SpringCard PC/SC Service for Android and its companion Library provide support for the readers' contactless (NFC/RFID) slot only.
Using the SAM or smartcard slots of TwistyWriter HSP, CrazyWriter HSP or CSB HSP is out of the scope of this free software suite. Don't hesitate to contact us should you need a custom version to support all the slots of your SpringCard coupler.

1.3. AUDIENCE

This manual is designed for use by application developers. It assumes that the reader has expert knowledge of Android development and a basic knowledge of PC/SC and/or of the ISO 7816-4 standard for smartcards.

1.4. USEFUL LINKS

- PC/SC workgroup: http://www.pcscworkgroup.com
- Using a PC/SC reader (and card) in a Java environment thanks to the **javax.smartcardio** package: http://docs.oracle.com/javase/7/docs/jre/api/security/smartcardio/spec/javax/smartcardio/package-summary.html
- USB CCID specification: http://www.usb.org/developers/docs/devclass_docs/DWG_Smart-Card_CCID_Rev110.pdf

1.5. SUPPORT AND UPDATES

All Android-related information and material will be centralized under the tag 'Android' in SpringCard Tech Zone:

http://tech.springcard.com/tags/android

Useful related materials (product datasheets, application notes, sample software, HOWTOs and FAQs…) are available at SpringCard's web site:

www.springcard.com

Updated versions of this document and others are posted on this web site as soon as they are available.

For technical support enquiries, please refer to SpringCard support page, on the web at

www.springcard.com/support

2. QUICKSTART: TESTING YOUR SPRINGCARD READER WITH YOUR ANDROID DEVICE

SpringCard USB PC/SC Couplers, as the name says, are USB devices. To use them, the host system (aka the computer -or tablet) must be a fully-compliant USB host.

This feature is included in the system since Android 3.1, but not supported by every hardware running the system. Unfortunately, there is no reliable software-only solution to check whether a given Android device is a fully-compliant USB host, or not.

Therefore, before starting a long journey into Android development and debugging, it is important to try both **SpringCard PC/SC Service for Android** with its Demo application on your very device, just to make sure the hardware and the system software stack are OK.

2.1. REQUIREMENTS

To test your SpringCard PC/SC coupler with your Android device, you'll need

- A **SpringCard PC/SC Coupler**. It the following pages, we'll be using a **SpringCard Prox'N'Roll PC/SC HSP**, but everything will work identically with any other device based on the CSB6 or H663 core.
- An **Android tablet** running *(theoretically)* Android version ≥ 3.1. We've tested the solution only with Android 5.0 and 5.1 on a Nexus 7 (2013) and a Nexus 9 (2014), so the behaviour with earlier version could be different.
- A **USB adapter** (USB micro B/Male to USB A/Female) in case your tablet doesn't feature a USB A/Female port.

Be aware that the PC/SC Coupler will takes its power from the USB host, i.e. from the tablet. On most today's tablets, the battery is reloaded through a USB micro B/Female port, which is also the one we use to connect the Coupler.
As a consequence, it is impossible to keep the tablet powered while working with a USB device such as a SpringCard PC/SC Coupler.
If you're designing a kiosk or unattended solution which shall run 24/7, choose your tablet carefully; it must provide a mean to be externaly-powered even with a USB device is attached.

2.2. INSTALLING THE APPLICATIONS

2.2.1. SpringCard USB PC/SC Service

Install the **SpringCard USB PC/SC Service** package from Google Play Store.

Direct link:

https://play.google.com/store/apps/details?id=com.springcard.android_usb_pcsc.service

2.2.2. SpringCard USB PC/SC Demo application

Install the **SpringCard USB PC/SC Demo** package from Google Play Store.

Direct link:

https://play.google.com/store/apps/details?id=com.springcard.springcardpcscdemo

2.3. CONNECTING A COUPLER TO YOUR ANDROID TABLET

Ensure your Android tablet has enough battery to power the reader.

Connect your reader using a Micro USB B/Male to USB2.0 A/Female adapter:

(1) Android device

(2) USB device (Prox'N'Roll HSP)

The coupler is powered but not active (on Prox'N'Roll: LEDs blink yellow) and the Android system prompts to choose the application that will handle the newly connected USB device.

Select “SpringCard USB PCSC Service” and click the “Always” option.

Once done, the system activates the coupler (on Prox'N'Roll: smooth blue LED pulses).

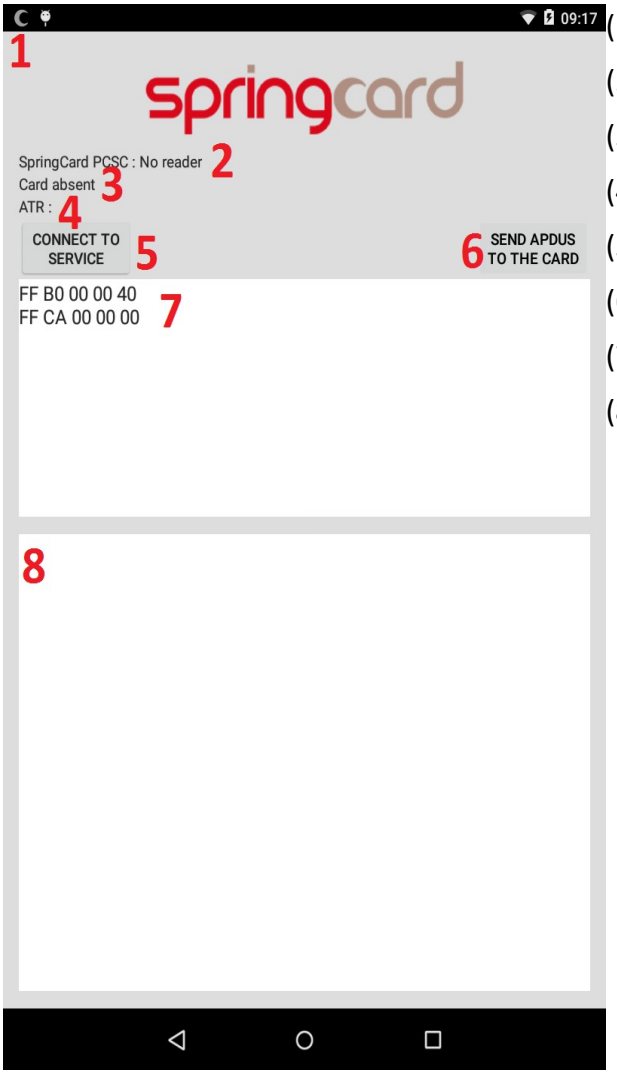
Troubleshooting:

If all coupler's LEDs stay OFF and nothing happen on the tablet's screen

- maybe the adapter cable is broken
- maybe the Android tablet doesn't feature the USB host hardware or software stack.

2.4. THE USB SPRINGCARD PC/SC DEMO APPLICATION

This application is no more than a basic interface to exchange APDUs with a contactless smartcard, or to the coupler's embedded interpreter.



(1) Service's notification icon

(2) Reader's product name

(3) Card's presence or not

(4) ATR retrieve from the card

(5) Connect to SpringCard USB PCSC Service

(6) Send APDU command to the card

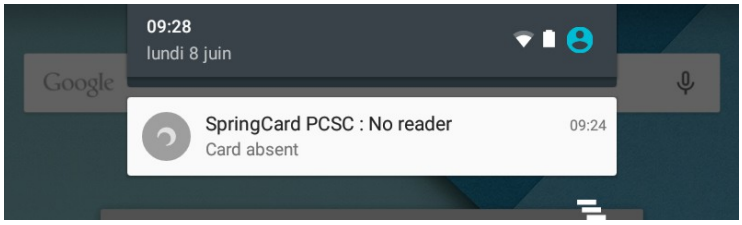
(7) Command list (one per line) to send to the card

(8) Command result log

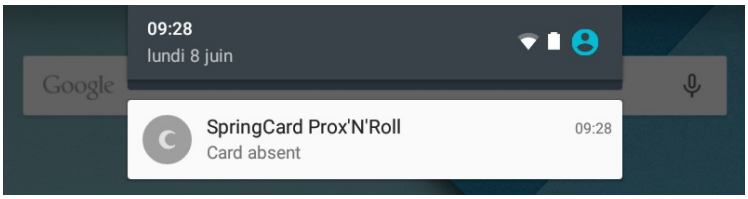
2.5. THE USB SPRINGCARD PC/SC SERVICE

The Service shows its state in the notification bar.

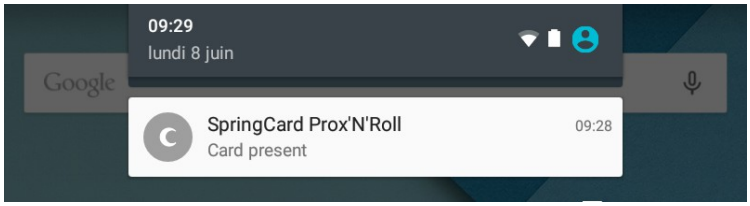
No Coupler connected:



Prox'N'Roll PC/SC Coupler connected to the tablet:

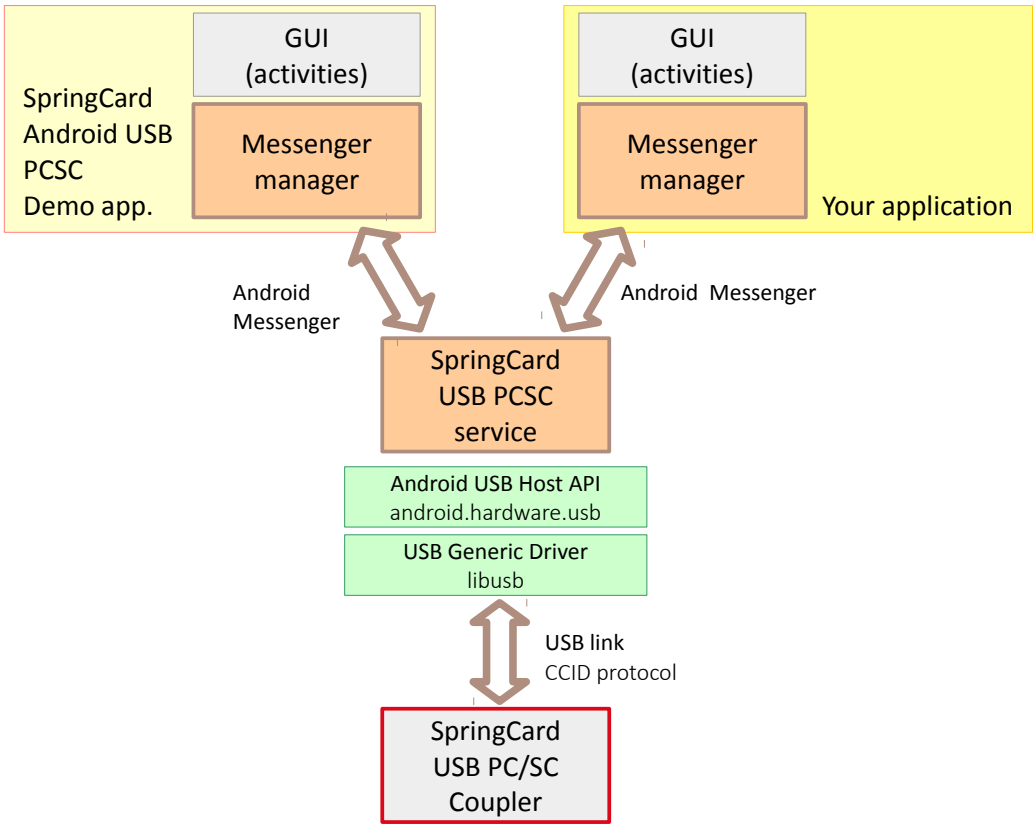


Contactless card, NFC or RFID tag present on the Coupler's antenna:



3. ARCHITECTURE OF SPRINGCARD USB PCSC FOR ANDROID

3.1. OVERALL PICTURE



3.2. THE SPRINGCARD ANDROID USB PC/SC SERVICE FOR ANDROID

- Is responsible of the USB communication between an Android device and SpringCard's PCSC compliant readers.
- Contains a list of compliant readers and will ask the user for relevant access rights.
- Has full access to those readers.

3.3. THE SPRINGCARD ANDROID USB PC/SC LIBRARY FOR ANDROID

- Communicate with the running service using Android's Messenger system only.

4. REBUILDING THE DEMO APPLICATION IN ANDROID STUDIO

4.1. REQUIREMENTS

Nothing is needed to build the demo application, but your must have the **SpringCard Android USB PC/SC Service** running on the Android device you'll be using for debugging and tests.

Download and install Android Studio on your computer: https://developer.android.com/sdk/

4.2. RETRIEVING THE PROJECT FROM GIT HUB

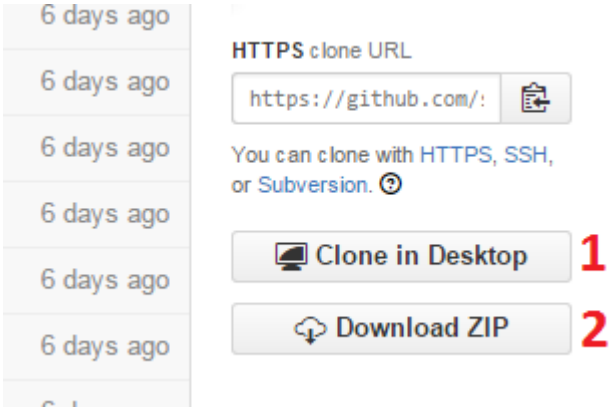
Method 1:

Use GIT to clone the project from GitHub:

```
git clone https://github.com/springcard/springcard.pcsc-android.sdk.git
```

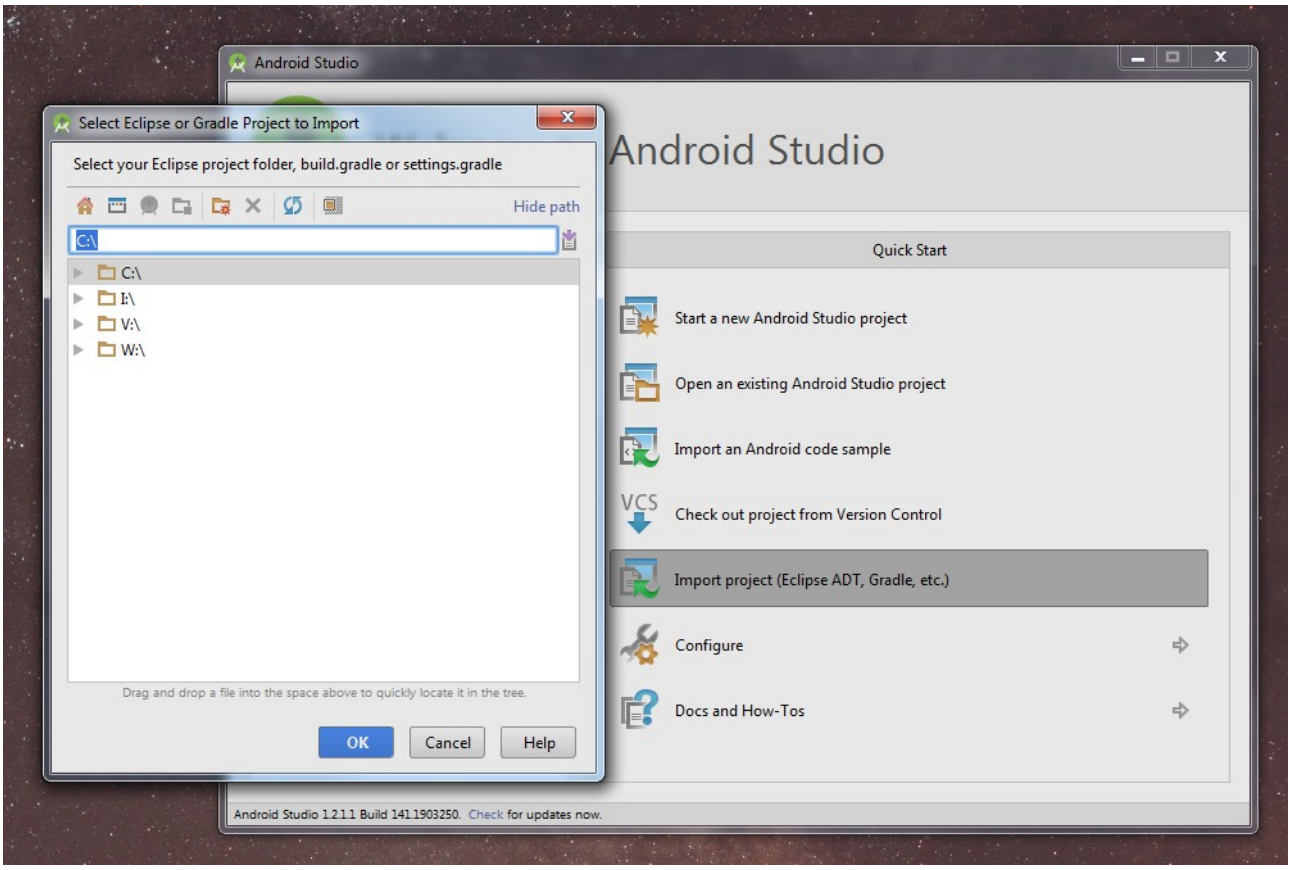
Method 2:

Download a ZIP archive of the project at:
https://github.com/springcard/springcard.pcsc-android.sdk/archive/master.zip



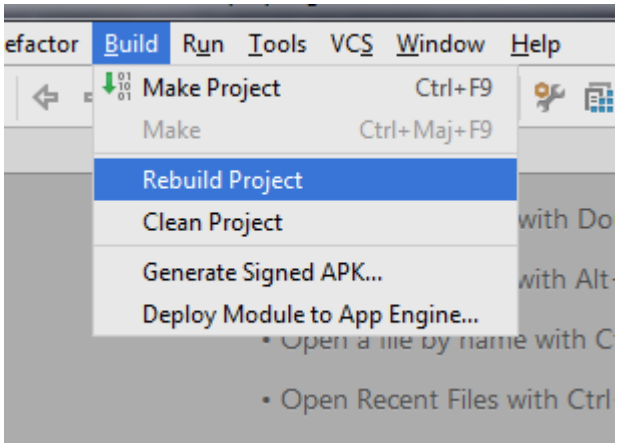
4.3. IMPORTING THE PROJECT IN ANDROID STUDIO

Launch Android Studio and import the project (select the folder where you have extracted the archive):



4.4. COMPILING THE PROJECT

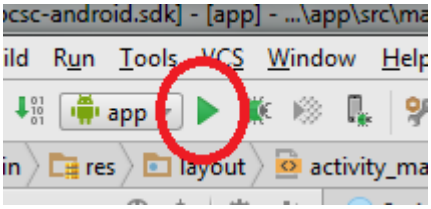
In Android Studio, select menu “Build”, option “Rebuild Project”:



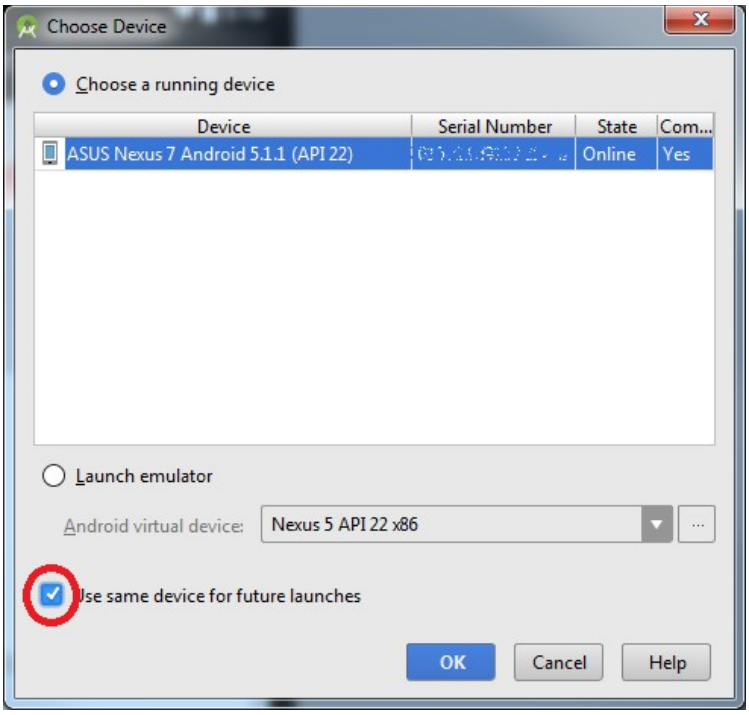
4.5. RUNNING THE PROJECT

Connect a compliant Android device to your computer (using an USB wire or using Wireless network). Your device must be in Developer mode.

Push the “run” button in Android studio:



Select your device from the list (an option is available to remember your choice):



5. DEVELOPING YOUR OWN APPLICATION USING SPRINGCARD USB PC/SC

5.1. ABSTRACT

The **SpringCard USB PC/SC Service** for Android runs in the background.

Any application that wants to communicate with a contactless card or a RFID/NFC tag through a **SpringCard PC/SC Coupler** will be bound to the Service and will communicate with it using Android Messenger sub-system.

All details regarding the Service / Messenger scheme are documented at
http://developer.android.com/guide/components/bound-services.html

Multiple application could bind the Service but only the one which is active in the foreground will receive answers from the Service.

The **SpringCard USB PC/SC Library** for Android is a set of Java files that provides a "high level" interface to the Service, using function calls inspired by the javax.smartcardio package.

5.2. IMPORTING THE LIBRARY (AS SOURCE CODE) INTO YOUR ANDROID STUDIO PROJECT

To be written

5.3. USING THE LIBRARY

In your application's source files, add the following import

```
import com.springcard.android_usb_pcsc.client
```

Your main activity shall provide the **SpringCardPCSC.Client** interface.

```
public class MainActivity
      extends Activity
      implements SpringCardPCSC.Client
```

Then your main activity shall check that the Service is installed, and then bind to the Service.

5.3.1. Is the Service installed?

The static method

```
static bool SpringCardPCSC.isServiceAvailable(Context context)
```

returns **`true`** if the Service is installed, and **`false`** otherwise.

You should instruct the user to install the Service's package if the Service is not installed.

5.3.2. Bind to the Service

The static method

```
static void SpringCardPCSC.bindToService(SpringCardPCSC.Client activity)
```

binds your application to the **SpringCard USB PC/SC Service** for Android. An exception is raised if the Library fails to bind the application to the Service.

From now on, any event that occurs on the Service-side (Coupler connected/removed from the system, card inserted/removed over the Coupler's antenna...) will be notified to the application through one of the callback methods defined within the **SpringCardPCSC.Client** interface.

5.3.3. Releasing the service when leaving

Don't forget to un-bind the application from the Service in the application's onDestroy event by calling this static method:

```
static void SpringCardPCSC.unbindService()
```

5.4. NOTIFICATION EVENTS

5.4.1. onTerminalInstalled

Your main activity (which implements the **SpringCardPCSC.Client** interface) implements the following method:

```
void onTerminalInstalled(SpringCardPCSC.CardTerminal terminal)
{
      /* A Coupler has been added to the system */
}
```

This method is invoked by the Library

- when starting, if a Coupler is already attached to the system
- every-time a new Coupler is attached to the system.

Use the methods of the **SpringCardPCSC.CardTerminal** object to retrieve

- the Coupler's name (**getName** method),
- whether there's a Card in the Coupler or not (**isCardPresent** method).

5.4.2. onTerminalRemoved

Your main activity (which implements the **SpringCardPCSC.Client** interface) implements the following method:

```
void onTerminalRemoved(SpringCardPCSC.CardTerminal terminal)
{
	/* A Coupler has been removed from the system */
}
```

This method is invoked by the Library every-time a Coupler is removed from the system.

When this callback is invoked, the application shall suppress any reference it owns to the CardTerminal object. Any copy of the CardTerminal object becomes invalid after return.

5.4.3. onCardInserted

Your main activity (which implements the **SpringCardPCSC.Client** interface) implements the following method:

```
void onCardInserted(SpringCardPCSC.CardTerminal terminal)
{
	/* A Card has been inserted in the Coupler */
}
```

This method is invoked by the Library

- when starting, if a Card is already present in the Coupler
- every-time a new Card is inserted into any Coupler attached to the system.

Use the **CardTerminal.connect** object to open a connection to the Card. You may then retrieve

- the Card's Answer To Reset (**Card.getATR** method),
- the **SpringCardPCSC.CardChannel** object that will be used to exchange APDUs with the card (**Card.getBasicChannel** method).

5.4.4. onCardRemoved

Your main activity (which implements the **SpringCardPCSC.Client** interface) implements the following method:

```
void onCardRemoved(SpringCardPCSC.CardTerminal terminal)
{
	/* A Card has been removed from the Coupler */
}
```

This method is invoked by the Library every-time a Card is removed from any Coupler attached to the system.

When this callback is invoked, the application shall suppress any reference it owns to the Card or CardChannel objects belonging to this CardTerminal. Any copy of these objects becomes invalid after return.

5.5. EXCHANGING APDUS WITH THE CARD

5.5.1. Sending a Command-APDU to the Card

The application communicates with the Card through the CommandAPDU method of a CardChannel object.

To retrieve the CardChannel object, the application shall

- implements the **onCardInserted** method
- when **onCardInserted** is called, retrieve the Card object using the **CardTerminal.connect** method,
- retrieve the CardChannel object using the **Card.getBasicChannel** method.

```
void onCardInserted(SpringCardPCSC.CardTerminal terminal)
{
      SpringCardPCSC.Card card = terminal.connect();
      SpringCardPCSC.CardChannel channel = card.getBasicChannel();
      channel.transmit(new SpringCardPCSC.CommandAPDU("00A404023F00"));
}
```

Observe that the transmit method doesn't return anything. The transmission is done asynchronously, and the callback **onResponseAPDU** is invoked when the Card (finally) answers.

5.5.2. onResponseAPDU - Retrieving the Response-APDU from the Card

Your main activity (which implements the **SpringCardPCSC.Client** interface) implements the following method:

```
void onResponseAPDU(SpringCardPCSC.CardChannel channel,
                    SpringCardPCSC.ResponseAPDU response)
{
      /* The Card has answered */
}
```

If response is null, the Card failed to answer (onCardRemoved will be called immediately afterwards).

6. LICENSE INFORMATION

6.1. SPRINGCARD USB PC/SC SERVICE FOR ANDROID

The **SpringCard USB PC/SC Service** for Android is provided free of charge, but is not an open-source software.

The following license applies:

```
SPRINGCARD SOFTWARE LICENSE AGREEMENT

This software is copyright (c) 2014-2015 SPRINGCARD.
All rights reserved.

SPRINGCARD is a registered trademark of PRO ACTIVE SAS,
France.

Redistribution and use in binary (object code) form, with or
without modification, are permitted provided that the
following conditions are met:

1. Redistributed object code shall be used only in
   conjunction with hardware products manufactured or
   distributed by SPRINGCARD,

2. Redistributed object code must reproduce the above
   copyright notice, this list of conditions and the
   disclaimer below in the documentation and/or other
   materials provided with the distribution,

3. The name of SPRINGCARD may not be used to endorse or
   promote products derived from this software or in any
   other form without specific prior written permission from
   SPRINGCARD.

THIS SOFTWARE IS PROVIDED "AS IS" FREE OF CHARGE.
SPRINGCARD SHALL NOT BE LIABLE FOR INFRINGEMENTS OF THIRD
PARTIES RIGHTS BASED ON THIS SOFTWARE.

ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED.

SPRINGCARD DOES NOT WARRANT THAT THE FUNCTIONS CONTAINED IN
THIS SOFTWARE WILL MEET THE USER'S REQUIREMENTS OR THAT THE
OPERATION OF IT WILL BE UNINTERRUPTED OR ERROR-FREE.

IN NO EVENT, UNLESS REQUIRED BY APPLICABLE LAW, SHALL
SPRINGCARD BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
ALSO, SPRINGCARD IS UNDER NO OBLIGATION TO MAINTAIN, CORRECT,
UPDATE, CHANGE, MODIFY, OR OTHERWISE SUPPORT THIS SOFTWARE.

Contact: www.springcard.com
```

6.2. SPRINGCARD USB PC/SC LIBRARY FOR ANDROID

The **SpringCard USB PC/SC Library** for Android is an open-source software project.

Note that the license forbids to use this software with products not provided by SpringCard.

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 doesn't 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

EDITOR'S INFORMATION

PRO ACTIVE SAS company with a capital of 227 000 €

RCS EVRY B 429 665 482

Parc Gutenberg, 2 voie La Cardon

91120 Palaiseau – FRANCE

CONTACT INFORMATION

For more information and to locate our sales office or distributor in your country or area, please visit

www.springcard.com