PMD15240-AA
- PUBLIC

USING SPRINGCARD PC/SC COUPLERS WITH ANDROID

Getting started guide

www.springcard.com

DOCUMENT IDENTIFICATION

Category	Developer's manual		
Family/Customer	PC/SC Couplers		
Reference	PMD15240	Version	AA
Status		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	09/06/15	Date printed	05/12/12

REVISION HISTORY

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

Contents

1. INTRODUCTION

1.1. ABSTRACT

This document provides a short guide for setting up the development environment to start working with a **SpringCard USB PC/SC Coupler** under Android using the **open source SpringCardPCSC library for Android**.

Furthermore, tips and hints are provided to re-compile the test project and integrate the library to 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, H663 and H512 groups.

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
- USB CCID specification: http://www.usb.org/developers/docs/devclass_docs/DWG_Smart-Card_CCID_Rev110.pdf

1.5. SUPPORT AND UPDATES

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. THE DEMO APPLICATION

2.1. REQUIREMENTS

- SpringCard Prox'N'Roll HSP card reader:

- Micro USB B/Male to USB2.0 A/Female adapter or Micro USB Host OTG Y-Cable with Power (depending on your Android device's type):

- Android 5.1 device with usb-host capability (Tested on Google Nexus 7/2012 and Nexus 9).
- SpringCard Android USB PCSC Service

2.2. INSTALLING THE SPRINGCARD USB PCSC SERVICE FOR ANDROID FROM GOOGLE PLAY STORE

Simply use the following link from your Android device (or on your computer using the same Google account):

http://market.android.com/details?id=com.springcard.springcardpcscdemo

2.3. INSTALLING THE SPRINGCARD ANDROID USB PCSC DEMO APPLICATION FROM GOOGLE PLAY STORE

Simply use the following link from your Android device (or on your computer using the same Google account):

http://market.android.com/details?id=com.springcard.android_usb_pcsc.service

2.4. CONNECTING A SPRINGCARD USB PC/SC COUPLER TO YOUR ANDROID DEVICE

2.4.1. Case A: reader is powered by the Android device

Ensure your Android device 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)

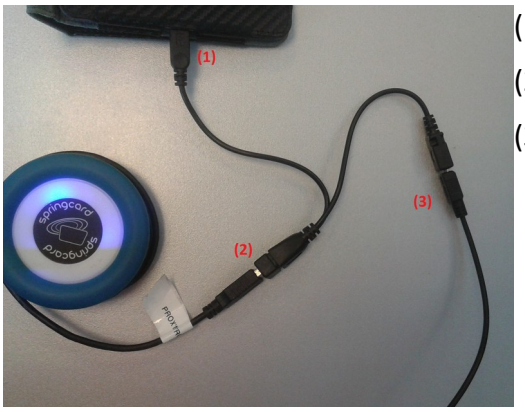
Connected reader will start blinking and the Android system will ask you which application to use to handle this reader. Select “SpringCard USB PCSC Service” and option “Always”. Once done, the reader should start blinking in blue.

No light at all on the reader:

- your reader is not correctly connected

- your Android device is not USB-Host compliant

2.4.2. Case B: Android device support battery charging in usb-host mode

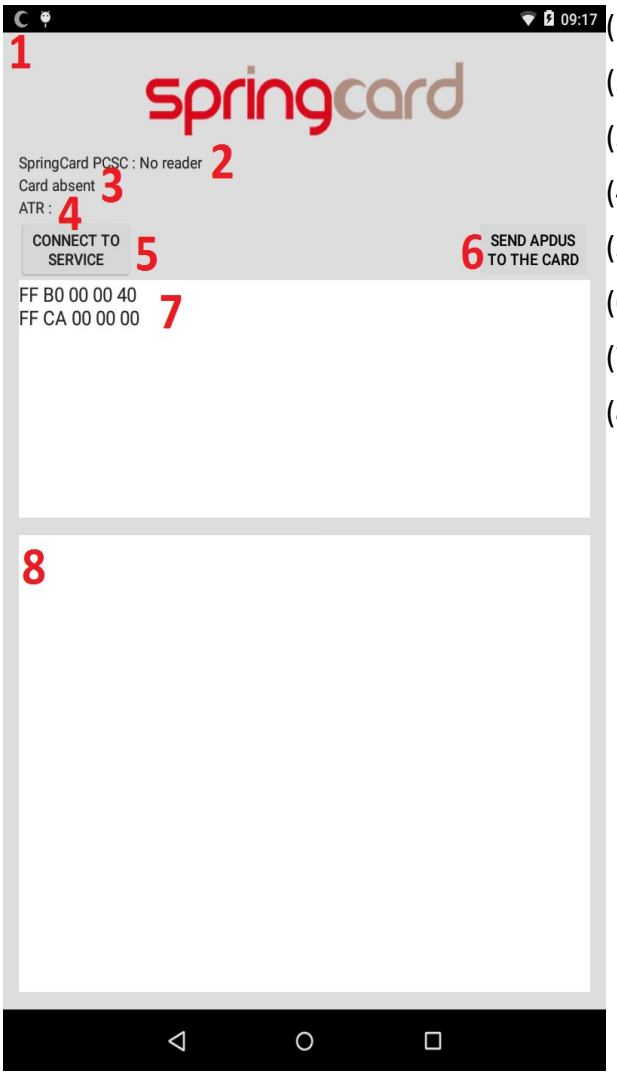


(1) Android device

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

(3) standard power supply

2.5. The application's screen



(1) 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

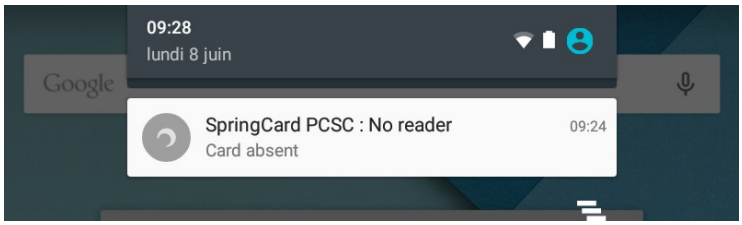
(6) Send APDU command to the card

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

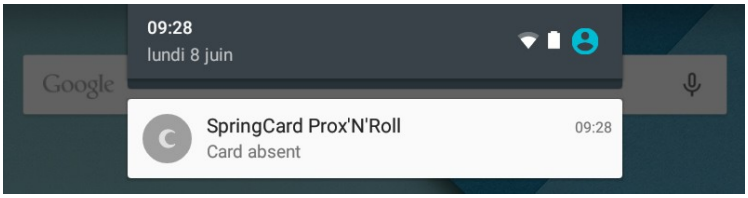
(8) Command result log

2.6. WORKING WITH CONTACTLESS CARDS

Reader's state is visible on the notification bar:

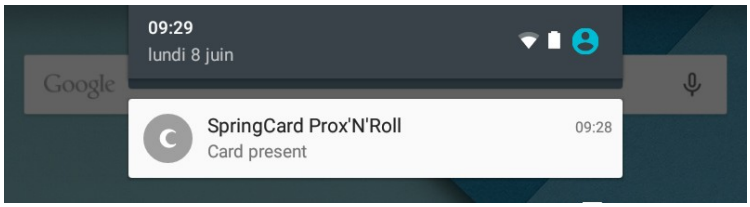


Connect a compliant reader and the product's name should appear on the notification bar:



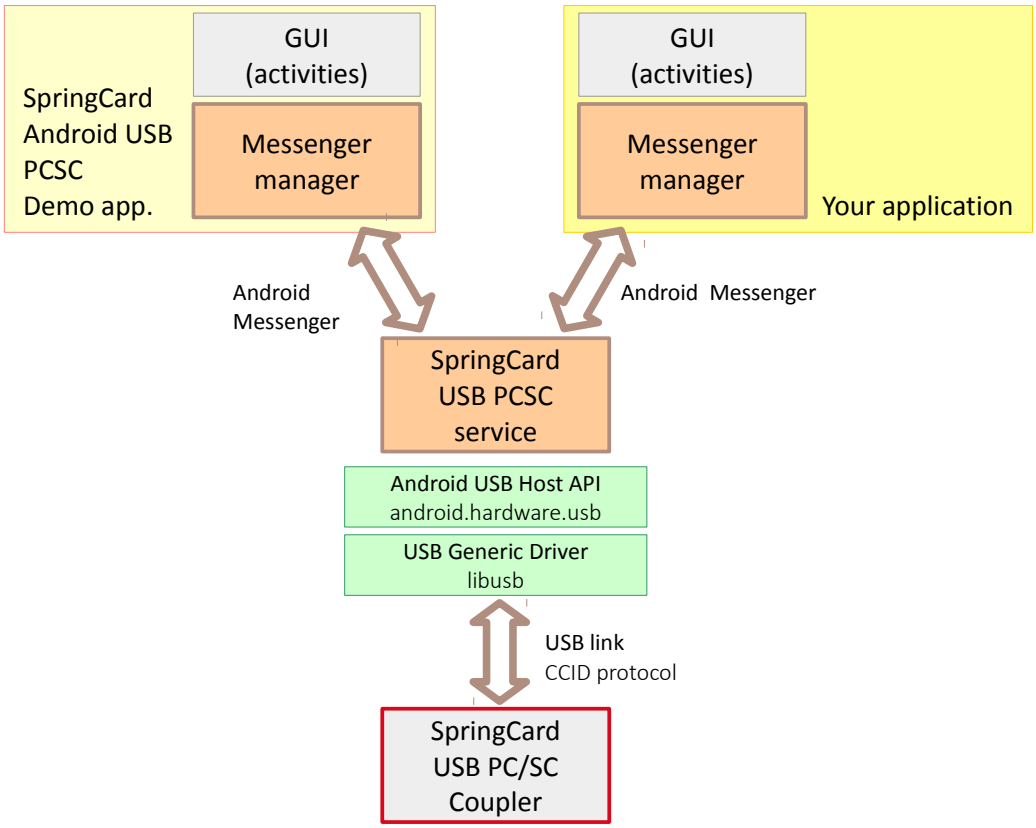
Place a “compliant” card on the reader. The reader should be in continuous green light state while the card remains on it.

Card presence indication could be seen in the notification bar:



3. ARCHITECTURE OF SPRINGCARD USB PCSC FOR ANDROID

3.1. OVERALL PICTURE



3.2. THE SPRINGCARD ANDROID USB PCSC 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 PCSC 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 PCSC service setup on your Android device to use it.

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

4.2. RETRIEVING THE PROJECT FROM GIT HUB

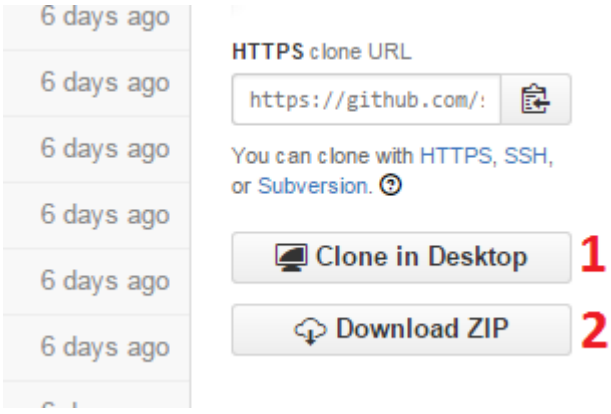
Method 1:

- Download and setup Git Hub Windows software on your computer (https://windows.github.com/).

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

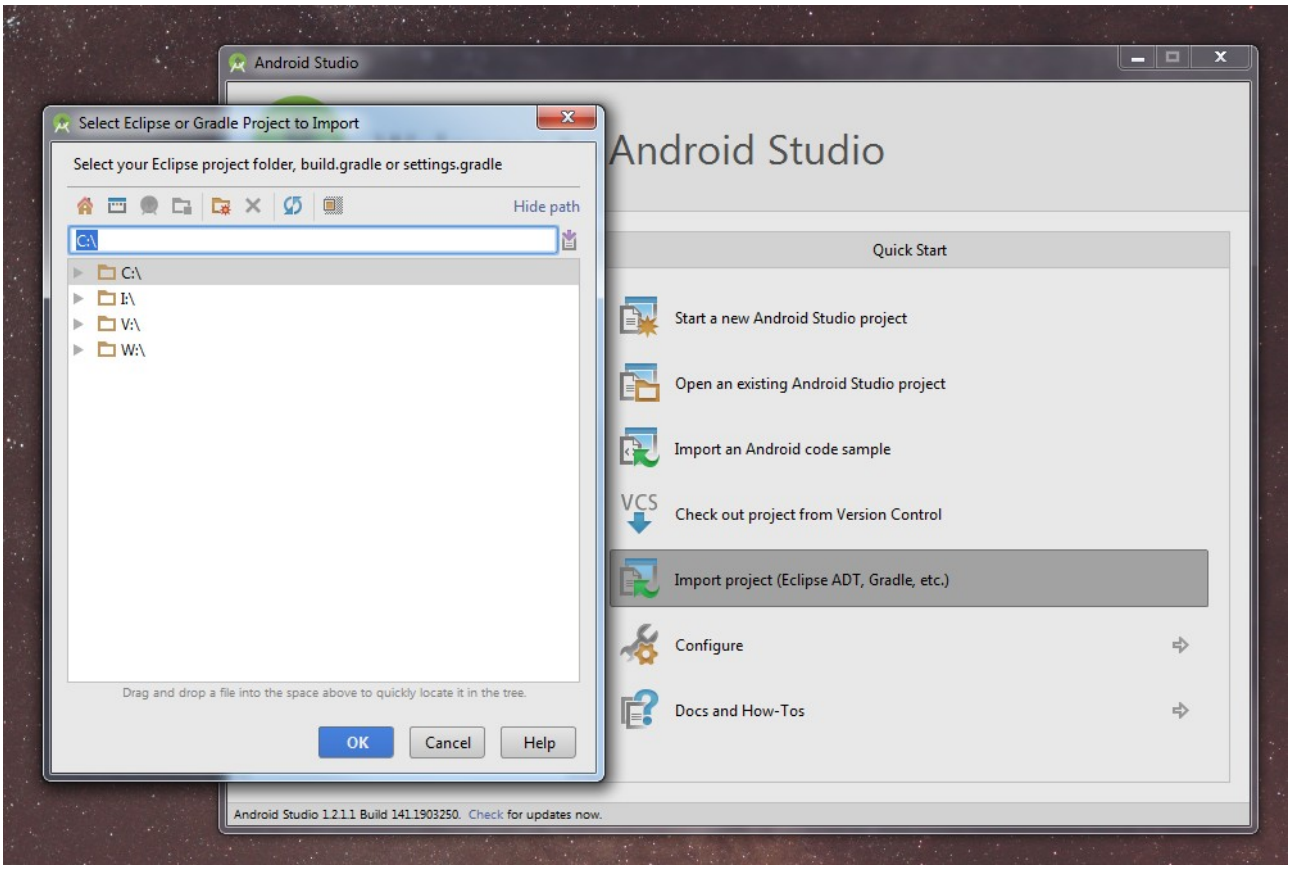
Method 2:

Download current version archive from git hub and extract the content on your computer: https://github.com/springcard/springcard.pcsc-android.sdk/archive/master.zip



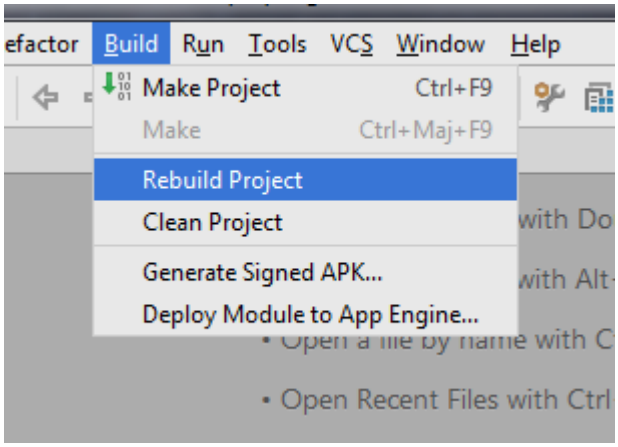
4.3. IMPORTING THE PROJECT IN ANDROID STUDIO

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



4.4. COMPILING THE PROJECT

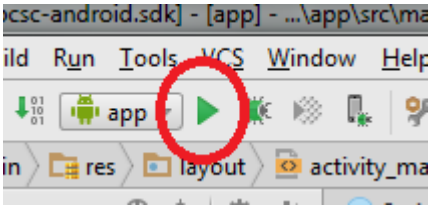
In Android Studio, select menu “Build”, option “Rebuild Project”:



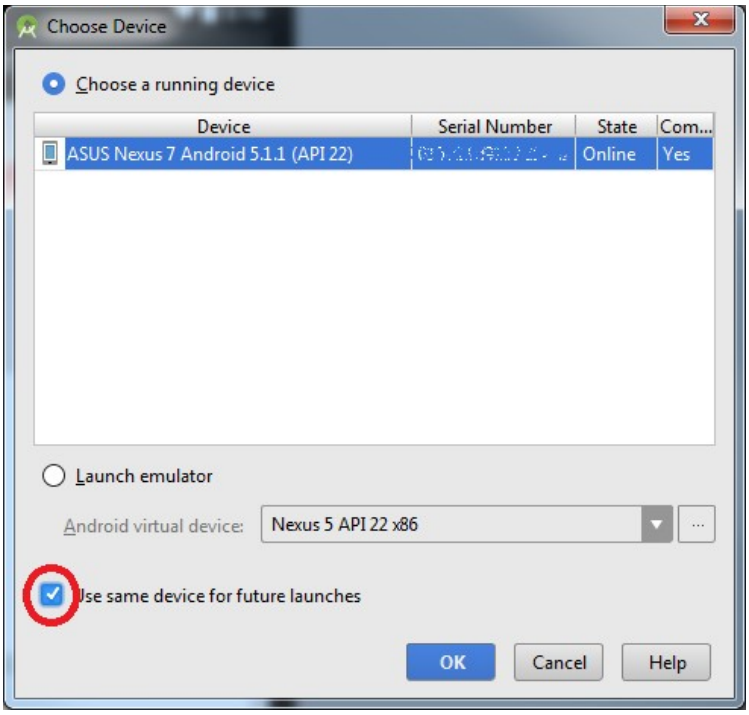
4.5. RUNNING THE PROJECT

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. DEVELOPMENT OF A NEW APPLICATION USING SPRINGCARD USB PCSC

5.1. ABSTRACT

The example application shows how to communicate with SpringCard Android USB PCSC service. SpringCard Android USB PCSC service ask for USB device access so client application doesn't need to use USB feature in the Manifest.xml file.

Client application must bind to the service before starting communication with it. Bind is done using an Intent to the service package name.

Once bound, client application could send queries and receive response (or event) from the service using Android's Messenger.

Multiple application could bind the service but only the last one (Active one) will receive event from it.

5.2. IMPORTING THE LIBRARY SOURCE CODE INTO YOUR PROJECT

All communications between Client and Service are done using Messenger. Client library simply consist of a static class for constant declaration and a service discovery function:

```
package com.springcard.springcardpcscdemo;
import android.content.Context;
import android.content.pm.PackageManager;
/**
 * Created by matthieu.b on 04/06/2015.
 */
public class SpringCardPCSCCli {
 public static final String packageName =
      "com.springcard.android_usb_pcsc.service";
  /* command list */
  public static final int                CMD_READER_STATE              = 1;
  public static final int                CMD_CARD_STATE                = 2;
```

```
  public static final int                 CMD_SEND_APDU                = 3;
  public static final int                 CMD_RECV_RAPDU               = 4;
  public static final int                 CMD_READER_FREE              = 5;

  /* error list */
  public static final int                 ERROR_NO_ERROR               = 10;
  public static final int                 ERROR_NO_READER              = 11;
  public static final int                 ERROR_WRONG_APDU_LENGTH      = 12;
  public static final int                 ERROR_INVALID_APDU           = 13;
  public static final int                 ERROR_CARD_COMMUNICATION     = 14;

  /* check if service is available on this device */
  public static boolean isPackageInstalled(Context context) {
    PackageManager pm = context.getPackageManager();
    try {
      pm.getPackageInfo(packageName, PackageManager.GET_ACTIVITIES);
      return true;
    } catch (PackageManager.NameNotFoundException e) {
      return false;
    }
  }
}
```

5.3. CHECK IF THE SERVICE IS AVAILABLE

This function check if the required service package is present on this system.

```
SpringCardPCSCCli.isPackageInstalled(Context context)
```

True is service is present, false otherwise.

5.4. BINDING THE SERVICES

Please note that client application loose service binding when the main activity is destroyed.

Client application must ensure that service is bound before sending queries to it.

```
/* bind to SpringCard Android USB PCSC service */
Intent intent = new Intent("com.springcard.android_usb_pcsc.service");
intent.setPackage("com.springcard.android_usb_pcsc.service");
bindService(intent, saupConnection, Context.BIND_AUTO_CREATE);
```

saupConnection is a ServiceConnection object and will receive bind/unbind indications.

5.5. ENUMERATING THE COUPLER

Response is automatically sent by the service on status change (only if binding is done and valid).

Ask the service to give coupler's informations:

```
/*****************************/
/* ask for reader information */
Message msg = Message.obtain();
Bundle bundle = new Bundle();
bundle.putInt("command", SpringCardPCSCCli.CMD_READER_STATE);
msg.setData(bundle);
msg.replyTo = localMessenger;
try {
  saupService.send(msg);
} catch (RemoteException e) {
  e.printStackTrace();
}
```

Response Messenger's bundle when a compliant coupler is present:

- error: ERROR_NO_ERROR
- reader: Product's name
- maxSlot: Number of slots available in this product

Response Messenger's bundle when there is no compliant coupler present:

- error: ERROR_NO_READER

- reader: “SpringCard PCSC : No reader”
- maxSlot: 0

5.6. QUERYING THE STATUS OF A COUPLER

Response is automatically sent by the service on status change (only if binding is done and valid).

Ask the service to give card informations:

```
/**********************/
/* ask for card status */
msg = Message.obtain();
bundle = new Bundle();
bundle.putInt("command", SpringCardPCSCCli.CMD_CARD_STATE);
/* using slot 0 (others slots not used currently) */
bundle.putInt("slot", 0);
msg.setData(bundle);
msg.replyTo = localMessenger;
try {
  saupService.send(msg);
} catch (RemoteException e) {
  e.printStackTrace();
}
```

Response Messenger's bundle when a card is present:

- error: ERROR_NO_ERROR
- reader: Product's name
- atr: “ATR : “+Card's ATR
- card: “Card present”

Response Messenger's bundle when there is no card present:

- error: ERROR_NO_ERROR
- reader: Product's name
- atr: “ATR : “
- card: “Card absent”

5.7. CONNECTING TO A SMARTCARD

SmartCard connection/disconnection is done automatically on APDUs exchanges.

5.8. EXCHANGING APDUS WITH THE SMARTCARD

Response is automatically sent by the service after APDU's execution.

```
/* Create a new command to the reader */
Message msg = Message.obtain();
Bundle bundle = new Bundle();
/* specify message type */
bundle.putInt("command", SpringCardPCSCCli.CMD_SEND_APDU);
/* apdu contains an apdu to execute on the card */
bundle.putString("apdu",anApdutoSend);
/* using slot 0 (others slots not used currently) */
bundle.putInt("slot", 0);
/* append data to message */
msg.setData(bundle);
/* specify caller (for callback) */
msg.replyTo = localMessenger;
try {
  saupService.send(msg);
} catch (RemoteException e) {
  e.printStackTrace();
}
```

Response Messenger's bundle when there is a valid APDU answer:

- error: ERROR_NO_ERROR
- rapdu: Response APDU

Response Messenger's bundle when there is an error with an APDU:

- error: ERROR_CARD_COMMUNICATION

5.9. RELEASING THE SMARTCARD AND THE COUPLER

Ask the reader to forgive our client application :

```
/*****************************/
/* free this reader          */
Message msg = Message.obtain();
Bundle bundle = new Bundle();
bundle.putInt("command", SpringCardPCSCCli.CMD_READER_FREE);
msg.setData(bundle);
msg.replyTo = localMessenger;
try {
  saupService.send(msg);
} catch (RemoteException e) {
  e.printStackTrace();
}
```

5.10. MULTI-THREADING & CALLBACKS CONSIDERATIONS

Query and response are asynchronous. Some cards could take time to process an APDU.

6. LICENSE INFORMATION

6.1. SPRINGCARD ANDROID USB PCSC SERVICE

SPRINGCARD SOFTWARE DEVELOPMENT KIT (SDK) LICENSE AGREEMENT

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

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

2. Redistributed source code, either modified or un-modified, must retain the above copyright notice, this list of conditions and the disclaimer below,

3. Redistribution of any modified code must be clearly identified "Code derived from original SPRINGCARD / PRO ACTIVE copyrighted source code", with a description of the modification and the name of its author,

4. 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,

5. The name of SPRINGCARD or PRO ACTIVE 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 or PRO ACTIVE.

THIS SOFTWARE IS PROVIDED BY PRO ACTIVE "AS IS" EITHER FREE
OF CHARGE OR INCLUDED IN A COMMERCIAL PRODUCT. PRO ACTIVE
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.

PRO ACTIVE 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 PRO ACTIVE 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, PRO ACTIVE IS UNDER NO OBLIGATION TO MAINTAIN, CORRECT, UPDATE, CHANGE, MODIFY, OR OTHERWISE SUPPORT THIS SOFTWARE.

Contact: www.springcard.com

6.2. SPRINGCARD ANDROID USB PCSC LIBRARY

SPRINGCARD SOFTWARE LICENSE AGREEMENT

Redistribution and use in binary (object code) form, with or without modification, is 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 BY SPRINGCARD "AS IS" EITHER FREE OF CHARGE OR INCLUDED IN A COMMERCIAL PRODUCT. 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

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