PMD19004-AA
DRAFT - PUBLIC

SpringCard PC/SC-like over BLE Library

Developer's Manual

www.springcard.com

Document Identification

Category	Specification		
Family/Customer	CCID PC/SC Couplers		
Reference	PMD19004	Version	AA
Status	Draft	Classification	Public
Keywords	SpringCore, Puck, PC/SC, CCID, BLE, Android, iOs, Java, Kotlin, Objective C, Swift		
Abstract			

File name	C:\Users\johann\ownCloud\SpringCard\En cours\[PMD19004-AA] BLE CCID library for Mobiles - Developer Manual.odt		
Date saved	09/01/19	Date printed	09/01/19

Revision History

Ver.	Date	Author	Valid. by		Approv. by	Details
			Tech.	Qual.		
AA	07/01/19	JDA				Draft

Contents

1. Introduction

1.1. Abstract

1.1.1. Context

SpringCard offers a wide range of contactless (NFC/RFID @13.56MHz) couplers and of contact (smartcard) couplers. All these devices are designed with PC/SC compliance in mind. PC/SC, short for "personal computer / smartcard" is a standard that eases the developer's job by hiding most of the specificities of any given smartcard reader behind a high-level API, a complex middleware architecture and an interoperable device driver. Unfortunately, this high-end approach is virtually limited to the Windows and Linux worlds, and to USB-based devices.

When it comes to mobile development (Android, iOS) and to smartcard readers that are Bluetooth Smart devices (or BLE, Bluetooth Low Energy), it is more efficient to operate them directly.

SpringCard contactless or contact couplers using BLE as main communication channel are not actually "PC/SC compliant" for they don't come with an interoperable device driver, yet they are still designed with the standard in mind, to ease the developer's job.

More than that, **SpringCard** offers a software Library to operate these BLE devices through a high-level API, that provides a counterpart for all key PC/SC functions (`SCardListReaders`, `SCardConnect`, `SCardTransmit`, `SCardDisconnect`).

1.1.2. Document identification

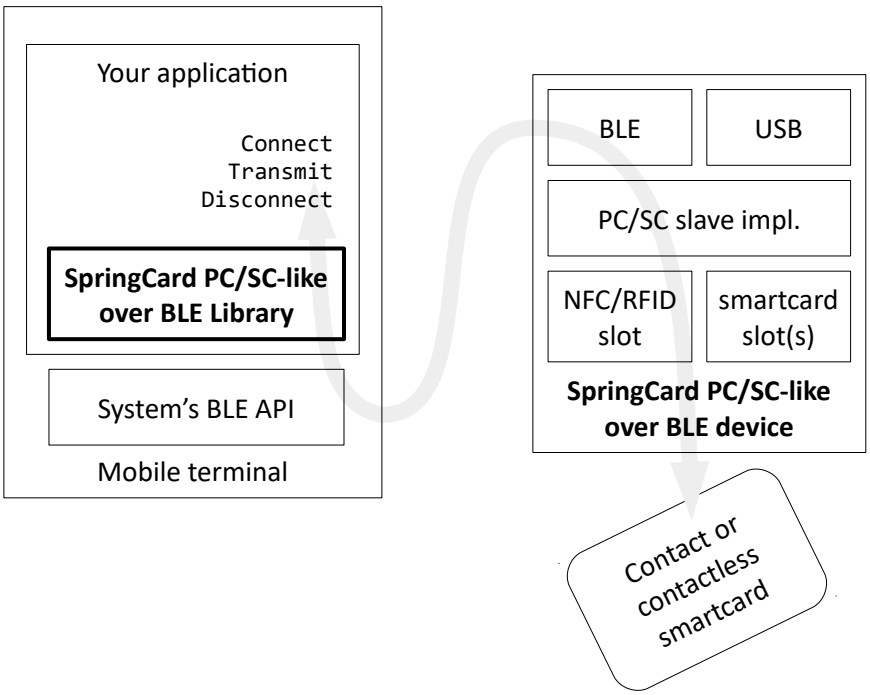
This document is the **Developer's Manual** of the **SpringCard PC/SC-like over BLE Library**, targetting **iOS** and **Android** systems.

It shall be considered as an introduction and an overall guide when using this Library; the detailed documentation of the API is generated from its source code thanks to Doxygen, and only made online at:

- https://docs.springcard.com/apis/iOS/PcscLikeOverBle/ for the iOS Library,
- https://docs.springcard.com/apis/Android/PcscLikeOverBle/ for the Android Library.

***Note:** developers who do not intend to use the Library and aim to work with the BLE device directly shall refer to doc. PMD15282 "SpringCard PC/SC couplers – Zero-driver CCID low-level implementation" instead.*

1.1.3. Architecture overview



This document covers the integration and the use the **SpringCard PC/SC-like over BLE Library** (bold box) into a mobile application.

1.2. Supported products

At the date of writing, the products covered by this document are:

Device name	Description	Platform
SpringCard Puck Blue	Desktop USB+BLE contactless coupler	SpringCore'18
AFCare dual	Mobile dual-slot contact coupler	SpringCore'18

1.3. Audience

This manual is designed for use by application developers. It assumes that the reader has expert knowledge of computer development and a basic knowledge of PC/SC, of the ISO 7816-4 standard for smart-cards, and of the NFC Forum's specifications for contactless cards.

Note: *Beginners are advised to read doc. PMD17041 "Smartcards and contactless smartcards – Integrator's and Implementer's Guide" as an introduction.*

1.4. 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

1.5. Related documents

1.5.1. PC/SC standard

Reference	Publisher	Title
PC/SC	**PC/SC Workgroup**	Interoperability Specification for ICCs and Personal Computer Systems Revision 2 Download link: https://www.pcscworkgroup.com/specifications/download/

1.5.2. Reference documents

Reference	Publisher	Title
PMD17041	SpringCard	Smartcards and contactless smartcards Integrator's and Implementer's Guide
PMD15282	SpringCard	PC/SC couplers Zero-driver CCID low-level implementation

2. Getting started with the Library

2.1. For iOS

2.1.1. Download the Library and the sample project

[To be written: HTH]

2.1.2. Compile and test the sample project

[To be written: HTH]

2.1.3. Use the Library in your own project

[To be written: HTH]

2.2. For Android

2.2.1. Download the Library and the sample project

[To be written: CRA]

2.2.2. Compile and test the sample project

[To be written: CRA]

2.2.3. Use the Library in your own project

[To be written: CRA]

2.3. Licence policy

On both platforms, the Library is made available in 3 forms:

- as a binary file without debug symbols (release version),
- as a binary file with debug symbols (debug version),

- as source code.

The **SpringCard SDK Licence**, reproduced below, grants you an unlimited right to redistribute the binary of the release version together with your application.

You shall not redistribute the binary of the debug version, nor the source code.

2.4. Getting support

SpringCard provides a free support service over the Library and its use together with SpringCard products. The support team is reachable only through email at support@springcard.com.

In order to get an accurate and efficient help, please always identify precisely the host device (manufacturer, product name and version, operating system version), the development environment and language you are using and the **SpringCard** device you are working with.

If you are reporting an issue with a given device or think you have found a bug, please reproduce the issue with the debug version of the Library and send us the detailed execution log, pointing out the line(s) showing the issue. It is a good practice to have also the application log its execution flow and the encountered errors or exceptions.

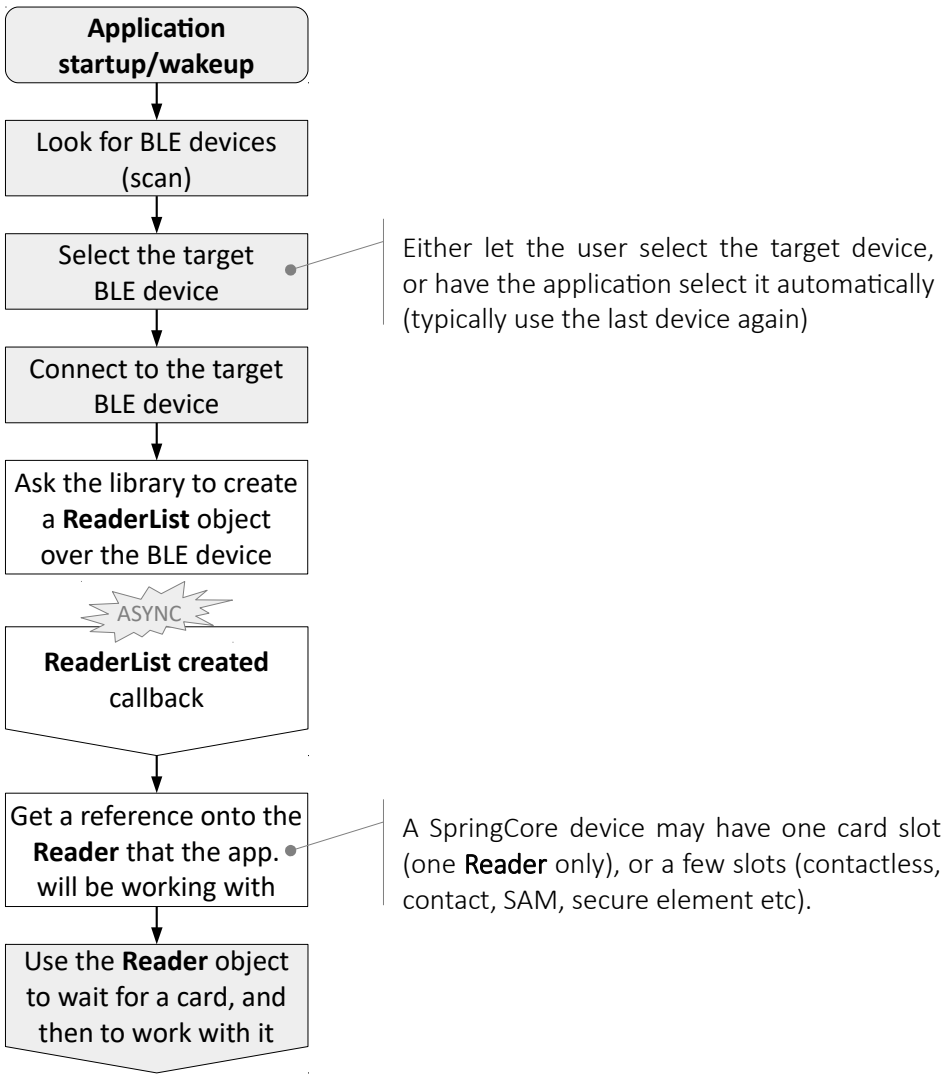
Sorry, but we are not able to provide assistance to an end-user or to a developer who is compiling his own version of the Library instead of using the provided binaries.

3. Flowchart of a typical application

3.1. Open a connection to a SpringCard BLE smartcard reader

In this first step the application shall

1. Enumerate the available BLE devices and initiate a BLE connection with the target device,
2. Initialize the **SpringCard PC/SC-like over BLE Library** onto the said device, and get an access to the target smartcard reader (slot).

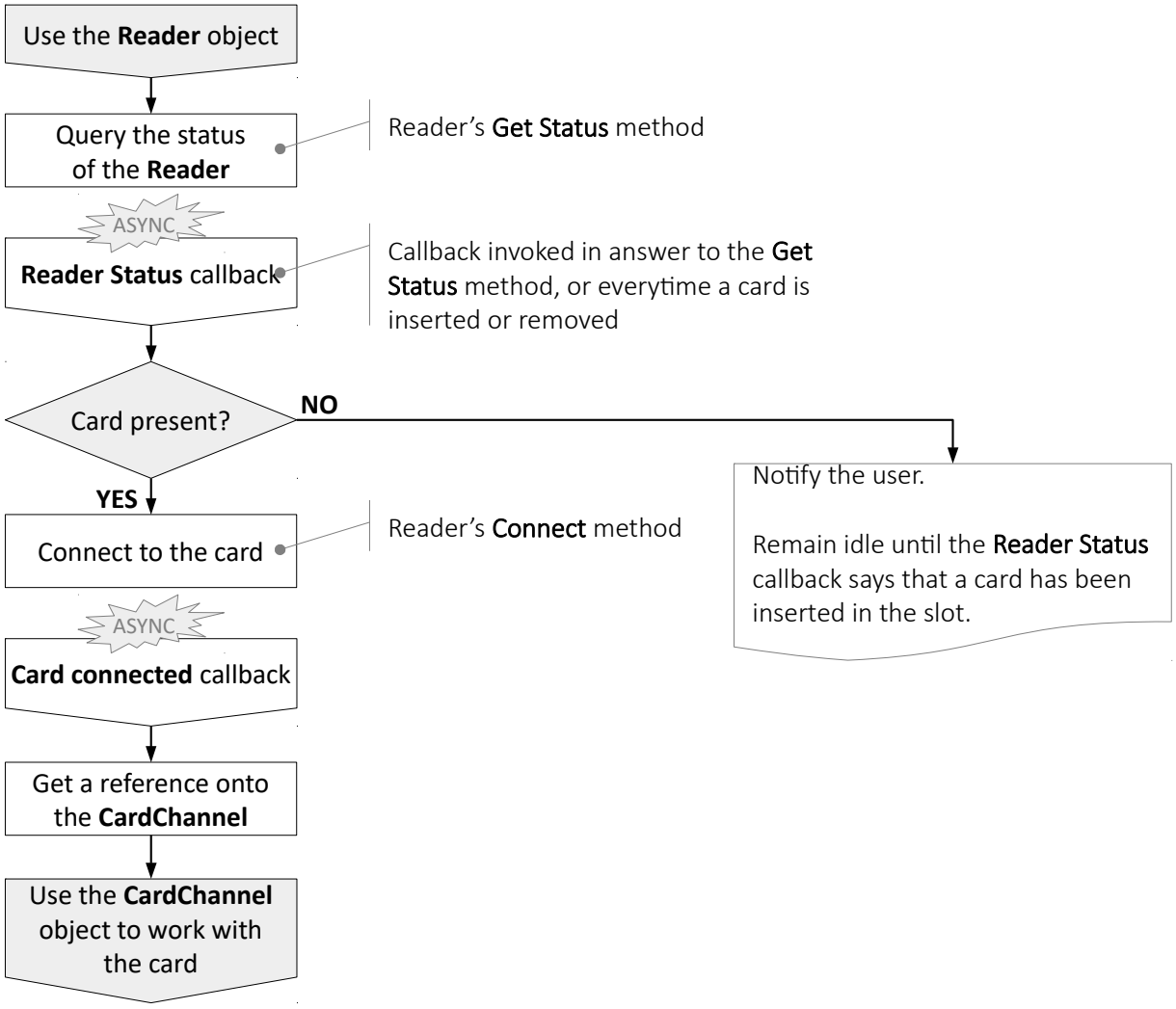


***Note:** in this diagram and the following, the greyed boxes refer to some actions/code that has no direct relationship with the Library. White boxes show the interactions with the Library.*

3.2. Open a connection to a smartcard

In this second step the application shall

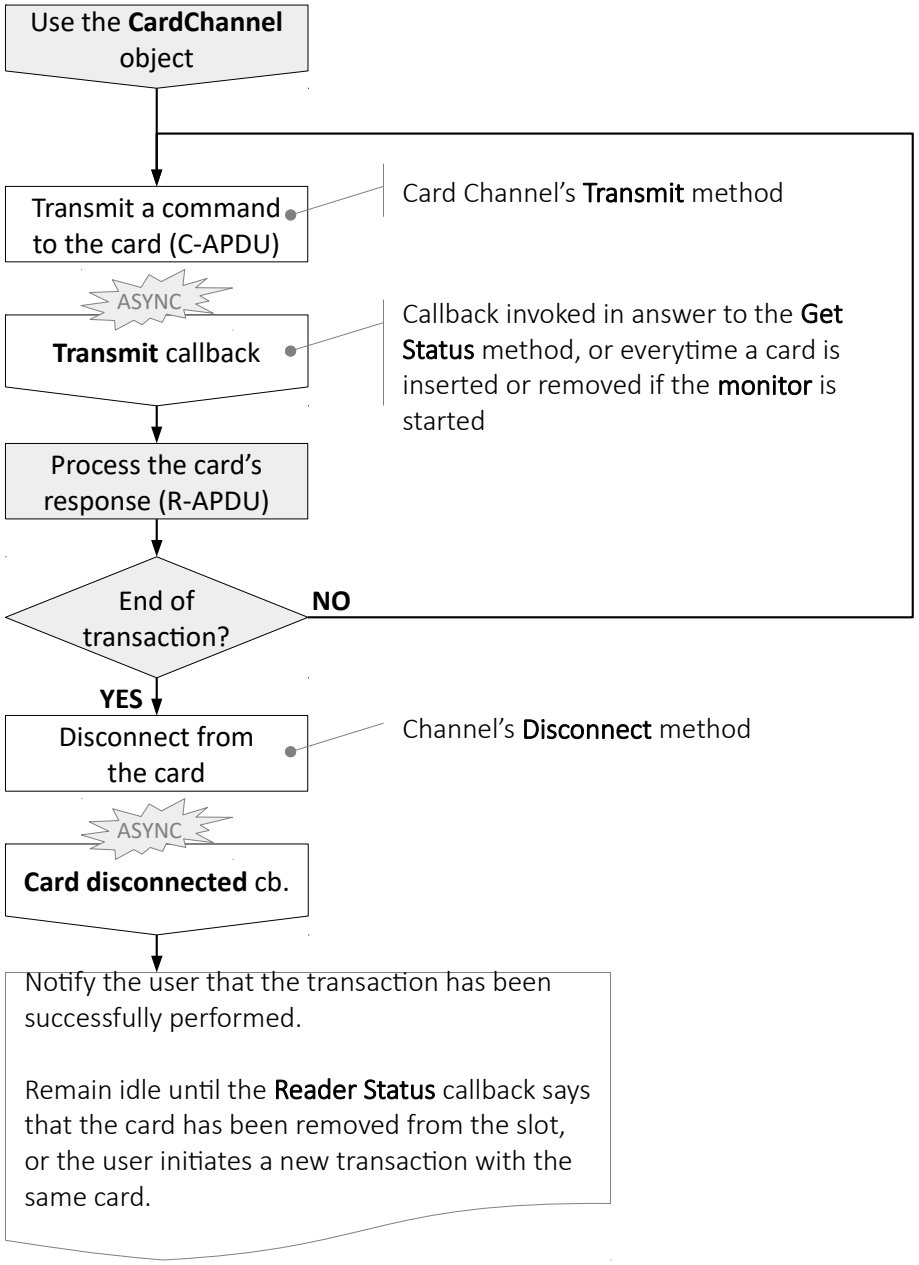
1. Verify that a smart-card is present in the reader, or wait until then,
2. Power up the smart-card and get a communication channel with it.



3.3. Perform the transaction with the smartcard

In this last step, the application runs its key algorithm by sending commands (C-APDU) to the card, and processing its responses (R-APDU).

At the end of the algorithm, the application shall typically disconnect from the card, so the reader may power it down (at least the card is reset to clean up its transaction context).



3.4. Disconnect from the SpringCard BLE smartcard reader

The application should disconnect from the device as soon as the user does not need/want to use smartcards anymore. This let the device power down the smartcards (if some are still present) and to enter a low-power mode, until the application initiates a connection again.

The application may also instruct the device to shut-down.

4. Per-system reference page

4.1. iOS implementation

This chapter lists the API's objects, methods and callbacks, but does not provide any detail regarding their parameters and usage precautions.

Please refer to the reference documentation, available online at:

https://docs.springcard.com/apis/iOS/PcscLikeOverBle/

4.1.1. Application-driven actions and callbacks

Type	*Method or callback*	*Remark*
Ask the library to create a ReaderList object over the BLE device		
async method	`SCardReaderList.create(...)`	Counterpart to PC/SC's `SCardListReaders`
async callback	`onReaderListDidCreate(readers, error)`	
Enumerate the readers in the ReaderList object		
property (int)	`readers.slotCount`	Number of slots
property (array of string)	`readers.slots[]`	Name of every slot
Get a reference onto one reader (i.e. a slot of a multi-slot device)		
method	`reader = readers.getReader(slot)`	`slot` could be either the index or the name
Query the status of the reader		
async method	`reader.getStatus()`	Counterpart to PC/SC's `SCardStatus`
async callback	`onReaderStatus(reader, present, powered)`	
Connect to the card (power up + open a communication channel with the card)		
async method	`reader.cardConnect()`	Counterpart to PC/SC's `SCardConnect`
async callback	`onCardDidConnect(channel, error)`	
Transmit a C-APDU to the card, receive a R-APDU in response		
async method	`channel.transmit(command)`	Counterpart to PC/SC's

		SCardTransmit
async callback	onTransmitDidResponse(**channel**, response, error)	

Disconnect from the card (close the communication channel + power down)

async method	**channel**.cardDisconnect()	Counterpart to PC/SC's SCardDisconnect
async callback	onCardDidDisconnect(**channel**)	

Connect to the card again (re-open an existing communication channel)

async method	**channel**.cardReconnect()	Counterpart to PC/SC's SCardReconnect
async callback	onCardDidConnect(**channel**)	Same as after **reader**.cardConnect()

Disconnect from the BLE device and release the library

async method	**readers**.close()	
async callback	onReaderListDidClose(**readers**)	

4.1.2. Callback fired following a device-initiated event

Type	*Callback*	*Remark*

The BLE device has been lost

async callback	onReaderListDidClose(**readers**)	Same callback as after **readers**.close()

A card is inserted into, or removed from an active reader

async callback	onReaderStatus(**reader**, present, powered)	Same callback as after **reader**.getStatus()

The reader fails to connect (or reconnect) to the card

async callback	onCardDidConnect(**channel**, error)	The **error** object is not null is case of an error.

The card is removed during a communication

async callback	onTransmitDidResponse(**channel**, response, error)	The **error** object is not null is case of an error. The onReaderStatus() callback fires afterwards to notify that the card has been removed.

4.1.3. Error handling

On iOS, all callbacks provide an **`error: Error`** parameter.

If `error` is `nil`, the execution is successful.

If `error` is not `nil`, the execution has failed. The application shall notify the user and, given the possible reason of the error, take appropriate decisions to either retry the last action or report that it is not able to go any further.

[To be written: HTH]

4.2. Android implementation

This chapter lists the API's objects, methods and callbacks, but does not provide any detail regarding their parameters and usage precautions.

Please refer to the reference documentation, available online at:

https://docs.springcard.com/apis/Android/PcscLikeOverBle/

4.2.1. Application-driven actions

Type	*Method or callback*	*Remark*
Ask the library to create a ReaderList object over the BLE device		
async method	`SCardReaderList.create(...)`	Counterpart to PC/SC's `SCardListReaders`
async callback (success)	`onReaderListCreated(readers)`	
async callback (error)	`onReaderListError(error)`	
Enumerate the readers in the ReaderList object		
property (int)	`readers.slotCount`	Number of slots
property (array of string)	`readers.slots[]`	Name of every slot
Get a reference onto one reader (i.e. a slot of a multi-slot device)		
method	`reader = readers.getReader(slot)`	`slot` could be either the index or the name
Query the status of the reader		
async method	`reader.getStatus()`	Counterpart to PC/SC's `SCardStatus`
async callback (success)	`onReaderStatus(reader, present, powered)`	
async callback (error)	`onReaderOrCardError(error)`	
Connect to the card (power up + open a communication channel with the card)		
async method	`reader.cardConnect()`	Counterpart to PC/SC's `SCardConnect`
async callback (success)	`onCardConnected(channel)`	
async callback (error)	`onReaderOrCardError(error)`	
Transmit a C-APDU to the card, receive a R-APDU in response		
async method	`channel.transmit(command)`	Counterpart to PC/SC's `SCardTransmit`

async callback (success)	`onTransmitDone(`**`channel`**`, response)`	
async callback (error)	`onReaderOrCardError(error)`	

Disconnect from the card (close the communication channel + power down)

async method	**`channel`**`.cardDisconnect()`	Counterpart to PC/SC's `SCardDisconnect`
async callback (success)	`onCardDisconnected(`**`channel`**`)`	
async callback (error)	`onReaderOrCardError(error)`	

Connect to the card again (re-open an existing communication channel)

async method	**`channel`**`.cardReconnect()`	Counterpart to PC/SC's `SCardReconnect`
async callback (success)	`onCardConnected(`**`channel`**`)`	Same as after **`reader`**`.cardConnect()`
async callback (error)	`onReaderOrCardError(error)`	

Disconnect from the BLE device and release the library

async method	**`readers`**`.close()`	
async callback (success)	`onReaderListClosed(`**`readers`**`)`	
async callback (error)	`onReaderListError(error)`	

4.2.2. Callback invoked on device-initiated events

Type	*Callback*	*Remark*

The BLE device has been lost

async callback	`onReaderListError(error)`	

A card is inserted into, or removed from an active reader

async callback	`onReaderStatus(reader, present, powered)`	Same callback as after **`reader`**`.getStatus()`

The card is removed during a connect or a transmit

async callback	`onReaderOrCardError(error)`	The `onReaderStatus()` callback fires afterwards to notify that the card has been removed.

4.2.3. Error handling

On Android, the callbacks don’t explicitly provide an error object.

[To be written: CRA]

5. Advanced features

5.1. Control methods

5.1.1. IOS implementation

Type	*Method or callback*	*Remark*
Send a direct command to the device, using the Reader object		
async method	`reader.control(command)`	Counterpart to PC/SC's `SCardControl`
async callback	`onControlDidResponse(readers, response, error)`	**Warning:** the callback targets the parent ReaderList, not the Reader itself
Send a direct command to the device, using the ReaderList object		
async method	`readers.control(command)`	Same as `readers.getReader(0).control()`
async callback	`onControlDidResponse(readers, response, error)`	

5.1.2. Android implementation

Type	*Method or callback*	*Remark*
Send a direct command to the device, using the Reader object		
async method	`reader.control(command)`	Counterpart to PC/SC's `SCardStatus`
async callback (success)	`onControlDone(readers, response)`	**Warning:** the callback targets the parent ReaderList, not the Reader itself
async callback (error)	`onReaderOrCardError(error)`	
Send a direct command to the device, using the ReaderList object		

async method	**readers**.control(command)	Same as readers.getReader(0).control()
async callback (success)	onControlDone(**readers**, response)	
async callback (error)	onReaderOrCardError(error)	

5.2. Secure communication

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 SPRINGCARD 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 SPRINGCARD 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. SPRINGCARD reserves the right to change the information at any time without notice.

SPRINGCARD doesn't warrant any results derived from the use of the products described in this document. SPRINGCARD 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. SPRINGCARD customers using or selling these products for use in such applications do so on their own risk and agree to fully indemnify SPRINGCARD for any damages resulting from such improper use or sale.

Copyright notice

Editor's information

SPRINGCARD 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