FEITIAN

Feitan Mobile Readers Android Developer Guide

V1.3

Feitian Technologies Co., Ltd.

Website: www.FTsafe.com

Revision History:

Date	Revision	Description
June. 2018	V1.0	Release of the first version
Feb. 2019	V1.1	Release the 1.1 version, add private API contents
March. 2019	V1.2	Release the 1.2 version, add ReaderAutoTurnOff API
Nov, 2021	V1.3	Add getBleFirmwareVersion API

Software Developer's Agreement

All Products of Feitian Technologies Co., Ltd. (Feitian) including, but not limited to, evaluation copies, diskettes, CD-ROMs, hardware and documentation, and all future orders, are subject to the terms of this Agreement. If you do not agree with the terms herein, please return the evaluation package to us, postage and insurance prepaid, within seven days of their receipt, and we will reimburse you the cost of the Product, less freight and reasonable handling charges.

1. Allowable Use – You may merge and link the Software with other programs for the sole purpose of protecting those programs in accordance with the usage described in the Developer's Guide. You may make archival copies of the Software.

2. Prohibited Use – The Software or hardware or any other part of the Product may not be copied, reengineered, disassembled, decompiled, revised, enhanced or otherwise modified, except as specifically allowed in item 1. You may not reverse engineer the Software or any part of the product or attempt to discover the Software's source code. You may not use the magnetic or optical media included with the Product for the purposes of transferring or storing data that was not either an original part of the Product, or a Feitian provided enhancement or upgrade to the Product.

3. Warranty – Feitian warrants that the hardware and Software storage media are substantially free from significant defects of workmanship or materials for a time period of twelve (12) months from the date of delivery of the Product to you.

4. Breach of Warranty – In the event of breach of this warranty, Feitian's sole obligation is to replace or repair, at the discretion of Feitian, any Product free of charge. Any replaced Product becomes the property of Feitian.

Warranty claims must be made in writing to Feitian during the warranty period and within fourteen (14) days after the observation of the defect. All warranty claims must be accompanied by evidence of the defect that is deemed satisfactory by Feitian. Any Products that you return to Feitian, or a Feitian authorized distributor, must be sent with freight and insurance prepaid.

EXCEPT AS STATED ABOVE, THERE IS NO OTHER WARRANTY OR REPRESENTATION OF THE PRODUCT, EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

5. Limitation of Feitian's Liability – Feitian's entire liability to you or any other party for any cause whatsoever, whether in contract or in tort, including negligence, shall not exceed the price you paid for the unit of the Product that caused the damages or are the subject of, or indirectly related to the cause of action. In no event shall Feitian be liable for any damages caused by your failure to meet your obligations, nor for any loss of data, profit or savings, or any other consequential and incidental damages, even if Feitian has been advised of the possibility of damages, or for any claim by you based on any third-party claim.

6. Termination – This Agreement shall terminate if you fail to comply with the terms herein. Items 2, 3, 4 and 5 shall survive any termination of this Agreement.

Contents

Chapter 1 Get Started

This document describes how to develop application based on Feitian Mobile SDK, including the development interfaces supported by mobile readers and how to develop applications based on these interfaces. This chapter covers the following topics:

- Application development interfaces and library of Feitian mobile readers
- Development of applications using PC/SC interfaces
- Development of applications using private interfaces

The Feitian combo SDK supported below readers:
IR301, BR301, BR301BLE, BR500, more information, please go our website see more details:
https://ftsafe.com/Products/reader

Below is the interface support by Android:

Reader name	IR301	BR301	BR301BLE	BR500
USB	✓	✓	✓	✓
Bluetooth 3.0	✗	✓	✗	✗
Bluetooth 4.0	✗	✗	✓	✓

Figure 1

The figure 1 showed the interface be supported by these readers, all readers had Micro USB interface, so you can using OTG cable with reader on Android platform. Alternatively, you can choose the Bluetooth 3.0 or 4.0 as communication channel work with reader.

1.1 Introduction of Android SDK

We provide two interfaces and libraries to access the mobile reader, the simplest **private API** based on java language and **PC/SC compatible API** based on C language. We prefer using private API for your new project, and the PC/SC compatible API for exist project which you already using on PC platform.

Below is list libs that we provided:

Figure 2

JAR Library – private API definition		
Private API related files	ftsafe.jar	This jar include communication API and private API, also include the service, you can include this jar into your project and using private API directly, without load.h and.a files)
Static Library – PC/SC API definition, all related files need include to your project		
PC/SC API related files	ft_pcsc.h	The define of PCSC API
	ft_socket.h	The define of socket API
	ft_tools.h	The define of string exchange
	ftsafe.jar	Based on above, the ftsafe.jar is service side, it will running as PCSC service to handle the request from.a. so in your project, you will include this file too.
	Libftpcsc_arm_0.1.a	The.a is PCSC client side, it send request to ftsafe.jar, for example, when user call ScardConnect, then.a will send request to ftsafe.jar to connect the reader, then the ftsafe.jar will call USB or Bluetooth framework do connect the reader and get data back

1.2 Install demo app to your android device

Install the APK and to do test the function, two demos will provided in future:

Demo #1 (On development): High quality application which using to tracking and debug the hardware device

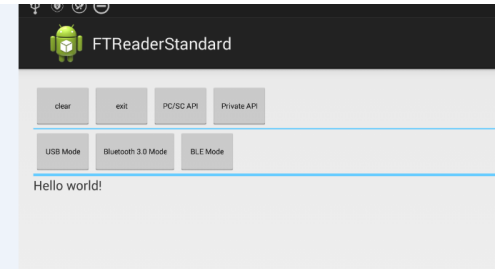
Demo #2: Functional buttons for show APIs

Download the APK from below:

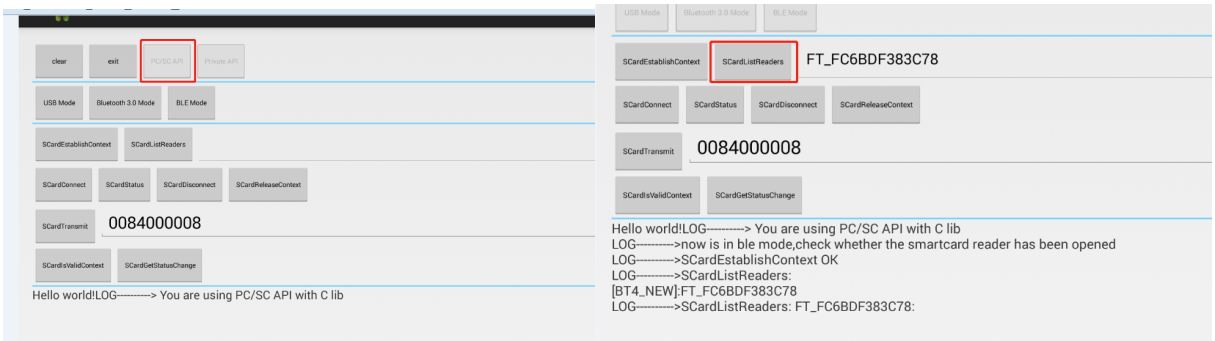
https://github.com/FeitianSmartcardReader/FEITIAN_MOBILE_READERS/raw/master/Android%20SDK/BIN/FTReaderStandard.apk

1.3 Test demo application

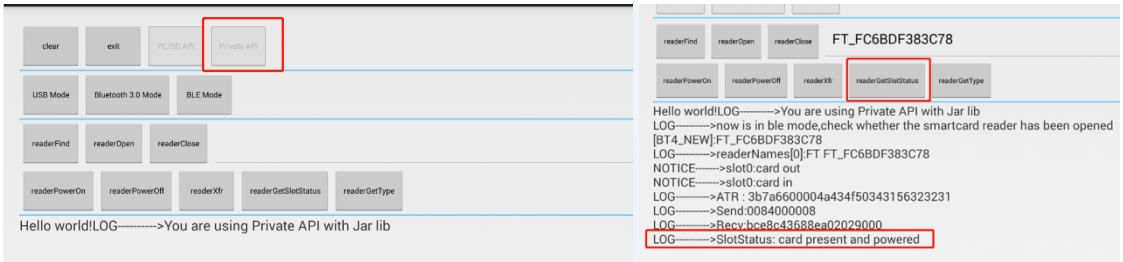
Below screenshot is main view, the first line give a reference for PC/SC API and Private API, the second line provide which reader interface are you using, USB Mode with OTG or Bluetooth 3.0 or Bluetooth 4.1/4.2.



Below screenshot is a view for PC/SC API:



Below screenshot show private API view:



Chapter 2 Android Integration – Private API

Download the android SDK

Download from https://github.com/FeitianSmartcardReader/FEITIAN_MOBILE_READERS/archive/master.zip

Build the example App

The project is developed based on eclipse, please using eclipse + NDK + Android on your PC. Import the whole project into your eclipse, build and run.

Make sure all related files already in the project which listed in figure 2.

We provide demo source code for integration, you can check the code and make your own code.

Develop your own application

2.1.1 Authority the permission in your project

Our SDK need access the USB and Bluetooth device, so before start development, we need open the authority. You can copy below code into your AndroidManifest.xml file.

```
<uses-feature android:name="android.hardware.usb.host" />
<uses-permission android:name="android.permission.USB_PERMISSION" />
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
<uses-permission android:name="android.permission.INTERNET" />
<uses-feature android:name="android.hardware.bluetooth_le" android:required="true"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.BLUETOOTH" />
```

2.1.2 Using Feitian private API

Feitian private API, support Bluetooth, OTG and Bluetooth smart interface reader from Feitian.

We suggestion customer using private API, because it is easy to use and without complex configuration, all API wrote by Java, so you can copy the ftsafe.jar into your project and using the private API directly.

Function	Description
FTReader	Constructor API
readerFind	Find the reader, this API will scan Bluetooth and USB, to list the reader found
readerOpen	Connect the reader
readerClose	Disconnect the reader
readerPowerOn	Do power on to the card
readerPowerOff	Do power off the card
readerXfr	Transfer APDU to card and get return data back
readerEscape	Transfer escape APDU to card
readerAutoTurnOff	Using to disable auto saving battery feature
readerGetSlotStatus	Get slot status, for example, card absent or present
readerGetType	Get reader type, the SDK support all Feitian mobile reader, so call this API can get specific reader type
readerGetSerialNumber	Get reader serial number
readerGetFirmwareVersion	Get reader firmware number
readerGetUID	Get reader UID
readerGetLibVersion	Get reader lib version
getBleFirmwareVersion	Get BLE reader Bluetooth version

2.1.3 FTReader Constructor API

The private API has definition in “com.ftsafe.readerScheme” package, the class name call FTReader

The FTReader constructor is defined as follows:

```
public FTReader(Context context, Handler handler, int FTREADER_TYPE)
Parameters:
    Context context : Input parameter for context from user
    Handler handler : Input parameter for handler from user
        Using to receive the notification in SDK, include below status:
            com.ftsafe.DK.USB_IN : Plug in USB reader
            com.ftsafe.DK.USB_OUT : Plug out USB reader
            com.ftsafe.DK.CARD_IN_MASK : Card present
            com.ftsafe.DK.CARD_OUT_MASK : Card absent/remove
            com.ftsafe.DK.BT3_NEW : Detected the Bluetooth 3.0 reader
            com.ftsafe.DK.BT4_NEW : Detected the Bluetooth 4.1/4.2 reader
            com.ftsafe.DK.BT4_ACL_DISCONNECTED : disconnect Bluetooth 4.1/4.2 reader

        int FTREADER_TYPE : Input parameter, using to tell SDK which reader will be using:
            com.ftsafe.DK.FTREADER_TYPE_USB : Using USB OTG reader
            com.ftsafe.DK.FTREADER_TYPE_BT3 : Using Bluetooth 3.0 reader
            com.ftsafe.DK.FTREADER_TYPE_BT4 : Using Bluetooth 4.1/4.2 reader

When using USB OTG, do plug out/in reader will receive the notification in Handler parameter, which
is com.ftsafe.DK. USB_IN or com.ftsafe.DK.USB_OUT
```

2.1.4 ReaderFind

The private API has definition in “com.ftsafe.readerScheme” package, the class name call FTReader

The FTReader constructor is defined as follows:

```
public Public void readerFind() throws FTException
For USB OTG, when detect the Feitian USB OTG reader, the handler will inform the user to apply USB
permission, and after got approval by user, will through the PID (Product ID) list all Feitian USB readers.

For Bluetooth 3.0 and 4.1/4.2 reader, when call this API, the SDK will filter the Bluetooth ID which
named “FT_”, and through Handler to inform user com.ftsafe.DK.BT3_NEW or com.ftsafe.DK.BT4_NEW.
```

If the reader cannot detected, make sure you have add access USB permission into your project:

```
<uses-feature android:name="android.hardware.usb.host" />
<uses-permission android:name="android.permission.USB_PERMISSION" />
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
```

And for Bluetooth 3.0 and Bluetooth 4.1/4.2, make sure you add below access permission into your project.

```
<uses-feature android:name="android.hardware.bluetooth_le" android:required="true"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.BLUETOOTH" />
```

2.1.5 readerOpen

The readerOpen API is defined as follows:

```
public String[] readerOpen(Object device) throws FTException
Parameter:
    Object device :
        USB OTG Reader: NULL (You also can input UsbDevice object, need get by customer code )
  Bluetooth 3.0/4.1/4.2: Input BluetoothDevice object (which is get from Handler )
Return :
        USB OTG Reader: Return reader list, support multi-slots, like R502 Dual interface reader
  Bluetooth 3.0/4.1/4.2: Return reader name or Bluetooth ID
```

2.1.6 readerClose

The readerClose API is defined as follows:

```
public String[] readerClose() throws FTException
The API will close all opened reader.
```

2.1.7 readerPowerOn

The readerPowerOn API using to power on the card, below is defined as follows:

```
public byte[] readerPowerOn(int index) throws FTException
Parameter :
    int index :
        USB OTG Device : Provide the card slot number, which is get from readerOpen
```

```
  Bluetooth 3.0/4.1/4.2 : 0
Return value : Card ATR
```

2.1.8 readerPowerOff

The readerPowerOff API using to power off the card, below is defined as follows:

```
public void readerPowerOff(int index) throws FTException
Parameter :
    int index :
        USB OTG Device : Provide the card slot number, which is get from readerOpen
  Bluetooth 3.0/4.1/4.2 : 0
```

2.1.9 readerXfr

The readerXfr API using to send command/APDU to card, below is defined as follows:

```
public byte[] readerXfr(int index,byte[] send) throws FTException
Parameter :
    int index : Card slot number
            - USB OTG is the index value of the card slot
            - Bluetooth 3.0/4.1/4.2 set to 0
    byte[] send :  Data/APDU for send to card
Return value :
        Response data from card
```

2.1.10 readerEscape

The readerEscape API using to send escape command to reader, below is defined as follows:

```
public byte[] readerEscape(int index, byte[] send) throws FTException
Parameter :
    int index : Card slot number
            - USB OTG is the index value of the card slot
            - Bluetooth 3.0/4.1/4.2 set to 0
    byte[] send : Data/APDU for send to reader
Return value :
    Response data from reader
```

2.1.11 readerAutoTurnOff

The readerAutoTurnOff API using to set the power saving feature, the BLE reader was designed with auto power on, if without any data communicate with reader, it will be going to turn off automatically, the purpose is to saving battery. Customer can through call this API to turn on/off this feature, below is defined as follows:

```
public byte[] readerAutoTurnOff (bool Value) throws FTException
Parameter：
    bool Value：
            - If the value = false, means the reader will switch off power saving feature, the device won’t going to turn off after 2 mins (the condition is without any data communication between reader and App)
            - If the value = true, means the reader switch on power saving feature, the reader will not be going to turn off by itself.
Return value：
    Response data from reader, 9000 means successful
```

2.1.12 readerGetSlotStatus

The readerGetSlotStatus API using to send get card slot status, below is defined as follows:

```
public int readerGetSlotStatus(int index) throws FTException
Parameter：
    int index：Card slot number
            - USB OTG is the index value of the card slot
            - Bluetooth 3.0/4.1/4.2 set to 0
    Card slot status：
        com.ftsafe.CARD_PRESENT_ACTIVE：Present card and already powered
        com.ftsafe.CARD_PRESENT_INACTIVE：Present card and un-power
        com.ftsafe.CARD_NO_PRESENT：Card absent
```

2.1.13 readerGetType

The readerGetType API using to get reader type, this API only works for OTG reader, not for Bluetooth reader, below is defined as follows:

```
public int readerGetType() throws FTException
Return value：
    com.ftsafe.READER_R301E
    com.ftsafe.READER_BR301FC4
    com.ftsafe.READER_BR500
    com.ftsafe.READER_R502_CL
```

```
com.ftsafe.READER_R502_DUAL
com.ftsafe.READER_BR301
com.ftsafe.READER_IR301_LT
com.ftsafe.READER_IR301
com.ftsafe.READER_VR504
com.ftsafe.READER_UNKNOW
```

2.1.14 readerGetSerialNumber

The readerGetSerialNumber API using to get reader serial number, below is defined as follows:

```
public byte[] readerGetSerialNumber() throws FTException
Return value： Reader serial number
```

2.1.15 readerGetFirmwareVersion

The readerGetFirmwareVersion API using to get reader firmware version, below is defined as follows:

```
public String readerGetFirmwareVersion() throws FTException
Return value： Firmware version
```

2.1.16 readerGetUID

The readerGetUID API using to get reader UID (user ID), the User ID is hex number which using to encrypted whole reader firmware and also using for reader license authorize, customer can write UID in bulk readers, and work with their application to bind the reader only can work with their App. Below is defined as follows:

```
public byte[] readerGetUID() throws FTException
Return value： User ID string
```

2.1.17 readerGetManufacturer

The readerGetManufacturer API using to get reader Manufacturer:

```
public byte[] readerGetManufacturer() throws FTException
Return value：Manufacturer name with hex
```

2.1.18 readerGetHardwareInfo

The readerGetHardwareInfo API using to get reader Manufacturer:

```
public byte[] readerGetHardwareInfo() throws FTException
Return value：Hardware info name with hex
```

2.1.19 readerGetReaderName

The readerGetReaderName API using to get reader name:

```
public byte[] readerGetReaderName() throws FTException
Return value：Hardware info name with hex
```

2.1.20 readerGetLibVersion

The readerGetLibVersion API using to get lib version:

```
public static String readerGetLibVersion()
```

2.1.21 Example:

Step #1: apply USB and Bluetooth permission first

Step #2: Call readerFind to register reader monitor, it will start filter Bluetooth and USB device, and feedback to user

Step #3: After get the list, call readerOpen API to open/connect the reader

Step #4: through the event API to check the card slot status

Step #5: If detect the card, then call readerPowerOn, the API will return your card ATR, you can call readerXfr API to send the APDU to the card, and get response data

Step #6: If you want operate the reader, you can call readerEscape or readerGetxxxx API to check the reader information, after finish your operation with card, you need call readerPowerOff the power off the card

Step #7: after done, please call readerClose to disconnect the reader and the reader will going to turn off automatically to save battery.

Chapter 3 Android Integration – PC/SC API

Since most PC software using PC/SC API, so support on this, we do wrap PC/SC API for users, below is the API already implemented, since it is wrap, so we just implement the important API, if you want have extra of these API, contact us we will support on this.

Function	Description	Implementation
SCardEstablishContext	Create a context handle for accessing the smart card database.	✓
SCardReleaseContext	Close the context handle for accessing the smart card database.	✓
SCardListReaders	Obtain the list of the smart card groups.	✓
SCardConnect	Connect to a smart card.	✓
SCardStatus	Provide the status of a reader.	✓
SCardTransmit	Transmit data with a smart card via T=0 or T=1 protocol.	✓
SCardDisconnect	Terminate a connection to a smart card.	✓
SCardIsValidContext	Determines whether a smart card context handle is valid.	✗
SCardReconnect	Re-establish a connection to a smart card.	✗
SCardBeginTransaction	The function waits for the completion of all other transactions before it begins. After the transaction starts, all other applications are blocked from accessing the smart card while the transaction is in progress.	✗
SCardEndTransaction	Completes a previously declared *transaction*, allowing other applications to resume interactions with the card.	✗
SCardGetStatusChange	Blocks execution until the current availability of the cards in a specific set of reader's change.	✓
SCardControl	Gives you direct control of the *reader*. You can call it any time after a successful call to SCardConnect and before a successful call to SCardDisconnect. The effect on the *state* of the reader depends on the control code.	✓
SCardListReaderGroups	Provides the list of *reader groups* that have previously been introduced to the system.	✗
SCardFreeMemory	Releases memory that has been returned from the resource manager using the SCARD_AUTOALLOCATE length designator.	✗
SCardCancel	Terminates all outstanding actions within a specific resource manager context. The only requests that you can cancel are those that require waiting for external action by the smart card or user. Any such outstanding action requests will terminate with a status indication that the action was canceled. This is especially useful to force outstanding SCardGetStatusChange calls to terminate.	✗
SCardGetAttrib	Retrieves the current reader attributes for the given handle. It does not affect the state of the reader, driver, or card.	✗

SCardSetAttrib	The SCardGetAttrib function retrieves the current reader attributes for the given handle. It does not affect the state of the reader, driver, or card.	✗

3.1 How to using PC/SC API develop your App

To using PC/SC API is complex than jar lib, because it using multi-language to implement, create makefile build with NDK together. To avoid the misunderstanding, we also create a demo code to explain how to setup and using PC/SC API in your application.

For example:

1. In our demo application, we wrote our demo App by C++, file is FtReaderTest.cpp, and you can find it in our demo code.
2. We create Android.mk using to build C++ file with ftpcsc_arm_0 static library, it will generated dynamic lib FtReaderTest.so
3. Create Tpcsc.java file, using javah command to generate com_example_ftreadertest_Tpcsc.h
4. Write UI and call API from jni

```
package com.example.ftreadertest;
import com.ftsafe.PcscServer;

final static int PORT = 0x096e;
PcscServer pcscServer = new PcscServer(PORT,MainActivity.this, mHandler);
                ftReader = pcscServer.getFtReaderObject();
new Tpcsc().testA(PORT);
```

Notice:

1. Before call API in JNI, you need start pcscServer first, the API need communication with ftsat.jar by socket.
2. The port is constant, so please keep it as 0x096E.

Chapter 4 Private control command

The reader also provides private control command for some users, you can call SCardControl/readerEscape API to send the below command to get your needs.

Turn ON/OFF auto power off feature

Bluetooth reader only:

- For Feitian Bluetooth reader, to have long usage for Bluetooth reader, we add save power feature in our firmware, which is the firmware will detect the communication data, if no any data through the firmware, then the reader will be going to power off.

Function	Command
Switch off automatically power off feature	A55A8031
Switch on automatically power off feature	A55A8030

For more information, please have a look the demo code.

Chapter 5 Check the reader and card slot event

The SDK provide a way to check the reader and card slot event automatically. Please check source code:

```
    private Handler mHandler = new Handler(){
        @Override
        public void handleMessage(Message msg){
            super.handleMessage(msg);
            TextView textView = (TextView) findViewById(R.id.textView);
            String log=null;
            switch (msg.what) {
            case -1:
                textView.setText("");
                return;
            case 0:
                log = msg.obj.toString();
                textView.append("LOG---------->"+log+"\n");
                break;
            case DK.USB_IN:
                textView.append("NOTICE------->USB Device In\n");
                /*try {
                    ftReader.readerFind();
                } catch (Exception e) {
                    textView.append("NOTICE------->USB Device In------->this should not
happend------>" + e.getMessage() + "\n");
                }*/
                break;
            case DK.USB_OUT:
                textView.append("NOTICE------->USB Device Out\n");
                break;
            case DK.PCSCSERVER_LOG:
                //textView.append("[PcscServerLog]:"+msg.obj+"\n");
                break;
            case DK.USB_LOG:
                //textView.append("[USBLog]:"+msg.obj+"\n");
                break;
            case DK.BT3_LOG:
                //textView.append("[BT3Log]:"+msg.obj+"\n");
                break;
            case DK.BT4_LOG:
                //textView.append("[BT4Log]:"+msg.obj+"\n");
                break;
            case DK.FTREADER_LOG:
```

```
                //textView.append("[FTReaderLog]:"+msg.obj+"\n");
                break;
            case DK.CCIDSCHEME_LOG:
                //textView.append("[CCIDSchemeLog]:"+msg.obj+"\n");
                break;
            case DK.BT3_NEW:
                BluetoothDevice dev1 = (BluetoothDevice) msg.obj;
                textView.append("[BT3_NEW]:"+dev1.getName()+"\n");
                arrayForBlueToothDevice.add(dev1);
                addSpinnerPrivate(dev1.getName());
                break;

            case DK.BT4_NEW:
                BluetoothDevice dev2 = (BluetoothDevice) msg.obj;
                textView.append("[BT4_NEW]:"+dev2.getName()+"\n");
                arrayForBlueToothDevice.add(dev2);
                addSpinnerPrivate(dev2.getName());
                break;
            case DK.BT4_ACL_DISCONNECTED:
                BluetoothDevice dev3 = (BluetoothDevice) msg.obj;
                textView.append("[BT4_ACL_DISCONNECTED]: "+dev3.getName()+" disconnected\n");
                break;
            default:
                if((msg.what & DK.CARD_IN_MASK) == DK.CARD_IN_MASK){
                    textView.append("NOTICE------->"  +  "slot"+(msg.what%DK.CARD_IN_MASK)+":card
in\n");
                    return;
                }else if((msg.what & DK.CARD_OUT_MASK) == DK.CARD_OUT_MASK){
                    textView.append("NOTICE------->"  +  "slot"+(msg.what%DK.CARD_OUT_MASK)+":card
out\n");
                    return;
                }
                break;
            }
        }
    };
```