FEITIAN

aR530 Developer Guide for Android

http://www.ftsafe.com

Revision History:

Date	Revision	Description
Jan, 2014	1.0	First version
13th, May, 2014	1.1	1. Removed MSR card support 2. Support find Specify card type 3. Add get reader hardware ID and firmware version
26th, May, 2014	1.2	1. Add error code 2. Modify API parameter
28th, May, 2014	1.3	Add GetCardInfoData() API
30th, May, 2014	1.4	Add comments in get card type API
11th, June, 2014	1.5	Change manual name to aR530
4th, Jan, 2015	1.6	Change the structure of the document
16th, April, 2015	1.7	Add Get reader UID API
2th, March, 2016	1.8	Add Buzzer Control 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. Overview

This document describes how to develop application based on Feitian aR530, and the document has guide developers to using API to do operate with Feitian aR530.

We through four parts to describe aR530 SDK.

1.1 First part, describe application dynamic library and JAR file, to let developer have a basic concept.

1.2 Second part, we do explain all support APIs, and have to do each API means

1.3 Third part, the details of APIs

1.4 Describe the development process and notice items

Chapter 2. Development Overview

2.1 Describe static library and JAR file

The aR530 SDK based on dynamic library (**.SO**) and JAR package (**.jar**), The below chart listed related file:

File	SDK Path
AudioKeySDK.jar	\Include (Audio jack communications - jar package)
cardReaderAPI.jar	\Include
Libcardreader_api.so	\lib (Audio jack communication interface library)

cardReaderAPI.jar is core file of aR530, it implement all API which defined in operation interface API. To be sure developer can be using aR530, the developer will be needed to include above files into project.

2.2 Describe operation APIs

Number	API name	Description	Status
initialization APIs			
1	Initial	Initialize the environment	Implemented
2	Release	Release resources	Implemented
Get info from reader and library			
3	GetDeviceID	Get the serial number from hardware	Implemented
4	GetFirmwareVersion	Get the firmware version rom hardware	Implemented
5	GetLibVersion	Get the lib version from library	Implemented
Contactless functions			
6	FTNFC_connect	Connect with NFC card (power on to card)	Implemented

7	FTNFC_disconnect	To disconnect card from reader(power off to card)	Implemented
8	FTNFC_transmitCmd	Send command to NFC card	Implemented
9	FTNFC_cardType	Return the card type, call this API after FTNFC_Connect	Implemented
10	GetCardInfoData	Return card information	Implemented
Mifare card functions			
11	GeneralAuthenticate	To authentication of the special block which need to do operation	Implemented
12	ReadBinary	Read block data	Implemented
13	ClassicBlockInitial	To do initialization of the special block	Implemented
14	ClassicReadValue	Read value	Implemented
15	ClassicStoreBlock	Write value	Implemented
16	ClassicIncrement	Increase value operation	Implemented
17	ClassicDecrement	Decrease value operation	Implemented
Handshake functions			
18	HandShake	Create connection between phone/table and aR530	Implemented
Buzzer control functions			
19	TrunOffAutoBuzzer	Trun off automatic beep	Implemented
20	BuzzerBeep	buzzer beep	Implemented

Chapter 3. Definitions

3.1 Error codes

The following is a list of commonly used errors. Since different cards produce different errors they must map over to these error messages.

```
//The firmware return status

public String errContent(int errCode) {

    switch (errCode) {

    case Card.CODE_FAIL:

        return "Fail";

    case Card.CODE_DEVICE_NOT_AVAILABLE:

        return "device is not available";

    case Card.CODE_CARD_NOT_CONNECTED:

        return "card is not connected";

    case Card.CODE_DEVICE_COMM_ERROR:

        return "communication error";

    case Card.CODE_PARAM_ERROR:

        return "illegal parameters";

    case Card.CODE_TIMEOUT:

        return "timeout";

    default:

        return "unkown error " + errCode;

    }

}
```

3.2 Card type

Can through card type to choose specify card

//The firmware return status

Class Card have below member objects

Card.CARD_TYPE.A_CARD

Card.CARD_TYPE.B_CARD

Card.CARD_TYPE.Felica_CARD

Card.CARD_TYPE.A_M1_CARD

Card.CARD_TYPE.B_M1_CARD

Card.CARD_TYPE.Topaz_CARD

Through call FTNFC_connect(Card.Type) to choose specify card, more information, please follow FEITIAN sample code

Chapter 4. API Reference

3.1 Initial function

Synopsis:
public native static int initial (Context con);
Parameters:
Context con IN the type must be 1 or 2, more information, please follow sample code
Description:
Initial context and environment before using
Example:
More information, please follow sample code.
Returns:
Reference errContent API

3.2 Release function

Synopsis:
public native static int release ();
Parameters:
NULL
Description:
Initial environment before use
Example:
More information, please follow sample code.
Returns:
Reference errContent API

4.3 Get reader information functions

4.3.1 GetDevicID

Synopsis:
public int GetDeviceID(byte[] deviceID, JKeyInt len);
Parameters:
DeviceID out using to saved reader hardware serial number
Len out Return length of hardware serial number

Description:
This function get device serial number from reader.

Example:
More information, please follow sample code.
Returns:
Reference error code section

4.3.2 GetDevUID

Synopsis:
public string GetDevUID();
Parameters: N/A
Description:
Get the reader UID(user ID).

Example:
More information, please follow sample code.
Returns:
Reference error code section

Synopsis:
public int GetDeviceID(byte[] deviceID, JKeyInt len);
Parameters:
DeviceID out using to saved reader hardware serial number
Len out Return length of hardware serial number

Description:
This function get device serial number from reader.

Example:

More information, please follow sample code.

Returns:

Reference error code section

4.4 Contactless section

4.4.1 FTNFC_connect

Synopsis:

public int FTNFC_connect(Card.CARD_TYPE[] cardTypes, int timeout);

Parameters:

Card.card_type[] in input array of card type

Card type can be below:

Card.CARD_TYPE.A_CARD

Card.CARD_TYPE.B_CARD

Card.CARD_TYPE.Felica_CARD

Card.CARD_TYPE.A_M1_CARD

Card.CARD_TYPE.B_M1_CARD

Card.CARD_TYPE.Topaz_CARD

Timeout in timeout while in scan card (second) at list 1 second

Description:

This function using to connect specify card

Example:

More information, please follow sample code.

Returns:

Reference error code section

4.4.2 FTNFC_transmitCmd

Synopsis:

public native static int FTNFC_ transmitCmd (byte[] sendData, byte[] recvData);

Parameters:

sendData IN command which will send to card

recvData OUT return data from card

Description:

This function use to do transfer data between reader and card.

Example:

More information, please follow sample code.

Returns:

Please check error section

4.4.3 FTNFC_disconnect

Synopsis:

public native static int FTNFC_disconnect ();

Parameters:

NULL

Description:

This function use to disconnect reader.

Example:

More information, please follow sample code.

Returns:

SUCCESS	Successful

4.4.4 FTNFCCardType

Synopsis:

public Card.CARD_TYPE FTNFC_cardType();

Parameters:

NULL

Description:

Return current card type, after FTNFC_connect to call

Example:

More information, please follow sample code.

Returns:

SUCCESS	Successful

4.4.5 GetCardInfoData

Synopsis:

public byte[] GetCardInfoData();

Parameters:

NULL

Description:

Return connected card information

Example:

More information, please follow sample code.

Returns:

Reference error code section

Notice:

A: return null if without any card connect

B: If the card connected, then return byte array which describe card information

<table>
<tr><th>Card type</th><th>Return data</th></tr>
<tr><td>Type A</td><td>
<table>
<tr><td>0x0A</td><td>Sak</td><td>Uid_len</td><td>UID</td></tr>
<tr><td>Type A</td><td>1 byte</td><td>Length of card UID</td><td>Card UID</td></tr>
</table>
</td></tr>
<tr><td>Type B</td><td>0x0B,ATQB,0x04,PUPI
<table>
<tr><td>0x0B</td><td>ATQB</td><td>0x04</td><td>PUPI</td></tr>
<tr><td>Type B</td><td>1 byte(the first four bits means maximum frame length, after four bits means protocol type)</td><td>Length of PUPI</td><td>4 bytes PUPI data</td></tr>
</table>
</td></tr>
<tr><td>Felica card</td><td>
<table>
<tr><td>0x0C</td><td>0x00</td><td>0x10</td><td>felica_id</td><td>pad_id</td></tr>
<tr><td>Felica</td><td>1 byte reserve</td><td>16 bytes data</td><td>8 bytes felica id</td><td>8bytes pad id</td></tr>
</table>
</td></tr>
</table>

Topaz card	0x0D	ATQA	id
	Topaz	1 byte	Topaz card ID

4.5 Mifare card section

4.5.1 GeneralAuthenticate

Synopsis:

public int GeneralAuthenticate(int blockNum, int keyType, byte[] key)

Parameters:

blockNum	IN	block number which will do operation
keyType	IN	key's type
key	IN	Key's byte code

Description:

To do authenticate for operation block

Example:

More information, please follow sample code.

Returns:

SUCCESS	Successful
Others	fail

4.5.2 ReadBinary (removed in new SDK)

Synopsis:

public int ReadBinary(int blockNum, byte[] data, int size)

Parameters:

blockNum	IN	block number which will do operation
data	OUT	return data which will be read
size	IN	size of how many data will be read

Description:

This function use to read block data

Example:

More information, please follow sample code.

Returns:

SUCCESS	Successful
Others	fail

4.5.3 ClassicBlockInitial

Synopsis:

public int ClassicBlockInitial(int blockNum)

Parameters:

blockNum	IN	block number which will do operation

Description:

To do initial of specify block

Example:

More information, please follow sample code.

Returns:

SUCCESS	Successful
Others	fail

4.5.4 ClassicReadValue

Synopsis:

public int ClassicReadValue(int blockNum, int[] valueAmount);

Parameters:

blockNum	IN	block number which will do operation
valueAmount	OUT	output block value into array

Description:

To read block value from card

Example:

More information, please follow sample code.

Returns:

For the error code, please follow error section

4.5.5 ClassicStoreBlock

Synopsis:

public int ClassicStoreBlock(int blockNum, int valueAmount);

Parameters:

blockNum	IN	block number which will do operation
valueAmount	IN	output block value into array

Description:

To write value into block

Example:

More information, please follow sample code.

Returns:

For the error code, please follow error section

4.5.6 ClassicIncrement

Synopsis:

public int ClassicIncrement(int blockNum, int valueAmount);

Parameters:

blockNum	IN	block number which will do operation
valueAmount	IN	Plus the value of the required

Description:

Plus the value operation

Example:

More information, please follow sample code.

Returns:

For the error code, please follow error section

4.5.7 ClassicDecrement

Synopsis:

public int ClassicDecrement(int blockNum, int valueAmount);

Parameters:

blockNum	IN	block number which will do operation
valueAmount	IN	Minus the value of the required

Description:

Minus the value operation

Example:

More information, please follow sample code.

Returns:

For the error code, please follow error section

4.6 Reader monitor function

4.6.1 OnInsertHeadSet

Synopsis:

public void OnInsertHeadSet ();

Parameters:

NULL

Description:

When audio jack insert to Phone then will execution this function

Example:

More information, please follow sample code.

4.6.2 OnInsertHeadSet

Synopsis:
public void OnPullHeadSet ();
Parameters:
NULL
Description:
When audio jack plug out from Phone then will execution this function
Example:
More information, please follow sample code.

4.7 Buzzer control function

4.7.1 TrunOffAutoBuzzer

Synopsis:
public void `TrunOffAutoBuzzer`();
Parameters:
NULL
Description:
When the function FTNFC_connect is return succeed ,the buzzer will beep.
This function using to turn off the beep.
Example:
More information, please follow sample code.

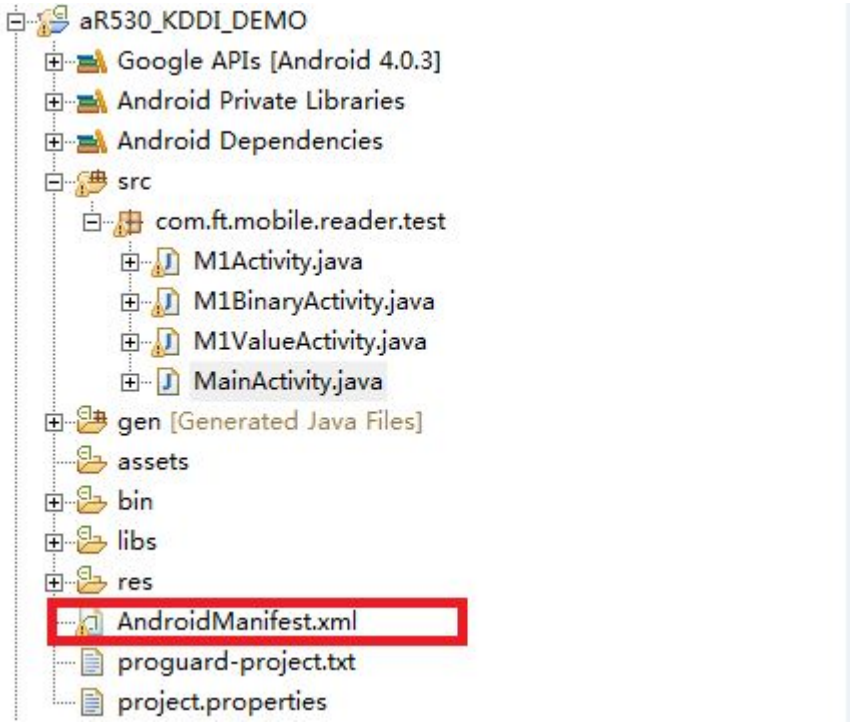
4.7.2 BuzzerBeep

Synopsis:
public void `BuzzerBeep`();
Parameters:
NULL
Description:
This function using to control buzzer beep.
Example:
More information, please follow sample code.

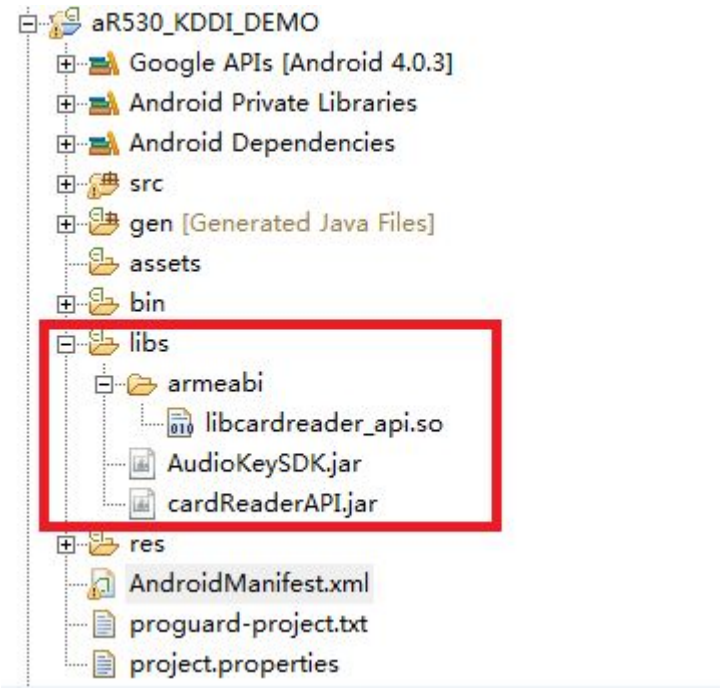
Chapter 5. Development notice item

5.1 Android project setting

1. Create a project and find **AndroidManifest.xml** in the root directory of the android project. Add four access related audio device and external storage device in a specific file, showed below:



2. Add aR530 dynamic library and JAR into your project, first, please do create libs directory, put AudioKeySDK.jar, cardReaderAPI.jar into libs directory, after create armeabi directory, put libcardreader_api.so into armeabi directory, as shown below:



5.2 Do initialization in your code

Before you call API to do operation, need to do initialization at first.

1. Call Initial() method which in card class (com.ft.mobile.reader.card), To call this API, you will need input context, as shown below:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    requestWindowFeature(Window.FEATURE_NO_TITLE);
    setContentView(R.layout.activity_main);

    myCard = new Card();
    myCard.Initial(this);
```

2. After get the audio jack broadcast, call HandShake() method to create connection between phone/table and aR530. The method can be found in (com.ft.mobile.reader.card). as shown below:

```
private BroadcastReceiver receiver = new BroadcastReceiver() {
    public void onReceive(Context context, Intent intent) {
        if (intent.hasExtra("state")) {
            if (intent.getIntExtra("state", 0) == 0) {
                tvConnectState.setText("not connected.");
                clearSpinner();
                adlg.setMessage("Device is not connected");
                adlg.show();
            } else if (intent.getIntExtra("state", 0) == 1) {
                adlg.hide();
                new HandshakeTask().execute();
            }
        }
    }
};
```

3. After created communication, you can call provided method in Card class(com.ft.mobile.reader.card) to do operation with aR530.

Chapter 6. Appendix and terms

Abbreviations and terms	Description
NFC	Near field communication (NFC) is a set of ideas and technologies that enable smartphones and other devices to establish radio communication with each other by touching them together or bringing them into proximity, typically a distance of 10 cm (3.9 in) or less.
aR530	Product name, which provided by Feitian, support ISO 14443 Type A and Type B/Felica/Mifare/Topaz cards