SUNMI ID Card Developer docs

(Including PRC ID card、Contactless & contacted card etc.)

©2020 Shanghai SUNMI Technology Co., Ltd. All rights reserved.

Doc Release Notes

Docs Version	Updated	SDK Version	Details	Author
1.0.0	2020/05/16	1.0.11	The original version	曾冬阳(Darren Zeng) Translated by Logan SUN

1. Introduction

1) SUNMI ID Card Service mainly serves hardware modules such as ID card reader and non-access card reader, and provides service interface to facilitate the integration of third-party applications, mainly serving SUNMI self-service devices.

ID card reader: China-Vision 100U & Mintek (some models).
Contactless card reader: Mintek related contactless reader module.

1.1. Hardware description

1. Single ID card reader (USB）: MT3F-00-A3DA & 100U module;
2. Contactless card reader (USB）: contactless CPU card, memory card (m1 s50 / 70) & financial IC card (the USB);
3. ID & contactless 2-in-1 card reader (USB): MT4D-00-A344— M1 + CPU card;
4. Contactless card reader (Serial port）—CPU card, memory card (m1 s50 / 70), financial IC &ID card, social security card (all use serial port 3 / ttyS3)
    1) Contactless (M1 + CPU) 2-in-1 card reader—MT3F-00-R131-SHSM;
    2) Contactless (M1 + CPU + EMID) 3-in-1 module—MT0A-00-R1D0-SM;

1.2. Contactless card Call process description

1.2.1. Contactless memory card API call flow

After the connection is established successfully, the general operation process based on the Mifare one card test as following:

1. Read and write data: establish a connection → seek card → verify → read and write data → abort the card
2. Read and write block value: establish connection → seek card → verify → read and write value → abort the card

1.2.2. Contactless CPU card API call flow

Based on the contactless CPU card in the software, the general operation process is:

Successful connection establishment → open card → send APDU command → set card status to halt → abort the card

1.2.3. Contact CPU card API call process

Based on the contact CPU card in the software, the general operation process is:

Successful connection establishment → Power-on reset → Send APDU command → Card power-off

2. Integration and interface description

2.1. Introduction

SUNMI ID Card Service provides an operational interface package that includes identification and anti-counterfeiting of ID Card documents such as ID cards, contactless cards, and passport OCR.

Resource file structure:
res/java/com.sunmi.idcardservice.IDCardInfo.java— 2nd generation ID card information;
res/aidl/com.sunmi.idcardservice.IDCardInfo.aidl— 2nd generation ID card information aidl file;
res/aidl/com.sunmi.idcardservice.CardCallback.aidl— 2nd generation ID card automatic card reading callback interface;
res/aidl/com.sunmi.idcardservice.IDCardServiceAidl.aidl— ID Card Service main interface;
interfaceres/aidl/com.sunmi.idcardservice.MiFareCardAidl.aidl— Contactless operation interface;

2.2. ID Card Service

2.2.1. Permission statement

The following permissions need to be added in AndroidManifest.xml:

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="com.sunmi.idcard.permission.ACCESS_SERVICE"/>
```

2.2.2. Connection service

Connection service：

```
private IDCardServiceAidl mService;
private ServiceConnection mConnection = new ServiceConnection() {
    @Override
    public void onServiceConnected(ComponentName name, IBinder service) {
        mService = IDCardServiceAidl.Stub.asInterface(service);
        Log.e("id_card", " ID Service is connected");
    }

    @Override
    public void onServiceDisconnected(ComponentName name) {
        mService = null;
        Log.e("id_card", "ID Service is disconnected");
    }
};
private void bindIDCardSerivice() {
    Intent intent = new Intent();
    Intent.setPackage("com.sunmi.idcardservice");
    Intent.setAction("com.sunmi.idcard");
    bindService(intent,mConnection,Context.BIND_AUTO_CREATE);
}
```

2.2.3. Unbinding service

Unbinding service：

```
unbindService(mConnection);
```

2.3. Interface definition

2.3.1. ID Card Service main interface (ID card)

ID Card Service main interface：IDCardServiceAidl.aidl

No.	Function and Description
1	**IDCardInfo readCard**() Read ID card information
2	void **readCardAuto**(CardCallback call) Read ID card information automatically
3	void **cancelAutoReading**() Cancel automatic ID card reading
4	**MiFareCardAidl getMiFareCardService**() Get contactless operation object

1. Read ID card information

Function: **IDCardInfo readCard**()
Parameters: none.
Return:
ID card information, field description: name: name, gender: gender, nation: country / nationality, birthday: date of birth, address: address, idCard: ID card number, department: issuing authority, startDate: effective date , EndDate: validity period, imageAddress: ID card avatar path, firstFinger, secondFinger: ID card fingerprint information, idUid: ID card UID(top 16 bit);

2. Read ID card information automatically

Function: void **readCardAuto**(**CardCallback** call)
Parameters: none.
call --> Automatic reading result callback, detailed refer to **3.1.**
Return: none.
Description:
After the application receives the result, it is necessary to cancel the automatic card reading immediately when there is no need to read the ID card.

3. Cancel automatic ID card reading

Function: void **cancelAutoReading**()
Parameters: none.
Returns: none.
Description: none.

4. Get contactless operation object

Function: **MiFareCardAidl getMiFareCardService**()
Parameters: none.
return:
When the return is empty, it means that the device hardware does not support contactless card operation; when it is not empty, refer to **2.3.2.**

Description: None.

2.3.2. Contactless Card Operation

Contactless operation interface: MiFareCardAidl.aidl

No.	Function and Description
1	int **rfhalt**() Contactless card - Set contactless card to halt state
2	int **openCupCard**(byte[] cardtype, byte[] snrlen, byte[] snr, byte[] atrlen, byte[] atr) Contactless CPU card function - Activate contactless card
3	int **exchangePro**(byte[] cmd, int cmdlen, byte[] resp, int[] resplen) Contactless CPU card function - Contactless card command interaction
4	int **getCPUCardState**(byte[] cardState) Contactless CPU card function - Get contactless CPU card status
5	int **rfCard**(byte[] cardtype, byte[] cardID) Contactless memory card - Activate contactless memory card
6	int **rfAuthEntication**(int mode, int nsecno, byte[] key) Contactless memory card - Contactless memory card certification sector
7	int **rfRead**(int nblock, byte[] readdata) Contactless memory card - Contactless memory card to read data
8	int **rfWrite**(int nblock, byte[] writedata) Contactless memory card - Write data with contactless memory card
9	int **rfReadVal**(int nblock, int[] readvalue) Contactless memory card - Get the value of the specified block address of the contactless memory card
10	int **rfInitVal**(int nblock, int writevalue) Contactless memory card - Set the value of the specified block address of the contactless memory card
11	int **rfIncrement**(int nblock, int incvalue) Contactless card - operation on the data value for the contactless memory card specified block address
12	int **rfDecrement**(int nblock, int decvalue) Contactless memory card - Perform a decrement operation on the data at the specified block address
13	int **readNAN**(int nCardType, byte[] Cardno, byte[] CardName, byte[] lpErrMsg) Contactless IC card - Read the financial IC card number and name
14	int **readSBInfo**(byte[] info, byte[] err) Contactless social Security Card - Read Social Security Card Information
15	int **getEMID**(byte[] datalen, byte[] data) Contactless ID card - read EMID card number
16	int **beep**(int delaytime) Contactless module - Buzzer

1. Contactless card - Set contactless card to halt state

Function: int **rfhalt**()
Parameters: none.
Return: <> 0: error, = 0: correct.
Description: None.

2. Contactless CPU card function - Activate contactless card

Function: int **openCupCard** (byte [] cardtype, byte [] snrlen, byte [] snr, byte [] atrlen, byte [] atr)
Parameters:
cardtype --> card type, cardtype [0]: 0AH-A card; 0BH-B card; 00H-not obtained.
snrlen --> Card UID length, unit: byte.
snr --> Card UID.
atrlen --> ATR length, unit: byte.
atr --> ATR response data.
Return: <> 0: error, = 0: correct.
Description:
The reader checks whether the card enters the induction zone and activates the card entering the induction zone. If there is a card in the induction area but the activation fails, the reader will directly return to "activation failed" without continuing to search for the card.
For example:

```
byte []cardtype = new byte[8];
byte []snrlen = new byte[8];
byte []snr = new byte[10];
byte []cardinfo = new byte[200];
byte []infolen =new byte[10];
openCupCard(delaytime, cardtype, snrlen, snr, infolen, cardinfo);
```

3. Contactless CPU card function - Contactless card command interaction

Functions: int **exchangePro** (byte [] cmd, int cmdlen, byte [] resp, int [] resplen)
parameter:
cmd --> data.
cmdlen --> byte length of sent data, unit: byte.
resp --> The returned data.
resplen --> The length of the returned data, unit: bytes.
Return: <> 0: error, = 0: correct.
Description:
This interface needs to activate the contactless CPU card (refer to **2.3.2.- 2.**)
For example:

```
int st;
byte[] send_hex = {0x01,0x02,0x03,0x04};
byte[] resp_hex = new byte[512];
int[] resplen = new int[10];
st = exchangePro(send_hex, 4, resp_hex, resplen);
```

4. Contactless CPU card function - Get contactless CPU card status

Function: int **getCPUCardState** (byte [] cardState)
parameter:
cardState --> Contactless CPU card status, 0: no card; 1: one card; 2: multiple cards.
Return: <> 0: error, = 0: correct.
Description:
Only contactless ID card module support, serial contactless module does not support.
For example:

```
byte[] status = new byte[5];
st = getCPUCardState(status);
```

5. Contactless memory card - Activate contactless memory card

Function: int **rfCard** (byte [] cardtype, byte [] cardID)
parameter:
cardtype --> card type, cardtype [0]: 0AH-TypeA card; 0BH-TypeB card.
cardID --> Card UID, 4 bytes.
Return: <> 0: error, = 0: correct.
Description:
The reader checks whether the card enters the induction zone and activates the card entering the induction zone. If there is a card in the induction area but the activation fails, the reader will directly return to "activation failed" without continuing to search for the card.
For example：

```
byte[] snr = new byte[20];
byte[] cardtype = new byte[8];
st = rfCard(cardtype, snr);
```

6. Contactless memory card - Contactless memory card certification sector

Function: int rfAuthEntication (int mode, int nsecno, byte [] key)
parameter:
mode --> Authentication mode, 0: KEYA mode; 1: KEYB mode.
nsecno --> block address (sector number * 4 + block address).
key --> Authentication key, 6 bytes.
Return: <> 0: error, = 0: correct.
Description:
This interface needs to activate the contactless card first (refer to **2.3.2.- 5**.).
For example：

```
byte[] key = new byte[10];
key[0] = (byte)0xFF;
key[1] = (byte)0xFF;
key[2] = (byte)0xFF;
key[3] = (byte)0xFF;
key[4] = (byte)0xFF;
key[5] = (byte)0xFF;
st = rfAuthEntication(0,4,key);
```

7. Contactless memory card - Contactless memory card to read data

Function: int rfRead (int nblock, byte [] readdata)
parameter:
nblock --> block address (sector number * 4 + block address).
readdate --> read data.
Return: <> 0: error, = 0: correct.
Description:
This interface need to activate contactless cards (refer to **2.3.2.- 5**.), contactless cards with authentication key, needs to be called after the authentication (refer to: **2.3.2. - 6.**).
For example：

```
byte[] rdata = new byte[32];
st = rfRead(4,rdata);
```

8. Contactless memory card - Write data with contactless memory card

Function: int **rfWrite** (int nblock, byte [] writedata)

parameter:
nblock --> block address (sector number * 4 + block address).
writedata --> written data.
Return: <> 0: error, = 0: correct.
Description:
This interface need to activate contactless cards (Refer to **2.3.2.- 5**.), contactless cards with authentication key, needs to be called after the authentication (Refer to: **2.3.2. - 6.**).
For example:

```
byte[] key = new byte[16];
Key[0] = 0x00 .....
st = rfWrite(4,wdata);
```

9. Contactless memory card - Get the value of the specified block address of the contactless memory card

Function: int **rfReadVal** (int nblock, int [] readvalue)
parameter:
nblock --> block address (sector number * 4 + block address).
readvalue --> The value read.
Return: <> 0: error, = 0: correct.
Description:
This interface needs to activate the contactless card first (Refer to **2.3.2.- 5**.), and the contactless card with key authentication needs to be authenticated and then called (Refer to **2.3.2. - 6.**).
For example:

```
int[] ivalue = new int[50];
st = rfReadVal(4,ivalue);
```

10. Contactless memory card - Set the value of the specified block address of the contactless memory card

Function: int **rfInitVal** (int nblock, int writevalue)
parameter:
nblock --> block address (sector number * 4 + block address).
writevalue --> The value written.
Return: <> 0: error, = 0: correct.
Description:
This interface needs to activate the contactless card first (Refer to **2.3.2.- 5**.), and the contactless card with key authentication needs to be authenticated and then called (Refer to **2.3.2. - 6.**).
For example:

```
st = rfInitVal(4,100)
```

11. Contactless card - Operation on the data value for the contactless memory card specified block address

Function: int **rfIncrement** (int nblock, int incvalue)
parameter:
nblock --> block address (sector number * 4 + block address).
incvalue --> numeric value.
Return: <> 0: error, = 0: correct.
Description:

This interface need to activate contactless cards (Refer to **2.3.2.- 5**.), contactless cards with authentication key, needs to be called after the authentication (Refer to: **2.3.2. - 6.**).

For example：

st = rfIncrement(4,20);

12. Contactless memory card - Perform a decrement operation on the data at the specified block address

Function: int **rfDecrement** (int nblock, int decvalue)

parameter:

nblock --> block address (sector number * 4 + block address).

decvalue --> numeric value.

Return: <> 0: error, = 0: correct.

Description:

This interface needs to activate contactless cards (Refer to **2.3.2.- 5**.), contactless cards with authentication key, needs to be called after the authentication (Refer to: **2.3.2. - 6.**).

For example：

st = rfDecrement(4,20);

13. Contactless IC card - Read the financial IC card number and name

Function: int **readNAN** (int nCardType, byte [] Cardno, byte [] CardName, byte [] lpErrMsg)

parameter:

nCardtype --> Card type, 0: contact CPU card, 1: contactless CPU card (currently only supports contactless IC cards).

Cardno --> Financial IC card number.

CardName --> Financial IC card name.

lpErrMsg --> error message.

Return: <> 0: error, = 0: correct.

Description:

Currently, only contactless IC cards are supported.

For example：

```
byte []szCardNo=new byte[512];
byte []szName=new byte[512];
byte []szErrinfo=new byte[1024];
int nCardType = 0x01;
st = readNAN(nCardType, szCardNo, szName, szErrinfo);
```

14. Contactless social Security Card - Read Social Security Card Information

Function: int **readSBInfo** (byte [] info, byte [] err)

parameter:

info --> social security card information, such as the output format: Card | ID number | Name | Sex | National | date of birth.

err --> exception information.

Return: <> 0: error, = 0: correct.

Description:

ID card contactless module does not support, only support serial port module.

For example：

```
byte[] szSocialCardBasicInfo = new byte[1024];
byte[] szErrinfo = new byte[1024];
int st = readSBInfo(szSocialCardBasicInfo, szErrinfo);
```

```
if (et == 0){
    String info = new String(szSocialCardBasicInfo, "gb2312").trim();
}
```

15. Contactless ID card - Read EMID card number

Function: int **getEMID** (byte [] datalen, byte [] data)
parameter:
datalen --> EMID card number length.
data --> EMID card number.
Return: <> 0: error, = 0: correct.
Description:
ID card contactless module does not support, only support serial port module.
For example：

```
byte[] dLen = new byte[8];
byte[] id = new byte[20];
int st = getEMID(dLen, id);
```

16. Contactless module- Buzzer

Function: int **beep** (int delaytime)
parameter:
delaytime --> Tweet duration (unit: 100ms)
Return: <> 0: error, = 0: correct.
Description: none.

3. Callback instructions

3.1. Card Callback

No.	Function and Description
1	void **getCardData**(**IDCardInfo** info**, int** code) throws RemoteException Automatic identification card callback

1. Card callback-successful operation

Function: void **getCardData**(**IDCardInfo** info**, int** code) throws RemoteException

parameter:

info --> ID card information, field description: name: name, gender: gender, nation: country / nationality, birthday: date of birth, address: address, idCard: ID card number, department: issuing authority, startDate: effective date , EndDate: validity period, imageAddress: ID card avatar path, firstFinger, secondFinger: ID card fingerprint information, idUid: ID card UID;

code --> Reading result: 1: Successful connection, -99: Connection failure, 10: Successful reading, -10: Unsuccessful reading, 99: Unplug USB device, 100: No permission for USB device.

Description:

None.

4. Appendix

4.1. Code explanation of contactless card

The function returns: -1: device does not support the contactless card;
-2: the device does not support this method;

StatusH	StatusL	Explanation	Troubleshooting suggestions	Error type
00H	00H	Command execution responds correctly	/	/
00H	01H	Optional function interfaces or parameters are not supported (including optional communication baud rate parameters, display function, etc.)	Check if the device supports this interface	Hardware error
Device operation Return code				
00H	11H	Command timeout	Check if the communication port is normal	Hardware error
00H	12H	Invalid communication handle	Check whether the device is connected or the communication port is normal	Software error
00H	13H	Open communication port error	Check whether the communication port is normal or whether the connection method is wrong	Software error
00H	14H	Communication port is already occupied	Turn off the device and turn on again to connect the device	Operation error
00H	15H	Get COM communication port status error	Check if this operation is performed under serial connection	Operation error
00H	16H	Get COM communication port status error	Check if this operation is performed under serial connection	Operation error
00H	17H	Error reading data from reader	Check if the communication port is normal	Software error
00H	18H	Error writing data to the reader	Check if the communication port is normal	Software error
Send and receive data packets Return code				
00H	23H	BCC error	Check if the communication port is normal or the protocol is wrong	Software error
00H	24H	The data length of the command is greater than the maximum length	Check if the command length is wrong (greater than 65530) or the magnetic stripe card reading timeout is too large (greater than 5000ms)	Software error
00H	25H	Data value error	Check whether the sending command is correct. If the range of non-receiving card bonus (M1) is exceeded, whether the type of contact memory card is set correctly, the storage path of the ID photo is correct, etc.	Software error
00H	26H	STX error	Check if the communication port is normal	Software error
00H	27H	ETX error	Check if the communication port is normal	Software error
Contactless card operation Return code				
30H	01H	It does not support contactless card users	Check whether the card reader in the area, whether the device supports a non-contact or contactless card is damaged.	Hardware error Or Operation error
30H	04H	Contactless user card is not activated	Activate contactless card	Software error
30H	05H	Contactless user card activation failed	Check whether the card reader in the area, or put the wrong type of card.	Operation error

30H	06H	No response from operating contactless user card (waiting for timeout)	Check if the APDU command is correct	Software error
30H	07H	There is an error in operating the contactless user card data	Check if the command is correct	Software error
30H	08H	Contactless user card halt failed	Remove the card at the specified time	Operation error
30H	09H	There are multiple cards in the induction zone	/	Operation error
Financial IC card operation Return code				
60H	01H	Does not support logical operations	/	Software error
60H	20H	Card type is incorrect (card status 6A82)	The instruction is wrong or the operation file is not found (please check if the operation file path is correct)	Software error
60H	21H	Insufficient balance (card status 9401)	Check if the card has a balance	Software error
60H	22H	Card function is not supported (card status 6A81)	Check if the card supports this function	Software error
60H	23H	Deduction failed (card status 9302)	Invalid MAC address	Software error
60H	30H	Card is not enabled	Card type is wrong	Software error
60H	31H	Card is not valid	/	Software error
60H	32H	Transaction details have no such record	/	Software error
60H	33H	Transaction details are not processed	/	Software error
60H	40H	Need to do anti-pull treatment	/	Software error
60H	41H	Error in anti-pull process, not the original card	/	Software error
60H	42H	Trading is interrupted, no capital loss	/	Software error