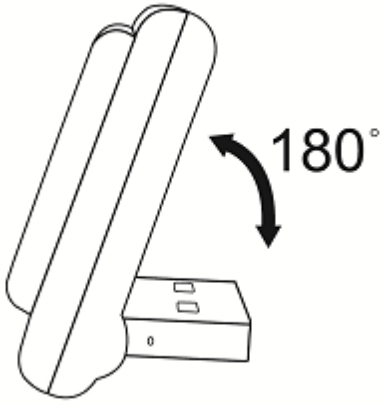

PNF-101 NFC Reader
Android Integration Guide

Version 1.0

10/19/2017

Revision History

Revision	Date	Description	By
1.0	2017/10/19	First version	Clyde Wang

Table of Contents

1 Overview

The “IT-101MU” is the code name for PNF-101, based on this framework, you have two ways to integrate the Bluetooth smart card.

- Using the IT-101MUActivity framework: Simple and fast especially for developing NEW smart card application.
- Using IT-101MU Reader Class directly: If you already have a framework such as javax.smartcardio, you can use IT-101MU Reader class to extend your framework.

You have two ways to send APDU commands to your smart card.

- Implement your Card class: it’s a good way to manage your APDU commands such as SelectFile, ReadRecord and VerifyPin etc., in one card class.
- Sending command directly: You still can choose to send command directly by the reader class.

2 Using the IT-101MUActivity framework

The “IT-101MU Demo” example is a good example of using IT-101MUActivity. Here's a snippet of the source codes:

Note: The source codes of IT-101MUActivity is at the util/myIT-101MUActivity.java of this project. You can modify it to fit your requirement if needed.

```
public class MainActivity extends IT-101MUActivity {

      public void onCreate(Bundle savedInstanceState) {

      ...

      btnReadCard = (Button)findViewById(R.id.btnReadCard);
      btnReadCard.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                  //DebugCommand();
                  //start the SCThread
                  ExecuteCard();
            }
      });
   ...
}
```

```
class SCThread extends IT-101MUActivity.SCThread {


    @Override
    protected void RunCommands() throws SCException {
        MifareClassic card = null;
        //Connect card and create the build-in MifareClassic instance
            card = (MifareClassic)ConnectCard("com.infothink.smartcard.MifareClassic", Card.PROTOCOL_ANY);


        String CardID = card.GetCardID();
      //Power-off the card
        card.Disconnect();
    }
    @Override
    protected void handleSCMessage(Message msg) {
        Events what = Events.values()[msg.what];
        switch (what)
        {
        case WAITING_READER_CONNECT:
            mMessage.setText("Connecting reader...");
            //Show the Progress Bar
            mWaitCard.setVisibility(View.VISIBLE);
        break;
        case READER_CONNECTED:
            //Hide the Progress Bar
            mWaitCard.setVisibility(View.GONE);
        break;
        case WAITING_CARD_INSERT:
            mMessage.setText("Connecting card... ");
            mWaitCard.setVisibility(View.VISIBLE);
            break;
        case WAITING_CARD_REMOVE:
            mMessage.setText("Please remove card.");
            mWaitCard.setVisibility(View.VISIBLE);
            break;
        case CARD_INSERTED:
```

```
            mMessage.setText("Waiting commands.");
            mWaitCard.setVisibility(View.GONE);
            break;
        case CARD_REMOVED:
        case RUN_COMMANDS_AFTER:
            mWaitCard.setVisibility(View.GONE);
        break;
        case RUN_COMMANDS_BEFORE:
            mMessage.setText("Reading Card.");
        break;
        case RUN_COMMANDS_ERROR:
            mWaitCard.setVisibility(View.GONE);
            //Get the Exception object
            Exception e = (Exception)msg.obj;
            mMessage.setText(e.getMessage());
        break;
        }
        }
    @Override
    protected IT-101MUActivity.SCThread createSCThread() {
        return new SCThread();
    }
}
```

3 Using IT-101MU Reader Class directly

IT-101MU class is the reader, and through its methods you can control its various functions, such as open/close connection, power on/off smart card or send APUD commands.
The example code is here:

```
private static final String ACTION_SC_PERMISSION =
"com.infothink.smartcard.USB_PERMISSION";


Log.v(TAG, "Creating IT-101MU Reader");
PendingIntent mPermissionIntent = PendingIntent.getBroadcast(this, 0, new
        Intent(ACTION_SC_PERMISSION), 0);
UsbManager mUsbManager = (UsbManager)getSystemService(Context.USB_SERVICE);
```

```
IT-101MU reader = new IT-101MU(mPermissionIntent, mUsbManage);
try {
    MifareClassic card = null;
    reader.logLevel(IT-101MU.LOG_APDU);
    reader.Open();

    reader.WaitCardEvent(IT-101MU.CARD_EVENT_DETECED);

    card = (EMVCard)ConnectCard("com.infothink.smartcard.MifareClassic",
Card.PROTOCOL_ANY);
    String CardID = card.GetCardID();
    //Power-off the card
    reader.DisConnectCard();
} catch (SCException e) {
    e.printStackTrace();
} finally {
    reader.Close();
}
```

The methods of IT-101MU are here:

- created IT-101MU instance

  ```
  IT-101MU(PendingIntent intent, UsbManager manager)
  ex:
  IT-101MU reader = new IT-101MU(mPermissionIntent, mUsbManager);
  ```

- Log APDU command and respond on the console

  ```
  reader.logLevel(IT-101MU.LOG_APDU);
  ```

- Open the Reader connection

  ```
  reader.Open();
  ```

- Close the Reader connection

```
reader.Close();
```

- Is Card presented in the reader

```
reader.isCardPresent();
```

- Waiting the card inserted to the reader

```
reader.WaitCardEvent(IT-101MU.CARD_EVENT_DETECTED);
```

- Waiting the card removed from the reader

```
reader.WaitCardEvent(IT-101MU.CARD_EVENT_REMOVED);
```

- Connect card (Power-on) and get ATR

```
byte[] ATR = reader.ConnectCard(Card.PROTOCOL_ANY);
```

- Connect card and create the card class (refer to 2.3 Implement your Card class)

```
MifareClassic card = (MifareClassic) mReader.ConnectCard("com.infothink.smartcard.MifareClassic ",
Card.PROTOCOL_ANY);
```

- Transmit the APDU command

```
byte[] data = reader.Transmit(apdu);
```

- Disconnect (Power-off) the card

```
reader.DisConnectCard();
```

4 Build-in Mifare Card class functions

- Key Authentication

```
private final byte KEY_TYPE_A = 0x60;
private final byte KEY_TYPE_B = 0x61;
byte[] Key = new byte[] {(byte) 0xFF, (byte) 0xFF, (byte) 0xFF, (byte) 0xFF, (byte) 0xFF, (byte) 0xFF};
reader.KeyAuthority(1, KEY_TYPE_A, Key); // Authenticate Key A on Block 1
```

- Read block data

```
byte[] data = reader.ReadBlock(1); // read block 1 data
```

- Write block data

```
reader.WriteBlock(1, data); // write block 1 data
```

- Value-Type data operation

```
int Value = ReadValue(1); // read value from block 1
reader.WriteValue(1, 1000); // write value 1000 to block 1

reader.IncreaseValue(1, 1); // increase value 1 to block 1
reader.TransferValue(1); // commit the block 1 value operation

reader.DecreaseValue(1, 2); // decrease value 2 to block 1
reader.RestoreValue(1); // rollback the block 1 value operation
```

5 Implement your Card class

Card class is created under the com.infothink.smartcard package to let user implement all the APDU commands logic in it. Developer extend s the Card class will need to implement IdentifyCard() method. IdentifyCard() is called when you use ConnectCard() to get your card instance. To make sure the card inserted is right by return true. Usually you'll get your card instance by writing the code like this:

card = (EMVCard)ConnectCard("com.infothink.EMVCard", Card.PROTOCOL_ANY);

Here is an example card class:

```
public class EMVCard extends Card {
    protected byte[] myATR;

    public EMVCard() {
    }

    @Override
    public boolean IdentifyCard(byte[] ATR){
        myATR = ATR;
        //Select the application ID
        if (SelectPSE() != null)
            return true;
        else
            return false;
        }

        public void SelectFile(byte[]id) throws SCException {
            if (aid == null)
                throw new NullPointerException("File id must not be null");

        byte[] cmd = new byte[id.length + 6];

        cmd[0] = 0x00;
        cmd[1] = (byte) 0xA4;
        cmd[2] = 0x04;
        cmd[3] = 0x00;
        cmd[4] = (byte) data.length;
        System.arraycopy(data, 0, cmd, 5, data.length);
        cmd[cmd.length -1] = 0x00;

        //Transmit the APDU command and expect 0x90 0x00 is returned
        return Transmit(cmd, 0x9000, "SELECT FILE");
    }
```

```
    public byte[] SelectPSE() throws SCException {...}

    public byte[] ReadRecord(byte SFI, byte no, int len) throws SCException {...}

}
```

In the card class you can call Transmit() to send APDU commands and get respond data. Sending APDU is like this:

Transmit(cmd, 0x9000, "SELECT FILE");

APDU commands status codes: SW1 (0x90), SW2 (0x00) will be check automatically right after the respond data is got. If status code is the same as you expected, respond data stripped by first two bytes is returned. Otherwise an “SW1/2 error” exception is thrown. You can still get the last command status code by call GetSW1SW2() or GetSW1SW2Bytes() anytime.

If it’s needed to power-off the smart card you can call DisConnectCard().

6 Sending command directly without Card class

```
private static final String ACTION_SC_PERMISSION =
"com.infothink.smartcard.USB_PERMISSION";

private void SendCommandWithOutCardClass() {
    PendingIntent mPermissionIntent = PendingIntent.getBroadcast(this, 0, new
        Intent(ACTION_SC_PERMISSION), 0);
    UsbManager mUsbManager = (UsbManager)getSystemService(Context.USB_SERVICE);

    IT-101MU reader = new IT-101MU(mPermissionIntent, mUsbManager, false);

    SendMessage(MESSAGE_STATUS, "Connecting Reader ...");

    try {

        reader.logLevel(IT-101MU.LOG_APDU); //unmark it if need debug
        reader.Open();

        if (!reader.isCardPresent())
            throw new SCException("Card not present!");

        SendMessage(MESSAGE_STATUS, "Connecting Card ...");

        byte[] ATR = reader.ConnectCard(Card.PROTOCOL_ANY);
        byte[] data = reader.Transmit(new byte[] {0x00, (byte) 0xA4, 0x00, 0x00});

        reader.DisConnectCard();

        return;
    } catch (Exception e) {
        e.printStackTrace();
        throw new SCException(e.getMessage());
    } finally {
        reader.Close();
    }
}
```