Developer documentation for SunmiPrinter

目 录

Introduction

V1, V1s, P1, p1-4g, T1, T1 mini and T2 have built-in cache thermosensitive printers, allowing the App to print heat sensitive tickets directly.There are two specifications for the built-in printer of sunmi products:

- 80mm wide, with cutting knife. Compatible with 58mm, T1 is equipped with this printer.
- 58mm wide, no cutting knife. V1 and P1 carry the printer.

App developers can call the built-in thermal printer in three ways:

1. The printer is invoked through AIDL
2. The printer is invoked through the built-in virtual bluetooth device
3. The H5 Web page invokes the printer through the JS bridge

Taking V1 device as an example, three call modes are introduced. This scheme is also applicable to V1s, P1, p1-4g, T1, T1 mini and T2 devices.

1. Connect to print service

1.1. Approach 1: AIDL

1.1.1. Introduction of AIDL

AIDL is the abbreviation of the Android Interface Definition language, which is a description language of the communication Interface between processes within Android, through which we can define the communication Interface between processes.Sunmi AIDL provides encapsulated common print instructions for developers to quickly access Sunmi printers, while Sunmi printer also supports ESC/POS instruction sets.

1.1.2. Use AIDL

The following five steps can be used to establish a connection:

1. Add the AIDL files that come with the resource files in the project (some machines also contain Java files).
2. Implement ServiceConnection in the code class that controls printing.
3. Call the ApplicationContext. BindService (), and transmission in the ServiceConnection implementation. Note: bindservice are blocking calls, means that the call did not immediately after the completion of the binding is successful, must be the serviceConnected shall prevail.
4. In the ServiceConnection. OnServiceConnected () implementation, you will receive an IBinder instance (called Service).
5. You can now print by calling the various methods defined in the IWoyouService interface.

Binding service example

```
Private ServiceConnection connService = new ServiceConnection() {
@ Override
Public void onServiceDisconnected(ComponentName name) {
    Toast. MakeText (this, "the service disconnected", Toast. LENGTH_LONG), show ();
    SetButtonEnable (false);
    WoyouService = null;
     Try {
        Thread.sleep (2000);
    } catch (interruptedexe) {
        // TODO auto-generated catch block
```

```
            E.p rintStackTrace ();
        }
        Binding ();
}
@ Override
Public void onServiceConnected (the ComponentName name, IBinder service) {
        WoyouService = IWoyouService. Stub. AsInterface (service);
        SetButtonEnable (true);
        Try {
            ServiceVersion = woyouService. GetServiceVersion ();
             Info. SetText ("service version :" + serviceVersion + "\n");
        } catch (RemoteException e) {
            // TODO auto-generated catch block
            E.p rintStackTrace ();
        }
}
Private void Binding () {
    Intent Intent = new Intent ();
    Intent. SetPackage (" woyou. Aidlservice. Jiuiv5 ");
    Intent. SetAction (" woyou. Aidlservice. Jiuiv5. IWoyouService ");
    BindService (intent, connService, Context. BIND_AUTO_CREATE);
}
```

1.1.3. AIDL interface definition description

After establishing a connection to the print service via AIDL, the commonly used print instruction interface encapsulated by IWoyouService can be invoked (IWoyouService interface function can be invoked only in AIDL connection mode).

1.1.3.1. Printer initialization and setting

Serial number	The method of
1	Void **printerInit( )** Printer initialization
2	void **printerSelfChecking(ICallback** callback**)** Printer self check

1. Printer initialization

Function: void printerInit()

Note: reset the printer's logical program (for example, typography, bolding, etc.), but do not clear the cache data, so the unfinished printing will continue after the reset.

2. Printer self check

Function: void printerSelfChecking (ICallback callback)

Parameter: callback→result callback.

Example:

```
WoyouService. PrinterSelfChecking (the callback);
```

1.1.3.2. Obtain equipment and printer information

Serial number	The method of
1	String **getPrinterSerialNo( )** Get the printer board serial number
2	String **getPrinterModal( )** Get printer interface (print head size)
3	String getPrinterVersion () Get the printer firmware version number
4	int **getPrintedLength(ICallback** callback**)**

	Get the print head print length
5	Build.MODEL (constant) Device name
6	int **updatePrinterState()** Get the latest status of the printer
7	String getServiceVersion () Get the print service version number

1. Get the printer interface (type of print head)

Function: String getPrinterModal ()

Return value: t1mini-58;T1mini - 80.

2. Get the latest status of the printer

Function: String updatePrinterState ()

The return value:

1 printer is normal

2 Printer update status

3 Get status exception

4 out of paper

5 overheating

6 open cover

7 cutter abnormal

8 cutter recovery

9 No black mark detected

Note: this interface is not currently supported by V1 devices.

1.1.3.3. ESC/POS instructions

Serial number	The method of
1	void **sendRAWData (**byte[] data, **ICallback** callback**)** Print the ESC/POS format instruction

1. Print the ESC/POS format instruction

Function: void sendRAWData(byte[] data, ICallback callback)

Parameters:

Data→ESC/POS instruction.

Callback→**result callback.**

Note: for instructions, refer to the ESC/POS instruction set.

Example:

```
    WoyouService. SendRAWData (new byte [] {0 x1b, 0 x45, 0 x01}, callback);    / 1B,45, 01 are font bold instructions
```

1.1.3.4. Black label printing

Serial number	Method/instruction
1	int **getPrinterMode**() Gets printer mode
2	int **getPrinterBBMDistance**() Paper distance!
3	Relevant mode Settings, please set in the system "Settings" print Settings.→

1. Gets printer mode

Function: int getPrinterMode()

The return value:

0→general mode;

1→Black label mode;

Note: only T1 and T2 devices are supported.

2. Gets black mark mode printer automatic paper distance

Function: int getPrinterBBMDistance()

Return value: paper distance (dot line).

Note: only T1 and T2 devices are supported.

1.1.3.5. Text printing

Serial number	Method/instruction
1	void **setAlignment**(int alignment, **ICallback** callback) Set alignment mode

2	void **setFontName**(String typeface, **ICallback** callback) Set print font (not currently supported)
3	void **setFontSize**(float fontsize, **ICallback** callback) Set font size
4	Esc /pos instruction: font bold {0x1B, 0x45, 0x1}, unbold {0x1B, 0x45, 0x0} Setting and unbolting
5	void **printText**(String text, **ICallback** callback) Print the word
6	void **printTextWithFont**(String text, String typeface, float fontsize, **ICallback** callback) Print the text of the specified font size
7	void **printOriginalText** **(**String text, **ICallback** callback**)** Print vector text

1. Set alignment mode

Function: void setalimajority (int alignment, ICallback callback)

Parameters:

Alignment alignment:

0→is left, 1→is center, 2→is right.

Callback→ **result callback.**

Note: global method, which affects subsequent printing, cancels the Settings when the printer is initialized.

Example:

```
WoyouService. SetAlignment (1, callback);
```

2. Set print font (not currently supported)

Function: void settfonname (String typeface, ICallback callback)

Parameters:

The typeface font name currently supports the font "gh", which is an equal-width Chinese font.

callback→result callback.

Note: global method, which affects subsequent printing, cancels Settings when the printer is initialized (fonts are not currently supported in existing versions).

Example:

```
WoyouService. SetFontName (" gh ", the callback);
```

3. Set font size

Function: void settfonsize (fonfloat tsize, ICallback callback)

Parameters:

Fontsize→fontsize.

callback→result callback.

Note: global method, to influence after printing, the initialization can cancel Settings, the font size is beyond standard international instruction way of printing, adjust the font size will affect the character width, number per line character will change too, so the monospaced font typesetting may be formed by disorder.

Example:

```
WoyouService. SetFontSize (36, callback);
```

4. Setting and unbolting

Instruction: font bold {0x1B, 0x45, 0x1}, unbold {0x1B, 0x45, 0x0}

Note: refer to this article 1.1.3.3. ESC/POS instruction.

Example:

```
WoyouService. SendRAWData (new byte [] {0 x1b, 0 x45, 0 x0}, callback);    // cancel font bold instruction
```

5. Print the word

Function: void printText(String text, in ICallback callback)

Parameters:

text → When the printed content is less than one or more lines, you need to add a line break "\n" at the end of the content to print it immediately, otherwise it will be cached in the buffer.

callback→result callback.

Note: to change the style of the printed text (such as alignment, font size, bold, etc.), set it before calling the printText method.

Example:

```
WoyouService. SetAlignment (1, callback);
WoyouService. SetFontSize (36, callback);
Woyouservice.printtext (" sunmi \n", callback);
```

6. Print the text of the specified font size

Function: void printTextWithFont(String text, String typeface, float fontsize, ICallback callback)

Parameters:

text → When the printed content is less than one or more lines, you need to add a line break "\n" at the end of the content to print it immediately, otherwise it will be cached in the buffer.

typeface → font name (currently not supported by the existing version,).→

fontsize → fontsize is only valid for this method.

callback→**result callback.**

Note: font Settings are only valid for this time.

Example:

```
WoyouService.PrintTextWithFont("sunmi\n","",36,callback);
```

7. Print vector text

Function: printvoid originaltext (String text, ICallback callback)

Parameters:

text → When the printed content is less than one or more lines, you need to add a line break "\n" at the end of the content to print it immediately, otherwise it will be cached in the buffer.

callback→**result callback.**

The return value:

Note: the text is output as the vector text width, that is, each character is different in width.

Example:

```
WoyouService. PrintOriginalText ("κ ρ χ κ μ ν κ λ ρ κ ν κ ν μ ρ τ υ ϕ\n", the callback);
```

1.1.3.6. Form printing

Serial number	Method/instruction
1	void **printColumnsText**(String[] colsTextArr, int[] colsWidthArr, int[] colsAlign, **ICallback** callback) Print a row of the table (no Arabic characters are supported)

2	void **printColumnsString**(String[] colsTextArr, int[] colsWidthArr, int[] colsAlign, **ICallback** callback) Print a row of the table to specify column width and alignment

1. Print a row of the table (not support Arabic)

Function: printvoid columnstext (String[] colsTextArr, int[] colsWidthArr, int[] colsAlign, ICallback callback)

Parameters:

ColsTextArr→ column text string array.→

ColsWidthArr →column width array, in English characters, each Chinese character takes up two English characters, each width greater than 0.→

ColsAlign →alignment: 0 to the left, 1 to the center, and 2 to the right.→

callback→result callback.

Note: the length of the array of the three parameters should be the same. If the width of colsText[I] is greater than that of colsWidth[I], the text should be newlined and Arabic characters are not supported.

Example:

```
   woyouService.printColumnsText(new String[]{"商米","商米","商米"}, new int[]{4,4,8}, new int[]{1,1,1},callback);
```

2. Print a row of the table to specify column width and alignment

Function: printvoid columnsstring (String[] colsTextArr, int[] colsWidthArr, int[] colsAlign, ICallback callback)

Parameters:

ColsTextArr→column text string array.

ColsWidthArr→column width weight is the proportion of each column.

ColsAlign→alignment: 0 to the left, 1 to the center, and 2 to the right.

callback→result callback.

Note: the array length of the three parameters should be the same. If the width of colsText[I] is greater than that of colsWidth[I], the text should be replaced.

Example:

```
   woyouService.printColumnsString(new String[]{"商米","商米","商米"}, new int[]{1,1,2}, new int[]{1,1,1},callback);
```

1.1.3.7. Print pictures

Serial number	The method of
1	void **printBitmap**(Bitmap bitmap, **ICallback** callback) Print pictures
2	void **printBitmapCustom**(Bitmap bitmap, int type, **ICallback** callback) Print picture (2)

1. **Print pictures**

Function: printvoid Bitmap (Bitmap Bitmap, ICallback callback)

Parameters:

Bitmap →image bitmap object.

callback→result callback.

Note: the maximum width is 384 pixels, beyond which the exception function cannot be printed and callback is called back.

Example:

```
WoyouService. PrintBitmap (bitmap, callback);
```

2. **Print picture (2)**

Function: printvoid bitmapcustom (Bitmap Bitmap,int type ICallback callback)

Parameters:

Bitmap image bitmap object (maximum width: 384 pixels, image over 1M cannot be printed).→

Type currently has two printing methods:→

0 →-printBitmap();

1→ black-and-white image with a threshold of 200

2 →grayscale images→

callback→result callback.

Note: the maximum width is 384 pixels, beyond which the exception function cannot be printed and callback is called back.

Example:

```
WoyouService. PrintBitmapCustom (bitmap, callback);
```

1.1.3.8. One-dimensional code and two-dimensional code printing

Serial number	The method of
1	void **printBarCode**(String data, int symbology, int height, int width, int textPosition, **ICallback** callback) Print barcode
2	void **printQRCode**(String data, int modulesize, int errorlevel, **ICallback** callback) Print QRcode

1. Print barcode

Function: printvoid barcode (String data, int symbology, int height, int width, int textPosition, ICallback callback)

Parameters:

Data →one-dimensional code content.

Symbology→ barcode type (0-8) :

0 → UPC-A

1 → UPC-E

2 → JAN13(EAN13)

3 → JAN8(EAN8)

4 → CODE39

5 → ITF

6 → CODABAR

7 → CODE93

8 → CODE128

Height →bar code height, value 1-255, default: 162.

Width →bar code width, value 2-6, default: 2.

TextPosition →text Position (0-3) :→

0 →no print text

1→ above the barcode

2→ below the barcode

3→ both

callback→result callback.

Note: maximum print content per encoding.

Example:

```
    woyouService.printBarCode("1234567890", 8, 162, 2, 2, callback);
```

2. Print qr code

Function: printvoid qrcode (String data, int modulesize, int errorlevel, ICallback callback)

Parameters:

Data →qr code content.

Modulesize →qr code block size, unit: dot, values 4 to 16.

Errorlevel qr code error correction level (0-3) :→

0→ error correction level L (7%)

1→error correction level M (15%)

2→ error correction level Q (25%)

3→error correction level H (30%)

callback→result callback.

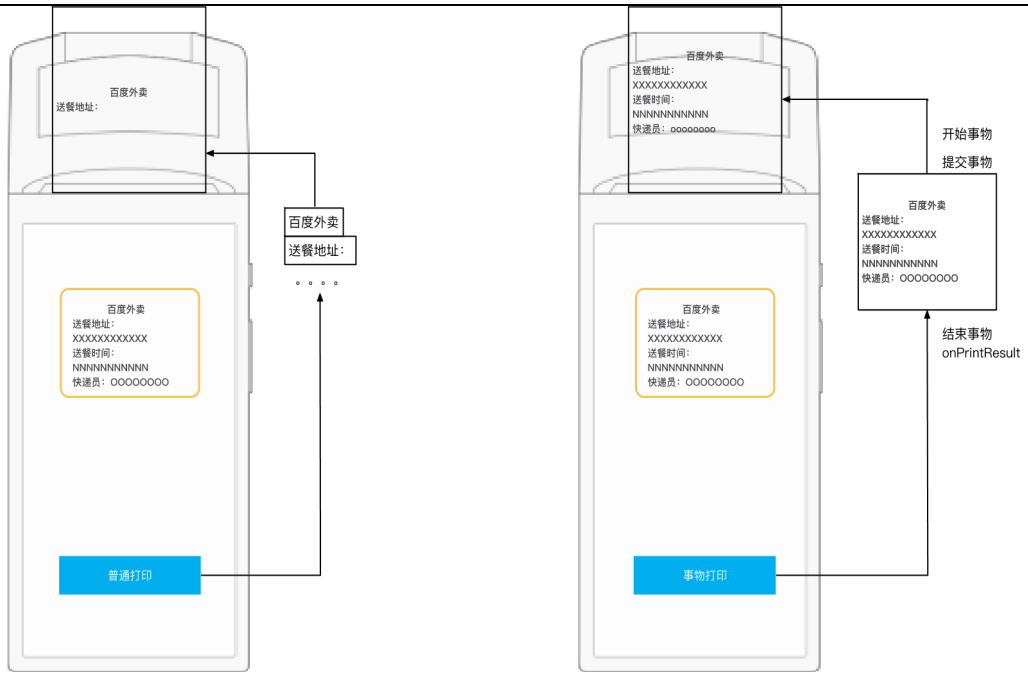
Note: in the normal printing state, after calling this method, the printing will be output directly. Refer to the qr code description for more information.

Example:

```
woyouService.printQrCode("商米科技", 4, 3, callback);
```

1.1.3.9. Transaction printing

Transaction print mode is suitable for the need to control print content and print the result feedback (whether print out receipts) demand, this pattern is equivalent to establish a transaction queue buffer, when developers into transaction print mode, will open a transaction queue, can increase the print method to the queue, the printer will not print the content immediately, after the transaction is committed, the printer will in turn in the queue tasks, perform end will receive feedback on the results of the transaction.



Notes for transaction printing:

1. After entering the buffer (transaction) printing, the successful submission will return a successful result, but if the printer is abnormal, such as paper shortage, overheating, etc., all the instruction tasks in the commit transaction will be lost, and will return the abnormal status. If the printer is abnormal before the execution of a single task or during execution, the order will not be printed;
2. When the command print and buffer (transaction) print are used alternately, if the printer is abnormal, the contents printed by the instruction will not be cleared!
3. After entering transaction print mode (also using other interface methods in 1.1.3.x. to output content) but not printing immediately, the output will be cached in the buffer, when calling exitPrinterBuffer(), commitPrinterBuffer(), etc. Will print.
4. Transaction print results callback onPrintResult in ICallback method (int code, String MSG) method (there will be a time consuming, to physical print out the paper, such as frequent use transaction print line is not recommended, will affect the printing speed, recommend the use of the whole receipts transactions printing), the corresponding code is as follows:
   a) 0 →printed successfully, MSG is null;
   b) 1 →printing failures, MSG as "woyou. Aidlservice. Jiuv5. ERROR_ACTION".
5. An example of the entire transaction printing pseudocode is as follows:

EnterPrinterBuffer (true) PrintText (/ * something * /) PrintBitmap bitmap resource shots (/ * * /) ..... (other print-related methods in 1.1.3.) CommitPrinterBuffer () / / exitPrinterBuffer (true) / / commitPrinterBufferWithCallback (the callback) / ExitPrinterBufferWithCallback (true, callback) / / in the callback OnPrintResult back off printing results

6. Description of specific methods:

Serial number	The method of
1	void **commitPrint**(TransBean[] tranBean, **ICallback** callback) The lib package transaction printing special interface
2	void **enterPrinterBuffer**(boolean clean) Enter transaction print mode
3	void **exitPrinterBuffer**(boolean commit) Exit transaction print mode
4	void **exitPrinterBufferWithCallback**(boolean commit, **ICallback** callback) Exit transaction print mode and callback results
5	void **commitPrinterBuffer**() Commit transaction printing
6	void **commitPrinterBufferWithCallback**(**ICallback** callback) Commit the transaction to print and callback the result

1. The lib package transaction printing special interface

Function: void commitPrint (TranBean[] TranBean, ICallback callback)

Parameters:

TranBean →prints the list of tasks.

callback→result callback.

Note: users using lib call this interface to start printing things, which requires printing services of more than 2.0.8.

Example:

```
woyouService.commitPrint(tranBean, callback);
```

2. Enter transaction mode

Function: void enterPrinterBuffer(Boolean clear)

Parameters:

Clear→clears buffer contents:

True→clears the last transaction to print uncommitted content;

False→does not clear that the last transaction printed uncommitted content, and the next commit will contain the last.

Note: enable and enter transaction printing mode, where data will not be printed immediately until the submission or exit of the submission. Print service 2.0.8 or more.

Example:

```
woyouService.enterPrinterBuffer(false);
```

3. Exit transaction mode

Function: void exitPrinterBuffer(Boolean commit)

Parameters:

commit →Whether commit prints out the buffer content:

True →prints everything in the transaction queue

False →does not print content in the transaction queue, which is saved until the next commit

Note: print service is required at least 2.0.8.

Example:

```
WoyouService. ExitPrinterBuffer (true);
```

4. Exit transaction print mode and callback results

Function: void exitPrinterBuffer(Boolean commit, ICallback callback)

Parameters:

commit →Whether commit prints out the buffer content:

True→ prints everything in the transaction queue

False →does not print content in the transaction queue, which is saved until the next commit

callback→result callback.

Note: it should be in P1 and V1s equipment and printing service 2.3.2 or above.

Example:

```
woyouService.exitPrinterBuffer(true);
```

5. **Commit transaction printing**

Function: void commitPrinterBuffer()

Note: all content in the transaction queue is submitted and printed, and it is still in transaction printing mode after that. The printing service needs to be more than 2.0.8.

Example:

```
woyouService.commitPrinterBuffer();
```

6. **Commit the transaction to print and callback the result**

Function: void commitPrinterBufferWithCallbacka (ICallback callback)

Parameters:

callback→result callback.

Note: it should be in P1 and V1s equipment and printing service 2.3.2 or above.

Example:

```
woyouService.commitPrinterBufferWithCallback(callback);
```

1.1.3.10. Paper is related

Serial number	The method of
1	void **lineWrap**(int n, **ICallback** callback) The printer runs on n lines of paper

1. **Print n lines**

Function: void lineWrap(int n, ICallback callback)

Parameters:

N→ lines of paper.

callback→result callback.

Note: it is mandatory to change lines. After finishing printing, n lines of paper will be taken.

Example:

```
woyouService.printBitmap(3, callback);
```

1.1.3.11. Cutter (paper cutting) correlation

Serial number	The method of
1	void **cutPaper**(**ICallback** callback) Paper cutting
2	int **getCutPagerTimes**() Obtain the cumulative number of cutting tool

1. Paper cutting

Function: void cutPaper (ICallback callback)

Parameters:

callback→result callback.

Note: equipment support is required.

Example:

```
woyouService.cutPager(callback);
```

2. Gets the number of cuts

Function: int getCutPaperTimes ()

Return value: number of cuts.

Note: equipment support is required.

1.1.3.12. Cashbox correlation

Serial number	The method of
1	void **openDrawer**(**ICallback** callback) Open the coffers
2	int **getOpenDrawerTimes**() Get the accumulated opening times of the cashbox
3	int **getDrawerStatus()**

	get status of the cashbox

1. Open the coffers

Function: void openDrawer (ICallback callback)

Parameters:

callback→result callback.

Note: equipment support is required.

Example:getting status of a cashbox from the interface depends on whether the cashbox provides query function.

```
woyouService.openDrawer(callback);
```

2. Get the accumulated opening times of the money box

Function: int getCutPaperTimes ()

Return value: the money box has been opened multiple times.

Note: equipment support is required.

3. Get status of the cashbox

Function: int getDrawerStatus **()**

Return value: open/close

Note: get status of a cashbox from the interface depends on whether the cashbox provices query fuction.

1.2.Mode 2: virtual bluetooth

1.2.1. Introduction to virtual bluetooth

In the list of V1's bluetooth devices, you can see a bluetooth device "InnerPrinter" that has been paired and will always exist.Virtual bluetooth supports ESC/POS instruction sets.

Some of the special instructions are sunmi custom instructions, such as:

function	instruction
Opening instruction	Byte [5] : 0x10 0x14 0x00 0x00 0x00 0x00
Cutting knife command to cut all	Byte [4] : 0x1d 0x56 0x42 0x00
Cutting instruction (leave a bit uncut on the left)	Byte [4] : 0x1d 0x56 0x41 0x00

1.2.2. Use of virtual bluetooth

1. Connect to the bluetooth device.
2. Splicing the instruction and text content into Bytes.
3. Send it to InnerPrinter.
4. The underlying printing service drives the printing device to complete printing.

Note: BluetoothUtil is a bluetooth tool class, which is used to connect the InnerPrinter of the virtual bluetooth device.

1.2.2.1. Tools such as BluetoothUtil and standard bluetooth connection tools

```
public class BluetoothUtil {
private static final UUID PRINTER_UUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
private static final String Innerprinter_Address = "00:11:22:33:44:55";
public static BluetoothAdapter getBTAdapter() {
 return BluetoothAdapter.getDefaultAdapter();
}

public static BluetoothDevice getDevice(BluetoothAdapter bluetoothAdapter) {
    BluetoothDevice innerprinter_device = null;
    Set<BluetoothDevice> devices = bluetoothAdapter.getBondedDevices();
    for (BluetoothDevice device : devices) {
      if (device.getAddress().equals(Innerprinter_Address)) {
          innerprinter_device = device;
          break;
      }
    }
    return innerprinter_device;
}

public static BluetoothSocket getSocket(BluetoothDevice device) throws IOException {
    BluetoothSocket socket = device.createRfcommSocketToServiceRecord(PRINTER_UUID);
    socket.connect();
    return socket;
}

public static void sendData(byte[] bytes, BluetoothSocket socket) throws IOException {
    OutputStream out = socket.getOutputStream();
    out.write(bytes, 0, bytes.length);
    out.close();
}
```

1.2.2.2. Bluetooth connection printing service case

1: Get BluetoothAdapter
BluetoothAdapter btAdapter = BluetoothUtil. GetBTAdapter (); **If (btAdapter == = null) {** Toast. MakeText (getBaseContext (), "both Please Open Bluetooth!"Toast), LENGTH_LONG), show (); **The return;** }
2. Get Sunmi's printer innerbluetoothdevice
BluetoothDevice device = BluetoothUtil. GetDevice (btAdapter); **If (device == null) {** Toast.maketext (getBaseContext(),"Please Make Sure Bluetooth has InnterPrinter!"Toast), LENGTH_LONG), show (); **The return;** }
3: Generate a order data, user add data here
Byte [] data = null;
4: Using InnerPrinter print data
BluetoothSocket socket = null; The socket = BluetoothUtil. GetSocket (device); BluetoothUtil. SendData publishes the event (data, socket);

1.2.2.3. Matters needing attention

Bluetooth permission declaration needs to be added in the App project to use bluetooth devices:

The < USES - permission android: name = "android. Permission. BLUETOOTH" / > < USES - the permission of the android: name = "android. Permission. BLUETOOTH_ADMIN" / >

1.3.Approach 3: JS in HTML

1.3.1. The introduction of JS

Calling the printer through JS essentially means operating android's native code in HTML through the JS bridge, and printing through bluetooth printing or AIDL. (the sunmi printer itself is not a network printer, and the web application cannot communicate with the printer directly, so it needs to have applications that accept data on android.)

1.3.2. Use HTML

The following is the specific invocation process:

1. Define the methods to use in android in HTML.

```
Document. GetElementsByTagName (' a ') [0] addEventListener (' click ',
The function () {
        Var a = "wellcome to sunmi";    Javascript: lee. FunAndroid (a);
       Return false;
}, false);
```

2. Initialize the WebView.

```
WebView mWebView = (WebView) findViewById(rivd.w v_view);
 // set the code
MWebView. GetSettings (.) setDefaultTextEncodingName (" utf-8 ");
 / / support js
MWebView. GetSettings (.) setJavaScriptEnabled (true);MWebView. SetWebChromeClient
(new WebChromeClient ());
```

3. Initialize the print service

```
Intent = new Intent();
Intent. SetPackage (" woyou. Aidlservice. Jiuiv5 ");Intent. SetAction (" woyou. Aidlservice.
Jiuiv5. IWoyouService ");
StartService (intent); / / Start printer service
BindService (intent, connService, Context. BIND_AUTO_CREATE);
```

4. Add listening to the WebView and call it in the onPageFinished callback method

WebView. AddJavascriptInterface (new JsObject (), 'lee');

In the JsObject class, a method to be specified in HTML is defined by the @javascriptinterface, in which a small ticket is printed by calling AIDL.

```
// add a page to listen for the class
```

```
MWebView. SetWebViewClient (new WebViewClientDemo ());
WebViewClientDemo extends WebViewClient {
    @ Override
    Public Boolean shouldOverrideUrlLoading(WebView, String url) {
        // when opening a new link, use the current WebView instead of the other
    browsers in the system
        The loadUrl (url);
        Return true;
    }
    @ Override
    Public void onPageFinished(WebView view, String url) {
        Super. OnPageFinished (view, url);
        / * *
        * sign up for JavascriptInterface, where the name "lee" is optional
        * if you use "lee, "in HTML, just use the name of the lee.method ()
        * you can call the same name method in MyJavascriptInterface with the same
parameters
        * /
        MWebView. AddJavascriptInterface (new JsObject (), "lee");
    }
}
The class JsObject {
    @ JavascriptInterface
    Public void funAndroid(final String I) {
        Toast.maketext (getApplicationContext(), "calling the local method funAndroid by
    js." + I, toast.length_short). Show ();
        Try {
            WoyouService. PrinterSelfChecking (the callback); //Using AIDL to print
        something.
        } catch (RemoteException e) {
            E.p rintStackTrace ();
        }
    }
}
```

5. Load the HTML file, and when you click on the button in the HTML, you print a small ticket

```
// load HTML containing js
MWebView. LoadData (" ", "text/HTML", null);
file:///android_asset/test.html mWebView. LoadUrl (" ");// here is the page where your business HTML is located
```

2. State feedback

2.1. Printer status feedback

By broadcast: the user needs to establish a broadcast receiver to listen to the broadcast.

function	The Action
Printer ready	"Woyou. Aidlservice. Jiuv5. INIT_ACTION"
Printer update	"Woyou. Aidlservice. Jiuv5. FIRMWARE_UPDATING_ACITON"
can print	"Woyou. Aidlservice. Jiuv5. NORMAL_ACTION"
print errors	"Woyou. Aidlservice. Jiuv5. ERROR_ACTION"
No paper	"Woyou. Aidlservice. Jiuv5. OUT_OF_PAPER_ACTION"
The print head is overheating	"Woyou. Aidlservice. Jiuv5. OVER_HEATING_ACITON"
Print head temperature returned to normal	"Woyou. Aidlservice. Jiuv5. NORMAL_HEATING_ACITON"
Cover opened	"Woyou. Aidlservice. Jiuv5. COVER_OPEN_ACTION"
cover off anomaly	"Woyou. Aidlservice. Jiuv5. COVER_ERROR_ACTION"
Abnormal cutter 1- clip cutter	"Woyou. Aidlservice. Jiuv5. KNIFE_ERROR_ACTION_1"
Abnormal cutter 2- cutter repair	"Woyou. Aidlservice. Jiuv5. KNIFE_ERROR_ACTION_2"
Printer firmware is being upgraded	"Woyou. Aidlservice. Jiuv5. FIRMWARE_UPDATING_ACITON"
Printer firmware upgrade failed	"Woyou. Aidlservice. Jiuv5. FIRMWARE_FAILURE_ACITON"

Undetected printer	"Woyou. Aidlservice. Jiuv5. PRINTER_NON_EXISTENT_ACITON"
No black mark detected	"Woyou. Aidlservice. Jiuv5. BLACKLABEL_NON_EXISTENT_ACITON"

2.2. Instruction callback instructions

2.2.1. ICallback interface method description

Serial number	The method of
1	Void onRunResult (Boolean isSuccess) Instruction execution result
2	Void onReturnString (String result) The instruction execution result returns a String
3	Void onRaiseException(int code, String MSG) Exception message return
4	Void onPrinterResult(int code, String MSG) Print feedback for things

1. **Instruction execution result**

Function: void onRunResult (Boolean isSuccess)

Parameters:

IsSuccess →execution results:

True →execution succeeded

False →execution failed

Note: this interface returns processing results that refer to command processing execution results, rather than printing out paper processing results.

2. **The instruction execution result returns a String**

Function: void onRrturnString (String result)

Parameters:

Result →execution result

3. **Exception message return**

Function: void onRaiseException (int code, String MSG)

Parameters:

Code →exception code.

MSG →exception description.

Note: see abnormal information comparison.

4. **Print feedback for things**

Function: void onPrintresult (int code, String MSG)

Parameters:

Code →status code:

0 →success.

1 →failed.

MSG is null when it succeeds, and an exception description when it fails.→

Note: the return result of this interface is the result of instruction processing and physical printing output (there will be a certain amount of time, to wait for physical printing out of the paper).

2.2.2. Example Callback object

```
New ICallback. Stub () {
    @ Override
    Public void onRunResult(Boolean b) throws RemoteException {
        // instruction execution results
    }
    @ Override
    Public void onReturnString(String s) throws RemoteException {
        // instruction execution results return String
    }
    @ Override
    Public void onRaiseException(int I, String s) throws RemoteException {
        // exception information returned
    }
    @ Override
    Public void onPrintResult(int I, String s) throws RemoteException {
        // print feedback on things
    }
}
```

2.2.3. Exception information comparison table

code	MSG
- 1	"Command is not support,index #"; // # represents the # byte error
2 -	"# encoding is not support"; // # represents the # byte error

- 3	"Oops,add task failed (the buffer of the task queue is 10M),please try later";
4 -	"The create command failed";
- 5	"Illegal parameter";
- 6	"Param found null pointer"

3. Introduction to printing service parameters

打印纸的宽度 58　　有效打印宽度 48mm　　有效打印像素点 384

Note: Sunmi printer supports 58mm and 80mm printing paper. This document takes 58mm printing paper as a case to illustrate the supporting parameters of the printer. 80mm printing paper has similar specifications.

A piece of 58 printing paper has a width of 58mm and an effective printing width of 48mm. The effective print width ACTS as 384 pixels. The depth of V1 paper slot is 40mm, and the maximum diameter of the paper can be 40mm

3.1. Printer resolution

The printer resolution is 205DPI, and the calculation formula is as follows

DPI = = 8 384 dots / 48 mm dots / 1 mm = 205 dots/in = 205

3.2. Check whether there is a printer

T1 devices are divided into two versions: printer hardware and non-printer hardware. Users can judge from the software through the printer query interface. Only for T1 device users.

Query interface:

Function: Settings.Global. GetInt (getContentResolver(), "sunmi_printer", 0);

Parameter: fixed value.

The return value:

0 →no printer

1. →There is a printer

-1 →is in the query

3.3. Font description

The default font is 24, 24*24 matrix in Chinese and 12*24 matrix in English.

3.4. Qr code description

The sunmi printer prints qr codes, each of which has 4 pixel points (less than 4 scan codes cannot be resolved). Maximum support for version19 (93*93) mode.

3.5. Picture description

The sunmi printer supports a maximum print image size of 1M with a maximum support width of 384 pixels. If the user needs to print an image that is more than 1M or 384 pixels wide, the image needs to be compressed manually.

```
If (mBitmap == null){
      / * * * * * images within 1M * * * * /
      MBitmap = BitmapFactory. DecodeResource (getResources (), R.r aw. Sunmi);
      / * * * * * images over 1M * * * * * * * * * * * * /
      MBitmap1 = BitmapFactory. DecodeResource (getResources (), R.r aw. Sunmi1);
}
Compressed picture * * * * * * * * * * * * * * * * /
Double gh = (double) mBitmap. GetWidth () / 384;
If (mBitmap1 == = null){
     MBitmap2 = bitmaputils.zoombitmap (mBitmap1, 384, (int)(mbitmap.getheight ()/gh));
}
/ * * * * * * * compressed * * * * * * * * /
Try {
      WoyouService. SetAlignment (1, callback);
      WoyouService. PrintBitmap (mBitmap, callback);
/ /   WoyouService. PrintBitmap (mBitmap2, callback);
      WoyouService. LineWrap (3, null);
} catch (RemoteException e) {
      // TODO auto-generated catch block
      E.p rintStackTrace ();// when the picture is too large and not compressed, the printing service will feedback the exception
}
```

3.6. Barcode description

V1_printerservice2.1.13V1 (slightly different for different models)

coding	instructions
code39	Print up to 13 Numbers
code93	Print up to 17 Numbers

code128	Print up to 15 Numbers, Code128 is divided into three categories: {A,{B,{C;Class A: capital letters, Numbers, punctuation, etc. Class C: pure Numbers;The default class B encoding, to use class A and class C encoding, add "{A" and "{C" before the content, such as "{A2344A", "{C123123", "{A1A{B13Bxc{C12".
ean8	8 digit number (last check digit), effective length 8 digits
ean13	The effective length is 13 digits, the last of which is the check digit
ITF	You want to enter a number that is valid for less than 14 bits and must be even
Codebar	Requires 0-9 and 6 special characters, up to 18 digits printed
UPC - E	8 digit Numbers (last check digit)
UPC - A	12 digit Numbers (last check digit)

3.7. Character set Settings

The default charracter setting is simplified Chinese.

The command to switch the character set is as follows:

Close multibyte(single byte fit for Europe area) : 0x1C 0x2E

Open multibyte(fit for east Asia) : 0x1C 0x26

Set Single byte character Set: 0x1B 0x74 [parameter]

Single byte character set parameter list:

[para] [character set] [area]

0 "CP437";

2 "CP850";

3 "CP860";

4 "CP863";

5 "CP865";

13 "CP857";

14 "CP737";

15 "CP928";

16 "Windows - 1252";

17 "CP866";

18 "CP852";

19 "CP858";

```
21 "CP874";
33 "Windows - 775";
34 "CP855";
36 "CP862";
37 "CP864";
254 "CP855";
Set multibyte character Set: 1C 43 [parameter]
```

Multibyte character set parameter list:

```
[para] [character set]
0 x00 | | 0 x48 "GB18030";
/ / 0 x01 | | 0 x49 "BIG5".
0 x02 | | 0 x50 "KSC5601";
(byte) 0 XFF "utf-8";
```

3.8. Black label printing instructions

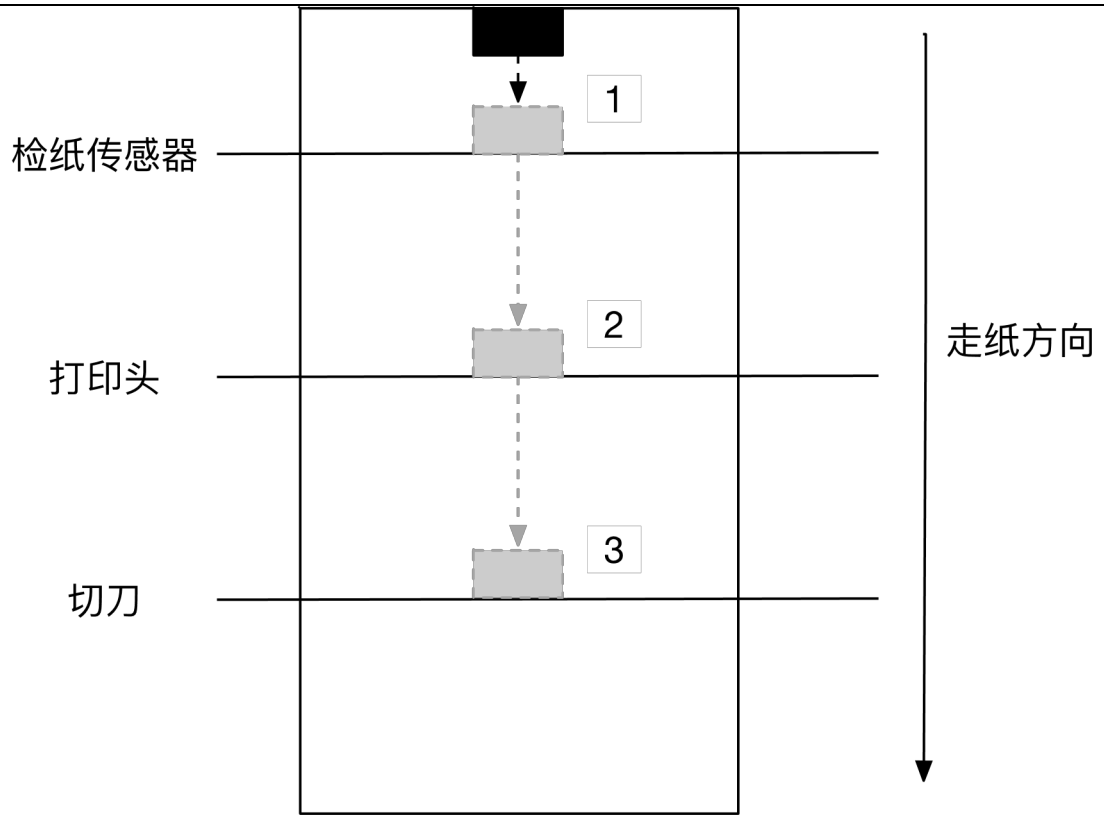
Shangmi T1 can be printed using standard black label printing paper.

: to the requirement of black label printing paper maker m T1 black label print and black label printers, T1, m hardware does not have sensors to detect black mark, but using the test paper sensor of reflectivity is less than 6% (700-1000 nm range infrared wavelengths) materials as the principle of the paperless, made a similar black label printing function. Because of the different principles, the printing mode of simi T1 black label cannot accurately locate the black label position like the black label printer, and there is a certain error.

Requirement for black mark position: black mark needs to be in the middle position of paper level before it can be detected by paper sensor.

Realization principle of quotient meter T1 black label:

Test paper sensor and the print head and the cutter location is not a level, black label paper will first after test paper sensor position (figure 1), and then through the print head position (figure 2), finally reach the cutter location position (figure 3) paper cutting out of the storehouse.



In black mark mode, after the paper sensor detects that there is no paper, the printer will automatically walk a distance of 7mm. If the paper sensor fails to detect the paper before the end of the walk, the paper will be treated as if there is no paper. If there is any paper detected before you go, follow the black label.

When the printing content overrides the black mark position, the printing will continue. That is, when the black mark reaches the position in figure 2, there is still a printing task.

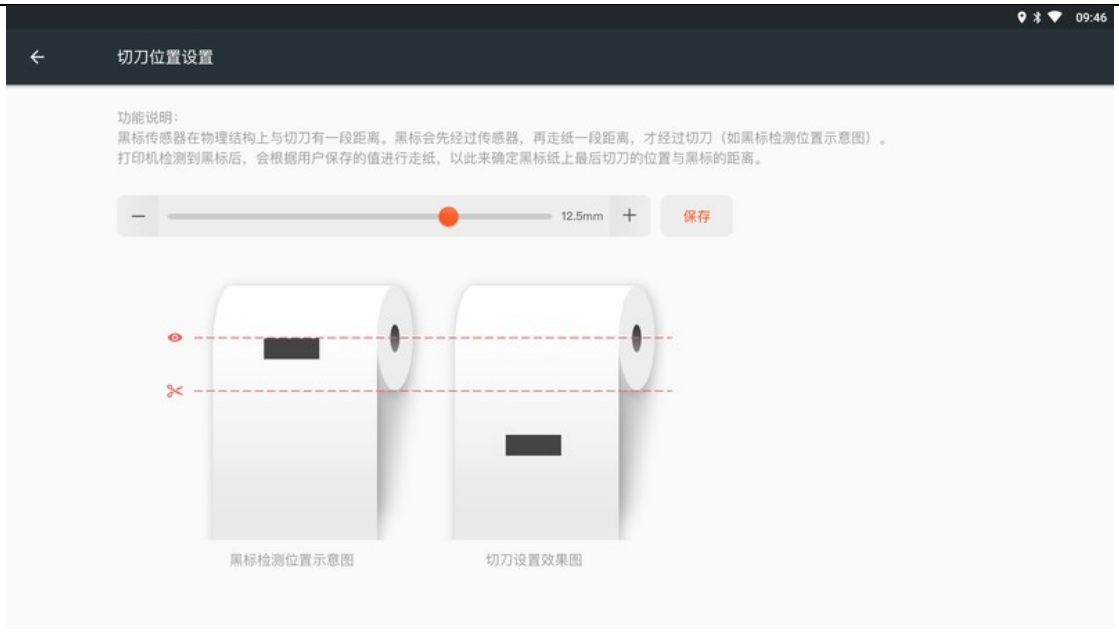
In this mechanism, print the final location and inspection paper sensor need to keep a certain distance (print the content of the final with black label edge distance is 6 mm) to ensure that the print content within 2 black label.

Black label mode instruction sequence:

1. Send print content
2. Input paper to black mark instruction {0x1c, 0x28, 0x4c, 0x02, 0x00, 0x42, 0x31}
3. Enter the cutter command.

After the printing is completed, the next black mark is automatically retrieved and the knife is cut at the specified position.

Users can modify the black label mode and the position of cutting tool in the built-in print management of setting -> print->.



Document update instructions

Serial number	Date	Version	Content	Authors
1.0.0			The original version	Arthur
1.1.170301			Added: print picture specifications Add: no printer hardware query interface description	Arthur
1.1.170315			Added: aidl cutter interface Add: aidl cash counter interface Add: aidl access to the cutter number interface Add: aidl access to the money cabinet open times interface	Arthur
1.1.170322			Added: aidl print interface with feedback Add: callback thing prints the result feedback Modification: barcode format specification	Arthur

1.1.170329			Added: description of character set modification	Arthur
1.1.170615			Add: things print instructions	Arthur
1.1.170726			Modification: instructions for AIDL calls	Arthur
1.1.170802			Added: T1 black label printing instructions	Arthur
1.1.170803			Added: T1 black label instruction description	Arthur
1.1.180523	2018/04/22		Modification: new document format, standardization and arrangement	Darren
1.1.180601	2018/06/01		Add: print image extension interface, ICallback instance description	Darren
1.1.180612	2018/06/12		Modification: bold instruction and part description	Darren
1.1.180629	2018/06/29		Modification: ICallback interface method error and description	Darren