Table of Contents

The First Part: Overview

Android SDK provides the developers with the SDK which can easily communicate with printer in Android. The communication way supports WIFI and Bluetooth. This SDK is reliable with easy functions and it can be expanded. It also encapsulates the processes including connecting the printer, sending and receiving the printer commands, and changing graphic file to printer command. in other words, user can develop a multifunctional software which can communicate with the printer, without paying attention to the details related to some protocol and the complicated communication process. The printer commands please refer to printer programming manual.

Android SDK supports Android 3.1 or above.

Note: wireless printing V3.0 has the Receipt Recovery function, through executing the bidirectional transmission protocol, when printer has malfunction or leaving the valid range of wireless signal, and the communication network is malfunctioned, printer can achieve the completeness of the receipt by printing again or continuous printing.

The Second Part: Program Example

1. Import RemotePrinter Library File

Eclipse develop tool is used as an example in the following.

(1) Copy RemotePrinter.jar to the catalogue of the developing item, this demonstrate is put into the root directory of the item.

(2) Select the item; right click the menu, Build Path -> Configure Build Path…, as shown in Figure 2-1:

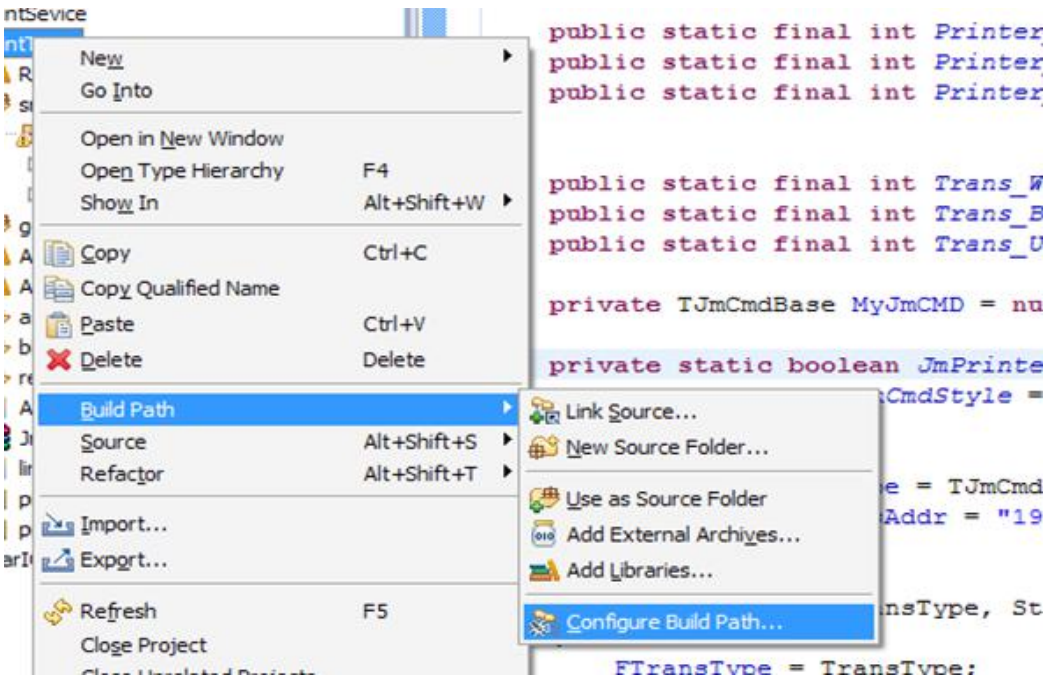

Figure 2-1

(3) Add RemotePrinter.jar to the item(Add JARS…), as shown in Figure 2-2:

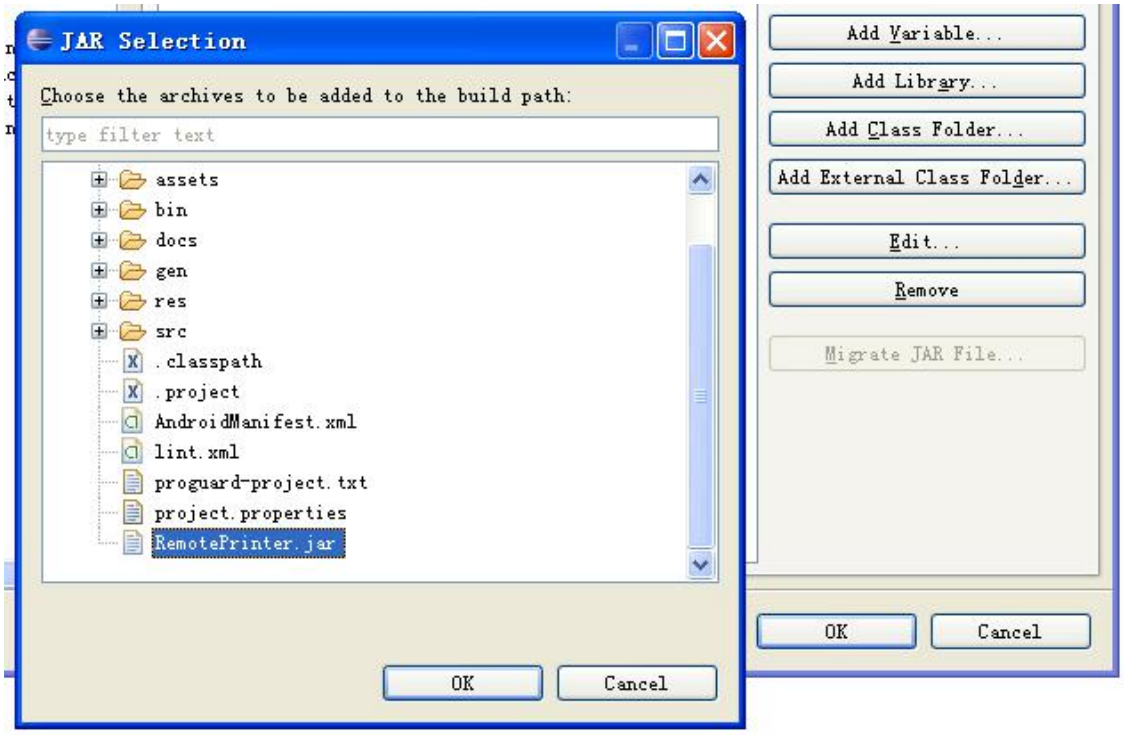

Figure 2-2

(4) Move RemotePrinter to the front, as shown in Figure 2-3:

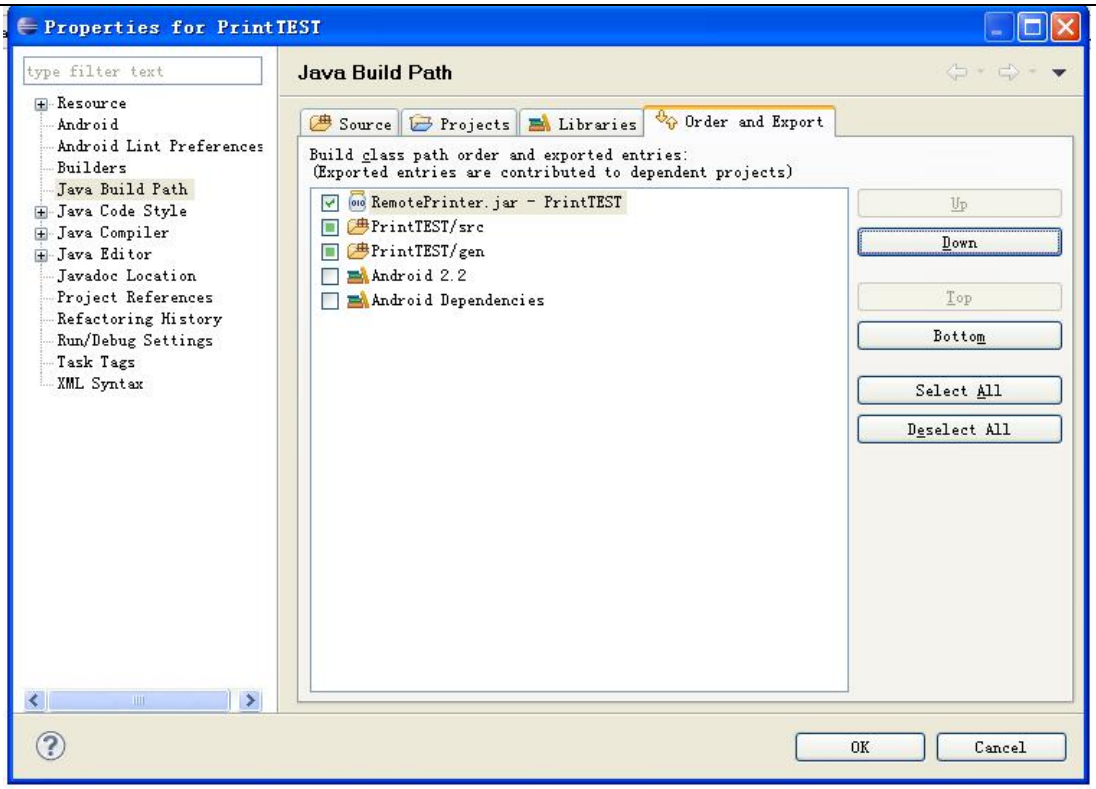

Figure 2-3

2. Simple Example

Watching a simple example is the easiest way to learn the SDK. We assume that JAR has been loaded to our project.

2.1 Text Printing (Two communication ways)

1.Using WiFi communication way

Firstly, get the IP address and other information of the wifi printer, use static method getWifiPrinterIP of RemotePrinter class, in the call-back method parameter; DeviceInfo object encapsulates printer's IP address, printer type and Mac address. The codes are as follows:

```
private int searchTime = 24; //Finding device 24 seconds

RemotePrinter.getWifiPrinterIP(searchTime,new RefleshHandler(){

                @Override
                public void deviceFound(DeviceInfo deviceInfo){

                    String mIP = deviceInfo.ip;//IP address
                    String mPrinterType = deviceInfo.type;//Printer type
                    String mMac = deviceInfo.mac;//Mac address
```

```
                    }
}
```

Assume: IP is 192.168.43.250, port number is 9100.
Now we use printer to print “Hello World”.
（1）the code of traditional way is as follows:

```
import com.printer.RemotePrinter;
import com.printer.VAR.PrinterType;
import com.printer.VAR.TransType;

String tmpSendData = "Hello Word\r\n";
// Step 1: Create RemotePrinter class instance, import the parameter, including the data transmissionway,
   printer IP address and port number.
RemotePrinter MyPrinter = new RemotePrinter(TransType.TRANS_WIFI, "192.168.43.250:9100");
// Step 2: Call open(false) method to open the one-way communication way
if (MyPrinter.open(false)) {
  //Step 3: Split the data, each package is 1K（it can not be over 2K）
      List<byte[]> list = ByteArrayUtils.fen(tempSendData.getBytes(),1024);
      for(int i=0;i<list.size();i++){
// Step 4: Call sendData() function to send the data
         MyPrinter.sendData(list.get(i));
// Step 5: Delay properly (according to the printer state), there may be messy code because the data is sent
         quickly or the printer buffer is small.
         Thread.sleep(500);
      }
} else {
            ……        //Fail to open
}
MyPrinter.close();              //Close communication way
MyPrinter = null;                   //Release RemotePrinter instance
}
```

(2) use the Receipt Recovery way to print, it doesn’t need to split and delay the data, there mainly be two more steps, call the start print task function before sending data, and call finish printing function at the end, data can be sent one time or many times between the period, so that the data can be transferred in

batches when there are much data. The codes are as follows (the following codes are used to introduce the steps):

```
import com.printer.RemotePrinter;
import com.printer.VAR.PrinterType;
import com.printer.VAR.TransType;

String tmpSendData = "Hello Word\r\n";
// Step 1: Create RemotePrinter class instance, import the parameter, including the data transmissionway, printer IP address and port number.
RemotePrinter MyPrinter = new RemotePrinter(TransType.TRANS_WIFI, "192.168.43.250:9100");
// Step 2: Call open(true) method to open the one-way communication way
if (MyPrinter.open(true)) {
// Step 3: Call startSendData() to start the printing task
//each stage can use a loop to process, it doesn't have to use while to realize
      boolean startFlag = myPrinter.startSendData();
      while(!startFlag){
         //In fact, the prompt frame pops out according to the wrong information; it will ask whether the user is to continue to print?
   Thread.sleep(3000); // Imitate the dialog box time
      //If user selects to continue printing, then it will execute:
         startFlag = myPrinter.continueSendData();
         //If user select to cancel printing, it will execute:
         //Return;
      }
//Step 4: Call sendData() to send data (can be called many times)
      int sendDataLength = myPrinter.sendData(tempData.getBytes());
      boolean sendFlag = false;
      if(sendDataLength == tempData.getBytes().length){
         sendFlag = true;
      }
      while(!sendFlag){
         // In fact, the prompt frame pops out according to the wrong information; it will ask whether the user is to continue to print?
         Thread.sleep(3000); // Imitate the dialog box time
      // If user selects to continue printing, then it will execute:
         sendFlag = myPrinter.continueSendData();
         //If the user selects to cancel printing, but the task is started, it will execute the following steps to cancel printing task:
         //1.Clear the printer buffer
```

```
        //    boolean cleanFlag = myPrinter.cleanPrinterCache();
        //    if(cleanFlag){
        //2. Finish the print task
        //       boolean finishFlag = myPrinter.finishSendData();
        //          if(finishFlag){
       //              //it is successful to cancel the task.
        //          }else{
        //              // Fail to cancel print task (if the error occurs during printing, it is treated as cancelling print
   task successfully, return to true)..
     //            }
     //       }else{
     //          // Fail to cancel print task
     //       }
     //    return; //restart
     }
//step 5: Call finishSendData() to finish the print task
     boolean finishFlag = myPrinter.finishSendData();
     while(!finishFlag){
        // In fact, the prompt frame pops out according to the wrong information; if the error occurs during
printing, it should remind the user to restart printing, not continue to print, otherwise, it should continue to
print.
        Thread.sleep(3000);
        finishFlag = myPrinter.continueSendData();
        //If user select “Cancel”, finish sending and start again.
        //return;
     }
     //Sent successfully
     MyPrinter.close();              //Close the communication way in order to connect again conveniently
     MyPrinter = null;               //Release RemotePrinter instance
}
```

2.Use the Bluetooth way

It is similar between the Bluetooth and WIFI, it is different when injecting the parameter in the step 1, suppose that the printer Bluetooth Mac address is：74:51:BA:52:07:B4，we now use the printer to print “Hello World”. The tradition code is used as the example:

```
import com.printer.RemotePrinter;
import com.printer.VAR.PrinterType;
import com.printer.VAR.TransType;

String tmpSendData = "Hello Word\r\n";
// Step 1: Create RemotePrinter class instance, inject the parameter, including the way of data transfer and
```

```
    printer Bluetooth Mac address.
RemotePrinter MyPrinter = new RemotePrinter(TransType.TRANS_BT, "74:51:BA:52:07:B4");
// Step 2: Call open(false) to open the one-way communication way
if (MyPrinter.open(false)) {
// Step 3: Call sendData() to send data
      MyPrinter.sendData(tmpSendData.getBytes())
} else {
            ……        // Fail to open
}
MyPrinter.close();              //Close the communication way
MyPrinter = null;                   //Release RemotePrinter instance
}
```

3.Use USB way

When using USB, UsbPrinter should be created, not RemotePrinter.

Note: if using the USB way, the Android system must support the usb printer driver.

Now we use printer to print “Hello World”. the codes are as follows (the Receipt Recovery way is the same with before or please refer to the example code)：

```
import com.printer.RemotePrinter;
import com.printer.VAR.PrinterType;
import com.printer.VAR.TransType;

String tmpSendData = "Hello Word\r\n";
// Step 1: Create UsbPrinter class instance, inject Context object parameter
UsbPrinter MyPrinter = new UsbPrinter(context);
// Step 2: Call openUsbPrinter() open the communication way and transfer one overtime parameter (ms)，
// and whether the Receipt Recovery mark is turned on（true denotes turning on）
// in the process of opening, printer will request if the printer device does not have the USB permission, when overtime, even
// pressing OK to get the permission belongs to open unsuccessfully, at this time, try again, or set a more reasonable time.
if (MyPrinter.openUsbPrinter(timeout，flag)) {
// Step 3: Call sendData() to send data
      MyPrinter.sendData(tmpSendData.getBytes())
} else {
            ……        // Fail to open
}
MyPrinter.close();              //Close the communication way
MyPrinter = null;                   //Release UsbPrinter instance
}
```

2.2 Graphic Printing

To print graphic, graphic needs transforming to graphic command data, then send the data to printer. The printer type should be confirmed first because the graphic command data of each printer may be different.

Call the static function ConvertImage of RemotePrinter class, specify the printer type explicitly. the WiFi communication way codes are as follows:

```
import com.printer.RemotePrinter;
import com.printer.VAR.PrinterType;
import com.printer.VAR.TransType;


// surFileName is the original image path (including name). the image formats are  JPG, PNG,  BMP and etc.
//Specify the printer type as thermal printer (Note: the printer type here is only for reference,the specific type will be gained by getPrinterType function after connecting successfully, please refer to the instance program.
byte[] tmpBuf = null;
tmpBuf = RemotePrinter.ConvertImage(PrinterType.PT_THERMAL, surFileName);
if (tmpBuf != null) { // If the graphic command data is created successfully
    RemotePrinter myJmPrinter = new RemotePrinter(TransType.TRANS_WIFI,"192.168.43.250:9100");
    //Print according to part 2.1
    ……
    myJmPrinter.close(); // Close the communication way
    myJmPrinter = null; // Release RemotePrinter instance
}
```

The Third Part: Detailed Information of Class and Function

1. Class Inheritance Relationship

java.lang.Object

└ com.print.printer.RemotePrinter

└ com.print.printer.UsbPrinter

public class RemotePrinter extends java.lang.Object

public class UsbPrinter extends java.lang.Object

2. Construction Function

2.1 RemotePrinter Construction Function

Function prototype

public **RemotePrinter** (TransType transType, String transAddr)

Description

This function creates RemotePrinter instance, and set the transfer parameter at the same time.

Parameter

transType：Transfer type parameter; see Table **3-2-1**

transAddr: when select the communication way as WIFI, the value is IP:Port, for example: the IP of the printer is 192.168.43.250, Port is 9100， then the parameter which should be injected is “192.168.43.250:9100”. When select the communication way as Blue Tooth, the value is the address of Blue Tooth.

Transfer parameter	Value	Description
TRANS_WIFI	0	Communication way is WIFI.
TRANS_BT	1	Communication way is Bluetooth.

Table 3-2-1 TransType value

Return value

If the function is called successfully, return to the class instance, otherwise, return to null.

2.2 UsbPrinter Construction Function

Function prototype

public **UsbPrinter**(Context context)

Description

This function creates UsbPrinter instance, and set the parameter at the same time.

Parameter

context：Android environment object

Return value

If the function is called successfully, return to the class instance, otherwise, return to null.

3. Communication Function

3.1 open Function

Function prototype（Two）

1.boolean **open** (boolean flag);

2.boolean **open**(int timeout,boolean flag);

Description

Open the communication way, the first open function is used to wireless type printer, the second function is used in USB type printer. When all the communication operation has been finished, call close to close the communication way. Open and close appear in pair. There is one or several communication operation between them.

Parameter

timeout：set the waiting time to get the usb permission, as the Android system cannot get the USB permission when applying for it, it needs to set a time to wait, the specific time depends on the situation, the function may return to false because the time is too short. But the permission may have been gained, user can call it again.

flag： The mark that begins the Receipt Recovery function. True represents turning on， false represents turning off. Note: after turning on, using false cannot switch to the original way, it can recover only by turning off the printer.

Return value

If the function is called successfully, return to true，otherwise, return to false，in order to get more error information, please call getLastErrorCode and getMessage.

Related function

close

3.2 close Function

Function prototype

boolean **close** ();

Description

Close the communication way. When the communication begins, call open to open the communication way. When all the communication operation has been finished, call close to close the communication way. Open and close appear in pair. There is one or several communication operation between them. Please close the communication way in time, or other tasks may not be able to conduct because the communication way is over occupied.

Return value

If the function is called successfully, return to true，otherwise, return to false，in order to get more error information, please call getLastErrorCode and getMessage.

Related function

open

3.3 startSendData Function（Used in Receipt Recovery mode ）

Function prototype

public boolean **startSendData**();

Description

This function is used to initialize the printer and check whether the printer status is normal or not, such as whether the paper is out, the cover is opened or the last print task hasn't finished. This is the begin period of the print task.

Return value

If the function is called successfully, return to true，otherwise, return to false，in order to get more error information, please call getLastErrorCode and getMessage. When returning to false, continuous sending can be conducted, please refer to continuous sending function.

Related function

continueSendData

3.4 sendData Function

Function prototype

public int **sendData**(byte[] data);

Description

This function is to send data to the printer. This is the sending period of the print task.

Parameter

data：the data is waiting to be sent. Byte type array. In traditional way, the size of array is not over 2K and it is splited by the developer. Please refer to the example. In the receipt recovery mode, it doesn't need to split and delay.

Return value

If the function is called successfully, return the sent byte number, otherwise, return -1. in order to get more error information, please call getLastErrorCode and getMessage. If the data is sent unsuccessfully, continuous sending can be conducted, please refer to continuous sending function.

Related function

continueSendData

3.5 finishSendDataFunction（Used in Receipt Recovery mode）

Function prototype

public boolean **finishSendData**();

Description

This function is to finish the print task, and check whether there is error during printing. This is the ending period of the print task.

Return value

If the function is called successfully, return to true，otherwise, return to false，in order to get more error information, please call getLastErrorCode and getMessage. When returning to false, continuous sending can be conducted, please refer to continuous sending function.

Related function

continueSendData

3.6 continueSendDataFunction（Used in Receipt Recovery mode）

Function prototype

public boolean **continueSendData**();

Description

This function is to continue to send one of the period between beginning, sending and ending. Note: It will only continue sending one period. If the function is called when starting the task is failed, it will continue to send the beginning period, if the function is called when sending is failed, it will continue to send the sending period.

Return value

If the continuous sending is successful, return true，otherwise, return false，in order to get more information, please call getLastErrorCode and getMessage. When returning to false, continuous sending can be conducted.

Related function

startSendData

sendData

finishSendData

3.7 cleanPrinterCacheFunction（Used in Receipt Recovery mode）

Function prototype

public boolean **cleanPrinterCache**();

Description

This function is to clear the data in the printer buffer. When printing a half, the sending period is failed because some printer problems occur (like paper out), user can select not to continue sending, if you want to restart the print task, the print task must be terminated, which means the finishSendData function should be called. But the task cannot be terminated because there are some data remaining in the printer buffer, the function needs calling before finishing the print task, the task can be finished after the cache is cleared.

Return value

If the function is called successfully, return to true，otherwise, return to false，in order to get more error information, please call getLastErrorCode and getMessage.

Related function

finishSendData

4. Auxiliary Function

4.1 Image Conversion Function

	convertImage (String srcFileName)	**convertImage**(Bitmap srcImage)
Parameter	Original image path (including name). this image format can be JPG, PNG, BMP and other common image format	Bitmap type object
Return value	byte[]	byte[]

Description

This function is used to convert image to image command data, call sendData to send the image command data to the printer so as to realize the image printing.

Program example: refer to the second part *Graphic Printing* of RemotePrinter.

Related function

RemotePrinter，sendData

4.2 getWifiPrinterIP Function

Function prototype

public static void **getWifiPrinterIP**(int timeout,RefleshHandler handler);

Description

This function is used to get the wifi printer IP, type and mac address by UDP broadcast.

Parameter

timeout： time out, 0 is the default value, the unit is second.

handler: call-back interface, handle the search result.

Related function

cancelGetWifiPrinterIP

4.3 cancelGetWifiPrinterIP Function

Function prototype

public static void **cancelGetWifiPrinterIP**();

Description

This function is used to cancel searching wifi printer, call it before closing the page, or the connection will not be off at once.

Related function

getWifiPrinterIP

4.4 isConnected Function

Function prototype

public boolean **isConnected**();

Description

This function is used to check whether the connection is normal or not.

Return value

If it keeps connecting, return to true，otherwise, return to false，in order to get more error information, please call getLastErrorCode and getMessage.

Related function

open

4. 5 getUsbPrinterStatus Function

Get the USB printer status (1 byte), each of which represents the following information:

Bit(s)	instructions
7..6	reserved
5	1 = paper exhaust，0 = normal
4	1 = online，0 = offline
3	1 = no error，0 = error
2..0	reserved

Want to know whether to judge bit3 on the line, specifically what type of error, here can only find two (bit4 and bit5).

Function prototype

public byte[] getUsbPrinterStatus(int timeout);

4.6 getLastErrorCode Function

Function prototype

int **getLastErrorCode** ();

String **getMessage**();

Description

getLastErrorCode function gets the last error during communication. The error code is a 16-bit value. Developer should call getLastErrorCode at once after the function returns to the marked error code. Please refer to the Error Code Meaning List for the specific content of the error code.

getMessage function will explain the related error.

Return value

If the function is called successfully, return to the last error code and error message during the communication.

Program example：

```
//Create RemotePrinter class instance, and set the communication parameter (Communication way:WIFI,  IP: 192.168.43.250,  port: 9100)
RemotePrinter myPrinter = new RemotePrinter(TransType.TRANS_WIFI, "192.168.43.250:9100");
if (myPrinter.Open()) { // Open the communication way, get the printer type automatically.
    ……  //Operation after opening
} else {
    // Fail to open the communication way
    int ErrCode    = myPrinter.getLastErrorCode();
    String ErrMsg = myPrinter.getMessage();
    System.out.print("error code："+ ErrCode);
    System.out.print("error message："+ ErrMsg);
}
……
```

4.7 html2PrintData Function

Function prototype

public static byte[] html2PrintData(String htmlString);

Description

Convert HTML to the corresponding print data in accordance with the conversion protocol defined by the company. (For the contents of the agreement, see the corresponding manual.)

Return value

HTML corresponding print data

4.8 JSON2PrintData Function

Function prototype

public static byte[] JSON2PrintData(String jsonString);

Description

The JSON is converted into the corresponding print data according to the conversion protocol defined by the Imaging Group. (For the contents of the agreement, see the corresponding manual.)

Return value

JSON corresponding print data

4.9 getPrinterModel Function

Function prototype

public String getPrinterModel()

Description

Return to the printer model.

Return value

Printer model string

Appendix

Appendix 1 Function List

Function	Description
Construction Function	
RemotePrinter	Create RemotePrinter instance.
Communication Function	
open	Open the communication way.
close	Close the communication way.
startSendData	Check the printer state and initialize.
sendData	Send data to the printer.
finishSendData	Finish the print task, and check whether there is error during printing.
continueSendData	Continue to send one of the periods between beginning, sending and ending.
cleanPrinterCache	Clear the data in the printer buffer.
Auxiliary Function	
convertImage	Convert image to image command data.
getWifiPrinterIP	Get the wifi printer IP, type and mac address.
cancelGetWifiPrinterIP	Cancel searching wifi printer.
getLastErrorCode	Get the last error code.
isConnected	Check whether the printer is connected.
getLastErrorCode	Get the error code
getMessage	Get the error message

Appendix 2 Error Code Meaning List

Error code	Meaning	Note
0x1001	WIFI connection error	
0x1002	WIFI malfunction when disconnecting	
0x1003	WIFI object is none	WIFI has not been connected or has been disconnected
0x1004	WIFI malfunction when sending data	
0x1005	WIFI malfunction when receiving data	Maybe time out or there is no data returning from printer
0x2001	Bluetooth connection error	
0x2002	Error occurs when the Bluetooth is disconnected	

0x2003	Bluetooth object is none	Bluetooth has not been connected or has been disconnected
0x2004	error occurs when the Bluetooth sends data	
0x2005	Error occurs when Bluetooth receives data	Maybe time out or there is no data
0x4001	Image format error	Image format is not supported
0x4002	The file doesn't exist	
0x4003	Fail to convert the image to the dot matrix data	
0x4041	Send error	The current data package cannot be sent
0x405	Send data error	Error occurs when getting the printer type
0x4051	The head of the data package error	Package head encapsulation error
0x4052	Data parity error	CRC is inconsistent
0x4053	Printer receive buffer full	
0x4054	Receive error	
0x406	Package type error	Printer returns to wrong package type
0x407	reset stage error	
0x4071	One more reset error	
0x408	Task beginning stage error	
0x409	Task ending stage error	
0x501	Can not receive data	The signal may be weak or the printer is powered off
0x502	Can not receive data package	Data but not package type is received
0x5021	Can not receive data in 5 seconds (1)	data timing
0x5022	Can not receive data in 5 seconds (2)	receive data, but couldn't find the package head timing
0x5023	Can not receive data in 5 seconds (3)	Timing after receiving the whole data package
0x503	Can not receive data in 2 seconds (1)	package sequence number error timing
0x5031	Can not receive data in 2 seconds (2)	
0x504	Can not receive data in 1 second (1)	Uncompleted data package timing
0x5041	Can not receive data in 1 second (2)	Package head is not enough to time

0x505	Error occurs when receiving socket	
0x6001	offline	
0x6002	Cash drawer is opened	
0x6003	Cover is opened	
0x6004	Paper out	
0x6005	Cutter error	
0x6006	Other error	
0x701	Protocol switch error	
0x702	Clear the cache error	
0x703	One more clear failure	
0x704	Fail to check state	
0x7041	Fail to check the state again	
0x705	There are unfinished print task	it may be failed to cancel task or there is error when finishing the task
0x801	There is error during printing	Paper is out, cover is opened and etc.
0xFFFE	fail to get the printer type	it may be the connection problem
0xFFFF	Unknown error	