

BluPrints Thermal Receipt Printer

Developer Guide (2 Inch-58/mm)

AadharShila Mobility Solutions Pvt. Ltd.

Web: <http://www.bluprints.in/>

VERSION CONTROL: 6.2 / March 2021

PLEASE NOTE THAT THIS SDK VERSION, NAMELY, BluPrints_SDK6.1 IS APPLICABLE FOR BluPrints SCRYBE PRINTER FIRMWARE VERSIONS 8.0 AND ABOVE.

DOCUMENT NAME: ANDROID DEVELOPER GUIDE RELEASE DATE: 24-March-2021

SDK SUPPORTED: BluPrints_SDK6.1 FIRMWARE SUPPORTED: 8.0 AND ABOVE

TABLE OF CONTENTS

BluPrints SDK	03
BluPrintsPrinter Class.....	03
BluPrintsScrybeDevice.....	08
WI FI Printer	11
Packet Protocol.....	11
QR Code Generation	15

SDK (Software Development Kit) for ANDROID

This document describes the use of BLUPRINTS SCRYBE Thermal Printer SDK for Bluetooth and Wi-Fi Operations. This SDK provide an Interface between an Android Application and the BLUPRINTS Thermal Printer namely SCRYBE. The SDK is android based, and requires at least 10 level of android SDK and java compiler. It comprises of the following Interfaces, Classes and functions.

Com. bluPrints.api consists of Classes – BLUPRINTSPrinter, BLUPRINTSScrybeDevice, and BLUPRINTSWifiPrinter.

BluPrintsPrinter: BluPrintsPrinter class handles all the Bluetooth related functions. These include

1. **setFontType**: public void setFontType(byte FONT) throws IOException

In order to set any of the above four fonts in the printer, you need to call this function and pass the desired font.

Normal Font: FONT_NORMAL

Calibri Font: FONT_001

Tahoma Font: FONT_002

Verdana Font: FONT_003

2. **setFontSize**: public void setFontSize(byte DIMENSION) throws IOException

In order to set the font size to Double width or double height of a particular line in the printer, you need to call this function and pass the desired parameter of Double Width or Double Height. Please note that double width functionality will not work properly in case of 2 inch 48 character per line Printer. Alternatively, you can use the sendByte function and simply pass the desired parameter DOUBLE_WIDTH or DOUBLE_HEIGHT as defined below.

3. **sendByte** :- public void send Byte(byte bt) throws IOException

In order to send any of specific byte to the printer via Bluetooth, you need to call this function and pass the desired byte.

4. **sendByteArray**: public void send ByteArrayBT(byte[] byteArr) throws IOException

In order to send any of specific byte array (sequence) to the printer via Bluetooth, you need to call this function and pass the desired Byte Array.

5. **printlnNegative**: public void prinlnNegative() throws IOException

NEGATIVE_CHAR = 0X0E

In order to print any of specific line in Negative Text Mode you need to call this function.

Alternatively, you can use the sendByte function and simply pass the value of NEGATIVE_CHAR as defined above.

6. **enableUnderline**: public void enableUnderline() throws IOException

UNDERLINE = 0X15

In order to underline any of specific line you need to call this function. Alternatively, you can use the sendByte function and simply pass the value of UNDERLINE as defined above.

7. **setLineFeed**: public void setLineFeed(int noOfFeed) throws IOException

LINE_FEED = 0X0A

In order to print specific number of blank lines or line feeds you need to call this function.

Alternatively, you can use the sendByte function and simply pass the value of LINE_FEED as defined above. You can call sendByte multiple times for printing multiple blank lines.

8. **setCarriageReturn**: public void setCarriageReturn() throws IOException

CARRIAGE_RETURN = 0X0D

In order to print a single blank line you need to call this function. Alternatively, you can use the sendByte function and simply pass the value of CARRIAGE_RETURN as defined above. You can call sendByte multiple times for printing multiple blank lines.

9. **print**: public void print(String text) throws IOException

This function is used to print a specified text (in the form of a string) to the printer.

General Guidelines for Text Printing

- Send the command to print the various fonts and styles
- If multiple commands are given then last command will be effective only.
- Location to apply – at the start of the line to print

Following Printing commands are for formatting the text of a single line

Command Type	Packets	Hex Value
Normal Font	CTRL+F	0x06
Font 001: Calibri Font	CTRL+F	0x06
Font 001: Calibri Font	CTRL+C	0x03
Font 002: Tahoma Font	CTRL+T	0x14
Font 003: Verdana Font	CTRL+V	0x16
Double Width	CTRL+D	0x04
Double Height	CTRL+H	0x08
Negative Char	CTRL+N	0x0E
Underline	CTRL+U	0x15
Line Feed (New Line)	ENTER	0x0A
Carriage Return(New Line)	ENTER	0x0D

Rules to apply commands:

- ✓ You can change the fonts, as well as Double width, double height, Negative and underline characters. Command should be sent before the line to print
- ✓ Should not repeat the command more than one time at the start of line
- ✓ Line feed of carriage return should be kept in the end of line if length of text is less than 32. E.g. To print underline. First send 0x15 (in hex) and then text, say
“Hello world” and then carriage Return. Then it will print Hello world

Function for Non –Text Printing

Print Barcode:

public void printBarCode(String Barcode, BARCODE_TYPE Btype, BARCODE_HIEGHT bHieght)
throws IOException

BARCODE_TYPE_CODE39 = 0X45

This function is used to print a 2-Dimensional Barcode on the Printer, generated automatically according to the string passed to this function. The string should be of Capital English Letters or Numerals. The maximum characters in the strings can be up to

11. Under Barcode Type, the Printer supports BARCODE_TYPE_CODE39. The Barcode Height supported is DOUBLEDENSITY_FULLHEIGHT.

The following packet is generated internally within this function and sent to the printer:

```
barcodePacket[0] = 0x1D; 'GS' barcodePacket[1] = 0x6B; 'k'  
barcodePacket[2] = BARCODE_TYPE_CODE39; 0x45 barcodePacket[3] = (byte)  
(barcodeBytes.length + 2); //length of barcode data barcodePacket[4]  
= 0x2A;
```

```
barcodePacket[5] to barcodePacket[length of barcode string] = BarcodeBytes;  
barcodePacket[Length of barcode string + 1] = 0x2A;
```

printImage:

public void printImage(Bitmap originalBitmap) throws IOException

This function is used to print an Image in the form of a Raster Image, based on the standard ESC/POS Raster Image command set of GS v protocol. You need to pass the bitmap of the desired image to be printed. Please note that this function can be used to print QR Codes as well, by first generating the QR code from a given string and thereafter, sending its bitmap to the PrintImage

Function. The source code for generating the QR Code has been provided at the end of this document.

printTextAsImage:

public void printTextAsImage(String texttoconvert) throws IOException

This function is used to print any text (multilingual, Unicode type characters, etc) in the form of a Raster Image. You need to pass the String that you need to be printed as an image. The main utility of this function is that the user can easily input any multilingual string, that can be printed as is on the printer, so as to enable printing in any language irrespective of the fonts. (Printing characters of Urdu, Gujrati, Arabic, Oriya, Tamil, Tamil, Telugu, etc.)

this function is that the print is so clear and fast that there is no perceptible difference to an end user in understanding whether this text has been printed in the form of text or in the form of an image.

printBitImage:

public void printBitImage (BitMap original ,Context context, byte imageAlignment) throws IOException

IMAGE_LEFT_ALIGNMENT = 0x6C;

IMAGE_CENTER_ALIGNMENT = 0x63; IMAGE_RIGHT_ALIGNMENT = 0x72;

This is a deprecated function that has only been kept for backward compatibility. PrintBitImage function uses the standard ESC * algorithm for printing of a bitmap The maximum image size should be in the following range: - 355 X 500 (WxH) pixels.

For image printing, you need to half the pixel size of height of an image to get the desired width and height.

E.g.: - Suppose you need to print a logo of dimensions 355 X 300 (WXH) pixels, then half the size of the height of an image i.e. 150 pixel for one time in your code by scaling function (as explained below). Width will remain same.

Bitmapscaled_bitmap = bitmap.createScaledBitmap(bitmap, 355, 150, false); 355 pixel= Width (Original Width)150 pixel= Height (Height will get double i.e. 300 pixels as original height of an image while printing).

Likewise, you can print the logo of desired dimensions. Note: - Maximum Width size is 355 pixels Maximum Height size is 500 pixels.

BlueprintsScrybeDevice:

This class is used to instantiate a BLUETOOTH SCRYBE Device, containing the Context, the Bluetooth Adapter, Bluetooth Device and Bluetooth Socket. An object of BluPrintsScrybeDevice once instantiated, is capable of returning the BluPrintsPrinter object that has been described previously. This class has the following functions:

- 1. Constructor:** For Creating the Object: `public BluPrintsScrybeDevice (BluPrintsScrybe)`
- 2. startDiscover:** `public void startDiscover(Context iContext)`
By calling this method, a list of local Bluetooth devices will be returned.
- 3. pairDevice:** `Public String pairDevice(String printerName)`

Before pairing any printer by name, first call the method `startDiscover(Context iContext)` and get the result in the method `public void onDiscoveryComplete(ArrayList<String>aemPrinterList)` of the Interface `IAemScrybe`. Example:

```
BluPrintsScrybeDevice m_AemScrybeDevice = new BluPrintsScrybeDevice (newIAemScrybe()  
{  
  @Override  
  Public void onDiscoveryComplete(ArrayList<String>aemPrinterList)  
  {  
    // TODO Auto-generated method stub  
  }  
});
```

Now call the method `public String pairDevice(String printerName)` which returns a String which may be:

NOT_SCANNED when you call this method before scanning. DEVICE_NOT_FOUND when the printer is not found. PAIRED when the device is successfully paired FAILED_TO_PAIRED when the device is failed to paired

- 4. connectToPrinter:** `public Boolean connectToPrinter(String printerName)` throws `IOException`
By calling this method, printer gets connected.

- 5. disconnectPrinter:** `public Boolean disconnectPrinter()` throws `IOException` By calling this method, the connected printer gets disconnected.

- 6. BluPrintsPrinter:** `public BluPrintsPrinter getBluPrintsPrinter()`
By calling this method, you can create the object of the class `BluPrintsPrinter`.

- 7. ArrayList:** `public ArrayList<String>getPairedPrinters()` By calling this method, a list of paired printers is returned.

- 8. getSDKVersion:** `public String getSDKVersion()` By calling this method, SDK version is obtained.

- 9. BtConnStatus:** `public Boolean BtConnStatus()`

This method returns True if the Printer is already connected on Bluetooth (Bluetooth connection is already alive). It returns False if the connection status is False, i.e. if Bluetooth socket is disconnected.

10. onScanRFID: public void onScanRFID(String Buffer) This method is for RFID data.

11. onScanPacket: public void onScanPacket(String Buffer) This method is for getting various responses.

Response Name	Type	Explanation
No Paper	Error	When there is no paper roll
Printer Head High Temp	Error	Printer mechanism is not connected / High temp of Printer Head
Low Battery	Error	Battery is low
Battery Charging	Notification	Battery is charging
Paper OK	Response	Paper roll is present

FUNCTIONS FOR WIFI PRINTER

BluPrintsWifiPrinter:

This Class is for communicating with BluPrints SCRYBE Wifi Series Printers. This class does the entire processing for various functions such as generating the bytes for Barcode and Image. In order to send these bytes out on a physical TCP/IP Socket, there needs to be a separate Client class to be maintained by the user in the main Java Application Project. This class has been purposely kept outside from the SDK so that the user can customize the messages thrown by the interfaces and accordingly process the commands. For ease of use the BluPrintsWifiPrinter Class has been provided so as to carry out complex processing of Images and Barcode packet generation. Since all these functions return array of Bytes, these bytes can be easily send out to the client socket by the user. MyClient.java class has been provided as extending AsyncTask for the user to override the DoInBackground and onPostExecute functions according to the functionality as needed.

The functions of BluPrintsWifiPrinter class are described below:

1. printBarcode: public byte[] printBarcode(String barcodeData, BARCODE_TYPE Btype, BARCODE_HEIGHT bHeight) throws IOException BARCODE_TYPE_CODE39 = 0X45

This function is used to return the byte array for printing a 2-Dimensional Barcode on the Printer, generated automatically according to the string passed to this function. The string should be of Capital English Letters or Numerals. The maximum characters in the strings can be up to 11. Under Barcode Type, the Printer supports BARCODE_TYPE_CODE39. The Barcode

Height supported is DOUBLEDENSITY_FULLHEIGHT. The following packet is generated internally within this function and sent to the printer:

```
barcodePacket[0] = 0x1D; 'GS' barcodePacket[1] = 0x6B; 'k'  
barcodePacket[2] = BARCODE_TYPE_CODE39; 0x45 barcodePacket[3] = (byte)  
(barcodeBytes.length + 2); //length of barcode data barcodePacket[4] = 0x2A;  
barcodePacket[5] to barcodePacket[length of barcode string] = BarcodeBytes;  
barcodePacket[Length of barcode string + 1] = 0x2A;
```

2. printImage: public byte[] printImage(Bitmap originalBitmap) throws IOException This function returns the byte array for printing an Image in the form of a Raster Image, based on the standard ESC/POS Raster Image command set of GS v protocol. You need to pass the bitmap of the desired image to be printed. Please note that this

3. printTextAsImage: public void printTextAsImage(String TextToConvert) throws I Exception This

function returns the byte array for printing any text (multilingual, Unicode type characters, etc) in the form of a Raster Image. You need to pass the String that you need to be printed as an image. The main utility of this function is that the user can easily input any multilingual string, that can be printed as is on the printer, so as to enable printing in any language irrespective of the fonts. (Printing characters of Urdu, Gujrati, Arabic, Oriya, Tamil, Tamil, Telugu, etc.). The benefit of this function is that the print is so clear and fast that there is no perceptible difference to an end user in understanding whether this text has been printed in the form of text or in the form of an image.

MyClient Class for TCP/IP Socket for WIFI Connection:

```
public class MyClient extends AsyncTask<Void, Void, Socket> { String response = ""; public  
Socket socket = null;  
public static final int BUFFER_SIZE = 2048;  
private PrintWriter out = null; private BufferedReader in = null; private String dstAddress = null;  
private int dstPort = 9100; private Context context; private String text; public static boolean  
wifiConnection=false;  
public static boolean connection=true;  
*/  
public MyClient(String text, Context context) { this.dstAddress = text; this.dstPort = 9100;  
this.context = context;  
}  
public void configSocket(String host)  
{  
this.dstAddress = host; this.dstPort = 9100;  
}  
@Override  
protected Socket doInBackground(Void... arg0) {  
String message = ""; int charsRead = 0; char[] buffer = new char[BUFFER_SIZE]; try {
```

```

this.socket = new Socket(dstAddress, dstPort); Log.i("SocketConnection",this.socket+""); out =
new PrintWriter(socket.getOutputStream()); in = new BufferedReader(new
InputStreamReader(socket.getInputStream())); wifiConnection = true;
} catch (UnknownHostException e) {
// TODO Auto-generated catch block wifiConnection = false; connection = false;
e.printStackTrace();
this.response = "UnknownHostException: " + e.toString(); Log.e("Exception",
"UnknownHostException: " + e.getMessage()); } catch (IOException e) {
wifiConnection = false; connection = false; e.printStackTrace();
this.response = "IOException: " + e.toString();
Log.e("Exception", "IOException: " + e.getMessage()); } finally { if (socket != null) {
}

}
return this.socket;
}
@Override
protected void onPostExecute(Socket result) { try{ if (result!=null)
{
((MainActivity)context).onSuccess(this.response,this.socket,dstAddress,dstPort,out);
}
else
{
((MainActivity)context).onError(this.response);
}
//super.onPostExecute(result);
}
catch (Exception ec){ ((MainActivity)context).onError(this.response);
}}
public void sendDataOnSocket(String message,PrintWriter out) { if (message != null)
{
out.write(message); out.flush();
}
}
public void sendBytesOnSocket(byte[] btPkt, int numBytes,Socket socket) { try {
socket.getOutputStream().write(btPkt, 0, numBytes);
} catch (IOException e) {
// showAlert("error in outputstream " + e.toString());
((MainActivity)context).onError(this.response);
}
}
public int disconnectWithServer()
{
int retVal = 0;
if (socket != null)
{

```

```

if (socket.isConnected())
{
try { socket.close(); while(true)
{
if(socket.isClosed() == true)
{
retVal = 1; return retVal; }
}
} catch (IOException e) { ((MainActivity)context).onError(this.response); } }
}
return retVal; }
}

```

Packet Protocol

The communication is done in either of three modes:

a. Bluetooth Mode

b. Wi-Fi Mode

Device reads the packet, which is connected to host application, or you can connect through any one of the above three modes. Device reads the packet and parses it, then processes the command.

Bluetooth (Host) to Printer Commands:

To send commands to printer first add [ESC] i.e. 0x1B in starting and then append the query packet. Host application to Printer command:

Bluetooth (Host) to printer Commands:

To send commands to printer first add [ESC] i.e. 0x1B in starting and then append the query packet. Host application to Printer command:

ESC(0x1B)	Start delimiter (0x7E)	BP*	Separator	Packet Type	Separator	Packet Data	Delimiter (0x5E)
1 Byte	1 Byte	2 Byte	1 Byte	2 Byte	1 Byte	-	1Byte

***BP: Bluetooth (host) to Printer:**

Command Type	Packets
*Get Printer Status :	[ESC]~BP GET PRN_ST^
Get Printer Configurat	[ESC]~BP GET CONFIG ^
Get Battery Status	[ESC]~BP GET BAT_ST ^
Get Printer Version Sta	[ESC]~BP GET PRNVER ^ (Returns Ver 1.1.3.21 for Bluetooth and 1.1.3.20 for Non BT)

Printer Power OFF	[ESC]~BP PRN PWROFF ^
Test LED	[ESC]~BP TST TSTLED ^
Test Print	[ESC]~BP TST PRINT ^
Set Printer Configuration	[ESC]~BP CON Name,Password,FontType,EnableEncryption,MSRTimeOut,ICTimeOut^

BT to Printer

Responses from Printer:

Packet Structure:

Start Delimiter (0x7E)	PB*	Separator (1)	Packet Type	Separator (1)	Response	End delimiter (0x5E)
1 Byte	2 Byte	1 Byte	3 Byte	1 Byte	-	1 Byte

*PB: Printer to Android (Host)

Response table:

Response Name	Type	Packets
No Paper	Error	~PB PRN NOPAPER^
Printer Head High Temp:	Error	~PB PRN HGHTEMP^
Printer Platen	Error	~PB PRN PLTNREL^
Wrong Packet:	Error	~PB PRN PKTEROR^
MSR Swiped	Notification	~PB CRD MSR_SWP^
Low Battery	Error	~PB BAT LOWBATT^
Battery Charging	Notification	~PB BAT BATCHG1^

Battery Not Charging	Notification	~PB BAT BATCHG0^
Battery Charge Status	Notification	~PB BAT BAT%3d%%^
Printer Configuration	Response	~PB CON Name, Password, FontType, EnableEncryption, MSRTIMEOUT, ICTIMEOUT^
Printer Power Off	Response	~PB PRN PWR_OFF^

Bit Map Image Print – Note that this method of Printing is only maintained for the purpose of backward compatibility. The Faster, easier, clearer and more standard way of printing is by using Print Image function and passing it a Bitmap object.

1. Select the image either from PC or from mobile's SD card.
2. Convert the image into monochrome bmp.
3. Resize the image to fit into the printer paper area if it is exceeding
4. Now read the processed image through fileinputstream and save it into byte array.
5. Make the packet of image and then send it to printer over output stream.

Mode M (In Hex)	Mode M Description	Vertical Dot	Horizontal Dot	Max Dot/Line
0x08	double height	1	1	384 (48bytes)
0x04	double width	2	1	384

Mode Table

Alignment (a)	Hex Value
Left align	0x6C
Centre align	0x63
Right align	0x72

Alignment

QR Code Generation Function:

```
Public void onPrintQRCodeRaster (View v) throws WriterException, IOException {
String text= editText.getText().toString(); if(text.isEmpty()){
showAlert("Write Text To Generate QR Code");
}
else {
Writer writer = new QRCodeWriter();
String finalData = Uri.encode(text, "UTF-8"); showAlert("QR " + text);
try {
BitMatrix bm = writer.encode(finalData, BarcodeFormat.QR_CODE, 300, 300); Bitmap bitmap =
Bitmap.createBitmap(300, 300, Config.ARGB_8888); for (int i = 0; i < 300; i++) {
for (int j = 0; j < 300; j++) {
bitmap.setPixel(i, j, bm.get(i, j) ? Color.BLACK: Color.WHITE);
}
}
Bitmap resizedBitmap = null;
int numChars = glbPrinterWidth;
if(numChars == 32){
resizedBitmap = Bitmap.createScaledBitmap(bitmap, 384, 384, false);
m_BluPrintsPrinter.printImage(resizedBitmap);
m_BluPrints.setCarriageReturn();
m_BluPrints.setCarriageReturn();
m_BluPrints.setCarriageReturn();
}else {
resizedBitmap = Bitmap.createScaledBitmap(bitmap, 384, 430, false);
m_BluPrints.printImageThreeInch(resizedBitmap);
m_BluPrints.setCarriageReturn();
m_BluPrints.setCarriageReturn();
m_BluPrints.setCarriageReturn();
}
} catch (Writer Exception e) { showAlert("Error WrQR: " + e.toString()); }
```