

Android-SDK Development Document

V2.3

A. Introduction

1. Software package name: com.printer.bluetooth.android
2. Classes name:

Class Name	Discription
BluetoothDiscover	Discovering printer which has been paired with bluetooth
BluetoothPrinter.PrinterType	Enumerate class of printer type
BluetoothPrinter	A class that controlling the operation of the printer.
FontProperty	A class that setting printer's font.
PrintGraphics	A class that printing image.
Table	A class that can print a table according to the given data
Utils	A class that provides some useful tools.

B. Class "BluetoothPrinter" provides the following method:

1. Instance method:

- a) `BluetoothPrinter(BluetoothDevice device)`

Initialise BluetoothPrinter class by passing BluetoothDevice parameter.

- b) `BluetoothPrinter(String macAddress, int flag)`

Initialise BluetoothPrinter class by MAC address of Bluetooth device, the second parameter is not used.

- c) `BluetoothPrinter(String deviceName)`

deviceName is the name of Bluetooth device which has be paired. If there are more than one devices has same name, only the first one can be initiallised (this method is out of date). **Retaining this method is just for compatibility with old version.**

Notice: it's better to use `isPrinterNull()` method to check if the class BluetoothPrinter is Null when you use device name or macAddress to initialize the Bluetooth printer.

2. Open And Colse Connection method:

- a) `openConnection()` establish connection with the printer. `(open()`

method is deprecated)

- b) `closeConnection()` close connection with the printer. (`close()` method is deprecated)

3. Print Method:

- a) `printText(String content)`

Print common text. The parameter is the content. (`send()` method is deprecated)

- b) `printByteData(byte[] content)`

Print byte data. using this method developer can send printing commands to printer directly, see the following sample:

```
byte[] command = new byte[3];  
command[0] = 27;  
command[1] = 97;  
command[2] = 49;  
printByteData(command);
```

- c) `printImage(String path/Bitmap bm)`

Print image. The parameter is image file name or the path name.

- d) `PrintBarCode(String code)`

Print Bar Code. The parameter is bar code number.

If bar code type is Two-dimensional, the parameter need split by comma,

Call `setBarCode()` to set a bar code.

- e) `PrintTable(Table table)`

Print table data. The parameter is a class of Table, for details of class Table see part D.

- f) `PrintTitle()`

Print title. call the method of `setTitle()` to set title, sub title and icon.

4. Set Method:

- a) `setHandler(Handler handler)`

Pass a Handler for receive message of connection, the message contains connecting, connect success, and connect failed. The const value is:

`Handler_Connect_Connecting`, `Handler_Connect_Success` ,

`Handler_Connect_Failed`, `Handler_Connect_Closed`.

Handler_Message_Error, Handler_Message_Read

b) `setEncoding(String encoding)`

Set character encoding of the printer. Default is GBK。

c) `setTitle(String title, String subTitle, Bitmap logo)`

Set company title. Parameters is: title, sub title, company logo.

d) `setBarCode(int param1, int param2, int param3, byte type)`

Set bar code param:

type: bar code type。Default is CODE128. Const value start with
BAR_CODE_TYPE_, All const value is:

One-dimensional:

UPC_A, UPC_E, JAN13, JAN8, CODE39,
ITF, CODABAR, CODE93, CODE128。

Two-dimensional:

PDF417, DATAMATRIX, QRCODE。

Bar Code type is One-dimensional, the other param is:

param1: bar code width, $2 \leq n \leq 6$, default is 2.

param2: bar code height, $1 \leq n \leq 255$, default is 162.

param3: bar code note position, 0-don't print, 1-above, 2-below, 3-both above
and below.

Bar Code type is Two-dimensional, the other param is:

1. PDF417

param1: The characters per line, $1 \leq n \leq 30$ 。

param2: Error correction level, $0 \leq n \leq 8$ 。

param3: Longitudinal magnification。

2. DATA MATRIX

param1: height, $0 \leq n \leq 144$ (0:auto select)。

param2: width, $8 \leq n \leq 144$ (when param1 is zero, param2 Invalid)。

param3: Longitudinal magnification。

3. QR CODE

param1: Graphical version, $1 \leq n \leq 30$ (0:auto select)。

param2: Error correction level,

$n = 76, 77, 81, 72 (L: 7\%, M: 15\%, Q: 25\%, H: 30\%)$.

param3: Longitudinal magnification.

e) *setCurrentPrintType(PrinterType currentPrintType)*

Set printer type, parameter is PrinterType, such as T9,T3.

f) *setCharacterMultiple(int x, int y)*

Set character width and height. Parameter x is width, y is height. $0 \leq x, y \leq 7$, default is 0.

g) *setLeftMargin(int nL, int nH)*

Set left edge distance of print area, usually nH value is 0.

h) *setPrintModel(boolean isBold, boolean isDoubleHeight,
boolean isDoubleWidth, boolean isUnderLine)*

Set print model. Parameters are:

isBold: bold

isDoubleHeight: double height.

isDoubleWidth: double width.

isUnderLine: under line.

i) *setPrinter(int command)*

Set the printer. Parameter is printer command. All const value of command is :

COMM_INIT_PRINTER: init printer(equal to method init())

COMM_WAKE_PRINTER: wake up printer

COMM_PRINT_AND_RETURN_STANDARD: page model print and return to standard

COMM_PRINT_AND_NEWLINE: print and move to next line.

COMM_PRINT_AND_ENTER: print and enter.

COMM_MOVE_NEXT_TAB_POSITION: move to the position of next tab.

COMM_DEF_LINE_SPACING: restore default line space.

j) *setPrinter(int command, int value)*

Set the printer. First parameter is printer command; the second command is value of command.

COMM_PRINT_AND_WAKE_PAPER_BY_LNCH: print and wake paper "value "

height (Inch)

`COMM_PRINT_AND_WAKE_PAPER_BY_LINE`: print and wake paper “value” lines

`COMM_CLOCKWISE_ROTATE_90`: clock wise rotate 90degree, 0-false, 1-true

`COMM_LINE_HEIGHT`: set line height

`COMM_CHARACTER_RIGHT_MARGIN`: set character right margin

`COMM_ALIGN`: align model. Three model’s const value is:

`COMM_ALIGN_LEFT`: left margin

`COMM_ALIGN_CENTER`: center margin

`COMM_ALIGN_RIGHT`: right margin

k) `setAutoReceiveData` (boolean auto)

Set whether to automatically receive the return value, if the auto is true, you need to set Handler, which calls `setHandler` method, you receive a return value, it will send `Handler_Message_Read` message, where `msg.arg1` is to receive data length, `msg.obj` is specific data .

```
int length = msg.arg1;
```

```
byte[] bytes = (byte[])msg.obj;
```

```
String recStr = new String(bytes);
```

`Handler_Message_Error` :Receive data error message indicates that, in general, is a connection problem.

C. `PrintGraphics` provide method:

1. `initCanvas`(int width)

Init the canvas. Parameter is canvas width.

2. `initPaint`()

Init the paint.

3. `init`(PrinterType printerType)

Init method contains above two methods. Parameter is PrinterType.

If use this method. The canvas was init to max width. Such as pass T9, the width is 72mm.

4. `setFontProperty`(FontProperty fp)

Set font property. Parameter is a FontProperty type,

FontProperty is a collection of font property. User can call method of `setFont()` to set detail property.

If don't use this method, you also can use the following method:

`setLineWidth(float w)` set paint width.

`setTextSize(int size)` set text size.

`setItalic(boolean italic)` set whether italic.

`setStrikeThruText(boolean strike)` set whether strikethrough.

`setUnderlineText(boolean underline)` set whether under line.

`setFakeBoldText(boolean fakeBold)` set fake bold (only set this can't print in Chinese.)

5. `drawText(float x, float y, String nStr)`

Draw text on the canvas. Parameters x and y is text coordinate in the left bottom corner. Y must greater than 0.

6. `drawLine(float startX, float startY, float stopX, float stopY)`

Draw a line. Parameters startX, startY is start coordinate; stopX, stopY is end coordinate.

7. `drawRectangle(float left, float top, float right, float bottom)`

Draw a rectangle. Parameters are the distance of edge to the left and top.

8. `drawEllips(float left, float top, float right, float bottom)`

Draw an ellipse. Parameters is coordinate of edge which a bounding rectangle of the ellipse.

9. `drawImage(float left, float top, String path)`

Draw an Image. Parameters left and top is image's coordinate in the left top corner. "path" is image path.

10. `drawImage(float left, float top, Bitmap image)`

Draw an Image. "image" is bitmap file of image.

11. `drawImage(float left, float top, Context context, int imageId)`

Draw an Image. "context" is application context. "imageId" is resource id of the image.

12. `saveCanvasImage(String path)`

Save the canvas image. Parameter is the path (`printPng()` is deprecated).

13. `getCanvasImage()`

Get the canvas image. Return a bitmap (`printDraw()` and `printDrawDot()` are deprecated).

14. `getCharacterWidth(int textSize)`

Get text size of the special character width. Parameter is text size.

D. Table

1. `Table(String column, String regular, int[] columnWidth)`

Parameter column is table title column, separate by the regular. Such as: "index, unit price, number, price".

Parameter regular: the separator of the column data. Such as ",", ".".

Parameter Column width: width of all columns. One Chinese character width is 2, one English character is 1.

2. `add(String row)`

Add a row data to the table. Data form should equals with table title. If the table cell width exceeds the limit, printer can word wrap, if want manual line, can add "\n" in where you want.

3. `get(int location)`

Get table data of specified line. 0 is table title.

4. `getTableData()`

Get a collection of table data. Return value is List.

5. `setHasSeparator(boolean hasSeparator)`

Set whether has separator of "=====" before and end of the table.

6. `hasSeparator()`

Judge whether the table has set print separator.

E. FontProperty:

1. **setFont(boolean bBold, boolean bItalic, boolean bUnderLine, boolean bStrikeout, int iSize, Typeface sFace)**

Set font, parameters are:

- 1: set whether bold.
- 2: set whether italic.
- 3: set whether has under line.
- 4: set whether has strikethrough
- 5: set font size.
- 6: set font type (usually set null, use default font)

2. **initTypeface(Context mContext,String path)**

Init the font type. The first parameter is the application context; the second parameter is the file name of the font data in the assets directory

3. **initTypefaceToString(String familyName, int style)**

Init the font type. The first parameter is the name of the font family. Such as "DroidSerif-Bold", the second parameter is the style (normal, bold, italic) of the typeface. e.g. NORMAL, BOLD.

4. **setBold(boolean bBold)**

Set whether bold.

5. **setItalic(boolean bItalic)**

Set whether italic.

6. **setUnderLine(boolean bUnderLine)**

Set whether has under line.

7. **setStrikeout(boolean bStrikeout)**

Set whether has strikethrough

8. **setSize(int iSize)**

Set font size.

9. **setFace(Typeface sFace)**

Set font type.