CoiNel USB Printer
Android SDK User Manual

Version: 1.1.0

CoiNel USB Printer - Android SDK User Manual

Table of Contents

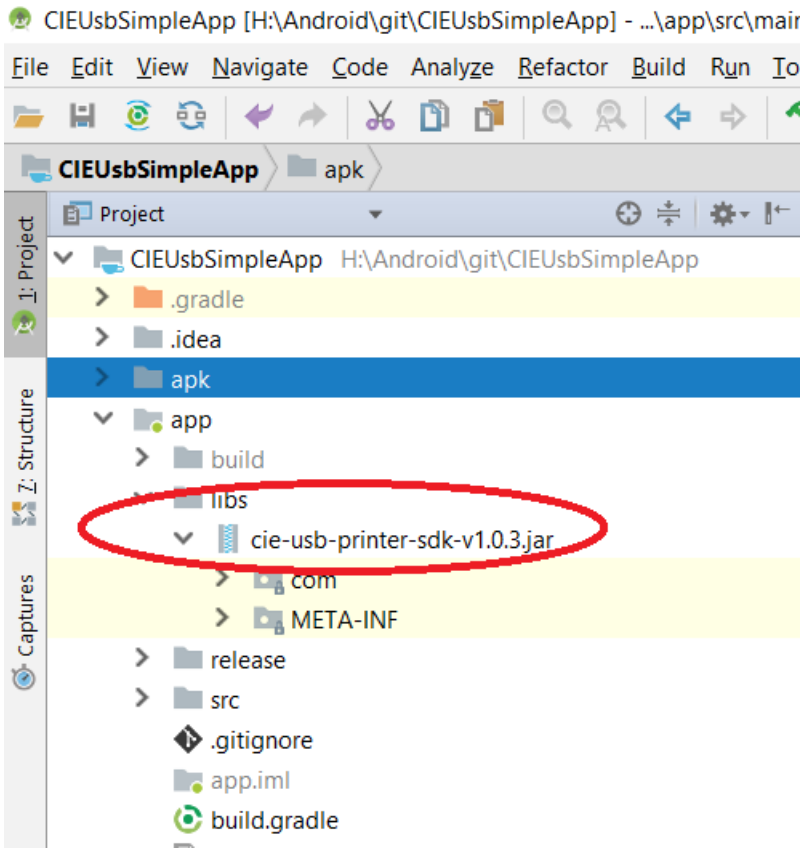
CoiNel USB Printer - Android SDK User Manual

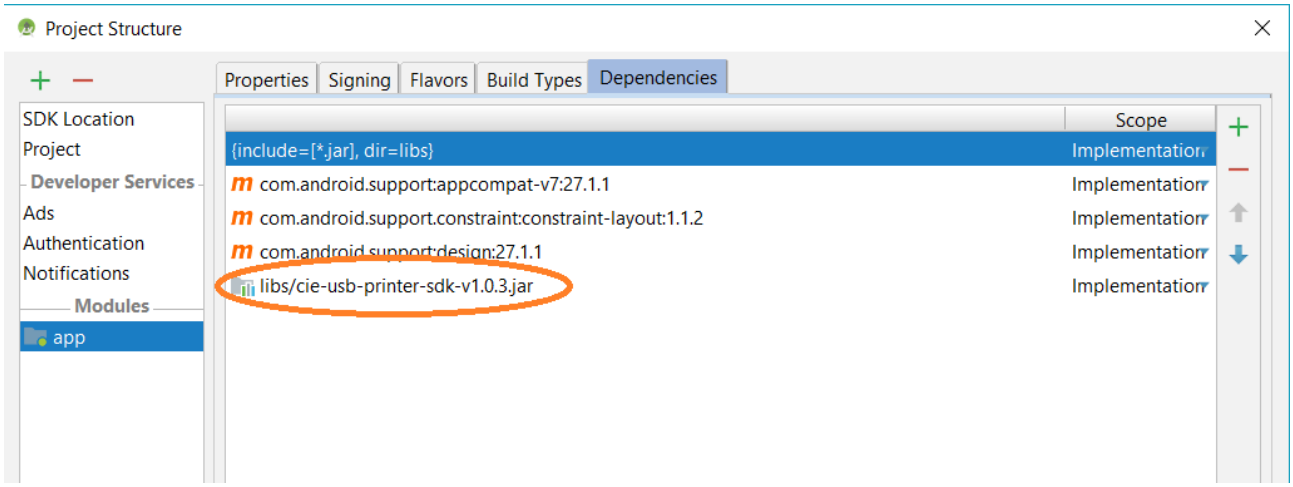
Introduction

This document describes the android SDK for the COINEL USB printer. This SDK supports android version 14 (ICS – Icecream Sandwich) and above. The COINEL USB printer SDK is a jar library that you have to add to your android project. The accompanying android demo project code shows how to use all the functionality of the COINEL USB printer.

Adding SDK to your android project in Android Studio

After creating your android project in android studio. Create a Libs folder, if not already present and copy the SDK Jar into that folder. Open Module Settings (F4) add the jar file as a dependency.





Using COINEL SDK in your android project

Using the Coinel USB SDK is pretty simple. Please refer to the CIE USB Print Demo App in github for sample code implementation.

Approach to printer integration.

The Printer that you plan to integrate uses USB Technology for communication with your Android device.

Please remember your Android Device should support USB OTG / USB Host

As you already know, an “Activity” in an android app is active only when it is in foreground, if an activity goes into background for some reason (like you get a call or home button is pressed or some other app starts etc...) the android systems calls methods to dismantle the activity. In some cases this dismantlement is only partial; the activity still resides in memory (such as when the user switches to another app), and can still come back to the foreground. If the user returns to that activity, the activity resumes from where the user left off. The system’s likelihood of killing a given process—along with the activities in it—depends on the state of the activity at the time. Activity state and ejection from memory provides more information on the relationship between state and vulnerability to ejection.

Your App Architechture / design can contain one or more **activities** and can also contain **fragments**.

Fragments by design are transient in nature, this means that a fragment can be started within an activity and closed before that Activity is completed. An activity has a bigger life cycle and can better controlled.

Therefore it is not advised to have any printer integration code in a fragment.

Always only an activity should contain the printer integration code.

Please note that an Activity which contains printer integration code may start / trigger another activity, such situation should be avoided, if not avoidable suitable precautions

should be taken such that the printer is disconnected before switching to another activity.

Here is the list of things to do for printer integration

1) Android Manifest File

please add the following permissions, which allows access to USB for your app.

```
<uses-feature android:name="android.hardware.usb.host" android:required="false"/>
```

2) connecting to the Printer

An activity which contains printer integration code should override at least the following methods

a) onCreate – should contain the printer initialization code.

```
mPrinter.initService(MainActivity.this);
```

b) onResume – call the printer's onResume method

```
mPrinter.onActivityResume();
```

c) onPause – call printers onPause method

```
mPrinter.onActivityPause();
```

d) onStart – Register a Local Broadcast Receiver, to receive messages from printer.

```
IntentFilter intentFilter = new IntentFilter();
intentFilter.addAction(RECEIPT_PRINTER_MESSAGES);
LocalBroadcastManager.getInstance(this).registerReceiver(ReceiptPrinterMessageRece
iver, intentFilter);
```

e) onStop – Stop Listening to the Local Broadcast Receiver.

```
LocalBroadcastManager.getInstance(this).unregisterReceiver(ReceiptPrinterMessageRe
ceiver)
```

for details, please refer to the sample code.

3) Listening to messages from the printer.

Once the printer initialized, the printer SDK keeps sending any status and other messages from the printer via Local Broadcast.

When the Local Broadcast Receiver is registered it receives various messages from the printer. Please see the demo code for the details.

4) Printing data to the printer.

Communicating with the printer is always an asynchronous job. When you send a command it takes time to get a reply. Hence the broadcast receiver helps in receiving the messages.

The best way to handle a print job would be

a) collect the data to be printed, either in an activity / fragment and using network etc..

b) send all the collected data to the base activity where the printer integration is done.

c) connect to the printer

d) format the print data and send commands to the printer (via a async task thread)

e) disconnect the printer

f) if needed exit the base activity where the printer connection is handled.

g) to handle another print job, iterate through the cycle again.

Printing anything to the printer will also take time to complete the print job. Hence it is advisable to use an Asynchronous task to do the main printing. As this would free the Main UI thread of the print load. This is clearly shown in the demo app.

Printer Lifecycle Methods

```
initService(Context context);
```
This method is to be called in the **onCreate** method of the activity. This will initialize the connection to the printer.

```
OnActivityResume();
```
This method is to be called in the **onResume** method of the activity. This will ensure proper connection to the printer.

```
OnActivityPause();
```
This method is to be called in the **onPause** method of the activity. This will ensure proper connection to the printer.

Generic Printer Methods

```
resetPrinter();
```

This command is used to reset the printer settings returning it to the default state.

```
printerTest();
```

This command will print default information and test all commands internally and print out the status.

```
setPrinterWidth(PrinterWidth pw);
```

This will set the printer size. Has to called before sending any other command.

PrinterWidth is a enum, exposed by the SDK which can be either one of

PRINT_WIDTH_48MM – to be used for 2 Inch Printer.

PRINT_WIDTH_72MM – to be used for 3 Inch Printer.

PRINT_WIDTH_104MM – to be used for 4 Inch Printer.

```
String GetBatteryStatus();
```
This method returns the battery charge percentage as text. It also broadcasts the same message.

```
String getFirmwareVersion();
```
This method returns the printer firmware version. It also broadcasts the same message.

```
String getPaperStatus();
```
This method returns if the paper is present or not. It also broadcasts the same message.

```
String getPlattenStatus();
```
This method returns if the platten (printer paper tray cover) is closed or not. It also broadcasts the same message.

```
int getPrinterStatus();
```
This returns **1** if it is connected, else returns **0**

```
String getPrintHeadTemperature();
```
This method returns the printers print head temperature. It is also broadcast as a message.

```
setAutoOffTime(int iAutoOffMin);
```
This method sets the time in minutes the printer will automatically turn off (to save battery). If the input minutes is 0, then the printer stays always on. If the value is between 1 – 60 minutes then the auto time off is set.

```
setDebugService(boolean dbg);
```
This methods turns on/off the SDK debug messages, which can be seen in the logcat. Please enable this only in development version.

```
setPrintMode(int mode);
```
This method sets the batch print method on/off, 1 to switch on and 0 to switch off. This method is not used often, should be used only in special cases.

```
batchPrint();
```
This method, flushes all the batched command to the printer. To be used only when print mode is set to 1 (batch mode).This method is not used often, should be used only in special cases.

```
sendBytes(byte[] b);
```

This method is used to send ESC/POS print commands directly to the printer. This is usually not used, to be used with caution.

Text Printing & Formating Methods

```
printTextLine(String txt);
```

Prints the given ascii text, please note you can print a maximum of 42 / 48 / 72 characters on a single line for 2 / 3 / 4 Inch Printer respectively.

```
printLineFeed();
```
Print Line Feed, move forward by a line.

```
pixelLineFeed(int n);
```
Feeds paper of n Pixels. n should be between 0 - 255

```
setCharRightSpacing(int n);
```
Sets the right-side character spacing to n vertical unit. n should be between 0 – 255. Once set this will be effective, until the printer is reset / switched off.

```
setLineSpacing(int n);
```
Sets the line spacing to n lines. n should be between 0 – 255. Once set this will be effective, until the printer is reset / switched off.

```
setTab();
```
This command inserts a tab at the current print position. Usually used for column allignment.

```
setAlignmentRight();
setAlignmentCenter();
setAlignmentLeft();
```
These commands makes the printer print the given text with the requested alignment.

```
setBoldOn();
setBoldOff();
```
This command sets/unsets the print font boldness.

```
setFontStyle(boolean bBold, boolean bUnderline, FontStyle fs, FontType fontType);
```

This command is used to set font styles. **Bold** can be set on/off, text **Underline** can be set to on/off.

FontStyle is an enum exposed by the SDKwhich can set the style to

NORMAL, DOUBLE_WIDTH, DOUBLE_HEIGHT, DOUBLE_WIDTH_HEIGHT

FontType is an enum exposed by the SDK which selects the font to be printed from one of the two

FONT_A, FONT_B

```
setPrintDensity(PrintDensity pd);
```

This command sets the print density to the desired level the PrintDensity is an enum which is exposed by the SDK with the values FADE, NORMAL, STRONG;

Print (Unicode) Text

```
printUnicodeText(txt);

printUnicodeText(txt, Layout.Alignment a, TextPaint tp);
```

Input:

String text : the Unicode text to print

Alignment : Left / Center / right Alignment.

TextPaint : Text Paint Object, which defines the text properties.

Table Printing

SDK Version 2.1.5 and above supports printing of data in table format. Tables from 2 to 6 Columns can be printed. See the demo source code for easy understanding.

```
PrintTable(PrintColumnParam pcp1, PrintColumnParam pcp2);

PrintTable(PrintColumnParam pcp1, PrintColumnParam pcp2,PrintColumnParam pcp3);

PrintTable(PrintColumnParam pcp1, PrintColumnParam pcp2,PrintColumnParam pcp3,PrintColumnParam pcp4);

PrintTable(PrintColumnParam pcp1, PrintColumnParam pcp2, PrintColumnParam pcp3, PrintColumnParam pcp4, PrintColumnParam pcp5);

PrintTable(PrintColumnParam pcp1, PrintColumnParam pcp2, PrintColumnParam pcp3, PrintColumnParam pcp4, PrintColumnParam pcp5, PrintColumnParam pcp6);
```

Input - PrintColumnParam : Table Col x Data. The PrintColumnParam is a bean (data structure) which contains all the relevant content and formatting information for the column.

```
PrintColumnParam {
  int iColWidthPct;       //Width of the column in percentage.  It is a good practice
                          //that all col width add up to 100.
  int iFontSize;          //Required Font Size
  String[] sColData;      //Col Data to be printed as a String Array, 1 element for
                          //each row.
  Layout.Alignment alignment; // Cell Alignment
  Typeface typeFace;      // Specific type face (Font) to be used.
}
```

Print Binarized (Bitonal) Image

```
printBinarizedImage (String path, boolean invert, int threshold, int align);
```

Input:

String path : image file path

boolean invert : to invert the image or not,

int threshold : Image threshold, pass 0 to auto calculate

int align : image alignment

0 – Left align , 1 – Center align, 2 Right align.

Returns : true when the image is printed successfully

Print Binarized (Bitonal) Image

```
printBinarizedImage (Bitmap bitmap, boolean invert, int threshold, int align);
```

Input:

Bitmap bitmap : Image Bitmap

boolean invert : to invert the image or not,

int threshold : Image threshold, pass 0 to auto calulate

int align : image alignment

0 – Left align , 1 – Center align, 2 Right align.

Returns : true when the image is printed successfully

Save Binarized Image

```
saveBinarizedImage (String file, boolean bInvertBitmap, int threshold,
int imageAlign, int imageId);
```

Input:

String file : image file path,

boolean invert : To invert the image or not,

int threshold : Image threshold, pass 0 to auto calulate

int imageAlign : image alignment

0 – Left align , 1 – Center align, 2 Right align.

Int imageId : Image Id, can be 1 – 9 (used when printing saved images.)

Returns : true when the image is saved successfully

Print Gray scale Image

```
printGrayScaleImage (String path, int align);
```

Input:

String path : image file path

int align : image alignment

0 – Left align , 1 – Center align, 2 Right align.

Returns : true when the image is printed successfully

Print Gray scale Image

```
printGrayScaleImage (Bitmap bitmap, int align);
```

Input:

Bitmap bitmap : Image Bitmap

int align :image alignment

0 – Left align , 1 – Center align, 2 Right align.

Returns : true when the image is printed successfully

Save Gray Scale Image

```
saveGrayScaleImage (String file, int imageAlign, int imageId);
```

Input:

String file : image file path,

int imageAlign : image alignment

0 – Left align , 1 – Center align, 2 Right align.

Int imageId : Image Id, can be 1 – 9 (used when printing saved images.)

Returns : true when the image is saved successfully

Print Barcode

```
printBarcode (String data, Barcode type, int width, int height, int imageAlign);
```

Input:

String data : Barcode data,

Barcode type : Barcode format, see below for supported formats.

int Width : width of the barcode (max width should not exceed paper width in pixels),

int Height : height of the barcode.

Int imageAlign : barcode image alignment

0 – Left align , 1 – Center align, 2 Right align.

Returns : true if the bar code printed successfully

Supported Barcode Formats : AZTEC, CODABAR, CODE_39, CODE_93, CODE_128, DATA_MATRIX, EAN_8, EAN_13, ITF, MAXICODE, PDF_417, RSS_14, RSS_EXPANDED, UPC_A, UPC_E and UPC_EAN_EXTENSION

Print QR Code

```
printQRcode (String data);
printQRcode (String data, int imageAlign);
printQRcode (String data, int qrCodeWidth, int imageAlign);
```

Input:

String data : Barcode data,

Int qrCodeWidth : Width of the QR Code to be printed.

Int imageAlign : barcode image alignment

0 – Left align , 1 – Center align, 2 Right align.

Returns : true if the bar code printed successfully

Developer Notes

Sample Code to print a bill

```
// Bill Header Start
mPrinter.setAlignmentCenter();
mPrinter.setBoldOn();
mPrinter.setCharRightSpacing(10);
mPrinter.printTextLine("\nMY COMPANY BILL\n");
mPrinter.setBoldOff();
mPrinter.setCharRightSpacing(0);
mPrinter.printTextLine("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
mPrinter.pixelLineFeed(50);
// Bill Header End
// Bill Details Start
mPrinter.setAlignmentLeft();
mPrinter.printTextLine("Customer Name     : Aditya    \n");
mPrinter.printTextLine("Customer Order ID : 00067     \n");
mPrinter.printTextLine("------------------------------\n");
mPrinter.printTextLine("  Item      Quantity     Price\n");
mPrinter.printTextLine("------------------------------\n");
mPrinter.printTextLine("  Item 1           1       1.00\n");
mPrinter.printTextLine("  Bags            10    2220.00\n");
mPrinter.printTextLine("  Next Item      999   99999.00\n");
mPrinter.printLineFeed();
mPrinter.printTextLine("------------------------------\n");
mPrinter.printTextLine("  Total              107220.00\n");
mPrinter.printTextLine("------------------------------\n");
mPrinter.printLineFeed();
mPrinter.setFontStyle(true,true, FontStyle.DOUBLE_HEIGHT, FontType.FONT_A);
mPrinter.printTextLine("    Thank you ! Visit Again   \n");
mPrinter.setFontStyle(false,false, FontStyle.NORMAL, FontType.FONT_A);
mPrinter.printLineFeed();
mPrinter.printTextLine("********************************\n");
mPrinter.printLineFeed();
mPrinter.printTextLine("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n");
mPrinter.printLineFeed();
// Bill Footer End
//Clearance for Paper tear
mPrinter.printLineFeed();
mPrinter.printLineFeed();
mPrinter.resetPrinter();
```

Printing an Image

We can print directly any image of image type jpg, png, bmp. Here is the command to print image directly

```
int imageAlignment = 1; // Center Align
int threshold = 0; // Auto Calculate
boolean r;
if (bImgAlgoGrayScale) {
     r = mPrinter.printGrayScaleImage(fileUri.getPath(), imageAlignment);
}
else {
     r = mPrinter.printBinarizedImage(fileUri.getPath(), Invert, threshold,
imageAlignment);
}
if (r) {
     Toast.makeText(this, "Image Printed", Toast.LENGTH_SHORT).show();
}
else {
     Toast.makeText(this, mPrinter.getPrinterStatusMessage(), Toast.LENGTH_SHORT).show();
}
```

Saving an Image

We can save any image to Printer , the image will be retained in memory even after printer is switched off. Here is the command to print image

```
Boolean Invert;
int imageAlignment = 1; // Center Align
int threshold = 0; // Auto Calculate
Invert = !bInvertBitmap;
boolean r;

if (bImgAlgoGrayScale) {
     r = mPrinter.saveGrayScaleImage(fileUri.getPath(), imageAlignment, indexNumber);
}
else {
     r = mPrinter.saveBinarizedImage(fileUri.getPath(), Invert, threshold,
imageAlignment, indexNumber);
}
if (r) {
     Toast.makeText(this, "Image saved on sImageStoreIndex " + indexNumber,
               Toast.LENGTH_SHORT).show();
}
else {
     Toast.makeText(this, mPrinter.getPrinterStatusMessage(), Toast.LENGTH_SHORT).show();
}
```

Image Printing

For a 2 Inch Printer the max image width is 384 pixels.

For a 3 Inch Printer the max image width is 576 pixels

For a 4 Inch Printer the max image with is 832 pixels.

The with of the image should always be a multiple of 8.

If the input image is bigger in size then it will be scaled proportionately based on the aspect ratio of the image.

If there are lines in the image, ensure that they are at least 2 pixels in width to ensure visibility.

If the image contains letters and characters, please endure they are bold and legible, this will make the letters to be printed clear and understandable.

Printing Barcodes

We can Print the barcode, pass the parameters barcode data, barcode format, width, height to print bar code as shown below.

mPrinter.printBarcode(txt, BarcodeFormat.***CODE_128***, ***BARCODE_WIDTH***, ***BARCODE_HEIGHT, ImageAlign***);

the printed barcode will look like

The following barcode formats are supported AZTEC, CODABAR, CODE_39, CODE_93, CODE_128, DATA_MATRIX, EAN_8, EAN_13, ITF, MAXICODE, PDF_417, RSS_14, RSS_EXPANDED, UPC_A, UPC_E and UPC_EAN_EXTENSION

If you need to print HRI (Human Readable Interpretation) of the Barcode, you have to print the text yourself, or please see the ESC Commands manual and use the ESC Commands to print barcode and HRI.

Printing QR codes

We can print QR code, pass the parameter QR code data to print QR code as shown below

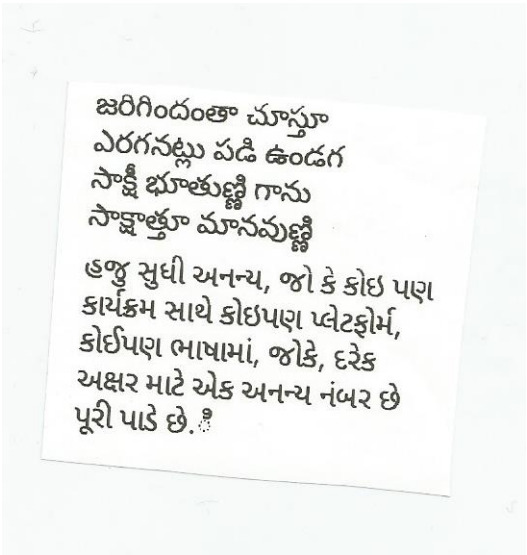
```
mPrinter.printQRcode(data);

mPrinter.printQRcode(data, imageAlign);

mPrinter.printQRcode(data, qrCodeWidth, imageAlign);
```

the printed barcode will look like

Printing in (unicode) various language

You can print in any language of our choice on the printer. If the language is supported by the Android OS and the respective font files are available.

```
printUnicodeText(txt);
```



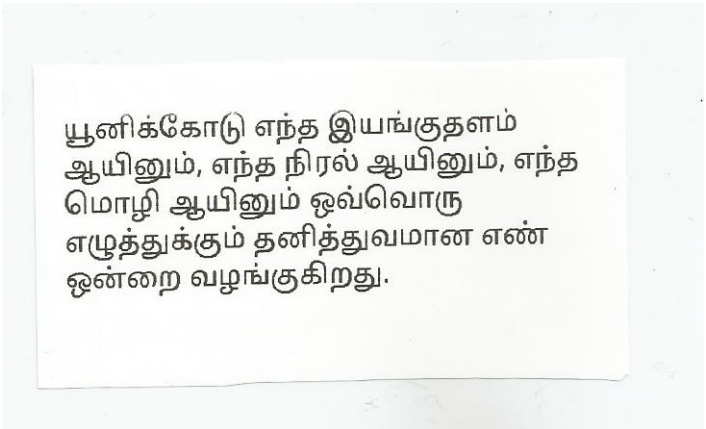


Image Print as 2 Column Table

SDK Version 1.1.0 and above supports printing of 2 Images Side by side. See the demo source code for easy understanding.

```
PrintImageTable(PrintImageColumn pic1, PrintImageColumn pip2);
```

Input - PrintImageColumn : Image Table Col x Data. The PrintImageColumn is a bean (data structure) which contains all the relevant content and formatting information for the image in the column.

```
PrintColumnParam {
  Bitmap bmp;              //Image to be printed as a bitmap. It is a good practice to
                           //have the image as a bitonal Black/White Image
  int iColWidthPct;        //Width of the column in percentage.  It is a good practice
                           //that both col width add up to 100.
  Layout.Alignment alignment; // Cell Alignment
}
```

Black Mark

SDK Version 1.1.0 and above supports finding the next black mark and moving to the start of next form.

```
moveBlackMarkFeed(int mm);
```

Input – mm : After finding the Black mark scroll requested mm to the starting of the next form.

```
findBlackMarkAndMove(int mm);
```

Input – mm : After finding the Black mark scroll requested mm to the starting of the next form.

Returns : true when the black mark is found and moved to successfully

Local Broadcast Receiver

When Integrating the printer we need a Local Broadcast Receiver to listen to all the messages sent by the SDK/Printer.

This is the way that the messages from the Printer and SDK are conveyed to the app.

Here is the sample code (as in the demo app)

```
private final BroadcastReceiver ReceiptPrinterMessageReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        DebugLog.logTrace("Printer Message Received");
        Bundle b = intent.getExtras();
        switch (b.getInt(RECEIPT_PRINTER_STATUS)) {
            case RECEIPT_PRINTER_ATTACHED:
                  tvStatus.setText("Printer Attached");
                  break;
            case RECEIPT_PRINTER_DETATCHED:
                tvStatus.setText("Printer Detached");
                  break;
            case RECEIPT_PRINTER_REQUESTING_PERMISSION:
                tvStatus.setText("Requesting Printer permission");
               break;
            case RECEIPT_PRINTER_PERMISSION_REJECTED:
                tvStatus.setText("Printer Permission Denied");
                break;
            case RECEIPT_PRINTER_PERMISSION_GRANTED:
                tvStatus.setText("Printer Permission Granted; Ready");
                break;
           case RECEIPT_PRINTER_CONN_STATE_CONNECTING:
                tvStatus.setText(R.string.printer_connecting);
                break;
            case RECEIPT_PRINTER_CONN_STATE_CONNECTED:
                tvStatus.setText(R.string.printer_connected);
                new AsyncPrint().execute();
                break;
            case RECEIPT_PRINTER_CONN_DEVICE_NAME:
                savePrinterMac(b.getString(RECEIPT_PRINTER_NAME));
                break;
            case RECEIPT_PRINTER_NOTIFICATION_ERROR_MSG:
                String n = b.getString(RECEIPT_PRINTER_MSG);
                tvStatus.setText(n);
                break;
            case RECEIPT_PRINTER_NOTIFICATION_MSG:
                String m = b.getString(RECEIPT_PRINTER_MSG);
                tvStatus.setText(m);
                break;
            case RECEIPT_PRINTER_NOT_CONNECTED:
                tvStatus.setText("Status : Printer Not Connected");
                break;
            case RECEIPT_PRINTER_NOT_FOUND:
                tvStatus.setText("Status : Printer Not Found");
                break;
            case RECEIPT_PRINTER_SAVED:
                tvStatus.setText(R.string.printer_saved);
                break;
        }
    }
};
```