API Reference Guide

Sato Printer SDK

Android

Ver. 1.00

www.satoworldwide.com

Table of Contents

1. Content Confirmation

This SDK Manual contains the description of the library API that is required for the development of Android applications.

1-1 Supported Android Version

• Android OS 3.1 (Honeycomb, Android API 12) and later.

1-2 List of Supported Printer / Interface

Model	Wi-Fi	Bluetooth	BLE	USB
PV3	O	O	O	O
PV4	O	O	O	O

※BLE : Bluetooth Low Energy

1-3 Numerical notation of Manual

• Numerical notations in this manual are written in decimal, yet for numeric notations presenting with "0x" correspond to hexadecimal numbers.

[Example]

Differentiation between decimal and hexadecimal

Value	Decimal notation	Hex notation
4	4	0x04
10	10	0x0A
76	76	0x4C

1-4 Available range of X/Y coordinates for each model

Model	Min Width	Max Width
PV3	0	576
PV4	0	832

1-5 Package Contents

1-5-1 Manual

• Manual_Sato_Printer_SDK_API Reference Guide_english_V*.**
※ Refer to Manual folder.

1-5-2 Library

Library location / name	Description
libs/Sato_PrinterSDK_V[xxx].jar	Library for enabling the label printer
libs/android-support-v4.jar	Android support library
libs/image/OpenCV-[x.x.xx].jar	Library for printing images
libs/image/CPU-Type/libopencv_java.so	Native library for printing images

※The contents in square brackets change depending on the version of the library.

1-5-3 Sample Source Code

Sample location / name	Description
sample/SatoPrinterSample	Sample application for printer enabling

1-6 List of Supported Methods

Method		Remarks
General	*Printer*	
Search	*findBluetoothPrinter*	
	findNetworkPrinter	
	findUsbPrinter	
Connection	*connect*	
	isConnected	
	disconnect	
Print	*print*	
Text	*drawText*	
	drawVectorFontText	
Barcode	*draw1dBarcode*	
	drawMaxicode	
	drawPdf417	
	drawQrCode	
	drawDataMatrix	
	drawAztec	
	drawCode49	
	drawCodaBlock	
	drawMicroPDF417	
	drawIMBBarcode	
	drawMSIBarcode	
	drawBarcodePlessey	
	drawTLC39Barcode	
	drawRSSBarcode	
Block & Circle	*drawBlock*	
	drawTowBlock	
	drawCircle	
Image	*drawBitmap*	
	drawCompressionImage	
Status	*getStatus*	
Cut	*setAutoCutter*	Cutter mounted model only
Information	*getPrinterInformation*	
	printInformation	
Printer Setting	*initializePrinter*	
	setOrientation	
	setCharacterSet	
	setPrintingType	

Method		Remarks
Printer Setting	*setMargin*	
	setBackFeedOption	
	setLength	
	setBufferMode	
	clearBuffer	
	setRewinder	Rewinder mounted model only
	setSpeed	
	setOffset	
	setDensity	
	setCutterPositionb	
	setWidth	
Direct I/O	*executeDirectIo*	

2. PrinterSDK Class Reference

2-1 Overview

• Printer Class is the main object that controls the printer operation.

2-2 Methods

2-2-1 Printer

Generate the object of Printer.

[Declaration]

• void Printer(Context context, Handler handler, Looper looper);

[Return Value]

• Context context : UI context to enable system service
• Handler handler : Message handler to receive events
• Looper looper : Looper to process message queue
Enter null if message queue is not processed separately

⚠Caution	If the handler is not created, the events from the library cannot be received.

[Example]

```
Public Class SampleClass{
   Printer mPrinter;

   private Handler backHandler;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      …
      backgroundThread.start();
      mPrinter = new Printer(this, backHandler, Looper.myLooper());
   }

   Handler mainHandler = new Handler(){
      @Override
      public void handleMessage(Message msg){
         switch(
            case Printer.MESSAGE_STATE_CHANGE:
            …
            break;
            case …
            …
         }
      }
   }
   private Thread backgroundThread = new Thread(){
      @Override
      public void run() {
         Looper.prepare();
          backHandler = new Handler(Looper.myLooper()){
             @Override
             public void handleMessage(Message msg) {
                switch(msg.what){
                case Printer.MESSAGE_STATE_CHANGE:
                   Message message = new Message();
                    message.what = msg.what;
                    message.obj = msg.obj;
                    message.arg1 = msg.arg1;
                    …
                   mainHandler.sendMessage(message);
                   break;
                 case …
                    …
                    break;
                }
             }
          };
          Looper.loop();
      }
   };
}
```

2-2-2 findBluetoothPrinters

Search for paired Bluetooth printers.

[Declaration]
• void findBluetoothPrinter();

[Return Value]
• Set<BluetoothDevice>: List of paired printers
• null: No paired Bluetooth printer, or Bluetooth on the mobile phone is turned off

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      …
      mPrinter.findBluetoothPrinter();
   }

   private final Handler mHandler = new Handler() {
      @Override
      public void handleMessage(Message msg) {
         switch (msg.what) {
            …
            case Printer.MESSAGE_BLUETOOTH_DEVICE_SET:
               if(msg.obj == null) {
                  Log.i(“TAG”, “Bluetooth Device not found !”);
               } else {
                  Set<BluetoothDevice> devices = (Set<BluetoothDevice>)msg.obj;
               }
               break;
            }
       }
   }
}
```

2-2-3 findNetworkPrinters

Search the network printers.

[Declaration]`
• void findNetworkPrinters(int timeout);

[Return Value]
• int timeout: Printer search time (unit: milliseconds)

[Return Value]
• Set<String>: List of detected Wi-Fi printers
• null: No printer found

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      …
      mPrinter.findNetworkPrinters ();
   }

   private final Handler mHandler = new Handler() {
      @Override
      public void handleMessage(Message msg) {
         switch (msg.what) {
            …
            case Printer.MESSAGE_NETWORK_DEVICE_SET:
               if(msg.obj == null) {
                  Log.i(“TAG”, “Network Device not found !”);
               } else {
                  Set<String> devices = (Set<String>)msg.obj;
               }
               break;
            }
       }
   }
}
```

2-2-4 findUsbPrinters

Search for USB printers.

[Declaration]
• void findUsbPrinters()

[Return Value]
• Set<UsbDevice>: List of detected USB
• null: No USB found

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      …
      mPrinter.findUsbPrinter();
   }

   private final Handler mHandler = new Handler() {
      @Override
      public void handleMessage(Message msg) {
         switch (msg.what) {
            …
            case Printer.MESSAGE_USB_DEVICE_SET:
               if(msg.obj == null) {
                  Log.i(“TAG”, “USB Device not found !”);
               } else {
                  Set<UsbDevice> devices = (Set<UsbDevice>)msg.obj;
               }
               break;
            }
       }
   }
}
```

2-2-5 connect (Only Bluetooth Classic)

This method tries to connect the printer.

[Declaration]
• void connect(String address)

[Return Value]
• String address : Bluetooth Printer Mac Address

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      …
      String address = “74:F0:7F:xx:xx:xx”;
      mPrinter.connect(address);
   }

   private final Handler mHandler = new Handler() {
      @Override
      public void handleMessage(Message msg) {
         switch (msg.what) {
            case Printer.MESSAGE_STATE_CHANGE:
               switch (msg.arg1) {
                  case Printer.STATE_CONNECTED:
                     Log.i(”TAG”, “Device is connected !”);
                  break;
                  case Printer.STATE_CONNECTING:
                     Log.i(“TAG”, “Device is connecting !”);
                  break;
                  case Printer.STATE_NONE:
                     Log.i(“TAG”, “connect is failed or disconnected!”);
                  break;
               }
            }
         }
      }
   }
}
```

2-2-6 connect (Bluetooth Classic / BLE)

This method tries to connect the printer.

[Declaration]
• void connect(String address, int type)

[Return Value]
• String address : Mac Address of Printer
• int type : Bluetooth kind (0 : Classic / 1 : BLE)

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      …
      String address = “74:F0:7F:xx:xx:xx”;
      Int type = Printer.BLUETOOTH_CLASSIC;
      mPrinter.connect(address, type);
   }

   private final Handler mHandler = new Handler() {
      @Override
      public void handleMessage(Message msg) {
         switch (msg.what) {
            case Printer.MESSAGE_STATE_CHANGE:
               switch (msg.arg1) {
                  case Printer.STATE_CONNECTED:
                     Log.i(“TAG”, “Device is connected !”);
                  break;
                  case Printer.STATE_CONNECTING:
                     Log.i(“TAG”, “Device is connecting !”);
                  break;
                  case Printer.STATE_NONE:
                    Log.i(“TAG”, “connect is failed or disconnected!”);
                  break;
               }
         }
      }
   }
}
```

2-2-7 connect (Wi-Fi)

This method tries to connect the printer.

[Declaration]
• void connect (String address, int port, int timeout)

[Return Value]
• String address : IP address of Printer
• int port : Port Number of Printer (Default : 9100)
• int timeout: Maximum connection time of printer

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      …
      String address = “192.168.x.x”;
      Int port = 9100;
      Int timeout = 5000;
      mPrinter.connect(address, port, timeout);
   }

   private final Handler mHandler = new Handler() {
      @Override
      public void handleMessage(Message msg) {
         switch (msg.what) {
            case Printer.MESSAGE_STATE_CHANGE:
               switch (msg.arg1) {
                  case Printer.STATE_CONNECTED:
                     Log.i(“TAG”, “Device is connected !”);
                  break;
                  case Printer.STATE_CONNECTING:
                     Log.i(“TAG”, “Device is connecting !”);
                  break;
                  case Printer.STATE_NONE:
                     Log.i(“TAG”, “connect is failed or disconnected!”);
                  break;
               }
         }
      }
   }
}
```

2-2-8 connect (Only USB)

This method tries to connect the printer.

[Declaration]
• void connect(UsbDevice device)

[Return Value]
• UsbDevice device: Currently connected USB printer

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      …
      mPrinter.connect(UsbDevice);
   }

   private BroadcastReceiver mUsbReceiver = new BroadcastReceiver(){
      @Override
      public void onReceive(Context context, Intent intent) {
         String action = intent.getAction();
         if(UsbManager.ACTION_USB_DEVICE_ATTACHED.equals(action)) {
            Log.i(“TAG”, “USB is connected !”);
         } else if(UsbManager.ACTION_USB_DEVICE_DETACHED.equals(action)) {
            Log.i(“TAG”, “USB is disconnected !”);
         }
      }
   };
}
```

2-2-9 isConnected

This method checks the status of printer connection.

[Declaration]
• boolean isConnected();

[Return Value]
• true : printer is connected
• false : printer is not connected

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      …
      boolean isConnected = mPrinter.isConnected();

      if(isConnected) {
         Log.i(“TAG”, “Device is connected !”);
      } else {
         Log.i(“TAG”, “connect is failed !”);
      }
   }
}
```

2-2-10 disconnect

This method disconnects the printer.

[Declaration]
• void disconnect();

[Example]

```
Public Class SampleClass{
    Printer mPrinter = new Printer(this, mHandler, null);

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        disconnect();
    }
}
```

2-2-11 print

Print the contents of the printer buffer.

[Declaration]
• void print(int set, int copy);

[Return Value]
• int set : Number of label sets
• int copy : Number of label sets

[Example]

```
Public Class SampleClass{
    Printer mPrinter = new Printer(this, mHandler, null);

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        …

        if(mPrinter.isConnected) {
            int set = 1;
            int copy = 1;
            mPrinter.print(set, copy);
        } else {
            return;
        }
    }
}
```

2-2-12 drawText

Draw text (Device Font) on the image buffer

[Declaration]

• void drawText(String data, int horizontalPosition, int verticalPosition, int fontSize
, int horizontalMultiplier, int verticalMultiplier, int rightSpace
, int rotation, boolean reverse, boolean bold, int alignment)

[Return Value]

• String data: String to be printed
• int horizontalPosition : X coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int verticalPosition : Y coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int fontSize : Font Choice

Code	Value	Description
FONT_SIZE_6	48	9 X 15 (dots)
FONT_SIZE_8	49	12 X 20 (dots)
FONT_SIZE_10	50	16 X 25 (dots)
FONT_SIZE_12	51	19 X 30 (dots)
FONT_SIZE_15	52	24 X 38 (dots)
FONT_SIZE_20	53	32 X 40 (dots)
FONT_SIZE_30	54	48 X 76 (dots)
FONT_SIZE_14	55	22 X 34 (dots)
FONT_SIZE_18	56	28 X 44 (dots)
FONT_SIZE_24	57	37 X 58 (dots)
FONT_SIZE_KOREAN1	97	16 X 16 (dots) (ASCII 9 X 15)
FONT_SIZE_KOREAN2	98	24 X 24 (dots) (ASCII 12 X 24)
FONT_SIZE_KOREAN3	99	20 X 20 (dots) (ASCII 12 X 20)
FONT_SIZE_KOREAN4	100	26 X 26 (dots) (ASCII 16 X 30)
FONT_SIZE_KOREAN5	101	20 X 26 (dots) (ASCII 16 X 30)
FONT_SIZE_KOREAN6	102	38 X 38 (dots) (ASCII 22 X 34)
FONT_SIZE_GB2312	109	24 X 24 (dots) (ASCII 12 X 24)
FONT_SIZE_BIG5	110	24 X 24 (dots) (ASCII 12 X 24)
FONT_SIZE_SHIFT_JIS	106	24 X 24 (dots) (ASCII 12 X 24)

• int horizontalMultiplier: The horizontal multiplier of font (range : 1~4)
• int verticalMultiplier: The vertical multiplier of font (range : 1~4)
• int rightSpace: The right space of a character (ex: 5, +3, -10…)
• int rotation : Rotation

Code	Value	Description
ROTATION_NONE	0	No rotation
ROTATION_90_DEGREES	1	90 degrees of rotation
ROTATION_180_DEGREES	2	180 degrees of rotation
ROTATION_270_DEGREES	3	270 degrees of rotation.

• boolean reverse: Reverse style (true : Enable, false : Disable)
• boolean bold: **Bold** style (true : Enable, false : Disable)

• int alignment : Alignment

Code	Value	Description
TEXT_ALIGNMENT_NONE	48	No align
TEXT_ALIGNMENT_LEFT	70	Align to left
TEXT_ALIGNMENT_RIGHT	76	Align to right
TEXT_ALIGNMENT_RIGHT_TO_LEFT	82	Print characters from right to left

[Note]

• Contents requested by this API will be printed when **2-2-11 print API** is called.

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void drawText(){
      mPrinter.drawText("Device Font Text test !!"
                        , 100
                        , 80
                        , Printer.FONT_SIZE_10
                        , 1
                        , 1
                        , 0
                        , Printer.ROTATION_90_DEGREES
                        , false
                        , true
                        , Printer.TEXT_ALIGNMENT_LEFT);
      mPrinter.print(1, 1);
   }

}
```

2-2-13 drawVectorFontText

Draw Vector Font on the image buffer.

[Declaration]

• void drawVectorFontText(String data, int horizontalPosition, int verticalPosition
, int font, int width, int height, int rightSpace
, boolean bold, boolean reverse, boolean italic
, int rotation, int alignment, int direction)

[Return Value]

• String data : String to be printed
• int horizontalPosition : X coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int verticalPosition : Y coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int font : Font select

Code	Value	Description
VECTOR_FONT_ASCII	85	ASCII (1Byte code)
VECTOR_FONT_KS5601	75	KS5601 (2Byte code)
VECTOR_FONT_BIG5	66	BIG5 (2Byte code)
VECTOR_FONT_GB2312	71	GB2312 (2Byte code)
VECTOR_FONT_SHIFT_JIS	74	Shift-JIS (2Byte code)
VECTOR_FONT_OCR_A	97	OCR-A (1Byte code)
VECTOR_FONT_OCR_B	98	OCR-B (1Byte code)

• int width : Font Width (range : 1~1500)
• int height**:** Font height (range : 1~1500)
• int rightSpace: the right space (example : 5, +3, -10…).
• boolean bold: **Bold** style (true : Enable, false : Disable)
• boolean reverse: Reverse style (true : Enable, false : Disable)
• boolean Italic: *Italic style* (true : Enable, false : Disable)
• int rotation : Font Rotation

Code	Value	Description
ROTATION_NONE	0	No rotation
ROTATION_90_DEGREES	1	90 degrees of rotation
ROTATION_180_DEGREES	2	180 degrees of rotation
ROTATION_270_DEGREES	3	270 degrees of rotation.

• int alignment : Alignment

Code	Value	Description
VECTOR_FONT_TEXT_ALIGNMENT_LEFT	76	Align to left
VECTOR_FONT_TEXT_ALIGNMENT_RIGHT	82	Align to right
VECTOR_FONT_TEXT_ALIGNMENT_CENTER	67	Align to center
VECTOR_FONT_TEXT_DIRECTION_LEFT_TO_RIGHT	0	Print characters from left to right
VECTOR_FONT_TEXT_DIRECTION_RIGHT_TO_LEET	1	Print characters from right to left

• int direction : The printing direction of string

[Note]

• Contents requested by this API will be printed when **2-2-11 print API** is called.

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void drawVectorFontText (){
      mPrinter.drawVectorFontText("Vector Font Text Test"
                , 50
                , 100
                , Printer.VECTOR_FONT_KS5601
                , 40
                , 40
                , 0
                , true
                , false
                , false
                , Printer.ROTATION_NONE
                , Printer.VECTOR_FONT_TEXT_ALIGNMENT_LEFT
                , Printer.VECTOR_FONT_TEXT_DIRECTION_LEFT_TO_RIGHT
                );

      mPrinter.print(1, 1);
   }
}
```

2-2-14 draw1dBarcode

Draw 1D Barcode on the image buffer.

[Declaration]

• void draw1dBarcode(String data, int horizontalPosition, int verticalPosition
, int barcodeSelection, int narrowBarWidth
, int wideBarWidth, int height, int rotation, int hri
, int quietZoneWidth)

[Return Value]

• String data : The barcode value to be printed
• int horizontalPosition : X coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int verticalPosition : Y coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int barcodeSelection: Barcode style

Code	Value	Description
BARCODE_CODE39	0	Code39
BARCODE_CODE128	1	Code128
BARCODE_I2OF5	2	Interleaved 2of5
BARCODE_CODABAR	3	Codabar
BARCODE_CODE93	4	Code93
BARCODE_UPC_A	5	UPC-A
BARCODE_UPC_E	6	UPC-E
BARCODE_EAN13	7	EAN13
BARCODE_EAN8	8	EAN8
BARCODE_UCC_EAN128	9	UCC/EAN128
BARCODE_CODE11	10	Code11
BARCODE_PLANET	11	Planet
BARCODE_INDUSTRIAL_2OF5	12	Industrial 2of5
BARCODE_STANDARD_2OF5	13	Standard 2of5
BARCODE_LOGMARS	14	Logmars
BARCODE_UPC_EAN_EXTENSIONS	15	UPC/EAN Extensions
BARCODE_POSTNET	16	Postnet

• int narrowBarWidth: The width of the narrow bar
• int wideBarWidth: The width of the wide bar
• int height : Barcode height
• int rotation : Barcode Rotation

Code	Value	Description
ROTATION_NONE	0	No rotation
ROTATION_90_DEGREES	1	90 degrees of rotation
ROTATION_180_DEGREES	2	180 degrees of rotation
ROTATION_270_DEGREES	3	270 degrees of rotation.

• int hri: The print position of barcode value

Code	Value	Description
HRI_NOT_PRINTED	0	HRI is not used.
HRI_BELOW_FONT_SIZE_1	1	Position of HRI: Below barcode Font Size: 1
HRI_ABOVE_FONT_SIZE_1	2	Position of HRI: Above barcode Font Size: 1
HRI_BELOW_FONT_SIZE_2	3	Position of HRI: Below barcode Font Size: 2
HRI_ABOVE_FONT_SIZE_2	4	Position of HRI: Above barcode Font Size: 2
HRI_BELOW_FONT_SIZE_3	5	Position of HRI: Below barcode Font Size: 3
HRI_ABOVE_FONT_SIZE_3	6	Position of HRI: Above barcode Font Size: 3
HRI_BELOW_FONT_SIZE_4	7	Position of HRI: Below barcode Font Size: 4
HRI_ABOVE_FONT_SIZE_5	8	Position of HRI: Above barcode Font Size: 4

• int quietZoneWidth : margin (range : 0~20)

[Note]

• Contents requested by this API will be printed when **2-2-11 print API** is called.

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void drawBarcode (){
      mPrinter.draw1dBarcode("1234567890128"
                                    , 90
                                    , 200
                                    , Printer.BARCODE_CODE128
                                    , 1
                                    , 2
                                    , 240
                                    , Printer.ROTATION_NONE
                                    , 0
                                    , 0);

      mPrinter.print(1, 1);
   }
}
```

2-2-15 drawMaxicode

Draw MaxiCode Barcode on the image buffer.

[Declaration]
• void drawMaxicode(String data, Int horizontalPosition, int verticalPosition, int mode)

[Return Value]
• String data : The barcode value to be printed
• int horizontalPosition : X coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• Int verticalPosition : Y coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• Int mode *:* Mode of MaxiCode

Code	Value	Description
MAXICODE_MODE_0	0	MaxiCode Mode 0
MAXICODE_MODE_2	2	MaxiCode Mode 2
MAXICODE_MODE_3	3	MaxiCode Mode 3
MAXICODE_MODE_4	4	MaxiCode Mode 4

[Note]
• Contents requested by this API will be printed when **2-2-11 print API** is called.

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void drawBarcode (){
      mPrinter.drawMaxicode("Maxicode Test"
                            , 100
                            , 100
                            , Printer.MAXICODE_MODE2);
      mPrinter.print(1, 1);
   print();
   }
}
```

2-2-16 drawPdf417

Draw PDF417 Barcode on the image buffer.

[Declaration]

• void drawPdf417(String data, int horizontalPosition, int verticalPosition
, Int maxRow, int maxColumn, int errorCorrection\
, int compression, int hri, int orignPoint, int width
, int height, int rotation)

[Return Value]

• String data : The barcode value to be printed
• int horizontalPosition : X coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int verticalPosition : Y coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int maxRow: Maximum horizontal space (range : 3~90)
• int maxColumn: Maximum vertical space (range : 1~30)
• int errCorrection : Error Correction Level

Code	Value	Description
PDF417_ERROR_CORRECTION_LEVEL0	0	Error Correction Level 0
PDF417_ERROR_CORRECTION_LEVEL1	1	Error Correction Level 1
PDF417_ERROR_CORRECTION_LEVEL2	2	Error Correction Level 2
PDF417_ERROR_CORRECTION_LEVEL3	3	Error Correction Level 3
PDF417_ERROR_CORRECTION_LEVEL4	4	Error Correction Level 4
PDF417_ERROR_CORRECTION_LEVEL5	5	Error Correction Level 5
PDF417_ERROR_CORRECTION_LEVEL6	6	Error Correction Level 6
PDF417_ERROR_CORRECTION_LEVEL7	7	Error Correction Level 7
PDF417_ERROR_CORRECTION_LEVEL8	8	Error Correction Level 8

• int compression : Data Compression Method

Code	Value	Description
DATA_COMPRESSION_TEXT	0	2char / codeword
DATA_COMPRESSION_NUMERIC	1	2.93 char / codeword
DATA_COMPRESSION_BINARY	2	1.2bytes / codeword

• int hri: HRI printing option

Code	Value	Description
PDF417_HRI_NOT_PRINTED	0	Do not print HRI
PDF417_HRI_BELOW_BARCODE	1	Print below barcode

• int barcodeOriginPoint: The reference point of barcode

Code	Value	Description
BARCODE_ORIGIN_POINT_CENTER	0	Specify the reference point as the middle of the barcode
BARCODE_ORIGIN_POINT_UPPER_LEFT	1	Specify the reference point as the upper left corner of the barcode

• int width : Module Width (range : 2~9)
• int height : Barcode Height (range : 4~99)
• int rotation : Barcode Rotation

Code	Value	Description
ROTATION_NONE	0	No rotation
ROTATION_90_DEGREES	1	90 degrees of rotation
ROTATION_180_DEGREES	2	180 degrees of rotation
ROTATION_270_DEGREES	3	270 degrees of rotation.

[Note]
• Contents requested by this API will be printed when **2-2-11 print API** is called.

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void drawBarcode (){

      mPrinter.drawPdf417("PDF417 Test"
                          , 100
                          , 100
                          , 10
                          , 5
                          , Printer.PDF417_ERROR_CORRECTION_LEVEL0
                          , Printer.DATA_COMPRESSION_TEXT
                          , Printer.PDF417_HRI_NOT_PRINTED
                          , Printer.BARCODE_ORIGIN_POINT_CENTER
                          , 5
                          , 50
                          , Printer.ROTATION_NONE);
      mPrinter.print(1, 1);
   }
}
```

2-2-17 drawQrCode

Draw QRCode Barcode on the image buffer.

[Declaration]

• void drawQrCode(String data, int horizontalPosition, int verticalPosition
, int model, int eccLevel, int size, int rotation)

[Return Value]

• String data : The barcode value to be printed
• int horizontalPosition : X coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int verticalPosition : Y coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model).
• int model : QRCode Model

Code	Value	Description
QR_CODE_MODEL1	1	QR Model 1
QR_CODE_MODEL2	2	QR Model 2

• int eccLevel : Error Correction Level

Code	Value	Description
ECC_LEVEL_7	76	Error Correction Level 7
ECC_LEVEL_15	77	Error Correction Level 15
ECC_LEVEL_25	82	Error Correction Level 25
ECC_LEVEL_30	72	Error Correction Level 30

• int size: Barcode size (range : 1~9)
• int rotation : Barcode Rotation

Code	Value	Description
ROTATION_NONE	0	No rotation
ROTATION_90_DEGREES	1	90 degrees of rotation
ROTATION_180_DEGREES	2	180 degrees of rotation
ROTATION_270_DEGREES	3	270 degrees of rotation.

[Note]

• Contents requested by this API will be printed when **2-2-11 print API** is called.

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void drawBarcode (){
      mPrinter.drawQrCode("QR Code Test"
                          , 100
                          , 150
                          , Printer.QR_CODE_MODEL1
                          , Printer.ECC_LEVEL_15
                          , 1
                          , Printer.ROTATION_NONE);
      mPrinter.print(1, 1);
   }
}
```

2-2-18 drawDataMatrix

Draw Data Matrixon the image buffer.

[Declaration]
• void drawDataMatrix(String data, int horizontalPosition, int verticalPosition
, int size, int rotation)

[Return Value]
• String data : The barcode value to be printed
• int horizontalPosition: X coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int verticalPosition: Y coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model).
• int size: Barcode size (range : 1~4)
• int rotation : Barcode Rotation

Code	Value	Description
ROTATION_NONE	0	No rotation
ROTATION_90_DEGREES	1	90 degrees of rotation
ROTATION_180_DEGREES	2	180 degrees of rotation
ROTATION_270_DEGREES	3	270 degrees of rotation.

[Note]
• Contents requested by this API will be printed when **2-2-11 print API** is called.

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void drawBarcode (){
      mPrinter.drawDataMatrix("DataMatrix Test"
                              , 50
                              , 100
                              , 2
                              , Printer.ROTATION_NONE);
      mPrinter.print(1, 1);
   }
}
```

2-2-19 drawAztec

Draw Aztec Barcode the image buffer.

[Declaration]

• void drawAztec(String data, int horizontalPosition, int verticalPosition
, int size, boolean extendedChannel, int eccLevel
, boolean menuSymbol, int numberOfSymbols
, String optionalID, int rotation)

[Return Value]

• String data : The barcode value to be printed
• int horizontalPosition : X coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int verticalPosition : Y coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int size: Barcode size (range : 1~10)
• boolean extendedChannel: Select whether to enable extended channel code
• int eccLevel : Error Correction Level

Error control and symbol size/type	Value
Default error correction level	0
Error correction percentage	1~99

• boolean menuSymbol: Select whether to enable menu symbol
(true:Enable, false:Disable)
• int numberOfSymbols : Number of symbols for structured append: (1 ~ 26)
• String optionalID : Optional ID field for structured append: ID field string
(Maximum 24 character)
• int rotation : Barcode Rotation

Code	Value	Description
ROTATION_NONE	0	No rotation
ROTATION_90_DEGREES	1	90 degrees of rotation
ROTATION_180_DEGREES	2	180 degrees of rotation
ROTATION_270_DEGREES	3	270 degrees of rotation.

[Note]

• Contents requested by this API will be printed when **2-2-11 print API** is called.

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void drawBarcode (){
      mPrinter.drawAztec("Aztec Test"
                         , 50
                         , 100
                         , 4
                         , false
                         , Printer.ECC_LEVEL_15
                         , false
                         , 10
                         , null
                         , Printer.ROTATION_NONE);
      mPrinter.print(1, 1);
   }
}
```

2-2-20 drawCode49

Draw Code49 Barcode the image buffer.

[Declaration]

• void drawCode49(String data, int horizontalPosition, int verticalPosition
, int widthNarrow, int widthWide, int height, int hri
, int startingMode, int rotation)

[Return Value]

• String data : The barcode value to be printed
• int horizontalPosition : X coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int verticalPosition : Y coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int narrowBarWidth: The width of the narrow bar
• int wideBarWidth: The width of the wide bar
• int height : Barcode height
• int hri: HRI printing position of barcode

Code	Value	Description
CODE49_HRI_NOT_PRINTED	0	Not printed
CODE49_HRI_BELOW_BARCODE	1	Below the bar code
CODE49_HRI_ABOVE_BARCODE	2	Above the bar code

• int startingMode : Starting Mode

Code	Value	Description
CODE49_STRING_MODE_REGULAR_ALPHANUMERIC	0	Regular Alphanumeric Mode
CODE49_STRING_MODE_MULTIPLE_READ_ALPHANUMERIC	1	Multiple Read Alphanumeric
CODE49_STRING_MODE_REGULAR_NUMERIC	2	Regular Numeric Mode
CODE49_STRING_MODE_GROUP_ALPHANUMERIC	3	Group Alphanumeric Mode
CODE49_STRING_MODE_REGULAR_ALPHANUMERIC_SHIFT1	4	Regular Alphanumeric Shift 1
CODE49_STRING_MODE_REGULAR_ALPHANUMERIC_SHIFT2	5	Regular Alphanumeric Shift 2
CODE49_STRING_MODE_AUTOMATIC_MODE	7	Automatic Mode

• int rotation : Barcode Rotation

Code	Value	Description
ROTATION_NONE	0	No rotation
ROTATION_90_DEGREES	1	90 degrees of rotation
ROTATION_180_DEGREES	2	180 degrees of rotation
ROTATION_270_DEGREES	3	270 degrees of rotation.

[Note]

• Contents requested by this API will be printed when **2-2-11 print API** is called.

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void drawBarcode (){
      mPrinter.drawCode49("Code49 Test"
                          , 100
                          , 50
                          , 2
                          , 4
                          , 100
                          , Printer.HRI_BELOW_FONT_SIZE_1
                          , Printer. CODE49_STRING_MODE_AUTOMATIC_MODE
                          , Printer.ROTATION_NONE);
      mPrinter.print(1, 1);
   }
}
```

2-2-21 drawCodaBlock

Draw CodaBlock Barcode the image buffer.

[Declaration]

• void drawCodaBlock(String data, int horizontalPosition, int verticalPosition
, int widthNarrow, int widthWide, int height
, boolean securityLevel, Int dataColumns, char mode
, int encode)

[Return Value]

• String data : The barcode value to be printed
• int horizontalPosition : X coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int verticalPosition : Y coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int narrowBarWidth: The width of the narrow bar
• int wideBarWidth: The width of the wide bar
• int height : Barcode height
• boolean securityLevel: Select whether to enable securityLevel
(true : Enable, false : Disable)
• int dataColumns: The number of characters per line (range : 2~62)
• char mode: Barcode printing mode

Code	Value	Description
CODABLOCK_MODE_A	'A'	Enable the character set of code 39
CODABLOCK_MODE_E	'E'	Enable the character set of code 128
CODABLOCK_MODE_F	'F'	Enable the character set of code 128. Function 1 is added automatically.

• int encode: The number of lines to be encoded

Mode	Value
CODABLOCK_MODE_A	1 ~ 18
CODABLOCK_MODE_E	2 ~ 4
CODABLOCK_MODE_F	2 ~ 4

[Note]

• Contents requested by this API will be printed when **2-2-11 print API** is called.

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void drawBarcode (){
      mPrinter.drawCodaBlock("CodaBlock Test"
                                    , 100
                                    , 100
                                    , 2
                                    , 4
                                    , 100
                                    , false
                                    , 10
                                    , Printer.CODABLOCK_MODE_A
                                    , 10);
      mPrinter.print(1, 1);
   }
}
```

2-2-22 drawMicroPDF417

Draw Micro PDF417 Barcode the image buffer.

[Declaration]

• void drawMicroPDF417(String data, int horizontalPosition, int verticalPosition , int moduleWidth, int height, int mode , int rotation)

[Return Value]

• String data : The barcode value to be printed
• int horizontalPosition : X coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int verticalPosition : Y coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int moduleWidth: Module width (range : 2~8)
• int height : Barcode height
• int mode : Mode of MicroPDF417 Barcode

Mode	Number of Data Columns	Number of Data Rows	% of Cws for EC	Max Alpha Characters	Max Digits
0	1	11	64	6	8
1	1	14	50	12	17
2	1	17	41	18	26
3	1	20	40	22	32
4	1	24	33	30	44
5	1	28	29	38	55
6	2	8	50	14	20
7	2	11	41	24	35
8	2	14	32	36	52
9	2	17	29	46	67
10	2	20	28	56	82
11	2	23	28	64	93
12	2	26	29	72	105
13	3	6	67	10	14
14	3	8	58	18	26
15	3	10	53	26	38
16	3	12	50	34	49
17	3	15	47	46	67
18	3	20	43	66	96
19	3	26	41	90	132
20	3	32	40	114	167
21	3	38	39	138	202
22	3	44	38	162	237
23	4	6	50	22	32
24	4	8	44	34	49
25	4	10	40	46	67
26	4	12	38	58	85
27	4	15	35	76	111
28	4	20	33	106	155
29	4	26	31	142	208
30	4	32	30	178	261
31	4	38	29	214	313
32	4	44	28	250	366
33	4	4	50	14	20

• int rotation : Barcode Rotation

Code	Value	Description
ROTATION_NONE	0	No rotation
ROTATION_90_DEGREES	1	90 degrees of rotation
ROTATION_180_DEGREES	2	180 degrees of rotation
ROTATION_270_DEGREES	3	270 degrees of rotation.

[Note]

• Contents requested by this API will be printed when **2-2-11 print API** is called.

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void drawBarcode (){
      mPrinter.drawMicroPDF417("MicroPDF417 Test"
                                    , 100
                                    , 100
                                    , 4
                                    , 100
                                    , 0
                                    , Printer.ROTATION_NONE);
      mPrinter.print(1, 1);
   }
}
```

2-2-23 drawIMBBarcode

Request IMB barcode printing to the image buffer.

[Declaration]
• void drawIMBBarcode(String data, int horizontalPosition, int verticalPosition
, boolean hri, int rotation)

[Return Value]
• String data : The barcode value to be printed
• int horizontalPosition : X coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int verticalPosition : Y coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• boolean hri: Select whether to print HRI (true: Print, false: Not print)
• int rotation : Barcode Rotation

Code	Value	Description
ROTATION_NONE	0	No rotation
ROTATION_90_DEGREES	1	90 degrees of rotation
ROTATION_180_DEGREES	2	180 degrees of rotation
ROTATION_270_DEGREES	3	270 degrees of rotation.

[Note]
• Contents requested by this API will be printed when **2-2-11 print API** is called.

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void drawBarcode (){
      mPrinter.drawIMBBarcode("Inteligent Mail Barcode Test"
                              , 100
                              , 100
                              , true
                              , Printer.ROTATION_NONE);
      mPrinter.print(1, 1);
   }
}
```

2-2-24 drawMSIBarcode

Draw MSI Barcode the image buffer.

[Declaration]

• void drawMSIBarcode(String data, int horizontalPosition, int verticalPosition
, int widthNarrow, int widthWide, int height
, int checkDigit, boolean printCheckDigit, int hri
, int rotation)

[Return Value]

• String data : The barcode value to be printed
• int horizontalPosition : X coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int verticalPosition : Y coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int narrowBarWidth: The width of the narrow bar
• int wideBarWidth: The width of the wide bar
• int height : Barcode height
• int checkDigit: Selection of checkDigit option

Code	Value	Description
MSI_BARCODE_CHECKDIGIT_NONE	0	No Check Digit
MSI_BARCODE_CHECKDIGIT_1MOD10	1	Check Digit 1 Mod 10
MSI_BARCODE_CHECKDIGIT_2MOD10	2	Check Digit 2 Mod 10
MSI_BARCODE_CHECKDIGIT_1MOD11_AND_1MOD_10	3	Check Digit 1 Mod 10

• int printCheckDigit: Select whether to include the check digit in HRI
(true: Include false: Not include)
• int hri: Data value printing position of barcode

Code	Value	Description
HRI_NOT_PRINTED	0	HRI is not used.
HRI_BELOW_FONT_SIZE_1	1	Position of HRI: Below barcode Font Size: 1
HRI_ABOVE_FONT_SIZE_1	2	Position of HRI: Above barcode Font Size: 1
HRI_BELOW_FONT_SIZE_2	3	Position of HRI: Below barcode Font Size: 2
HRI_ABOVE_FONT_SIZE_2	4	Position of HRI: Above barcode Font Size: 2
HRI_BELOW_FONT_SIZE_3	5	Position of HRI: Below barcode Font Size: 3
HRI_ABOVE_FONT_SIZE_3	6	Position of HRI: Above barcode Font Size: 3
HRI_BELOW_FONT_SIZE_4	7	Position of HRI: Below barcode Font Size: 4
HRI_ABOVE_FONT_SIZE_5	8	Position of HRI: Above barcode Font Size: 4

• int rotation : Barcode Rotation

Code	Value	Description
ROTATION_NONE	0	No rotation
ROTATION_90_DEGREES	1	90 degrees of rotation
ROTATION_180_DEGREES	2	180 degrees of rotation
ROTATION_270_DEGREES	3	270 degrees of rotation.

[Note]

• Contents requested by this API will be printed when **2-2-11 print API** is called.

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void drawBarcode (){
      mPrinter.drawMSIBarcode("943457842"
                              , 100
                              , 100
                              , 2
                              , 4
                              , 100
                              , Printer.MSI_BARCODE_CHECKDIGIT_1MOD10
                              , true
                              , Printer.HRI_BELOW_FONT_SIZE_1
                              , Printer.ROTATION_NONE);
      mPrinter.print(1, 1);
   }
}
```

2-2-25 drawPlesseyBarcode

Draw Plessey Barcode the image buffer.

[Declaration]

• void drawPlesseyBarcode(String data, int horizontalPosition, int verticalPosition
, int widthNarrow, int widthWide, int height
, boolean printCheckDigit, int hri, int rotation)

[Return Value]

• String data : The barcode value to be printed
• int horizontalPosition : X coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int verticalPosition : Y coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int narrowBarWidth: The width of the narrow bar
• int wideBarWidth: The width of the wide bar
• int height : Barcode height
• printCheckDigit: Select whether to include the check digit in HRI
(true: Include false: Not include)
• int hri: Data value printing position of barcode

Code	Value	Description
PLESSEY_BARCODE_HRI_NOT_PRINTED	0	Not printed
PLESSEY_BARCODE_HRI_BELOW_BARCODE	1	Below the bar code
PLESSEY_BARCODE_HRI_ABOVE_BARCODE	2	Above the bar code

• int rotation : Barcode Rotation

Code	Value	Description
ROTATION_NONE	0	No rotation
ROTATION_90_DEGREES	1	90 degrees of rotation
ROTATION_180_DEGREES	2	180 degrees of rotation
ROTATION_270_DEGREES	3	270 degrees of rotation.

[Note]

• Contents requested by this API will be printed when **2-2-11 print API** is called.

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void drawBarcode (){
      mPrinter.drawPlesseyBarcode("12345678"
                                  , 100, 100, 2
                                  , 4
                                  , 100
                                  , true
                                  , Printer.HRI_BELOW_FONT_SIZE_1
                                  , Printer.ROTATION_NONE);
      mPrinter.print(1, 1);
   }
}
```

2-2-26 drawTLC39Barcode

Draw TLC39 Barcode the image buffer.

[Declaration]

• void drawTLC39Barcode(String data, int horizontalPosition, int verticalPosition
, int widthN
arrow, int widthWide, int height
, int rowHeightOfMicroPDF417
, int narrowWidthOfMicroPDF417, int rotation)

[Return Value]

• String data : The barcode value to be printed
• int horizontalPosition : X coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int verticalPosition : Y coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int narrowBarWidth: The width of the narrow bar Code39
• int wideBarWidth: The width of the wide bar Code39
• int height : Code39 Barcode height
• int rowHeightOfMicroPDF417: The height of microPDF417 row
• int narrowWidthOfMicroPDF417: The width of microPDF417 narrow bar
• int rotation : Barcode Rotation

Code	Value	Description
ROTATION_NONE	0	No rotation
ROTATION_90_DEGREES	1	90 degrees of rotation
ROTATION_180_DEGREES	2	180 degrees of rotation
ROTATION_270_DEGREES	3	270 degrees of rotation.

[Note]

• Contents requested by this API will be printed when **2-2-11 print API** is called.

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void drawBarcode (){
      mPrinter.drawTLC39Barcode("123456,ABCD12345678901234"
                                , 100
                                , 100
                                , 2
                                , 4
                                , 100
                                , 3
                                , 2
                                , Printer.ROTATION_NONE);
      mPrinter.print(1, 1);
   }
}
```

2-2-27 drawRSSBarcode

Draw RSS Barcode on the image buffer.

[Declaration]

• void drawRSSBarcode(String data, int horizontalPosition, int verticalPosition
, int barcodeType, int magnification, int separator
, int BarHeight, int SegmentWidth, int rotation)

[Return Value]

• String data : The barcode value to be printed
• int horizontalPosition : X coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int verticalPosition : Y coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int barcodeType: RSS barcode type

Code	Value	Description
BARCODE_TYPE_RSS14	0	RSS14
BARCODE_TYPE_RSS14_TRUNCATED	1	RSS14 truncated
BARCODE_TYPE_RSS14_STACKED	2	RSS14 stacked
BARCODE_TYPE_RSS14_STACKED_OMNIDIRECTIONAL	3	RSS14 Stacked omnidirectional
BARCODE_TYPE_RSS_LIMITIED	4	RSS limited
BARCODE_TYPE_RSS_EXPANDED	5	RSS Expanded
BARCODE_TYPE_RSS_UPCA	6	RSS UPC A
BARCODE_TYPE_RSS_UPCE	7	RSS UPC E
BARCODE_TYPE_RSS_EAN13	8	EAN13
BARCODE_TYPE_RSS_EAN8	9	EAN 8
BARCODE_TYPE_RSS_UCC_EAN128_CCAB	10	EAN128 CC-A/B
BARCODE_TYPE_RSS_UCC_EAN128_CCC	11	EAN128 CC-C

• int magnification: Magnification (range : 1~10)
• int separator: The height of separator (range : 1~2)
• int barHeight : Barcode height
• int segmentWidth : segmentWidth (range : 0~22)
• int rotation : Barcode Rotation

Code	Value	Description
BLOCK_OPTION_LINE_OVERWRITING	79	Line Overwriting
BLOCK_OPTION_LINE_EXCLUSIVE_OR	69	Line Exclusive OR
BLOCK_OPTION_LINE_DELETE	68	Line Delete
BLOCK_OPTION_SLOPE	83	Slope(a oblique line)
BLOCK_OPTION_BOX	66	Box

[Note]

• Contents requested by this API will be printed when **2-2-11 print API** is called.

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void drawBarcode (){
      mPrinter .drawRSSBarcode("12345678901|this is composite info"
                                          , 100
                                          , 100
                                          , Printer.BARCODE_TYPE_RSS14
                                          , 2
                                          , 1
                                          , 20
                                          , 10
                                          , Printer.ROTATION_NONE);
      mPrinter.print(1, 1);
   }
}
```

2-2-28 drawBlock

Enter lines, blocks, boxes, and oblique lines in the image buffer.

[Declaration]

• void drawBlock(int horizontalStartPosition, int verticalStartPosition
, int horizontalEndPosition, int verticalEndPosition
, int option, int thickness)

[Return Value]

• int horizontalStartPosition: Horizontal start position (X)
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int verticalStartPosition: Vertical start position (Y)
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int horizontalEndPosition: Horizontal end position (X)
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int verticalEndPosition: Vertical end position (Y)
(Reference: 1-3 Available range of X/Y coordinates for each model).
• int option: Block printing option
• int thickness: Thickness

[Note]

• Contents requested by this API will be printed when **2-2-11 print API** is called.

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void drawBox (){
      mPrinter.drawBlock(0
                         , 0
                         , 100
                         , 100
                         , Printer.BLOCK_OPTION_BOX
                         , 3);
      mPrinter.print(1,1);
   }
}
```

2-2-29 drawTowBlock

Enter two blocks in the image buffer.

[Declaration]

• void drawTowBlock(int horizontalStartPosition, int verticalStartPosition
, int horizontalEndPosition, int verticalEndPosition
, int option, int horizontalStartPositionSquare2
, int verticalStartPositionSquare2, int horizontalEndPositionSquare2
, int verticalEndPositionSquare2, int optionSquare2)

[Return Value]

• int horizontalStartPosition: The horizontal start position of the first block (X)
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int verticalStartPosition: The vertical start position of the first block (Y)
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int horizontalEndPosition: The horizontal end position of the first block (X)
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int verticalEndPosition: The vertical end position of the first block (Y)
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int option : The block printing option of the first block
• int horizontalStartPositionSquare2: The horizontal start position of the second block (X)
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int verticalStartPositionSquare2: The vertical start position of the second block (Y)
(Reference: 1-3 Available range of X/Y coordinates for each model).
• int horizontalEndPositionSquare2: The horizontal end position of the second block (X)
(Reference: 1-3 Available range of X/Y coordinates for each model).
• int verticalEndPositionSquare2: The vertical end position of the second block (Y)
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int optionSquare2: The block printing option of the second block

[Note]

• Contents requested by this API will be printed when **2-2-11 print API** is called.

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void drawBox (){
      mPrinter.drawTowBlock(100
                            , 100
                            , 300
                            , 300
                            , Printer.BLOCK_OPTION_LINE_EXCLUSIVE_OR
                            , 400
                            , 400
                            , 500
                            , 500
                            , Printer.BLOCK_OPTION_LINE_EXCLUSIVE_OR);
      mPrinter.print(1,1);
   }
}
```

2-2-30 drawCircle

Enter a circle in the image buffer.

[Declaration]
• drawCircle(int horizontalStartPosition, int verticalStartPosition, int size, int multiplier)

[Return Value]
• int horizontalStartPosition : X coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int verticalStartPosition : Y coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int size: Printing option

Code	Value	Description	
CIRCLE_SIZE_DIAMETER5	1	40 × 40 (dot)	5 mm
CIRCLE_SIZE_DIAMETER7	2	56 × 56 (dot)	7 mm
CIRCLE_SIZE_DIAMETER9	3	72 × 72 (dot)	9 mm
CIRCLE_SIZE_DIAMETER11	4	88 × 88 (dot)	11 mm
CIRCLE_SIZE_DIAMETER13	5	104 × 104 (dot)	13 mm
CIRCLE_SIZE_DIAMETER21	6	168 × 168 (dot)	21 mm

• int multiplier: Multiplier (range : 1~4)

[Note]
• Contents requested by this API will be printed when **2-2-11 print API** is called.

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void drawCircle (){
      mPrinter.drawCircle(50
                          , 50
                          , Printer.CIRCLE_SIZE_DIAMETER5
                          , 1);
      mPrinter.print(1,1);
   }
}
```

2-2-31 drawBitmap

Enter an image in the image buffer.

[Declaration]
• drawBitmap(String pathname, int horizontalStartPosition, int verticalStartPosition
, int width, int level, boolean dithering)

[Return Value]
• String pathname: Image path
• int horizontalStartPosition : X coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int verticalStartPosition : Y coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int width: The width of image to be printed
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int level: The brightness of image to be printed
• boolean dithering: Whether to apply dithering (true: enable, false : disable)

[Note]
• Contents requested by this API will be printed when **2-2-11 print API** is called.

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void drawImage (){
   String path = "/…";
      mPrinter.drawBitmap(path
                                 , 100
                                 , 100
                                 , 100
                                 , 20
                                 , true);
      mPrinter.print(1,1);
   }
}
```

2-2-32 drawBitmap

Enter an image in the image buffer.

[Declaration]
• drawBitmap(Bitmap bitmap, int horizontalStartPosition, int verticalStartPosition
, int width, int level, boolean dithering)

[Return Value]
• Bitmap bitmap: Image object to be printed
• int horizontalStartPosition : X coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int verticalStartPosition : Y coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int width: The width of image to be printed
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int level: The brightness of image to be printed
• boolean dithering: Whether to apply dithering (true: enable, false : disable)

[Note]
• Contents requested by this API will be printed when **2-2-11 print API** is called.

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void drawImage (){
   Bitmap bitmap = …;
      mPrinter.drawBitmap(bitmap
                                 , 100
                                 , 100
                                 , 100
                                 , 20
                                 , true);
      mPrinter.print(1,1);
   }
}
```

2-2-33 drawCompressionImage

Request image printing to the image buffer (applying image compression algorithm)

[Declaration]

• drawCompressionImage(Bitmap bitmap, int horizontalStartPosition,
int verticalStartPosition, int width, int level, boolean dithering)

[Return Value]

• Bitmap bitmap : Bitmap object
• int horizontalStartPosition : X coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int verticalStartPosition : Y coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int width: The width to be printed
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int height: The length to be printed
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int level: The brightness of image to be printed
• boolean dithering: Whether to apply dithering (true: enable, false : disable)

[Note]

• Contents requested by this API will be printed when **2-2-11 print API** is called.

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void drawCompressImage (){
      mPrinter.drawCompressionImage(bitmap
                                    , 0
                                    , 10
                                    , 200
                                    , 200
                                    , 80
                                    , true)
      mPrinter.print(1,1);
   }
}
```

2-2-34 drawCompressionImage

Request image printing to the image buffer (applying image compression algorithm)

[Declaration]

• drawCompressionImage(String path, int horizontalStartPosition,
int verticalStartPosition, int width, int level, boolean dithering)

[Return Value]

• Bitmap bitmap : Image path
• int horizontalStartPosition : X coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int verticalStartPosition : Y coordinate at start position
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int width: The width to be printed
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int height: The length to be printed
(Reference: 1-3 Available range of X/Y coordinates for each model)
• int level: The brightness of image to be printed
• boolean dithering: Whether to apply dithering (true: enable, false : disable)

[Note]

• Contents requested by this API will be printed when **2-2-11 print API** is called.

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void drawCompressImage (){
      mPrinter.drawCompressionImage(path
                                    , 0
                                    , 10
                                    , 200
                                    , 200
                                    , 80
                                    ,true)
      mPrinter.print(1,1);
   }
}
```

2-2-35 getStatus

Check the status of the printer.

[Declaration]
• void getStatus(boolean checkImageBuffer)

[Return Value]
• boolean checkImageBuffer: Check whether the return value is 2 byte
(true : 2 bytes, false : 1 byte)

[Value]

Code	Value	Description
STATUS_NORMAL	0x00	Normal
STATUS_1ST_BYTE_PAPER_EMPTY	0x80	No paper
STATUS_1ST_BYTE_COVER_OPEN	0x40	Paper cover open
STATUS_1ST_BYTE_CUTTER_JAMMED	0x20	Cutter jam
STATUS_1ST_BYTE_TPH_OVERHEAT	0x10	Header overheat
STATUS_1ST_BYTE_AUTO_SENSING_FAILURE	0x08	Gap detection failure
STATUS_1ST_BYTE_RIBBON_END_ERROR	0x04	No ribbon
STATUS_2ND_BYTE_BUILDING_IN_IMAGE_BUFFER	0x00000080	Building the label in the image buffer
STATUS_2ND_BYTE_PRINTING_IN_IMAGE_BUFFER	0x00000040	Printing the label in the image buffer
STATUS_2ND_BYTE_PAUSED_IN_PEELER_UNIT	0x00000020	The printed label is stuck in the peeler

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …

   private void getStatus(){
      mPrinter.getStatus(true);
   }
   private final Handler mHandler = new Handler(){
      @Override
      public void handleMessage(Message msg){
         switch (msg.what){
         ...

         case Printer.MESSAGE_READ :
         switch (msg.arg1){
            case Printer.PROCESS_GET_STATUS:
            byte[] report = (byte[]) msg.obj;

            //1Byte Message
            if((report[0] & Printer.STATUS_1ST_BYTE_PAPER_EMPTY)
                        == Printer.STATUS_1ST_BYTE_PAPER_EMPTY){
                  Log.i("TAG", "Paper is Empty");
            }
             if((report[0] & Printer.STATUS_1ST_BYTE_COVER_OPEN)
                        == Printer.STATUS_1ST_BYTE_COVER_OPEN){
                  Log.i("Printer cover open");
            }

            ...

            //2Byte Message
            if(report.length == 2){
             if((report[1] &
                Printer.STATUS_2ND_BYTE_PAUSED_IN_PEELER_UNIT)==
                     Printer.STATUS_2ND_BYTE_PAUSED_IN_PEELER_UNIT){
                  Log.i("TAG", " Issued label is paused in peeler unit");
               }
               …
            }
            break;
            }
         break;
         }
      }
   }
}
```

2-2-36 setAutoCutter

Change AutoCutter settings for models with auto cutter.

[Declaration]
• void setAutoCutter(boolean enabled, int cuttingPeriod)

[Return Value]
• boolean enabled : whether to use AutoCutter (true : Enable false : Disable)
• int cuttingPeriod : set cutting interval

[Note]
• Set the cutting interval to 2 and then enter the 2-2-11 print parameter as 3, and when printing three copies, the printer cut after printing 2 sheets, and cut after printing the last one.

⚠ **Caution**	- Enable only when auto cutter is installed. - Always set to Disable if auto cutter is not installed.

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void cutterSetting(){
      mPrinter.setAutoCutter(true, 10);
   }
}
```

2-2-37 getPrinterInformation

Request information about the printer, such as model name and firmware version.

[Declaration]
• void getPrinterInformation (int param)

[Return Value]
• int param: Information to request to the printer

Code	Value	Description
PRINTER_INFORMATION_MODEL_NAME	0	The model name of the connected printer
PRINTER_INFORMATION_FIRMWARE_VERSION	2	The firmware version of the connected printer

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void getPrinterInformation(){
      mPrinter.getPrinterInformation(
                  Printer.PRINTER_INFORMATION_MODEL_NAME);
   }
}
```

2-2-38 printInformation

Print out the printer information.

[Declaration]
• void printInformation()

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void getPrinterInformation(){
      mPrinter.printInformation();
   }
}
```

2-2-39 initializePrinter

Initialize the printer setting.

[Declaration]
• void initializePrinter();

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void printerSetting (){
      mPrinter.initializePrinter();
   }
}
```

2-2-40 setOrientation

Set the printing orientation

[Declaration]
• void setOrientation(int orientation);

[Return Value]
• int orientation: Printing orientation

Code	Value	Description
ORIENTATION_TOP_TO_BOTTOM	84	Print from top to bottom
ORIENTATION_BOTTOM_TO_TOP	66	Print from bottom to top

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void printerSetting (){
      mPrinter.setOrientation(Printer.ORIENTATION_BOTTOM_TO_TOP);
   }
}
```

2-2-41 setCharacterSet

Set internationalCharacterSet and code page.

[Declaration]
• void setCharacterSet(int internationalCharacterSet, int codePage);

[Return Value]
• int internationalCharacterSet: The set of character

Code	Value	Description
INTERNATIONAL_CHARACTER_SET_USA	0	U.S.A
INTERNATIONAL_CHARACTER_SET_FRANCE	1	France
INTERNATIONAL_CHARACTER_SET_GERMANY	2	Germany
INTERNATIONAL_CHARACTER_SET_UK	3	U.K
INTERNATIONAL_CHARACTER_SET_DENMARK1	4	Denmark I
INTERNATIONAL_CHARACTER_SET_SWEDEN	5	Sweden
INTERNATIONAL_CHARACTER_SET_ITALY	6	Italy
INTERNATIONAL_CHARACTER_SET_SPAIN1	7	Spain I
INTERNATIONAL_CHARACTER_SET_NORWAY	8	Norway
INTERNATIONAL_CHARACTER_SET_DENMARK2	9	Denmark II
INTERNATIONAL_CHARACTER_SET_JAPAN	10	Japan
INTERNATIONAL_CHARACTER_SET_SPAIN2	11	Spain II
INTERNATIONAL_CHARACTER_SET_LATIN_AMERICA	12	Latin America
INTERNATIONAL_CHARACTER_SET_KOREA	13	Korea
INTERNATIONAL_CHARACTER_SET_SLOVENIA_CROATIA	14	Slovenia/ Croatia
INTERNATIONAL_CHARACTER_SET_CHINA	15	China

• int codePage : Code page

Code	Value	Description	
CODE_PAGE_CP437_USA	0	CP437	U.S.A
CODE_PAGE_CP850_LATIN1	1	CP850	Latin1
CODE_PAGE_CP852_LATIN2	2	CP 852	Latin2
CODE_PAGE_CP860_PORTUGUESE	3	CP 860	Portuguese
CODE_PAGE_CP863_CANADIAN_FRENCH	4	CP 863	Canadian French
CODE_PAGE_CP865_NORDIC	5	CP 865	Nordic
CODE_PAGE_WCP1252_LATIN1	6	WCP 1252	Latin I
CODE_PAGE_CP865_WCP1252_EUROPEAN_COMBINED	7	CP 865 + WCP 1252	European Combined
CODE_PAGE_CP857_TURKISH	8	CP 857	Turkish
CODE_PAGE_CP737_GREEK	9	CP 737	Greek
CODE_PAGE_WCP1250_LATIN2	10	WCP 1250	Latin 2
CODE_PAGE_WCP1253_GREEK	11	WCP 1253	Greek
CODE_PAGE_WCP1254_TURKISH	12	WCP 1254	Turkish
CODE_PAGE_CP855_CYRILLIC	13	CP 855	Cyrillic
CODE_PAGE_CP862_HEBREW	14	CP 862	Hebrew
CODE_PAGE_CP866_CYRILLIC	15	CP 866	Cyrillic
CODE_PAGE_WCP1251_CYRILLIC	16	WCP 1251	Cyrillic
CODE_PAGE_WCP1255_HEBREW	17	WCP 1255	Hebrew
CODE_PAGE_CP928_GREEK	18	CP 928	Greek
CODE_PAGE_CP864_ARABIC	19	CP 864	Arabic
CODE_PAGE_CP775_BALTIC	20	CP 775	Baltic
CODE_PAGE_WCP1257_BALTIC	21	WCP1257	Baltic
CODE_PAGE_CP858_LATIN1_EURO	22	CP858	Latin 1 + Euro

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void printerSetting (){
      mPrinter.setCharacterSet(
                        Printer.INTERNATIONAL_CHARACTER_SET_USA
                       , Printer.CODE_PAGE_CP437_USA);
   }
}
```

2-2-42 setPrintingType

Set the printing type.

[Declaration]
• void setPrintingType(int type)

[Return Value]
• int type: Printing type

Code	Value	Description
PRINTING_TYPE_DIRECT_THERMAL	100	Direct thermal
PRINTING_TYPE_THERMAL_TRANSFER	116	Thermal transfer

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void printerSetting (){
      mPrinter.setPrintingType (
            Printer.PRINTING_TYPE_DIRECT_THERMAL);
   }
}
```

2-2-43 setMargin

Set the margin of image buffer.

[Declaration]
• void setMargin(int horizontalMargin, int verticalMargin);

[Return Value]
• int horizontalMargin : Horizontal margin
• int verticalMargin : Vertical margin

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void printerSetting (){
      mPrinter.setMargin(10, 10);
   }
}
```

2-2-44 setBackFeedOption

Set whether to perform back-feed before print start.

[Declaration]
• void setBackFeedOption(boolean enabled, int quantity);

[Return Value]
• boolean enabled: Whether to enable back-feed (true: Enable, false: Disable)
• int quantity: Back-feed length (0 : Default)

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void printerSetting (){
      mPrinter.setBackFeedOption(true, 10);
   }
}
```

2-2-45 setBufferMode

Set the image buffer mode.

[Declaration]
• void setBufferMode(boolean doubleBuffering);

[Return Value]
• boolean doubleBuffering: Whether to enable double buffer (true: Enable, false: Disable)

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void printerSetting (){
      mPrinter.setBufferMode (false);
   }
}
```

2-2-46 clearBuffer

Clear the contents of the image buffer and prepare to create a new label.

[Declaration]
• void clearBuffer ();

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void printerSetting (){
      mPrinter.clearBuffer ();
   }
}
```

2-2-47 setLength

Set the printer's paper length, gap / black mark length, and paper type.

[Declaration]
• void setLength(int labelLength, int gapLength, int mediaType, int offsetLength);

[Return Value]
• int labelLength: Paper length
• int gapLength: Gap length, or black mark thickness
• int mediaType: Paper type

Code	Value	Description
MEDIA_TYPE_GAP	71	Gap
MEDIA_TYPE_CONTINUOUS	67	Continuous
MEDIA_TYPE_BLACK_MARK	66	Black mark

• int offsetLength: The gap between gap, or black mark and cutting line

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void printerSetting (){
      mPrinter.setLength (1216
                          , 1
                          , Printer.MEDIA_TYPE_GAP
                          , 1);
   }
}
```

2-2-48 setRewinder

Select whether to enable the rewinder.

[Declaration]
• void setRewinder(boolean enabled)

[Return Value]
• boolean enabled : whether to use Rewinder (true : Enable, false : Disable)

⚠ Caution	- Enable only in models with rewinder. - For models without rewinder, always set to false.

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void printerSetting (){
      mPrinter.setRewinder (false);
   }
}
```

2-2-49 setSpeed

Set the printing speed of printer.

[Declaration]
• void setSpeed(int speed);

[Return Value]
• int speed : Printer Speed

Code	Value	Description
SPEED_25IPS	0	Print 2.5 Inch per second
SPEED_30IPS	1	Print 3.0 Inch per second
SPEED_40IPS	2	Print 4.0 Inch per second
SPEED_50IPS	3	Print 5.0 Inch per second
SPEED_60IPS	4	Print 6.0 Inch per second
SPEED_70IPS	5	Print 7.0 Inch per second
SPEED_80IPS	6	Print 8.0 Inch per second

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void printerSetting (){
      mPrinter.setSpeed(Printer.SPEED_30IPS);
   }
}
```

2-2-50 setOffset

Save (set) offset length between black marks(or gap) and dotted lines

[Declaration]
• void setOffset(int offset);

[Return Value]
• int offset : offset (range : -100~100)

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void printerSetting (){
      mPrinter.setOffset (40);
   }
}
```

2-2-51 setDensity

Set print Density.

[Declaration]
• void setDensity(int density)

[Return Value]
• int density : Density level (range : 0~20)

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void printerSetting (){
      mPrinter.setDensity(10);
   }
}
```

2-2-52 setCutterPosition

Set the cutting position of label.

[Declaration]
• void setCutterPosition(int position)

[Return Value]
• int position : tear-off/cutter position (range : -100~100)

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void printerSetting (){
      mPrinter.setCutterPosition (0);
   }
}
```

2-2-53 setWidth

Set the image buffer width.

[Declaration]
• void setWidth(int labelWidth)

[Return Value]
• int labelWidth: Image buffer width

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void printerSetting (){
      mPrinter.setWidth (800);
   }
}
```

2-2-54 executeDirectIo

Executes the printing by commands to the printer.
Please refer to SLCS(Programming) Manual for command generation.

[Declaration]
• executeDirectIo(String command, boolean hasResponse, int responseLength)

[Return Value]
• String command: Instruction data generated by SLCS
• boolean hasResponse: Whether to have return value (true: Yes, false: No)
• int responseLength: The length of return value

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void printDirectIo (){
      mPrinter.executeDirectIo("CB\n" +
                               "SW800\n" +
                               "SM10,0\n" +
                               "BD100,300,300,500,O\n" +
                               "BD400,300,700,500,B,30\n" +
                               "P1", false, 0);
   }
}
```

2-2-55 executeDirectIo

Directly transfer the commands of the byte array to the printer.
Please refer to our SLCS (programming) manual for command generation.

[Declaration]
• executeDirectIo(byte[] command, boolean hasResponse, int responseLength)

[Return Value]
• byte[] command: Byte array of command generated by SLCS
• boolean hasResponse: Whether to have return value (true: Yes, false: No)
• int responseLength: The length of return value

[Example]

```
Public Class SampleClass{
   Printer mPrinter = new Printer(this, mHandler, null);
   …
   private void printDirectIo (){
      String =
      String command ="SS3\n" + // Set Speed to 5 ips
                              "SD20\n" +      // Set Density level to 20
                              "SW800\n" +   // Set Label Width 800
                              "SOT\n" +       // Set Printing Orientation from Top to Bottom
                              "T26,20,0,1,1,0,0,N,N,'Font - 6 pt'\n" +
                              "T26,49,1,1,1,0,0,N,N,'Font - 8 pt'\n" +
                              "T26,81,2,1,1,0,0,N,N,'Font - 10 pt'\n" +
                              "T26,117,3,1,1,0,0,N,N,'Font - 12 pt'\n" +
                              "T26,156,4,1,1,0,0,R,N,'Font - 15 pt'\n" +
                              "T26,200,5,1,1,0,0,N,N,'Font - 20 pt'\n" +
                              "T26,252,6,1,1,0,0,N,N,'Font - 30 pt'\n" +
                              "P1";

      mPrinter.executeDirectIo(command.getBytes(), false, 0);
   }
}
```

3. Constant Value

The constant values that are enabled in the provided SDK are defined in the "Printer.java" file.

3-1 Alignments

3-1-1 Device Font Alignment

It is the property that defines the alignment value of the device font.

Code	Value	Description
TEXT_ALIGNMENT_NONE	48	No Align
TEXT_ALIGNMENT_LEFT	70	Align to left
TEXT_ALIGNMENT_RIGHT	76	Align to right
TEXT_ALIGNMENT_RIGHT_TO_LEFT	82	Print characters from right to left

3-1-2 Vector Font Alignment

It is the property that defines the alignment value of the vector font.

Code	Value	Description
VECTOR_FONT_TEXT_ALIGNMENT_LEFT	76	Align to left
VECTOR_FONT_TEXT_ALIGNMENT_RIGHT	82	Align to right
VECTOR_FONT_TEXT_ALIGNMENT_CENTER	67	Align to center
VECTOR_FONT_TEXT_DIRECTION_LEFT_TO_RIGHT	0	Print characters from left to right
VECTOR_FONT_TEXT_DIRECTION_RIGHT_TO_LEET	1	Print characters from right to left

3-2 Barcode HRI

• The below constants are used to specify the position and font of HRI when printing barcodes that supports HRI.

Code	Value	Description
HRI_NOT_PRINTED	0	HRI is not used.
HRI_BELOW_FONT_SIZE_1	1	Position of HRI: Below barcode Font Size: 1
HRI_ABOVE_FONT_SIZE_1	2	Position of HRI: Above barcode Font Size: 1
HRI_BELOW_FONT_SIZE_2	3	Position of HRI: Below barcode Font Size: 2
HRI_ABOVE_FONT_SIZE_2	4	Position of HRI: Above barcode Font Size: 2
HRI_BELOW_FONT_SIZE_3	5	Position of HRI: Below barcode Font Size: 3
HRI_ABOVE_FONT_SIZE_3	6	Position of HRI: Above barcode Font Size: 3
HRI_BELOW_FONT_SIZE_4	7	Position of HRI: Below barcode Font Size: 4
HRI_ABOVE_FONT_SIZE_5	8	Position of HRI: Above barcode Font Size: 4

3-3 MaxiCode Modes

• MaxiCode Mode constants are used to set the barcode option when printing Maxi code barcode.

Code	Value	Description
MAXICODE_MODE_0	0	MaxiCode Mode 0
MAXICODE_MODE_2	2	MaxiCode Mode 2
MAXICODE_MODE_3	3	MaxiCode Mode 3
MAXICODE_MODE_4	4	MaxiCode Mode 4

3-4 1D Barcode Types

• One-dimensional Barcode Types constants are used to set the barcode option when printing 1D barcode.

Code	Value	Description
BARCODE_CODE39	0	Code39
BARCODE_CODE128	1	Code128
BARCODE_I2OF5	2	Interleaved 2of5
BARCODE_CODABAR	3	Codabar
BARCODE_CODE93	4	Code93
BARCODE_UPC_A	5	UPC-A
BARCODE_UPC_E	6	UPC-E
BARCODE_EAN13	7	EAN13
BARCODE_EAN8	8	EAN8
BARCODE_UCC_EAN128	9	UCC/EAN128
BARCODE_CODE11	10	Code11
BARCODE_PLANET	11	Planet
BARCODE_INDUSTRIAL_2OF5	12	Industrial 2of5
BARCODE_STANDARD_2OF5	13	Standard 2of5
BARCODE_LOGMARS	14	Logmars
BARCODE_UPC_EAN_EXTENSIONS	15	UPC/EAN Extensions
BARCODE_POSTNET	16	Postnet

3-5 Barcode Origin Point

• Barcode Origin Point constants are used to set the reference origin position of barcode.

Code	Value	Description
BARCODE_ORIGIN_POINT_CENTER	0	Set the reference point of the barcode to the center.
BARCODE_ORIGIN_POINT_UPPER_LEFT	1	Set the reference point of the barcode in the upper left corner.

3-6 Error Correction Level

• Error Correction Level constants are used to set the level of error correction for possible corruption of barcode.

Code	Value	Description
ECC_LEVEL_7	76	Error Correction Level 7
ECC_LEVEL_15	77	Error Correction Level 15
ECC_LEVEL_25	82	Error Correction Level 25
ECC_LEVEL_30	72	Error Correction Level 30

3-7 Data Compression Method

• Data Compression Method constants are used to specify the data compression property.

Code	Value	Description
DATA_COMPRESSION_TEXT	0	2char / codeword
DATA_COMPRESSION_NUMERIC	1	2.93 char / codeword
DATA_COMPRESSION_BINARY	2	1.2bytes / codeword

3-8 QRCode Model

• QR Code Model constants are used to set the options in printing QR barcode.

Code	Value	Description
QR_CODE_MODEL1	1	QR Model 1
QR_CODE_MODEL2	2	QR Model 2

3-9 Code 49 Starting Mode

• Code 49 Starting Mode constants are used to set the properties of Starting Mode in printing Code 49 barcode.

Code	Value	Description
CODE49_STRING_MODE_REGULAR_ALPHANUMERIC	0	Regular Alphanumeric Mode
CODE49_STRING_MODE_MULTIPLE_READ_ALPHANUMERIC	1	Multiple Read Alphanumeric
CODE49_STRING_MODE_REGULAR_NUMERIC	2	Regular Numeric Mode
CODE49_STRING_MODE_GROUP_ALPHANUMERIC	3	Group Alphanumeric Mode
CODE49_STRING_MODE_REGULAR_ALPHANUMERIC_SHIFT1	4	Regular Alphanumeric Shift 1
CODE49_STRING_MODE_REGULAR_ALPHANUMERIC_SHIFT2	5	Regular Alphanumeric Shift 2
CODE49_STRING_MODE_AUTOMATIC_MODE	7	Automatic Mode

3-10 Codablock Mode

• Codablock Mode constants are used to set the options when printing Codablock barcode.

Code	Value	Description
CODABLOCK_MODE_A	'A'	Code 39 character set is used.
CODABLOCK_MODE_E	'E'	Code 128 character set is used.
CODABLOCK_MODE_F	'F'	Code 128 character set is used. Function 1 is added automatically.

3-11 Check Digit Option

• These constants are used to set the Check Digit property when printing MSI barcode.

Code	Value	Description
MSI_BARCODE_CHECKDIGIT_NONE	0	No Check Digit
MSI_BARCODE_CHECKDIGIT_1MOD10	1	Check Digit 1 Mod 10
MSI_BARCODE_CHECKDIGIT_2MOD10	2	Check Digit 2 Mod 10
MSI_BARCODE_CHECKDIGIT_1MOD11_AND_1MOD_10	3	Check Digit 1 Mod 10

3-12 RSS Barcode Type

• RSS Barcode Type constants are used to set the barcode type when printing RSS barcode.

Code	Value	Description
BARCODE_TYPE_RSS14	0	RSS14
BARCODE_TYPE_RSS14_TRUNCATED	1	RSS14 truncated
BARCODE_TYPE_RSS14_STACKED	2	RSS14 stacked
BARCODE_TYPE_RSS14_STACKED_OMNIDIRECTIONAL	3	RSS14 Stacked omnidirectional
BARCODE_TYPE_RSS_LIMITIED	4	RSS limited
BARCODE_TYPE_RSS_EXPANDED	5	RSS Expanded
BARCODE_TYPE_RSS_UPCA	6	RSS UPC A
BARCODE_TYPE_RSS_UPCE	7	RSS UPC E
BARCODE_TYPE_RSS_EAN13	8	EAN13
BARCODE_TYPE_RSS_EAN8	9	EAN 8
BARCODE_TYPE_RSS_UCC_EAN128_CCAB	10	EAN128 CC-A/B
BARCODE_TYPE_RSS_UCC_EAN128_CCC	11	EAN128 CC-C

3-13 Rotation Degrees

• Rotation Degrees constants are used to set the rotation property of the printing.

Code	Value	Description
ROTATION_NONE	0	No rotation
ROTATION_90_DEGREES	1	90 degrees of rotation
ROTATION_180_DEGREES	2	180 degrees of rotation
ROTATION_270_DEGREES	3	270 degrees of rotation.

3-14 Device Fonts

• Device Fonts constants are used to set the property of Device Font.

Code	Value	Description
FONT_SIZE_6	48	9 X 15 (dots)
FONT_SIZE_8	49	12 X 20 (dots)
FONT_SIZE_10	50	16 X 25 (dots)
FONT_SIZE_12	51	19 X 30 (dots)
FONT_SIZE_15	52	24 X 38 (dots)
FONT_SIZE_20	53	32 X 40 (dots)
FONT_SIZE_30	54	48 X 76 (dots)
FONT_SIZE_14	55	22 X 34 (dots)
FONT_SIZE_18	56	28 X 44 (dots)
FONT_SIZE_24	57	37 X 58 (dots)
FONT_SIZE_KOREAN1	97	16 X 16 (dots) (ASCII 9 X 15)
FONT_SIZE_KOREAN2	98	24 X 24 (dots) (ASCII 12 X 24)
FONT_SIZE_KOREAN3	99	20 X 20 (dots) (ASCII 12 X 20)
FONT_SIZE_KOREAN4	100	26 X 26 (dots) (ASCII 16 X 30)
FONT_SIZE_KOREAN5	101	20 X 26 (dots) (ASCII 16 X 30)
FONT_SIZE_KOREAN6	102	38 X 38 (dots) (ASCII 22 X 34)
FONT_SIZE_GB2312	109	24 X 24 (dots) (ASCII 12 X 24)
FONT_SIZE_BIG5	110	24 X 24 (dots) (ASCII 12 X 24)
FONT_SIZE_SHIFT_JIS	106	24 X 24 (dots) (ASCII 12 X 24)

3-15 Vector Fonts

• Vector Fonts constants are used to set the property of Vector Fonts.

Code	Value	Description
VECTOR_FONT_ASCII	85	ASCII (1Byte code)
VECTOR_FONT_KS5601	75	KS5601 (2Byte code)
VECTOR_FONT_BIG5	66	BIG5 (2Byte code)
VECTOR_FONT_GB2312	71	GB2312 (2Byte code)
VECTOR_FONT_SHIFT_JIS	74	Shift-JIS (2Byte code)
VECTOR_FONT_OCR_A	97	OCR-A (1Byte code)
VECTOR_FONT_OCR_B	98	OCR-B (1Byte code)

3-16 Draw Block Options

• Draw Block Options constants are used to set the Draw Block Options.

Code	Value	Description
BLOCK_OPTION_LINE_OVERWRITING	79	Line Overwriting
BLOCK_OPTION_LINE_EXCLUSIVE_OR	69	Line Exclusive OR
BLOCK_OPTION_LINE_DELETE	68	Line Delete
BLOCK_OPTION_SLOPE	83	Slope(a oblique line)
BLOCK_OPTION_BOX	66	Box

3-17 Draw Circle Sizes

• Draw Circle Sizes constants are used to set the property related to the size when using the Draw Circle Method.

Code	Value	Description	
CIRCLE_SIZE_DIAMETER5	1	40 × 40 (dot)	5 mm
CIRCLE_SIZE_DIAMETER7	2	56 × 56 (dot)	7 mm
CIRCLE_SIZE_DIAMETER9	3	72 × 72 (dot)	9 mm
CIRCLE_SIZE_DIAMETER11	4	88 × 88 (dot)	11 mm
CIRCLE_SIZE_DIAMETER13	5	104 × 104 (dot)	13 mm
CIRCLE_SIZE_DIAMETER21	6	168 × 168 (dot)	21 mm

3-18 International Character Set

• These constants are used to set the International Character Set.

Code	Value	Description
INTERNATIONAL_CHARACTER_SET_USA	0	U.S.A
INTERNATIONAL_CHARACTER_SET_FRANCE	1	France
INTERNATIONAL_CHARACTER_SET_GERMANY	2	Germany
INTERNATIONAL_CHARACTER_SET_UK	3	U.K
INTERNATIONAL_CHARACTER_SET_DENMARK1	4	Denmark I
INTERNATIONAL_CHARACTER_SET_SWEDEN	5	Sweden
INTERNATIONAL_CHARACTER_SET_ITALY	6	Italy
INTERNATIONAL_CHARACTER_SET_SPAIN1	7	Spain I
INTERNATIONAL_CHARACTER_SET_NORWAY	8	Norway
INTERNATIONAL_CHARACTER_SET_DENMARK2	9	Denmark II
INTERNATIONAL_CHARACTER_SET_JAPAN	10	Japan
INTERNATIONAL_CHARACTER_SET_SPAIN2	11	Spain II
INTERNATIONAL_CHARACTER_SET_LATIN_AMERICA	12	Latin America
INTERNATIONAL_CHARACTER_SET_KOREA	13	Korea
INTERNATIONAL_CHARACTER_SET_SLOVENIA_CROATIA	14	Slovenia/Croatia
INTERNATIONAL_CHARACTER_SET_CHINA	15	China

3-19 Code Pages

• These constants are used to set the Code Page.

Code	Value	Description	
CODE_PAGE_CP437_USA	0	CP437	U.S.A
CODE_PAGE_CP850_LATIN1	1	CP850	Latin1
CODE_PAGE_CP852_LATIN2	2	CP 852	Latin2
CODE_PAGE_CP860_PORTUGUESE	3	CP 860	Portuguese
CODE_PAGE_CP863_CANADIAN_FRENCH	4	CP 863	Canadian French
CODE_PAGE_CP865_NORDIC	5	CP 865	Nordic
CODE_PAGE_WCP1252_LATIN1	6	WCP 1252	Latin I
CODE_PAGE_CP865_WCP1252_EUROPEAN_COMBINED	7	CP 865 + WCP 1252	European Combined
CODE_PAGE_CP857_TURKISH	8	CP 857	Turkish
CODE_PAGE_CP737_GREEK	9	CP 737	Greek
CODE_PAGE_WCP1250_LATIN2	10	WCP 1250	Latin 2
CODE_PAGE_WCP1253_GREEK	11	WCP 1253	Greek
CODE_PAGE_WCP1254_TURKISH	12	WCP 1254	Turkish
CODE_PAGE_CP855_CYRILLIC	13	CP 855	Cyrillic
CODE_PAGE_CP862_HEBREW	14	CP 862	Hebrew
CODE_PAGE_CP866_CYRILLIC	15	CP 866	Cyrillic
CODE_PAGE_WCP1251_CYRILLIC	16	WCP 1251	Cyrillic
CODE_PAGE_WCP1255_HEBREW	17	WCP 1255	Hebrew
CODE_PAGE_CP928_GREEK	18	CP 928	Greek
CODE_PAGE_CP864_ARABIC	19	CP 864	Arabic
CODE_PAGE_CP775_BALTIC	20	CP 775	Baltic
CODE_PAGE_WCP1257_BALTIC	21	WCP1257	Baltic
CODE_PAGE_CP858_LATIN1_EURO	22	CP858	Latin 1 + Euro

3-20 Printing Type

• These constants are used to set the Printing type.

Code	Value	Description
PRINTING_TYPE_DIRECT_THERMAL	100	Direct thermal
PRINTING_TYPE_THERMAL_TRANSFER	116	Thermal transfer

3-21 Media Type

• These constants are used to set the print media type.

Code	Value	Description
MEDIA_TYPE_GAP	71	Gap
MEDIA_TYPE_CONTINUOUS	67	Continuous
MEDIA_TYPE_BLACK_MARK	66	Black mark

3-22 Speed Value

• These constants are used to set the print speed.

Code	Value	Description
SPEED_25IPS	0	Print 2.5 Inch per second
SPEED_30IPS	1	Print 3.0 Inch per second
SPEED_40IPS	2	Print 4.0 Inch per second
SPEED_50IPS	3	Print 5.0 Inch per second
SPEED_60IPS	4	Print 6.0 Inch per second
SPEED_70IPS	5	Print 7.0 Inch per second
SPEED_80IPS	6	Print 8.0 Inch per second

3-23 Orientation

• These constants are used to set the printing direction.

Code	Value	Description
ORIENTATION_TOP_TO_BOTTOM	84	Print from top to bottom
ORIENTATION_BOTTOM_TO_TOP	66	Print from bottom to top

3-24 Printer Status

• These constants are used to check the printer error status.

Code	Value	Description
STATUS_NORMAL	0x00	Normal
STATUS_1ST_BYTE_PAPER_EMPTY	0x80	Paper empty
STATUS_1ST_BYTE_COVER_OPEN	0x40	Cover open
STATUS_1ST_BYTE_CUTTER_JAMMED	0x20	Cutter jammed
STATUS_1ST_BYTE_TPH_OVERHEAT	0x10	Thermal head(TPH) overheat
STATUS_1ST_BYTE_AUTO_SENSING_FAILURE	0x08	Gap detection error (Auto-sensing failure)
STATUS_1ST_BYTE_RIBBON_END_ERROR	0x04	Ribbon end
STATUS_2ND_BYTE_BUILDING_IN_IMAGE_BUFFER	0x00000080	Building a label in the image buffer
STATUS_2ND_BYTE_PRINTING_IN_IMAGE_BUFFER	0x00000040	Printing a label in the image buffer
STATUS_2ND_BYTE_PAUSED_IN_PEELER_UNIT	0x00000020	The printed label is stuck in peeler

3-25 Printer Information

• It is the property on printer information.

Code	Value	Description
PRINTER_INFORMATION_MODEL_NAME	0	Model name of the connected printer
PRINTER_INFORMATION_FIRMWARE_VERSION	2	The firmware version of the connected printer

3-26 MicroPDF417 Mode List

• It is the property on MicroPDF417 barcode mode.

Mode	Number of Data Columns	Number of Data Rows	% of Cws for EC	Max Alpha Characters	Max Digits
0	1	11	64	6	8
1	1	14	50	12	17
2	1	17	41	18	26
3	1	20	40	22	32
4	1	24	33	30	44
5	1	28	29	38	55
6	2	8	50	14	20
7	2	11	41	24	35
8	2	14	32	36	52
9	2	17	29	46	67
10	2	20	28	56	82
11	2	23	28	64	93
12	2	26	29	72	105
13	3	6	67	10	14
14	3	8	58	18	26
15	3	10	53	26	38
16	3	12	50	34	49
17	3	15	47	46	67
18	3	20	43	66	96
19	3	26	41	90	132
20	3	32	40	114	167
21	3	38	39	138	202
22	3	44	38	162	237
23	4	6	50	22	32
24	4	8	44	34	49
25	4	10	40	46	67
26	4	12	38	58	85
27	4	15	35	76	111
28	4	20	33	106	155
29	4	26	31	142	208
30	4	32	30	178	261
31	4	38	29	214	313
32	4	44	28	250	366
33	4	4	50	14	20

3-27 PDF417 Barcode HRI

• Property for HRI of PDF 417 barcode.

Code	Value	Description
PDF417_HRI_NOT_PRINTED	0	Not printed
PDF417_HRI_BELOW_BARCODE	1	Below the bar code

3-28 CODE49 Barcode HRI

• Property for HRI of CODE49barcode.

Code	Value	Description
CODE49_HRI_NOT_PRINTED	0	Not printed
CODE49_HRI_BELOW_BARCODE	1	Below the bar code
CODE49_HRI_ABOVE_BARCODE	2	Above the bar code

3-29 PLESSEY Barcode HRI

• Property for HRI of PLESSEY barcode.

Code	Value	Description
PLESSEY_BARCODE_HRI_NOT_PRINTED	0	Not printed
PLESSEY_BARCODE_HRI_BELOW_BARCODE	1	Below the bar code
PLESSEY_BARCODE_HRI_ABOVE_BARCODE	2	Above the bar code

3-30 MSI Barcode HRI

• Property for HRI of MSI barcode.

Code	Value	Description
MSI_BARCODE_HRI_NOT_PRINTED	0	Not printed
MSI_BARCODE_HRI_BELOW_BARCODE	1	Below the bar code
MSI_BARCODE_HRI_ABOVE_BARCODE	2	Above the bar code

4. Appendix

4-1 Development environment settings

4-1-1 Setting manifest authority

• Bluetooth authority

- BLUETOOTH: Bluetooth communication authority such as connection request, connection acceptance, and data transmission
- BLUETOOTH_ADMIN: Authority of device search start and Bluetooth setting

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

• Wi-Fi authority

- ACCESS_WIFI_STATE: Authority to check Wi-Fi connection
- CHANGE_WIFI_STATE: Authority to check Wi-Fi status change
- CHANGE_WIFI_MULTICAST_STATE: Authority for enabling Wi-Fi multicast mode

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_MULTICAST_STATE" />
```

• Internet authority

- INTERNET: Authority for opening network socket
- ACCESS_NETWORK_STATE: Authority of access to network information

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

• Storage authority

- WRITE_EXTERNAL_STORAGE: Authority of enabling external storage

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

• USB authority

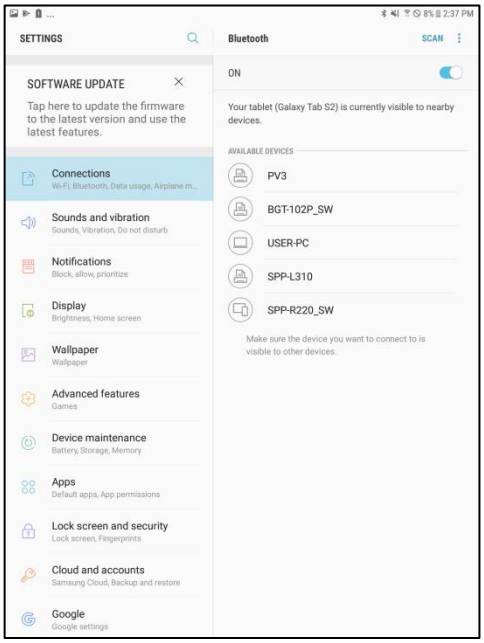
- ACTION_USB_DEVICE_ATTACHED: Authority of connecting USB

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
    <action android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED" />
</intent-filter>
<meta-data
    android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED"
    android:resource="@xml/device_filter" />
```

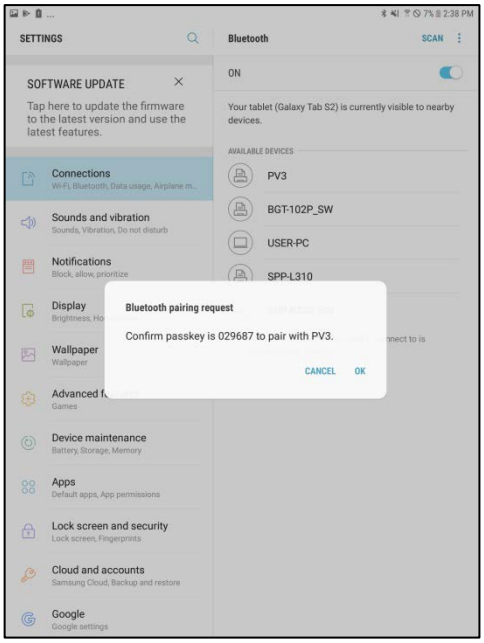
4-1-2 Connecting Android Devices

The following screenshot was taken from an Android 4.2 smartphone. Screenshots and titles may be different depending on the Android operating system or device.

1) Bluetooth
   1. Select Settings.
   2. The Bluetooth on the Android device and the printer should be turned on.
   3. Select Bluetooth for setting.

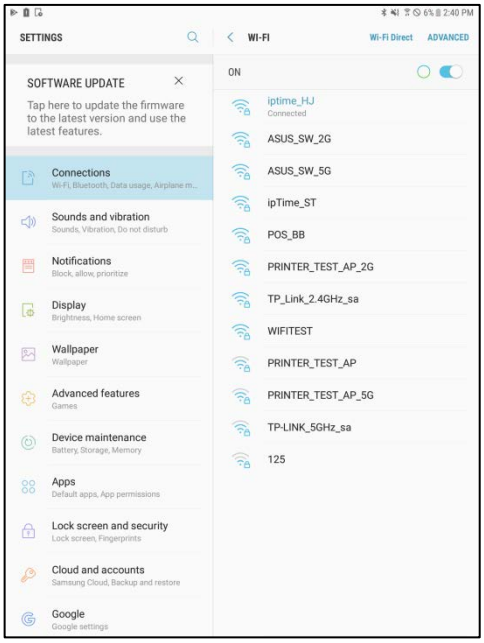


   4. Select Scan.
   5. Select and pair the printer to be connected.
   6. Enter your PIN code. Initial PIN code is "0000".



2) Wi-Fi

1. Connect the printer to a network AP (Access Point) and assign an IP address or set it to DHCP. As printer is initially set to ad-hoc, it should be set up once with the Net Configuration Tool included on the master CD.
2. Select Settings.
3. Wi-Fi on the Android device and the printer should be turned on.
4. Connect to the network to which the printer is connected.

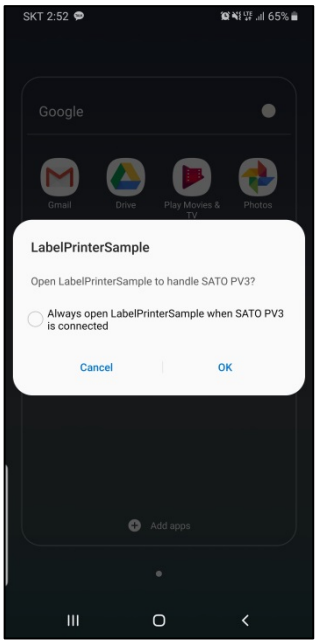


5. You do not need an additional setup to connect your Android device to the printer's TCP / IP port.

3) USB

1. In USB Android devices, OS version 3.1 or later can connect USB peripherals.
2. Android device does not need to have a specific driver or printer software installed.
3. The required USB cable may be different depending on your smartphone or tablet.

   Most Android devices are provided without an A to B USB cable. Mini / Micro USB cable or adapter / dock may be needed. Make sure you enable the right cable for the Android device you want to enable.
4. When connecting the printer for the first time, the following message may appear depending on the Android device.



5. To connect a USB peripheral, enter the following code into AndroidManifest.xml and res / xml / device_filter.xml.

[AndroidManifest.xml]

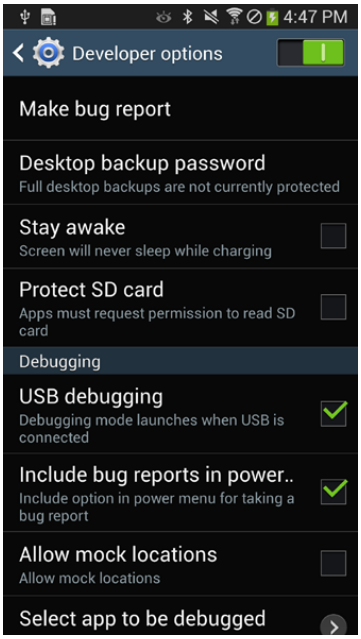
```
<intent-filter>
    <action
android:name=”android.hardware.usb.action.USB_DEVICE_ATTACHED”/>
</intent-filter>
<meta-data
    android:name=”android.hardware.usb.action.USB_DEVICE_ATTACHED”
    android:resource=”@xml/device_filter” />
```

[device_filter.xml]

```
<?xml version=”1.0” encoding=”utf-8”>
<resources>
    <usb-device
        class=”7”
        protocol=”2”
        subclass=”1”
        vendor-id=”5380” />
</resources>
```

4-1-3 Setting Android device developer options

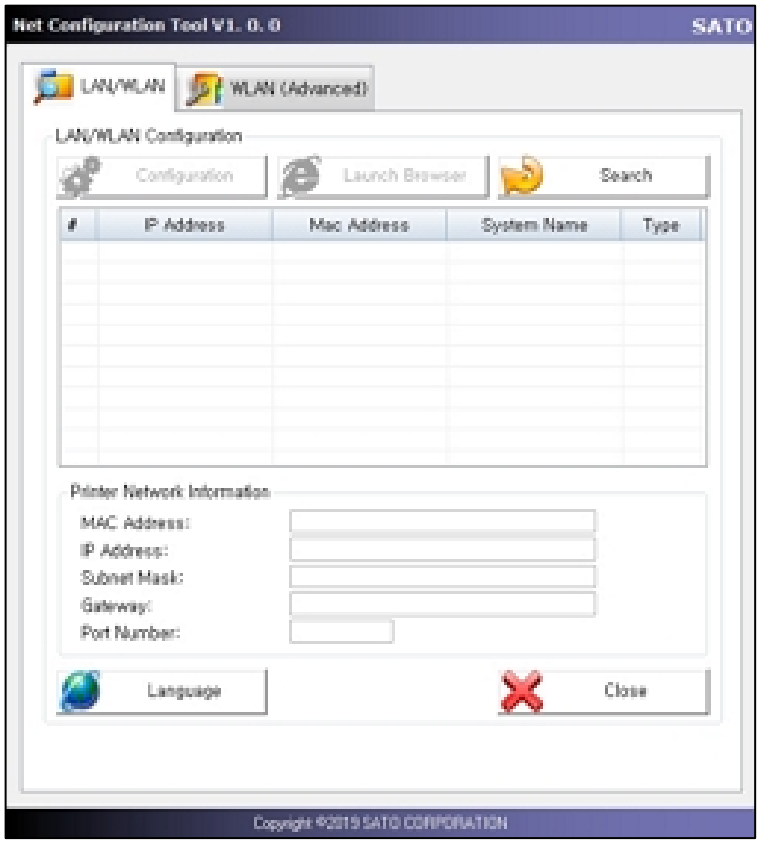
1. Select Settings.
2. Select Developer options.
3. Enable USB debugging



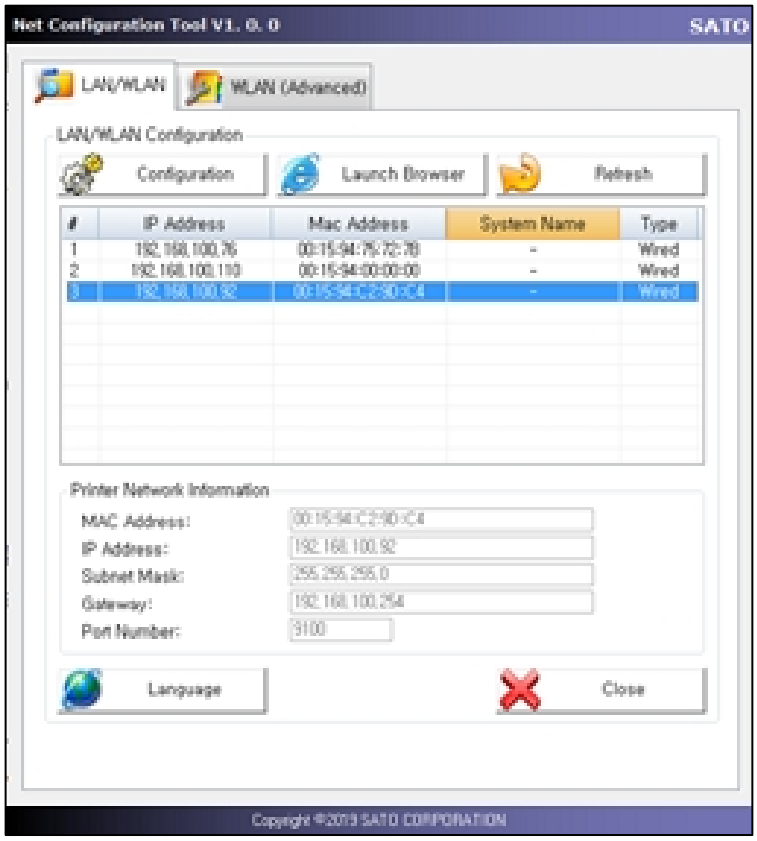
4. If you cannot find Developer options
   1) Select Settings.
   2) Select About device.
   3) Select Software info.
   4) Activate Developer options by hitting Build number.

4-1-4 Net Configuration Tool Enable

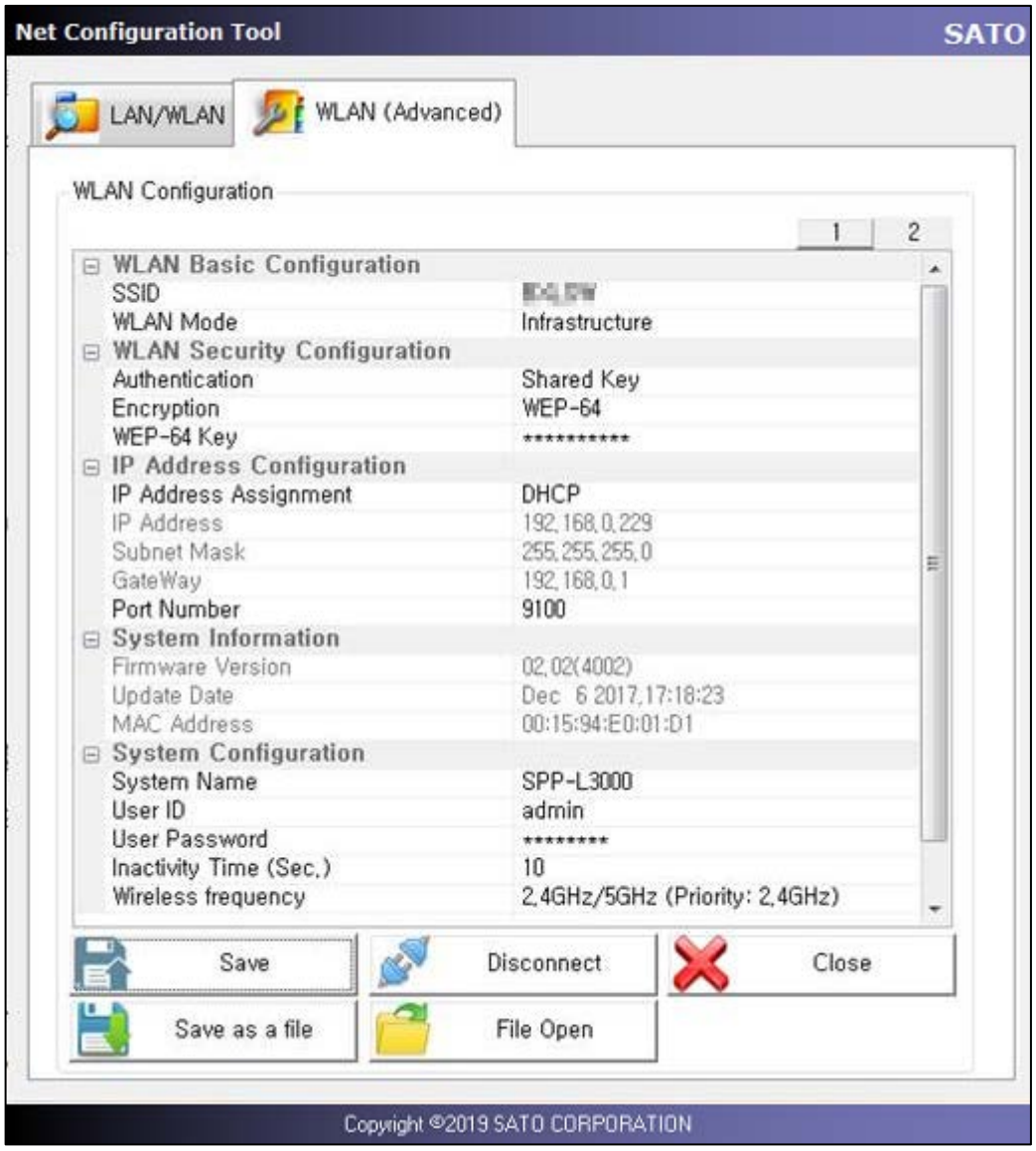
1. Run the downloaded Configuration Tool.



2. Click Search tab to search for the currently connected printer to the AP.



4. If the desired printer is not detected, move to the WLAN (Advanced) tab to connect the device.



5. Enter the network settings to connect the printer.

6. When setting is finished, click Save and restart the printer.

7. Enter the WLAN of the printer's interface and check the IP address.

Caution

Some semiconductor devices are easily damaged by static electricity. You should turn the printer “OFF”, before you connect or remove the cables on the rear side, in order to guard the printer against the static electricity. If the printer is damaged by the static electricity, you should turn the printer “OFF”.

Revision history

Ver.	Date	Page	Description
1.00	27.09.19	-	New