

TSPL SDK v1.0.01 Interface documentation

Document modification records

i. Instructions for Use

1.1 Engineering Configuration Description

1.2 obfuscation configuration

1.3 Initialization

II. Sample Demonstration

2.1 Connect Printer Process

2.1.1 Bluetooth Search Printer

2.1.2 Connecting the printer

III. Interface Description

PrinterHelper introduced

3.1.2 Bluetooth connection

3.1.3 WIFI connection

3.1.4 USB connection

3.1.5 Connection status

3.1.7 Disconnection

3.2.1 Instantiation Print Class

3.2.2 Page Label Start Command

3.2.3 Page tab end command (except line mode)

3.2.4 Print Self-Test Page

3.2.5 Set spacing for label paper

3.2.6 Clear print buffer contents

3.2.7 Text Printing

3.2.8 Print Direction

3.2.9 Barcode

3.2.10 Print QR Code

3.2.11 Print Lines

3.2.12 Print Picture

- 3.2.13 Paper cutting
- 3.2.14 Print Rectangular Box
- 3.2.15 Send data
- 3.2.16 Read Data Function
- 3.2.17 Bold Text
- 3.2.18 Get Printer Status
- 3.2.19 Set black mark position
- 3.2.20 Text paragraph printing
- 3.2.21 Print PDF417
- 3.2.22 Set printer density
- 3.2.23 Set printer speed
- 3.2.24 Set Inverse Box
- 3.2.25 Set CodePage
- 4.1.1 Enable SDK to send data logs

Document modification records

serial Number	version Number	modify content	modifier	review	date modified
01	v1.0.0	document creation	邱贤	林德春	2025-05-13
02	v1.0.01	Added setting code	邱贤	林德春	2025-11-17

i. Instructions for Use

1.1 Engineering Configuration Description

1. This SDK interface needs to run on Android system

2. Put the lzo-sdk-v x.x.x.aar android-common-sdk-vx.x.x.aar into the project libs directory, and add build.gradle to introduce aar

```
▼ build.gradle Java |
1  implementation(mapOf("name" to "lzo-sdk-v1.00.00", "ext" to "aar"))
2  implementation(mapOf("name" to "android-common-sdk-v1.00.00", "ext" to "aar"))
```

3. Bluetooth permission configuration

```
▼ AndroidManifest.xml XML |
1  <uses-permission android:name="android.permission.BLUETOOTH" />
2  <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />

3  <uses-permission android:name="android.permission.BLUETOOTH_CONNECT"/>
4  <uses-permission android:name="android.permission.BLUETOOTH_SCAN"/>
5  <uses-permission android:name="android.permission.BLUETOOTH_ADVERTISE"/>
>
```

```
▼ Dynamically configure location permissions in code Kotlin |
1  disposable = RxPermissions(this).request(
2      Manifest.permission.BLUETOOTH,
3      Manifest.permission.ACCESS_FINE_LOCATION,
4      Manifest.permission.ACCESS_COARSE_LOCATION
5  )
6  .subscribe {
7      if (it) {
8      }}
```

1.2 obfuscation configuration

1. Need to configure the code in the proguard-rules.pro to prevent SDK from being confused

```
▼ Do not confuse the following classes Kotlin |
1  -keep class com.common.sdk.** { *; }
2  -keepnames class com.common.sdk.**
3  -keeppackageNames com.common.sdk.**
```

1.3 Initialization

1. Call the following code in the Application.

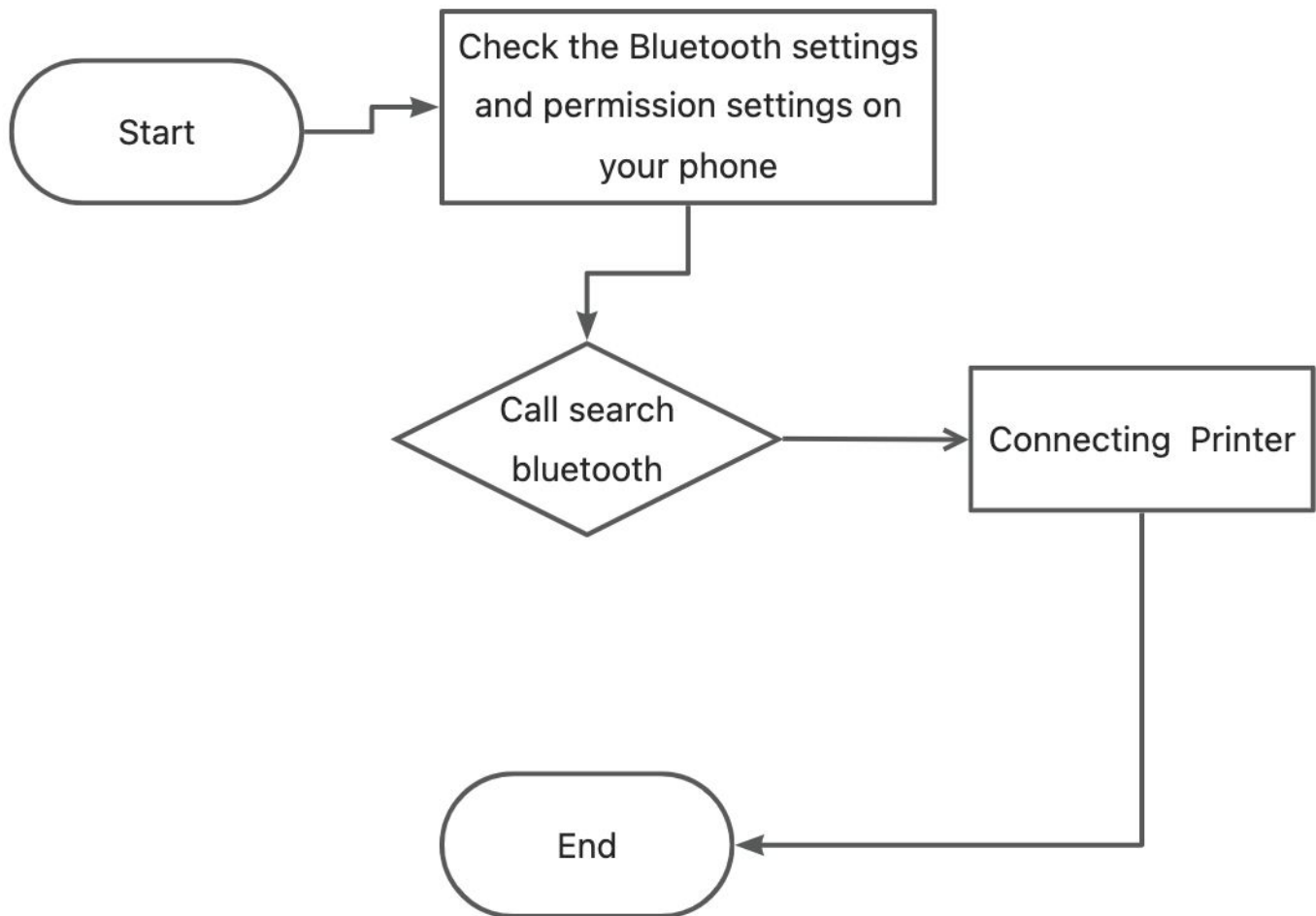
▼ initialization

Kotlin |

```
1 PrinterHelper.init(this);
```

II. Sample Demonstration

2.1 Connect Printer Process



2.1.1 Bluetooth Search Printer

▼ Registering Bluetooth Broadcaster

Java

```
1 private void registerBroadcast() {
2     IntentFilter intent = new IntentFilter();
3     intent.addAction(BluetoothDevice.ACTION_FOUND);
4     intent.addAction(BluetoothDevice.ACTION_BOND_STATE_CHANGED);
5     intent.addAction(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
6     context.registerReceiver(mReceiver, intent);
7 }
8
```

▼ Search Bluetooth

Java

```
1 RxPermissions rxPermissions = new RxPermissions((AppCompatActivity) context);
2 rxPermissions.request(Manifest.permission.BLUETOOTH_ADMIN,
3     Manifest.permission.BLUETOOTH,
4     Manifest.permission.ACCESS_FINE_LOCATION)
5 .subscribe(aBoolean -> {
6     if (aBoolean) {
7         if (null == mBluetoothAdapter) {
8             mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
9         }
10        if (!mBluetoothAdapter.isEnabled()) {
11            mBluetoothAdapter.enable();
12        }
13        if (mBluetoothAdapter.isDiscovering()) {
14            mBluetoothAdapter.cancelDiscovery();
15        }
16        mBluetoothAdapter.startDiscovery();//Turn on Search
17    } else {
18        Utility.show(context, "no bluetooth permission");
19    }
20 });
```

```
1 private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
2     @SuppressWarnings("MissingPermission")
3     @Override
4     public void onReceive(Context context, Intent intent) {
5         String action = intent.getAction();
6         if (BluetoothDevice.ACTION_FOUND.equals(action)) {
7             BluetoothDevice device =
8                 intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
9             if (device.getBluetoothClass().getMajorDeviceClass() == 1536)
10            {
11                //classic Bluetooth protocol
12            }
13            break;
14        }
15    };
```

2.1.2 Connecting the printer

```
1 val connection = BTConnection()
2 val resultCode = connection.connectBT(mac!!)
3 //resultCode:
4 //0: connection successful,
5 //-1: connection failed (timeout),
6 //-2: connection failed (parameter error),
7 //-3: connection failed (uninitialized))
```

III. Interface Description

PrinterHelper introduced

initialization PrinterHelper

```
1 public class App extends Application {
2     @Override
3     public void onCreate() {
4         super.onCreate();
5         PrinterHelper.init(this);
6     }
7 }
```

3.1.2 Bluetooth connection

- Pay attention to permission configuration

```
1 Bluetooth connection requires configuration permissions,
2 add in the AndroidManifest.xml file
3 <uses-permission android:name="android.permission.BLUETOOTH" />
4 <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
5 <uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
6 <uses-permission android:name="android.permission.BLUETOOTH_SCAN" />
7 <uses-permission android:name="android.permission.BLUETOOTH_ADVERTISE" />
8
9 Then dynamically obtain permissions in the code
10 RxPermissions rxPermissions = new RxPermissions(this);
11 rxPermissions.request(Manifest.permission.BLUETOOTH_ADMIN,
12                     Manifest.permission.BLUETOOTH,
13                     Manifest.permission.BLUETOOTH_CONNECT,
14                     Manifest.permission.BLUETOOTH_SCAN,
15                     Manifest.permission.ACCESS_FINE_LOCATION)
16 .subscribe(aBoolean -> {
17     if (aBoolean) {
18         //Permissions obtained successfully
19     }
20 });
```

- Description

connect printer via Bluetooth mac address

Bluetooth connect

Kotlin

```
1 fun connectBT(mac: String): Int
```

- Parameters

parameters	description
mac	bluetooth address (uppercase)

- Examples

Bluetooth connection example

Java

```
1 val connection = BTConnection()
2 val resultCode = connection.connectBT(mac!!)
3 //resultCode:
4 //0: connection successful,
5 //-1: connection failed (timeout),
6 //-2: connection failed (parameter error),
7 //-3: connection failed (uninitialized))
```

3.1.3 WIFI connection

- Note Configuration Permissions

▼ wifi permission configuration

Java

```

1  Wi-Fi connection requires configuration permissions,
2  add in AndroidManifest.xml file
3  <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
4  <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
5  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
6  <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
7  <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"
   />
8  <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
9
10 Dynamic permissions are required in the code
11 RxPermissions rxPermissions = new RxPermissions(this);
12 rxPermissions.request(Manifest.permission.ACCESS_COARSE_LOCATION,
13                      Manifest.permission.ACCESS_FINE_LOCATION)
14 .subscribe(aBoolean -> {
15     if (aBoolean) {
16         //Permissions obtained successfully
17     }
18 });

```

- Description

connect through the IP address of the printer

▼ wifi connect

Kotlin

```

1  fun connectWifi(ip: String): Int

```

- Parameters

parameters	description
ip	the IP address of the printer (example: 192.168.1.100)

- Examples

▼ Wifi Connection Example

Java

```
1 val connection = WifiConnect()
2 val resultCode = connection.connectWifi(ip)
3 //resultCode:
4 //0: connection successful,
5 //-1: connection failed (timeout),
6 //-2: connection failed (parameter error),
7 //-3: connection failed (uninitialized))
```

3.1.4 USB connection

- permission configuration

▼ USB permission configuration

Java

```
1 Add in AndroidManifest.xml file
2 <uses-feature android:name="android.hardware.usb.host" />
3 <uses-permission android:name="android.permission.USB_PERMISSION" />
4 <uses-permission android:name="android.permission.ACCESS_USB_DEVICE" />
5 <meta-data android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED" />
```

- Description

connecting the printer via usbdevice

▼ USB connection interface

Java

```
1 fun connectUSB(usbDevice: UsbDevice): Int
```

- Parameters

parameters	description
usbDevice	usb device class provided by android system

- Examples

```

1  val usbConnect = USBConnect()
2  val connectUSB = usbConnect.connectUSB(device!!)
3  //resultCode:
4  //0: connection successful,
5  //-1: connection failed (timeout),
6  //-2: connection failed (parameter error),
7  //-3: connection failed (uninitialized))

```

Get USB Device

```

1  //device: You can refer to the Demo to obtain it,
2  //or you can refer to the following code
3  private void connectUSB() { // Traverse all USB devices in the system
4      mUsbManager = (UsbManager) thisCon.getSystemService(Context.USB_SERVICE);
5      HashMap<String, UsbDevice> deviceList = mUsbManager.getDeviceList();
6      Iterator<UsbDevice> deviceIterator = deviceList.values().iterator();
7
8      boolean HavePrinter = false;
9      while (deviceIterator.hasNext()) {
10         device = deviceIterator.next();
11         int count = device.getInterfaceCount();
12         for (int i = 0; i < count; i++) {
13             UsbInterface intf = device.getInterface(i);
14             if (intf.getInterfaceClass() == 7) { // Represents the printer type
15                 HavePrinter = true;
16                 if (mPermissionIntent != null) {
17                     // Apply for USB permission
18                     mUsbManager.requestPermission(device, mPermissionIntent);
19                 }
20             }
21         }
22     }
23 }

```

▼ Register USB system permission broadcast

Java

```

1  Intent intent = new Intent(ACTION_USB_PERMISSION);
2  intent.setPackage(thisCon.getPackageName());
3  if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S) {
4      mPermissionIntent = PendingIntent.getBroadcast(thisCon, 0, intent, FLAG_MUTABLE);
5  } else {
6      mPermissionIntent = PendingIntent.getBroadcast(thisCon, 0, intent, 0);
7  }
8  IntentFilter filter = new IntentFilter(ACTION_USB_PERMISSION);
9  filter.addAction(UsbManager.ACTION_USB_DEVICE_DETACHED);
10 filter.addAction(BluetoothDevice.ACTION_ACL_DISCONNECTED);
11 filter.addAction(BluetoothAdapter.ACTION_STATE_CHANGED);
12 if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
13     thisCon.registerReceiver(mUsbReceiver, filter, RECEIVER_EXPORTED);
14 } else {
15     thisCon.registerReceiver(mUsbReceiver, filter);
16 }

```

▼ Receiving Broadcasts

Java

```

1  private val mUsbReceiver: BroadcastReceiver = object : BroadcastReceiver() {
2      override fun onReceive(context: Context, intent: Intent) {
3          try {
4              val action = intent.action
5              if (ACTION_USB_PERMISSION == action) {
6                  synchronized(this) {
7                      val device =
8                          intent.getParcelableExtra<Parcelable>(UsbManager.EXTRA_DEVICE)
9                      as UsbDevice?
10                     if (intent.getBooleanExtra(UsbManager.EXTRA_PERMISSION_GRANTED, false)) {
11                         val usbConnect = USBConnect()
12                         val connectUSB = usbConnect.connectUSB(device!!)
13                         if (connectUSB == 0) {
14                             //Connection successful
15                         } else {
16                             //Connection failed
17                         }
18                     } else {
19                         return
20                     }
21                 }
22             } catch (e: Exception) {
23             }
24         }
25     }

```

3.1.5 Connection status

- **Description**

determine whether the printer is connected

▼ Connection Status Interface

Java |

```
1 fun isConnect(): Boolean
2 //This interface cannot obtain real-time status.
3 //Bluetooth can be obtained through system Bluetooth broadcast.
4 //Refer to Demo
```

- **Examples**

▼ Connection Status Example

Java |

```
1 connection.isConnect()
2 //true: connected,
3 //false: not connected
4 //connection is obtained by different connection classes,
5 //Bluetooth is BTConnection,
6 //wifi is WifiConnect,
7 //USB is USBConnect
```

3.1.7 Disconnection

- **Description**

disconnect a connected device



Java |

```
1 disconnect(): Boolean
```

- **Examples**

```

1 connection?.disconnect()
2 //connection is obtained by different connection classes,
3 //Bluetooth is BTConnection,
4 //wifi is WifiConnect,
5 //USB is USBConnect

```

3.2.1 Instantiation Print Class

- **Description**

all interfaces related to printing are in the PrinterTSPLHelper class

```

1 class PrinterTSPLHelper(baseConnection: BaseConnection)

```

- **Parameters**

parameters	description
baseConnection	get by creating a connection

- **Examples**

```

1 val tsplHelper = PrinterTSPLHelper(baseConnection)

```

3.2.2 Page Label Start Command

- **Description**

this interface can set some parameters for printing labels, such as width and height.

```

1  /**
2   * Set the tag start parameter
3   * @param width String Width of the printing area (in mm)
4   * @param height String Height of the printing area (in mm)
5   * @return Boolean Whether the sending is successful
6   */
7  boolean printAreaSize(String width, String height)

```

- Examples

```

1  tsplHelper.printAreaSize("100", "100")
2  tsplHelper.printText("0","0","9","0","1","1","TEXT");//Print text TEXT
3  tsplHelper.print("1","1")

```

3.2.3 Page tab end command (except line mode)

- Description

will let the printer print

```

1  //strnum: Number of prints.
2  //strcopies: Counter (default 1).
3  boolean print(String strnum, String strcopies)

```

- Examples

```

1  tsplHelper.printAreaSize("100", "100")
2  tsplHelper.printText("0","0","9","0","1","1","TEXT");//Print text TEXT
3  tsplHelper.print("1","1")

```

3.2.4 Print Self-Test Page

- **Description**

let the printer print a self-test page (with some printer parameters)



Java |

```
1  boolean selfTest()
```

- **Examples**



Java |

```
1  tsplHelper.selfTest()
```

3.2.5 Set spacing for label paper

- **Description**

set label seam parameters



Kotlin |

```
1  //Distance: The distance between two labels (unit: mm)
2  //Offset: The distance between the content in the label
3  //and the bottom of the label (unit: mm)
4  boolean gap(String distance, String offset)
```

- **Examples**



Java |

```
1  tsplHelper.gap("5","5")
```

3.2.6 Clear print buffer contents

- **Description**

clear the contents of the printer buffer area to prevent duplication of contents



Java |

```
1  boolean cls()
```

- Examples

```
1  tsplHelper.cls()
```

3.2.7 Text Printing

- Description

this interface is used for text printing

```
1  (1) boolean printText(String x_pos,String y_pos,String font,String
2                      rotation,String x_multiplication,String y_multiplication,
3                      int alignment,String code_data)
4  //x_pos: Text starting x coordinate
5  //y_pos: Text starting y coordinate
6  //font: 0: Monotype CG Triumvirate Bold Condensed, font width and height is
           stretchable
7      1: 8 x 12 fixed pitch dot font
8      2: 12 x 20 fixed pitch dot font
9      3: 16 x 24 fixed pitch dot font
10     4: 24 x 32 fixed pitch dot font
11     5: 32 x 48 dot fixed pitch font
12     6: 14 x 19 dot fixed pitch font OCR-B
13     7: 21 x 27 dot fixed pitch font OCR-B
14     8: 14 x25 dot fixed pitch font OCR-A
15     9: Only this font can print Chinese.
16
17  //Rotation: Print direction.
18     0 : No rotation
19     90 : degrees, in clockwise direction
20    180 : degrees, in clockwise direction
21    270 : degrees, in clockwise direction
22  //x_multiplication: The multiple of text stretching in the x-axis direction
23  // (font size 0 and font size 9 represent the actual size of the font, in px).
24  //y_multiplication: The multiple of text stretching in the y-axis direction
25  // (font size 0 and font size 9 represent the actual size of the font, in px).
26  //Alignment: Alignment (some older models do not support this function,
27  // you can remove this parameter)
28     1: Left Align
29     2: Center
30     3: Right Align
31  //code_data: Text data.
```

```

1  (2) boolean printText(String x_pos,String y_pos,String font,
2                        String rotation,int size,int alignment,String code_da
   ta)
3  //x_pos: Text starting x coordinate
4  //y_pos: Text starting y coordinate
5  //font: Font: Same as above
6  //rotation: Print orientation.
7      0 : No rotation
8      90 : degrees, in clockwise direction
9      180 : degrees, in clockwise direction
10     270 : degrees, in clockwise direction
11  //size: Font size 1~7.
12  //Alignment: Alignment (some older models do not support this function,
13  // you can remove this parameter)
14      1: Left Align
15      2: Center
16      3: Right Align
17  //code_data: Text data.

```

- Examples

```

1  tsplHelper.printAreaSize("100","200")
2  tsplHelper.cls();
3  tsplHelper.printText("0","0","1","0","0","0","TEXT")
4  tsplHelper.print("1","1")

```

3.2.8 Print Direction

- Description

used to set the overall orientation of the printed content

```

1  boolean direction(String direction)
2  //direction:
3      0: Vertical.
4      1: Horizontal.

```

- Examples

```

1  tsplHelper.printAreaSize("100","200")
2  tsplHelper.direction("0");
3  tsplHelper.printText("0","0","0","0","0","TEXT")
4  tsplHelper.print("1","1")

```

3.2.9 Barcode

- Description

this interface is used to print bar codes

```

1  boolean printBarcode(String x_pos, String y_pos, String code_type,
2                      String height, String readable, String rotation,
3                      String narrow, String wide, String code_data)
4  parameter:
5      x_pos: The start abscissa of the barcode.
6      y_pos: The start ordinate of the barcode.
7      code_type: Barcode Type:
8          128, 128M, EAN128 , 39 , 93, UPCA , MSI , ITF14 , EAN13
9      height: Barcode height in PX.
10     readable: Whether the barcode data is visible
11         0: not readable
12         1: human readable
13     rotation: Barcode Orientation:
14         0 : No rotation
15         90 : degrees, in clockwise direction
16         180 : degrees, in clockwise direction
17         270 : degrees, in clockwise direction
18     narrow : The unit width of the narrow bar (default 1).
19     Wide: The unit width of the wide barcode (default 1).
20     code_data: Barcode data.

```

- Examples

```

1  tsplHelper.printAreaSize("100","200")
2  tsplHelper.cls();
3  tsplHelper.printBarcode("0","0","128","100","1","0","1","1","1234567890")
4  tsplHelper.print("1","1")

```

3.2.10 Print QR Code

- Description

the interface is used to print two-dimensional code

Java |

```
1  boolean printQRCode(String x_pos,String y_pos,String ecc_level,String width,
2                          String mode,String rotation,String code_data)
3  parameter:
4      rotation: Barcode Orientation:
5          0 : No rotation
6          90 : degrees, in clockwise direction
7          180 : degrees, in clockwise direction
8          270 : degrees, in clockwise direction
9      X: The starting abscissa of the QR code.
10     Y: The starting ordinate of the QR code.
11     ecc_level: Error correction level
12         L : 7%
13         M : 15%
14         Q : 25%
15         H : 30%
16     width: Width: 1~10
17     Mode: mode
18         A: Auto
19         M: Manual
20     Data: The data of the QR code.
```

- Examples

Java |

```
1  tsplHelper.printAreaSize("100","200")
2  tsplHelper.cls();
3  tsplHelper.printQRCode("0","0","M","6","A","0","1234567890")
4  tsplHelper.print("1","1")
```

3.2.11 Print Lines

- Description

this interface can be used to print lines

▼ Java |

```
1 boolean bar(String x_pos,String y_pos,String width, String height
2 Parameter:
3     x_pos: The starting X coordinate.
4     y_pos: The starting Y coordinate.
5     width: The width of the line (unit: PX).
6     height: The height of the line (unit: PX).
```

- Examples

▼ Java |

```
1 tsplHelper.printAreaSize("100","200")
2 tsplHelper.bar("10","10","100","2");//Horizontal line
3 tsplHelper.bar("10","10","2","100");//Vertical Line
4 tsplHelper.print("1","1")
```

3.2.12 Print Picture

- Description

this interface is used to print pictures

▼ Java |

```
1 int printImage(String x_pos, String y_pos, Bitmap bmp , boolean isNegate,
2               boolean isLZ0, ImageTypeEnum imageTypeEnum )
3 Parameter:
4     X: The x-coordinate at which the image starts.
5     Y: The y-coordinate at the beginning of the image.
6     bmp: Picture of the Bitmap object.
7     isNegate:The picture is reversed.
8         true: Normal display.
9         false: Inverted display.
10    isLZ0:Whether or not to compress (the printer must support compression
11    ).
12    imageTypeEnum: Picture algorithms.
13        THRESHOLD: Binary arithmetic.
14        RASTERMONO: Jitter algorithms.
15        POLYMERIZATION: Aggregation algorithms.
```

- Examples

```
▼ Kotlin |
1 //The label height should be set to be longer than the height of the image.
2 tsplHelper.printAreaSize("100","200")
3 tsplHelper.cls()
4 tsplHelper.printImage("10","10",bitmap,true,false,ImageTypeEnum.RASTERMONO)
5 tsplHelper.print("1","1")
```

3.2.13 Paper cutting

- Description

this interface is used for cutter

```
▼ Java |
1 boolean cut()
```

- Examples

```
▼ Java |
1 tsplHelper.cut()
```

3.2.14 Print Rectangular Box

- Description

this interface is used to print rectangular boxes

```

1  boolean box(String x_start,String y_start,String x_end, String y_end,
2              String thickness)
3  parameter:
4      x_start: X coordinates in the upper left corner.
5      y_start: Y coordinates in the upper left corner.
6      x_end: X coordinates in the lower right corner.
7      y_end: Y coordinates in the lower right corner.
8      thickness: Line width.

```

- Examples

```

1  tsplHelper.printAreaSize("100","200")
2  tsplHelper.cls()
3  tsplHelper.box("10","10","100","100","2")
4  tsplHelper.print("1","1")

```

3.2.15 Send data

- Description

the interface can send data directly to the printer

```

1  boolean writeData(byte[] bData)
2  parameter:
3      bData: The data that needs to be sent to the printer.

```

- Examples

```

1  tsplHelper.writeData(new byte[]{0x0d,0x0a})

```

3.2.16 Read Data Function

- **Description**

this interface can directly read the data returned by the printer



Java |

```
1 byte[] readData(int timeout)
2 parameter:
3     timeout: Timeout period in milliseconds.
```

- **Examples**



Java |

```
1 bData = tsplHelper.readData(2000)//bData is the data that is read
```

3.2.17 Bold Text

- **Description**

this interface is used for text font bold



Java |

```
1 boolean bold(int bold)
2 parameter:
3     bold: 0: No bolding, 1: Bold.
```

- **Examples**



Java |

```
1 tsplHelper.bold(1)
```

3.2.18 Get Printer Status

- **Description**

this interface is used to obtain the printer status

▼ getPrinterStatus

Java |

```
1  int getPrinterStatus()
2  Return:
3      PrinterTSPLHelper.STATUS_DISCONNECT:    Disconnect
4      PrinterTSPLHelper.STATUS_TIMEOUT:       The query timed out
5      PrinterTSPLHelper.STATUS_OK:           The printer is fine
6      PrinterTSPLHelper.STATUS_COVER_OPENED:  Open the lid
7      PrinterTSPLHelper.STATUS_NOPAPER:      Lack of paper
8      PrinterTSPLHelper.STATUS_OVER_HEATING: superheating
9      PrinterTSPLHelper.STATUS_PRINTING:     Printing
```

• Examples



Kotlin |

```
1  val printerStatus = tsplHelper.getPrinterStatus()
```

3.2.19 Set black mark position

• Description

this interface is used to set the Black Label



Java |

```
1  boolean setBlackPosition(int position)
2  parameter:
3      position:
4          1: 2-inch right front black mark,
5          2: 3-inch right front black mark,
6          3: 2-inch right rear black mark,
7          4: 3-inch right rear black mark.
```

• Examples



Kotlin |

```
1  tsplHelper.setBlackPosition(1);
```

3.2.20 Text paragraph printing

- **Description**

this interface is used to print multiple lines of text. If the width exceeds the set width, the text will wrap automatically.

Java |

```
1  boolean printBlock(int startX, int startY, int width, int height,
2                      int font, int rotation, int multiplicationX,
3                      int multiplicationY, int space, int alignment,
4                      String content)
5  parameter:
6      startX: x coordinate in the upper left corner of the text box.
7      startY: y coordinate in the upper left corner of the text box.
8      width: width of the text box.
9      height: height of the text box.
10     font: font size (0: 16*16, 1: 24*24)
11     rotation: Rotation direction. (0, 90, 180, 270)
12     multiplicationX: Font X-axis magnification.
13     multiplicationY: Font Y-axis direction developed multiples.
14     space: Line spacing.
15     alignment: (1: Align left, 2: Center, 3: Align right)
16     content: Print content.
```

- **Examples**

Kotlin |

```
1  tsplHelper.printAreaSize("100","200")
2  tsplHelper.cls()
3  tsplHelper.printBlock(0,0,100,100,0,0,2,2,16,2,"Test,Test,Test,Test")
4  tsplHelper.print("1","1")
```

3.2.21 Print PDF417

- **Description**

this interface is used to print PDF417 type bar code.

```

1  boolean printPDF417(int xPos, int yPos, int width, int height,
2                      int rotate, ArrayList<String> option,
3                      String expression)
4  parameter:
5      xPos: x coordinate.
6      yPos: y coordinate.
7      width: width of the barcode.
8      height: height of the barcode.
9      rotate: Rotation direction. (0, 90, 180, 270)
10     option: Barcode parameters. (Can be transmitted null)
11         P Data compression method
12             0: Auto-encoding
13             1: Binary mode
14         E Error correction level. Range: 0 to 8
15         M The center pattern of the barcode area
16             0: pattern is printed in the upper left area
17             1: pattern is printed in the central area
18         Ux, y, c Display text.
19             x: Text x-coordinate
20             y: Text y coordinates
21             c: The maximum number of characters in a line
22         W Module width. Range: 2 to 9
23         H Column high. Range: 4 to 99
24         R Maximum number of rows
25         C Maximum number of columns
26         T Truncation.
27             0: Not truncated
28             1: truncation
29         Lm Content length, 1<m<2048
30     expression: Barcode content.

```

- Examples

```

1  tsplHelper.printAreaSize("100", "100");
2  tsplHelper.cls();
3  ArrayList<String> option = new ArrayList<>();
4  option.add("P1");
5  option.add("E4");
6  option.add("M1");
7  option.add("U100,500,10");
8  option.add("W6");
9  option.add("H6");
10 option.add("R60");
11 option.add("C4");
12 option.add("T1");
13 option.add("L297");
14 String data = "Data" +
15 "compression method: P1" +
16 "Error correction level: E4" +
17 "Center pattern in barcode area: M1" +
18 "Human Readable: Yes: U100,500,10" +
19 "Module Width 6 dots: W6" +
20 "Bar Height 6 dots: H6" +
21 "Maximum Number of Rows: 60 Rows: R60" +
22 "Maximum number of columns: 4 Cols: C4" +
23 "Truncation:1: T1" +
24 "Expression length:297: L297";
25 tsplHelper.printPDF417(50, 50, 900, 600, 0, option, data);
26 tsplHelper.print("1", "1");

```

3.2.22 Set printer density

- Description

this interface is used to set the printer concentration, and the concentration range should be determined according to the specific printer model.

```

1  boolean density(String density)
2  parameter:
3      density: Concentration. (0~15)

```

- Examples

```
1 tsplHelper.density("5");
```

3.2.23 Set printer speed

- Description

this interface is used to set the speed of the printer, and the speed range is determined according to the specific printer model.

```
1 boolean speed(String speed)
2 parameter:
3     speed: print speed. (1~12)
```

- Examples

```
1 tsplHelper.speed("5");
```

3.2.24 Set Inverse Box

- Description

the interface is used to print the effect of white characters on a black background.

```
1 boolean reverse(String xStart, String yStart, String width, String height)
2 parameter:
3     xStart: X coordinate.
4     yStart: Y coordinate.
5     width: The width of the reverse white area.
6     height: The height of the reverse white area.
```

- Examples

```

1  tsplHelper.printAreaSize("100","200")
2  tsplHelper.cls()
3  tsplHelper.printText("100","100","9","0","32","32","REVERSE" )
4  tsplHelper.reverse("90","90","128","40")
5  tsplHelper.print("1","1")

```

3.2.25 Set CodePage

- Description

This interface is used to set up the code pages of the printer and SDK.

```

1  boolean codepage(String codepage)
2  parameter:
3      codepage: code pages.
4  USA: United States, BRI: British, GER: German, FRE: French, DAN: Danish,
5  ITA: Italian, SPA: Spanish, SWE: Swedish, SWI: Swiss, 437: United States,
6  850: Multilingual, 852: Slavic, 860: Portuguese, 863: Canadian/French,
7  865: Nordic, 857: Turkish, 1250: Central European, 1252: Latin I,
8  1253: Greek, 1254: Turkish, 936: Chinese, Iran: Persian, Iran II: Persian I
   I

```

- Examples

```

1  tsplHelper.codepage("437")

```

4.1.1 Enable SDK to send data logs

- Description

this function is used to record the data sent by the SDK to the printer and is used to determine whether the data is correct. The path to log is in

/sdcard/Android/data/app package name/files/Documents/SDK_log.txt the app package name is the package name of your project. Opening method:

```
1 Constants.isWriteLog = true  
2 Constants.isHex = true
```