

Common SDK v1.0.0 Interface documentation

Document modification records

I. Instructions for Use

1.1 Engineering Configuration Description

1.2 Obfuscated 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 Get the printer SN

3.2.3 Get the printer model

3.2.4 Get the printer version number

3.2.5 Set the Get Print Results switch

3.2.6 Set the printer paper type

3.2.7 Get the printer paper type

4.1.1 Paper type

Document modification records

serial Number	version Number	modify content	modifier	review	date modified
01	v1.0.0	document creation	qiux	ldc	2025-09-09

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 Obfuscated 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.

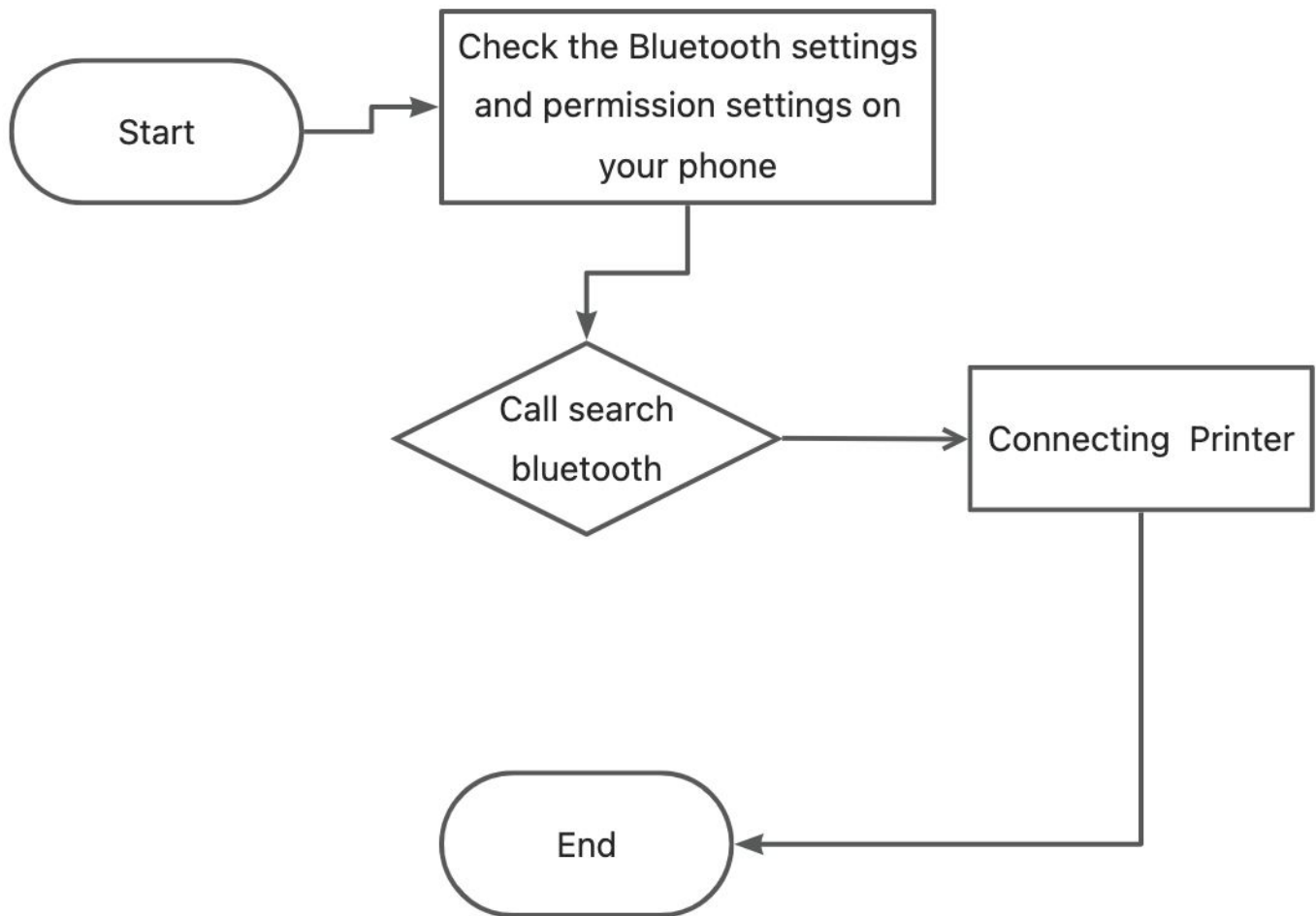
▼ initialization

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
```

```
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();//开启搜索
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

Bluetooth connection

Kotlin

```
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

initialize PrinterHelper at APP startup

initialization

Java

```
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

Bluetooth permission configuration

Java

```

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

```

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

```

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  />
9  <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
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

```

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

- pay attention to permission configuration

▼

USB permission configuration

Java |

```

1  //The USB connection requires configuration permissions,
2  //which are added in the AndroidManifest.xml file
3  <uses-feature android:name="android.hardware.usb.host" />
4  <uses-permission android:name="android.permission.USB_PERMISSION" />
5  <uses-permission android:name="android.permission.ACCESS_USB_DEVICE" />
6  <meta-data android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED" /
>

```

- Description

connecting the printer via usbdevice

▼

USB permission configuration

Java |

```

1  fun connectUSB(usbDevice: UsbDevice): Int

```

- Parameters

parameters	description
usbDevice	usb device class provided by android system

- Examples

Java

```

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))

```

Java

```

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 print-related interfaces are in the PrinterCommonHelper class

```

1  class PrinterCommonHelper(baseConnection: BaseConnection)

```

- **Parameters**

parameters	description
baseConnection	get by creating a connection

- **Examples**

```

1  val commandHelper = PrinterCommonHelper(baseConnection)

```

3.2.2 Get the printer SN

- **Description**

This interface can obtain the serial number of the printer

```
1  /**Get the printer SN
2      * @return
3      */
4  public String getPrinterSN()
```

- Examples

```
1  val printerSN = commandHelper?.printerSN
```

3.2.3 Get the printer model

- Description

This interface can get the model of the printer

```
1  public String getPrinterModel()
```

- Examples

```
1  val printerModel = commandHelper?.printerModel
```

3.2.4 Get the printer version number

- Description

This interface can obtain the firmware version number of the printer

```
1  public String getPrinterVersion()
```

- Examples



Java |

```
1 val printerVersion = commandHelper?.printerVersion
```

3.2.5 Set the Get Print Results switch

- Description

This interface can set a switch on whether the printer needs to return the printer results



Kotlin |

```
1 //isOpen(true:open, false:close)
2 public boolean setRePrintSwitch(boolean isOpen)
```

- Examples



Java |

```
1 commandHelper?.setRePrintSwitch(true)
```

3.2.6 Set the printer paper type

- Description

This interface can set different paper types for the printer



Java |

```
1 //typeEnum:Enumeration of paper types that can be set (see 4.1.1 for details)
2 //return true:Send successful, false: Failed to send
3 public boolean setPrinterPaperType(PaperTypeEnum typeEnum)
```

- Examples



Java |

```
1 commandHelper?.setPrinterPaperType(PaperTypeEnum.values()[which])
```

3.2.7 Get the printer paper type

- Description

This interface can get the paper type that is currently set by the printer

▼ Kotlin |

```
1  /**
2   * @return int Paper type (see 4.1.1 for details)
3   */
4   public int getPrinterPaperType()
```

- Examples

▼ Java |

```
1  val printerVersion = commandHelper?.printerPaperType
```

4.1.1 Paper type

值	描述
PaperTypeEnum.CONTINUOUS	Continuous paper
PaperTypeEnum.LABEL	Label paper
PaperTypeEnum.TWO_FORMER_BLACK_LABEL	2 inches of black label
PaperTypeEnum.THREE_FORMER_BLACK_LABEL	3 inches of black label
PaperTypeEnum.TWO_AFTER_BLACK_LABEL	2 inches back black mark
PaperTypeEnum.THREE_AFTER_BLACK_LABEL	3 inches back black mark