

# SDK Integration Guide

## Document Revision History

| No. | Version | Changes         | Author   | Reviewer   | Date       |
|-----|---------|-----------------|----------|------------|------------|
| 1   | v1.0.0  | Initial version | Qiu Xian | Lin Dechun | 2026.04.11 |

## 1. SDK Package Purpose

Provides interfaces suitable for the Android platform, enabling customers to integrate our printers into their software systems by calling our Android APIs. Through our provided interfaces, customers can achieve seamless integration between printers and software systems.

## 2. Supported Printer Models

| No. | Printer Language | Firmware Version |
|-----|------------------|------------------|
| 01  | ESC              | All versions     |
| 02  | CPCL             | All versions     |
| 03  | TSPL             | All versions     |
| 04  | ZPL              | All versions     |

**Note:** Devices with other printer languages or firmware versions outside the supported range are not supported.

## 3. SDK Usage Evaluation

### 3.1 Operating Environment

#### Android Version

- Android 5.0 and above

#### Android Devices

- Android phones
- Android tablets

- Android development boards

## Development Environment

- JDK: Version 1.6 or above
- Android Studio: Version 2.3.2 or above
- Android SDK: Version 1.6 or above

## 3.2 SDK Demo Experience Process

If you need to evaluate the integration, you can experience the SDK Demo first:

1. Open Android Studio editor
  2. Import SDK Demo
  3. Compile the project
  4. Run on physical device
  5. Experience SDK Demo
- 

## 4. SDK Integration

---

### 4.1 Development Integration Guide

If you are a developer, the following steps will help you get started with SDK integration:

1. **View SDK DEMO:** Open SDK DEMO with Android Studio and refer to the sample code to understand how to use the SDK correctly.
2. **View API Documentation:** Check the SDK API documentation for detailed interface descriptions.
3. **Read Usage Instructions:** Before starting development, carefully read the "Usage Instructions" section in the documentation to understand project configuration, permission configuration, and device settings.
4. **SDK Call Flow:** The sample demonstration section in the documentation contains the SDK call flow for Android. Please study it carefully as it is crucial for successful integration.

### 4.2 Important Notes

- For Bluetooth-related issues, please refer to the precautions document
  - If you need further assistance or have any questions, please contact our support team
- 

## 5. Bluetooth Usage Precautions

---

### 5.1 Enable Bluetooth

Before using Bluetooth functionality, ensure that the device Bluetooth is enabled.

### 5.2 Enable GPS

For Android 6.0 and above, Bluetooth scanning requires GPS location service to be enabled.

## 5.3 Permission Configuration

Configure the following permissions in `AndroidManifest.xml`:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
<uses-permission android:name="android.permission.BLUETOOTH_ADVERTISE" />
<uses-permission android:name="android.permission.BLUETOOTH_SCAN" />
```

## 5.4 Dynamic Permission Request

Request permissions dynamically in code (using RxPermissions):

```
RxPermissions(this).request(
    Manifest.permission.BLUETOOTH,
    Manifest.permission.ACCESS_FINE_LOCATION,
    Manifest.permission.ACCESS_COARSE_LOCATION,
    Manifest.permission.BLUETOOTH_SCAN,
    Manifest.permission.BLUETOOTH_CONNECT
)
    .subscribe {
        if (it) {
            // Permission granted successfully
        }
    }
```

---

## 6. Project Configuration

### 6.1 Gradle Dependency Configuration

Add dependencies in your project's `build.gradle`:

```
implementation files('libs/UxPrint-sdk-v1.xx.xx.aar')
implementation files('libs/lzo-sdk-v1.xx.xx.aar')
```

### 6.2 ProGuard Configuration

Add the following rules in `proguard-rules.pro`:

```
-keep class com.printer.sdk.** { *; }
-keep interface com.printer.sdk.** { *; }
```

## 7. Permission Configuration

Add corresponding permissions in `AndroidManifest.xml` based on the connection method used:

### 7.1 Bluetooth Connection Permissions

```
<!-- Basic Bluetooth permissions -->
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />

<!-- Android 12+ Bluetooth permissions -->
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
<uses-permission android:name="android.permission.BLUETOOTH_SCAN" />

<!-- Location permissions (required for Bluetooth scanning) -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

### 7.2 WiFi Connection Permissions

```
<!-- WiFi network permissions -->
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
```

### 7.3 USB Connection Permissions

```
<!-- USB permission -->
<uses-feature android:name="android.hardware.usb.host" />
```

USB device filter configuration (optional):

```
<activity android:name=".UsbPermissionActivity">
    <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" />
</activity>
```

## 8. SDK Initialization

Initialize the SDK in your Application class:

### 8.1 Basic Initialization

```
class MyApplication : Application() {
    override fun onCreate() {
        super.onCreate()
        PrinterSDK.initialize(this)
    }
}
```

### 8.2 Initialization with Logging

```
class MyApplication : Application() {
    override fun onCreate() {
        super.onCreate()
        PrinterSDK.initialize(
            applicationContext = this,
            enableIoLog = true,
            ioLogConfig = PrinterIoLogConfig(
                enableReadLog = true,
                enableWriteLog = true,
                logAsHexString = true
            )
        )
    }
}
```

**Log File Path:**

- File name: `SDK_log.txt`
- Primary path:  
`/storage/emulated/0/Android/data/<package_name>/files/Documents/SDK_log.txt`
- Fallback path: `/data/data/<package_name>/files/SDK_log.txt`

---

## 9. Quick Start

---

### 9.1 Bluetooth Connection Example

```
// Create Bluetooth connection
val connection = BluetoothConnection()

// Connect to printer
lifecycleScope.launch {
    val result = connection.connect(
        ConnectionParams.Bluetooth(address = "00:11:22:33:44:55")
    )
    when (result) {
        is Result.Success -> {
            // Connected successfully, send data
            connection.write(printData)
        }
        is Result.Failure -> {
            Log.e("Bluetooth", "Connection failed: ${result.error.message}")
        }
    }
}
```

### 9.2 WiFi Connection Example

```
// Create WiFi connection
val connection = WiFiConnection()

// Connect to printer
lifecycleScope.launch {
    val result = connection.connect(
        ConnectionParams.WiFi(
            ip = "192.168.1.100",
            port = 9100,
            timeout = 5000
        )
    )
    when (result) {
        is Result.Success -> {
            // Connected successfully, send data
            connection.write(printData)
        }
        is Result.Failure -> {
            Log.e("WiFi", "Connection failed: ${result.error.message}")
        }
    }
}
```

```
}
```

## 9.3 USB Connection Example

```
// Get USB device
val usbManager = getSystemService(Context.USB_SERVICE) as UsbManager
val deviceList = usbManager.deviceList

if (deviceList.isNotEmpty()) {
    val device = deviceList.values.first()

    // Request USB permission
    val permissionIntent = PendingIntent.getBroadcast(
        this, 0, Intent(ACTION_USB_PERMISSION), PendingIntent.FLAG_IMMUTABLE
    )
    usbManager.requestPermission(device, permissionIntent)

    // Create USB connection
    val connection = USBConnection(this)

    // Connect to printer
    lifecycleScope.launch {
        val result = connection.connect(ConnectionParams.USB(device))
        when (result) {
            is Result.Success -> {
                // Connected successfully, send data
                connection.write(printData)
            }
            is Result.Failure -> {
                Log.e("USB", "Connection failed: ${result.error.message}")
            }
        }
    }
}
}
```

---

## 10. FAQ

### Q1: Cannot scan Bluetooth devices?

#### Solution:

1. Ensure Bluetooth is enabled
2. For Android 6.0+, ensure GPS is enabled
3. Ensure location permissions are granted
4. Ensure Bluetooth-related permissions are granted

## Q2: Bluetooth connection failed?

### Solution:

1. Ensure the printer is powered on with sufficient battery
2. Ensure the printer is not connected to another device
3. Try re-pairing the Bluetooth device
4. Check if the Bluetooth address is correct

## Q3: WiFi connection timeout?

### Solution:

1. Ensure the phone and printer are on the same network
2. Check if the printer IP address is correct
3. Check if the port is correct (default 9100)
4. Try pinging the printer IP address

## Q4: USB connection not responding?

### Solution:

1. Ensure the USB cable is properly connected
2. Ensure USB permission is granted
3. Check if the device supports USB Host mode

---

## 11. Technical Support

---

If you need further assistance or have any questions, please contact our support team. They will be happy to provide you with support and guidance.