

Connection SDK 接口文档

文档修改记录

序号	版本号	修改内容	修改者	评审者	修改日期
1	v1.0.0	初始版本	邱贤	林德春	2026.04.11
2	v1.0.1	新增错误码 UXErrorCode 说明, PrinterError 增加 errorCode 字段	邱贤	林德春	2026.08.04

1. 使用说明

1.1 工程配置说明

在项目的 `build.gradle` 中添加依赖:

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

1.2 权限配置

根据使用的连接方式, 在 `AndroidManifest.xml` 中添加相应权限:

蓝牙连接权限

```
<!-- 蓝牙基础权限 -->
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />

<!-- Android 12+ 蓝牙权限 -->
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
<uses-permission android:name="android.permission.BLUETOOTH_SCAN" />
```

WiFi连接权限

```

<!-- WiFi网络权限 -->
<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" />

```

USB连接权限

```

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

```

USB设备过滤器配置（可选）：

```

<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>

```

1.3 混淆配置

在 `proguard-rules.pro` 中添加：

```

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

```

1.4 初始化

在 Application 中初始化 SDK：

```

class MyApplication : Application() {
    override fun onCreate() {
        super.onCreate()
        // 基础初始化
        PrinterSDK.initialize(this)

        // 或带日志功能的初始化
        PrinterSDK.initialize(
            applicationContext = this,
            enableIoLog = true,
            ioLogConfig = PrinterIoLogConfig(
                enableReadLog = true,
                enableWriteLog = true,
            )
        )
    }
}

```

```
        logAsHexString = true
    )
}
}
```

日志文件路径：

- 文件名： SDK_log.txt
- 优先路径： /storage/emulated/0/Android/data/<包名>/files/Documents/SDK_log.txt
- 备用路径： /data/data/<包名>/files/SDK_log.txt

2. 接口目录

Connection SDK 接口文档

文档修改记录

1. 使用说明

1.1 工程配置说明

1.2 权限配置

蓝牙连接权限

WiFi连接权限

USB连接权限

1.3 混淆配置

1.4 初始化

2. 接口目录

3. SDK 初始化

3.1 initialize

3.2 PrinterIoLogConfig

4. 连接参数类

4.1 ConnectionParams.WiFi

4.2 ConnectionParams.Bluetooth

4.3 ConnectionParams.USB

5. 连接接口 (PrinterConnection)

5.1 connect

5.2 disconnect

5.3 isConnected

5.4 write

5.5 read

6. 蓝牙连接类 (BluetoothConnection)

7. WiFi连接类 (WiFiConnection)

8. USB连接类 (USBConnection)

9. 附录

9.1 错误类型说明

9.2 返回值类型

3. SDK 初始化

3.1 initialize

描述：初始化 SDK，必须在 Application 的 onCreate 中调用。

方法签名：

```
fun initialize(  
    applicationContext: Context,  
    enableIoLog: Boolean = false,  
    ioLogConfig: PrinterIoLogConfig = PrinterIoLogConfig()  
)
```

参数：

参数名	类型	必填	默认值	说明
applicationContext	Context	是	-	Application Context
enableIoLog	Boolean	否	false	是否启用IO日志
ioLogConfig	PrinterIoLogConfig	否	PrinterIoLogConfig()	IO日志配置

示例代码：

```
class MyApplication : Application() {  
    override fun onCreate() {  
        super.onCreate()  
        // 基础初始化  
        PrinterSDK.initialize(this)  
  
        // 或带日志功能的初始化  
        PrinterSDK.initialize(  
            applicationContext = this,  
            enableIoLog = true,  
            ioLogConfig = PrinterIoLogConfig(  
                enableReadLog = true,  
                enableWriteLog = true,  
                logAsHexString = true  
            )  
        )  
    }  
}
```

3.2 PrinterIoLogConfig

描述：IO日志配置类。

属性：

属性名	类型	必填	默认值	说明
enableReadLog	Boolean	否	true	是否启用读取日志
enableWriteLog	Boolean	否	true	是否启用写入日志
logAsHexString	Boolean	否	true	是否以十六进制字符串格式记录日志

日志文件路径：

- 文件名： `SDK_log.txt`
- 优先路径： `/storage/emulated/0/Android/data/<包名>/files/Documents/SDK_log.txt`
- 备用路径（当外部存储不可用时）： `/data/data/<包名>/files/SDK_log.txt`

示例代码：

```
val config = PrinterIoLogConfig(  
    enableReadLog = true,  
    enableWriteLog = true,  
    logAsHexString = true  
)  
PrinterSDK.initialize(this, enableIoLog = true, ioLogConfig = config)
```

4. 连接参数类

4.1 ConnectionParams.WiFi

描述：WiFi 连接参数。

属性：

属性名	类型	必填	默认值	说明
ip	String	是	-	打印机 IP 地址
port	Int	否	9100	端口号
timeout	Long	否	5000	连接超时时间（毫秒）

示例代码：

```
val params = ConnectionParams.WiFi(  
    ip = "192.168.1.100",  
    port = 9100,  
    timeout = 5000  
)
```

4.2 ConnectionParams.Bluetooth

描述：蓝牙连接参数。

属性：

属性名	类型	必填	说明
address	String	是	蓝牙设备地址（格式：XX:XX:XX:XX:XX:XX）

示例代码：

```
val params = ConnectionParams.Bluetooth(  
    address = "00:11:22:33:44:55"  
)
```

4.3 ConnectionParams.USB

描述：USB 连接参数。

属性：

属性名	类型	必填	说明
device	UsbDevice	是	USB 设备对象

示例代码：

```
val usbManager = getSystemService(Context.USB_SERVICE) as UsbManager  
val deviceList = usbManager.deviceList  
val device = deviceList.values.first()  
  
val params = ConnectionParams.USB(device = device)
```

5. 连接接口 (PrinterConnection)

5.1 connect

描述：连接打印机。

方法签名：

```
suspend fun connect(params: ConnectionParams): Result<Unit>
```

参数：

参数名	类型	必填	说明
params	ConnectionParams	是	连接参数 (WiFi/Bluetooth/USB)

返回值: `Result<Unit>` 连接结果

示例代码:

```
// WiFi连接
val connection = WiFiConnection()
val result = connection.connect(ConnectionParams.WiFi("192.168.1.100"))
when (result) {
    is Result.Success -> Log.d("Connection", "Connected")
    is Result.Failure -> Log.e("Connection", "Failed:
    ${result.error.message}")
}
```

5.2 disconnect

描述: 断开打印机连接。

方法签名:

```
suspend fun disconnect(): Result<Unit>
```

返回值: `Result<Unit>` 断开结果

示例代码:

```
connection.disconnect()
```

5.3 isConnected

描述: 判断是否已连接。

方法签名:

```
val isConnected: Boolean
```

返回值: 是否已连接

示例代码:

```
if (connection.isConnected) {
    // 已连接
}
```

5.4 write

描述：向打印机写入数据。

方法签名：

```
suspend fun write(data: ByteArray): Result<Unit>
```

参数：

参数名	类型	必填	说明
data	ByteArray	是	要写入的数据

返回值：Result<Unit> 写入结果

示例代码：

```
val data = byteArrayOf(0x1B, 0x40) // ESC @ 初始化打印机
connection.write(data)
```

5.5 read

描述：从打印机读取数据。

方法签名：

```
suspend fun read(timeout: Long = 2000): Result<ByteArray>
```

参数：

参数名	类型	必填	默认值	说明
timeout	Long	否	2000	超时时间（毫秒）

返回值：Result<ByteArray> 读取的数据

示例代码：

```

val result = connection.read(timeout = 3000)
when (result) {
    is Result.Success -> {
        val data = result.value
        // 处理数据
    }
    is Result.Failure -> Log.e("Read", "Failed: ${result.error.message}")
}

```

6. 蓝牙连接类 (BluetoothConnection)

描述：蓝牙连接实现类。

权限配置：

```

<!-- 蓝牙基础权限 -->
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />

<!-- Android 12+ 蓝牙权限 -->
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
<uses-permission android:name="android.permission.BLUETOOTH_SCAN" />

```

示例代码：

```

// 创建蓝牙连接
val connection = BluetoothConnection()

// 连接打印机
lifecycleScope.launch {
    val result = connection.connect(
        ConnectionParams.Bluetooth(address = "00:11:22:33:44:55")
    )
    when (result) {
        is Result.Success -> {
            // 连接成功，发送数据
            connection.write(printData)
        }
        is Result.Failure -> {
            Log.e("Bluetooth", "Connection failed: ${result.error.message}")
        }
    }
}

```

7. WiFi连接类 (WiFiConnection)

描述：WiFi 连接实现类。

权限配置：

```
<!-- WiFi网络权限 -->
<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" />
```

示例代码：

```
// 创建WiFi连接
val connection = WiFiConnection()

// 连接打印机
lifecycleScope.launch {
    val result = connection.connect(
        ConnectionParams.WiFi(
            ip = "192.168.1.100",
            port = 9100,
            timeout = 5000
        )
    )
    when (result) {
        is Result.Success -> {
            // 连接成功，发送数据
            connection.write(printData)
        }
        is Result.Failure -> {
            Log.e("WiFi", "Connection failed: ${result.error.message}")
        }
    }
}
```

8. USB连接类 (USBConnection)

描述：USB 连接实现类。

权限配置：

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

示例代码：

```
// 获取USB设备
```

```
val usbManager = getSystemService(Context.USB_SERVICE) as UsbManager
val deviceList = usbManager.deviceList

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

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

    // 创建USB连接
    val connection = USBConnection(this)

    // 连接打印机
    lifecycleScope.launch {
        val result = connection.connect(ConnectionParams.USB(device))
        when (result) {
            is Result.Success -> {
                // 连接成功, 发送数据
                connection.write(printData)
            }
            is Result.Failure -> {
                Log.e("USB", "Connection failed: ${result.error.message}")
            }
        }
    }
}
```

9. 附录

9.1 错误类型说明

错误类型	说明	可能的触发场景
ConnectionError	连接错误	打印机连接失败、网络连接问题、蓝牙配对失败
CommunicationError	通信错误	数据传输失败、打印机响应异常、通信超时
InvalidParameterError	参数无效错误	传入的参数值超出范围、参数格式不正确
TimeoutError	超时错误	操作执行超时、打印机无响应
NotConnectedError	未连接错误	尝试在未连接状态下执行操作
UnsupportedOperationError	不支持的操作错误	打印机型号不支持的功能、协议不支持的命令
UnknownError	未知错误	其他未分类的错误情况

错误码说明：接口返回 `Result.Failure` 时，`PrinterError` 携带 `errorCode` 字段（对应 `UXPErrorCode.value`）。每个错误类型默认取下表兜底错误码，SDK 各调用点会传入更具体的错误码；完整错误码表见《UXPErrorCode 错误码说明》（Guide 目录）。

兜底错误码	数值	说明
ConnectionError	1001	连接错误
CommunicationError	1002	通讯错误
InvalidParameterError	1003	参数无效
TimeoutError	1004	超时
NotConnectedError	1005	未连接
UnsupportedOperationError	1006	不支持的操作
UnknownError	1007	未知错误

错误处理示例：

```
lifecycleScope.launch {
    val result = connection.connect(ConnectionParams.USB(device))
    when (result) {
        is Result.Success -> {
            // 连接成功，发送数据
            connection.write(printData)
        }
        is Result.Failure -> {
            when (result.error) {
                is PrinterError.ConnectionError -> {
```

```
Log.e("USB", "Connection error: ${result.error.message}")
// 尝试重新连接
}
is PrinterError.CommunicationError -> {
    Log.e("USB", "Communication error:
${result.error.message}")
    // 检查打印机状态
}
is PrinterError.InvalidParameterError -> {
    Log.e("USB", "Invalid parameter: ${result.error.message}")
    // 修正参数值
}
is PrinterError.TimeoutError -> {
    Log.e("USB", "Timeout error: ${result.error.message}")
    // 增加超时时间或检查打印机状态
}
is PrinterError.NotConnectedError -> {
    Log.e("USB", "Not connected: ${result.error.message}")
    // 先连接打印机
}
is PrinterError.UnsupportedOperationError -> {
    Log.e("USB", "Unsupported operation:
${result.error.message}")
    // 使用替代方案
}
is PrinterError.UnknownError -> {
    Log.e("USB", "Unknown error: ${result.error.message}")
    // 记录错误并尝试恢复
}
}
}
}
```

9.2 返回值类型

类型	说明
Result.Success	成功，包含返回值
Result.Failure	失败，包含错误信息