

ESC 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

```
<!-- 基础权限 -->
<uses-permission android:name="android.permission.INTERNET" />
<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_SCAN" />
```

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

```
// 创建连接
val connection = BluetoothConnection()
connection.connect(ConnectionParams.Bluetooth(address = "00:11:22:33:44:55"))

// 创建 EscPrinter 实例
val printer = EscPrinter(connection)

// 初始化打印机
printer.initialize()
```

2. 接口目录

ESC 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

1.3 混淆配置

1.4 初始化

2. 接口目录

3. initialize

3a. setEncoding

4. printText

5. printBarcode

6. printQRCode

7. printImage

8. feedPaper

9. cutPaper

10. getStatus

11. openCashDrawer

12. setCodePage

13. setCharacterSet

14. printTable

15. setAlignment

16. setFontStyle

17. setFontMultiplier

18. setLineSpacing

19. setLeftMargin

20. setReversePrinting

21. beep

22. selfTest

23. printPDF417

24. setPageMode

25. setPageModePrintArea

- 26. clearPageModeData
- 27. setPageModePrintDirection
- 28. setPageModeAbsolutePosition
- 29. printPageModeData
- 30. printPageModeText
- 31. setStandardMode
- 32. drawPageRectangle
- 33. drawPageLine
- 34. printTwoColorImage
- 35. setAntiSkipPrinting
- 36. setPrintTaskId
- 37. setPrintEndSwitch
- 38. getPrintResult
- 39. cleanMissPrintData
- 40. checkMissPrintEnd
- 41. getPrinterSN
- 42. getPrinterFirmwareVersion
- 43. getPrinterManufacturer
- 44. getPrinterModel
- 45. setMissPrintListener
- 46. stopMissPrintListener
- 47. 附录
 - 47.1 对齐方式枚举 (Alignment)
 - 47.2 条码类型枚举 (BarcodeType)
 - 47.3 抖动算法枚举 (DitheringAlgorithm)
 - 47.4 切纸模式枚举 (CutMode)
 - 47.5 下划线模式枚举 (UnderlineMode)
 - 47.6 压缩类型枚举 (CompressionType)
 - 47.7 条码文本位置枚举 (BarcodeTextPosition)
 - 47.8 错误类型说明

3. initialize

描述：初始化打印机，恢复默认设置。

方法签名：

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

返回值： `Result<Unit>` 初始化结果

示例代码：

```
printer.initialize()
```

3a. setEncoding

描述：设置全局编码。

方法签名：

```
fun setEncoding(encoding: String)
```

参数：

参数名	类型	必填	说明
encoding	String	是	编码名称（如 "GBK", "UTF-8"）

示例代码：

```
printer.setEncoding("UTF-8")
```

4. printText

描述：打印文本。

方法签名：

```
suspend fun printText(  
    text: String,  
    options: TextOptions = TextOptions()  
) : Result<Unit>
```

参数：

参数名	类型	必填	默认值	说明
text	String	是	-	要打印的文本
options	TextOptions	否	TextOptions()	文本选项

TextOptions 属性：

属性名	类型	默认值	说明
bold	Boolean	false	是否加粗
underline	Boolean	false	是否下划线
align	TextAlign	LEFT	对齐方式
fontSmall	Boolean	false	是否小字体（9x17），否则为大字体（12x24）
reverse	Boolean	false	是否反白打印
widthMultiplier	Int	0	宽度倍数（0-8）
heightMultiplier	Int	0	高度倍数（0-8）

示例代码：

```
// 简单打印
printer.printText("Hello World")

// 带选项打印
printer.printText(
    text = "Hello World",
    options = TextOptions(
        bold = true,
        underline = true,
        align = TextAlign.CENTER,
        fontSmall = false
    )
)
```

5. printBarcode

描述：打印条码。

方法签名：

```
suspend fun printBarcode(
    data: String,
    type: BarcodeType = BarcodeType.CODE128,
    options: BarcodeOptions = BarcodeOptions()
): Result<Unit>
```

参数：

参数名	类型	必填	默认值	说明
data	String	是	-	条码数据
type	BarcodeType	否	CODE128	条码类型
options	BarcodeOptions	否	BarcodeOptions()	条码选项

BarcodeOptions 属性：

属性名	类型	默认值	说明
height	Int	50	条码高度
width	Int	2	条码宽度
textPosition	BarcodeTextPosition	BELOW	文本位置
rotation	Int	0	旋转角度（ESC模式下无效）

BarcodeType 枚举值：

值	说明
UPCA	UPC-A
UPCE	UPC-E
EAN13	EAN-13
EAN8	EAN-8
CODE39	Code 39
ITF	ITF
CODABAR	Codabar
CODE93	Code 93
CODE128	Code 128

示例代码：

```
printer.printBarcode(  
    data = "123456789012",  
    type = BarcodeType.CODE128,  
    options = BarcodeOptions(  
        height = 80,  
        width = 2,  
        textPosition = BarcodeTextPosition.BELOW,  
        rotation = 0  
    )  
)
```

6. printQRCode

描述：打印二维码。

方法签名：

```
suspend fun printQRCode(  
    data: String,  
    options: QRCodeOptions = QRCodeOptions()  
) : Result<Unit>
```

参数：

参数名	类型	必填	默认值	说明
data	String	是	-	二维码数据
options	QRCodeOptions	否	QRCodeOptions()	二维码选项

QRCodeOptions 属性：

属性名	类型	默认值	说明
size	Int	6	二维码大小（1-16）
errorCorrection	QRErrorCorrection	M	纠错等级
rotation	Int	0	旋转角度（ESC模式下无效）

QRErrorCorrection 枚举值：

值	说明
L	7% 纠错
M	15% 纠错
Q	25% 纠错
H	30% 纠错

示例代码：

```
printer.printQRCode(  
    data = "https://www.example.com",  
    options = QRCodeOptions(  
        size = 8,  
        errorCorrection = QRErrorCorrection.H,  
        rotation = 0  
    )  
)
```

7. printImage

描述：打印图片。

方法签名（Bitmap）：

```
suspend fun printImage(  
    bitmap: Bitmap,  
    options: ImageOptions = ImageOptions()  
): Result<Unit>
```

参数：

参数名	类型	必填	默认值	说明
bitmap	Bitmap	是	-	要打印的图片
options	ImageOptions	否	ImageOptions()	图片选项

方法签名（字节数组）：

```
suspend fun printImage(  
    imageData: ByteArray,  
    width: Int,  
    height: Int,  
    options: ImageOptions = ImageOptions()  
) : Result<Unit>
```

参数：

参数名	类型	必填	默认值	说明
imageData	ByteArray	是	-	图像数据（单色字节数组）
width	Int	是	-	图像宽度
height	Int	是	-	图像高度
options	ImageOptions	否	ImageOptions()	图片选项

ImageOptions 属性：

属性名	类型	默认值	说明
ditheringAlgorithm	DitheringAlgorithm	DIFFUSION	抖动算法
threshold	Int	-1	二值化阈值（-1为自动计算）
compression	CompressionType	NONE	压缩类型
maxWidth	Int?	null	最大宽度

DitheringAlgorithm 枚举值：

值	说明
THRESHOLD	简单阈值
DIFFUSION	扩散抖动
CLUSTER	聚集抖动

示例代码：

```
val bitmap = BitmapFactory.decodeResource(resources, R.drawable.logo)
printer.printImage(
    bitmap = bitmap,
    options = ImageOptions(
        ditheringAlgorithm = DitheringAlgorithm.DIFFUSION,
        threshold = -1,
        compression = CompressionType.NONE,
        maxWidth = null
    )
)
```

8. feedPaper

描述：走纸。

方法签名：

```
suspend fun feedPaper(lines: Int = 1): Result<Unit>
```

参数：

参数名	类型	必填	默认值	说明
lines	Int	否	1	走纸行数

返回值：`Result<Unit>` 走纸结果

示例代码：

```
printer.feedPaper(5)
```

9. cutPaper

描述：切纸。

方法签名：

```
suspend fun cutPaper(partial: Boolean = false): Result<Unit>
```

参数：

参数名	类型	必填	默认值	说明
partial	Boolean	否	false	是否半切（true为半切，false为全切）

示例代码：

```
printer.cutPaper() // 全切
printer.cutPaper(partial = true) // 半切
```

10. getStatus

描述：获取打印机状态。

方法签名：

```
suspend fun getStatus(): Result<PrinterStatus>
```

返回值： `Result<PrinterStatus>` 打印机状态

PrinterStatus 属性：

属性名	类型	说明
isOnline	Boolean	是否在线
isPaperOut	Boolean	是否缺纸
isPaperNearEnd	Boolean	纸张是否即将用完
isCoverOpen	Boolean	盖板是否打开
isError	Boolean	是否有错误
isCutterError	Boolean	切刀是否有错误
isVoltageError	Boolean	电压是否有错误
isOverheating	Boolean	是否过热
isPrinting	Boolean	是否正在打印（ESC 模式下无效）
isRibbonOut	Boolean	是否缺碳带（ZPL）（ESC 模式下无效）
temperature	Int?	温度（ESC 模式下无效）
batteryLevel	Int?	电池电量（ESC 模式下无效）

示例代码：

```

val result = printer.getStatus()
when (result) {
    is Result.Success -> {
        val status = result.value
        if (status.isPaperOut) {
            Log.w("Printer", "No paper!")
        }
    }
    is Result.Failure -> Log.e("Printer", "Error: ${result.error.message}")
}

```

11. openCashDrawer

描述：打开钱箱。

方法签名：

```
suspend fun openCashDrawer(pin: Int = 0): Result<Unit>
```

参数：

参数名	类型	必填	默认值	说明
pin	Int	否	0	钱箱引脚（0或1）

返回值：`Result<Unit>` 操作结果

示例代码：

```
printer.openCashDrawer()
```

12. setCodePage

描述：设置代码页。

方法签名：

```
suspend fun setCodePage(codePage: ESCCommandGenerator.CodePage): Result<Unit>
```

参数：

参数名	类型	必填	说明
codePage	ESCCommandGenerator.CodePage	是	代码页枚举

示例代码：

```
printer.setCodePage(ESCCommandGenerator.CodePage.UTF_8)
```

13. setCharacterSet

描述：设置字符集。

方法签名：

```
suspend fun setCharacterSet(characterSet: ESCCommandGenerator.CharacterSet):  
Result<Unit>
```

参数：

参数名	类型	必填	说明
characterSet	ESCCommandGenerator.CharacterSet	是	字符集枚举

示例代码：

```
printer.setCharacterSet(ESCCommandGenerator.CharacterSet.USA)
```

14. printTable

描述：打印表格。

方法签名：

```
suspend fun printTable(  
    rows: List<List<ESCCommandGenerator.TableColumn>>  
) : Result<Unit>
```

参数：

参数名	类型	必填	说明
rows	List<List<ESCCommandGenerator.TableColumn>>	是	表格数据，每行是一个列的列表

TableColumn 数据类：

属性名	类型	默认值	说明
text	String	-	列文本内容
width	Int	-	列宽度（字符数）
align	TextAlign	TextAlign.LEFT	文本对齐方式（LEFT、CENTER、RIGHT）

width 参数说明：

- 单位：字符数
- 建议值：
 - 80mm 打印机：总宽度不超过 48 字符
 - 58mm 打印机：总宽度不超过 32 字符
- 注意：所有列的宽度总和不应超过打印机的最大打印宽度

示例代码：

```
// 打印带表头和数据的多行表格
printer.printTable(
    rows = listOf(
        // 表头行
        listOf(
            ESCCommandGenerator.TableColumn("商品", 20, TextAlign.LEFT),
            ESCCommandGenerator.TableColumn("数量", 10, TextAlign.CENTER),
            ESCCommandGenerator.TableColumn("价格", 15, TextAlign.RIGHT)
        ),
        // 数据行 1
        listOf(
            ESCCommandGenerator.TableColumn("苹果", 20, TextAlign.LEFT),
            ESCCommandGenerator.TableColumn("2", 10, TextAlign.CENTER),
            ESCCommandGenerator.TableColumn("10.00", 15, TextAlign.RIGHT)
        ),
        // 数据行 2
        listOf(
            ESCCommandGenerator.TableColumn("香蕉", 20, TextAlign.LEFT),
            ESCCommandGenerator.TableColumn("3", 10, TextAlign.CENTER),
            ESCCommandGenerator.TableColumn("15.00", 15, TextAlign.RIGHT)
        ),
        // 合计行
        listOf(
            ESCCommandGenerator.TableColumn("合计", 20, TextAlign.LEFT),
            ESCCommandGenerator.TableColumn("5", 10, TextAlign.CENTER),
            ESCCommandGenerator.TableColumn("25.00", 15, TextAlign.RIGHT)
        )
    )
)
```

使用注意事项：

- 1. 表格总宽度不应超过打印机的最大打印宽度，否则会导致内容被截断
- 2. 当文本长度超过列宽度时，文本会被截断以适应列宽
- 3. 不同字体大小会影响表格布局，使用较大字体时应相应减少列宽
- 4. 建议使用等宽字体以获得更好的表格对齐效果
- 5. 对于多行表格，确保每行的列数保持一致

15. setAlignment

描述：设置对齐方式。

方法签名：

```
suspend fun setAlignment(align: TextAlign): Result<Unit>
```

参数：

参数名	类型	必填	说明
align	TextAlign	是	对齐方式

TextAlign 枚举值：

值	说明
LEFT	左对齐
CENTER	居中
RIGHT	右对齐

示例代码：

```
printer.setAlignment(TextAlign.CENTER)
printer.printText("居中文本")
```

16. setFontStyle

描述：设置字体样式。

方法签名：

```
suspend fun setFontStyle(
    bold: Boolean = false,
    underline: Boolean = false
): Result<Unit>
```

参数：

参数名	类型	必填	默认值	说明
bold	Boolean	否	false	是否加粗
underline	Boolean	否	false	是否下划线

示例代码：

```
printer.setFontStyle(bold = true, underline = true)
printer.printText("加粗带下划线")
```

17. setFontMultiplier

描述：设置字体倍宽倍高。

方法签名：

```
suspend fun setFontMultiplier(
    widthMultiplier: Int = 0,
    heightMultiplier: Int = 0
): Result<Unit>
```

参数：

参数名	类型	必填	默认值	说明
widthMultiplier	Int	否	0	宽度倍数（0-8）
heightMultiplier	Int	否	0	高度倍数（0-8）

示例代码：

```
printer.setFontMultiplier(widthMultiplier = 2, heightMultiplier = 2)
printer.printText("放大文本")
```

18. setLineSpacing

描述：设置行间距。

方法签名：

```
suspend fun setLineSpacing(spacing: Int): Result<Unit>
```

参数：

参数名	类型	必填	说明
spacing	Int	是	行间距（点数，默认30）

示例代码：

```
printer.setLineSpacing(20)
```

19. setLeftMargin

描述：设置左边距。

方法签名：

```
suspend fun setLeftMargin(margin: Int): Result<Unit>
```

参数：

参数名	类型	必填	说明
margin	Int	是	左边距（点数）

示例代码：

```
printer.setLeftMargin(50)
```

20. setReversePrinting

描述：设置反白打印。

方法签名：

```
suspend fun setReversePrinting(enable: Boolean): Result<Unit>
```

参数：

参数名	类型	必填	说明
enable	Boolean	是	是否启用反白

示例代码：

```
printer.setReversePrinting(true)
printer.printText("反白文本")
printer.setReversePrinting(false)
```

21. beep

描述：蜂鸣。

方法签名：

```
suspend fun beep(times: Int = 1, t1: Int = 1, t2: Int = 1): Result<Unit>
```

参数：

参数名	类型	必填	默认值	说明
times	Int	否	1	蜂鸣次数
t1	Int	否	1	ON 时间（毫秒）
t2	Int	否	1	OFF 时间（毫秒）

示例代码：

```
printer.beep(times = 3, t1 = 200, t2 = 100)
```

22. selfTest

描述：打印自检页。

方法签名：

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

返回值：`Result<Unit>` 操作结果

示例代码：

```
printer.selfTest()
```

23. printPDF417

描述：打印 PDF417 二维码。

方法签名：

```
suspend fun printPDF417(  
    data: String,  
    columns: Int = 0,  
    rows: Int = 0,  
    width: Int = 3,  
    height: Int = 3,  
    errorCorrection: Int = 1  
): Result<Unit>
```

参数：

参数名	类型	必填	默认值	说明
data	String	是	-	二维码数据
columns	Int	否	0	列数（0为自动）
rows	Int	否	0	行数（0为自动）
width	Int	否	3	模块宽度
height	Int	否	3	模块高度
errorCorrection	Int	否	1	纠错等级（0-8）

示例代码：

```
printer.printPDF417(  
    data = "PDF417 Data",  
    columns = 0,  
    rows = 0,  
    width = 3,  
    height = 3,  
    errorCorrection = 1  
)
```

24. setPageMode

描述：设置页面模式。

方法签名：

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

示例代码：

```
printer.setPageMode()
```

25. setPageModePrintArea

描述：设置页面模式打印区域。

方法签名：

```
suspend fun setPageModePrintArea(  
    horizontal: Int,  
    vertical: Int,  
    width: Int,  
    height: Int  
): Result<Unit>
```

参数：

参数名	类型	必填	说明
horizontal	Int	是	水平位置（点）
vertical	Int	是	垂直位置（点）
width	Int	是	宽度（点）
height	Int	是	高度（点）

示例代码：

```
printer.setPageModePrintArea(0, 0, 576, 800)
```

26. clearPageModeData

描述：清除页面模式数据。

方法签名：

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

示例代码：

```
printer.clearPageModeData()
```

27. setPageModePrintDirection

描述：设置页面模式打印方向。

方法签名：

```
suspend fun setPageModePrintDirection(  
    direction: ESCCommandGenerator.PageModeDirection  
): Result<Unit>
```

参数：

参数名	类型	必填	说明
direction	PageModeDirection	是	打印方向

PageModeDirection 枚举值：

值	说明
LEFT_TO_RIGHT	从左到右
BOTTOM_TO_TOP	从下到上
RIGHT_TO_LEFT	从右到左
TOP_TO_BOTTOM	从上到下

示例代码：

```
printer.setPageModePrintDirection(ESCCommandGenerator.PageModeDirection.LEFT_T  
O_RIGHT)
```

28. setPageModeAbsolutePosition

描述：设置页面模式绝对位置。

方法签名：

```
suspend fun setPageModeAbsolutePosition(  
    xPosition: Int,  
    yPosition: Int  
): Result<Unit>
```

参数：

参数名	类型	必填	说明
xPosition	Int	是	X坐标（点），原点在左上角
yPosition	Int	是	Y坐标（点），原点在左上角

示例代码：

```
printer.setPageModeAbsolutePosition(100, 200)
```

29. printPageModeData

描述：打印页面模式数据。

方法签名：

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

示例代码：

```
printer.printPageModeData()
```

30. printPageModeText

描述：页面模式打印文本。

方法签名：

```
suspend fun printPageModeText(  
    text: String,  
    encoding: String = Companion.encoding  
) : Result<Unit>
```

参数：

参数名	类型	必填	默认值	说明
text	String	是	-	要打印的文本
encoding	String	否	全局encoding	编码

示例代码：

```
printer.printPageModeText("Hello World")
```

31. setStandardMode

描述：设置标准模式。

方法签名：

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

示例代码：

```
printer.setStandardMode()
```

32. drawPageRectangle

描述：绘制页面矩形。

方法签名：

```
suspend fun drawPageRectangle(  
    x: Int,  
    y: Int,  
    width: Int,  
    height: Int,  
    lineWidth: Int  
): Result<Unit>
```

参数：

参数名	类型	必填	说明
x	Int	是	X坐标（点），原点在左上角
y	Int	是	Y坐标（点），原点在左上角
width	Int	是	宽度（点）
height	Int	是	高度（点）
lineWidth	Int	是	线宽（点），范围：0-255

示例代码：

```
printer.drawPageRectangle(50, 50, 200, 100, 2)
```

33. drawPageLine

描述：绘制页面线条。

方法签名：

```
suspend fun drawPageLine(  
    x1: Int,  
    y1: Int,  
    x2: Int,  
    y2: Int,  
    lineWidth: Int  
): Result<Unit>
```

参数：

参数名	类型	必填	说明
x1	Int	是	起点X坐标（点），原点在左上角
y1	Int	是	起点Y坐标（点），原点在左上角
x2	Int	是	终点X坐标（点），原点在左上角
y2	Int	是	终点Y坐标（点），原点在左上角
lineWidth	Int	是	线宽（点），范围：0-255

示例代码：

```
printer.drawPageLine(50, 50, 200, 200, 2)
```

34. printTwoColorImage

描述：打印双色图片。

方法签名：

```
suspend fun printTwoColorImage(  
    blackImageData: ByteArray,  
    redImageData: ByteArray,  
    width: Int,  
    height: Int  
): Result<Unit>
```

参数：

参数名	类型	必填	说明
blackImageData	ByteArray	是	黑色图像数据
redImageData	ByteArray	是	红色图像数据
width	Int	是	宽度（点）
height	Int	是	高度（点）

方法签名（Bitmap）：

```
suspend fun printTwoColorImage(  
    blackBitmap: Bitmap,  
    redBitmap: Bitmap,  
    options: ImageOptions = ImageOptions()  
): Result<Unit>
```

示例代码：

```
printer.printTwoColorImage(blackBitmap, redBitmap)
```

35. setAntiSkipPrinting

描述：设置防跳印。

方法签名：

```
suspend fun setAntiSkipPrinting(enable: Boolean): Result<Unit>
```

参数：

参数名	类型	必填	说明
enable	Boolean	是	是否启用防跳印

示例代码：

```
printer.setAntiSkipPrinting(true)
```

36. setPrintTaskId

描述：设置打印任务ID。

方法签名：

```
suspend fun setPrintTaskId(taskId: Int): Result<Unit>
```

参数：

参数名	类型	必填	说明
taskId	Int	是	任务ID（1-9999）

示例代码：

```
printer.setPrintTaskId(1001)
```

37. setPrintEndSwitch

描述：设置打印结束开关。

方法签名：

```
suspend fun setPrintEndSwitch(enable: Boolean): Result<Unit>
```

参数：

参数名	类型	必填	说明
enable	Boolean	是	是否启用打印结束开关

示例代码：

```
printer.setPrintEndSwitch(true)
```

38. getPrintResult

描述：获取打印结果状态。

方法签名：

```
suspend fun getPrintResult(timeout: Long): Result<Int>
```

参数：

参数名	类型	必填	默认值	说明
timeout	Long	否	30000	超时时间（毫秒）

返回值: `Result<Int>` 返回打印结果状态码 (0表示成功, -2表示超时)

示例代码:

```
val result = printer.getPrintResult(30000)
when (result) {
    is Result.Success -> {
        when (result.value) {
            0 -> Log.d("Printer", "Print successful")
            -2 -> Log.w("Printer", "Print timeout")
            else -> Log.w("Printer", "Print result: ${result.value}")
        }
    }
    is Result.Failure -> Log.e("Printer", "Error: ${result.error.message}")
}
```

39. cleanMissPrintData

描述: 清除欠打印数据。

方法签名:

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

示例代码:

```
printer.cleanMissPrintData()
```

40. checkMissPrintEnd

描述: 检查欠打印结束。

方法签名:

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

示例代码:

```
printer.checkMissPrintEnd()
```

41. getPrinterSN

描述: 获取打印机序列号。

方法签名：

```
suspend fun getPrinterSN(): Result<String>
```

返回值： `Result<String>` 打印机序列号

示例代码：

```
val result = printer.getPrinterSN()
when (result) {
    is Result.Success -> Log.d("Printer", "SN: ${result.value}")
    is Result.Failure -> Log.e("Printer", "Error: ${result.error.message}")
}
```

42. getPrinterFirmwareVersion

描述：获取打印机固件版本。

方法签名：

```
suspend fun getPrinterFirmwareVersion(): Result<String>
```

返回值： `Result<String>` 固件版本

示例代码：

```
val result = printer.getPrinterFirmwareVersion()
when (result) {
    is Result.Success -> Log.d("Printer", "Firmware: ${result.value}")
    is Result.Failure -> Log.e("Printer", "Error: ${result.error.message}")
}
```

43. getPrinterManufacturer

描述：获取打印机制造商。

方法签名：

```
suspend fun getPrinterManufacturer(): Result<String>
```

返回值： `Result<String>` 制造商名称

示例代码：

```
val result = printer.getPrinterManufacturer()
when (result) {
    is Result.Success -> Log.d("Printer", "Manufacturer: ${result.value}")
    is Result.Failure -> Log.e("Printer", "Error: ${result.error.message}")
}
```

44. getPrinterModel

描述：获取打印机型号。

方法签名：

```
suspend fun getPrinterModel(): Result<String>
```

返回值：`Result<String>` 型号

示例代码：

```
val result = printer.getPrinterModel()
when (result) {
    is Result.Success -> Log.d("Printer", "Model: ${result.value}")
    is Result.Failure -> Log.e("Printer", "Error: ${result.error.message}")
}
```

45. setMissPrintListener

描述：设置欠打印监听器。

方法签名：

```
fun setMissPrintListener(listener: MissPrintListener?)
```

参数：

参数名	类型	必填	说明
listener	MissPrintListener?	是	欠打印监听器

MissPrintListener 事件类型：

事件	说明
FINISH	打印完成
ERROR	错误
END	结束
CLEAN	清除

示例代码：

```
printer.setMissPrintListener(object : MissPrintListener {
    override fun onEvent(event: MissPrintEvent, taskId: Int?) {
        when (event) {
            MissPrintEvent.FINISH -> Log.d("Printer", "Print finished, taskId: $taskId")
            MissPrintEvent.ERROR -> Log.e("Printer", "Print error")
            MissPrintEvent.END -> Log.d("Printer", "Print end")
            MissPrintEvent.CLEAN -> Log.d("Printer", "Print clean")
        }
    }
})
```

46. stopMissPrintListener

描述：停止欠打印监听器。

方法签名：

```
fun stopMissPrintListener()
```

示例代码：

```
printer.stopMissPrintListener()
```

47. 附录

47.1 对齐方式枚举 (Alignment)

值	说明
LEFT	左对齐
CENTER	居中
RIGHT	右对齐

47.2 条码类型枚举 (BarcodeType)

值	说明
UPC_A	UPC-A
UPC_E	UPC-E
EAN13	EAN-13
EAN8	EAN-8
CODE39	Code 39
ITF	ITF
CODABAR	Codabar
CODE93	Code 93
CODE128	Code 128

47.3 抖动算法枚举 (DitheringAlgorithm)

值	说明
THRESHOLD	简单阈值
DIFFUSION	扩散抖动
CLUSTER	聚集抖动

47.4 切纸模式枚举 (CutMode)

值	说明
FULL	全切
PARTIAL	半切

47.5 下划线模式枚举 (UnderlineMode)

值	说明
NONE	无下划线
SINGLE	单线
DOUBLE	双线

47.6 压缩类型枚举 (CompressionType)

值	说明
NONE	无压缩
LZO	LZO 压缩

47.7 条码文本位置枚举 (BarcodeTextPosition)

值	说明
NONE	无文本
ABOVE	文本在上方
BELOW	文本在下方
BOTH	文本在上下方

47.8 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：

```
val result = printer.printText("Hello, World!")
when (result) {
    is Result.Success -> {
        Log.d("Printer", "Print successful")
    }
    is Result.Failure -> {
        when (result.error) {
            is PrinterError.ConnectionError -> {
                Log.e("Printer", "Connection error: ${result.error.message}")
                // 尝试重新连接
            }
            is PrinterError.CommunicationError -> {
                Log.e("Printer", "Communication error:
${result.error.message}")
                // 检查打印机状态
            }
            is PrinterError.InvalidParameterError -> {
                Log.e("Printer", "Invalid parameter: ${result.error.message}")
                // 修正参数值
            }
            is PrinterError.TimeoutError -> {
                Log.e("Printer", "Timeout error: ${result.error.message}")
                // 增加超时时间或检查打印机状态
            }
            is PrinterError.NotConnectedError -> {
                Log.e("Printer", "Not connected: ${result.error.message}")
                // 先连接打印机
            }
        }
    }
}
```

```
        is PrinterError.UnsupportedOperationError -> {
            Log.e("Printer", "Unsupported operation:
${result.error.message}")
            // 使用替代方案
        }
        is PrinterError.UnknownError -> {
            Log.e("Printer", "Unknown error: ${result.error.message}")
            // 记录错误并尝试恢复
        }
    }
}
}
```