

CPCL 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 CpclPrinter 实例
val printer = CpclPrinter(connection)
```

2. 接口目录

CPCL 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. printText

4. printBarcode

5. printQRCode

6. printImage

7. print

8. getStatus

9. setPrintArea

10. setBold

11. backFeed

12. form

13. setAlign

14. setContrast

15. setSpeed

16. drawBox

17. drawLine

18. drawInverseLine

19. setMagnification

20. beep

21. setPageWidth

22. printReverse

23. setLayer

24. saveForm

25. loadForm

26. deleteForm

27. selfTest

28. printTextWithFont

29. printTextBlock

- 30. printBackgroundText
- 31. printPDF417
- 32. cutPaper
- 33. setCountry
- 34. setUnderline
- 35. openEndStatus
- 36. setTaskId
- 37. getEndStatus
- 38. getPrintResult
- 39. getPrintResultWithoutId
- 40. printCodepageTextCPCL
- 41. printTextCPCL
- 42. autCenter
- 43. autLinePlus
- 44. autLine
- 45. clear
- 46. setEncoding
- 47. 附录
 - 47.1 字体枚举 (CpclFont)
 - 47.2 旋转角度枚举 (Rotation)
 - 47.3 条码类型枚举 (BarcodeType)
 - 47.4 抖动算法枚举 (DitheringAlgorithm)
 - 47.5 错误类型说明

3. printText

描述：打印文本。

方法签名：

```
suspend fun printText(  
    text: String,  
    x: Int = 0,  
    y: Int = 0,  
    options: CpclTextOptions = CpclTextOptions()  
): 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
x	Int	否	0	X坐标（点）
y	Int	否	0	Y坐标（点）
options	CpclTextOptions	否	CpclTextOptions()	文本选项

CpclTextOptions 属性：

属性名	类型	默认值	说明
fontSize	Int	24	字体大小（CPCL字体ID：0,1,2,3,4,8,16,24,48,55），详细说明见下方
bold	Int	0	加粗强度（0=关闭, 1-5=加粗强度）

fontSize 详细说明：

值	说明
0	24x24或12x24，视中英文而定（泰语：24x48）
1	7x19（英文），24x24（繁体）
2	8x8
3	20x20或10x20，视中英文而定
4	32x32或16x32，由ID3字体宽高各放大2倍
8	24x24或12x24，视中英文而定
16	16x16或8x16，视中英文而定
24	24x24或12x24，视中英文而定
48	48x48或24x48，视中英文而定
55	16x16或8x16，视中英文而定

示例代码：

```
printer.printText(  
    text = "Hello CPCL",  
    x = 10,  
    y = 10,  
    options = CpclTextOptions(  
        fontSize = 24,  
        bold = 3  
    )  
)
```

4. printBarcode

描述：打印条码。

方法签名：

```
suspend fun printBarcode(  
    data: String,  
    type: BarcodeType,  
    x: Int = 0,  
    y: Int = 0,  
    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	是	-	条码类型
x	Int	否	0	X坐标（点）
y	Int	否	0	Y坐标（点）
options	BarcodeOptions	否	BarcodeOptions()	条码选项

BarcodeOptions 属性：

属性名	类型	默认值	说明
width	Int	2	条码宽度（1-6）
height	Int	60	条码高度（点）
showText	Boolean	true	是否显示条码下方文字
textAbove	Boolean	false	文字是否在条码上方
rotation	Int	0	旋转角度（0/90/180/270）

示例代码：

```
printer.printBarcode(  
    data = "1234567890",  
    type = BarcodeType.CODE128,  
    x = 10,  
    y = 50,  
    options = BarcodeOptions(  
        width = 2,  
        height = 80,  
        showText = true  
    )  
)
```

5. printQRCode

描述：打印二维码。

方法签名：

```
suspend fun printQRCode(  
    x: Int,  
    y: Int,  
    data: String,  
    size: Int = 6,  
    options: QRCodeOptions = QRCodeOptions()  
): Result<Unit>
```

参数：

参数名	类型	必填	默认值	说明
x	Int	是	-	X坐标（点）
y	Int	是	-	Y坐标（点）
data	String	是	-	二维码数据
size	Int	否	6	二维码大小（1-16）
options	QRCodeOptions	否	QRCodeOptions()	二维码选项

QRCodeOptions 属性：

属性名	类型	默认值	说明
errorCorrection	ErrorCorrectionLevel	M	纠错等级
rotation	Rotation	NORMAL	旋转角度

示例代码：

```
printer.printQRCode(  
    x = 100,  
    y = 100,  
    data = "https://www.example.com",  
    size = 8,  
    options = QRCodeOptions(errorCorrection = ErrorCorrectionLevel.H)  
)
```

6. printImage

描述：打印图片。

方法签名：

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

参数：

参数名	类型	必填	默认值	说明
x	Int	是	-	X坐标（点）
y	Int	是	-	Y坐标（点）
bitmap	Bitmap	是	-	要打印的图片
options	ImageOptions	否	ImageOptions()	图片选项

ImageOptions 属性：

属性名	类型	默认值	说明
threshold	Int	-1	二值化阈值（-1为自动计算）
algorithm	DitheringAlgorithm	DIFFUSION	抖动算法

示例代码：

```
val bitmap = BitmapFactory.decodeResource(resources, R.drawable.logo)
printer.printImage(
    x = 10,
    y = 200,
    bitmap = bitmap,
    options = ImageOptions(algorithm = DitheringAlgorithm.DIFFUSION)
)
```

7. print

描述：执行打印。

方法签名：

```
suspend fun print(copies: Int = 1): Result<Unit>
```

参数：

参数名	类型	必填	默认值	说明
copies	Int	否	1	打印份数

返回值： `Result<Unit>` 打印结果

示例代码：

```
// 添加打印内容后执行打印
printer.printText(10, 10, "Label Content")
printer.printQRCode(100, 50, "QR Data")
printer.print(copies = 1)
```

8. 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

示例代码：

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

9. setPrintArea

描述：设置打印区域。

方法签名：

```
suspend fun setPrintArea(width: Int, height: Int): Result<Unit>
```

参数：

参数名	类型	必填	说明
width	Int	是	打印区域宽度（点）
height	Int	是	打印区域高度（点）

示例代码：

```
printer.setPrintArea(width = 576, height = 400)
```

10. setBold

描述：设置加粗。

方法签名：

```
suspend fun setBold(level: Int): Result<Unit>
```

参数：

参数名	类型	必填	说明
level	Int	是	加粗程度（0-5），0表示关闭加粗，1-5表示加粗强度

示例代码：

```
printer.setBold(3)
printer.printText(10, 10, "Bold Text")
```

11. backFeed

描述：退纸。

方法签名：

```
suspend fun backFeed(length: Int): Result<Unit>
```

参数：

参数名	类型	必填	说明
length	Int	是	退纸长度（点）

示例代码：

```
printer.backFeed(100)
```

12. form

描述：走纸到撕纸位。

方法签名：

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

返回值： `Result<Unit>` 操作结果

示例代码：

```
printer.form()
```

13. setAlign

描述： 设置对齐方式。

方法签名：

```
suspend fun setAlign(alignment: Alignment): Result<Unit>
```

参数：

参数名	类型	必填	说明
alignment	Alignment	是	对齐方式（LEFT/CENTER/RIGHT）

示例代码：

```
printer.setAlign(Alignment.CENTER)
```

14. setContrast

描述： 设置打印浓度。

方法签名：

```
suspend fun setContrast(contrast: Int): Result<Unit>
```

参数：

参数名	类型	必填	说明
contrast	Int	是	浓度值（0-3）

示例代码：

```
printer.setContrast(2)
```

15. setSpeed

描述：设置打印速度。

方法签名：

```
suspend fun setSpeed(speed: Int): Result<Unit>
```

参数：

参数名	类型	必填	说明
speed	Int	是	速度值（0-3）

示例代码：

```
printer.setSpeed(2)
```

16. drawBox

描述：绘制方框。

方法签名：

```
suspend fun drawBox(  
    x0: Int,  
    y0: Int,  
    x1: Int,  
    y1: Int,  
    thickness: Int  
): Result<Unit>
```

参数：

参数名	类型	必填	说明
x0	Int	是	左上角X坐标（点）
y0	Int	是	左上角Y坐标（点）
x1	Int	是	右下角X坐标（点）
y1	Int	是	右下角Y坐标（点）
thickness	Int	是	线宽（点）

示例代码：

```
printer.drawBox(x0 = 10, y0 = 10, x1 = 210, y1 = 110, thickness = 2)
```

17. drawLine

描述：绘制线条。

方法签名：

```
suspend fun drawLine(  
    x0: Int,  
    y0: Int,  
    x1: Int,  
    y1: Int,  
    thickness: Int  
): Result<Unit>
```

参数：

参数名	类型	必填	说明
x0	Int	是	起点X坐标（点）
y0	Int	是	起点Y坐标（点）
x1	Int	是	终点X坐标（点）
y1	Int	是	终点Y坐标（点）
thickness	Int	是	线宽（点）

示例代码：

```
printer.drawLine(x0 = 10, y0 = 50, x1 = 300, y1 = 50, thickness = 1)
```

18. drawInverseLine

描述：绘制反色线条（用于在已打印内容上做反白处理）。

方法签名：

```
suspend fun drawInverseLine(
    x0: Int,
    y0: Int,
    x1: Int,
    y1: Int,
    thickness: Int
): Result<Unit>
```

参数：

参数名	类型	必填	说明
x0	Int	是	起点X坐标（点）
y0	Int	是	起点Y坐标（点）
x1	Int	是	终点X坐标（点）
y1	Int	是	终点Y坐标（点）
thickness	Int	是	线宽（点）

示例代码：

```
printer.drawInverseLine(x0 = 10, y0 = 50, x1 = 300, y1 = 50, thickness = 2)
```

19. setMagnification

描述：设置字体放大倍数。

方法签名：

```
suspend fun setMagnification(width: Int, height: Int): Result<Unit>
```

参数：

参数名	类型	必填	说明
width	Int	是	宽度倍数（1-8）
height	Int	是	高度倍数（1-8）

示例代码：

```
printer.setMagnification(width = 2, height = 2)
```

20. beep

描述：蜂鸣。

方法签名：

```
suspend fun beep(times: 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

示例代码：

```
printer.beep(times = 3)
```

21. setPageWidth

描述：设置页宽。

方法签名：

```
suspend fun setPageWidth(width: Int): Result<Unit>
```

参数：

参数名	类型	必填	说明
width	Int	是	页宽（点）

示例代码：

```
printer.setPageWidth(576)
```

22. printReverse

描述：反白打印（反转已打印内容区域，需先打印内容再调用）。

方法签名：

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

返回值：Result<Unit> 操作结果

示例代码：

```
printer.printReverse()
```

23. setLayer

描述：设置图层。

方法签名：

```
suspend fun setLayer(layer: Int): Result<Unit>
```

参数：

参数名	类型	必填	说明
layer	Int	是	图层编号

示例代码：

```
printer.setLayer(1)
```

24. saveForm

描述：保存表单。

方法签名：

```
suspend fun saveForm(formName: String): Result<Unit>
```

参数：

参数名	类型	必填	说明
formName	String	是	表单名称

示例代码：

```
printer.saveForm(formName = "myForm")
```

25. loadForm

描述：加载表单。

方法签名：

```
suspend fun loadForm(formName: String): Result<Unit>
```

参数：

参数名	类型	必填	说明
formName	String	是	表单名称

示例代码：

```
printer.loadForm(formName = "myForm")
```

26. deleteForm

描述：删除表单。

方法签名：

```
suspend fun deleteForm(formName: String): Result<Unit>
```

参数：

参数名	类型	必填	说明
formName	String	是	表单名称

示例代码：

```
printer.deleteForm(formName = "myForm")
```

27. 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()
```

28. printTextWithFont

描述：使用指定字体打印文本。

方法签名：

```
suspend fun printTextWithFont(  
    text: String,  
    fontName: String,  
    xScale: Int,  
    yScale: Int,  
    x: Int,  
    y: Int,  
    encoding: String = "GBK"  
): 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
fontName	String	是	-	字体名称
xScale	Int	是	-	横向缩放倍数
yScale	Int	是	-	纵向缩放倍数
x	Int	是	-	X坐标（点）
y	Int	是	-	Y坐标（点）
encoding	String	否	GBK	文本编码

示例代码：

```
printer.printTextWithFont(  
    text = "Hello",  
    fontName = "001",  
    xScale = 1,  
    yScale = 1,  
    x = 10,  
    y = 10  
)
```

29. printTextBlock

描述：打印文本块（按指定宽度自动换行）。

方法签名：

```
suspend fun printTextBlock(  
    text: String,  
    x: Int,  
    y: Int,  
    width: Int,  
    font: String = "0",  
    size: Int = 8,  
    lineHeight: Int = 0,  
    encoding: String = "GBK"  
) : 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
x	Int	是	-	X坐标（点）
y	Int	是	-	Y坐标（点）
width	Int	是	-	文本块宽度（点）
font	String	否	"0"	字体名称，如 "0" 表示默认字体
size	Int	否	8	字号（点）
lineSpace	Int	否	0	行间距（点）
encoding	String	否	GBK	文本编码

示例代码：

```
printer.printTextBlock(  
    text = "Hello World",  
    x = 10,  
    y = 10,  
    width = 200  
)
```

30. printBackgroundText

描述：打印带背景的文本。

方法签名：

```
suspend fun printBackgroundText(  
    text: String,  
    x: Int,  
    y: Int,  
    size: Int,  
    background: Int,  
    encoding: String = "GBK"  
) : 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
x	Int	是	-	X坐标（点）
y	Int	是	-	Y坐标（点）
size	Int	是	-	字号
background	Int	是	-	背景模式（0=正常, 1=反白）
encoding	String	否	GBK	文本编码

示例代码：

```
printer.printBackgroundText(  
    text = "Hello",  
    x = 10,  
    y = 10,  
    size = 24,  
    background = 0  
)
```

31. printPDF417

描述：打印 PDF417 二维码。

方法签名：

```
suspend fun printPDF417(  
    x: Int,  
    y: Int,  
    data: String,  
    options: PDF417Options = PDF417Options()  
): Result<Unit>
```

参数：

参数名	类型	必填	默认值	说明
x	Int	是	-	X坐标
y	Int	是	-	Y坐标
data	String	是	-	二维码数据
options	PDF417Options	否	PDF417Options()	PDF417选项

示例代码：

```
printer.printPDF417(  
    x = 10,  
    y = 100,  
    data = "PDF417 Data"  
)
```

32. 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	是否部分切纸

返回值：`Result<Unit>` 操作结果

示例代码：

```
// 完全切纸
printer.cutPaper()

// 部分切纸
printer.cutPaper(partial = true)
```

33. setCountry

描述：设置国家/地区。

方法签名：

```
suspend fun setCountry(country: String): Result<Unit>
```

参数：

参数名	类型	必填	说明
country	String	是	国家/地区代码

支持的国家/地区代码：

代码	国家/地区
USA	美国
UK	英国
Germany	德国
France	法国
Sweden	瑞典
Norway	挪威
Italy	意大利
Spain	西班牙
Denmark	丹麦
Belgium	比利时
Portugal	葡萄牙
Japan	日本
Korea	韩国
China	中国
Taiwan	台湾
HongKong	香港

示例代码：

```
printer.setCountry(country = "USA")
```

34. setUnderline

描述：设置下划线。

方法签名：

```
suspend fun setUnderline(enabled: Boolean): Result<Unit>
```

参数：

参数名	类型	必填	说明
enabled	Boolean	是	是否启用下划线

示例代码：

```
printer.setUnderline(enabled = true)
```

35. openEndStatus

描述：开启结束状态。

方法签名：

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

参数：

参数名	类型	必填	说明
enable	Boolean	是	是否开启结束状态

示例代码：

```
printer.openEndStatus(enable = true)
```

36. setTaskId

描述：设置任务ID。

方法签名：

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

参数：

参数名	类型	必填	说明
taskId	Int	是	任务ID

示例代码：

```
printer.setTaskId(taskId = 123)
```

37. getEndStatus

描述：获取结束状态。

方法签名：

```
suspend fun getEndStatus(timeout: Long = 2000): Result<Int>
```

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<Int>` 状态码（0=成功, 非0=失败）

示例代码：

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

38. getPrintResult

描述：获取打印结果状态。

方法签名：

```
suspend fun getPrintResult(timeout: Long = 2000): Result<PrintTaskResult>
```

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<PrintTaskResult>` 打印任务结果

示例代码：

```

val result = printer.getPrintResult()
when (result) {
    is Result.Success -> {
        val printResult = result.value
        Log.d("Printer", "Print result: taskId=${printResult.taskId},
status=${printResult.status}")
    }
    is Result.Failure -> Log.e("Printer", "Error: ${result.error.message}")
}

```

39. getPrintResultWithoutId

描述：获取无ID的打印结果。

方法签名：

```
suspend fun getPrintResultWithoutId(timeout: Long = 2000): Result<Int>
```

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<Int>` 打印结果状态码（0=成功, 非0=失败）

示例代码：

```

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

```

40. printCodepageTextCPCL

描述：打印代码页文本。

方法签名：

```
suspend fun printCodepageTextCPCL(
    text: String,
    x: Int,
    y: Int,
    font: Int,
    flags: Int,
    rotation: CpclTextRotation = CpclTextRotation.NORMAL,
    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
x	Int	是	-	X坐标
y	Int	是	-	Y坐标
font	Int	是	-	字体
flags	Int	是	-	标志位，按位组合： bit0：加粗 bit1：反色 bit2：倍宽 bit3：倍高
rotation	CpclTextRotation	否	CpclTextRotation.NORMAL	旋转角度
encoding	String	否	Companion.encoding	编码

示例代码：

```
printer.printCodepageTextCPCL(
    text = "Hello",
    x = 10,
    y = 10,
    font = 0,
    flags = 0
)
```

41. printTextCPCL

描述：打印CPCL文本。

方法签名：

```
suspend fun printTextCPCL(
    text: String,
    x: Int,
    y: Int,
    font: Int,
    flags: Int,
    isCenter: Boolean,
    width: Int,
    rotation: CpclTextRotation = CpclTextRotation.NORMAL,
    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
x	Int	是	-	X坐标
y	Int	是	-	Y坐标
font	Int	是	-	字体
flags	Int	是	-	标志位，按位组合： bit0：加粗 bit1：反色 bit2：倍宽 bit3：倍高
isCenter	Boolean	是	-	是否居中
width	Int	是	-	宽度
rotation	CpclTextRotation	否	CpclTextRotation.NORMAL	旋转角度
encoding	String	否	Companion.encoding	编码

示例代码：

```
printer.printTextCPCL(
    text = "Hello CPCL",
    x = 10,
    y = 10,
    font = 0,
    flags = 0,
    isCenter = true,
    width = 200
)
```

42. autCenter

描述：自动居中打印。

方法签名：

```
suspend fun autCenter(  
    text: String,  
    x: Int,  
    y: Int,  
    width: Int,  
    size: Int,  
    isDouble: Boolean,  
    rotation: CpclTextRotation = CpclTextRotation.NORMAL,  
    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
x	Int	是	-	X坐标
y	Int	是	-	Y坐标
width	Int	是	-	宽度
size	Int	是	-	字体大小
isDouble	Boolean	是	-	是否双倍大小
rotation	CpclTextRotation	否	CpclTextRotation.NORMAL	旋转角度
encoding	String	否	Companion.encoding	编码

示例代码：

```
printer.autCenter(  
    text = "Centered Text",  
    x = 0,  
    y = 10,  
    width = 576,  
    size = 8,  
    isDouble = false  
)
```

43. autLinePlus

描述：自动行打印增强版。

方法签名：

```
suspend fun autLinePlus(
    x: Int,
    y: Int,
    width: Int,
    size: Int,
    isBold: Boolean,
    isDouble: Boolean,
    text: String,
    encoding: String = Companion.encoding
): Result<Unit>
```

参数：

参数名	类型	必填	默认值	说明
x	Int	是	-	X坐标
y	Int	是	-	Y坐标
width	Int	是	-	宽度
size	Int	是	-	字体大小
isBold	Boolean	是	-	是否加粗
isDouble	Boolean	是	-	是否双倍大小
text	String	是	-	要打印的文本
encoding	String	否	Companion.encoding	编码

示例代码：

```
printer.autLinePlus(
    x = 10,
    y = 10,
    width = 200,
    size = 8,
    isBold = true,
    isDouble = false,
    text = "Auto Line Plus"
)
```

44. autLine

描述：自动行打印。

方法签名：

```
suspend fun autLine(
    x: Int,
    y: Int,
    width: Int,
    size: Int,
    isBold: Boolean,
    isDouble: Boolean,
    text: String,
    encoding: String = Companion.encoding
): Result<Unit>
```

参数：

参数名	类型	必填	默认值	说明
x	Int	是	-	X坐标
y	Int	是	-	Y坐标
width	Int	是	-	宽度
size	Int	是	-	字体大小
isBold	Boolean	是	-	是否加粗
isDouble	Boolean	是	-	是否双倍大小
text	String	是	-	要打印的文本
encoding	String	否	Companion.encoding	编码

示例代码：

```
printer.autLine(
    x = 10,
    y = 10,
    width = 200,
    size = 8,
    isBold = false,
    isDouble = false,
    text = "Auto Line"
)
```

45. clear

描述：清除打印缓冲区。

方法签名：

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

返回值： `Result<Unit>` 操作结果

示例代码：

```
printer.clear()
```

46. setEncoding

描述： 设置编码。

方法签名：

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

参数：

参数名	类型	必填	说明
encoding	String	是	编码格式

示例代码：

```
printer.setEncoding("GBK")
```

47. 附录

47.1 字体枚举 (CpclFont)

值	说明
FONT_A	字体A (12x24)
FONT_B	字体B (9x17)
FONT_C	字体C (9x24)

47.2 旋转角度枚举 (Rotation)

值	说明
NORMAL	正常（0度）
ROTATE_90	旋转90度
ROTATE_180	旋转180度
ROTATE_270	旋转270度

47.3 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
CODE93	Code 93
CODE128	Code 128
CODABAR	Codabar
ITF	ITF

47.4 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

47.5 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 `UXPErrCode.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《UXPErrCode 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
    }
}
}
}
```