

ESC SDK API Documentation

Document Revision History

No.	Version	Changes	Author	Reviewer	Date
1	v1.0.0	Initial version	Qiu Xian	Lin Dechun	2026.04.11
2	v1.0.1	Add UXPErrrorCode error code description, PrinterError gains errorCode field	Qiu Xian	Lin Dechun	2026.08.04

1. Getting Started

1.1 Project Configuration

Add the following 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')
```

1.2 Permission Configuration

```
<!-- Basic permissions -->
<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 ProGuard Configuration

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

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

1.4 Initialization

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

// Create EscPrinter instance
val printer = EscPrinter(connection)

// Initialize printer
printer.initialize()
```

2. API Reference

ESC SDK API Documentation

Document Revision History

1. Getting Started
 - 1.1 Project Configuration
 - 1.2 Permission Configuration
 - 1.3 ProGuard Configuration
 - 1.4 Initialization
2. API Reference
3. initialize
- 3a. setEncoding
4. printText
5. printBarcode
6. printQRCode
7. printImage
8. feedPaper
9. cutPaper
10. getStatus
11. openCashDrawer
12. printTable
13. setAlignment
14. setFontStyle
15. setFontMultiplier
16. setLineSpacing
17. setLeftMargin
18. setReversePrinting
18. beep
19. selfTest
20. printPDF417
21. setPageMode
22. setPageModePrintArea
23. clearPageModeData
24. setPageModePrintDirection
25. setPageModeAbsolutePosition
26. printPageModeData
27. printPageModeText

- 28. [setStandardMode](#)
- 29. [drawPageRectangle](#)
- 30. [drawPageLine](#)
- 31. [printTwoColorImage](#)
- 32. [setAntiSkipPrinting](#)
- 33. [setPrintTaskId](#)
- 34. [setPrintEndSwitch](#)
- 35. [getPrintResult](#)
- 36. [cleanMissPrintData](#)
- 37. [checkMissPrintEnd](#)
- 38. [getPrinterSN](#)
- 39. [getPrinterFirmwareVersion](#)
- 40. [getPrinterManufacturer](#)
- 41. [getPrinterModel](#)
- 42. [setMissPrintListener](#)
- 43. [stopMissPrintListener](#)
- 44. [Appendix](#)
 - 44.1 [TextAlign Enum \(TextAlign\)](#)
 - 44.2 [Barcode Type Enum \(BarcodeType\)](#)
 - 44.3 [Dithering Algorithm Enum \(DitheringAlgorithm\)](#)
 - 44.4 [Underline Mode Enum \(UnderlineMode\)](#)
 - 44.5 [Compression Type Enum \(CompressionType\)](#)
 - 44.6 [Barcode Text Position Enum \(BarcodeTextPosition\)](#)
 - 44.7 [Page Mode Direction Enum \(PageModeDirection\)](#)
 - 44.8 [Error Types](#)

3. initialize

Description: Initialize the printer and restore default settings.

Method Signature:

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

Return Value: `Result<Unit>` Initialization result

Example Code:

```
printer.initialize()
```

3a. setEncoding

Description: Set global encoding.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
encoding	String	Yes	Encoding name (e.g., "GBK", "UTF-8")

Example Code:

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

4. printText

Description: Print text.

Method Signature:

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

Parameters:

Parameter	Type	Required	Default	Description
text	String	Yes	-	Text to print
options	TextOptions	No	TextOptions()	Text options

TextOptions Properties:

Property	Type	Default	Description
bold	Boolean	false	Bold
underline	Boolean	false	Underline
align	TextAlign	LEFT	Alignment
fontSmall	Boolean	false	Small font mode (9x17), otherwise large font (12x24)
reverse	Boolean	false	Reverse (inverse) printing
widthMultiplier	Int	0	Width multiplier (0-8)
heightMultiplier	Int	0	Height multiplier (0-8)

Example Code:

```
// Simple print
printer.printText("Hello World")

// Print with options
printer.printText(
    text = "Hello World",
    options = TextOptions(
        bold = true,
        underline = true,
        align = TextAlign.CENTER,
        fontSmall = false
    )
)
```

5. printBarcode

Description: Print barcode.

Method Signature:

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

Parameters:

Parameter	Type	Required	Default	Description
data	String	Yes	-	Barcode data
type	BarcodeType	No	CODE128	Barcode type
options	BarcodeOptions	No	BarcodeOptions()	Barcode options

BarcodeOptions Properties:

Property	Type	Default	Description
height	Int	50	Barcode height
width	Int	2	Barcode width
textPosition	BarcodeTextPosition	BELOW	Text position
rotation	Int	0	Rotation angle (not effective in ESC mode)

BarcodeType Enum Values:

Value	Description
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

Example Code:

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

6. printQRCode

Description: Print QR code.

Method Signature:

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

Parameters:

Parameter	Type	Required	Default	Description
data	String	Yes	-	QR code data
options	QRCodeOptions	No	QRCodeOptions()	QR code options

QRCodeOptions Properties:

Property	Type	Default	Description
size	Int	6	QR code size (1-16)
errorCorrection	QRErrorCorrection	M	Error correction level
rotation	Int	0	Rotation angle (not effective in ESC mode)

QRErrorCorrection Enum Values:

Value	Description
L	7% error correction
M	15% error correction
Q	25% error correction
H	30% error correction

Example Code:

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

7. printImage

Description: Print image.

Method Signature (Bitmap):

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

Parameters:

Parameter	Type	Required	Default	Description
bitmap	Bitmap	Yes	-	Image to print
options	ImageOptions	No	ImageOptions()	Image options

Method Signature (Byte Array):

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

Parameters:

Parameter	Type	Required	Default	Description
imageData	ByteArray	Yes	-	Image data (monochrome byte array)
width	Int	Yes	-	Image width
height	Int	Yes	-	Image height
options	ImageOptions	No	ImageOptions()	Image options

ImageOptions Properties:

Property	Type	Default	Description
ditheringAlgorithm	DitheringAlgorithm	DIFFUSION	Dithering algorithm
threshold	Int	-1	Binarization threshold (-1 for auto-calculate)
compression	CompressionType	NONE	Compression type
maxWidth	Int?	null	Maximum width

DitheringAlgorithm Enum Values:

Value	Description
THRESHOLD	Simple threshold
DIFFUSION	Diffusion dithering
CLUSTER	Cluster dithering

Example Code:

```
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

Description: Feed paper.

Method Signature:

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

Parameters:

Parameter	Type	Required	Default	Description
lines	Int	No	1	Number of lines to feed

Return Value: `Result<Unit>` Feed result

Example Code:

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

9. cutPaper

Description: Cut paper.

Method Signature:

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

Parameters:

Parameter	Type	Required	Default	Description
partial	Boolean	No	false	Whether to partially cut (true for partial, false for full)

Example Code:

```
printer.cutPaper() // Full cut  
printer.cutPaper(partial = true) // Partial cut
```

10. getStatus

Description: Get printer status.

Method Signature:

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

Return Value: `Result<PrinterStatus>` Printer status

PrinterStatus Properties:

Property	Type	Description
isOnline	Boolean	Is online
isPaperOut	Boolean	Is paper out
isPaperNearEnd	Boolean	Is paper near end
isCoverOpen	Boolean	Is cover open
isError	Boolean	Has error
isCutterError	Boolean	Is cutter error
isVoltageError	Boolean	Is voltage error
isOverheating	Boolean	Is overheating
isPrinting	Boolean	Is printing (not effective in ESC mode)
isRibbonOut	Boolean	Is ribbon out (ZPL) (not effective in ESC mode)
temperature	Int?	Temperature (not effective in ESC mode)
batteryLevel	Int?	Battery level (not effective in ESC mode)

Example Code:

```
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

Description: Open cash drawer.

Method Signature:

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

Parameters:

Parameter	Type	Required	Default	Description
pin	Int	No	0	Cash drawer pin (0 or 1)

Return Value: `Result<Unit>` Operation result

Example Code:

```
printer.openCashDrawer()
```

12. printTable

Description: Print table.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
rows	List<List<ESCCCommandGenerator.TableColumn>>	Yes	Table data, each row is a list of columns

TableColumn Properties:

Property	Type	Default	Description
text	String	-	Column text content
width	Int	-	Column width (characters)
align	TextAlign	TextAlign.LEFT	Text alignment (LEFT, CENTER, RIGHT)

Width Parameter Description:

- Unit: Characters
- Recommended values:
 - 80mm printer: Total width not exceeding 48 characters
 - 58mm printer: Total width not exceeding 32 characters
- Note: The total width of all columns should not exceed the maximum print width of the printer

Example Code:

```
// Print multi-row table with header and data
printer.printTable(
    rows = listOf(
        // Header row
        listOf(
            ESCCommandGenerator.TableColumn("Item", 20, TextAlign.LEFT),
            ESCCommandGenerator.TableColumn("Qty", 10, TextAlign.CENTER),
            ESCCommandGenerator.TableColumn("Price", 15, TextAlign.RIGHT)
        ),
        // Data row 1
        listOf(
            ESCCommandGenerator.TableColumn("Apple", 20, TextAlign.LEFT),
            ESCCommandGenerator.TableColumn("2", 10, TextAlign.CENTER),
            ESCCommandGenerator.TableColumn("10.00", 15, TextAlign.RIGHT)
        ),
        // Data row 2
        listOf(
            ESCCommandGenerator.TableColumn("Banana", 20, TextAlign.LEFT),
            ESCCommandGenerator.TableColumn("3", 10, TextAlign.CENTER),
            ESCCommandGenerator.TableColumn("15.00", 15, TextAlign.RIGHT)
        ),
        // Total row
        listOf(
            ESCCommandGenerator.TableColumn("Total", 20, TextAlign.LEFT),
            ESCCommandGenerator.TableColumn("5", 10, TextAlign.CENTER),
            ESCCommandGenerator.TableColumn("25.00", 15, TextAlign.RIGHT)
        )
    )
)
```

Usage Notes:

1. The total table width should not exceed the maximum print width of the printer, otherwise content will be truncated
2. When text length exceeds column width, the text will be truncated to fit the column width
3. Different font sizes affect table layout, reduce column width accordingly when using larger fonts
4. Monospaced fonts are recommended for better table alignment
5. For multi-row tables, ensure each row has the same number of columns

13. setAlignment

Description: Set alignment.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
align	TextAlign	Yes	Alignment

TextAlign Enum Values:

Value	Description
LEFT	Left aligned
CENTER	Center aligned
RIGHT	Right aligned

Example Code:

```
printer.setAlignment(TextAlign.CENTER)
printer.printText("Centered text")
```

14. setFontStyle

Description: Set font style.

Method Signature:

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

Parameters:

Parameter	Type	Required	Default	Description
bold	Boolean	No	false	Bold
underline	Boolean	No	false	Underline

Example Code:

```
printer.setFontStyle(bold = true, underline = true)
printer.printText("Bold with underline")
```

15. setFontMultiplier

Description: Set font width and height multiplier.

Method Signature:

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

Parameters:

Parameter	Type	Required	Default	Description
widthMultiplier	Int	No	0	Width multiplier (0-8). Note: 0 means no scaling (1x)
heightMultiplier	Int	No	0	Height multiplier (0-8). Note: 0 means no scaling (1x)

Example Code:

```
printer.setFontMultiplier(widthMultiplier = 2, heightMultiplier = 2)  
printer.printText("Enlarged text")
```

16. setLineSpacing

Description: Set line spacing.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
spacing	Int	Yes	Line spacing (dots, default 30)

Example Code:

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

17. setLeftMargin

Description: Set left margin.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
margin	Int	Yes	Left margin (dots)

Example Code:

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

18. setReversePrinting

Description: Set reverse printing.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
enable	Boolean	Yes	Enable reverse printing

Example Code:

```
printer.setReversePrinting(true)
printer.printText("Reverse text")
printer.setReversePrinting(false)
```

18. beep

Description: Beep.

Method Signature:

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

Parameters:

Parameter	Type	Required	Default	Description
times	Int	No	1	Number of beeps
t1	Int	No	1	ON time (milliseconds)
t2	Int	No	1	OFF time (milliseconds)

Example Code:

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

19. selfTest

Description: Print self-test page.

Method Signature:

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

Return Value: `Result<Unit>` Operation result

Example Code:

```
printer.selfTest()
```

20. printPDF417

Description: Print PDF417 barcode.

Method Signature:

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

Parameters:

Parameter	Type	Required	Default	Description
data	String	Yes	-	Barcode data
columns	Int	No	0	Number of columns (0 for auto)
rows	Int	No	0	Number of rows (0 for auto)
width	Int	No	3	Module width
height	Int	No	3	Module height
errorCorrection	Int	No	1	Error correction level (0-8)
errorMode	Int	No	0	Error mode
options	Int	No	0	Options

Example Code:

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

21. setPageMode

Description: Set page mode.

Method Signature:

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

Example Code:

```
printer.setPageMode()
```

22. setPageModePrintArea

Description: Set page mode print area.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
horizontal	Int	Yes	Horizontal position (dots)
vertical	Int	Yes	Vertical position (dots)
width	Int	Yes	Width (dots)
height	Int	Yes	Height (dots)

Example Code:

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

23. clearPageModeData

Description: Clear page mode data.

Method Signature:

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

Example Code:

```
printer.clearPageModeData()
```

24. setPageModePrintDirection

Description: Set page mode print direction.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
direction	PageModeDirection	Yes	Print direction

PageModeDirection Enum Values:

Value	Description
LEFT_TO_RIGHT	Left to right
BOTTOM_TO_TOP	Bottom to top
RIGHT_TO_LEFT	Right to left
TOP_TO_BOTTOM	Top to bottom

Example Code:

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

25. setPageModeAbsolutePosition

Description: Set page mode absolute position.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
xPosition	Int	Yes	X position (dots), origin at top-left
yPosition	Int	Yes	Y position (dots), origin at top-left

Example Code:

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

26. printPageModeData

Description: Print page mode data.

Method Signature:

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

Example Code:

```
printer.printPageModeData()
```

27. printPageModeText

Description: Print page mode text.

Method Signature:

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

Parameters:

Parameter	Type	Required	Default	Description
text	String	Yes	-	Text to print
encoding	String	No	Global encoding	Encoding

Example Code:

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

28. setStandardMode

Description: Set standard mode.

Method Signature:

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

Example Code:

```
printer.setStandardMode()
```

29. drawPageRectangle

Description: Draw page rectangle.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
x	Int	Yes	X position (dots), origin at top-left
y	Int	Yes	Y position (dots), origin at top-left
width	Int	Yes	Width (dots)
height	Int	Yes	Height (dots)
lineWidth	Int	Yes	Line width (dots), range: 0-255

Example Code:

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

30. drawPageLine

Description: Draw page line.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
x1	Int	Yes	Start X position (dots), origin at top-left
y1	Int	Yes	Start Y position (dots), origin at top-left
x2	Int	Yes	End X position (dots), origin at top-left
y2	Int	Yes	End Y position (dots), origin at top-left
lineWidth	Int	Yes	Line width (dots), range: 0-255

Example Code:

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

31. printTwoColorImage

Description: Print two-color image.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
blackImageData	ByteArray	Yes	Black image data
redImageData	ByteArray	Yes	Red image data
width	Int	Yes	Width (dots)
height	Int	Yes	Height (dots)

Method Signature (Bitmap):

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

Example Code:

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

32. setAntiSkipPrinting

Description: Set anti-skip printing.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
enable	Boolean	Yes	Whether to enable anti-skip printing

Example Code:

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

33. setPrintTaskId

Description: Set print task ID.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
taskId	Int	Yes	Task ID (1-9999)

Example Code:

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

34. setPrintEndSwitch

Description: Set print end switch.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
enable	Boolean	Yes	Whether to enable print end switch

Example Code:

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

35. getPrintResult

Description: Get print result status.

Method Signature:

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

Parameters:

Parameter	Type	Required	Default	Description
timeout	Long	No	30000	Timeout in milliseconds

Return Value: `Result<Int>` Print result status code (0 means success, -2 means timeout)

Example Code:

```
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}")
}
```

36. cleanMissPrintData

Description: Clean miss print data.

Method Signature:

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

Example Code:

```
printer.cleanMissPrintData()
```

37. checkMissPrintEnd

Description: Check miss print end.

Method Signature:

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

Example Code:

```
printer.checkMissPrintEnd()
```

38. getPrinterSN

Description: Get printer serial number.

Method Signature:

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

Return Value: `Result<String>` Printer serial number

Example Code:

```
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}")
}
```

39. getPrinterFirmwareVersion

Description: Get printer firmware version.

Method Signature:

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

Return Value: `Result<String>` Firmware version

Example Code:

```
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}")
}
```

40. getPrinterManufacturer

Description: Get printer manufacturer.

Method Signature:

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

Return Value: `Result<String>` Manufacturer name

Example Code:

```

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}")
}

```

41. getPrinterModel

Description: Get printer model.

Method Signature:

```

suspend fun getPrinterModel(): Result<String>

```

Return Value: `Result<String>` Model

Example Code:

```

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}")
}

```

42. setMissPrintListener

Description: Set miss print listener.

Method Signature:

```

fun setMissPrintListener(listener: MissPrintListener?)

```

Parameters:

Parameter	Type	Required	Description
listener	MissPrintListener?	Yes	Miss print listener

MissPrintListener Event Types:

Event	Description
FINISH	Print finished
ERROR	Error
END	End
CLEAN	Clean

Example Code:

```
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")
        }
    }
})
```

43. stopMissPrintListener

Description: Stop miss print listener.

Method Signature:

```
fun stopMissPrintListener()
```

Example Code:

```
printer.stopMissPrintListener()
```

44. Appendix

44.1 TextAlign Enum (TextAlign)

Value	Description
LEFT	Left aligned
CENTER	Center aligned
RIGHT	Right aligned

44.2 Barcode Type Enum (BarcodeType)

Value	Description
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

44.3 Dithering Algorithm Enum (DitheringAlgorithm)

Value	Description
THRESHOLD	Simple threshold
DIFFUSION	Diffusion dithering
CLUSTER	Cluster dithering

44.4 Underline Mode Enum (UnderlineMode)

Value	Description
NONE	No underline
SINGLE	Single line
DOUBLE	Double line

44.5 Compression Type Enum (CompressionType)

Value	Description
NONE	No compression
LZO	LZO compression

44.6 Barcode Text Position Enum (BarcodeTextPosition)

Value	Description
NONE	No text
ABOVE	Text above
BELOW	Text below
BOTH	Text above and below

44.7 Page Mode Direction Enum (PageModeDirection)

Value	Description
LEFT_TO_RIGHT	Left to right
BOTTOM_TO_TOP	Bottom to top
RIGHT_TO_LEFT	Right to left
TOP_TO_BOTTOM	Top to bottom

44.8 Error Types

Error Type	Description	Possible Scenarios
ConnectionError	Connection error	Printer connection failure, network issues, Bluetooth pairing failure
CommunicationError	Communication error	Data transmission failure, printer response error, communication timeout
InvalidParameterError	Invalid parameter error	Parameter value out of range, incorrect parameter format
TimeoutError	Timeout error	Operation execution timeout, printer no response
NotConnectedError	Not connected error	Attempting to perform operations in unconnected state
UnsupportedOperationError	Unsupported operation error	Printer model does not support the feature, protocol does not support the command
UnknownError	Unknown error	Other unclassified error situations

Error Code Note: When an API returns `Result.Failure`, the `PrinterError` carries an `errorCode` field (corresponds to `UXPErrorCode.value`). Each error type defaults to its fallback error code below; SDK call sites pass more specific error codes. See the UXPErrorCode Error Code Documentation (Guide folder) for the full `UXPErrorCode` table.

Fallback Code	Value	Description
ConnectionError	1001	Connection error
CommunicationError	1002	Communication error
InvalidParameterError	1003	Invalid parameter
TimeoutError	1004	Timeout
NotConnectedError	1005	Not connected
UnsupportedOperationError	1006	Unsupported operation
UnknownError	1007	Unknown error

Error Handling Example:

```
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}")
                // Try to reconnect
            }
        }
    }
}
```

```

    }
    is PrinterError.CommunicationError -> {
        Log.e("Printer", "Communication error:
${result.error.message}")
        // Check printer status
    }
    is PrinterError.InvalidParameterError -> {
        Log.e("Printer", "Invalid parameter: ${result.error.message}")
        // Correct parameter values
    }
    is PrinterError.TimeoutError -> {
        Log.e("Printer", "Timeout error: ${result.error.message}")
        // Increase timeout or check printer status
    }
    is PrinterError.NotConnectedError -> {
        Log.e("Printer", "Not connected: ${result.error.message}")
        // Connect to printer first
    }
    is PrinterError.UnsupportedOperationError -> {
        Log.e("Printer", "Unsupported operation:
${result.error.message}")
        // Use alternative solution
    }
    is PrinterError.UnknownError -> {
        Log.e("Printer", "Unknown error: ${result.error.message}")
        // Log error and try to recover
    }
}
}
}

```