

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

2. API Reference

CPCL 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. 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. Appendix
 - 47.1 Font Enum (CpclFont)
 - 47.2 Rotation Enum (Rotation)
 - 47.3 Barcode Type Enum (BarcodeType)
 - 47.4 Dithering Algorithm Enum (DitheringAlgorithm)
 - 47.5 Error Types

3. printText

Description: Print text.

Method Signature:

```
suspend fun printText(  
    text: String,  
    x: Int = 0,  
    y: Int = 0,  
    options: CpclTextOptions = CpclTextOptions()  
) : Result<Unit>
```

Parameters:

Parameter	Type	Required	Default	Description
text	String	Yes	-	Text to print
x	Int	No	0	X coordinate (dots)
y	Int	No	0	Y coordinate (dots)
options	CpclTextOptions	No	CpclTextOptions()	Text options

CpclTextOptions Properties:

Property	Type	Default	Description
fontSize	Int	24	Font size (CPCL font ID: 0,1,2,3,4,8,16,24,48,55), see details below
bold	Int	0	Bold intensity (0=off, 1-5=intensity)

fontSize Detailed Explanation:

Value	Description
0	24x24 or 12x24, depends on language (Thai: 24x48)
1	7x19 (English), 24x24 (Traditional Chinese)
2	8x8
3	20x20 or 10x20, depends on language
4	32x32 or 16x32, ID3 font scaled 2x
8	24x24 or 12x24, depends on language
16	16x16 or 8x16, depends on language
24	24x24 or 12x24, depends on language
48	48x48 or 24x48, depends on language
55	16x16 or 8x16, depends on language

Example Code:

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

4. printBarcode

Description: Print barcode.

Method Signature:

```
suspend fun printBarcode(
    data: String,
    type: BarcodeType,
    x: Int = 0,
    y: Int = 0,
    options: BarcodeOptions = BarcodeOptions()
): Result<Unit>
```

Parameters:

Parameter	Type	Required	Default	Description
data	String	Yes	-	Barcode data
type	BarcodeType	Yes	-	Barcode type
x	Int	No	0	X coordinate (dots)
y	Int	No	0	Y coordinate (dots)
options	BarcodeOptions	No	BarcodeOptions()	Barcode options

BarcodeOptions Properties:

Property	Type	Default	Description
width	Int	2	Barcode width (1-6)
height	Int	60	Barcode height (dots)
showText	Boolean	true	Show text below barcode
textAbove	Boolean	false	Text above barcode
rotation	Int	0	Rotation angle (0/90/180/270)

BarcodeType Detailed Description:

Value	Description	Supported Data Format
UPC_A	UPC-A	11-12 digits
UPC_E	UPC-E	6-8 digits
EAN13	EAN-13	12-13 digits
EAN8	EAN-8	7-8 digits
CODE39	Code 39	Letters, numbers, special characters
CODE93	Code 93	Letters, numbers, special characters
CODE128	Code 128	ASCII character set
CODABAR	Codabar	Numbers and special characters
ITF	ITF	Even number of digits

Example Code:

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

5. printQRCode

Description: Print QR code.

Method Signature:

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

Parameters:

Parameter	Type	Required	Default	Description
x	Int	Yes	-	X coordinate (dots)
y	Int	Yes	-	Y coordinate (dots)
data	String	Yes	-	QR code data
size	Int	No	6	QR code size (1-16)
options	QRCodeOptions	No	QRCodeOptions()	QR code options

QRCodeOptions Properties:

Property	Type	Default	Description
errorCorrection	ErrorCorrectionLevel	M	Error correction level
rotation	Rotation	NORMAL	Rotation angle

Example Code:

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

6. printImage

Description: Print image.

Method Signature:

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

Parameters:

Parameter	Type	Required	Default	Description
x	Int	Yes	-	X coordinate (dots)
y	Int	Yes	-	Y coordinate (dots)
bitmap	Bitmap	Yes	-	Image to print
options	ImageOptions	No	ImageOptions()	Image options

ImageOptions Properties:

Property	Type	Default	Description
threshold	Int	-1	Binarization threshold (-1 for auto-calculate)
algorithm	DitheringAlgorithm	DIFFUSION	Dithering algorithm

Example Code:

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

7. print

Description: Execute print.

Method Signature:

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

Parameters:

Parameter	Type	Required	Default	Description
copies	Int	No	1	Number of copies

Return Value: `Result<Unit>` Print result

Example Code:

```
// Add print content then execute print
printer.printText(text = "Label Content", x = 10, y = 10)
printer.printQRCode(x = 100, y = 50, data = "QR Data")
printer.print(copies = 1)
```

8. 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	Printer online status
isPaperOut	Boolean	Paper out status
isPaperNearEnd	Boolean	Paper near end status
isCoverOpen	Boolean	Cover open status
isError	Boolean	Error status
isCutterError	Boolean	Cutter error status
isVoltageError	Boolean	Voltage error status
isOverheating	Boolean	Overheating status
isPrinting	Boolean	Printing status
isRibbonOut	Boolean	Ribbon out status (ZPL protocol)
temperature	Int?	Temperature (Celsius), may be null
batteryLevel	Int?	Battery level (0-100%), may be null

Example Code:

```
val result = printer.getStatus()
when (result) {
    is Result.Success -> {
        val status = result.value
        when {
            status.isPaperOut -> Log.w("Printer", "No paper!")
            status.isCoverOpen -> Log.w("Printer", "Cover is open!")
            status.isOverheating -> Log.w("Printer", "Printer is
overheating!")
            status.isError -> Log.e("Printer", "Printer error!")
            status.isOnline -> Log.d("Printer", "Printer is online")
        }
    }
    is Result.Failure -> Log.e("Printer", "Error: ${result.error.message}")
}
```

9. setPrintArea

Description: Set print area.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
width	Int	Yes	Print area width (dots)
height	Int	Yes	Print area height (dots)

Example Code:

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

10. setBold

Description: Set bold.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
level	Int	Yes	Bold level (0-5), 0=off, 1-5=intensity

Example Code:

```
printer.setBold(3)
printer.printText(text = "Bold Text", x = 10, y = 10)
```

11. backFeed

Description: Back feed paper.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
length	Int	Yes	Back feed length (dots)

Example Code:

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

12. form

Description: Feed to tear position.

Method Signature:

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

Return Value: `Result<Unit>` Operation result

Example Code:

```
printer.form()
```

13. setAlign

Description: Set alignment.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
alignment	Alignment	Yes	Alignment (LEFT/CENTER/RIGHT)

Example Code:

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

14. setContrast

Description: Set print contrast.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
contrast	Int	Yes	Contrast value (0-3)

Example Code:

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

15. setSpeed

Description: Set print speed.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
speed	Int	Yes	Speed value (0-3)

Example Code:

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

16. drawBox

Description: Draw box.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
x0	Int	Yes	Top-left X coordinate (dots)
y0	Int	Yes	Top-left Y coordinate (dots)
x1	Int	Yes	Bottom-right X coordinate (dots)
y1	Int	Yes	Bottom-right Y coordinate (dots)
thickness	Int	Yes	Line width (dots)

Example Code:

```
printer.drawBox(x0 = 10, y0 = 10, x1 = 200, y1 = 100, thickness = 2)
```

17. drawLine

Description: Draw line.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
x0	Int	Yes	Start X coordinate (dots)
y0	Int	Yes	Start Y coordinate (dots)
x1	Int	Yes	End X coordinate (dots)
y1	Int	Yes	End Y coordinate (dots)
thickness	Int	Yes	Line width (dots)

Example Code:

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

18. drawInverseLine

Description: Draw inverse line.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
x0	Int	Yes	Start X coordinate (dots)
y0	Int	Yes	Start Y coordinate (dots)
x1	Int	Yes	End X coordinate (dots)
y1	Int	Yes	End Y coordinate (dots)
thickness	Int	Yes	Line width (dots)

Example Code:

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

19. setMagnification

Description: Set font magnification.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
width	Int	Yes	Width multiplier (1-8)
height	Int	Yes	Height multiplier (1-8)

Example Code:

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

20. beep

Description: Beep.

Method Signature:

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

Parameters:

Parameter	Type	Required	Default	Description
times	Int	No	1	Number of beeps

Example Code:

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

21. setPageWidth

Description: Set page width.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
width	Int	Yes	Page width (dots)

Example Code:

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

22. printReverse

Description: Reverse print (inverts previously printed content area).

Method Signature:

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

Parameters: None

Example Code:

```
printer.printText(text = "Some Text", x = 10, y = 10)  
printer.printReverse()
```

23. setLayer

Description: Set layer.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
layer	Int	Yes	Layer number

Example Code:

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

24. saveForm

Description: Save form.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
name	String	Yes	Form name

Example Code:

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

25. loadForm

Description: Load form.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
name	String	Yes	Form name

Example Code:

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

26. deleteForm

Description: Delete form.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
name	String	Yes	Form name

Example Code:

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

27. selfTest

Description: Self test.

Method Signature:

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

Return Value: `Result<Unit>` Operation result

Example Code:

```
printer.selfTest()
```

28. printTextWithFont

Description: Print text with specified font.

Method Signature:

```
suspend fun printTextWithFont(  
    text: String,  
    x: Int,  
    y: Int,  
    font: String,  
    size: Int,  
    encoding: String = Companion.encoding  
): Result<Unit>
```

Parameters:

Parameter	Type	Required	Description
text	String	Yes	Text to print
x	Int	Yes	X coordinate (dots)
y	Int	Yes	Y coordinate (dots)
font	String	Yes	Font name, e.g., "0" for default font
size	Int	Yes	Font size
encoding	String	No	Encoding

Example Code:

```
printer.printTextWithFont(
    text = "Hello",
    x = 10,
    y = 10,
    font = "001",
    size = 24
)
```

29. printTextBlock

Description: Print text block.

Method Signature:

```
suspend fun printTextBlock(
    text: String,
    x: Int,
    y: Int,
    width: Int,
    size: Int,
    flags: Int,
    lineHeight: Int,
    encoding: String = Companion.encoding
): Result<Unit>
```

Parameters:

Parameter	Type	Required	Description
text	String	Yes	Text to print
x	Int	Yes	X coordinate (dots)
y	Int	Yes	Y coordinate (dots)
width	Int	Yes	Width (dots)
size	Int	Yes	Font size (dots)
flags	Int	Yes	Flags
lineSpace	Int	Yes	Line space
encoding	String	No	Encoding

Example Code:

```
printer.printTextBlock(
    text = "Hello World",
    x = 10,
    y = 10,
    width = 200,
    size = 24,
    flags = 0,
    lineSpace = 0
)
```

30. printBackgroundText

Description: Print text with background.

Method Signature:

```
suspend fun printBackgroundText(
    text: String,
    x: Int,
    y: Int,
    width: Int,
    height: Int,
    font: Int,
    flags: Int,
    rotation: CpclTextRotation = CpclTextRotation.NORMAL,
    encoding: String = Companion.encoding
): Result<Unit>
```

Parameters:

Parameter	Type	Required	Description
text	String	Yes	Text to print
x	Int	Yes	X coordinate (dots)
y	Int	Yes	Y coordinate (dots)
width	Int	Yes	Width (dots)
height	Int	Yes	Height (dots)
font	Int	Yes	Font
flags	Int	Yes	Flags
rotation	CpclTextRotation	No	Rotation
encoding	String	No	Encoding

Example Code:

```
printer.printBackgroundText(
    text = "Hello",
    x = 10,
    y = 10,
    width = 100,
    height = 30,
    font = 0,
    flags = 0
)
```

31. printPDF417

Description: Print PDF417 barcode.

Method Signature:

```
suspend fun printPDF417(
    data: String,
    x: Int = 0,
    y: Int = 0,
    xd: Int = 2,
    yd: Int = 6,
    columns: Int = 2,
    securityLevel: Int = 2
): Result<Unit>
```

Parameters:

Parameter	Type	Required	Default	Description
data	String	Yes	-	Barcode data
x	Int	No	0	X coordinate (dots)
y	Int	No	0	Y coordinate (dots)
xd	Int	No	2	Horizontal module width (dots)
yd	Int	No	6	Vertical module height (dots)
columns	Int	No	2	Data columns (1-30)
securityLevel	Int	No	2	Security level (0-8)

Example Code:

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

32. cutPaper

Description: Cut paper.

Method Signature:

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

Parameters:

Parameter	Type	Required	Default	Description
partial	Boolean	No	false	Partial cut

Return Value: `Result<Unit>` Operation result

Example Code:

```
// Full cut
printer.cutPaper()

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

33. setCountry

Description: Set country/region.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
country	String	Yes	Country/region code

Supported Country/Region Codes:

Code	Country/Region
USA	United States
UK	United Kingdom
Germany	Germany
France	France
Sweden	Sweden
Norway	Norway
Italy	Italy
Spain	Spain
Denmark	Denmark
Belgium	Belgium
Portugal	Portugal
Japan	Japan
Korea	Korea
China	China
Taiwan	Taiwan
HongKong	Hong Kong

Example Code:

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

34. setUnderline

Description: Set underline.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
enabled	Boolean	Yes	Enable underline

Example Code:

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

35. openEndStatus

Description: Open end status.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
enable	Boolean	Yes	Enable end status

Example Code:

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

36. setTaskId

Description: Set task ID.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
taskId	Int	Yes	Task ID

Example Code:

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

37. getEndStatus

Description: Get end status.

Method Signature:

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

Parameters:

Parameter	Type	Required	Default	Description
timeout	Long	No	2000	Timeout (milliseconds)

Return Value: `Result<Int>` Status code (0=success, non-zero=failure)

Example Code:

```
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

Description: Get print result status.

Method Signature:

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

Parameters:

Parameter	Type	Required	Default	Description
timeout	Long	No	2000	Timeout (milliseconds)

Return Value: `Result<PrintTaskResult>` Print task result

Example Code:

```
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

Description: Get print result without ID.

Method Signature:

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

Parameters:

Parameter	Type	Required	Default	Description
timeout	Long	No	2000	Timeout (milliseconds)

Return Value: `Result<Int>` Print result status code (0=success, non-zero=failure)

Example Code:

```
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

Description: Print codepage text.

Method Signature:

```
suspend fun printCodepageTextCPCL(  
    text: String,  
    x: Int,  
    y: Int,  
    font: Int,  
    flags: Int,  
    rotation: CpclTextRotation = CpclTextRotation.NORMAL,  
    encoding: String = Companion.encoding  
): Result<Unit>
```

Parameters:

Parameter	Type	Required	Default	Description
text	String	Yes	-	Text to print
x	Int	Yes	-	X coordinate (dots)
y	Int	Yes	-	Y coordinate (dots)
font	Int	Yes	-	Font (0=standard, 1=small)
flags	Int	Yes	-	Flags, bit flags combination
rotation	CpclTextRotation	No	CpclTextRotation.NORMAL	Rotation
encoding	String	No	Companion.encoding	Encoding

flags Parameter Detailed Description:

Bit	Value	Description
bit0	1	Bold
bit1	2	Inverse
bit2	4	Double width
bit3	8	Double height

flags Combination Examples:

flags Value	Effect
0	Normal text
1	Bold
3	Bold + Inverse
5	Bold + Double width
12	Double width + Double height

Example Code:

```
printer.printCodepageTextCPCL(
    text = "Hello",
    x = 10,
    y = 10,
    font = 0,
    flags = 1
)
```

41. printTextCPCL

Description: Print CPCL text.

Method Signature:

```
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>
```

Parameters:

Parameter	Type	Required	Default	Description
text	String	Yes	-	Text to print
x	Int	Yes	-	X coordinate (dots)
y	Int	Yes	-	Y coordinate (dots)
font	Int	Yes	-	Font size (0, 1, 16, 24, 48)
flags	Int	Yes	-	Flags, bit flags combination
isCenter	Boolean	Yes	-	Center align
width	Int	Yes	-	Width (dots)
rotation	CpclTextRotation	No	CpclTextRotation.NORMAL	Rotation
encoding	String	No	Companion.encoding	Encoding

Example Code:

```
printer.printTextCPCL(
    text = "Hello CPCL",
    x = 10,
    y = 10,
    font = 24,
    flags = 0,
    isCenter = true,
    width = 200
)
```

42. autCenter

Description: Auto center print.

Method Signature:

```
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>
```

Parameters:

Parameter	Type	Required	Default	Description
text	String	Yes	-	Text to print
x	Int	Yes	-	X coordinate (dots)
y	Int	Yes	-	Y coordinate (dots)
width	Int	Yes	-	Width (dots)
size	Int	Yes	-	Font size
isDouble	Boolean	Yes	-	Double size
rotation	CpclTextRotation	No	CpclTextRotation.NORMAL	Rotation
encoding	String	No	Companion.encoding	Encoding

Example Code:

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

43. autoLinePlus

Description: Auto line plus.

Method Signature:

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

Parameters:

Parameter	Type	Required	Default	Description
x	Int	Yes	-	X coordinate (dots)
y	Int	Yes	-	Y coordinate (dots)
width	Int	Yes	-	Width (dots)
size	Int	Yes	-	Font size
isBold	Boolean	Yes	-	Bold
isDouble	Boolean	Yes	-	Double size
text	String	Yes	-	Text to print
encoding	String	No	Companion.encoding	Encoding

Example Code:

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

44. autLine

Description: Auto line.

Method Signature:

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

Parameters:

Parameter	Type	Required	Default	Description
x	Int	Yes	-	X coordinate (dots)
y	Int	Yes	-	Y coordinate (dots)
width	Int	Yes	-	Width (dots)
size	Int	Yes	-	Font size
isBold	Boolean	Yes	-	Bold
isDouble	Boolean	Yes	-	Double size
text	String	Yes	-	Text to print
encoding	String	No	Companion.encoding	Encoding

Example Code:

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

45. clear

Description: Clear print buffer.

Method Signature:

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

Return Value: `Result<Unit>` Operation result

Example Code:

```
printer.clear()
```

46. setEncoding

Description: Set encoding.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
encoding	String	Yes	Encoding format

Example Code:

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

47. Appendix

47.1 Font Enum (CpclFont)

Value	Description
FONT_A	Font A (12x24)
FONT_B	Font B (9x17)
FONT_C	Font C (9x24)

47.2 Rotation Enum (Rotation)

Value	Description
NORMAL	Normal (0 degrees)
ROTATE_90	Rotate 90 degrees
ROTATE_180	Rotate 180 degrees
ROTATE_270	Rotate 270 degrees

47.3 Barcode Type Enum (BarcodeType)

Value	Description
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 Dithering Algorithm Enum (DitheringAlgorithm)

Value	Description
THRESHOLD	Simple threshold
DIFFUSION	Diffusion dithering
CLUSTER	Cluster dithering

47.5 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(text = "Hello, World!", x = 0, y = 0)
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}")
            }
            is PrinterError.CommunicationError -> {
                Log.e("Printer", "Communication error:
${result.error.message}")
            }
            is PrinterError.InvalidParameterError -> {
                Log.e("Printer", "Invalid parameter: ${result.error.message}")
            }
            is PrinterError.TimeoutError -> {
                Log.e("Printer", "Timeout error: ${result.error.message}")
            }
            is PrinterError.NotConnectedError -> {
                Log.e("Printer", "Not connected: ${result.error.message}")
            }
            is PrinterError.UnsupportedOperationError -> {
                Log.e("Printer", "Unsupported operation:
${result.error.message}")
            }
        }
    }
}
```

```
is PrinterError.UnknownError -> {  
    Log.e("Printer", "Unknown error: ${result.error.message}")  
}  
}  
}
```