

TSPL 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 TsplPrinter instance
val printer = TsplPrinter(connection)

// Set label size (prerequisite for label printing)
printer.setSize(width = 60, height = 40)
printer.setGap(gap = 2, offset = 0)

// Set encoding (optional)
printer.setEncoding("GBK")
```

2. API Reference

TSPL 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. setSize
4. setGap
5. setBlackMark
6. setOffset
7. setReference
8. setDirection
9. setSpeed
10. setDensity
11. setCodePage
12. setEncoding
13. setBold
14. setBlackMarkPosition
15. printText
16. printBarcode
17. printQRCode
18. printImage
19. drawBox
20. drawBar
21. drawCircle
22. drawEllipse
23. erase
24. reverse
25. printPDF417

- 26. [printDataMatrix](#)
- 27. [printMaxiCode](#)
- 28. [putPcx](#)
- 29. [printWaterMark](#)
- 30. [print](#)
- 31. [clear](#)
- 32. [feedPaper](#)
- 33. [backFeed](#)
- 34. [limitFeed](#)
- 35. [formFeed](#)
- 36. [home](#)
- 37. [cutPaper](#)
- 38. [sound](#)
- 39. [selfTest](#)
- 40. [Appendix](#)
 - 40.1 [Font Enum \(TsplFont\)](#)
 - 40.2 [Rotation Enum \(Rotation\)](#)
 - 40.3 [Barcode Type Enum \(BarcodeType\)](#)
 - 40.4 [Dithering Algorithm Enum \(DitheringAlgorithm\)](#)
 - 40.5 [Error Types](#)

3. setSize

Description: Set label size. **Prerequisite for label printing, must be called before printing.**

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
width	Int	Yes	Label width (mm)
height	Int	Yes	Label height (mm)

Example Code:

```
printer.setSize(width = 60, height = 40)
```

4. setGap

Description: Set label gap.

Method Signature:

```
suspend fun setGap(distance: Int, offset: Int = 0): Result<Unit>
```

Parameters:

Parameter	Type	Required	Default	Description
distance	Int	Yes	-	Gap height (mm)
offset	Int	No	0	Offset (mm)

Example Code:

```
printer.setGap(distance = 2, offset = 0)
```

5. setBlackMark

Description: Set black mark mode.

Method Signature:

```
suspend fun setBlackMark(height: Int, offset: Int = 0): Result<Unit>
```

Parameters:

Parameter	Type	Required	Default	Description
height	Int	Yes	-	Black mark height (mm)
offset	Int	No	0	Offset (mm)

Example Code:

```
printer.setBlackMark(height = 10, offset = 0)
```

6. setOffset

Description: Set print offset.

Method Signature:

```
suspend fun setOffset(distance: Int): Result<Unit>
```

Parameters:

Parameter	Type	Required	Description
distance	Int	Yes	Offset (mm)

Example Code:

```
printer.setOffset(0)
```

7. setReference

Description: Set reference point.

Method Signature:

```
suspend fun setReference(x: Int, y: Int): Result<Unit>
```

Parameters:

Parameter	Type	Required	Description
x	Int	Yes	X coordinate (dots)
y	Int	Yes	Y coordinate (dots)

Example Code:

```
printer.setReference(x = 0, y = 0)
```

8. setDirection

Description: Set print direction.

Method Signature:

```
suspend fun setDirection(direction: Int): Result<Unit>
```

Parameters:

Parameter	Type	Required	Description
direction	Int	Yes	Print direction (0=left-to-right, 1=right-to-left). Use the Direction enum (see Appendix 40.2)

Example Code:

```
printer.setDirection(0)
```

9. setSpeed

Description: Set print speed.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
speed	Int	Yes	Print speed (1-14, inches/second)

Example Code:

```
printer.setSpeed(4)
```

10. setDensity

Description: Set print density.

Method Signature:

```
suspend fun setDensity(density: Int): Result<Unit>
```

Parameters:

Parameter	Type	Required	Description
density	Int	Yes	Density value (0-15, 0=lightest, 15=darkest)

Example Code:

```
printer.setDensity(10)
```

11. setCodePage

Description: Set code page.

Method Signature:

```
suspend fun setCodePage(codePage: String): Result<Unit>
```

Parameters:

Parameter	Type	Required	Description
codePage	String	Yes	Code page name

Supported Code Pages:

Code Page	Description
PC437	USA
PC850	Multilingual
PC852	Latin 2
PC860	Portuguese
PC863	Canadian French
PC865	Nordic
PC857	Turkish
PC737	Greek
ISO-8859-1	Western European (Latin-1)
ISO-8859-2	Central European (Latin-2)
ISO-8859-5	Cyrillic
ISO-8859-7	Greek
ISO-8859-9	Turkish
GBK	Simplified Chinese (GB2312 extended)
GB2312	Simplified Chinese
Big5	Traditional Chinese
UTF-8	Unicode

Example Code:

```
printer.setCodePage("GBK")
```

12. setEncoding

Description: Set encoding. Used to set the character encoding for communication between printer and computer.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
encoding	String	Yes	Encoding format (e.g., GBK, UTF-8, GB2312, etc.)

Example Code:

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

13. setBold

Description: Set bold.

Method Signature:

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

Parameters:

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

Example Code:

```
printer.setBold(1)
```

14. setBlackMarkPosition

Description: Set black mark position.

Method Signature:

```
suspend fun setBlackMarkPosition(mode: Int): Result<Unit>
```

Parameters:

Parameter	Type	Required	Description
mode	Int	Yes	Mode (0=disable, 1=enable)

Example Code:

```
printer.setBlackMarkPosition(0)
```

15. printText

Description: Print text.

Method Signature:

```
suspend fun printText(  
    text: String,  
    x: Int = 0,  
    y: Int = 0,  
    options: TextOptions = TextOptions()  
): 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	TextOptions	No	TextOptions()	Text options

TextOptions Properties:

Property	Type	Default	Description
font	TsplFont	FONT_0	Font
fontSize	Int?	null	Font size (dots)
xMultiplier	Int	1	Width multiplier (1-10)
yMultiplier	Int	1	Height multiplier (1-10)
bold	Boolean	false	Not supported by TSPL
underline	Boolean	false	Not supported by TSPL
align	TextAlign	LEFT	Not supported by TSPL
fontSmall	Boolean	false	Not supported by TSPL
reverse	Boolean	false	Not supported by TSPL
rotation	Rotation	NORMAL	Rotation angle

Example Code:

```
printer.printText(
    text = "TSPL Label",
    x = 50,
    y = 30,
    options = TextOptions(
        font = TsplFont.FONT_0,
        fontSize = 24,
        xMultiplier = 2,
        yMultiplier = 2
    )
)
```

16. 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
showText	Boolean	true	Show text below barcode
textAbove	Boolean	false	Text above barcode
rotation	Rotation	NORMAL	Not supported by TSPL

BarcodeType Detailed Description:

Value	Description	Supported Data Format
CODE128	Code 128	ASCII character set
CODE128A	Code 128 A	Uppercase letters and numbers
CODE128B	Code 128 B	Uppercase and lowercase letters and numbers
CODE128C	Code 128 C	Numbers only (even number of digits)
CODE39	Code 39	Letters, numbers, special characters
CODE93	Code 93	Letters, numbers, special characters
EAN13	EAN-13	12-13 digits
EAN8	EAN-8	7-8 digits
UPC_A	UPC-A	11-12 digits
UPC_E	UPC-E	6-8 digits
CODABAR	Codabar	Numbers and special characters
ITF	ITF	Even number of digits

Example Code:

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

17. printQRCode

Description: Print QR code.

Method Signature:

```
suspend fun printQRCode(
    data: String,
    x: Int = 0,
    y: Int = 0,
    options: QRCodeOptions = QRCodeOptions()
): Result<Unit>
```

Parameters:

Parameter	Type	Required	Default	Description
data	String	Yes	-	QR code data
x	Int	No	0	X coordinate (dots)
y	Int	No	0	Y coordinate (dots)
options	QRCodeOptions	No	QRCodeOptions()	QR code options

QRCodeOptions Properties:

Property	Type	Default	Description
moduleSize	Int	6	Module size (1-16, dots)
errorCorrection	ErrorCorrectionLevel	M	Error correction level
rotation	Rotation	NORMAL	Not supported by TSPL

Example Code:

```
printer.printQRCode(
    data = "https://www.example.com",
    x = 200,
    y = 150,
    options = QRCodeOptions(
        moduleSize = 8,
        errorCorrection = ErrorCorrectionLevel.H
    )
)
```

18. printImage

Description: Print image.

Method Signature:

```
// Overload 1: Using byte array
suspend fun printImage(
    imageData: ByteArray,
    width: Int,
    height: Int,
    x: Int = 0,
    y: Int = 0,
    options: ImageOptions = ImageOptions()
): Result<Unit>

// Overload 2: Using Bitmap
suspend fun printImage(
    bitmap: Bitmap,
    x: Int = 0,
    y: Int = 0,
    options: ImageOptions = ImageOptions()
): Result<Unit>
```

Parameters:

Parameter	Type	Required	Default	Description
imageData	ByteArray	Yes (overload 1)	-	Image byte data
width	Int	Yes (overload 1)	-	Image width (dots)
height	Int	Yes (overload 1)	-	Image height (dots)
bitmap	Bitmap	Yes (overload 2)	-	Image to print
x	Int	No	0	X coordinate (dots)
y	Int	No	0	Y coordinate (dots)
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
maxWidth	Int?	null	Maximum width (will auto-scale if exceeded, dots)

Example Code:

```
val bitmap = BitmapFactory.decodeResource(resources, R.drawable.logo)
printer.printImage(
    bitmap = bitmap,
    x = 50,
    y = 200,
    options = ImageOptions(algorithm = DitheringAlgorithm.DIFFUSION)
)
```

19. drawBox

Description: Draw box.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
x	Int	Yes	Top-left X coordinate (dots)
y	Int	Yes	Top-left Y coordinate (dots)
width	Int	Yes	Box width (dots)
height	Int	Yes	Box height (dots)
thickness	Int	Yes	Line width (dots)

Example Code:

```
printer.drawBox(x = 10, y = 10, width = 300, height = 200, thickness = 2)
```

20. drawBar

Description: Draw solid bar.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
x	Int	Yes	Top-left X coordinate (dots)
y	Int	Yes	Top-left Y coordinate (dots)
width	Int	Yes	Width (dots)
height	Int	Yes	Height (dots)

Example Code:

```
printer.drawBar(x = 10, y = 100, width = 300, height = 5)
```

21. drawCircle

Description: Draw circle.

Method Signature:

```
suspend fun drawCircle(  
    x: Int,  
    y: Int,  
    diameter: Int,  
    thickness: Int  
): Result<Unit>
```

Parameters:

Parameter	Type	Required	Description
x	Int	Yes	Center X coordinate (dots)
y	Int	Yes	Center Y coordinate (dots)
diameter	Int	Yes	Diameter (dots)
thickness	Int	Yes	Line width (dots)

Example Code:

```
printer.drawCircle(x = 150, y = 150, diameter = 100, thickness = 2)
```

22. drawEllipse

Description: Draw ellipse.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
x	Int	Yes	Top-left X coordinate (dots)
y	Int	Yes	Top-left Y coordinate (dots)
width	Int	Yes	Ellipse width (dots)
height	Int	Yes	Ellipse height (dots)
thickness	Int	Yes	Line width (dots)

Example Code:

```
printer.drawEllipse(x = 50, y = 50, width = 200, height = 100, thickness = 2)
```

23. erase

Description: Erase area.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
x	Int	Yes	Top-left X coordinate (dots)
y	Int	Yes	Top-left Y coordinate (dots)
width	Int	Yes	Width (dots)
height	Int	Yes	Height (dots)

Example Code:

```
printer.erase(x = 0, y = 0, width = 100, height = 50)
```

24. reverse

Description: Reverse area.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
x	Int	Yes	Top-left X coordinate (dots)
y	Int	Yes	Top-left Y coordinate (dots)
width	Int	Yes	Width (dots)
height	Int	Yes	Height (dots)

Example Code:

```
printer.reverse(x = 0, y = 0, width = 100, height = 50)
```

25. printPDF417

Description: Print PDF417 barcode.

Method Signature:

```
suspend fun printPDF417(
    x: Int,
    y: Int,
    width: Int,
    height: Int,
    rotate: Int,
    options: List<String> = emptyList(),
    expression: String
): 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)
height	Int	Yes	-	Height (dots)
rotate	Int	Yes	-	Rotation angle (0/90/180/270)
options	List	No	emptyList()	Options list, each element in the format 'key=value'
expression	String	Yes	-	Expression

Example Code:

```
printer.printPDF417(  
    x = 50,  
    y = 200,  
    width = 100,  
    height = 50,  
    rotate = 0,  
    expression = "PDF417 Data"  
)
```

26. printDataMatrix

Description: Print DataMatrix barcode.

Method Signature:

```
suspend fun printDataMatrix(  
    x: Int,  
    y: Int,  
    width: Int,  
    height: Int,  
    xm: Int,  
    row: Int,  
    col: Int,  
    expression: String  
) : Result<Unit>
```

Parameters:

Parameter	Type	Required	Description
x	Int	Yes	X coordinate (dots)
y	Int	Yes	Y coordinate (dots)
width	Int	Yes	Width (dots)
height	Int	Yes	Height (dots)
xm	Int	Yes	X module count (1-16)
row	Int	Yes	Row count
col	Int	Yes	Column count
expression	String	Yes	Expression

Example Code:

```
printer.printDataMatrix(
    x = 50,
    y = 200,
    width = 100,
    height = 100,
    xm = 10,
    row = 10,
    col = 10,
    expression = "DataMatrix Data"
)
```

27. printMaxiCode

Description: Print MaxiCode barcode.

Method Signature:

```
suspend fun printMaxiCode(
    x: Int,
    y: Int,
    mode: Int,
    classType: Int,
    country: Int,
    post: String,
    lm: String,
    message: String
): Result<Unit>
```

Parameters:

Parameter	Type	Required	Description
x	Int	Yes	X coordinate (dots)
y	Int	Yes	Y coordinate (dots)
mode	Int	Yes	Mode (2 or 3)
classType	Int	Yes	Class type (0-255)
country	Int	Yes	Country code
post	String	Yes	Postal code
lm	String	Yes	Location info (e.g., postal code, etc.)
message	String	Yes	Message content

Example Code:

```
printer.printMaxiCode(  
    x = 100,  
    y = 100,  
    mode = 2,  
    classType = 0,  
    country = 1,  
    post = "12345",  
    lm = "",  
    message = "MaxiCode Data"  
)
```

28. putPcx

Description: Put PCX file.

Method Signature:

```
suspend fun putPcx(x: Int, y: Int, filename: String): Result<Unit>
```

Parameters:

Parameter	Type	Required	Description
x	Int	Yes	X coordinate (dots)
y	Int	Yes	Y coordinate (dots)
filename	String	Yes	Filename

Example Code:

```
printer.putPcx(x = 50, y = 50, filename = "logo.pcx")
```

29. printWaterMark

Description: Print watermark.

Method Signature:

```
suspend fun printWaterMark(density: Int): Result<Unit>
```

Parameters:

Parameter	Type	Required	Description
density	Int	Yes	Density (0-10, 0=lightest, 10=darkest)

Example Code:

```
printer.printWaterMark(5)
```

30. print

Description: Execute print. **Prerequisite** for label printing, must be called after all print content.

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:

```
printer.setSize(60, 40)
printer.setGap(2, 0)
printer.printText(50, 30, "Label Content")
printer.printQRCode(200, 150, "QR Data")
printer.print(copies = 1)
```

31. clear

Description: Clear buffer.

Method Signature:

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

Return Value: `Result<Unit>` Operation result

Example Code:

```
printer.clear()
```

32. feedPaper

Description: Feed paper.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
lines	Int	Yes	Number of lines to feed

Example Code:

```
printer.feedPaper(20)
```

33. backFeed

Description: Back feed.

Method Signature:

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

Parameters:

Parameter	Type	Required	Description
len	Int	Yes	Back feed length (mm)

Example Code:

```
printer.backFeed(10)
```

34. limitFeed

Description: Limit feed.

Method Signature:

```
suspend fun limitFeed(maxLen: Int): Result<Unit>
```

Parameters:

Parameter	Type	Required	Description
maxLen	Int	Yes	Maximum length (mm)

Example Code:

```
printer.limitFeed(100)
```

35. formFeed

Description: Feed to tear position.

Method Signature:

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

Return Value: `Result<Unit>` Operation result

Example Code:

```
printer.formFeed()
```

36. home

Description: Return to home position.

Method Signature:

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

Return Value: `Result<Unit>` Operation result

Example Code:

```
printer.home()
```

37. cutPaper

Description: Cut paper.

Method Signature:

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

Return Value: `Result<Unit>` Operation result

Example Code:

```
printer.cutPaper()
```

38. sound

Description: Sound beep.

Method Signature:

```
suspend fun sound(level: Int, interval: Int): Result<Unit>
```

Parameters:

Parameter	Type	Required	Description
level	Int	Yes	Volume level (1-10)
interval	Int	Yes	Interval (milliseconds)

Example Code:

```
printer.sound(level = 5, interval = 200)
```

39. selfTest

Description: Execute self-test.

Method Signature:

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

Return Value: `Result<Unit>` Operation result

Example Code:

```
printer.selfTest()
```

40. Appendix

40.1 Font Enum (TsplFont)

Value	Description
FONT_0	Font 0 (8x12)
FONT_1	Font 1 (12x16)
FONT_2	Font 2 (16x24)
FONT_3	Font 3 (24x32)
FONT_4	Font 4 (32x48)
FONT_5	Font 5 (14x19)
FONT_6	Font 6 (21x27)
FONT_7	Font 7 (28x36)
FONT_8	Font 8 (56x72)
FONT_9	Font 9 (21x27)
FONT_10	Font 10 (16x20)
FONT_ROMAN	Roman font
FONT_SANS	Sans-serif font

40.2 Rotation Enum (Rotation)

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

40.3 Barcode Type Enum (BarcodeType)

Value	Description
CODE128	Code 128
CODE128A	Code 128 A
CODE128B	Code 128 B
CODE128C	Code 128 C
CODE39	Code 39
CODE93	Code 93
EAN13	EAN-13
EAN8	EAN-8
UPC_A	UPC-A
UPC_E	UPC-E
CODABAR	Codabar
ITF	ITF

40.4 Dithering Algorithm Enum (DitheringAlgorithm)

Value	Description
THRESHOLD	Simple threshold
DIFFUSION	Diffusion dithering
CLUSTER	Cluster dithering

40.5 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("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}")
            }
        }
    }
}
```

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