

# Class

## XZPLCommand

```
class XZPLCommand: NSObject
```

### getCommand()

#### Get the Print Command

```
func getCommand() -> Data!
```

#### Description

Retrieve the current ZPL object's print data.

#### Return Value

- NSData object containing the print data.

### setCharEncoding(\_:)

#### Set Character Encoding

```
func setCharEncoding(_ encoding: String.Encoding) -> XZPLCommand!
```

#### Parameters

- `encoding`  
Encoding format, default is GB\_18030\_2000.

#### Return Value

- XZPLCommand object.

### addData(\_:)

#### Add Custom Data

```
func addData(_ customData: Data) -> XZPLCommand!
```

## Parameters

- `customData`  
Custom data content.

## Return Value

- XZPLCommand object.
- 

## start()

### Start of the Label

```
func start() -> XZPLCommand!
```

## Description

This method is used for the beginning of the label.

**Note:** The start command must be added at the beginning of the print content.

## Return Value

- XZPLCommand object.
- 

## end()

### End of the Label

```
func end() -> XZPLCommand!
```

## Description

End of the label format. Calling this method will print the label.

**Note:** The end command must be added at the end of the print content.

## Return Value

- XZPLCommand object.
- 

## addTextAt(x:y:content:)

---

# Text Printing

```
func addTextAt(x: Int, y: Int, content: String) -> XZPLCommand!
```

## Parameters

- `x`  
Starting x value of the text.
- `y`  
Starting y value of the text.
- `content`  
Text content.

## Return Value

- XZPLCommand object.

---

## addTextAt(x:y:fontName:content:)

# Text Printing

```
func addTextAt(x: Int, y: Int, fontName: String, content: String) -> XZPLCommand!
```

## Parameters

- `x`  
Starting x value of the text.
- `y`  
Starting y value of the text.
- `fontName`  
Text font type, default is FNT\_F.

Variable	Basic Font Size (Height x Width)
FNT_A	9 x 5
FNT_B	11 x 7
FNT_C	18 x 10
FNT_D	18 x 10
FNT_E	28 x 15
FNT_F	26 x 13
FNT_G	60 x 40
FNT_O	15 x 12

- `content`  
Text content.

## Return Value

- XZPLCommand object.

# addTextAt(x:y:fontName:sizeW:sizeH:content:)

## Text Printing

```
func addTextAt(x: Int, y: Int, fontName: String, sizeW: Int, sizeH: Int, content: String) -> XZPLCommand!
```

## Parameters

- `x`  
Starting x value of the text.
- `y`  
Starting y value of the text.
- `fontName`  
Text font type, default is FNT\_F.
- `sizeW`  
Effective text width, default is the basic size. Use integer multiples of the basic size.
- `sizeH`  
Effective text height, default is the basic size. Use integer multiples of the basic size.
- `content`  
Text content.

## Return Value

- XZPLCommand object.

---

# addTextAt(x:y:fontName:rotation:sizeW:sizeH:content:)

## Text Printing

```
func addTextAt(x: Int, y: Int, fontName: String, rotation: RotationZPL, sizeW: Int, sizeH: Int, content: String) -> XZPLCommand!
```

## Parameters

- `x`  
Starting x value of the text.
- `y`  
Starting y value of the text.
- `fontName`  
Text font type, default is FNT\_F.
- `rotation`  
Clockwise rotation angle, default is ZPLRotation0.

Variable	Description
ZPLRotation0	No rotation
ZPLRotation90	Rotate 90 degrees
ZPLRotation180	Rotate 180 degrees
ZPLRotation270	Rotate 270 degrees

- `sizeW`  
Effective text width, default is the basic size. Use integer multiples of the basic size.
- `sizeH`  
Effective text height, default is the basic size. Use integer multiples of the basic size.
- `content`  
Text content.

## Return Value

- XZPLCommand object.

---

# setCustomFont(\_:extension:alias:codePage:)

## Set Custom Font

```
func setCustomFont(_ fontName: String, extension: String, alias: String, codePage: CodePageZPL) -> XZPLCommand!
```

### Parameters

- `fontName`  
Font library name.
- `extension`  
Font name suffix.
- `alias`  
Font alias, corresponding to the `fontName` in `addText`. Range: `A` to `Z` and `0` to `9`.
- `codePage`  
Refer to the `CodePageZPL` enumeration for details:

Value	Encoding (Code Page)	Code Page Number
ZPLCodePageUSA1	USA 1	0 (Code Page 850)
ZPLCodePageUTF8	UTF-8	28

### Return Value

- XZPLCommand object.

---

## setPrinterWidth(\_:)

### Set Printer Width

```
func setPrinterWidth(_ width: Int) -> XZPLCommand!
```

### Parameters

- `width`  
Paper width in dots.

### Return Value

- XZPLCommand object.

---

## setLabelLength(\_:)

## Set Label Length

```
func setLabelLength(_ height: Int) -> XZPLCommand!
```

### Parameters

- `height`  
Label length in dots. Range: `1` to `32000`, not exceeding the maximum label size.

### Return Value

- XZPLCommand object.
- 

## addReverseAt(x:y:width:height:)

### Reverse Area

```
func addReverseAt(x: Int, y: Int, width: Int, height: Int) -> XZPLCommand!
```

### Parameters

- `x`  
Starting x value of the area.
- `y`  
Starting y value of the area.
- `width`  
Area width.
- `height`  
Area height.

### Return Value

- XZPLCommand object.
- 

## addReverseAt(x:y:width:height:radius:)

### Reverse Area with Rounded Corners

```
func addReverseAt(x: Int, y: Int, width: Int, height: Int, radius: Int) -> XZPLCommand!
```

## Parameters

- `x`  
Starting x value of the area.
- `y`  
Starting y value of the area.
- `width`  
Area width.
- `height`  
Area height.
- `radius`  
Rounding degree, range: `0~8`, default is `0`.

## Return Value

- XZPLCommand object.
- 

## addBoxAt(x:y:width:height:thickness:)

---

## Draw Rectangle

```
func addBoxAt(x: Int, y: Int, width: Int, height: Int, thickness: Int) -> XZPLCommand!
```

## Parameters

- `x`  
Starting x value of the rectangle.
- `y`  
Starting y value of the rectangle.
- `width`  
Rectangle width.
- `height`  
Rectangle height.
- `thickness`  
Line thickness.

## Return Value

- XZPLCommand object.
- 

## addBoxAt(x:y:width:height:thickness:radius:)

---



# Draw Rectangle with Rounded Corners

```
func addBoxAt(x: Int, y: Int, width: Int, height: Int, thickness: Int, radius: Int) -> XZPLCommand!
```

## Parameters

- `x`  
Starting x value of the rectangle.
- `y`  
Starting y value of the rectangle.
- `width`  
Rectangle width.
- `height`  
Rectangle height.
- `thickness`  
Line thickness.
- `radius`  
Rounding degree, range: `0~8`, default is `0`.

## Return Value

- XZPLCommand object.

---

# addGraphicDiagonalLineAt(x:y:orientation:width:height:thickness:)

## Draw Diagonal Line

```
func addGraphicDiagonalLineAt(x: Int, y: Int, orientation: DiagonalDirection, width: Int, height: Int, thickness: Int) -> XZPLCommand!
```

## Parameters

- `x`  
Horizontal starting position.
- `y`  
Vertical starting position.
- `orientation`  
Direction of the diagonal line.

Variable	Description
DiagonalDirectionRight	Right-leaning diagonal line (or <code>/</code> )
DiagonalDirectionLeft	Left-leaning diagonal line (or <code>\</code> )

- `width`  
Frame width (range: `1-32000`, unit: dot).
- `height`  
Frame height (range: `1-32000`, unit: dot).
- `thickness`  
Border thickness (range: `1-32000`, unit: dot).

## Return Value

- XZPLCommand object.

---

# addGraphicEllipseAt(x:y:width:height:thickness:)

---

## Graphic Ellipse

```
func addGraphicEllipseAt(x: Int, y: Int, width: Int, height: Int, thickness: Int) ->
XZPLCommand!
```

## Parameters

- `x`  
Horizontal starting position.
- `y`  
Vertical starting position.
- `width`  
Ellipse width, range: `3-4095`, unit: dot.
- `height`  
Ellipse height, range: `3-4095`, unit: dot.
- `thickness`  
Border thickness, range: `2-4095`, unit: dot.

## Return Value

- XZPLCommand object.

---

# addGraphicCircleAt(x:y:diameter:thickness:)

---

# Graphic Circle

```
func addGraphicCircleAt(x: Int, y: Int, diameter: Int, thickness: Int) -> XZPLCommand!
```

## Parameters

- `x`  
Horizontal starting position.
- `y`  
Vertical starting position.
- `diameter`  
Circle diameter, range: 3–4095, unit: dot.
- `thickness`  
Border thickness, range: 1–4095, unit: dot.

## Return Value

- XZPLCommand object.

---

# addBarcodeAt(x:y:codeType:content:)

## 1D Barcode

```
func addBarcodeAt(x: Int, y: Int, codeType: ZPLBarcodeType, content: String) -> XZPLCommand!
```

## Parameters

- `x`  
Starting x value of the barcode.
- `y`  
Starting y value of the barcode.
- `codeType`  
Barcode type. See the `ZPLBarcodeType` enumeration:

Variable	Description
ZPLBarCode11	Code 11
ZPLBarCode25	Code 25
ZPLBarCode39	Code 39
ZPLBarCodeEAN8	EAN-8
ZPLBarCodeUPCE	UPC-E
ZPLBarCode93	Code 93
ZPLBarCode128	Code 128
ZPLBarCodeEAN13	EAN-13
ZPLBarCodeCODA	CODABAR
ZPLBarCodeMSI	MSI
ZPLBarCodePLESSEY	PLESSEY
ZPLBarCodeUPCEAN	UPC-EAN
ZPLBarCodeUPCA	UPC-A

- `content`  
Barcode text content.

## Return Value

- XZPLCommand object.

# addBarcodeAt(x:y:codeType:content:height:)

## Add 1D Barcode

```
func addBarcodeAt(x: Int, y: Int, codeType: ZPLBarCodeType, content: String, height: Int) -> XZPLCommand!
```

## Parameters

- `x`  
Starting x value of the barcode.
- `y`  
Starting y value of the barcode.
- `codeType`  
Barcode type. See the specific `ZPLBarCodeType` enumeration.

- `content`  
Barcode text content.
- `height`  
Barcode height. Default is 50 dots.

## Return Value

- XZPLCommand object.

# addBarcodeAt(x:y:codeType:ratio:textPosition:content:width:height:)

## Add 1D Barcode with Additional Options

```
func addBarcodeAt(x: Int, y: Int, codeType: ZPLBarCodeType, ratio: RotationZPL,
  textPosition: HriTextZPL, content: String, width: Int, height: Int) -> XZPLCommand!
```

## Parameters

- `x`  
Starting x value of the barcode.
- `y`  
Starting y value of the barcode.
- `codeType`  
Barcode type. See the specific `ZPLBarCodeType` enumeration.
- `ratio`  
Rotation angle. See the `RotationZPL` enumeration.
- `textPosition`  
Text position relative to the barcode. Default is `ZPLHriTextBelow`.  
Possible values:

Variable	Description
<code>ZPLHriTextNone</code>	No human-readable text
<code>ZPLHriTextBelow</code>	Human-readable text below the barcode
<code>ZPLHriTextAbove</code>	Human-readable text above the barcode

- `content`  
Barcode text content.
- `width`  
Barcode module width. Default is 2 dots.

- `height`

Barcode height. Default is 50 dots.

## Return Value

- XZPLCommand object.
- 

# addQRCodeAt(x:y:factor:text:)

---

## QR Code

```
func addQRCodeAt(x: Int, y: Int, factor: Int, text: String) -> XZPLCommand!
```

## Parameters

- `x`  
Starting x value of the QR code.
- `y`  
Starting y value of the QR code.
- `factor`  
Magnification factor, range: 1-10, default is 3.
- `text`  
QR code content.

## Return Value

- XZPLCommand object.
- 

# printImageAt(x:y:image:)

---

## Print Image

```
func printImageAt(x: Int, y: Int, image: UIImage) -> XZPLCommand
```

## Parameters

- `x`  
Starting x value of the image.
- `y`  
Starting y value of the image.
- `image`  
Image object to print.

## Return Value

- XZPLCommand object.
- 

# printImageAt(x:y:wRatio:hRatio:image:)

---

## Print Image

```
func printImageAt(x: Int, y: Int, wRatio: Int, hRatio: Int, image: NSImage) ->
XZPLCommand!
```

## Parameters

- `x`  
Starting x value of the image.
- `y`  
Starting y value of the image.
- `wRatio`  
Width scaling ratio, range: 1–10.
- `hRatio`  
Height scaling ratio, range: 1–10.
- `image`  
Image object.

## Return Value

- XZPLCommand object.
- 

# downloadGraphic(\_:name:image:)

---

## Download Graphic

```
func downloadGraphic(_ source: ZPLDeviceType, name: String, image: NSImage) ->
XZPLCommand!
```

## Parameters

- `source`  
Storage device for the image. Default is `DeviceTypeR`. Optional types are defined in `ZPLDeviceType`.
- `name`  
Name of the stored image. Accepted values: 1 to 8 alphanumeric characters. If no name is specified, `UNKNOWN` is used.

- `image`  
Image object.

## Return Value

- XZPLCommand object.

---

# printGraphicAt(x:y:source:name:mx:my:)

## Call Graphic

```
func printGraphicAt(x: Int, y: Int, source: ZPLDeviceType, name: String, mx: Int, my: Int) -> XZPLCommand!
```

## Parameters

- `x`  
Starting x value of the image.
- `y`  
Starting y value of the image.
- `source`  
Storage device for the image. Default is `DeviceTypeR`. Optional types are defined in `ZPLDeviceType`.
- `name`  
Name and extension obtained when downloading the image.
- `mx`  
Magnification factor in the x-axis direction. Default is `1`, range: 1–10.
- `my`  
Magnification factor in the y-axis direction. Default is `1`, range: 1–10.

## Return Value

- XZPLCommand object.

---

# deleteDownloadGraphic(\_:name:)

## Delete Downloaded Graphic

```
func deleteDownloadGraphic(_ source: ZPLDeviceType, name: String) -> XZPLCommand!
```



## Parameters

- `source`  
Storage device for the image. Default is `DeviceTypeR`. Optional types are defined in `ZPLDeviceType`.
- `name`  
Name of the stored image. Accepted values: 1 to 8 alphanumeric characters. If no name is specified, `UNKNOWN` is used.

## Return Value

- XZPLCommand object.
- 

## addPrintCount(\_:)

---

### Set Print Quantity

```
func addPrintCount(_ count: Int) -> XZPLCommand!
```

## Parameters

- `count`  
Number of labels to print.

## Return Value

- XZPLCommand object.
- 

## setPrintSpeed(\_:)

---

### Set Print Speed

```
func setPrintSpeed(_ speed: Int) -> XZPLCommand!
```

## Parameters

- `speed`  
Print speed in inches per second, range: 1–14.

## Return Value

- XZPLCommand object.
- 

## direction(\_:)

---

# Rotate Label

```
func direction(_ n: Bool) -> XZPLCommand!
```

## Parameters

- `n`  
YES = normal orientation, NO = rotate 180 degrees. Default: YES.

## Return Value

- XZPLCommand object.
- 

# setPrintDensity(\_:)

## Set Print Density

```
func setPrintDensity(_ density: Int) -> XZPLCommand!
```

## Parameters

- `density`  
Print density, range: 0–30.

## Return Value

- XZPLCommand object.
-