

Webdemo Printer Command List

Version 1.0

Version	Date	Description
1.0	2025-08-14	First release

No part of this manual may be reproduced or transmitted in any form or by any means without prior written consent of ShenZhen Pay Device Technology Co., Ltd.

Contents

1. Overview.....	3
2. Quick Start.....	5
3. Workflow.....	6
4. API.....	7

1. Overview

This document introduces the printing API in webdemo. You can modify below core source code as follow in the WebSDK If you want customize some API.

Websocket proxy APK:

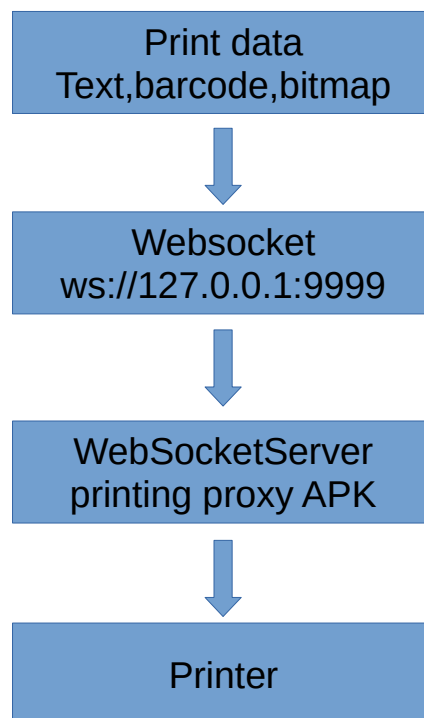
[WebSocketServer/app/src/main/java/com/paydevice/websocketserver/proxy/PrinterProxy.java](#)

[WebSocketServer/app/src/main/java/com/paydevice/websocketserver/SmartPosProxyService.java](#)

Webdemo javascript class:

[webdemo/js/printer.js](#)

The print data always send to printer via local websocket, the data flow chart as below.



if you want use websocket directly, below is json format:

The websocket send command json format:

```
{"dev":"printer","cmd":"checkPaper","args":[]}
```

```
{"dev":"printer","cmd":"sendData","args":["Hello,World!"]}
```

The websocket onmessage callback json format:

```
{"dev":"printer","cmd":"checkPaper","ack":"true"}
```

field "dev" always "printer".

field "cmd" is API name in API list.

field "args" depends on the API in "cmd" field whether requires arguments, its always an array and is empty array if no need arguments.

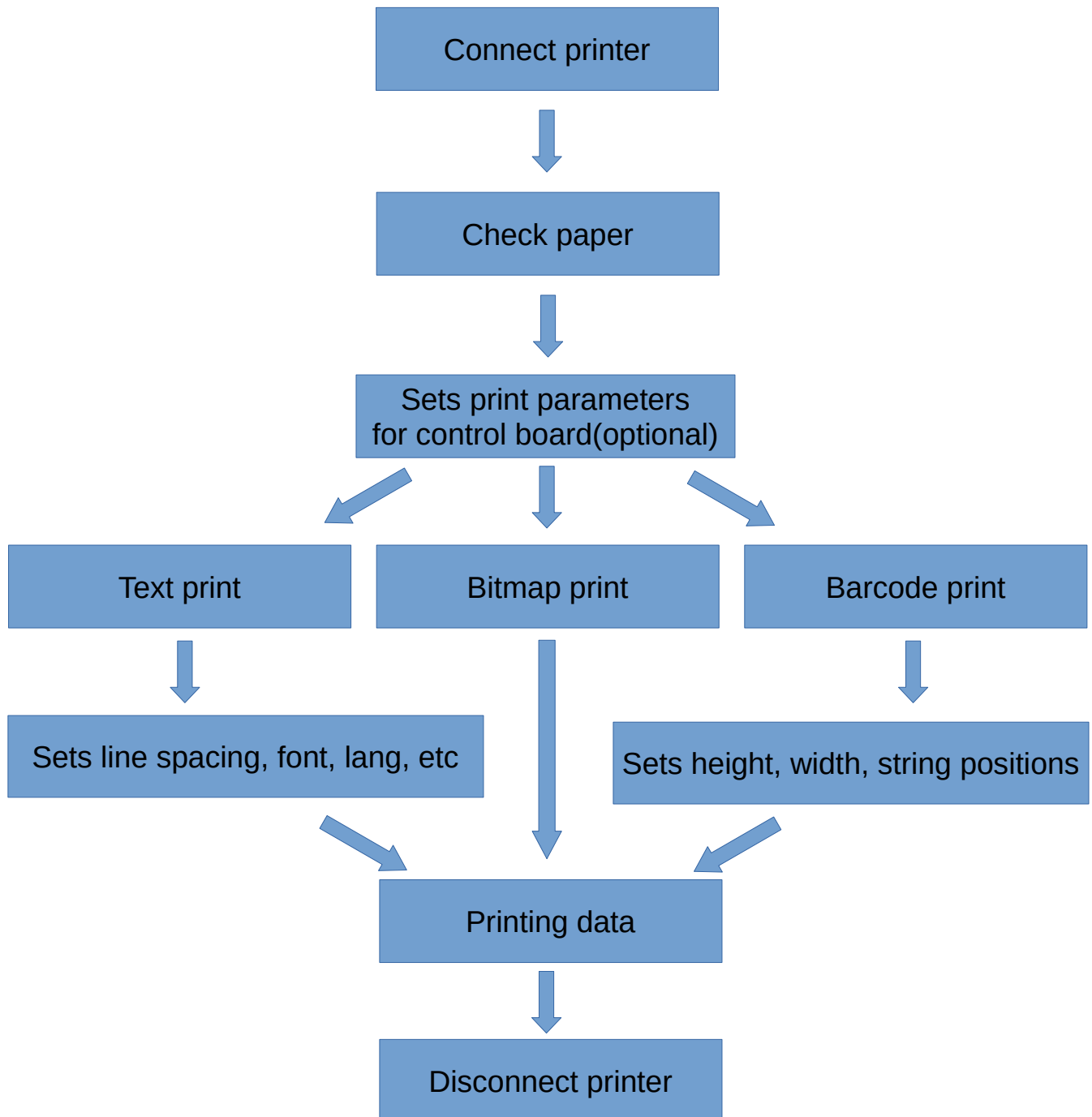
field "ack" depends on the API in cmd field wether have return value.

2. Quick Start

The following example shows how to printing "Hello, World!".

```
<html lang="en">
<head>
  <script type="text/javascript" charset="utf-8">
    const ws = new WebSocket("ws://127.0.0.1:9999");
    ws.onopen = function() {
      var cmd = {dev:"printer",cmd:"connectBuiltinPrinter",args:[58]}; // connect printer
      ws.send(JSON.stringify(cmd));
      cmd = {dev:"printer",cmd:"checkPaper",args:[null]}; // check for paper
      ws.send(JSON.stringify(cmd));
    }
    ws.onmessage = function(msg) {
      var json = JSON.parse(msg.data);
      if (json.dev === "printer") {
        if (json.cmd === "checkPaper") {
          if (json.ack === "true") {
            var cmd = {dev:"printer",cmd:"sendData",args:["Hello,World!"]}; // printing
            ws.send(JSON.stringify(cmd));
            cmd = {dev:"printer",cmd:"cmdLineFeed",args:["5"]}; // feed 5 lines
            ws.send(JSON.stringify(cmd));
            md = {dev:"printer",cmd:"disconnect"}; // disconnect printer
            ws.send(JSON.stringify(cmd));
            ws.close() // disconnect websocket
          } else {
            console.log("no paper");
          }
        }
      }
    }
  </script>
</head>
</html>
```

3. Workflow



4. API

Glossary

dots: The number of print dots that is the smallest unit of print.

1 dot = 0.125 mm

max printing width:

58mm printer: 48mm(384dots * 0.125mm), maximum dots is 384

80mm printer: 72mm(576dots * 0.125mm), maximum dots is 576

HRI: Barcode human readable interpretation

Constant definition

Cut paper mode, valid when printer have cutter

```
const FULL_CUT = 0;
```

```
const HALF_CUT = 1;
```

Align mode

```
const ALIGN_LEFT = 0;           //align left
```

```
const ALIGN_MIDDLE = 1;         //align middle
```

```
const ALIGN_RIGHT = 2;          //align right
```

Print mode, valid for text printing

```
const FONT_DEFAULT = 0;          //font 12x24
```

```
const FONT_SMALL = 1<<0;        //font 9x17, valid for ascii character
```

```
const FONT_INVERSE = 1<<1;      //white/black reverse
```

```
const FONT_UPSIDE_DOWN = 1<<2;  //upside down
```

```
const FONT_EMPHASIZED    = 1<<3; //bold
const FONT_DOUBLE_HEIGHT = 1<<4; //double height
const FONT_DOUBLE_WIDTH  = 1<<5; //double width
const FONT_ROTATE         = 1<<6;  //rotate 90° clockwise
const FONT_UNDERLINE     = 1<<7;   //enable underline
```

Underline height

```
const UNDERLINE_ZERO = 0; //no underline
const UNDERLINE_HIGH_1 = 1; //hight 1 dots
const UNDERLINE_HIGH_2 = 2; //hight 2 dots
```

Barcode HRI positions

```
const CODEBAR_STRING_MODE_NONE = 0; //no HRI
const CODEBAR_STRING_MODE_ABOVE = 1; //barcode above
const CODEBAR_STRING_MODE_BELOW = 2; //barcode below
const CODEBAR_STRING_MODE_BOTH = 3; //above and below
const CODEBAR_STRING_FONT_A = 0; //HRI font size A
const CODEBAR_STRING_FONT_B = 1; //HRI font size B
```

1D barcode encoding format

```
const UPC_A = 65;
const UPC_E = 66;
const EAN13 = 67;
const EAN8 = 68;
const CODE39 = 69;
```



```
const I25 = 70;
const CODEBAR = 71;
const CODE93 = 72;
const CODE128 = 73;
const CODE11 = 74;
const MSI = 75;
```

QR correction level

```
const QR_ECC_LEVEL_L = 1;
const QR_ECC_LEVEL_M = 2;
const QR_ECC_LEVEL_Q = 3;
const QR_ECC_LEVEL_H = 4;
```

Code page, Details refer: https://en.wikipedia.org/wiki/Code_page

```
const CODE_PAGE_CP437    = 0;//English
const CODE_PAGE_KATAKANA = 1;//Japanese
const CODE_PAGE_CP850    = 2;//Western Europe
const CODE_PAGE_CP860    = 3;//Portuguese
const CODE_PAGE_CP863    = 4;//Canadian-French
const CODE_PAGE_CP865    = 5;//Nordic
const CODE_PAGE_CP1251   = 6;//Cyrillic
const CODE_PAGE_CP866    = 7;//Cyrillic2
const CODE_PAGE_MIK      = 8;//Cyrillic,Bulgarian
const CODE_PAGE_CP755    = 9;//East Europe, Latvian2
const CODE_PAGE_IRAN     = 10;//Iran System encoding standard
```

```
const CODE_PAGE_CP862      = 15; //Hebrew
const CODE_PAGE_CP1252     = 16; //Latin1
const CODE_PAGE_CP1253     = 17; //Greek
const CODE_PAGE_CP852      = 18; //Latin2
const CODE_PAGE_CP858      = 19; //West Europe
const CODE_PAGE_IRAN2      = 20; //Iran2
const CODE_PAGE_LATVIAN    = 21; //Latvian
const CODE_PAGE_CP864      = 22; //Arabic
const CODE_PAGE_ISO_8859_1 = 23; //West Europe
const CODE_PAGE_CP737      = 24; //Greek
const CODE_PAGE_CP1257     = 25; //Baltic
const CODE_PAGE_THAI       = 26; //Thai1
const CODE_PAGE_CP720      = 27; //Arabic
const CODE_PAGE_CP855      = 28; //Cyrillic script
const CODE_PAGE_CP857      = 29; //Turkish
const CODE_PAGE_CP1250     = 30; //Central Europe
const CODE_PAGE_CP775      = 31; //Estonian, Lithuanian, Latvian
const CODE_PAGE_CP1254     = 32; //Turkish
const CODE_PAGE_CP1255     = 33; //Hebrew
const CODE_PAGE_CP1256     = 34; //Arabic
const CODE_PAGE_CP1258     = 35; //Vietnamese
const CODE_PAGE_ISO_8859_2 = 36; //Latin2
const CODE_PAGE_ISO_8859_3 = 37; //Latin3
const CODE_PAGE_ISO_8859_4 = 38; //Baltic
const CODE_PAGE_ISO_8859_5 = 39; //Cyrillic
```

```
const CODE_PAGE_ISO_8859_6 = 40;//Arabic
const CODE_PAGE_ISO_8859_7 = 41;//Greek
const CODE_PAGE_ISO_8859_8 = 42;//Hebrew
const CODE_PAGE_ISO_8859_9 = 43;//Turkish
const CODE_PAGE_ISO_8859_15 = 44;//Latin9
const CODE_PAGE_THAI2      = 45;//Thai2
const CODE_PAGE_CP856       = 46;//Hebrew
const CODE_PAGE_CP874       = 47;//Thai
const CODE_PAGE_SHIFT_JIS   = 96;
const CODE_PAGE_EUC_KR      = 97;
const CODE_PAGE_BIG5        = 98;//Traditional Chinese
const CODE_PAGE_GB18030     = 99;//Simplified Chinese
```

Bitmap zoom mode, valid for API cmdPrintBitmapFromNVRAM

```
const BITMAP_ZOOM_NONE = 0;
const BITMAP_ZOOM_WIDTH = 1;
const BITMAP_ZOOM_HEIGHT = 2;
const BITMAP_ZOOM_BOTH = 3;
```

API details

* Connect the built-in printer, such as model FH100-A3-D, FH070A1-S

`connectBuiltinPrinter(paperWidth)`

parameters: `paperWidth` the paper width, such as 58 or 80

example:

send json to websocket

```
{"dev":"printer","cmd":"connectBuiltinPrinter","args":[58]}
```

websocket callback

```
{"dev":"printer","cmd":"connectBuiltinPrinter","ack":"true"}
```

`true` connect succeeded

`false` connect failed

* Connect external serial port printer

`connectSerialPrinter(port, baudrate, paperWidth)`

parameters: `port` the serial port name, such as `"/dev/ttyS9"`

`baudrate` the serial port baudrate such as 115200

`paperWidth` the paper width, such as 58 or 80

example:

send json to websocket

```
{"dev":"printer","cmd":"connectSerialPrinter","args":["/dev/ttyS9",115200,58]}
```

websocket callback

```
{"dev":"printer","cmd":"connectSerialPrinter","ack":"true"}
```

`true` connect succeeded

`false` connect failed

* Connect external usb printer

connectUsbPrinter(vid, pid, paperWidth)

parameters: **vid** the vendor id of the usb printer, decimal integer
pid the product id of the usb printer, decimal integer
paperWidth the paper width, such as 58 or 80

example:

send json to websocket

```
{"dev":"printer","cmd":"connectUsbPrinter","args":[17992,32768,80]}
```

websocket callback

```
{"dev":"printer","cmd":"connectUsbPrinter","ack":"true"}
```

true connect succeeded

false connect failed

* Disconnect printer

disconnect()

Note: Serialport printer need wait print finish before disconnect. Also remember invoke disconnect after print finish.

example:

send json to websocket

```
{"dev":"printer","cmd":"disconnect","args":[]}
```

* Check paper status

checkPaper()

example:

send json to websocket

```
{"dev":"printer","cmd":"checkPaper","args":[]}
```

websocket callback

```
{"dev":"printer","cmd":"checkPaper","ack":"true"}
```

return value in ack field:

true have paper

false no paper

* Get printer type

getPrinterType()

example:

send json to websocket

```
{"dev":"printer","cmd":"getPrinterType","args":[]}
```

websocket callback

```
{"dev":"printer","cmd":"getPrinterType","ack":"serial"}
```

return value in ack field:

serial serialport printer, such as built in printer in model FH100-A3-D, FH070A1-S, etc.

usb USB printer

* Jump next tab position

cmdJumpTab()

Note: tab position depends on **cmdSetTable(offsets)**

example:

send json to websocket

```
{"dev":"printer","cmd":"cmdSetTable","args":["2,9"]}
```

```
{"dev":"printer","cmd":"cmdJumpTab","args":[]}
```

```
{"dev":"printer","cmd":"sendData","args":["item1"]}
```

```
{"dev":"printer","cmd":"cmdJumpTab","args":[]}
```

```
{"dev":"printer","cmd":"sendData","args":["item2"]}
{"dev":"printer","cmd":"cmdUnSetTable","args":[]}
```

*Sets horizontal table position

cmdSetTable(offsets)

parameters: [offset](#) position array, array length range 1-16. tab unit is 9 dots for 9x17 font or 12 dots for 12x24 font.

example:

send json to websocket

```
{"dev":"printer","cmd":"cmdSetTable","args":["2,9,19"]}
```

incorrect format:

```
{"dev":"printer","cmd":"cmdSetTable","args":[2,9,19]}
```

it means table have 3 column, there are three offsets 2,9,19

*Unset horizontal table position

cmdUnSetTable()

example:

send json to websocket

```
{"dev":"printer","cmd":"cmdUnSetTable","args":[]}
```

* To feed line

cmdLineFeed()

Note: The line spacing depends on [cmdSetDefaultLineSpacing\(\)](#) and [cmdSetLineSpacing\(n\)](#)

example:

send json to websocket

```
{"dev":"printer","cmd":"cmdLineFeed","args":[]}
```

cmdLineFeed(n)

parameters: **n** the number of the lines

Note: The line spacing depends on [cmdSetDefaultLineSpacing\(\)](#) and [cmdSetLineSpacing\(n\)](#)

example:

send json to websocket

```
{"dev":"printer","cmd":"cmdLineFeed","args":[5]}
```

* Set global character encoding

setStringEncoding(encoding)

parameters: **encoding** character encoding

example:

send json to websocket

```
{"dev":"printer","cmd":"setStringEncoding","args":["CP437"]}
```

* Send text

sendData(data)

parameters: **data** text string

Note: Termal printer used code page for different language and all text need to be converted into the corresponding encoding then the printer could printing correctly.

Typically, we only need to print in one language, you simply need set the global string encoding by function [setStringEncoding\(encoding\)](#) and [cmdSetPrinterLanguage\(code\)](#) then invoke function [sendData\(data\)](#) to print.

example:

send json to websocket

```
{"dev":"printer","cmd":"sendData","args":[]}
```

* Set the line spacing to default

[cmdSetDefaultLineSpacing\(\)](#)

Note: Default line spacing is 32 dots

example:

send json to websocket

```
{"dev":"printer","cmd":"cmdSetDefaultLineSpacing","args":[]}
```

* Set the line spacing to n dots

[cmdSetLineSpacing\(dots\)](#)

parameters: [dots](#) number of the dots

example:

send json to websocket

```
{"dev":"printer","cmd":"cmdSetLineSpacing","args":[40]}
```

* Set the align mode

[cmdSetAlignMode\(mode\)](#)

parameters: [mode](#) align mode

mode=ALIGN_LEFT	align left, default mode
mode=ALIGN_MIDDLE	align middle
mode=ALIGN_RIGHT	align right

example:

send json to websocket

```
{"dev":"printer","cmd":"cmdSetAlignMode","args":[0]}
```

* Set print offset

`cmdSetPrintOffset(offset)`

parameters: `offset` start position offset dots

Note: Only valid in current text line

example:

send json to websocket

```
{"dev":"printer","cmd":"cmdSetPrintOffset","args":[48]}
```

* Set print mode

`cmdSetPrintMode(mode)`

parameters: `mode`

bit 0: default(set 1 used small font)

bit 1: black/white inverse

bit 2: upside down

bit 3: emphasized

bit 4: double height

bit 5: double width

bit 6: rotate

bit 7: underline

Note: Set the corresponding bit to 1 takes effect. The underline no effect if set rotated 90° clockwise or inverse

example:

send json to websocket

```
{"dev":"printer","cmd":"cmdSetPrintMode","args":[0]}
```

* Set the height of underline

cmdSetUnderlineHeight(n)

parameters: **n** dots

n=UNDERLINE_ZERO no underline

n=UNDERLINE_HIGH_1 height is 1 dots

n=UNDERLINE_HIGH_2 height is 2 dots

example:

send json to websocket

```
{"dev":"printer","cmd":"cmdSetUnderlineHeight","args":[2]}
```

* Set the multiple scale for font's width and height

cmdSetFontScaleSize(wScale, hScale)

parameters: **wScale** width multiple 0~7

hScale height multiple 0~7

example:

send json to websocket

```
{"dev":"printer","cmd":"cmdSetFontScaleSize","args":[2,2]}
```

* Set the position of the barcode string(HRI)

cmdSetBarCodeStringPosition(position)

parameters: **position** position of the barcode string

position=CODEBAR_STRING_MODE_NONE no string

position=CODEBAR_STRING_MODE_ABOVE above the barcode

position=CODEBAR_STRING_MODE_BELOW below the barcode

position=CODEBAR_STRING_MODE_BOTH both above and below the barcode

example:

send json to websocket

```
{"dev":"printer","cmd":"cmdSetBarCodeStringPosition","args":[2]}
```

* Set the height of the barcode

[cmdSetBarCodeHeight\(n\)](#)

parameters: [n](#) height(unit: dots)

$1 \leq n \leq 255$ (default is: 50 dots)

example:

send json to websocket

```
{"dev":"printer","cmd":"cmdSetBarCodeHeight","args":[80]}
```

* Set the width of the basic line

[cmdSetBarCodeWidth\(n\)](#)

parameters: [n](#) width(unit: dots)

$n = 2$ or $n = 3$ (default is: 3 dots)

Note: some long CODE128 try set $n=2$ to reduce total width of the barcode.

example:

send json to websocket

```
{"dev":"printer","cmd":"cmdSetBarCodeWidth","args":[2]}
```

* Set the left spacing of the barcode

cmdSetBarCodeLeftSpacing(n)

parameters: **n** left spacing(unit: dots)

example:

send json to websocket

```
{"dev":"printer","cmd":"cmdSetBarCodeLeftSpacing","args":[24]}
```

* Printing barcode

cmdBarCodePrint(type, text)

parameters: **type** the encoding of the barcode, such as CODE128

txt HRI of the barcode

example:

send json to websocket

```
{"dev":"printer","cmd":"cmdBarCodePrint","args":[72,"ABC1234567890"]}
```

* Printing bitmap

cmdBitmapPrint(path, leftOffset, topOffset)

parameters: **path** bmp saved path in the system storage

leftOffset horizontal offset dots

topOffset vertical offset dots

Note: limited $\text{leftOffset} + \text{bmp width} < 384$ (smaller then 576 for 80mm printer)

example:

send json to websocket

```
{"dev":"printer","cmd":"cmdBitmapPrint","args":["Download/logo.bmp",0,0]}
```

* Print bitmap in split mode

`cmdBitmapPrintEx(path, leftOffset, topOffset)`

parameters: `path` bmp saved path in the system storage

`leftOffset` horizontal offset dots

`topOffset` vertical offset dots

Note: bitmap height must be an integer multiple of 8, the printer will split the bitmap into multiple parts with height 24 dots and print them one by one.

example:

send json to websocket

```
{"dev":"printer","cmd":"cmdBitmapPrintEx","args":["Download/logo.bmp",0,0]}
```

* Set heating parameter of the printer control board

`cmdSetHeatingParam(dots, time, interval)`

parameters: `dots` maximum heating dots, range 0-255, unit 8dots. default is 7

`time` heating time, range 0-255, unit 10us. default is 80

`interval` heating time interval, range 0-255, unit 10us. default is 2

Note: This feature is typically used to adjust print clarity. Formula for maximum heating dots: $8 * (n + 1)$

The more heating dots are enabled, the higher the peak current consumption during printing and the faster the printing speed. However, excessive current may exceed the maximum operating current of the power adapter.

The longer the heating time, the higher the print density, but the slower the printing speed. If the heating time is too short, blank pages may occur.

The longer the heating interval, the clearer the print, but the slower the printing speed.

Generally, there is no need adjust this parameter, and some control board may not support this feature.

example:

send json to websocket

```
{"dev":"printer","cmd":"cmdSetHeatingParam","args":[80,100,10]}
```

* Set the print density

`cmdSetPrintDensity(density, delay)`

parameters: `density` print density, range 0-31(real density: 50%+5%*density)

`delay` delay time, range 0-7 (delay time: delay*250us)

Note: This function is used to set the print density. maybe different brands of paper needs different value.

example:

send json to websocket

```
{"dev":"printer","cmd":"cmdSetPrintDensity","args":[20,2]}
```

* Set the code page

`cmdSetPrinterLanguage(code)`

parameters: `code` code page, default is CODE_PAGE_CP437

example:

send json to websocket

```
{"dev":"printer","cmd":"cmdSetPrinterLanguage","args":[0]}
```

* Print QR code

`cmdQrCodePrint(version, ecc, text)`

parameters: `version` QR version 0~16(n dots x n dots square)

`ecc` QR correction level 1~4

`text` QR string data

example:

send json to websocket

```
{"dev":"printer","cmd":"cmdQrCodePrint","args":[8,2,"https://www.pay-device.com"]}
```

* Cut paper(Only valid for usb printer)

[cmdCutPaper\(mode\)](#)

parameters: [mode](#) FULL_CUT,HALF_CUT

example:

send json to websocket

```
{"dev":"printer","cmd":"cmdCutPaper","args":[1]}
```

* Print bitmap from NVRAM by index

[cmdPrintBitmapFromNVRAM\(index, zoom\)](#)

parameters: [index](#) bitmap index (1~255)

[zoom](#) zoom mode

BITMAP_ZOOM_NONE: original Size

BITMAP_ZOOM_WIDTH: double width

BITMAP_ZOOM_HEIGHT: double height

BITMAP_ZOOM_BOTH: double width and double height

example:

send json to websocket

```
{"dev":"printer","cmd":"cmdPrintBitmapFromNVRAM","args":[1,0]}
```

* Save bitmaps into NVRAM

[saveBitmaptoNVRAM\(\)](#)

Note: NVRAM is a storage in printer control board, it always hold even after power off. Bitmap width and height must be an integer multiple of 8.

Warning: You should not write NVRAM often, it maybe cause NVRAM broken.

Recommended less than 10 times in one day. You should save bitmaps only once and then just to print every times.

Usually, NVRAM only used to save large bitmaps(or have large area black area) which cannot print by `cmdBitmapPrint()` or `cmdBitmapPrintEx()`.

example:

send json to websocket

```
{"dev": "printer", "cmd": "saveBitmaptoNVRAM", "args": ["Download/logo.bmp", "Download/logo_large.bmp"]}
```

websocket callback

```
{"dev": "printer", "cmd": "saveBitmaptoNVRAM", "ack": "true"}
```

return value in ack field:

`true` save to NVRAM succeeded

`false` save to NVRAM failed

* Delete all bitmaps form NVRAM

`cmdDeleteBitmapFromNVRAM()`

example:

send json to websocket

```
{"dev": "printer", "cmd": "cmdDeleteBitmapFromNVRAM", "args": []}
```

websocket callback

```
{"dev": "printer", "cmd": "cmdDeleteBitmapFromNVRAM", "ack": "true"}
```

return value in ack field:

`true` delete from NVRAM succeeded

`false` delete from NVRAM failed

* Printing helper functions

Printing one deviding line

`printDevidingLine()`

Note: For some printer's firmware maybe not support print solid line.

example:

send json to websocket

```
{"dev":"printer","cmd":"printDevidingLine","args":[]}
```

Printing text and feed one line

`printLine(text)`

parameters: `text` the text data

example:

send json to websocket

```
{"dev":"printer","cmd":"printLine","args":["Hello,World!"]}
```

Printing two columns of text in one line and first column of text is align left, the second column of text is align right.

`printLineCombo(textLeft, textRight)`

parameters: `textLeft` left column of text, align left

`textRight` right column of text, align right

example:

send json to websocket

```
{"dev":"printer","cmd":"printLineCombo","args":["English","Hello World"]}
```

Printing multi-column text in one line and you can set width and align mode for each column of text.

`printLineMulti(arrStr, arrWidth, arrAlign)`

parameters: `arrStr` the text array

`arrWidth` the width ratio array

`arrAlign` the align mode array

example:

send json to websocket

```
{"dev":"printer","cmd":"printLineMulti","args":  
[["Pizza","x123","123.23"],["6,3,3"],["0,2,2"]]}
```

parameters explain:

the first column of text is "Pizza" and its width ratio is 6 and align left in current column.

the second column of text is "x123" and its width ratio is 3 and align right in current column.

the third column of text is "123.23" and its width ratio is 3 and align right in current column.

for 58mm printer the total width in one line is 384 dots, so 6:3:3 means total ratio is 6+3+3=12, then 1 ratio value is $384 \div 12 = 32$ dots.

first column width is $32 \times 6 = 192$ dots

second column width is $32 \times 3 = 96$ dots

third column width is $32 \times 3 = 96$ dots

In fact, you can also implemented similar layout by used APIs [cmdSetTable](#), [cmdJumpTab](#), [sendData](#). more details please refer [webdemo/js/demo.js](#).

Printing one line text by bitmap print, the usage are same as [printLine](#). you can use this method to print not supported code page or some string cannot printing correctly(such as [Arabic ligature](#)), but this method have a bit

slow printing speed.

`printLineByBitmap(text)`

`printLineComboByBitmap(textLeft, textRight)`

`printLineMultiByBitmap(arrStr, arrWidth, arrAlign)`