EPSON
EXCEED YOUR VISION

ePOS-Print API Ver.1.2 User's Manual

Overview

Describes the features and development environment.

Sample Program

Describes how to use the sample program and how to build a system.

Programming Guide

Describes how to write programs in Web application development.

ePOS-Print API

Describes the ePOS-Print API.

ePOS-Print Canvas API

Describes the ePOS-Print CanvasAPI.

Appendix

Describes the specifications for printers used for ePOS-Print, how to use the ePOS-Print API code creation tool, and the rendering of images in HTML5 Canvas.

M00042101
Rev.B

Cautions

- No part of this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Seiko Epson Corporation.
- The contents of this document are subject to change without notice. Please contact us for the latest information.
- While every precaution has taken in the preparation of this document, Seiko Epson Corporation assumes no responsibility for errors or omissions.
- Neither is any liability assumed for damages resulting from the use of the information contained herein.
- Neither Seiko Epson Corporation nor its affiliates shall be liable to the purchaser of this product or third parties for damages, losses, costs, or expenses incurred by the purchaser or third parties as a result of: accident, misuse, or abuse of this product or unauthorized modifications, repairs, or alterations to this product, or (excluding the U.S.) failure to strictly comply with Seiko Epson Corporation's operating and maintenance instructions.
- Seiko Epson Corporation shall not be liable against any damages or problems arising from the use of any options or any consumable products other than those designated as Original EPSON Products or EPSON Approved Products by Seiko Epson Corporation.

Trademarks

EPSON® and ESC/POS® are registered trademarks of Seiko Epson Corporation in the U.S. and other countries.

Windows® and Internet Explorer® are either registered trademarks or trademarks of Microsoft Corporation in the United States and other countries.

Safari™ and TrueType® are either registered trademarks or trademarks of Apple Inc. in the United States and other countries.

Android™ and Google Chrome™ are either registered trademarks or trademarks of Google Inc. in the United States and other countries.

Mozilla® and Firefox® are either registered trademarks or trademarks of Mozilla Foundation in the United States and other countries.

IOS® is registered trademarks or trademarks of Cisco in the United States and other countries.

ESC/POS® Command System

EPSON has been taking industry's initiatives with its own POS printer command system (ESC/POS). ESC/POS has a large number of commands including patented ones. Its high scalability enables users to build versatile POS systems. The system is compatible with all types of EPSON POS printers (excluding the TM-C100) and displays. Moreover, its flexibility makes it easy to upgrade the future. The functionality and the user-friendliness is valued around the world.

For Safety

Key to Symbols

The symbols in this manual are identified by their level of importance, as defined below. Read the following carefully before handling the product.

	Provides information that must be observed to avoid damage to your equipment or a malfunction.
	Provides important information and useful tips.

Restriction of Use

When this product is used for applications requiring high reliability/safety such as transportation devices related to aviation, rail, marine, automotive etc.; disaster prevention devices; various safety devices etc; or functional/precision devices etc, you should use this product only after giving consideration to including fail-safes and redundancies into your design to maintain safety and total system reliability. Because this product was not intended for use in applications requiring extremely high reliability/safety such as aerospace equipment, main communication equipment, nuclear power control equipment, or medical equipment related to direct medical care etc, please make your own judgment on this product's suitability after a full evaluation.

About this Manual

Aim of the Manual

This manual is intended to provide development engineers with all the information necessary for building/designing an ePOS-Print API system or developing/designing an ePOS-Print printer application, by using TM intelligent printers.

Manual Content

The manual is made up of the following sections:

Contents

Overview

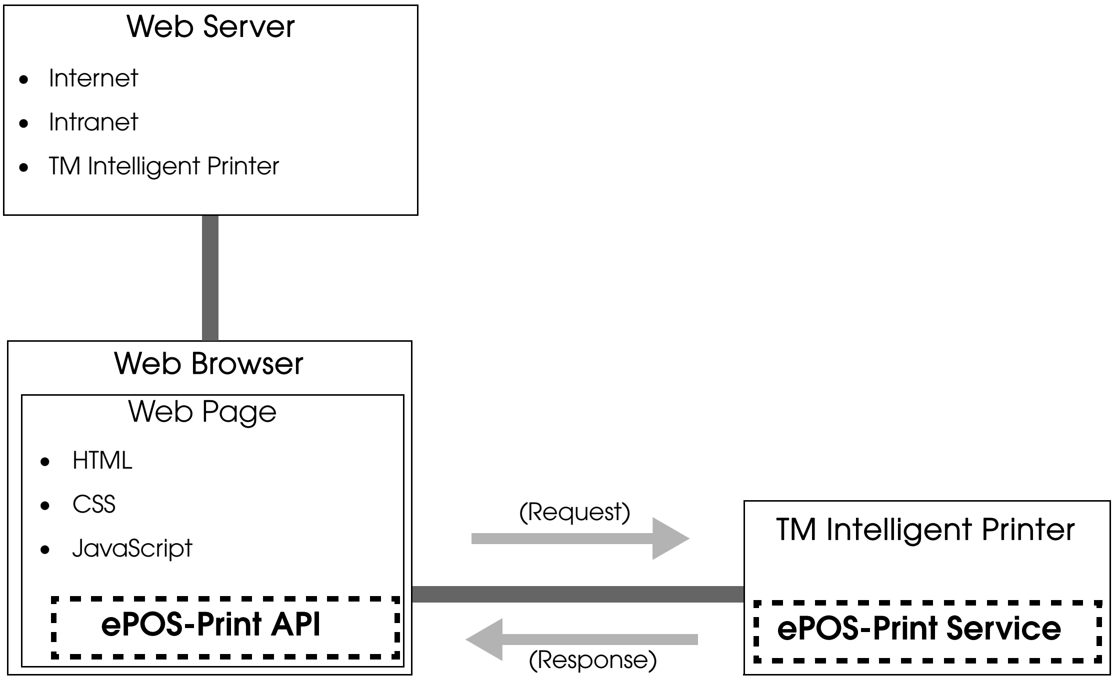
This chapter describes the features of and the specifications for ePOS-Print.

Overview of ePOS-Print

ePOS-Print is functionality to control POS printers in a multi-platform environment. Using ePOS-Print, data can be directly printed from Web browsers on personal computers, smart phones, or tablet computers to TM intelligent printers. In addition, print images rendered in HTML5 Canvas can be printed.

ePOS-Print provides the API for print commands.

When a print document (Request) is sent via HTTP from the host to the ePOS-Print Service of a TM intelligent printer, ePOS-Print processes the printing of that document and returns a response document (Response).



Features

- ❏ As long as it is in a network environment, a terminal with an HTML5-supported Web browser can perform printing from anywhere.
- ❏ Installation of drivers and plug-ins is not required.
- ❏ No PCs or servers are required for printing.
- ❏ Allows printing from public and private clouds.
- ❏ Allows printing in languages supported in Web browsers.
- ❏ Automatically checks the status of the TM printer before printing. There is no need for checking the status of the TM printer in advance. (Supported in Ver.1.2 and later)
- ❏ Does not respond to a printer's function to automatically send its status (AutoStatusBack). Instead, capable of sending an empty print command and checking the status of the TM printer based on the result of command transmission. (Supported in Ver.1.2 and later)
- ❏ To change the printer settings, utility programs dedicated to each printer or other utility programs should be used.
- ❏ Allows printing by TM printers via TM intelligent printers.
- ❏ Provides ePOS-Print API and ePOS-Print Canvas API.

<<ePOS-Print API>>

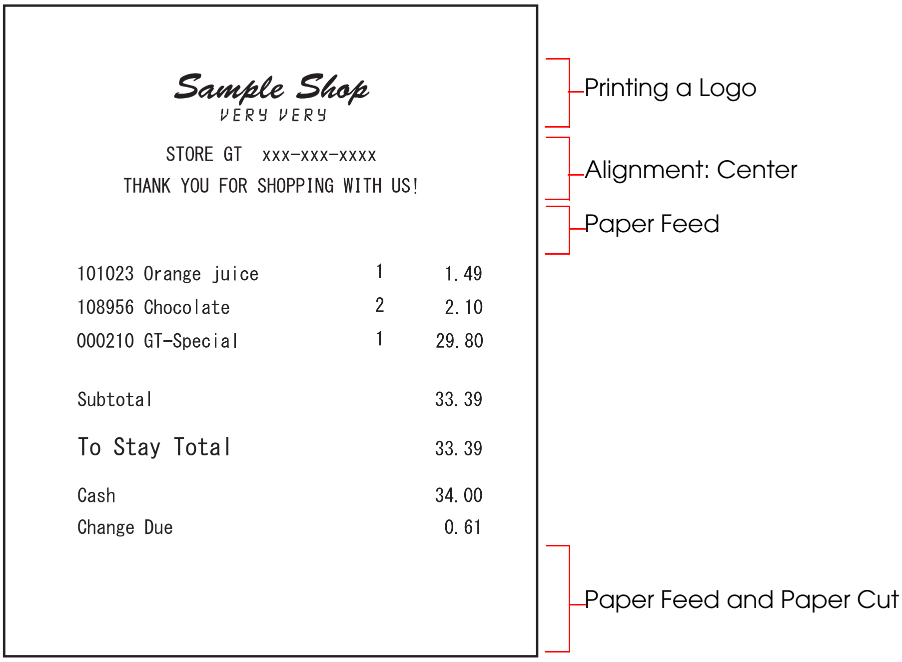
- Allows device fonts to be used for printing.
- Allows barcode printing.

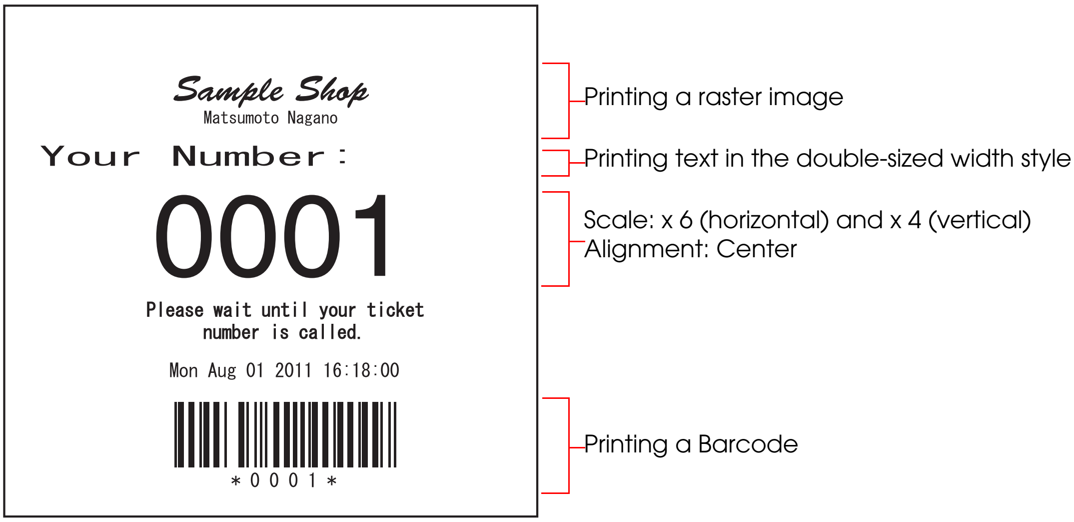
<<ePOS-Print Canvas API>>

- Allows printing of images rendered in HTML5 Canvas.
- Allows TrueType fonts to be used for printing.

Print Example

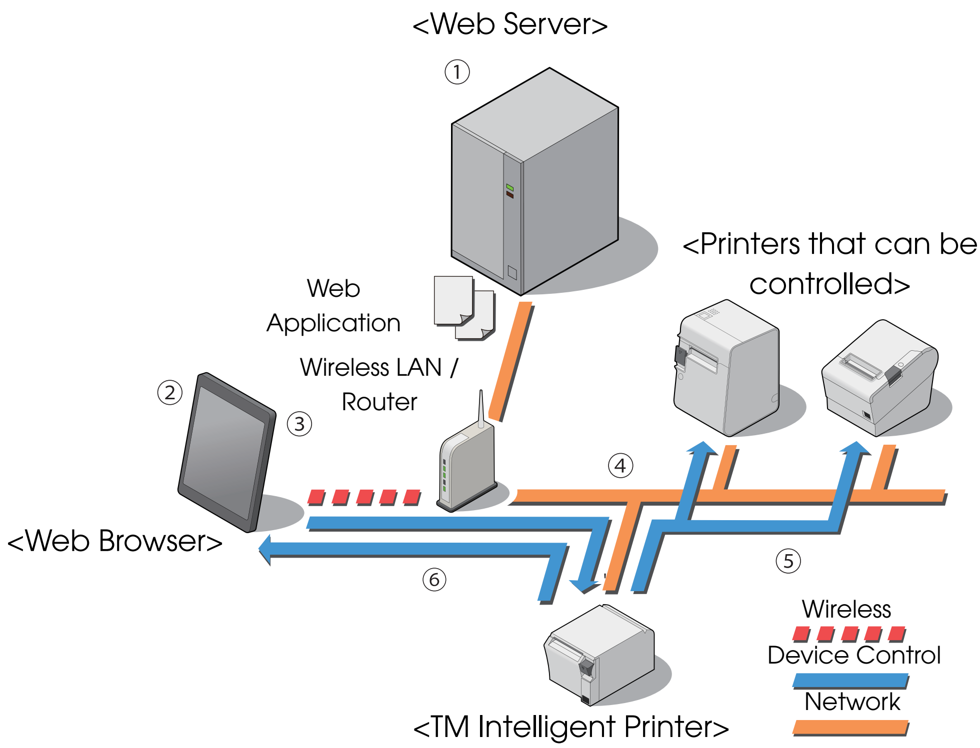
ePOS-Print API






Sample Shop
No. 012301
Enjoy!
FREE Coffee
Expires Mon Aug 01 2011
Sample Shop
16:27:24
Monochrome or Grayscale
Monochrome

Print Flow



1. A Web application is placed.
2. A Web browser displays the Web application.
3. The Web browser sends print data.
4. A TM intelligent printer sends the print data to printers that can be controlled.
5. The data is printed from printers that can be controlled.
6. The TM intelligent printer returns a response document to the terminal.

Features

Printing functions of ePOS-Print API

- Print setting (alignment/line feed space/text rotation/page mode)
- Character data setting (language/font (device font)/double-sizing/scale/smoothing/print position)
- Character style setting (inversion of black and white/underline/bold)
- Paper feed setting (in dots/in lines)
- Image printing (raster image/NV graphics)
- Barcode printing
  (For barcodes that can be printed by each model, refer to "Printer specifications" on page 153)
- Two dimensional symbol printing
  (For two dimensional symbols that can be printed by each model, refer to "Printer specifications" on page 153.)
- Ruled line setting
- Drawer kick function
- Buzzer function
- ESC/POS command transmission
- Response document acquisition (print result/printer status/system error status)

Printing functions of ePOS-Print Canvas API

- Printing of images (raster images) rendered in HTML5 Canvas
- Feed cut
- Response document acquisition (print result/printer status/system error status)

Operating Environment

Web Browser

HTML5-supported Web browser

- Windows Internet Explorer 9 or later
- Mozilla Firefox 3.6 or later
- Google Chrome 7 or later
- Safari in iOS4.0 or later
- Standard browser in Android 2.2 or later

Terminal

Terminal with an HTML5-supported Web browser

TM Intelligent Printer

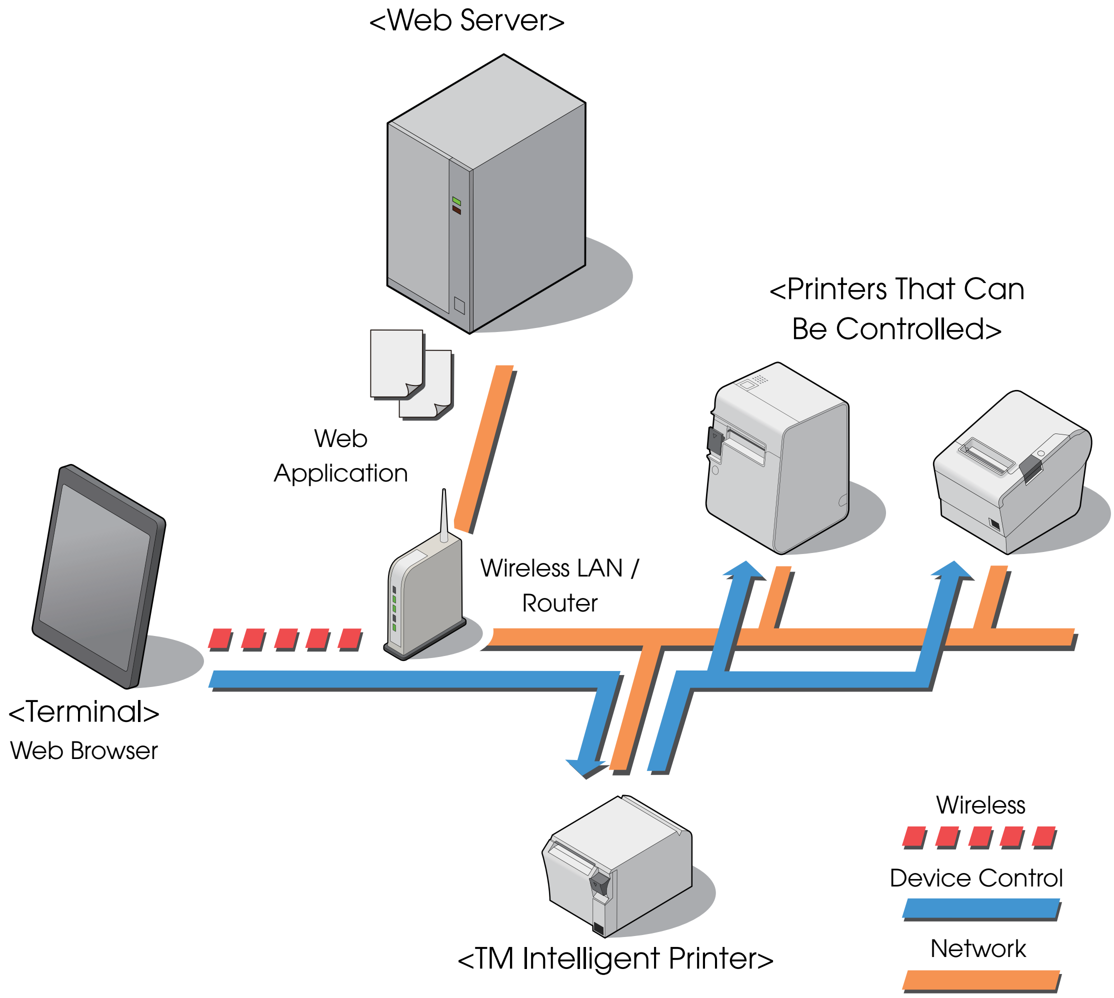
- TM-T88V-i
- TM-T70-i

TM Printers That Can Be Controlled

- TM-T88V
- TM-T70
- TM-T90

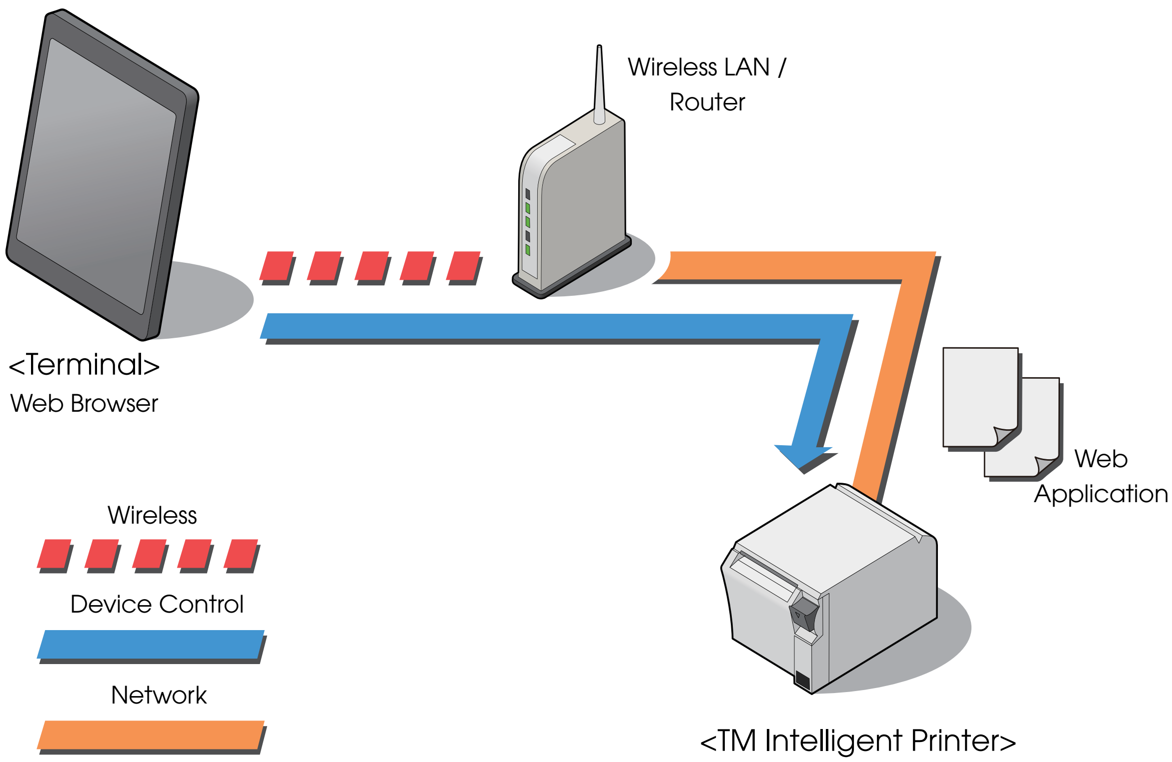
System Construction Example

Registering a Web Application Into the Web Server



- Web Server
  A Web application is placed.
- Terminal
  Executes the Web application using a browser (HTML5-supported Web browser).
- TM Intelligent Printer
  Receives/prints print data sent from the Web browser or controls other devices.
- Printers That Can Be Controlled
  Print the print data received from the TM intelligent printer.

Registering a Web Application Into a TM Intelligent Printer



- Terminal
  Executes the Web application using a browser.
- TM Intelligent Printer
  Receives and prints print data sent from the Web browser.

Contents in the Package

Manual

❏ ePOS-Print API User's Manual (This Document)

❏ ePOS-Print XML User's Manual

❏ TM-T88V-i Technical Reference Guide

❏ TM-T70-i Technical Reference Guide

SampleProgram

ePOS-Print_API_UM_E_Sample.zip

The following are included:

❏ epos-print-1.2.x.js (ePOS-Print JavaScript for embedding)

❏ index.html (Sampleprogram)

❏ epos-print-api.html (ePOS-Print API code creation tool)

❏ Rendering in HTML5 Canvas

- canvas-print-text.html(Rendering text)
- canvas-print-image.html(Rendering images)
- canvas-print-graph.html(Rendering graphics)
- canvas-print-hand.html(Rendering handwritten images)
- canvas-print-barcode.html(Rendering barcode)

Utility

❏ TM-T88V Utility

❏ TM Flash Logo Setup Utility

❏ TMNet WinConfig

Restrictsions

- ❏ The drawer and the buzzer cannot be used together.
- ❏ The buzzer function cannot be used if the printer is not provided with the buzzer.
- ❏ When multiple tones are set for raster images, intermitting printing may occur because the amount of data to print increases and white stripes may appear in the print result. (in Ver. 1.2 and later)
- ❏ The scan quality of barcodes/two-dimensional symbols printed as multiple-tone raster images cannot be guaranteed. Print them as two-tone images. (in Ver. 1.2 and later)

Sample Program

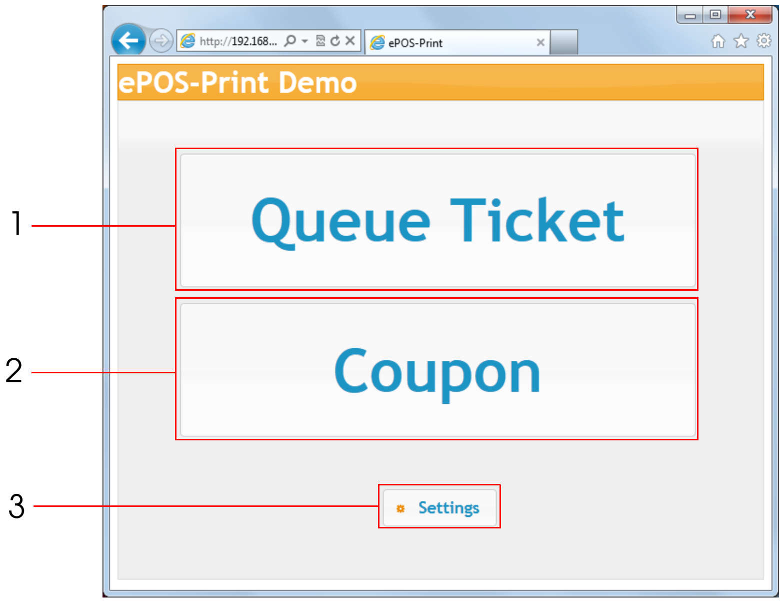
This chapter describes how to use the sample program.

- In this chapter, descriptions are made based on a system configuration using a Web server.
- Descriptions are made assuming that the Web server in this chapter is a Web server configured by using IIS (Microsoft Internet Information Services). If your Web server is used in a different environment, interpret the descriptions accordingly.

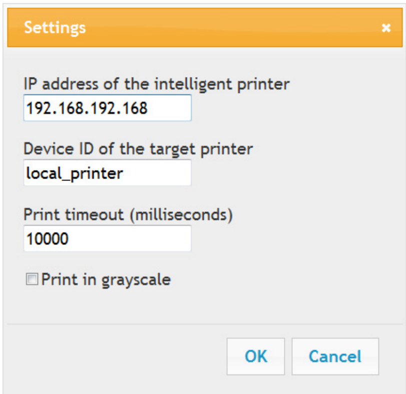
Sample Program System Overview

Sample Program Screen

The screen compositions for the sample program are as follows:



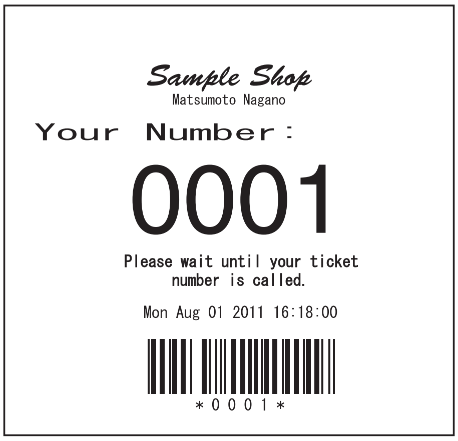
Item	Description
1 Queue Ticket	Prints queue ticket numbers. This is a sample program using the ePOS-Print API.
2 Coupon	Prints coupons. This is a sample program using the ePOS-Print Canvas API.

Item		Description
3 Settings		Displays the "Settings" screen. The screen is used to set the following: • IP address of the intelligent printer (Default : 192.168.192.168) • Device ID of the target printer (Default : local_printer) • Print timeout(milliseconds) (Default : 10000) • Prints coupons in gray scale (in Ver. 1.2 and later) (Default : No)

Print Image

The sample program prints the following:

Your Number
(ePOS-Print API)



Coupon
(ePOS-Print Canvas API)

Program Flow

From its initial display state up to print job completion, the sample program flows as below.

Queue ticket number issuance (ePOS-Print API)

1. Open the sample program Web page

↓

2. Settings

- ❏ IP address of the TM intelligent printer
- ❏ Device ID of the printer to be controlled
- ❏ Print timeout time

↓

3. Click "Issue queue ticket number"

↓

4. Creates print data (ePOS-Print Builder Object)

↓

5. Prints (ePOS-Print Object)

Action on the sample program

Action on the customer

2

Coupon issuance (ePOS-Print Canvas API)

1. Open the sample program Web page

2. Settings
   - ❏ IP address of the TM intelligent printer
   - ❏ Device ID of the printer to be used for printing
   - ❏ Print timeout time

3. Click "Issue coupon"

4. Render in HTML5 Canvas

5. Prints (ePOS-Print Canvas API Object)

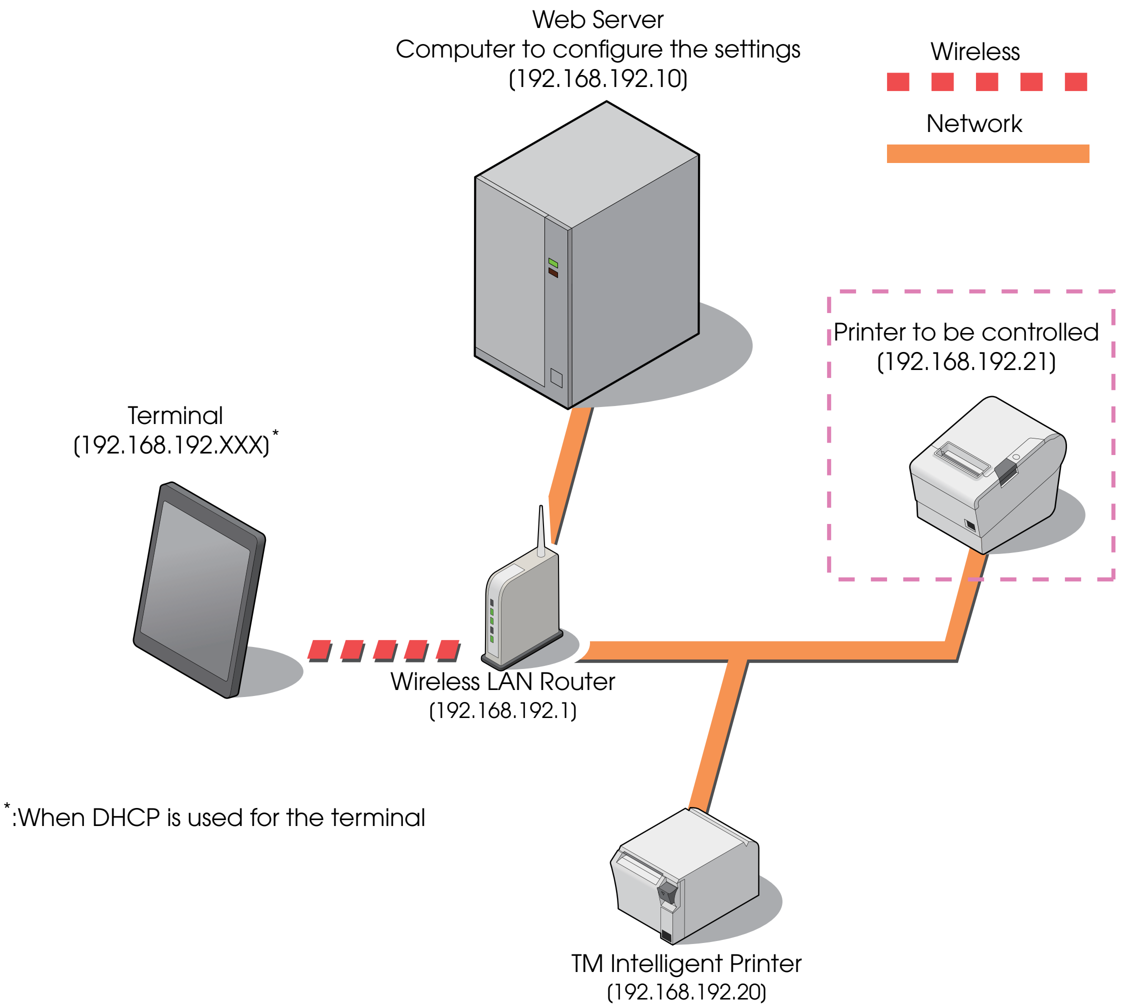
Action on the sample program

Action on the customer

Operating Environment

The system configuration diagram for the sample programs is as below.

- The figure below also describes an example of IP address settings as network settings.
- In the sample program, "Printer to be controlled" is not required. Refer to it if necessary.



- ❏ Web server/computer to configure the settings
  (Descriptions here are made assuming that the Web server is the same as the computer to configure the settings.)
- ❏ Wireless LAN Router
- ❏ TM Intelligent Printer (1 set)
  TM-T88V-i/ TM-T70-i
- ❏ Terminal
  Terminal with an HTML5-supported Web browser

Environment Settings

A flow for configuring the environment settings for the sample program is shown as follows:

1. Router Settings

Configure the settings such as SSID, IP address, DHCP, and allocated IP address. For details, refer to the manual for the device you are using.

2. Computer Settings

1. Network Settings

   Configure the network settings for the computer such as IP address.

2. Web Server Configuration

   Configure a Web server on the computer.

3.Registration of Sample Program (ePOS-Print_API_UM_E_Sample.zip) (p.27)

4. Terminal settings

Configure the wireless LAN (Wi-Fi) settings of the terminal to match the router settings so as to enable network connection. For details, refer to the manual for the device you are using.

5. Network Settings for the TM Intelligent Printer (p.28)

Configuration is done from a Web browser.

6. Network settings for the printer to be controlled

Configure these settings by using the network setting utility. For details, refer to the detailed instruction manual for the printer (these settings are not required in this sample program).

7. Device ID Settings (p.29)

Configuration is done from a Web browser(these settings are not required in this sample program).

8. Sample Program Settings (p.31)

Configuration is done from a Web browser(these settings are not required in this sample program).

Registration of Sample Program (ePOS-Print_API_UM_E_Sample.zip)

Register the sample program into the Web server.

> Download ePOS-Print_API_UM_E_Sample.zip.
> For details, refer to Contents in the package (p. 18).

Register the program according to the following procedure:

1 Start the Web server.

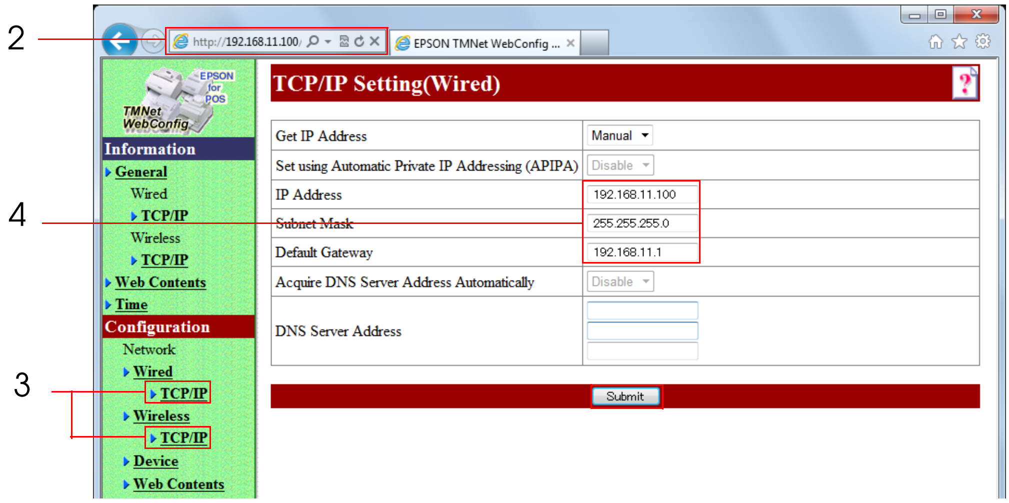
2 Explode the sample program (ePOS-Print_API_UM_E_Sample.zip) and then copy the exploded contents into the following folder:

Example: Web server configured by using IIS
System drive:\ Inetpub\wwwroot

> Copy the sample program as a user with administrator authority.

Network Settings for the TM Intelligent Printer

Use TMNet WebConfig to configure the network settings such as IP address for the printer.



Configure the settings according to the following procedure:

1 Connect the printer to the network and turn the power ON.

2 Start the Web browser and type the IP address of the TM intelligent printer interface into the address bar.

> The default value for the IP address of the TM intelligent printer is "192.168.192.168".

3 TMNet WebConfig starts.
Select as (Configuration) - (Wired / Wireless) - (TCP/IP).

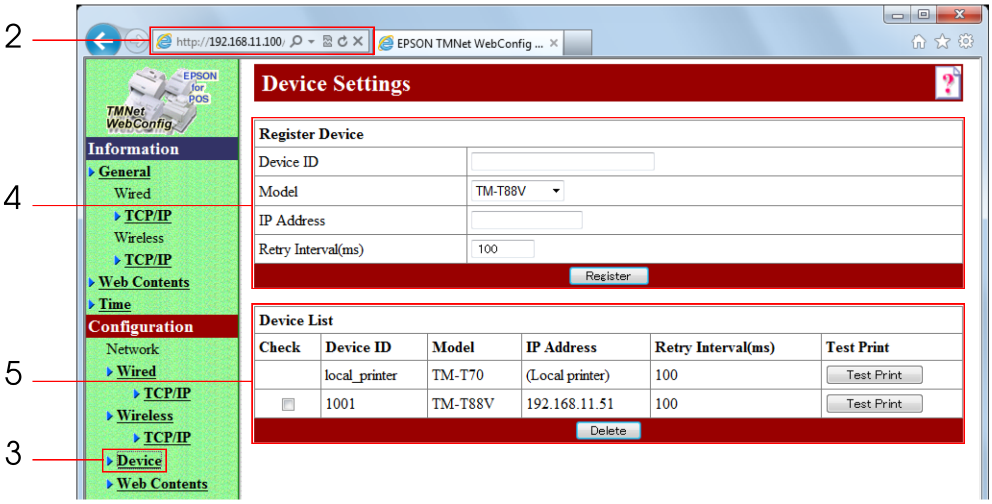
4 The “TCP/IP Setting” screen appears.
Configure the network settings for the TM intelligent printer and click (Submit).

5 Print the status sheet using the TM intelligent printer to check that the IP address has been updated.

Device ID Settings

Set the Device ID of the printer to be controlled by ePOS-Print into the TM intelligent printer. Use TMNet WebConfig to set the Device ID.

> In the sample program, "Device ID Settings" are not required. Refer to it if necessary.



Configure the settings according to the following procedure:

1. Connect all the printers to the network and turn their power ON.

2. Start the Web browser and enter the IP address set in Network Settings for the TM Intelligent Printer (p.28).

3. TMNet WebConfig starts.
   Select as (Configuration)-(Device).

4 The "Device Settings" screen appears. Set the following and click (Register).

Item	Description
Device ID	Specifies the ID to identify the printer to be controlled by ePOS-Print.
Model	Specifies the model of the printer to be controlled.
IP Address	Specifies the IP address of the printer to be controlled.
Retry Interval (ms)	Specifies the interval of retry toward the printer to be controlled, in milliseconds.

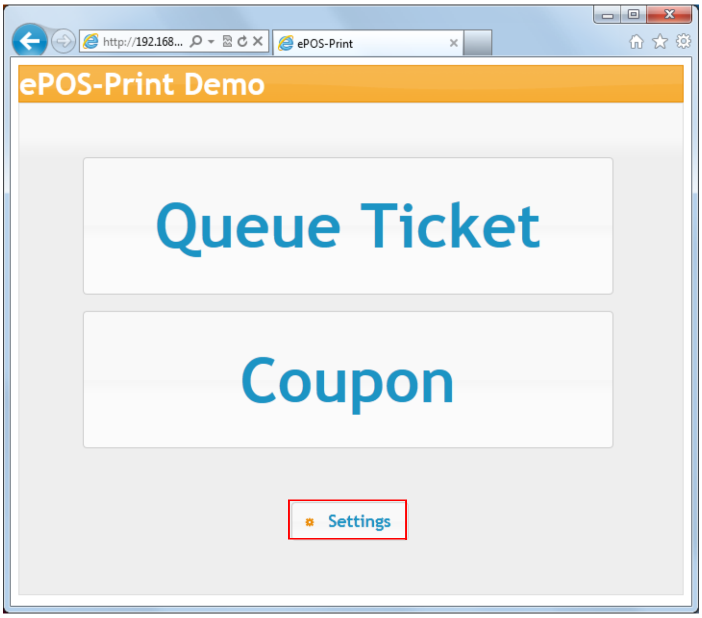
5 Information on the registered devices is displayed in (Device List).
Click (Test Print) for each registered printer to check that it operates correctly.

Sample Program Settings

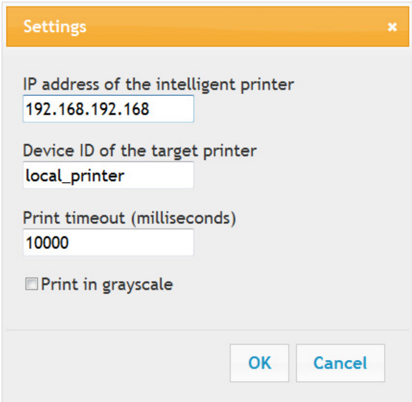
Configure the settings for the sample program according to the procedure below.

> In the sample program, "Device ID Settings" are not required. Refer to it if necessary.

1. Start the Web server.
2. Connect all the printers to the network and turn their power ON.
3. Open the following URL page using the Web browser.
   http://Web server IP address/index.html
4. The sample program page opens. Click (Settings).



5 The “Settings” screen appears. Specify the following and click (OK).



Item	Description
IP address of the intelligent printer	Specifies the IP address of the TM intelligent printer. (Default value: 192.168.192.168)
Device ID of the target printer	Specifies the Device ID of the printer to print queue ticket numbers and coupons. (Default value: local_printer)
Print timeout (millisecond)	Specifies the timeout time. (default : 10000)
Print in grayscale	Prints coupons in gray scale. (Default: No)

Programming Guide

This chapter describes how to write programs in the application development using ePOS-Print.

ePOS-Print API

Print Mode

There are two types of print modes: standard and page modes.

Standard mode

In standard mode, characters are printed line by line. The line feed space is adjusted based on the font size and the height of images, barcodes, etc. This mode is suitable for the type of printing such as printing receipts that requires the paper length to change according to the print space.

Page mode

In page mode, you set a print area, lay out data in it, and print the data in a batch operation. Characters, images, and barcodes are laid out in the print positions (coordinates).

Programming Flow

For the ePOS-Print API, programming is performed based on the following work flow:

1. Embedding of ePOS-Print API (p.35)

2. Print Document Creation (p.36)

- To create a text print document: (p.37)
- To create a graphic print document: (p.38)
- To create a page mode print document (p.39)

3. Transmission of Print Document (p.40)

4. Reception of Print Result (p.42)

- The TM intelligent printer checks the status of the TM printer to be used for printing and then starts printing operation. (in Ver. 1.2 and later)
- A status event helps check the status of the TM printer. For details on the procedure, refer to Reception of Status Event (p. 44). (in Ver. 1.2 and later)

Embedding of ePOS-Print API

The ePOS-Print API is provided so that ePOS-Print can be used from the JavaScript on the client side.
It is provided as JavaScript, and its file name is "epos-print-1.2.x.js".
The ePOS-Print API is used by embedding epos-print-1.2.x.js into applications.

Preparation

To use the ePOS-Print API, place epos-print-1.2.x.js on the Web server.

Embedding into Web pages

Embed the script into the Web page by using the HTML <script> tags.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>TITLE</title>
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
  function buildMessage() {
    var builder = new epson.ePOSBuilder();
                  .
                  .
  }
</script>
</head>

<body>
   .
   .
</body>
</html>
```

Embed

Print Document Creation

A print document is created using an ePOS-Print Builder object.
Create an ePOS-Print Builder object using the constructor for it; create a print document using the object's methods; and then acquire that print document using the toString method. For details, refer to List of API functions (p.53).
Refer to the following program for print document creation.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>TITLE</title>
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
  function buildMessage() {
    //Create an ePOS-Print Builder object
    var builder = new epson.ePOSBuilder();
    //Create a print document
    builder.addTextLang('en')
    builder.addTextSmooth(true);
    builder.addTextFont(builder.FONT_A);
    builder.addTextSize(3, 3);
    builder.addText('Hello,\tWorld!\n');
    builder.addCut(builder.CUT_FEED);
    //Acquire the print document
    var request = builder.toString();
  }
</script>
</head>
<body>
</body>
</html>
```

Create a print document

To create a text print document:

To create a text print document, store the font settings into the command buffer using text methods and then create a print document. Refer to the following program.

For the string "Hello World!", to create a print document based on the following settings:

- Font: FontA
- Scale: x 4 (horizontal) and x 4 (vertical)
- Style: Bold

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
  function buildMessage() {
    //Create an ePOS-Print Builder object
    var builder = new epson.ePOSBuilder();
    //Create a print document
    //<Configure the print character settings>
    builder.addTextLang('en');
    builder.addTextSmooth(true);
    builder.addTextFont(builder.FONT_A);
    builder.addTextSize(4, 4);
    builder.addTextStyle(false, false, true, undefined);
    //<Specify the print data>
    builder.addText('Hello,\tWorld!\n');
    builder.addCut(builder.CUT_FEED);
    //Acquire the print document
    var request = builder.toString();
  }
</script>
```

To create a graphic print document:

To create a graphic print document, store a raster image obtained by rendering an image in HTML5 Canvas into the command buffer using the addImage method. Refer to the following program.

To create a print document for the image file “logo.bmp”

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
  function buildMessage() {
    //Create an ePOS-Print Builder object
    var builder = new epson.ePOSBuilder();
    //Render an image in HTML5 Canvas
    var canvas = document.getElementByld('canvas');
    var context = canvas.getContext('2d');
    context.drawImage(document.getElementByld('logo'), 0, 0, 200, 70);
    //Create a print document
    builder.addTextAlign(builder.ALIGN_CENTER);
    builder.addImage(context, 0, 0, canvas.width, canvas.height, builder.COLOR_1);
    builder.addCut(builder.CUT_FEED);
    //Acquire the print document
    var request = builder.toString();
  }
</script>
```

This section describes how to print a raster image. In addition, there is also a method of printing graphics registered in the NV memory of the printer. For details, refer to addLogo method (p.75).

To create a page mode print document

When the addPageBegin method is stored in the command buffer, the page mode starts. Store the print area (addPageArea method) and the print start position (addPagePosition method) into the command buffer. Specify the print start position according to the print data. After that, store the methods into the command buffer to create print data. For the end of page mode, store the PageEnd method into the command buffer.

For the string "Hello World!", to create a print document based on the following settings:

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
  function buildMessage() {
       //Create an ePOS-Print Builder object
    var builder = new epson.ePOSBuilder();
      //Create a print document
      //<The page mode starts>
    builder.addPageBegin();
      //<Specify the page mode print area>
    builder.addPageArea(100, 50, 200, 100);
      //<Specify the page mode print position>
    builder.addPagePosition(0, 42);
      //<Specify the print data>
    builder.addTextLang('en');
    builder.addTextFont(builder.FONT_A);
    builder.addTextSize(4, 4);
    builder.addTextStyle(false, false, true, undefined);
    builder.addText('Hello,\tWorld!\n');
      //<The page mode ends>
    builder.addPageEnd();
    builder.addCut(builder.CUT_FEED);
      //Acquire the print document
    var request = builder.toString();
  }
</script>
```

Transmission of Print Document

A print document is sent using an ePOS-Print object.
Create an ePOS-Print object using the constructor and specify the end point address for the printer to be used for printing as well as the print document into the send method to send the document.
For the details about the printer end point address, refer to Printer End Point Address (p.41).
Refer to the following program.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>TITLE</title>
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
  function buildMessage() {
    //Create a print document
    var builder = new epson.ePOSBuilder();
    builder.addTextLang('en');
    builder.addTextSmooth(true);
    builder.addTextFont(builder.FONT_A);
    builder.addTextSize(3, 3);
    builder.addText('Hello,\tWorld!\n');
    builder.addCut(builder.CUT_FEED);
    var request = builder.toString();

    //Set the end point address
    var address = 'http://192.168.192.168/cgi-bin/epos/
                   service.cgi?devid=local_printer&timeout=10000';
    //Create an ePOS-Print object
    var epos = new epson.ePOSPrint(address);
    //Send the print document
    epos.send(request);
  }
</script>
</head>
<body>
</body>
</html>
```

Transmission of print document

Printer End Point Address

Specify the printer end point address in the following format:

http://(domain)/cgi-bin/epos/service.cgi?devid=(device ID)&timeout=(timeout time)

Items to specify	Description
Domain	Specify either the IP address or the domain name of the TM intelligent printer.
Device ID	Specifies the printer to be used for printing. Specify the Device ID registered using the TM intelligent printer's TMNet WebConfig.
Timeout period	Specifies the time to abort the process in milliseconds. The timeout parameter is optional; when it is omitted, 60 seconds (60000) is set. When the timeout period elapses, the print job is canceled; the data already interpreted by the printer before the start of the print abort process is printed.

Reception of Print Result

The print result can be received by setting a callback function using the onreceive property (p. 122) of the ePOS-Print object. The following information is obtained:

- Print result
- Error code
- Printer status

> The printer status can be obtained when communication with the printer is possible.

Refer to the following program. For the details about how to program a callback function in detail, refer to Error handling (p.43).

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>TITLE</title>
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
  function buildMessage() {
    //Create a print document
    var builder = new epson.ePOSBuilder();
    builder.addTextLang('en');
    builder.addTextSmooth(true);
    builder.addTextFont(builder.FONT_A);
    builder.addTextSize(3, 3);
    builder.addText('Hello,\tWorld!\n');
    builder.addCut(builder.CUT_FEED);
    var request = builder.toString();

    var address = 'http://192.168.192.168/cgi-bin/epos/
                   service.cgi?devid=local_printer&timeout=10000';
    //Create an ePOS-Print object
    var epos = new epson.ePOSPrint(address);
    //Set a response receipt callback function
    epos.onreceive = function (res) {
        //When the printing is not successful, display a message
        if (!res.success) {
            alert('A print error occurred');
        }
    }
    //Send the print document
    epos.send(request);
  }
</script>
</head>
<body>
</body>
</html>
```

Print result receipt callback function

Error handling

Refer to the following program for the error handling method by a callback function.

```
//Create an ePOS-Print object
var epos = new epson.ePOSPrint(address);
// Set a response receipt callback function
epos.onreceive = function (res) {
    // Obtain the print result and error code
    var msg = 'Print' + (res.success ? 'Success' : 'Failure') + '\nCode:' + res.code
                                                                  + '\nStatus:\n';
    // Obtain the printer status
    var asb = res.status;
    if (asb & epos.ASB_NO_RESPONSE) {
                msg += ' No printer response\n';
            }
            if (asb & epos.ASB_PRINT_SUCCESS) {
                msg += ' Print complete\n';
            }
            if (asb & epos.ASB_DRAWER_KICK) {
                msg += ' Status of the drawer kick number 3 connector pin = "H"\n';
            }
            if (asb & epos.ASB_OFF_LINE) {
                msg += ' Offline status\n';
            }
            if (asb & epos.ASB_COVER_OPEN) {
                msg += ' Cover is open\n';
            }
            if (asb & epos.ASB_PAPER_FEED) {
                msg += ' Paper feed switch is feeding paper\n';
            }
            if (asb & epos.ASB_WAIT_ON_LINE) {
                msg += ' Waiting for online recovery\n';
            }
            if (asb & epos.ASB_PANEL_SWITCH) {
                msg += ' Panel switch is ON\n';
            }
            if (asb & epos.ASB_MECHANICAL_ERR) {
                msg += ' Mechanical error generated\n';
            }
            if (asb & epos.ASB_AUTOCUTTER_ERR) {
                msg += ' Auto cutter error generated\n';
            }
            if (asb & epos.ASB_UNRECOVER_ERR) {
                msg += ' Unrecoverable error generated\n';
            }
            if (asb & epos.ASB_AUTORECOVER_ERR) {
                msg += ' Auto recovery error generated\n';
            }
            if (asb & epos.ASB_RECEIPT_NEAR_END) {
                msg += ' No paper in the roll paper near end detector\n';
            }
            if (asb & epos.ASB_RECEIPT_END) {
                msg += ' No paper in the roll paper end detector\n';
            }
            if (asb & epos.ASB_BUZZER) {
                msg += ' Sounding the buzzer (limited model)\n';
            }
            if (asb & epos.ASB_SPOOLER_IS_STOPPED) {
                msg += ' Stop the spooler\n';
    }
    //Display in the dialog box
    alert(msg);
}
```

3

Reception of Status Event

The status event notification function is used to check the printer status without printing. (in Ver. 1.2 and later) Refer to the following:

```
//Set the end point address
var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer
                                                                    &timeout=10000';
//Create an ePOS-Print Builder object
var builder = new epson.ePOSBuilder(address);

//Set an event callback function (cover open)
epos.oncoveropen = function () {
  alert('coveropen');
};

//Set an event callback function (paper near end)
epos.onpapernearend = function () {
  alert('papernearend');
};

//Enable status event operation
epos.open();
```

ePOS-Print Canvas API

For the ePOS-Print Canvas API, programming is performed based on the following work flow:

1. Embedding of ePOS-Print Canvas API (p.46)

2. Rendering in HTML5 Canvas (p.47)

3. Prints an Canvas image (p.48)

4. Reception of Print Result (p.49)

- The TM intelligent printer checks the status of the TM printer to be used for printing and then starts printing operation.
- A status event helps check the status of the TM printer. For details on the procedure, refer to Reception of Status Event (p. 44).

Embedding of ePOS-Print Canvas API

The ePOS-Print Canvas API is provided as JavaScript. And its file name is "epos-print-1.2.x.js".
It is used by embedding epos-print-1.2.x.js into applications.

Preparation

To use the ePOS-Print Canvas API, place epos-print-1.2.x.js on the Web server.

Embedding into Web pages

Embed the script into the Web page by using the HTML <script> tags.

Embed

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>TITLE</title>
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
  function drawCanvas() {
                 .
                 .
  }
</script>
</head>

<body>
   .
   .
</body>
</html>
```

Rendering in HTML5 Canvas

Render an image in HTML5 Canvas.

Rendering in HTML5 Canvas

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>TITLE</title>
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
  function drawCanvas() {
    // Rendering in HTML5 Canvas
    //<Obtain the context>
    var canvas = document.getElementByld('myCanvas');
    var context = canvas.getContext('2d');
    //<Render an image>
    context.clearRect(0, 0, 512, 480);
    context.drawImage(document.getElementByld('coffee'), 0, 0
                  , 512, 384);
    context.fillStyle = 'rgba(255, 255, 255, 0.5)';
    context.fillRect(0, 0, 512, 480);
    context.fillStyle = 'rgba(0, 0, 0, 1.0)';
    //<Render a water mark for the image>
    context.drawImage(document.getElementByld('wmark'), 0, 0);
    context.drawImage(document.getElementByld('wmark'), 256, 324);
    //<Render text>
    context.textAlign = 'center';
    context.textBaseline = 'alphabetic';
    context.font = 'bold normal normal 48px "Times New Roman", serif';
    context.fillText('FREE Coffee', 256, 224);

  }
</script>
</head>
<body>
  <canvas id="myCanvas" width="512" height="480"></canvas>
  <img id="coffee" src="img/coffee.jpg" alt="">
  <img id="wmark" src="img/wmark.png" alt="">
</body>
</html>
```

Prints an Canvas image

Content drawn in HTML5 Canvas is printed using the ePOS-Print Canvas API.
Create an ePOS-Print Canvas API object using the constructor; for the Print method, specify the end point address for the printer to be used for printing as well as the canvas content and whether to select paper cut; and then print a document. For the details about the printer end point address, refer to Printer End Point Address (p.41).
Refer to the following program.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>TITLE</title>
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
  function drawCanvas() {
    // Rendering in HTML5 Canvas
    //<Obtain the context>
    var canvas = document.getElementByld('myCanvas');
    var context = canvas.getContext('2d');
           .
           .
           .
    //Set the end point address
    var address = 'http://192.168.192.168/cgi-bin/epos/
                   service.cgi?devid=local_printer&timeout=10000';
    //Create an ePOS-Print Canvas API object
    var epos = new epson.CanvasPrint(address);
      //Print
    epos.print(canvas, true);
  }
</script>
</head>
<body>
  <canvas id="myCanvas" width="512" height="480"></canvas>
  <img id="coffee" src="img/coffee.jpg" alt="">
  <img id="wmark" src="img/wmark.png" alt="">
</body>
</html>
```

Transmission of print document

For the details about the printer end point address, refer to Printer End Point Address (p.41).

Reception of Print Result

The print result can be received by setting a callback function using the onreceive property (p. 122) of the ePOS-Print Canvas API object. The following information is obtained:

- Print result
- Error code
- Printer Status

> The printer status can be obtained when communication with the printer is possible.

Refer to the following program. For the details about how to program a callback function in detail, refer to Error handling (p.43).

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>TITLE</title>
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
  function drawCanvas() {
    // Rendering in HTML5 Canvas
    //<Obtain the context>
    var canvas = document.getElementByld('myCanvas');
    var context = canvas.getContext('2d');
              .
              .
              .

    //Set the end point address
    var address = 'http://192.168.192.168/cgi-bin/epos/
                   service.cgi?devid=local_printer&timeout=10000';
    //Create an ePOS-Print Canvas API object
    var epos = new epson.CanvasPrint(address);
    //Set a response receipt callback function
    epos.onreceive = function (res) {
        //When the printing is not successful, display a message
        if (!res.success) {
            alert('A print error occurred');
        }
    }
      //Print
    epos.print(canvas, true);
  }
</script>
</head>
<body>
  <canvas id="myCanvas" width="512" height="480"></canvas>
  <img id="coffee" src="img/coffee.jpg" alt="">
  <img id="wmark" src="img/wmark.png" alt="">
</body>
</html>
```

Print result receipt callback function

Error handling

Refer to the following program for the error handling method by a callback function.

```
var epos = new epson.CanvasPrint(address);
// Set a response receipt callback function
epos.onreceive = function (res) {
    // Obtain the print result and error code
    var msg = ''Print ' + (res.success ? 'Success' : 'Failure') + '\nCode:'
                                                          + res.code + '\nStatus:\n';
    // Obtain the printer status
    var asb = res.status;
    if (asb & epos.ASB_NO_RESPONSE) {
                msg += ' No printer response\n';
            }
            if (asb & epos.ASB_PRINT_SUCCESS) {
                msg += ' Print complete\n';
            }
            if (asb & epos.ASB_DRAWER_KICK) {
                msg += ' Status of the drawer kick number 3 connector pin = "H"\n';
            }
            if (asb & epos.ASB_OFF_LINE) {
                msg += ' Offline status\n';
            }
            if (asb & epos.ASB_COVER_OPEN) {
                msg += ' Cover is open\n';
            }
            if (asb & epos.ASB_PAPER_FEED) {
                msg += ' Paper feed switch is feeding paper\n';
            }
            if (asb & epos.ASB_WAIT_ON_LINE) {
                msg += ' Waiting for online recovery\n';
            }
            if (asb & epos.ASB_PANEL_SWITCH) {
                msg += ' Panel switch is ON\n';
            }
            if (asb & epos.ASB_MECHANICAL_ERR) {
                msg += ' Mechanical error generated\n';
            }
            if (asb & epos.ASB_AUTOCUTTER_ERR) {
                msg += ' Auto cutter error generated\n';
            }
            if (asb & epos.ASB_UNRECOVER_ERR) {
                msg += ' Unrecoverable error generated\n';
            }
            if (asb & epos.ASB_AUTORECOVER_ERR) {
                msg += ' Auto recovery error generated\n';
            }
            if (asb & epos.ASB_RECEIPT_NEAR_END) {
                msg += ' No paper in the roll paper near end detector\n';
            }
            if (asb & epos.ASB_RECEIPT_END) {
                msg += ' No paper in the roll paper end detector\n';
            }
            if (asb & epos.ASB_BUZZER) {
                msg += ' Sounding the buzzer (limited model)\n';
            }
            if (asb & epos.ASB_SPOOLER_IS_STOPPED) {
                msg += ' Stop the spooler\n';
    }
    //Display in the dialog box
    alert(msg);
}
```

Reception of Status Event

The status event notification function is used to check the printer status without printing. (in Ver. 1.2 and later) Refer to the following.

```
//Set the end point address
var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer
                                                                    &timeout=10000';

//Create an ePOS-Print Canvas API object
var epos = new epson.CanvasPrint(address);

//Set an event callback function (cover open)
epos.oncoveropen = function () {
  alert('coveropen');
};

//Set an event callback function (paper near end)
epos.onpapernearend = function () {
  alert('papernearend');
};

//Enable status event operation
epos.open();
```

ePOS-Print API

This chapter describes the ePOS-Print API.

List of API functions

ePOS-Print provides the following objects:

- ❏ ePOS-Print Builder (window.epson.ePOSBuilder) Object (p. 53)
- ❏ ePOS-Print (window.epson.ePOSPrint) Object (p. 55)

window.epson.ePOSBuilder Components

Element	API		Description	Page
Constructor				
	ePOS Builder		Initializes an ePOS-Print XML Builder object	57
Method				
	Text	addTextAlign	Adds a tag for the text alignment setting.	58
		addText-LineSpace	Adds a tag for the line feed space setting.	59
		addTextRotate	Adds a tag for the text rotation setting.	60
		addText	Adds a tag for printing text.	61
		addTextLang	Adds a tag for the target language setting.	62
		addTextFont	Adds a tag for the text font setting.	63
		addTextSmooth	Adds a tag for the text smoothing setting.	64
		addTextDouble	Adds a tag for specifying the double-sized text setting.	65
		addTextSize	Adds a tag for the text scale setting.	67
		addTextStyle	Adds a tag for the text style setting.	68
		addTextPosition	Adds a tag for specifying the print position of text.	70
	Paper Feed	addFeedUnit	Adds a tag for paper feeding (in dots).	71
		addFeedLine	Adds a tag for paper feeding (in lines).	72
	Graphic	addImage	Adds a tag for a raster image to be printed.	73
		addLogo	Adds a tag for an NV logo to be printed.	75
	Barcode	addBarcode	Adds a tag for a bar code to be printed.	76
		addSymbol	Adds a tag for a two-dimensional code to be printed.	80

Element	API		Description	Page
Method				
	Ruled line	addHLine	Adds a tag for a horizontal line to be printed.	85
		addVLineBegin	Adds a tag for starting a vertical line.	87
		addVLineEnd	Adds a tag for finishing a vertical line.	89
	Pagemode	addPageBegin	Adds a tag for switching to page mode.	91
		addPageEnd	Adds a tag for finishing page mode.	92
		addPageArea	Adds a tag for specifying the print area in page mode.	93
		addPageDirection	Adds a tag for specifying the print direction in page mode.	95
		addPagePosition	Adds a tag for specifying the print position in page mode.	97
		addPageLine	Adds a tag for drawing a line in page mode.	99
		addPageRectangle	Adds a tag for drawing a rectangle in page mode.	101
	Cut	addCut	Adds a tag for paper cut.	103
	Drawer kick-out	addPulse	Adds a tag for the drawer kick-out.	105
	Buzzer	addSound	Adds a tag for turning on the buzzer.	107
	Send Command	addCommand	Raster image halftone processing method	109
	Create a Print Document	toString	Raster image brightness correction value	110
Property				
	halftone		Raster image halftone processing method (in Ver. 1.2 and later)	111
	brightness		Raster image brightness correction value (in Ver. 1.2 and later)	112
	message		Message buffer	113
Constant				
	FONT_*		font	
	ALIGN_*		alignment	
	COLOR_*		color specification	
	HALFTONE_*		Halftone type (in Ver. 1.2 and later)	
	MODE_*		Color mode (in Ver. 1.2 and later)	
	BARCODE_*		bar code type	
	HRI_*		HRI position	
	SYMBOL_*		two-dimensional code type	
	LEVEL_*		error correction level	

Element	API	Description	Page
	LINE_*	line style	
	DIRECTION_*	page mode print direction	
	CUT_*	paper cut type	
	DRAWER_*	drawer kick-out connector	
	PULSE_*	drawer kick-out pulse length	
	PATTERN_*	buzzer sound pattern	

Numerical values to be set to parameters

In the ePOS-Print Builder object API, numerical values are set to some parameters. Set values with the following in mind:

- Unit
  Specify numbers in dots for units that represent length.
  (Print position, paper feed space, width and height of images and barcodes, etc.)
- Range
  Depending on the printer specifications, a specifiable range is predetermined. For details, refer to Printer specifications (p.153).
- Resolution
  The resolution varies depending on the printer. It affects the actual print size. The higher the resolution is, the smaller the print size becomes, and vice versa. For each printer's resolution, refer to Printer specifications (p.153).

window.epson.ePOSPrint Components

Element	API	Description	Page
Constructor			
	ePOS-Print	Initializes an ePOS-Print object	114
Method			
	send	Sends a message	115
	open	Enables status event operation (in Ver. 1.2 and later)	116
	close	Disables status event operation (in Ver. 1.2 and later)	117
Property			
	address	URL of the printer (in Ver. 1.2 and later)	118
	enabled	Enabling/disabling of status event (in Ver. 1.2 and later)	119
	interval	Printer status update interval (in Ver. 1.2 and later)	120
	status	Status	121

Element	API	Description	Page
Event			
	onreceive	Response message receipt event	122
	onerror	Communication error event	124
	onstatuschange	Status change event (in Ver. 1.2 and later)	125
	ononline	Online event (in Ver. 1.2 and later)	125
	onoffline	Offline event (in Ver. 1.2 and later)	126
	onpoweroff	Non-response event (in Ver. 1.2 and later)	126
	oncoverok	Cover close event (in Ver. 1.2 and later)	127
	oncoveropen	Cover open event (in Ver. 1.2 and later)	127
	onpaperok	Paper remaining event (in Ver. 1.2 and later)	128
	onpapernearend	Paper near end event (in Ver. 1.2 and later)	128
	onpaperend	Paper end event (in Ver. 1.2 and later)	129
	ondrawerclosed	Drawer close event (in Ver. 1.2 and later)	129
	ondraweropen	Drawer open event (in Ver. 1.2 and later)	130
Constant			
	ASB_*	Status	

ePOS-Print Builder Object

This objects creates a print document for printer control commands that specify strings or graphics to be printed, paper cut, etc.

Constructor

Constructor for an ePOS-Print Builder object.
Creates a new ePOS-Print Builder object and initializes it.

Syntax

ePOSBuilder`();`

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
  var builder = new epson.ePOSBuilder();
}
//-->
</script>
```

addTextAlign method

Adds the text alignment setting to the command buffer.

	This API setting also applies to barcodes/two dimensional symbols.
	When the page mode is selected for the print mode, to set text rotation, use the addPageDirection method (p. 95) instead of this API function.

Syntax

addTextAlign`(align);`

Parameter

- align : (Required parameter, Object type : String)
  Specifies the text alignment.

Constant(align)	Description
ALIGN_LEFT (default)	Alignment to the left
ALIGN_CENTER	Alignment to the center
ALIGN_RIGHT	Alignment to the right

Return value

Return value	Object type
ePOS-Print Builder Object	ePOS Builder

Exception

Exception	Object type
Parameter " ... " is invalid	Error

Example

To set alignment to the center:

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
  var builder = new epson.ePOSBuilder();
  builder.addTextAlign(builder.ALIGN_CENTER);
}
//-->
</script>
```

addTextLineSpace method

Adds the line feed space setting to the command buffer.

Syntax

```
addTextLineSpace(linespc);
```

Parameter

- linespc : (Required parameter, Object type : Number)
  Specifies the line feed space (in dots). Specifies an integer from 0 to 255.

Return value

Return value	Object type
ePOS-Print Builder Object	ePOS Builder

Exception

Exception	Object type
Parameter " ... " is invalid	Error

Example

To set the line feed space to 30 dots:

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
  var builder = new epson.ePOSBuilder();
  builder.addTextLineSpace(30);
}
//-->
</script>
```

4

addTextRotate method

Adds the text rotation setting to the command buffer.

This API setting also applies to barcodes/two dimensional symbols.

When the page mode is selected for the print mode, to set text rotation, use the addPageDirection method (p.95) instead of this API function.

Syntax

addTextRotate(rotate);

Parameter

- rotate : (Required parameter, Object type : Boolean)
  Specifies whether to rotate text.

Setting	Description
true or 11	Specifies rotated printing of text.
false or 0 (default)	Cancels rotated printing of text.

Return value

Return value	Object type
ePOS-Print Builder Object	ePOS Builder

Exception

Exception	Object type
Parameter " ... " is invalid	Error

Example

To set text rotation:

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
  var builder = new epson.ePOSBuilder();
  builder.addTextRotate(true);
}
//-->
</script>
```

addText method

Adds the printing of text to the command buffer.

Note	After printing text, to print content other than text, execute line feed or paper feed.
Caution	In page mode, characters are laid out in the current print position with the reference point being the character baseline dot (Printer specifications (p.153)).

Syntax

```
addText(data);
```

Parameter

- data : (Required parameter, Object type : String)
  Specify a character string to be printed.
  For the horizontal tab/line feed, use the following escape sequences:

String	Description
\t	Horizontal tab(HT)
\n	Line feed (LF)
\\	Carriage return

Return valueReturn value

Return value	Object type
ePOS-Print Builder Object	ePOS Builder

Exception

Exception	Object type
Parameter " ... " is invalid	Error

Example

To add character strings:

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
  var builder = new epson.ePOSBuilder();
  builder.addText('Hello,\t').addText('World\n');
}
//-->
</script>
```

addTextLang method

Adds the language setting to the command buffer.

Syntax

addTextLang`(lang);`

Parameter

- lang : (Required parameter, Object type : String)
  Specifies the target language.

Setting	Language
en(default)	English(ANK)
ja	Japanese

Return value

Return value	Object type
ePOS-Print Builder Object	ePOS Builder

Exception

Exception	Object type
Parameter " ... " is invalid	Error

Example

To set the language as English:

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
  var builder = new epson.ePOSBuilder();
  builder.addTextLang('en');
}
//-->
</script>
```

addTextFont method

Adds the text font setting to the command buffer.

Syntax

addTextFont(font);

Parameter

- font : (Required parameter, Object type : String)
  Specifies the font.

Constant (font)	Language
FONT_A (default)	Font A
FONT_B	Font B
FONT_C	Font C

Return value

Return value	Object type
ePOS-Print Builder Object	ePOS Builder

Exception

Exception	Object type
Parameter " ... " is invalid	Error

Example

To set the font B:

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
  var builder = new epson.ePOSBuilder();
  builder.addTextFont(builder.FONT_B);
}
//-->
</script>
```

4

addTextSmooth method

Adds the smoothing setting to the command buffer.

Syntax

addTextSmooth(smooth);

Parameter

- smooth : (Required parameter, Object type : Boolean)
  Specifies whether to enable smoothing.

Setting	Description
true or 1	Specifies smoothing.
false or 0 (default)	Cancels smoothing

Return value

Return value	Object type
ePOS-Print Builder Object	ePOS Builder

Exception

Exception	Object type
Parameter " ... " is invalid	Error

Example

To enable smoothing:

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
  var builder = new epson.ePOSBuilder();
  builder.addTextSmooth(true);
}
//-->
</script>
```

addTextDouble method

Adds the double-sized text setting to the command buffer.

Syntax

```
addTextDouble(dw, dh);
```

Parameter

- dw : (Optional parameter, Object type : Boolean)
  Specifies the double-sized width.

Setting	Description
true or 1	Specifies the double-sized width.
false or 0 (default)	Cancels the double-sized width
undefined (When not specified)	Retains the current setting for double-sized width.

- dh : (Optional parameter, Object type : Boolean)
  Specifies the double-sized height.

Setting	Description
true or 1	Specifies the double-sized height
false or 0 (default)	Cancels the double-sized height
undefined (When not specified)	Retains the current setting for double-sized height

When true or 1 is set for both the dw and dh parameters, double width and height characters are printed.

4

Return value

Return value	Object type
ePOS-Print Builder Object	ePOS Builder

Exception

Exception	Object type
Parameter " ... " is invalid	Error

Example

To set the size as double width and height:

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
  var builder = new epson.ePOSBuilder();
  builder.addTextDouble(true, true);
}
//-->
</script>
```

addTextSize method

Adds the text scale setting to the command buffer.

Syntax

```
addTextSize(width, height);
```

Parameter

- width : (Optional parameter, Object type : Number)
  Specifies the horizontal scale of text.

Setting	Description
Integer from 1 to 8	Horizontal scale (default : 1)
undefined (When not specified)	Retains the current setting for the horizontal scale.

- height : (Optional parameter, Object type : Number)
  Specifies the vertical scale of text.

Setting	Description
Integer from 1 to 8	Vertical scale (default : 1)
undefined (When not specified)	Retains the current setting for the vertical scale.

Return value

Return value	Object type
ePOS-Print Builder Object	ePOS Builder

Exception

Exception	Object type
Parameter " ... " is invalid	Error

Example

To set a horizontal scale of x 4 and a vertical scale of x 4:

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
  var builder = new epson.ePOSBuilder();
  builder.addTextSize(4, 4);
}
//-->
</script>
```

4

addTextStyle method

Adds the text style setting to the command buffer.

Syntax

```
addTextStyle(reverse, ul, em, color);
```

Parameter

- reverse : (Optional parameter, Object type : Boolean)
  Specifies inversion of black and white for text.

Setting	Description
true or 1	Specifies the inversion of black and white parts of characters.
false or 0 (default)	Cancels the inversion of black and white parts of characters.
undefined (When not specified)	Retains the current setting for inversion of black and white.

- ul : (Optional parameter, Object type : Boolean)
  Specifies the underline style.

Setting	Description
true or 1	Specifies underlining.
false or 0 (default)	Cancels underlining.
undefined (When not specified)	Retains the current underlining setting.

- em : (Optional parameter, Object type : Boolean)
  Specifies the bold style.

Setting	Description
true or 1	Specifies emphasized printing of characters.
false or 0 (default)	Cancels emphasized printing of characters.
undefined (When not specified)	Retains the current setting for emphasized printing.

- color : (Optional parameter, Object type : String)
  Specifies the color.

Setting	Description
COLOR_NONE	Characters are not printed.
COLOR_1 (default)	First color
COLOR_2	Second color
COLOR_3	Third color
COLOR_4	Fourth color
undefined (When not specified)	Retains the current color setting

Return value

Return value	Object type
ePOS-Print Builder Object	ePOS Builder

Exception

Exception	Object type
Parameter " ... " is invalid	Error

Example

To set the underline style:

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
  var builder = new epson.ePOSBuilder();
  builder.addTextStyle(undefined, true);
}
//-->
</script>
```

addTextPosition method

Adds the horizontal print start position of text to the command buffer.

Syntax

addTextPosition`(x);`

Parameter

- x : (Required parameter, Object type : Number)
  Specifies the horizontal print start position (in dots).
  Specifies an integer from 0 to 65535.

Return value

Return value	Object type
ePOS-Print Builder Object	ePOS Builder

Exception

Exception	Object type
Parameter " ... " is invalid	Error

Example

To set the print position at 120 dots from the left end:

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
  var builder = new epson.ePOSBuilder();
  builder.addTextPosition(120);
}
//-->
</script>
```

addFeedUnit method

Adds paper feeding in dots to the command buffer.

Syntax

addFeedUnit`(unit);`

Parameter

- unit : (Required parameter, Object type : Number)
  Specifies the paper feed space (in dots). Specifies an integer from 0 to 255.

Return value

Return value	Object type
ePOS-Print Builder Object	ePOS Builder

Exception

Exception	Object type
Parameter " ... " is invalid	Error

Example

To feed paper by 30 dots:

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
  var builder = new epson.ePOSBuilder();
  builder.addFeedUnit(30);
}
//-->
</script>
```

4

addFeedLine method

Adds paper feeding in lines to the command buffer.

Syntax

addFeedLine`(line);`

Parameter

- unit : (Required parameter, Object type : Number)
  Specifies the paper feed space (in lines). Specifies an integer from 0 to 255.

Return value

Return value	Object type
ePOS-Print Builder Object	ePOS Builder

Exception

Exception	Object type
Parameter " ... " is invalid	Error

Example

To feed paper by 3 lines:

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
  var builder = new epson.ePOSBuilder();
  builder.addFeedLine(3);
}
//-->
</script>
```

addImage method

Adds raster image printing to the command buffer.
Prints graphics rendered in HTML5 Canvas.
Converts the specified range in a RGBA full-color image of HTML5 Canvas into raster image data according to the settings of the halftone and brightness properties. One pixel in an image equals to one printer dot.
When an image contains any transparent color, the background color of the image is assumed to be white.

- To print a raster image at high speed, specify ALIGN_LEFT for the addTextAlign method (p. 58), and specify a multiple of 8 not exceeding the printer's paper width for the width parameter of this API.
- In page mode, a raster image is laid out in the current print position with the reference point being its bottom left dot. The print position will not move.

If an HTML5 Canvas image contains images downloaded from different domains, you cannot print the image. In this case, a security error occurs due to violation of the same origin policy of JavaScript.

Syntax

```
addImage(context, x, y, width, height, color, mode);
```

Parameter

- context : (Required parameter, Object type : Context)
  Specifies the 2D context of HTML5 Canvas.
- x : (Required parameter, Object type : Number)
  Specifies the horizontal start position in the print area. Specifies an integer from 0 to 65535.
- y : (Required parameter, Object type : Number)
  Specifies the vertical start position in the print area. Specifies an integer from 0 to 65535.
- width : (Required parameter, Object type : Number)
  Specifies the width of the print area. Specifies an integer from 0 to 65535.
- height : (Required parameter, Object type : Number)
  Specifies the height of the print area. Specifies an integer from 0 to 65535.
- color : (Optional parameter, Object type : String)
  Specifies the color.

Setting	Description
COLOR_NONE	Characters are not printed.
COLOR_1 (default)	First color
COLOR_2	Second color
COLOR_3	Third color
COLOR_4	Fourth color
undefined (When not specified)	First color

4

- mode : (Optional parameter, Object type : String)
  Specifies the color mode. (in Ver. 1.2 and later)

Setting	Description
MODE_MONO	Monochrome (two-tone)
MODE_GRAY16	Gray scale (16-tone)
undefined (When not specified)	Monochrome (two-tone)

Return value

Return value	Object type
ePOS-Print Builder Object	ePOS Builder

Exception

Exception	Object type
Parameter " ... " is invalid	Error

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
  var builder = new epson.ePOSBuilder();
  var canvas = document.getElementById('canvas');
  if (canvas.getContext) {
    var context = canvas.getContext('2d');
    builder.addImage(context, 0, 0, canvas.width, canvas.height);
  }
}
//-->
</script>
```

To print an image 300 dots wide and 300 dots high in page mode:

```
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');
var builder = new epson.ePOSBuilder();
builder.addPageBegin();
builder.addPageArea(0, 0, 300, 300);
builder.addPagePosition(0, 299);
builder.addImage(context, 0, 0, 300, 300);
builder.addPageEnd();
```

addLogo method

Adds NV logo printing to the command buffer.
Prints a logo registered in the NV memory of the printer.

- Using TM-T88V Utility or logo registration utility (TMFLogo), register a logo in the printer in advance.
- In page mode, a logo is laid out in the current print position with the reference point being its bottom left dot.

Syntax

```
addLogo(key1, key2);
```

Parameter

- key1 : (Required parameter, Object type : Number)
  Specifies the key code 1 of an NV logo. Specifies an integer from 0 to 255.
- key2 : (Required parameter, Object type : Number)
  Specifies the key code 2 of an NV logo. Specifies an integer from 0 to 255.

Return value

Return value	Object type
ePOS-Print Builder Object	ePOS Builder

Exception

Exception	Object type
Parameter " ... " is invalid	Error

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
  var builder = new epson.ePOSBuilder();
  builder.addLogo(48, 48);
}
//-->
</script>
```

addBarcode method

Adds barcode printing to the command buffer.

> In page mode, a barcode is laid out in the current print position with the reference point being its bottom left dot (except for HRI).

Syntax

```
addBarcode(data, type, hri, font, width, height);
```

Parameter

- data : (Required parameter, Object type : String)
  Specifies the barcode data as a string.

Barcode type	Description
UPC-A	When an 11-digit number is specified, a check digit is automatically added. When a 12-digit number is specified, the 12th digit is processed as a check digit but the check digit is not validated.
UPC-E	Specify 0 as the first digit. Specify the manufacturer code in the digits 2 to 6. Specify (right-align) the item code in the digits 7 to 11. The number of item code digits varies depending on the manufacturer code. Specify 0s in empty digits. When an 11-digit number is specified, a check digit is automatically added. When a 12-digit number is specified, the 12th digit is processed as a check digit but the check digit is not validated.
EAN13 JAN13	When an 12-digit number is specified, a check digit is automatically added. When a 13-digit number is specified, the 12th digit is processed as a check digit but the check digit is not validated.
EAN8 JAN8	When a 7-digit number is specified, a check digit is automatically added. When an 8-digit number is specified, the 8th digit is processed as a check digit but the check digit is not validated.
CODE39	When the first character is *, the character is processed as the start character. In other cases, a start character is automatically added.
ITF	Start and stop codes are automatically added. Check digits are not added or validated.

Barcode type	Description
CODABAR	Specify a start character (A to D, a to d). Specify a stop character (A to D, a to d). Check digits are not added or validated.
CODE93	Start and stop characters are automatically added. A check digit is automatically calculated and added.
CODE128	Specify a start character (CODE A, CODE B, CODE C). A stop character is automatically added. A check digit is automatically calculated and added. To encode each of the following characters, specify two characters starting with the character "{": FNC1: {1 FNC2: {2 FNC3: {3 FNC4: {4 CODE A: {A CODE B: {B CODE C: {C SHIFT: {S {: {{
GS1-128	A start character, FNC1, a check digit, and a stop character are automatically added. To automatically calculate and add a check digit for an application identifier (AI) and the subsequent data, specify the character "*" in the position of the check digit. You can enclose an application identifier (AI) in parentheses. The parentheses are used as HRI print characters and are not encoded as data. You can insert spaces between an application identifier (AI) and data. The spaces are used as HRI print characters and are not encoded as data. To encode each of the following characters, specify two characters starting with the character "{": FNC1: {1 FNC3: {3 (: {( ): {) *: {* {: {{
GS1 DataBar Omnidirectional	Specify a 13-digit global trade item number (GTIN) not including an application identifier (AI) or a check digit.
GS1 DataBar Truncated	
GS1 DataBar Limited	

Barcode type	Description
BARCODE_GS1_ DATABAR_EXPANDED	You can enclose an application identifier (AI) in parentheses. The parentheses are used as HRI print characters and are not encoded as data. To encode each of the following characters, specify two characters starting with the character "{": FNC1: {1 (: {( ): {)

To specify binary data that cannot be represented by character strings, use the following escape sequences.

String	Description
\xnn	Control code
\\	Back slash

- type : (Required parameter, Object type : String)
  Specifies the barcode type.

Constant (type)	Barcode type
BARCODE_UPC_A	UPC-A
BARCODE_UPC_E	UPC-E
BARCODE_EAN13	EAN13
BARCODE_JAN13	JAN13
BARCODE_EAN8	EAN8
BARCODE_JAN8	JAN8
BARCODE_CODE39	CODE39
BARCODE_ITF	ITF
BARCODE_CODABAR	CODABAR
BARCODE_CODE93	CODE93
BARCODE_CODE128	CODE128
BARCODE_GS1_128	GS1-128
BARCODE_GS1_DATABAR_OMNIDIRECTIONAL	GS1 DataBar Omnidirectional
BARCODE_GS1_DATABAR_TRUNCATED	GS1 DataBar Truncated
BARCODE_GS1_DATABAR_LIMITED	GS1 DataBar Limited
BARCODE_GS1_DATABAR_EXPANDED	GS1 Databar Expanded

- hri : (Optional parameter, Object type : String)
  Specifies the HRI position.

Constant (hri)	Description
HRI_NONE (default)	HRI not printed
HRI_ABOVE	Above the bar code
HRI_BELOW	Below the bar code
HRI_BOTH	Both above and below the bar code

- font : (Optional parameter, Object type : String)
  Specifies the HRI font.

Constant (font)	Language
FONT_A(default)	Font A
FONT_B	Font B
FONT_C	Font C

- width : (Optional parameter, Object type : Number)
  Specifies the width of each module in dots. Specifies an integer from 2 to 6.
- height : (Optional parameter, Object type : Number)
  Specifies the barcode height in dots. Specifies an integer from 1 to 255.

Return value

Return value	Object type
ePOS-Print Builder Object	ePOS Builder

Exception

Exception	Object type
Parameter " ... " is invalid	Error

Example

To print barcodes:

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
  var builder = new epson.ePOSBuilder();
  builder.addBarcode('01234567890', builder.BAROCDE_UPC_A,
        builder.HRI_BELOW, undefined, 2, 64);
  builder.addBarcode('01234500005', builder.BAROCDE_UPC_E);
  builder.addBarcode('201234567890', builder.BAROCDE_EAN13);
  builder.addBarcode('201234567890', builder.BAROCDE_JAN13);
  builder.addBarcode('2012345', builder.BAROCDE_EAN8);
  builder.addBarcode('2012345', builder.BAROCDE_JAN8);
  builder.addBarcode('ABCDE', builder.BAROCDE_CODE39);
  builder.addBarcode('012345', builder.BAROCDE_ITF);
  builder.addBarcode('A012345A', builder.BAROCDE_CODABAR);
  builder.addBarcode('ABCDE', builder.BAROCDE_CODE93);
  builder.addBarcode('{Babcde', builder.BAROCDE_CODE128);
  builder.addBarcode('(01)201234567890*', builder.BAROCDE_GS1_128);
  builder.addBarcode('0201234567890',
        builder.BAROCDE_GS1_DATABAR_OMNIDIRECTIONAL);
  builder.addBarcode('0201234567890',
        builder.BAROCDE_GS1_DATABAR_TRUNCATED);
  builder.addBarcode('0201234567890',
        builder.BAROCDE_GS1_DATABAR_LIMITED);
  builder.addBarcode('(01)2012345678903',
        builder.BAROCDE_GS1_DATABAR_EXPANDED);
}
//-->
</script>
```

addSymbol method

Adds two-dimensional symbol printing to the command buffer.

> In page mode, a two-dimensional symbol is laid out in the current print position with the reference point being its bottom left dot.

Syntax

```
addSymbol(data, type, level, width, height, size);
```

Parameter

- data : (Required parameter, Object type : String)
  Specifies two-dimensional symbol data as a character string.

<table>
<tr><th>2D-Code type</th><th>Description</th></tr>
<tr><td>Standard PDF417</td><td rowspan="2">Convert the character string to the string in UTF-8, apply the escape sequence, and then encode the string.
The data area can contain up to 928 code words in a maximum of 90 rows, each of which can contain up to 30 code words.</td></tr>
<tr><td>Truncated PDF417</td></tr>
<tr><td>QR Code Model 1</td><td rowspan="2">Convert the character string to the string in Shift-JIS, apply the escape sequence, and then encode the string based on the data type as shown below.
Number: 0 to 9
Alphanumeric character:
0 to 9, A to Z, space, $, %, *, +, -, ., /, :
Kanji character: Shift-JIS value
8-bit, byte data:
0x00 to 0xff</td></tr>
<tr><td>QR Code Model 2</td></tr>
</table>

2D-Code type	Description
MaxiCode Mode 2	Convert the character string to the string in UTF-8, apply the escape sequence, and then encode the string. In Modes 2 and 3, when the first piece of data is ()>\ x1e01\x1dyy (where yy is a two-digit number), this is processed as the message header, and the subsequent data is processed as the primary message. In other cases, from the first piece of data, data is processed as the primary message. In Mode 2, specify the primary message in the following format: Postal code (1- to 9-digit number) GS:(\x1d) ISO country code (1- to 3-digit number) GS:(\x1d) Service class code (1- to 3-digit number) In Mode 3, specify the primary message in the following format: Postal code (1 to 6 pieces of data convertible by Code Set A) GS:(\x1d) ISO country code (1- to 3-digit number) GS:(\x1d) Service class code (1- to 3-digit number)
MaxiCode Mode 3	
MaxiCode Mode 4	
MaxiCode Mode 5	
MaxiCode Mode 6	
GS1 DataBar Stacked	Convert the character string to the string in UTF-8, apply the escape sequence, and then encode the string. Specify a 13-digit global trade item number (GTIN) not including an application identifier (AI) or a check digit.
GS1 DataBar Stacked Omnidirectional	
GS1 DataBar Expanded Stacked	Convert the character string to the string in UTF-8, apply the escape sequence, and then encode the string. You can enclose an application identifier (AI) in parentheses. The parentheses are used as HRI print characters and are not encoded as data. To encode each of the following characters, specify two characters starting with the character "{": FNC1: {1 (: {( ): {)

To specify binary data that cannot be represented by character strings, use the following escape sequences.

String	Description
\xnn	Control code
\\	Back slash

- type : (Required parameter, Object type : String)
  Specifies the two-dimensional symbol type.

Constant (type)	2D-Code type
SYMBOL_PDF417_STANDARD	Standard PDF417
SYMBOL_PDF417_TRUNCATED	Truncated PDF417
SYMBOL_QRCODE_MODEL_1	QR Code Model 1
SYMBOL_QRCODE_MODEL_2	QR Code Model 2
SYMBOL_MAXICODE_MODE_2	MaxiCode Mode 2
SYMBOL_MAXICODE_MODE_3	MaxiCode Mode 3
SYMBOL_MAXICODE_MODE_4	MaxiCode Mode 4
SYMBOL_MAXICODE_MODE_5	MaxiCode Mode 5
SYMBOL_MAXICODE_MODE_6	MaxiCode Mode 6
SYMBOL_GS1_DATABAR_STACKED	GS1 DataBar Stacked
SYMBOL_GS1_DATABAR_STACKED_ OMNIDIRECTIONAL	GS1 DataBar Stacked Omnidirectional
SYMBOL_GS1_DATABAR_EXPANDED_STACKED	GS1 DataBar Expanded Stacked

- level : (Optional parameter, Object type : String)
  Specifies the error correction level.

Constant (level)	Description
LEVEL_0	PDF417 error correction level 0
LEVEL_1	PDF417 error correction level 1
LEVEL_2	PDF417 error correction level 2
LEVEL_3	PDF417 error correction level 3
LEVEL_4	PDF417 error correction level 4
LEVEL_5	PDF417 error correction level 5
LEVEL_6	PDF417 error correction level 6
LEVEL_7	PDF417 error correction level 7
LEVEL_8	PDF417 error correction level 8
LEVEL_L	QR Code error correction level L
LEVEL_M	QR Code error correction level M
LEVEL_Q	QR Code error correction level Q
LEVEL_H	QR Code error correction level H
LEVEL_DEFAULT	Default level

- Select the level according to the two-dimensional symbol type.
- For MaxiCode and two-dimensional GS1 DataBar, select LEVEL_DEFAULT.

- width : (Optional parameter, Object type : Number)
  Specifies the module width. Specifies an integer from 0 to 255.

MaxiCode is ignored.

- height : (Optional parameter, Object type : Number)
  Specifies the module height. Specifies an integer from 0 to 255.

QR Code and MaxiCode are ignored.

- size : (Optional parameter, Object type : Number)
  Specifies the two-dimensional symbol maximum size. Specifies an integer from 0 to 65535.

QR Code and MaxiCode are ignored.

Return value

Return value	Object type
ePOS-Print Builder Object	ePOS Builder

Exception

Exception	Object type
Parameter " ... " is invalid	Error

Example

To print two-dimensional symbols:

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
  var builder = new epson.ePOSBuilder();
  builder.addSymbol('ABCDE', builder.SYMBOL_PDF417_STANDARD);
  builder.addSymbol('ABCDE', builder.SYMBOL_QRCODE_MODEL_2,
        builder.LEVEL_Q);
  builder.addSymbol('908063840\x1d850\x1d001\x1d\x04',
        builder.SYMBOL_MAXICODE_MODE_2);
  builder.addSymbol('0201234567890', builder.SYMBOL_
        GS1_DATABAR_STACKED);
  builder.addSymbol('0201234567890',
        builder.SYMBOL_GS1_DATABAR_STACKED_OMNIDIRECTIONAL);
  builder.addSymbol('(01)02012345678903',
        builder.SYMBOL_GS1_DATABAR_EXPANDED_STACKED);
}
//-->
</script>
```

addHLine method

Adds horizontal line printing to the command buffer.
Draws horizontal lines.

> ⚠ Not available in page mode.

Syntax

```
addHLine(x1, x2, style);
```

Parameter

- x 1: (Required parameter, Object type : Number)
  Specifies the start position of the horizontal line (in dots). Specifies an integer from 0 to 65535.
- x2 : (Required parameter, Object type : Number)
  Specifies the end position of the horizontal line (in dots). Specifies an integer from 0 to 65535.
- style : (Optional parameter, Object type : String)
  Specifies the line type.

Constant (style)	Description
LINE_THIN	Solid line: Thin
LINE_MEDIUM	Solid line: Medium
LINE_THICK	Solid line: Thick
LINE_THIN_DOUBLE	Double line: Thin
LINE_MEDIUM_DOUBLE	Double line: Medium
LINE_THICK_DOUBLE	Double line: Thick
undefined (When not specified)	Solid line: Thin

Return value

Return value	Object type
ePOS-Print Builder Object	ePOS Builder

Exception

Exception	Object type
Parameter " ... " is invalid	Error

Example

To draw double horizontal lines in the following positions:

- Between 100 dots and 200 dots from the left end
- Between 400 dots and 500 dots from the left end

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
  var builder = new epson.ePOSBuilder();
  builder.addHLine(100, 200, builder.LINE_THIN_DOUBLE);
  builder.addHLine(400, 500, builder.LINE_THIN_DOUBLE);
}
//-->
</script>
```

addVLineBegin method

Adds the beginning of vertical line to the command buffer. Starts to draw vertical lines.

	Not available in page mode.
	Vertical lines are drawn until their end is specified by addVLineEnd (p. 89). Use this API function with addVLineEnd.

Syntax

```
addVLineBegin(x, style);
```

Parameter

- x : (Required parameter, Object type : Number)
  Specifies the start position of the vertical line (in dots). Specifies an integer from 0 to 65535.
- style : (Optional parameter, Object type : String)
  Specifies the line type.

Constant (style)	Description
LINE_THIN	Solid line: Thin
LINE_MEDIUM	Solid line: Medium
LINE_THICK	Solid line: Thick
LINE_THIN_DOUBLE	Double line: Thin
LINE_MEDIUM_DOUBLE	Double line: Medium
LINE_THICK_DOUBLE	Double line: Thick
undefined (When not specified)	Solid line: Thin

Return value

Return value	Object type
ePOS-Print Builder Object	ePOS Builder

Exception

Exception	Object type
Parameter " ... " is invalid	Error

Example

To draw thin vertical lines at 100 dots and 200 dots from the left end:

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
  var builder = new epson.ePOSBuilder();
  builder.addVLineBegin(100).addVLineBegin(200);
  builder.addFeedUnit(100);
  builder.addVLineEnd(100).addVLineEnd(200);
}
//-->
</script>
```

addVLineEnd method

Adds the end of vertical line to the command buffer. Finishes drawing vertical lines.

⚠	Not available in page mode.
✎	Use this API function with addVLineBegin (p. 87).

Syntax

```
addVLineEnd(x, style);
```

Parameter

- x : (Required parameter, Object type : Number)
  Specifies the end position of the vertical line (in dots). Specifies an integer from 0 to 65535.
- style : (Optional parameter, Object type : String)
  Specifies the type of the line you want to finish drawing.

Constant (style)	Description
LINE_THIN	Solid line: Thin
LINE_MEDIUM	Solid line: Medium
LINE_THICK	Solid line: Thick
LINE_THIN_DOUBLE	Double line: Thin
LINE_MEDIUM_DOUBLE	Double line: Medium
LINE_THICK_DOUBLE	Double line: Thick
undefined (When not specified)	Solid line: Thin

Return value

Return value	Object type
ePOS-Print Builder Object	ePOS Builder

Exception

Exception	Object type
Parameter " ... " is invalid	Error

Example

To draw thin vertical lines at 100 dots and 200 dots from the left end:

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
  var builder = new epson.ePOSBuilder();
  builder.addVLineBegin(100).addVLineBegin(200);
  builder.addFeedUnit(100);
  builder.addVLineEnd(100).addVLineEnd(200);
}
//-->
</script>
```

addPageBegin method

Adds the switching to page mode to the command buffer. The page mode process starts.

> Vertical lines are processed in page mode until their end is specified by PageEnd (p. 92). Use this API function with PageEnd.

Syntax

addPageBegin`();`

Return value

Return value	Object type
ePOS-Print Builder Object	ePOS Builder

Example

To print the characters "ABCDE" in page mode:

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
  var builder = new epson.ePOSBuilder();
  builder.addPageBegin();
  builder.addText('ABCDE');
  builder.addPageEnd();
}
//-->
</script>
```

addPageEnd method

Adds the end of page mode to the command buffer. The page mode process ends.

> Use this API function with addPageBegin (p. 91).

Syntax

addPageEnd();

Return value

Return value	Object type
ePOS-Print Builder Object	ePOS Builder

Example

To print the characters "ABCDE" in page mode:

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
  var builder = new epson.ePOSBuilder();
  builder.addPageBegin();
  builder.addText('ABCDE');
  builder.addPageEnd();
}
//-->
</script>
```

addPageArea method

Adds the print area in page mode to the command buffer.
Specifies the print area in page mode (coordinates). After this API function, specify a print data API function such as the addText method.

⚠	Specify a print area to cover the content to be printed. If the print data extends beyond the print area, the print result will be such that the print data has been printed incompletely.
✎	Use this API function by inserting it between addPageBegin (p. 91) and PageEnd (p. 92).

Syntax

```
addPageArea(x, y, width, height);
```

Parameter

- x : (Required parameter, Object type : Number)
  Specifies the origin of the horizontal axis (in dots). Specifies an integer from 0 to 65535. 0 is the left end of the printer's printable area.
- y : (Required parameter, Object type : Number)
  Specifies the origin of the vertical axis (in dots). Specifies an integer from 0 to 65535. 0 is the position in which no paper feed has been performed.
- width : (Required parameter, Object type : Number)
  Specifies the width of the print area (in dots). Specifies an integer from 0 to 65535.
- height : (Required parameter, Object type : Number)
  Specifies the height of the print area (in dots). Specifies an integer from 0 to 65535.

⚠	Determine the width and height of the print area according to the print direction setting. Otherwise, the print data might not be printed completely.

Return value

Return value	Object type
ePOS-Print Builder Object	ePOS Builder

Exception

Exception	Object type
Parameter " ... " is invalid	Error

Example

To specify the print area with the origin (100, 50), a width of 200 dots, and a height of 30 dots and print the characters "ABCDE":

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
  var builder = new epson.ePOSBuilder();
  builder.addPageBegin();
  builder.addPageArea(100, 50, 200, 30);
  builder.addText('ABCDE');
  builder.addPageEnd();
}
//-->
</script>
```

addPageDirection method

Adds the page mode print direction setting to the command buffer. Specifies the print direction in page mode. This function can be omitted if rotation is not required.

Use this API function by inserting it between addPageBegin (p. 91) and PageEnd (p. 92).

Syntax

```
addPageDirection(dir);
```

Parameter

- dir : (Required parameter, Object type : String)
  Specifies the print direction in page mode.

Constant (dir)	Description
DIRECTION_LEFT_TO_RIGHT(default)	Left to right (No rotation.Data is printed from the top left corner to the right.)
DIRECTION_BOTTOM_TO_TOP	Bottom to top (Counterclockwise rotation by 90 degrees. Data is printed from the bottom left corner to the top.)
DIRECTION_RIGHT_TO_LEFT	Right to left (Rotation by 180 degrees.Data is printed from the bottom right corner to the left.)
DIRECTION_TOP_TO_BOTTOM	Top to bottom (Clockwise rotation by 90 degrees. Data is printed from the top right corner to the bottom.)

Return value

Return value	Object type
ePOS-Print Builder Object	ePOS Builder

Exception

Exception	Object type
Parameter " ... " is invalid	Error

Example

To print the characters "ABCDE" by rotating them 90 degrees clockwise:

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
  var builder = new epson.ePOSBuilder();
  builder.addPageBegin();
  builder.addPageArea(100, 50, 30, 200);
  builder.addPageDirection(builder.DIRECTION_TOP_TO_BOTTOM);
  builder.addText('ABCDE');
  builder.addPageEnd();
}
//-->
</script>
```

addPagePosition method

Adds the page mode print-position-set area to the command buffer.
Specifies the print start position (coordinates) in the area specified by the addPageArea method.

> Use this API function by inserting it between addPageBegin (p. 91) and PageEnd (p. 92).

Syntax

```
addPagePosition(x, y);
```

Parameter

- x : (Required parameter, Object type : Number)
  Specifies the horizontal print position (in dots). Specifies an integer from 0 to 65535.
- y : (Required parameter, Object type : Number)
  Specifies the vertical print position (in dots). Specifies an integer from 0 to 65535.

> Specify the print start position (coordinates) according to the content to be printed. Refer to the following.
> * To print a character string:
>   Specify the left end of the baseline for the first character. This can be omitted for left-aligned printing of standard-sized characters. To print double-sized height characters, specify a value equal to or greater than 42 for y.
> * To print a barcode:
>   Specify the bottom left of the symbol. And specify the barcode height for y.
> * To print a graphic/logo:
>   Specify the bottom left of the graphic data. And specify the graphic data height for y.
> * To print a two-dimensional symbol:
>   Specify the top left of the symbol. This can be omitted when printing from the top left.

Return value

Return value	Object type
ePOS-Print Builder Object	ePOS Builder

Exception

Exception	Object type
Parameter " ... " is invalid	Error

Example

To specify (50,30) for the print start position in the area specified by the addPageArea method and print the characters "ABCDE":

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
  var builder = new epson.ePOSBuilder();
  builder.addPageBegin();
  builder.addPageArea(100, 50, 200, 100);
  builder.addPagePosition(50, 30);
  builder.addText('ABCDE');
  builder.addPageEnd();
}
//-->
</script>
```

addPageLine method

Adds line drawing in page mode to the command buffer. Draws a line in page mode.

⚠	Diagonal lines cannot be drawn.
✎	Use this API function by inserting it between addPageBegin (p. 91) and PageEnd (p. 92).

Syntax

```
addPageLine(x1, y1, x2, y2, style);
```

Parameter

- x 1: (Required parameter, Object type : Number)
  Specifies the horizontal start position of the line (in dots). Specifies an integer from 0 to 65535.
- y 1: (Required parameter, Object type : Number)
  Specifies the vertical start position of the line (in dots). Specifies an integer from 0 to 65535.
- x2 : (Required parameter, Object type : Number)
  Specifies the horizontal end position of the line (in dots). Specifies an integer from 0 to 65535.
- y2 : (Required parameter, Object type : Number)
  ècSpecifies the vertical end position of the line (in dots). Specifies an integer from 0 to 65535.
- style : (Optional parameter, Object type : String)
  Specifies the line type.

Constant (style)	Description
LINE_THIN	Solid line: Thin
LINE_MEDIUM	Solid line: Medium
LINE_THICK	Solid line: Thick
LINE_THIN_DOUBLE	Double line: Thin
LINE_MEDIUM_DOUBLE	Double line: Medium
LINE_THICK_DOUBLE	Double line: Thick
undefined (When not specified)	Solid line: Thin

Return value

Return value	Object type
ePOS-Print Builder Object	ePOS Builder

Exception

Exception	Object type
Parameter " ... " is invalid	Error

Example

To draw a thin solid line between the start position (100, 0) and the end position (500, 0):

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
  var builder = new epson.ePOSBuilder();
  builder.addPageBegin();
  builder.addPageLine(100, 0, 500, 0, builder.LINE_THIN);
  builder.addPageEnd();
}
//-->
</script>
```

addPageRectangle method

Adds rectangle drawing in page mode to the command buffer. Draws a rectangle in page mode.

Use this API function by inserting it between addPageBegin (p. 91) and PageEnd (p. 92).

Syntax

```
addPageRectangle(x1, y1, x2, y2, style);
```

Parameter

- x 1: (Required parameter, Object type : Number)
  Specifies the horizontal start position of the line (in dots). Specifies an integer from 0 to 65535.
- y 1: (Required parameter, Object type : Number)
  Specifies the vertical start position of the line (in dots). Specifies an integer from 0 to 65535.
- x2 : (Required parameter, Object type : Number)
  Specifies the horizontal end position of the line (in dots). Specifies an integer from 0 to 65535.
- y2 : (Required parameter, Object type : Number)
  ècSpecifies the vertical end position of the line (in dots). Specifies an integer from 0 to 65535.
- style : (Optional parameter, Object type : String)
  Specifies the line type.

Constant (style)	Description
LINE_THIN	Solid line: Thin
LINE_MEDIUM	Solid line: Medium
LINE_THICK	Solid line: Thick
LINE_THIN_DOUBLE	Double line: Thin
LINE_MEDIUM_DOUBLE	Double line: Medium
LINE_THICK_DOUBLE	Double line: Thick
undefined (When not specified)	Solid line: Thin

Return value

Return value	Object type
ePOS-Print Builder Object	ePOS Builder

Exception

Exception	Object type
Parameter " ... " is invalid	Error

Example

To draw a rectangle with a thin double line, with the start position (100, 0) and the end position (500, 200) as its vertexes:

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
  var builder = new epson.ePOSBuilder();
  builder.addPageBegin();
  builder.addPageLine(100, 0, 500, 200, builder.LINE_THIN_DOUBLE);
  builder.addPageEnd();
}
//-->
</script>
```

addCut method

Adds paper cut to the command buffer. Sets paper cut.

> ⚠ Not available in page mode.

Syntax

```
addCut(type);
```

Parameter

- type : (Optional parameter, Object type : String)
  Specifies the paper cut type.

Setting	Description
CUT_NO_FEED	Cut without feeding (The paper is cut without being fed.)
CUT_FEED	Feed cut (The paper is fed to the cut position and then is cut.)
CUT_RESERVE	Cut reservation (Printing continues until the cut position is reached, at which the paper is cut.)
undefined (When not specified)	Feed cut (The paper is fed to the cut position and then is cut.)

Return value

Return value	Object type
ePOS-Print Builder Object	ePOS Builder

Exception

Exception	Object type
Parameter " ... " is invalid	Error

Example

To perform feed cut operation:

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
  var builder = new epson.ePOSBuilder();
  builder.addCut(builder.CUT_FEED);
}
//-->
</script>
```

addPulse method

Adds the drawer kick to the command buffer. Sets the drawer kick.

- Not available in page mode.
- The drawer and the buzzer cannot be used together.

Syntax

```
addPulse(drawer, time);
```

Parameter

- drawer : (Optional parameter, Object type : String)
  Specifies the drawer kick connector.

Setting	Description
DRAWER_1	Pin 2 of the drawer kick-out connector
DRAWER_2	Pin 5 of the drawer kick-out connector
undefined (When not specified)	Pin 2 of the drawer kick-out connector

- time : (Optional parameter, Object type : String)
  Specifies the ON time of the drawer kick signal.

Setting	Description
PULSE_100	100 ms
PULSE_200	200 ms
PULSE_300	300 ms
PULSE_400	400 ms
PULSE_500	500 ms
undefined (When not specified)	100 ms

Return value

Return value	Object type
ePOS-Print Builder Object	ePOS Builder

Exception

Exception	Object type
Parameter " ... " is invalid	Error

Example

To send a 100msec pulse signal to the pin 2 of the drawer kick connector:

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
  var builder = new epson.ePOSBuilder();
  builder.addPulse(builder.DRAWER_1, builder.PULSE_100);
}
//-->
</script>
```

addSound method

Adds the turning on of the buzzer to the command buffer. Sets the buzzer.

- Not available in page mode.
- The buzzer function and the drawer cannot be used together.
- This API function cannot be used if the printer is not provided with the buzzer.

Syntax

```
addSound(pattern, repeat);
```

Parameter

- pattern : (Optional parameter, Object type : String)
  Specifies the buzzer pattern.

Setting	Description
PATTERN_NONE	Stop
PATTERN_A	Pattern A
PATTERN_B	Pattern B
PATTERN_C	Pattern C
PATTERN_D	Pattern D
PATTERN_E	Pattern E
PATTERN_ERROR	Error sound pattern
PATTERN_PAPER_END	Pattern when there is no paper
undefined (When not specified)	Pattern A

- repeat : (Optional parameter, Object type : String)
  Specifies the number of repeats.

Setting	Description
0	The buzzer does not stop.
1 to 255	Number of repeats
undefined (When not specified)	One time

After “0” is specified for repeat, if you want to stop the buzzer, execute this API function and specify PATTERN_NONE for pattern.

Return value

Return value	Object type
ePOS-Print Builder Object	ePOS Builder

4

Exception

Exception	Object type
Parameter " ... " is invalid	Error

Example

To repeat the sound pattern A three times:

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
  var builder = new epson.ePOSBuilder();
  builder.addSound(builder.PATTERN_A, 3);
}
//-->
</script>
```

addCommand method

Adds commands to the command buffer. Sends ESC/POS commands.

> ESC/POS commands are not made public. For details, contact the dealer.

Syntax

```
addCommand(data);
```

Parameter

- data : (Optional parameter, Object type : String)
  Specifies ESC/POS command as a character string.

Return value

Return value	Object type
ePOS-Print Builder Object	ePOS Builder

Exception

Exception	Object type
Parameter " ... " is invalid	Error

toString method

Obtains a print document generated by an ePOS-Print Builder object.

Syntax

```
toString();
```

Return value

Return value	Object type
Document to be printed	String

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
  var builder = new epson.ePOSBuilder();
  var doc = builder.toString();
}
//-->
</script>
```

halftone property

Halftone processing method. (in Ver. 1.2 and later)

Object type

String

Description

The halftone processing method to be applied to monochrome (two-tone) printing is specified.
The default value is HALFTONE_DITHER.

Constant	Description
HALFTONE_DITHER (default)	Dithering, suitable for printing graphics only.
HALFTONE_ERROR_DIFFUSION	Error diffusion, suitable for printing text and graphics together.
HALFTONE_THRESHOLD	Threshold, suitable for printing text only.

Example

To set the halftone type as error diffusion:

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
var builder = new epson.ePOSBuilder();
var canvas = document.getElementById('canvas');
if (canvas.getContext) {
  var context = canvas.getContext('2d');
  builder.halftone = epos.HALFTONE_ERROR_DIFFUSION;
  builder.addImage(context, 0, 0, canvas.width, canvas.height);
  }
}
//-->
</script>
```

brightness property

Brightness correction value. (in Ver. 1.2 and later)

Object type

Number

Description

A gamma value in the range 0.1-10.0 is specified for the brightness correction value.
The default value is 1.0.

Example

To set brightness as 2.2:

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
var builder = new epson.ePOSBuilder();
var canvas = document.getElementById('canvas');
if (canvas.getContext) {
  var context = canvas.getContext('2d');
  builder.brightness = 2.2;
  builder.addImage(context, 0, 0, canvas.width, canvas.height);
  }
}
//-->
</script>
```

message property

Command buffer.

Object type

String

Description

Commands, which are usually added by methods of the ePOS-Print Builder object, can be operated directly from this property for addition or deletion.

Example

To clear the command and reset it to the initial state:

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function buildMessage() {
var builder = new epson.ePOSBuilder();
  builder.addText('ABCDE');
  builder.message = '';
}
//-->
</script>
```

ePOS-Print Object

Sends a print document created using an ePOS-Print Builder object to control the printer and monitor the transmission result or the communication status.

Constructor

Constructor for an ePOS-Print object. Creates a new ePOS-Print object and initializes it.

Syntax

```
ePOSPrint(address);
```

Parameter

- address : (Optional parameter, Object type : String)
  Specifies the URL of the printer to send a print document to. (in Ver. 1.2 and later)
  The URL is as follows:

http://(IP address of TM intelligent printer)/cgi-bin/epos/service.cgi?devid=(device ID of printer to be used for printing)&timeout=(timeout time)

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function sendMessage() {
  var address = 'http://192.168.192.168/cgi-bin/epos/
                 service.cgi?devid=local_printer';
  var epos = new epson.ePOSPrint(address);
}
//-->
</script>
```

send method

Sends a print document created using an ePOS-Print Builder object.

> A print document is obtained by executing the toString method (p. 110) of the ePOS-Print Builder object.

Syntax

```
send(request);
```

Parameter

- request : (Required parameter, Object type : String)
  Specifies the print document.

Exception

Exception	Object type
Parameter " ... " is invalid	Error
XMLHttpRequest is not supported	Error

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function printHellowWorld() {

  var builder = new epson.ePOSBuilder();
  builder.addText('Hello, World!\n');
  builder.addCut();
  var request = builder.toString();

  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
  var epos = new epson.ePOSPrint(address);
  epos.onreceive = function (res) { alert(res.success); };
  epos.onerror = function (err) { alert(err.status); };
  epos.send(request);
}
//-->
</script>
```

4

open method

Enables status event operation. (in Ver.1.2 and later)
Sends the status of the printer specified by the address property using an event.
Updates the status at the interval specified by the interval property.

Syntax

```
open();
```

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer'
  var epos = new epson.ePOSPrint(address);
  epos.oncoveropen = function () {
    alert('coveropen');
  };

function startMonitor() {
  epos.open();
}

function stopMonitor() {
  epos.close();
}
//-->
</script>
```

close method

Disables status event operation. (in Ver. 1.2 and later)

Syntax

```
close();
```

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer'
  var epos = new epson.ePOSPrint(address);
  epos.oncoveropen = function () {
    alert('coveropen');
  };

function startMonitor() {
  epos.open();
}

function stopMonitor() {
  epos.close();
}
//-->
</script>
```

address property

URL of the printer. (in Ver. 1.2 and later)

Object type

String

Description

The URL of the printer to be used for printing is specified.
The URL is shown as follows:

http://(IP address of TM intelligent printer)/cgi-bin/epos/service.cgi?devid=(device ID of printer to be used for printing)&timeout=(timeout time)

The default value is the address specified by the constructor.

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var epos = new epson.ePOSPrint();
  epos.address = 'http://192.168.192.168/cgi-bin/epos/
service.cgi?devid=local_printer';
  epos.oncoveropen = function () { alert('coveropen'); };
  epos.open();
//-->
</script>
```

enabled property

Retains the enabled/disabled setting for status event operation. (in Ver. 1.2 and later)

Object type

Boolean

Description

The enabled/disabled setting for status event operation is retained using a logical value. This is read-only. The default value is false.

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
  var epos = new epson.ePOSPrint(address);
  epos.oncoveropen = function () { alert('coveropen'); };
  epos.open();
  alert(epos.enabled);
//-->
</script>
```

interval property

Specifies the interval of upgrading the status. (in Ver. 1.2 and later)

Object type

Number

Description

The interval of upgrading the status is specified in milliseconds.

Default value: 3000 (three seconds)

Minimum value: 1000 (one second or longer)

When an invalid value is specified, it is assumed to be 3000.

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
  var epos = new epson.ePOSPrint(address);
  epos.interval = 1000;
  epos.oncoveropen = function () { alert('coveropen'); };
  epos.open();
//-->
</script>
```

status property

Status of the printer. (in Ver. 1.2 and later)

Object type

Number

Description

This is the status last obtained from the printer. This is read-only.
Default value: 0

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
  var epos = new epson.ePOSPrint(address);
  epos.onoffline = function () {
    alert(epos.status);
  };
  epos.open();
//-->
</script>
```

onreceive event

This property registers the callback function and obtains a response message receipt event.

Syntax

```
Function (response)
```

Parameter of the callback function

Parameters: response (See “Properties of the response object” on page 122.)

Name: Response message

Object type: Object

Properties of the response object

Property	Name	Object type
success (p. 122)	Print result	Boolean
code (p. 122)	Error code	String
status (p. 123)	Status	Number

- Value of success

Value	Description
true or 1	Printing succeeded.
false or 0	Printing failed.

- Value of code

Value	Description
'EPTR_AUTOMATICAL'	An automatically recoverable error occurred
'EPTR_COVER_OPEN'	A cover open error occurred
'EPTR_CUTTER'	An autocutter error occurred
'EPTR_MECHANICAL'	A mechanical error occurred
'EPTR_REC_EMPTY'	No paper in roll paper end sensor
'EPTR_UNRECOVERABLE'	An unrecoverable error occurred
'SchemaError'	The request document contains a syntax error
'DeviceNotFound'	The printer with the specified device ID does not exist
'PrintSystemError'	An error occurred on the printing system
'EX_BADPORT'	An error was detected on the communication port
'EX_TIMEOUT'	A print timeout occurred

- Value of status

Constant (status)	Description
ASB_NO_RESPONSE	No response from the TM printer
ASB_PRINT_SUCCESS	Printing is successfully completed
ASB_DRAWER_KICK	Status of the 3rd pin of the drawer kick-out connector = "H"
ASB_OFF_LINE	Offline
ASB_COVER_OPEN	The cover is open
ASB_PAPER_FEED	Paper is being fed by a paper feed switch operation
ASB_WAIT_ON_LINE	Waiting to be brought back online
ASB_PANEL_SWITCH	The paper feed switch is being pressed (ON)
ASB_MECHANICAL_ERR	A mechanical error occurred
ASB_AUTOCUTTER_ERR	An autocutter error occurred
ASB_UNRECOVER_ERR	An unrecoverable error occurred
ASB_AUTORECOVER_ERR	An automatically recoverable error occurred
ASB_RECEIPT_NEAR_END	No paper in roll paper near end sensor
ASB_RECEIPT_END	No paper in roll paper end sensor
ASB_BUZZER	A buzzer is on (only for applicable devices)
ASB_SPOOLER_IS_STOPPED	The spooler has stopped

Example

To create and send a print document.
To display the print result in a message box.

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function printHellowWorld() {

var builder = new epson.ePOSBuilder();
builder.addText('Hello, World!\n');
builder.addCut();
var request = builder.toString();

var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
var epos = new epson.ePOSPrint(address);
epos.onreceive = function (res) {
  var success = res.success;
  var code = res.code;
  var status = res.status;
  alert(success);
}
epos.send(request);
}
//-->
</script>
```

4

onerror event

This property registers the callback function and obtains a communication error event.

Syntax

```
Function (error)
```

Parameter of the callback function

Parameters: error (See "Properties of the error object" on page 124.)

Name: Communication error information

Object type: Object

Properties of the error object

Property	Name	Object type
status	HTTP Status	Number
responseText	Response text	String

Example

To create and send a print document.
To display the HTTP status code in a message box when a communication error occurs.

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function printHellowWorld() {

var builder = new epson.ePOSBuilder();
builder.addText('Hello, World!\n');
builder.addCut();
var request = builder.toString();

var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
var epos = new epson.ePOSPrint(address);
epos.onerror = function (err) {
  var status = err.status;
  var text = err.responseText;
  alert(status);
}
epos.send(request);
}
//-->
</script>
```

onstatuschange event

Registers a callback function to obtain a status change event. (in Ver. 1.2 and later)

Syntax

```
Function (status)
```

Parameter of the callback function

Parameters:	status
Name:	Status
Object type:	Number

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
  var epos = new epson.ePOSPrint(address);
  epos.onstatuschange = function (status) {
    alert(status);
  };
  epos.open();
//-->
</script>
```

ononline event

Registers a callback function to obtain a online event. (in Ver. 1.2 and later)

Object type

Function ()

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
  var epos = new epson.ePOSPrint(address);
  epos.ononline = function () {
    alert('online');
  };
  epos.open();
//-->
</script>
```

onoffline event

Registers a callback function to obtain a offline event. (in Ver. 1.2 and later)

Object type

Function ()

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
  var epos = new epson.ePOSPrint(address);
  epos.onoffline = function () {
    alert('offline');
  };
  epos.open();
//-->
</script>
```

onpoweroff event

Registers a callback function to obtain a non-response event. (in Ver. 1.2 and later)

Object type

Function ()

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
  var epos = new epson.ePOSPrint(address);
  epos.onpoweroff = function () {
    alert('poweroff');
  };
  epos.open();
//-->
</script>
```

oncoverok event

Registers a callback function to obtain a cover close event. (in Ver. 1.2 and later)

Object type

Function ()

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
  var epos = new epson.ePOSPrint(address);
  epos.oncoverok = function () {
    alert('coverok');
  };
  epos.open();
//-->
</script>
```

oncoveropen event

Registers a callback function to obtain a cover open event. (in Ver. 1.2 and later)

Object type

Function ()

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
  var epos = new epson.ePOSPrint(address);
  epos.oncoveropen = function () {
    alert('coveropen');
  };
  epos.open();
//-->
</script>
```

onpaperok event

Registers a callback function to obtain a paper remaining event. (in Ver. 1.2 and later)

Object type

Function ()

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
  var epos = new epson.ePOSPrint(address);
  epos.onpaperok = function () {
    alert('paperok');
  };
  epos.open();
//-->
</script>
```

onpapernearend event

Registers a callback function to obtain a paper near end event. (in Ver. 1.2 and later)

Object type

Function ()

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
  var epos = new epson.ePOSPrint(address);
  epos.onpapernearend = function () {
    alert('papernearend');
  };
  epos.open();
//-->
</script>
```

onpaperend event

Registers a callback function to obtain a paper end event. (in Ver. 1.2 and later)

Object type

Function ()

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
  var epos = new epson.ePOSPrint(address);
  epos.onpaperend = function () {
    alert('paperend');
  };
  epos.open();
//-->
</script>
```

ondrawerclosed event

Registers a callback function to obtain a drawer close event. (in Ver. 1.2 and later)

Object type

Function ()

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
  var epos = new epson.ePOSPrint(address);
  epos.ondrawerclosed = function () {
    alert('drawerclosed');
  };
  epos.open();
//-->
</script>
```

ondraweropen event

Registers a callback function to obtain a drawer open event. (in Ver. 1.2 and later)

Object type

Function ()

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
  var epos = new epson.ePOSPrint(address);
  epos.ondraweropen = function () {
    alert('draweropen');
  };
  epos.open();
//-->
</script>
```

ePOS-Print Canvas API

This chapter describes the ePOS-Print Canvas API.

List of ePOS-Print Canvas API functions

The ePOS-Print Canvas API provides the following object:

- ePOS-Print Canvas API (window.epson.CanvasPrint) object (p. 131)

window.epson.CanvasPrint Components

Element	API	Description	Page
Constructor			
	CanvasPrint	Initializes an ePOS-Print Canvas API object.	133
method			
	print	Prints an HTML5 Canvas image.	134
	open	Enables status event operation (in Ver. 1.2 and later)	136
	close	Disables status event operation (in Ver. 1.2 and later)	137
Property			
	address	URL of the printer (in Ver. 1.2 and later)	138
	enabled	Enabling/disabling of status event (in Ver. 1.2 and later)	139
	interval	Printer status update interval (in Ver. 1.2 and later)	140
	status	Status (in Ver. 1.2 and later)	141
	halftone	Raster image halftone processing method (in Ver. 1.2 and later)	142
	brightness	Raster image brightness correction value (in Ver. 1.2 and later)	143
Event			
	onreceive	Response message receipt event	144
	onerror	Communication error event	146
	onstatuschange	Status change event (in Ver. 1.2 and later)	147
	ononline	Online event (in Ver. 1.2 and later)	147
	onoffline	Offline event (in Ver. 1.2 and later)	148
	onpoweroff	Non-response event (in Ver. 1.2 and later)	148
	oncoverok	Cover close event (in Ver. 1.2 and later)	149

Element	API	Description	Page
	oncoveropen	Cover open event (in Ver. 1.2 and later)	149
	onpaperok	Paper remaining event (in Ver. 1.2 and later)	150
	onpapernearend	Paper near end event (in Ver. 1.2 and later)	150
	onpaperend	Paper end event (in Ver. 1.2 and later)	151
	ondrawerclosed	Drawer close event (in Ver. 1.2 and later)	151
	ondraweropen	Drawer open event (in Ver. 1.2 and later)	152
Constant			
	ASB_*	Response document status	
	HALFTONE_*	Halftone type	
	MODE_*	Color mode	

ePOS-Print Canvas API Object

Prints a print image rendered in HTML5 Canvas and monitors the print result or the communication status.

Constructor

Constructor for an ePOS-Print Canvas API object.
Creates a new ePOS-Print Canvas API object and initializes it.

Syntax

CanvasPrint(address);

Parameter

- address : (Optional parameter, Object type : String)
  Specifies the address property (URL of printer to be used for printing).
  The URL is as follows:

  http://(IP address of TM intelligent printer)/cgi-bin/epos/service.cgi?devid=(device ID of printer to be used for printing)&timeout=(timeout time)

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function printCanvas() {
 var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
 var epos = new epson.CanvasPrint(address);
}
//-->
</script>
```

print method

Prints an image rendered in HTML5 Canvas.
Converts the specified range in a RGBA full-color image of HTML5 Canvas into raster image data according to the settings of the halftone and brightness properties. One pixel in an image equals to one printer dot.
When an image contains any transparent color, the background color of the image is assumed to be white.

> ⚠ If an HTML5 Canvas image contains images downloaded from different domains, you cannot print the image. In this case, a security error occurs due to violation of the same origin policy of JavaScript.

Syntax

```
print(canvas, cut, mode);
```

Parameter

- canvas : (Required parameter, Object type : canvas)
  Specify the HTML5 Canvas object to be printed.
- cut : (Optional parameter, Object type : Boolean)
  Sets whether to cut paper.

Setting	Decription
true or 1	Cuts the paper after printing
false or 0	Does not cut the paper after printing
undefined	Does not cut the paper after printing

- mode : (Optional parameter, Object type : String)
  Specifies the color mode. (in Ver. 1.2 and later)

Setting	Decription
MODE_MONO	Monochrome (two-tone)
MODE_GRAY16	Multiple tones (16-tone)
undefined	Monochrome (two-tone)

Exception

Exception	Object type
Parameter " ... " is invalid	Error
XMLHttpRequest is not supported	Error
Canvas is not supported	Error

Example

To print Canvas(ID='myCanvas'):

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function printCanvas() {
var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
var canvas = document.getElementById('myCanvas');

var epos = new epson.CanvasPrint(address);
epos.onreceive = function (res) { alert(res.success); };
epos.onerror = function (err) { alert(err.status); };
epos.print(canvas);
}
//-->
</script>
```

open method

Enables status event operation. (in Ver.1.2 and later)
Sends the status of the printer specified by the address property using an event.
Updates the status at the interval specified by the interval property.

Syntax

```
open();
```

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
var epos = new epson.CanvasPrint(address);
  epos.oncoveropen = function () {
    alert('coveropen');
  };

function startMonitor() {
  epos.open();
}

function stopMonitor() {
  epos.close();
}
//-->
</script>
```

close method

Disables status event operation. (in Ver. 1.2 and later)

Syntax

```
close();
```

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
var epos = new epson.CanvasPrint(address);
  epos.oncoveropen = function () {
    alert('coveropen');
  };

function startMonitor() {
  epos.open();
}

function stopMonitor() {
  epos.close();
}
//-->
</script>
```

address property

URL of the printer. (in Ver. 1.2 and later)

Object type

String

Description

The URL of the printer to be used for printing is specified.
The URL is shown as follows:

http://(IP address of TM intelligent printer)/cgi-bin/epos/service.cgi?devid=(device ID of printer to be used for printing)&timeout=(timeout time)

The default value is the address specified by the constructor.

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var epos = new epson.CanvasPrint();
  epos.address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
  epos.oncoveropen = function () { alert('coveropen'); };
  epos.open();
//-->
</script>
```

enabled property

Retains the enabled/disabled setting for status event operation. (in Ver. 1.2 and later)

Object type

Boolean

Description

The enabled/disabled setting for status event operation is retained using a logical value. This is read-only. The default value is false.

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
  var epos = new epson.CanvasPrint(address);
  epos.oncoveropen = function () { alert('coveropen'); };
  epos.open();
  alert(epos.enabled);
//-->
</script>
```

interval property

Specifies the interval of upgrading the status. (in Ver. 1.2 and later)

Object type

Number

Description

The interval of upgrading the status is specified in milliseconds.

Default value: 3000 (three seconds)

Minimum value: 1000 (one second or longer)

When an invalid value is specified, it is assumed to be 3000.

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
  var epos = new epson.CanvasPrint(address);
  epos.interval = 1000;
  epos.oncoveropen = function () { alert('coveropen'); };
  epos.open();
//-->
</script>
```

status property

Status of the printer. (in Ver. 1.2 and later)

Object type

Number

Description

This is the status last obtained from the printer. This is read-only.
Default value: 0

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
  var epos = new epson.CanvasPrint(address);
  epos.onoffline = function () {
    alert(epos.status);
  };
  epos.open();
//-->
</script>
```

halftone property

Halftone processing method. (in Ver. 1.2 and later)

Object type

String

Description

The halftone processing method to be applied to monochrome (two-tone) printing is specified.
The default value is HALFTONE_DITHER.

Constant	Description
HALFTONE_DITHER	Dithering, suitable for printing graphics only.
HALFTONE_ERROR_DIFFUSION	Error diffusion, suitable for printing text and graphics together.
HALFTONE_THRESHOLD	Threshold, suitable for printing text only.

Example

To set the halftone type as error diffusion:

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
  var canvas = document.getElementById('myCanvas');

  var epos = new epson.CanvasPrint(address);
  epos.halftone = epos.HALFTONE_ERROR_DIFFUSION;
  epos.print(canvas);
//-->
</script>
```

brightness property

Brightness correction value. (in Ver. 1.2 and later)

Object type

Number

Description

A gamma value in the range 0.1-10.0 is specified for the brightness correction value.
The default value is 1.0.

Example

To set brightness as 2.2:

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
  var canvas = document.getElementById('myCanvas');

  var epos = new epson.CanvasPrint(address);
  epos.brightness = 2.2;
  epos.print(canvas);
//-->
</script>
```

onreceive event

This property registers the callback function and obtains a response message receipt event.

Syntax

```
Function (response)
```

Parameter of the callback function

Parameter: response (See "Properties of the response object" on page 144.)

Name: Response message

Object type: Object

Properties of the response object

Parameter	Name	Object type
success (p. 144)	Print result	Boolean
code (p. 144)	Error code	String
status (p. 145)	Status	Number

- Value of success

Value	Decription
true or 1	Printing succeeded.
false or 0	Printing failed.

- Value of code

Value	Decription
'EPTR_AUTOMATICAL'	An automatically recoverable error occurred
'EPTR_COVER_OPEN'	A cover open error occurred
'EPTR_CUTTER'	An autocutter error occurred
'EPTR_MECHANICAL'	A mechanical error occurred
'EPTR_REC_EMPTY'	No paper in roll paper end sensor
'EPTR_UNRECOVERABLE'	An unrecoverable error occurred
'SchemaError'	The request document contains a syntax error
'DeviceNotFound'	The printer with the specified device ID does not exist
'PrintSystemError'	An error occurred on the printing system
'EX_BADPORT'	An error was detected on the communication port
'EX_TIMEOUT'	A print timeout occurred

- Value of satus

Constant (status)	Decription
ASB_NO_RESPONSE	No response from the TM printer
ASB_PRINT_SUCCESS	Printing is successfully completed
ASB_DRAWER_KICK	Status of the 3rd pin of the drawer kick-out connector = "H"
ASB_OFF_LINE	Offline
ASB_COVER_OPEN	The cover is open
ASB_PAPER_FEED	Paper is being fed by a paper feed switch operation
ASB_WAIT_ON_LINE	Waiting to be brought back online
ASB_PANEL_SWITCH	The paper feed switch is being pressed (ON)
ASB_MECHANICAL_ERR	A mechanical error occurred
ASB_AUTOCUTTER_ERR	An autocutter error occurred
ASB_UNRECOVER_ERR	An unrecoverable error occurred
ASB_AUTORECOVER_ERR	An automatically recoverable error occurred
ASB_RECEIPT_NEAR_END	No paper in roll paper near end sensor
ASB_RECEIPT_END	No paper in roll paper end sensor
ASB_BUZZER	A buzzer is on (only for applicable devices)
ASB_SPOOLER_IS_STOPPED	The spooler has stopped

Example

To print Canvas(ID=myCanvas):
To display the print result in a message box.

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function printCanvas() {
var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
var canvas = document.getElementById('myCanvas');

var epos = new epson.CanvasPrint(address);
epos.onreceive = function (res) {
  var success = res.success;
  var code = res.code;
  var status = res.status;
  alert(success);
};
epos.print(canvas);
}
//-->
</script>
```

onerror event

This property registers the callback function and obtains a communication error event.

Syntax

```
Function (error)
```

Parameter of the callback function

Parameter:	error (See "Properties of the error object" on page 146.)
Name:	Communication error information
Object type:	Object

Properties of the error object

property	Name	Object type
status	HTTP status	Number
responseText	Responce text	String

Example

To print Canvas(ID=myCanvas):
To display the HTTP status code in a message box when a communication error occurs.

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
function printCanvas() {
var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
var canvas = document.getElementById('myCanvas');

var epos = new epson.CanvasPrint(address);
epos.onerror = function (err) {
  var status = err.status;
  var text = err.responseText;
  alert(status);
};
epos.print(canvas);
}
//-->
</script>
```

onstatuschange event

Registers a callback function to obtain a status change event. (in Ver. 1.2 and later)

Syntax

```
Function (status)
```

Parameter of the callback function

Parameters:	status
Name:	Status
Object type:	Number

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
  var epos = new epson.CanvasPrint(address);
  epos.onstatuschange = function (status) {
    alert(status);
  };
  epos.open();
//-->
</script>
```

ononline event

Registers a callback function to obtain a online event. (in Ver. 1.2 and later)

Object type

Function ()

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
  var epos = new epson.CanvasPrint(address);
  epos.ononline = function () {
    alert('online');
  };
  epos.open();
//-->
</script>
```

onoffline event

Registers a callback function to obtain a offline event. (in Ver. 1.2 and later)

Object type

Function ()

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
  var epos = new epson.CanvasPrint(address);
  epos.onoffline = function () {
    alert('offline');
  };
  epos.open();
//-->
</script>
```

onpoweroff event

Registers a callback function to obtain a non-response event. (in Ver. 1.2 and later)

Object type

Function ()

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
  var epos = new epson.CanvasPrint(address);
  epos.onpoweroff = function () {
    alert('poweroff');
  };
  epos.open();
//-->
</script>
```

oncoverok event

Registers a callback function to obtain a cover close event. (in Ver. 1.2 and later)

Object type

Function ()

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
  var epos = new epson.CanvasPrint(address);
  epos.oncoverok = function () {
    alert('coverok');
  };
  epos.open();
//-->
</script>
```

oncoveropen event

Registers a callback function to obtain a cover open event. (in Ver. 1.2 and later)

Object type

Function ()

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
  var epos = new epson.CanvasPrint(address);
  epos.oncoveropen = function () {
    alert('coveropen');
  };
  epos.open();
//-->
</script>
```

onpaperok event

Registers a callback function to obtain a paper remaining event. (in Ver. 1.2 and later)

Object type

Function ()

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
  var epos = new epson.CanvasPrint(address);
  epos.onpaperok = function () {
    alert('paperok');
  };
  epos.open();
//-->
</script>
```

onpapernearend event

Registers a callback function to obtain a paper near end event. (in Ver. 1.2 and later)

Object type

Function ()

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
  var epos = new epson.CanvasPrint(address);
  epos.onpapernearend = function () {
    alert('papernearend');
  };
  epos.open();
//-->
</script>
```

onpaperend event

Registers a callback function to obtain a paper end event. (in Ver. 1.2 and later)

Object type

Function ()

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
  var epos = new epson.CanvasPrint(address);
  epos.onpaperend = function () {
    alert('paperend');
  };
  epos.open();
//-->
</script>
```

ondrawerclosed event

Registers a callback function to obtain a drawer close event. (in Ver. 1.2 and later)

Object type

Function ()

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
  var epos = new epson.CanvasPrint(address);
  epos.ondrawerclosed = function () {
    alert('drawerclosed');
  };
  epos.open();
//-->
</script>
```

ondraweropen event

Registers a callback function to obtain a drawer open event. (in Ver. 1.2 and later)

Object type

Function ()

Example

```
<script type="text/javascript" src="epos-print-1.2.x.js"></script>
<script type="text/javascript">
<!--
  var address = 'http://192.168.192.168/cgi-bin/epos/service.cgi?devid=local_printer';
  var epos = new epson.CanvasPrint(address);
  epos.ondraweropen = function () {
    alert('draweropen');
  };
  epos.open();
//-->
</script>
```

Appendix

Printer specifications

TM-T88V-i

		80mm
Interface		Ethernet, Wireless LAN
Resolution		180 dpi x 180 dpi (W x H)
Print Width		512 dots
Characters in a Line	Font A	ANK: 42 characters
	Font B	ANK: 56 characters
Character Size	Font A	ANK: 12 dots x 24 dots (W x H)
	Font B	ANK: 9 dots x 17 dots (W x H)
Character Baseline	Font A	At the 21st dot from the top of the character
	Font B	At the 16th dot from the top of the character
Default Line Feed Space		30 dots
Color Specification		First color
Page Mode Default Area		512 dots x 831 dots (W x H)
Page Mode Maximum Area		512 dots x 1662 dots (W x H)
Raster image		Monochrome image, two-color image
Logo		Monochrome image, two-color image
Bar Code		UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF, CODABAR, CODE93, CODE128, GS1-128,GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Limited, GS1 Databar Expanded
Two-Dimensional Code		PDF417, QR Code, MaxiCode, GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional, GS1 DataBar Expanded Stacked (Composite Symbology not supported)

	80mm
Ruled Line	Not supported
Paper Cut	Cut, Feed cut
Drawer Kick-Out	Supported
Buzzer	Optional

TM-T88V

		80mm
Interface		Ethernet, Wireless LAN
Resolution		180 dpi x 180 dpi (W x H)
Print Width		512 dots
Characters in a Line	Font A	ANK: 42 characters
	Font B	ANK: 56 characters
Character Size	Font A	ANK: 12 dots x 24 dots (W x H)
	Font B	ANK: 9 dots x 17 dots (W x H)
Character Baseline	Font A	At the 21st dot from the top of the character
	Font B	At the 16th dot from the top of the character
Default Line Feed Space		30 dots
Color Specification		First color
Page Mode Default Area		512 dots x 831 dots (W x H)
Page Mode Maximum Area		512 dots x 1662 dots (W x H)
Raster image		Monochrome image, two-color image
Logo		Monochrome image, two-color image
Bar Code		UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF, CODABAR, CODE93, CODE128, GS1-128,GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Limited, GS1 Databar Expanded
Two-Dimensional Code		PDF417, QR Code, MaxiCode, GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional, GS1 DataBar Expanded Stacked (Composite Symbology not supported)
Ruled Line		Not supported
Paper Cut		Cut, Feed cut
Drawer Kick-Out		Supported
Buzzer		Optional

TM-T70-i

		80mm
Interface		Ethernet, Wireless LAN
Resolution		180 dpi x 180 dpi (W x H)
Print Width		512 dots
Characters in a Line	Font A	ANK: 42 characters
	Font B	ANK: 56 characters
Character Size	Font A	ANK: 12 dots x 24 dots (W x H)
	Font B	ANK: 9 dots x 17 dots (W x H)
Character Baseline	Font A	At the 21st dot from the top of the chara
	Font B	At the 15th dot from the top of the character
Default Line Feed Space		30 dots
Color Specification		First color
Page Mode Default Area		512 dots x 1662 dots (W x H)
Page Mode Maximum Area		512 dots x 1662 dots (W x H)mv
Raster image		Monochrome image
Logo		Monochrome image
Bar Code		UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF, CODABAR, CODE93, CODE128
Two-Dimensional Code		PDF417, QR Code
Ruled Line		Not supported
Paper Cut		Cut, Feed cut
Drawer Kick-Out		Supported
Buzzer		Not supported

TM-T70

		80mm
Interface		Ethernet, Wireless LAN
Resolution		180 dpi x 180 dpi (W x H)
Print Width		512 dots
Characters in a Line	Font A	ANK: 42 characters
	Font B	ANK: 56 characters
Character Size	Font A	ANK: 12 dots x 24 dots (W x H)
	Font B	ANK: 9 dots x 17 dots (W x H)
Character Baseline	Font A	At the 21st dot from the top of the chara
	Font B	At the 15th dot from the top of the character
Default Line Feed Space		30 dots
Color Specification		First color
Page Mode Default Area		512 dots x 1662 dots (W x H)
Page Mode Maximum Area		512 dots x 1662 dots (W x H)mv
Raster image		Monochrome image
Logo		Monochrome image
Bar Code		UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF, CODABAR, CODE93, CODE128
Two-Dimensional Code		PDF417, QR Code
Ruled Line		Not supported
Paper Cut		Cut, Feed cut
Drawer Kick-Out		Supported
Buzzer		Not supported

TM-T90

		58mm	60mm	80mm
Interface		Ethernet, Wireless LAN		
Resolution		180 dpi x 180 dpi (W x H)		
Print Width		360 dots	384 dots	512 dots
Font		For more information about what character codes can be printed, refer to the user's manual that came with the printer.		
Characters in a Line	Font A	ANK: 30 characters,	ANK: 32 characters	ANK: 42 characters
	Font B	ANK: 40 characters	ANK: 42 characters	ANK: 56 characters
Character Size	Font A	ANK: 12 dots x 24 dots (W x H)		
	Font B	ANK: 9 dots x 17 dots (W x H)		
Character Baseline	Font A	At the 21st dot from the top of the character		
	Font B	At the 16 th dot from the top of the character		
Default Line Feed Space		30 dots		
Color Specification		First color Second color, Second color (when two-color printing is set)		
Raster Image<image>		Monochrome image, Two color image		
Logo<logo>		Monochrome image, Two color image (To perform two-color printing, change the settings of the printer using the memory switch setting utility.)		
Bar Code <barcode>		UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF, CODABAR, CODE93, CODE128		
Two-Dimensional Code <symbol>		PDF417		
Ruled Line <hline>, <vline-xxx>		Not supported		
Page Mode Default Area		360 dots x 831 dots (W x H)	384 dots x 831 dots (W x H)	512 dots x 831 dots (W x H)
	when two-color printing is set	360 dots x 415 dots (W x H)	384 dots x 415 dots (W x H)	512 dots x 415 dots (W x H)

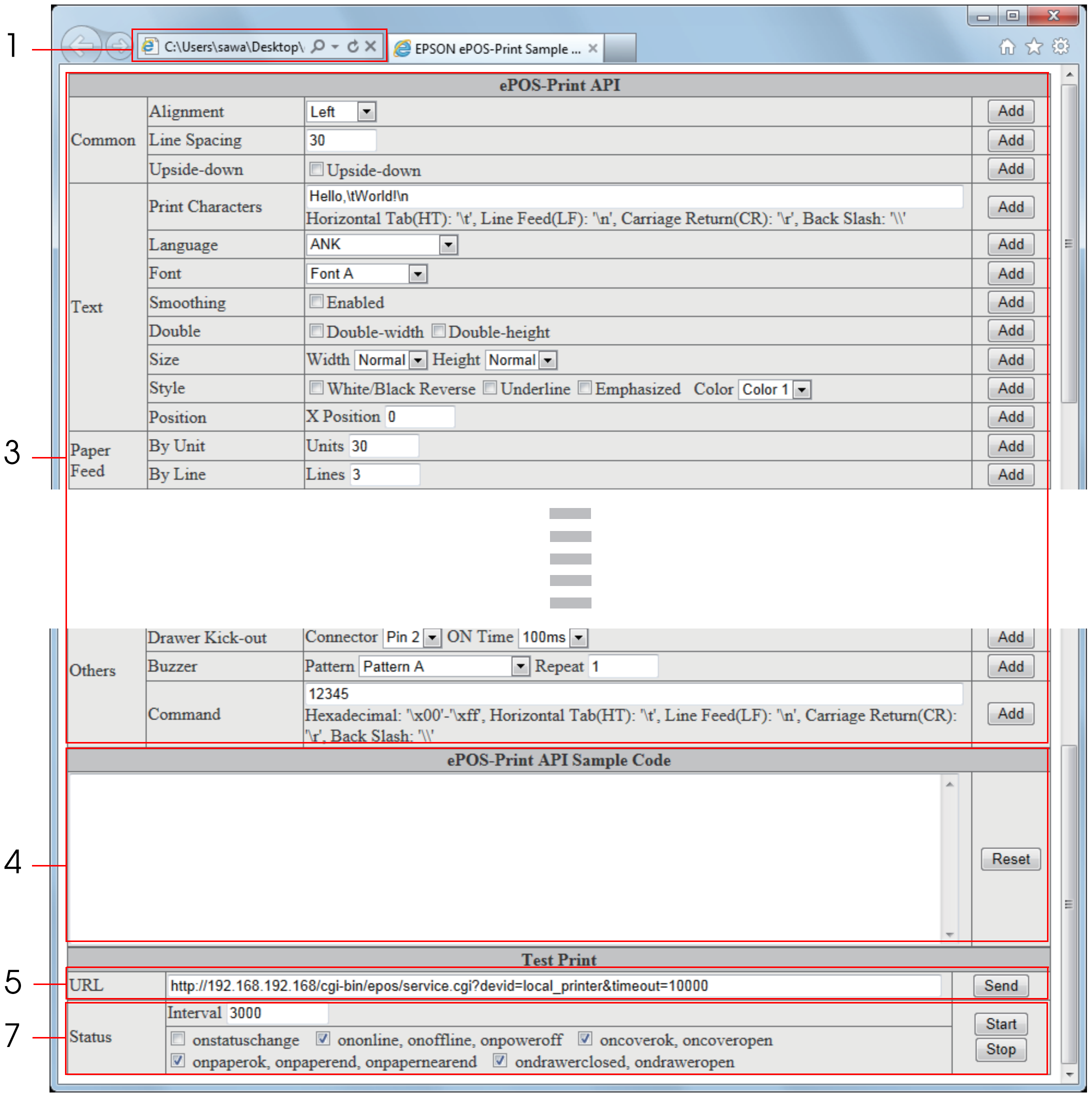
		58mm	60mm	80mm
Page Mode Maximum Area		360 dots x 1662 dots (W x H)	384 dots x 1662 dots (W x H)	512 dots x 1662 dots (W x H)
when two-color printing is set		360 dots x 831 dots (W x H)	384 dots x 831 dots (W x H)	512 dots x 831 dots (W x H)
Page Mode	Line<line>	Not supported		
	Rectangle <rectangle>			
Paper Cut <cut>		Cut, Feed cut		
Drawer Kick-Out <pulse>		Supported		
Buzzer <sound>		Supported via Drawer Kick-Out		
Command <command>		Supported		

ePOS-Print API code creation tool

This section describes how to use the ePOS-Print API code creation tool in the package.
This tool helps to arbitrarily create an ePOS-Print API (p. 53) sample code. Use this tool for your Web application development.

> This tool is embedded into the sample program. For the details about how to place this tool, refer to Environment Settings (p.26).

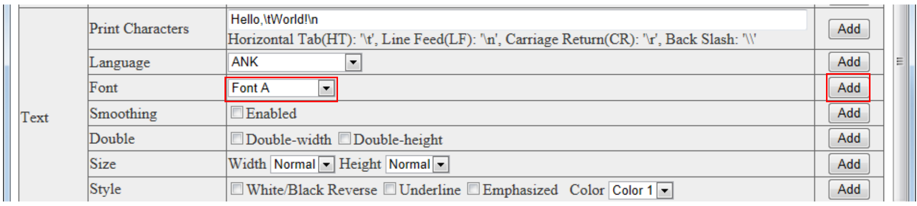
How to Use



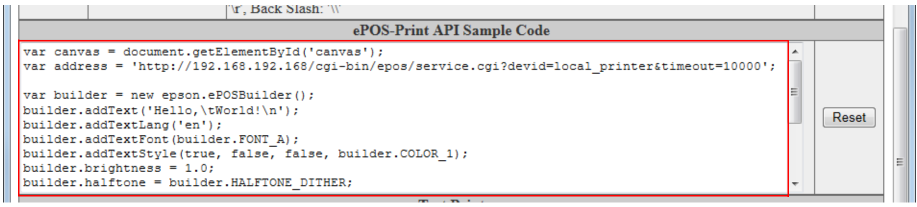
1 Open the following URL page using the Web browser.
http://(Web server IP address)/sample/epos-print-api.html

2 "EPSON ePOS-Print Sample Program" appears.

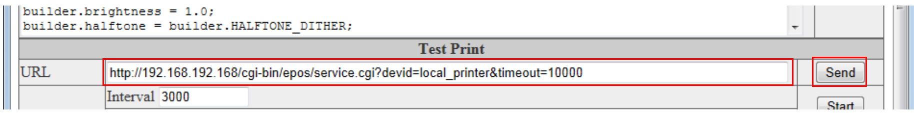
3 Set ePOS-Print API functions and click the (Add) button.



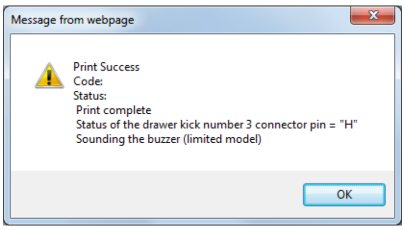
4 The source code for the functions added in procedure 3 is displayed in (ePOS-Print API sample code). You can also copy it for use.



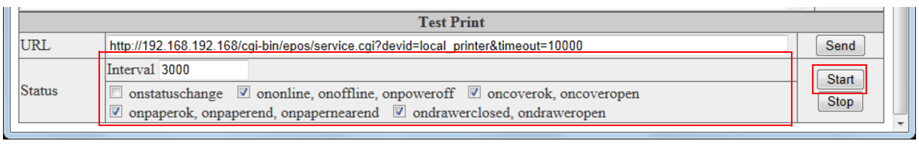
5 Enter the following URL into (URL) and click the (Send) button.
http://(IP address of TM intelligent printer)/cgi-bin/epos/service.cgi?devid=(device ID of printer to be used for printing)&timeout=(timeout time)



6 The print result is displayed.



7 To enable the status event function, click the (Start) button.
You can select the type for the event to send.



8 To disable the status event function, click the (Stop) button.

Rendering in HTML5 Canvas

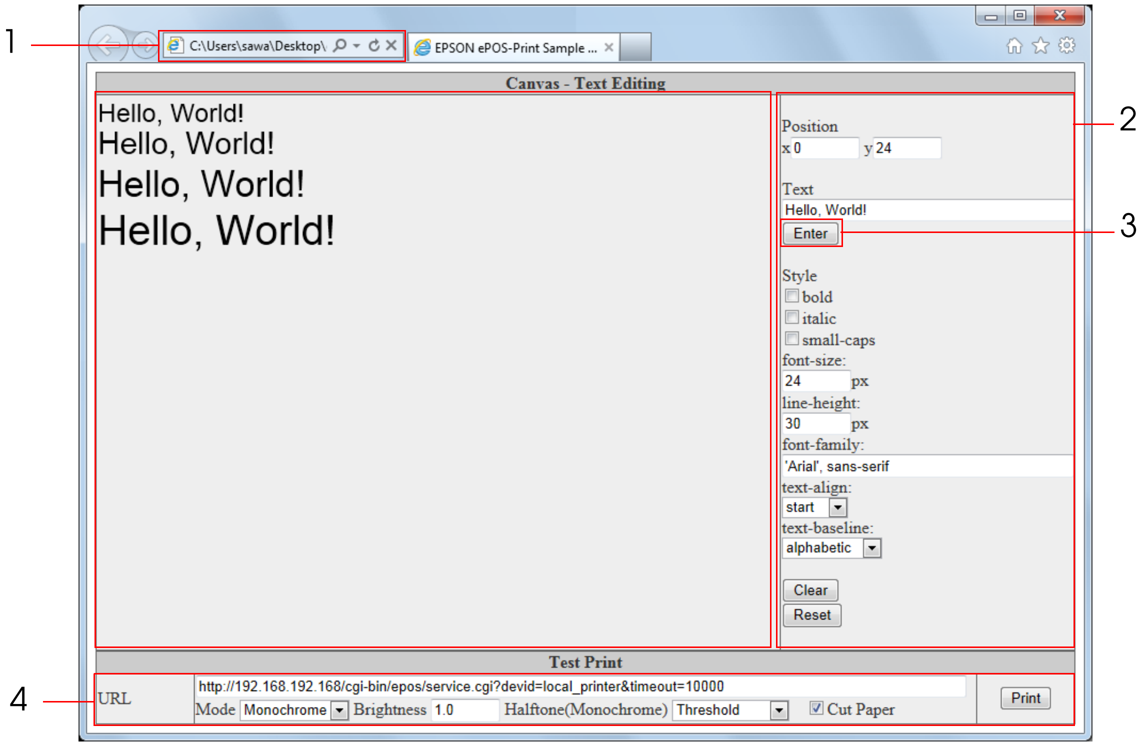
This section describes how to use Web pages using the ePOS-Print Canvas API in the package.
You can try how to render images in HTML5 Canvas and see what images can be rendered.
The following Web pages are available:

- Rendering Text (canvas-print-text.html) (p.162)
- Rendering Images (canvas-print-image.html) (p.164)
- Rendering Graphics (canvas-print-graph.html) (p.166)
- Rendering Handwritten Images (canvas-print-hand.html) (p.168)
- Rendering Barcode (canvas-print-barcode.html) (p.170)

The Web pages introduced here are embedded into the sample program. For the details about how to place them, refer to Environment Settings (p.26).

Rendering Text (canvas-print-text.html)

Print text in HTML5 Canvas and perform a test print.



1 Open the following URL page using the Web browser.
http://(Web server IP address)/sample/canvas-print-text.html

2 "EPSON ePOS-Print Sample Program" appears.
Set items on the right of the page. The following items can be set:

Item	Dscription
Position	Specify the rendering coordinates
Text	Specify the text to be printed
Style	Specify the text style
Clear	Clears the image drawn in the Canvas
Reset	Clears the image drawn in the Canvas. In addition, the settings are reset to their default values.

3 Click the (Enter) button.
The text is printed on Canvas on the left of the page according to the settings made on the right of the page.

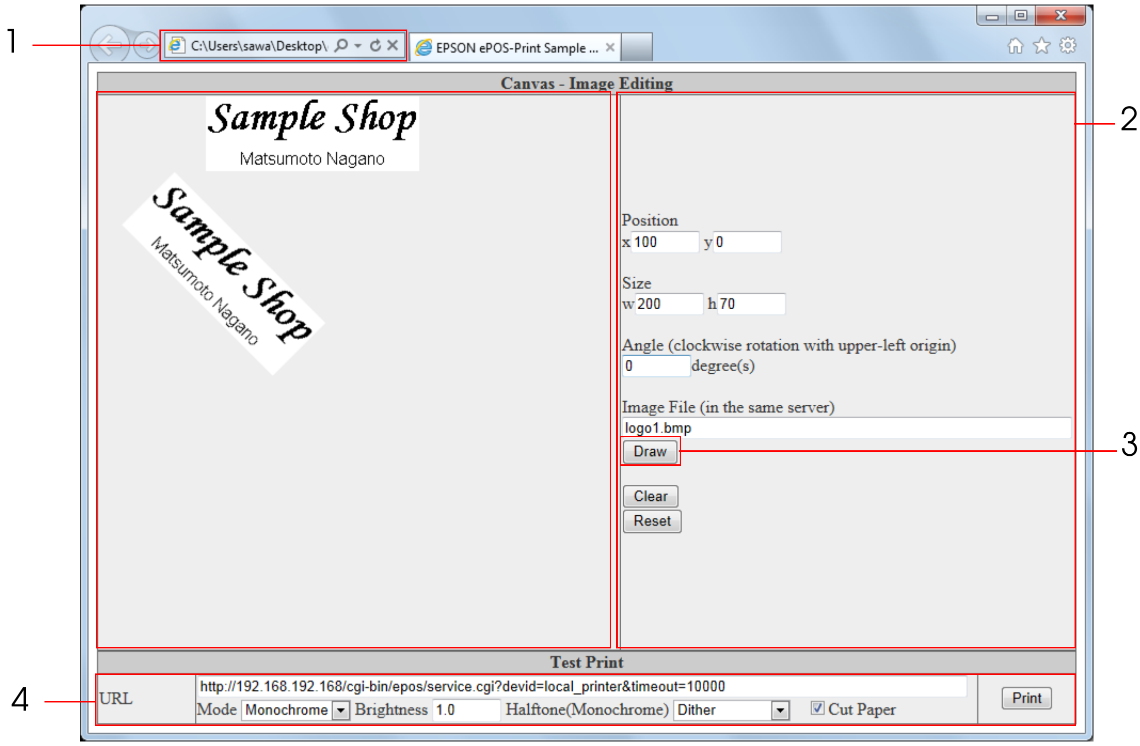
4 Set the following and click the (Print) button.

Item	Dscription
URL	Enter the following URL: http://(IP address of TM intelligent printer)/cgi-bin/epos/service.cgi?devid=(device ID of printer to be used for printing)&timeout=(timeout time)
Mode	Set the color mode (Monochrome, Grayscale).
Brightness	Adjust the brightness. (Gamma value in the range 0.1-10.0)
Halftone	Set the halftone processing method for monochrome printing (two-tone).
Cut Paper	When this item is selected, feed cut is performed after printing.

5 The print result is displayed.

Rendering Images (canvas-print-image.html)

Draw an image in HTML5 Canvas and perform a test print.



1 Open the following URL page using the Web browser.

http://(Web server IP address)/sample/canvas-print-image.html

2 “EPSON ePOS-Print Sample Program” appears.

Set items on the right of the page. The following items can be set:

Item	Description
Position	Specify the rendering coordinates
Size	Specify the width and height of the image.
Angle	Specify the rotation angle of the image. The rotation angle is counted clockwise from the top left corner.
Image File (in the same server)	Specify the path to the image file. In this Web page, specify the name of an image file placed under the same directory as this Web page.
Clear	Clears the image drawn in the Canvas.
Reset	Clears the image drawn in the Canvas. In addition, the settings are reset to their default values.

3 Click the (Draw) button.
The image is drawn on Canvas on the left of the page according to the settings made on the right of the page.

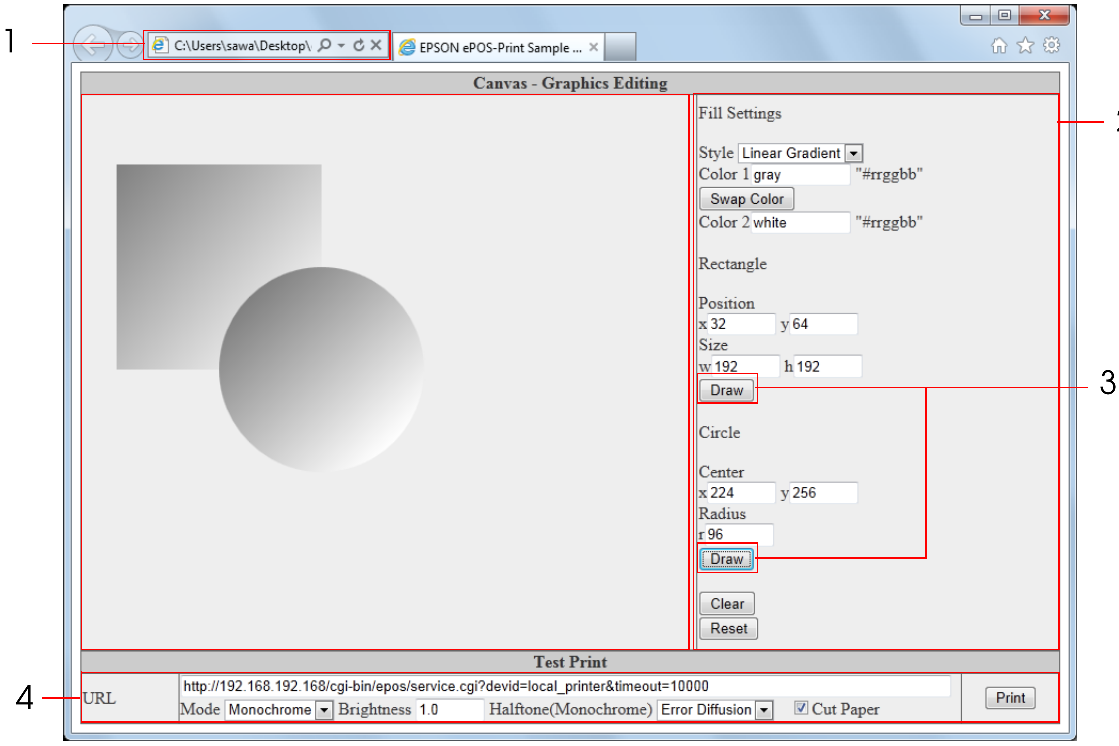
4 Set the following and click the (Print) button.

Item	Description
URL	Enter the following URL: http://(IP address of TM intelligent printer)/cgi-bin/epos/ service.cgi?devid=(device ID of printer to be used for printing)&timeout=(timeout time)
Mode	Set the color mode (Monochrome, Grayscale).
Brightness	Adjust the brightness. (Gamma value in the range 0.1-10.0)
Halftone	Set the halftone processing method for monochrome printing (two-tone).
Cut Paper	When this item is selected, feed cut is performed after printing.

5 The print result is displayed.

Rendering Graphics (canvas-print-graph.html)

Draw an image in HTML5 Canvas and perform a test print.



1 Open the following URL page using the Web browser.
http://(Web server IP address)/sample/canvas-print-graph.html

2 "EPSON ePOS-Print Sample Program" appears.
Set items on the right of the page. The following items can be set:

Item	Desction
Fill Settings	Specify the fill type and color
Rectangle	Specify the start coordinates, width and height.
Circle	Specify the central coordinates and radius.
Clear	Clears the image drawn in the Canvas
Reset	Clears the image drawn in the Canvas. In addition, the settings are reset to their default values.

3 Click the (Draw) button.
The image is drawn on Canvas on the left of the page according to the settings made on the right of the page.

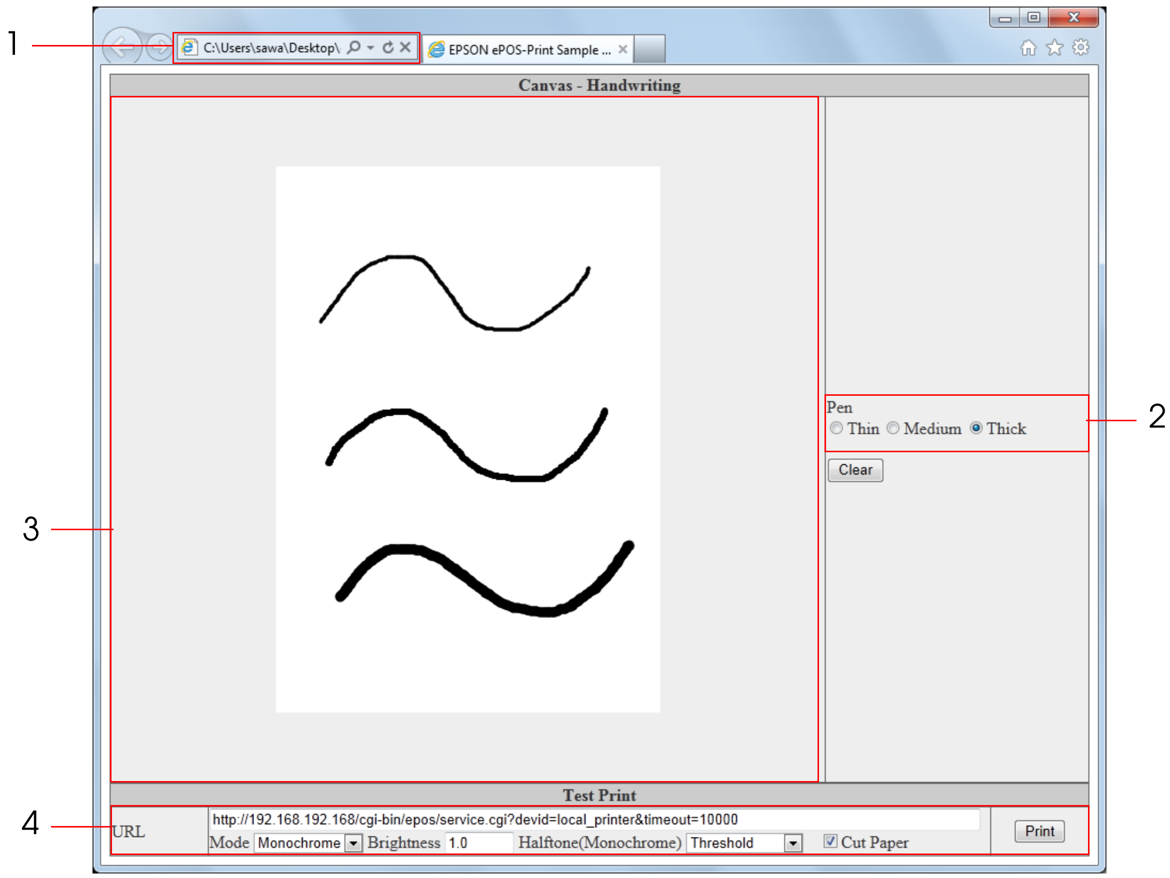
4 Set the following and click the (Print) button.

Item	Description
URL	Enter the following URL: http://(IP address of TM intelligent printer)/cgi-bin/epos/ service.cgi?devid=(device ID of printer to be used for printing)&timeout=(timeout time)
Mode	Set the color mode (Monochrome, Grayscale).
Brightness	Adjust the brightness. (Gamma value in the range 0.1-10.0)
Halftone	Set the halftone processing method for monochrome printing (two-tone).
Cut Paper	When this item is selected, feed cut is performed after printing.

5 The print result is displayed.

Rendering Handwritten Images (canvas-print-hand.html)

Draw a handwritten image and perform a test print.



1 Open the following URL page using the Web browser.
http://(Web server IP address)/sample/canvas-print-hand.html

2 "EPSON ePOS-Print Sample Program" appears. Set the size of the pen on the right of the page.

3 Draw a freehand line on Canvas on the left of the page. For the mouse, drag it to draw a line; for the touch screen monitor, draw a line on the touch screen.

> To erase the drawn image, click the [Clear] button.

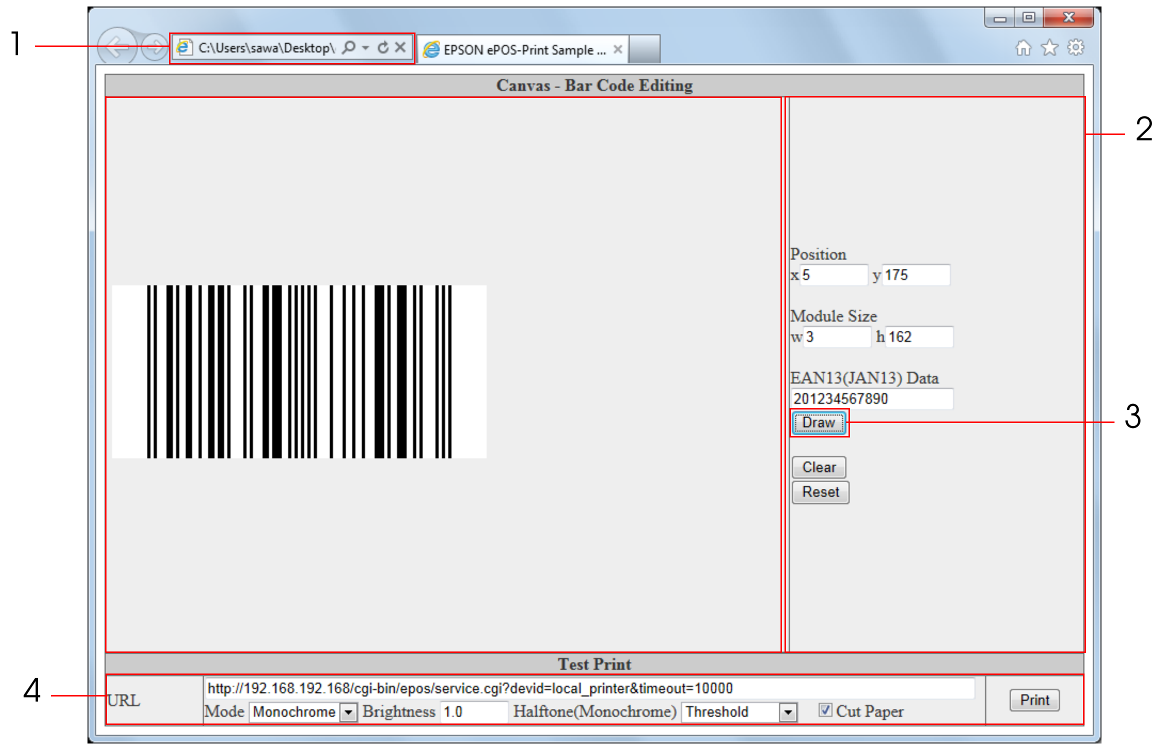
4 Set the following and click the (Print) button.

Item	Description
URL	Enter the following URL: http://(IP address of TM intelligent printer)/cgi-bin/epos/ service.cgi?devid=(device ID of printer to be used for printing)&timeout=(timeout time)
Mode	Set the color mode (Monochrome, Grayscale).
Brightness	Adjust the brightness. (Gamma value in the range 0.1-10.0)
Halftone	Set the halftone processing method for monochrome printing (two-tone).
Cut Paper	When this item is selected, feed cut is performed after printing.

5 The print result is displayed.

Rendering Barcode (canvas-print-barcode.html)

Draw a barcode in HTML5 Canvas and perform a test print.
In the following example, an EAN13, JAN13 or UPC-A is drawn.



1 Open the following URL page using the Web browser.
http://(Web server IP address)/sample/canvas-print-barcode.html

2 "EPSON ePOS-Print Sample Program" appears.
Set items on the right of the page. The following items can be set:

Item	Description
Position	Specify the rendering coordinates.
Module Size	Specify the width and height of the bars.
Data	Specify EAN13 (JAN13) data. For 12-digit numerical data, calculate and add the check digit. For 13-digit numerical data, verify the check digit. For UPC-A data, add 0 at the start of the string to make it 12-or 13-digit data.
Clear	Clears the image drawn in the Canvas.
Reset	Clears the image drawn in the Canvas. In addition, the settings are reset to their default values.

3 Click the (Draw) button.
The image is drawn on Canvas on the left of the page according to the settings made on the right of the page.

4 Set the following and click the (Print) button.

Item	Description
URL	Enter the following URL: http://(IP address of TM intelligent printer)/cgi-bin/epos/ service.cgi?devid=(device ID of printer to be used for printing)&timeout=(timeout time)
Mode	Set the color mode (Monochrome, Grayscale).
Brightness	Adjust the brightness. (Gamma value in the range 0.1-10.0)
Halftone	Set the halftone processing method for monochrome printing (two-tone).
Cut Paper	When this item is selected, feed cut is performed after printing.

5 The print result is displayed.