EPSON
EXCEED YOUR VISION

ePOS-Print SDK for iOS User's Manual

M00049108
Rev.I

Cautions

Trademarks

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 aims to provide development engineers with all the information necessary for the construction and design of a printing system that uses ePOS-Print SDK, and for the development and design of printer applications.

Manual Content

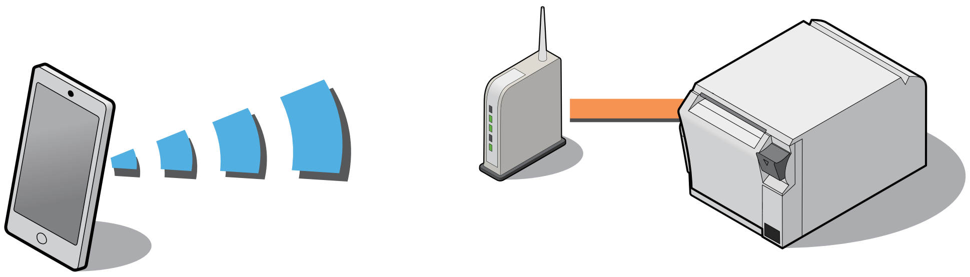
The manual is made up of the following sections:

Contents

Overview

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

Overview of ePOS-Print SDK



The ePOS-Print SDK for iOS is an SDK aimed at development engineers who are developing iOS applications for printing on an EPSON TM printer. Applications are developed using the APIs provided by ePOS-Print SDK.

The ePOS-Print SDK also has the "ePOS-Print SDK for Android" for Android applications.

APIs for transmitting/receiving commands to/from TM printers are also provided.
A command transmission/reception API cannot be used with the ePOS-Print API, EposPrint class.
For details on the command transmission/reception APIs, refer to Command Transmission/ Reception (p.145).

Features

- ❏ Allows printing to TM printers from iOS applications.
- ❏ Allows acquisition of TM printer status from iOS applications.

Function

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 (p.155).)
- ❏ 2D-code printing
  (For 2D-code that can be printed by each model, refer to Printer specifications (p.155).)
- ❏ Drawer kick function
- ❏ Buzzer function
- ❏ Paper layout setting
- ❏ Label / black mark paper feed setting
- ❏ ESC/POS command transmission
- ❏ Acquisition of response from printer (printing result / printer status / battery status)
- ❏ Compatible with Asian languages (simplified Chinese, traditional Chinese, Korean, Thai, Vietnamese)

Printer Search API

- ❏ Search for printers

Log Setting API

- ❏ Log output setting
  (This API allows to output log data to an iOS device's storage and a server that can establish TCP connection.)

Log data output to an iOS device can be saved on other computers using a USB connection.

Operating Environment

iOS Version

- ❏ iOS Ver.4.2 to 4.3.5
- ❏ iOS Ver.5.0 to 5.1.1
- ❏ iOS Ver.6.0 to 6.1.4
- ❏ iOS Ver.7.0 to 7.0.3

For the latest version, refer to the README file.

iOS Device

- ❏ iPhone (3G/3GS/4/4S/5/5s/5c)
- ❏ iPod touch (2nd generation/3rd generation/4th generation/5th generation)
- ❏ iPad/iPad 2/iPad(3rd generation)/iPad(4th generation)/iPad Mini

Printer

TM Printer	Interface	
	Wired LAN	Wi-Fi
TM-T88V	✔	✔
TM-T20	✔	-
TM-T20II	✔	-
TM-T70	✔	✔
TM-T70II	✔	✔
TM-T82	✔	-
TM-T82II	✔	-
TM-T81II	✔	-
TM-T90II	✔	✔
TM-P60(Receipt) Wi-Fi	-	✔
TM-P60(Peeler) Wi-Fi	-	✔
TM-P60II(Receipt) Wi-Fi	-	✔
TM-P60II(Peeler) Wi-Fi	-	✔
TM-P80	-	✔
TM-U220 Series	✔	✔

Development Environment

The following are necessary to develop an iOS application.

❏ Xcode Ver.4.2 to 4.6

❏ Xcode Ver.5.0

Contents in the Package

Package

File	Description
ePOS-Print.h	Header file that includes class definitions and error value / device type constant definitions.
libeposprint.a	Library for function execution. (armv6, armv7, armv7s, arm64, i386 supported)
ePOS-Print_Sample_iOS.zip	A sample program file.
README.en.txt	A readme file.
README.jp.txt	A readme file. (The Japanese-language edition)
EULA.en.txt	Contains the SOFTWARE LICENSE AGREEMENT.
EULA.jp.txt	Contains the SOFTWARE LICENSE AGREEMENT. (The Japanese-language edition)
ePOS-Print_SDK_iOS_E_Revx.pdf	This manual.
ePOS-Print_SDK_iOS_J_Revx.pdf	The Japanese-language edition of this manual.

Manual

The following manuals are available for ePOS-Print SDK for iOS.

❏ ePOS-Print SDK for iOS User’s Manual (This Document)

❏ ePOS-Print SDK for iOS Application Development - Setup Guide

Sample Program

For an iOS application for TM printers developed using ePOS Print SDK, the following program is available.

❏ ePOS-Print_Sample_iOS.zip

Download

For customers in North America, go to the following web site:

http://www.epsonexpert.com/

For customers in other countries, go to the following web site:

https://download.epson-biz.com/?service=pos

Restrictions

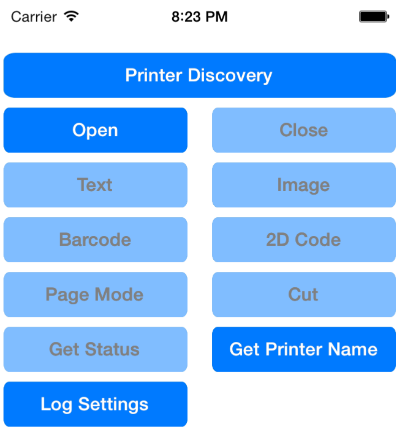
- ❏ A communication API (p.43) and command transmission/reception API (p.145) in the ePOS Print APIs cannot be used for the same device at the same time.
- ❏ A maximum of 16 device ports can be opened in the same application at the same time.
- ❏ More than one port cannot be opened for the same device at the same time.

Sample Program

This chapter describes how to use the sample program(ePOS-Print Sample Program for iOS).

- The sample program is provided for iOS application development engineers as an implementation sample of an iOS application that used ePOS-Print for iOS API.
- The sample program package is provided as an iOS application project for Xcode including Objective-C source files.

Functionality



The Sample Program has the following functionality.

❑ Searching for printers

❑ Opening of port

❑ Closing of port

❑ Text printing

❑ Graphic printing (image file printing)

❑ Barcode printing

❑ 2D-code printing

❑ Printing in page mode

❑ Paper cutting

❑ Printer status acquisition

❑ Acquisition of printer model name/language information

❑ Log output setting

❑ Display of status event

❑ Display of battery status event

The sample program does not contain a functionality for turning text / images / barcodes / etc.

Usage Environment

Usage Environment

- Xcode Ver.4.2 to 4.6
- Xcode Ver.5.0

Printer

- TM printer supported in ePOS-Print SDK.

For details about ways to construct a development environment, please refer to the "ePOS-Print SDK for iOS Application Development - Setup Guide".

Target device

- Device connected to a computer via USB

Environmental Construction

Follow the procedures below to use the sample program.

1. Extract the sample program zip file to a directory of your choosing.
2. Go to the directory you extracted the files to and double click on "ePOS-Print-Sample.xcodeproj".
3. Xcode will start up. Select your target device as the "Scheme."
4. Click the (Run) button on the upper left.
5. The sample program will be installed to the target iOS device, and then the program will start up.

How to Use the Program Sample

This section describes how to use the program sample for the following operations:

- Search for printers and printing (p.18)
- Acquisition of Printer Model Name (p.25)

Search for printers and printing

Use the sample program as follows:

1. Start the sample program. For details, refer to Environmental Construction (p.17).
2. Search for printers. Tap (Printer Discovery) on the main screen.
   The IP addresses of searched printers are displayed in a list.
3. Select the printer for use from the list of IP addresses displayed in procedure 2.
4. Open the printer's port. Tap (Open) on the main screen.
   The "IP Adress" of the printer selected in procedure 3 are displayed.
   Select (Printer Name) and (Language).
5. Set (Status Monitor).

Item	Description
Enabled	• ON: The status monitor is enabled and the printer status is monitored. • OFF: The status monitor is disabled.
Interval	When Enabled is turned ON, the status monitoring interval is set in units of milliseconds.

6. Tap (Open).

7 Execute the following processes:

Process	Description
Text printing	Tap (Text) on the main screen. For details, refer to Text printing (p.20).
Graphic printing	Tap (Image) on the main screen. For details, refer to Graphic printing (p.20).
Barcode printing	Tap (Barcode) on the main screen. For details, refer to Barcode printing (p.21).
2D-code printing	Tap (2D Code) on the main screen. For details, refer to 2D-code printing (p.21).
Printing in page mode	Tap (Page Mode) on the main screen. For details, refer to Printing in page mode (p.22).
Paper cutting	Tap (Cut) on the main screen. For details, refer to Paper cutting (p.22).
Printer status acquisition	Tap (Get Status) on the main screen.
Log output setting	Tap (Log Settings) on the main screen. For details, refer to Log output setting (p.22).

8 The following execution results will be displayed:

- Process execution result (error status / printer status / battery status)
  For details, refer to Process execution result (p.23).
- Method (API) execution error
  For details, refer to Method (API) execution error (p.24).

9 When all processing is finished, tap (Close) on the main screen, and close the printer's port.

Text printing

Execute the text printing according to the following procedure:

1. Enter a string to print for (Print Characters).

2. Specifies the character properties for the string to print. The following properties can be specified:

Property	Description
Font	Set the character font.
Align	Set the alignment.
Line Spacing	Set the line feed space.
Language	Set the language.
Size	Set the character scales (vertical / horizontal).
Style	Set the character style (bold / underlining).
X Position	Set the horizontal start position.
Feed Unit	Set the paper feed amount.

3. Tap (Print) to print.

Graphic printing

Execute the graphic printing according to the following procedure:

1. Tap (Select Image) to select an image file to print.

2. Tap (Color Mode) to select the tone.

   [Gray 16] (multiple tone printing) can only be selected on the TM-T88V/ TM-T70II/ TM-T90II model.

3. Tap (Halftone Method) to select the halftone treatment method.

4. Tap (Brightness) and input a value to specify brightness.

5. Tap (Print) to print.

Barcode printing

Execute the barcode printing according to the following procedure:

1 Set the following for barcodes:

Setting	Description
Type	Select the barcode type.
Data	Enter the barcode data.
HRI	Set the HRI position.
Font	Set the HRI font.
Module Size(Width, Height)	Set the barcode module size (width / height).

2 Tap (Print) to print.

2D-code printing

Execute the 2D-code printing according to the following procedure:

1 Select the 2D-code type using (Type).

2 Enter the 2D-code data for (Data).

3 Set the following for each 2D-code:

Setting	Description
Error Correction Level (PDF417,QR Code, Aztec Code, DataMatrix)	Set the error correction level.
Module Size(Width, Height)	Set the 2D-code module size (width / height)
Max Size	Set the maximum 2D-code size.

4 Tap (Print) to print.

Printing in page mode

Execute the printing in page mode according to the following procedure:

1. Enter a string to print for (Print Characters).
2. Set the print area using (Print Area).

Setting	Description
X	Set the origin of horizontal axis.
Y	Set the origin of vertical axis.
Width	Set the width for the print area.
Height	Set the height for the print area.

3. Tap (Print) to print.

Paper cutting

Execute the paper cutting according to the following procedure:

1. Set whether to cut after feeding paper using (Type).
2. Tap (Print) and execute cutting operation.

Log output setting

Use the following procedures:

1. Set whether to enable the log output function and the log output destination in (Enabled).
2. Set the following items according to the log output destination.

Setting	Description
IP Address	Specify the IP address for TCP communication.
Port	Specify the port number for TCP communication.
Log Size	Specify the maximum size of log data that can be saved on the device's storage.
Log Level	Set the level of log data to be output.

3. Set the method of saving the settings in (Save Settings Permanently).
4. Tap (Setting) to enable the log output settings.

Execution result

Process execution result

Any of the following will be displayed:

- Result: Any of the following statuses will be displayed:

String displayed	Description
SUCCESS	Succeeded
ERR_PARAM	An invalid parameter was passed.
ERR_ILLEGAL	Used in an illegal manner.
ERR_PROCESSING	Failed to execute the process.
ERR_TIMEOUT	The process was timed out.
ERR_CONNECT	Failed to connect to the device.
ERR_MEMORY	Could not secure the memory required for the process.
ERR_OFF_LINE	Offline.
ERR_FAILURE	Another error occurred.

- Status: Any of the following printer statuses will be displayed:

String displayed	Description
NO_RESPONSE	No response from the printer
PRINT_SUCCESS	Printing is successfully completed
DRAWER_KICK	Status of the 3rd pin of the drawer kick-out connector = "H" (Other than TM-P60, TM-P60II, TM-P80)
BATTERY_OFFLINE	Battery offline (TM-P60, TM-P60II, TM-P80)
OFF_LINE	Offline
COVER_OPEN	The cover is open
PAPER_FEED	Paper is being fed by a paper feed switch operation
WAIT_ON_LINE	Waiting to be brought back online
PANEL_SWITCH	The paper feed switch is being pressed (ON)
MECHANICAL_ERR	A mechanical error occurred
AUTOCUTTER_ERR	An autocutter error occurred
UNRECOVER_ERR	An unrecoverable error occurred
AUTORECOVER_ERR	An automatically recoverable error occurred
RECEIPT_NEAR_END	No paper in roll paper near end sensor
RECEIPT_END	No paper in roll paper end sensor

- Battery Status: The following will be displayed.

String displayed	Description
0xnnnn	Battery status value For details, refer to Battery Status (p.39).

Method (API) execution error

Any of the following will be displayed:

- Error Code: Any of the following statuses will be displayed:

String displayed	Description
ERR_PARAM	An invalid parameter was passed.
ERR_OPEN	The open process failed.
ERR_CONNECT	Failed to connect to the device.
ERR_TIMEOUT	All data couldn't be sent during the specified time.
ERR_MEMORY	Could not secure the memory required for the process.
ERR_ILLEGAL	Used in an illegal manner.
ERR_PROCESSING	Failed to execute the process.
ERR_UNSUPPORTED	An unsupported model or language of use has been specified.
ERR_OFF_LINE	Printer is offline.
ERR_FAILURE	Another error occurred.

- Method: The API in which a method execution error occurred is displayed.

Acquisition of Printer Model Name

A command transmission/reception API is used for acquisition of printer model name. For details, refer to Command Transmission/Reception (p.145).

Use the following procedure:

1. Start the sample program. For details, refer to Environmental Construction (p.17).
2. Search for printers. Tap (Printer Discovery) on the main screen.
   The IP addresses of searched printers are displayed in a list.
3. Select the printer for use from the list of IP addresses displayed in procedure 2.
4. Tap (Get Printer Name) on the main screen.
5. Tap (Get Printer Name).
6. The following will be displayed.

Content displayed	Description
Printer Name	Displays the model name of the printer.
Language	Displays the language specifications of the printer.

Programming Guide

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

For ways to construct a development environment for iOS applications that use ePOS-Print SDK for iOS, please refer to the "ePOS-Print SDK for iOS Application Development - Setup Guide".

How to Incorporate the ePOS-Print SDK for iOS

This section explains how to incorporate the ePOS-Print SDK for iOS.
Incorporate the SDK using following procedures.

1. Create a new project in Xcode.
2. Drag the provided Objective-C header (ePOS-Print.h) and drop it anywhere you like in the target project's hierarchy in Xcode's (Project Navigator).
3. Drag the provided static library (libeposprint.a) and drop it anywhere you like in the target project's hierarchy in Xcode's (Project Navigator).
4. Write the Objective-C header import declaration in the *.m source file(s) of the application you would like to use this SDK in as follows:

```
#import "ePOS-Print.h"
```

Print Mode

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

Standard mode

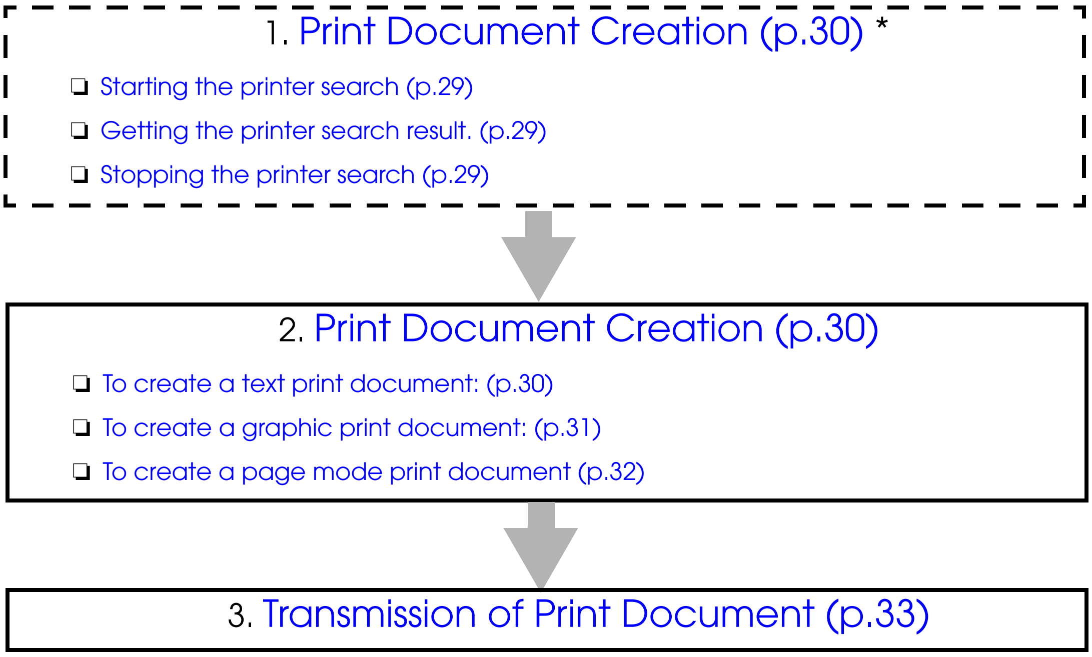
In standard mode, characters are printed line by line. The line feed space is adjusted based on the font size and the height of images, barcodes, etc. This mode is suitable for the type of printing such as printing receipts that requires the paper length to change according to the print space.

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

Perform programming following this flow.



1. Print Document Creation (p.30) *

❑ Starting the printer search (p.29)

❑ Getting the printer search result. (p.29)

❑ Stopping the printer search (p.29)

2. Print Document Creation (p.30)

❑ To create a text print document: (p.30)

❑ To create a graphic print document: (p.31)

❑ To create a page mode print document (p.32)

3. Transmission of Print Document (p.33)

*This is optional.

To ensure successful print operation, write a program in such a way that data is sent after checking the printer status. For the above procedure, refer to Printing After Checking the Printer Status (p.34).

Printer search

Starting the printer search

Use the EpsonIoFinder class's start (p.138) to start searching for printers. Please refer to the following code.

```
//Start Search
int errStatus =
 [EpsonIoFinder start:EPSONIO_OC_DEVTYPE_TCP FindOption:@"255.255.255.255"];
```

Getting the printer search result.

Use the EpsonIoFinder class's getResult (p.140) to get the result of the printer search.
Please refer to the following code.

```
int errStatus = EPSONIO_OC_SUCCESS;

//Get device list
NSArray *array = [EpsonIoFinder getResult:&errStatus];
```

Since the printer search takes time to complete, you might not receive any search results if you call the EpsonIoFinder class's getResult immediately after you call start.

Stopping the printer search

Use the EpsonIoFinder class's stop (p.139) to stop searching for printers. Please refer to the following code.

```
//Stop search
int errStatus = [EpsonIoFinder stop];
```

Print Document Creation

Create a print document using the EposBuilder class (p.41).
Create an EposBuilder class using the constructor for it and create a print document using APIs of the EposBuilder class. Use the programming example below for your reference.

```
//Initialize an EposBuilder class instance
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
     //Create a print document
     int errorStatus = EPOS_OC_SUCCESS;
     errorStatus = [builder addTextLang: EPOS_OC_LANG_EN];
     errorStatus = [builder addTextSmooth: EPOS_OC_TRUE];
     errorStatus = [builder addTextFont: EPOS_OC_FONT_A];
     errorStatus = [builder addTextSize: 3 Height: 3];
     errorStatus = [builder addText: @"Hello,\t"];
     errorStatus = [builder addText: @"World!\n"];
     errorStatus = [builder addCut: EPOS_OC_CUT_FEED];
     [builder release];
}
```

To create a text print document:

To create a text print document, using APIs for text, store the font settings in command buffers to create a print document. Use the programming example below for your reference.

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

```
//Initialize an EposBuilder class instance
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
     //Create a print document
     int errorStatus = EPOS_OC_SUCCESS;
     //<Configure the print character settings>
     errorStatus = [builder addTextLang: EPOS_OC_LANG_EN];
     errorStatus = [builder addTextSmooth: EPOS_OC_TRUE];
     errorStatus = [builder addTextFont: EPOS_OC_FONT_A];
     errorStatus = [builder addTextSize: 4 Height: 4];
     errorStatus = [builder addTextStyle: EPOS_OC_FALSE Ul: EPOS_OC_FALSE
                     Em: EPOS_OC_TRUE Color: EPOS_OC_PARAM_UNSPECIFIED];
     //<Specify the print data>
     errorStatus = [builder addText: @"Hello,\t"];
     errorStatus = [builder addText: @"World!\n"];
     errorStatus = [builder addCut: EPOS_OC_CUT_FEED];
}
```

To create a graphic print document:

To create a graphic print document, for graphics, store the UIImage class in the command buffers with addImage (p.64) of the EposBuilder class. Use the programming example below for your reference.

```
UIImage * imageData = [UIImage imageNamed:@"Sample.png"];

//Initialize an EposBuilder class instance
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
     //Create a print document
     int errorStatus = EPOS_OC_SUCCESS;
     errorStatus = [builder addImage: imageData X: 0 Y: 0 Width: 256 Height: 256
                   Color: EPOS_OC_PARAM_DEFAULT];
     errorStatus = [builder addCut: EPOS_OC_CUT_FEED];
}
```

For ways of graphic printing, you can also print the graphics registered in the printer's NV memory. For details, please refer to addLogo (p.66).

To create a page mode print document

The page mode starts by storing addPageBegin (p.79) of the EposBuilder class into a command buffer. Store the print area (addPageArea (p.81)) and the print start position (addPagePosition (p.85)) in command buffers. Specify the print start position according to the print data. Then, store APIs in command buffers and create print data. For the page mode end, store addPageEnd (p.80) in a command buffer. Use the programming example below for your reference.

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

- Page mode print area (in dots):
  Origin of horizontal axis: 100, origin of vertical axis: 50, width: 200, height: 100
- Page mode print positions (in dots):
  Horizontal print position: 0, vertical print position: 42
- Font: FontA
- Scale: x 2 (horizontal) and x 2 (vertical)
- Style: Bold

```
//Initialize an EposBuilder class instance
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
   Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    //Create a print document
    //<The page mode starts>
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addPageBegin];
    errorStatus = [builder addPageArea: 100 Y: 50 Width: 200 Height: 100];
    errorStatus = [builder addPagePosition: 0 Y: 42];
    //<Configure the print character settings>
    errorStatus = [builder addTextLang: EPOS_OC_LANG_EN];
    errorStatus = [builder addTextSmooth: EPOS_OC_TRUE];
    errorStatus = [builder addTextFont: EPOS_OC_FONT_A];
    errorStatus = [builder addTextSize: 2 Height: 2];
    errorStatus = [builder addTextStyle: EPOS_OC_FALSE Ul: EPOS_OC_FALSE
                   Em: EPOS_OC_TRUE Color: EPOS_OC_PARAM_UNSPECIFIED];
    //<Specify the print data>
    errorStatus = [builder addText: @"Hello,\t"];
    errorStatus = [builder addText: @"World!\n"];
    //<The page mode ends>
    errorStatus = [builder addPageEnd];
    errorStatus = [builder addCut: EPOS_OC_CUT_FEED];
}
```

Transmission of Print Document

Send a print document using the EposPrint class (p.43). Create an EposPrint class using the constructor for it, use sendData to specify the EposBuilder class instance that stores the command buffers for the print document, and send the document. Use the programming example below for your reference.

```
//Initialize an EposBuilder class instance
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
   Lang: EPOS_OC_MODEL_ANK];
if (builder != nil) {
    int errorStatus = EPOS_OC_SUCCESS;
    //Create a print document
    //<The page mode starts>
    errorStatus = [builder addTextLang: EPOS_OC_LANG_EN];
    errorStatus = [builder addTextSmooth: EPOS_OC_TRUE];
    errorStatus = [builder addTextFont: EPOS_OC_FONT_A];
    errorStatus = [builder addTextSize: 4 Height: 4];
    errorStatus = [builder addTextStyle: EPOS_OC_FALSE Ul: EPOS_OC_FALSE
                   Em: EPOS_OC_TRUE Color: EPOS_OC_PARAM_UNSPECIFIED];
    //<Specify the print data>
    errorStatus = [builder addText: @"Hello,\t"];
    errorStatus = [builder addText: @"World!\n"];
    errorStatus = [builder addCut: EPOS_OC_CUT_FEED];
    //Initialize an EposPrint class instance
    id printer = [[EposPrint alloc] init];
    long status;
    //Send a print document
    if (printer != nil) {
        //<Start communication with the printer>
        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                         DeviceName:@"192.168.192.168"];
        //<Send data>
        errorStatus = [printer sendData:builder Timeout:10000 Status:&status];
        //<End communication with the printer>
        errorStatus = [printer closePrinter];
        [printer release];
    }
    [builder release];
}
```

Printing After Checking the Printer Status

To ensure successful print operation, print after checking the printer status.
If the empty print data is transmitted and the printer is online, it will be printed.

Use the programming example below for your reference.

```
//Initialize an EposBuilder class instance
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_ANK];
if (builder != nil) {
    int errorStatus = EPOS_OC_SUCCESS;
    //Create a print document
    errorStatus = [builder addText: @"Hello,\t"];                                  (1)
    errorStatus = [builder addText: @"World!\n"];
    errorStatus = [builder addCut: EPOS_OC_CUT_FEED];

    //Initialize an EposBuilder class instance for confirmation                    (2)
    id conBuilder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
                                           Lang: EPOS_OC_MODEL_ANK];

    //Initialize an EposPrint class instance
    id printer = [[EposPrint alloc] init];
    long status;
    if (printer != nil) {
        //<Start communication with the printer>
        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                        DeviceName:@"192.168.192.168"];

        //<Send data for confirmation>                                             (3)
        errorStatus = [printer sendData:conBuilder Timeout:10000 Status:&status];

        if (errorStatus = EPOS_OC_SUCCESS && (status & EPOS_OC_ST_OFF_LINE )
             != EPOS_OC_ST_OFF_LINE ) {
            //<Send print data>                                                    (4)
            errorStatus = [printer sendData:builder Timeout:10000 Status:&status];
        }

        //<End communication with the printer>
        errorStatus = [printer closePrinter];
        [printer release];
    }
    [conBuilder release];
    [builder release];
}
```

1 Create print data.

2 To check the printer status, send empty print data.

3 Send the print data created in (2).

4 If the print data created in (2) was properly sent, and the printer is online, then send the print data created in (1).

Automatic Acquisition of Printer Status

In the ePOS-Print SDK, the printer status can be automatically notified to the application by means of callback. Refer to the following.

```
//Implementation of callback method for giving notification of printer status     (1)/(4)
- (void)onStatusChange:(NSString *)deviceName Status:(NSNumber *)status
{
  ///Process///
}

- (void)openPrinter
{
  id printer = [[EposPrint alloc] init];

  if ( printer != nil) {
    int errorStatus = EPOS_OC_SUCCESS;

    //Registration of the printer status notification destination callback method   (2)
    [printer setStatusChangeEventCallback @selector(onStatusChange:Status:)
                                          Target:self];

    //Start communications with the printer and monitoring of the printer status    (3)
     errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                            Name:@"192.168.192.168"
                            Enabled: EPOS_OC_TRUE
                            Interval:EPOS_OC_PARAM_DEFAULT];

    ///Process///

  }
}
```

1 Implement the notification destination callback method when events occur.

> In the above description, the callback method, which notifies the printer status at the intervals specified in openPrinter(For acquiring printer status) (p.102), is defined. ePOS-Print has callback method according to each printer status, for example, events such as cover open and drawer open. Use these according to the desired purpose of use. See the Event List (p.36) for the callback method that can be used with ePOS-Print.

2 Register the printer status notification destination.

3 Use openPrinter(For acquiring printer status) (p.102) to start monitoring of the printer status.

4 Notify the printer status to the event implemented in (1).

> When printer status notification is ended, it ends on the closePrinter (p.106) of the EposPrint class.

Event List

For details on the callback method, refer to API Reference (p.41), which explains the callback method registration API.

Event	Callback method registration API
Printer status notification	setStatusChangeEventCallback (p.111)
Online notification	setOnlineEventCallback (p.113)
Offline notification	setOfflineEventCallback (p.115)
Power off notification	setPowerOffEventCallback (p.116)
Cover close notification	setCoverOkEventCallback (p.118)
Cover open notification	setCoverOpenEventCallback (p.120)
Paper OK notification	setPaperOkEventCallback (p.122)
Paper near end notification	setPaperNearEndEventCallback (p.124)
Paper end notification	setPaperEndEventCallback (p.126)
Drawer close notification	setDrawerClosedEventCallback (p.128)
Drawer open notification	setDrawerOpenEventCallback (p.130)
Battery low notification	setBatteryLowEventCallback (p.132)
Battery OK notification	setBatteryOkEventCallback (p.134)
Battery status notification	setBatteryStatusChangeEventCallback (p.136)

Status

The following statuses are defined in ePOS-Print SDK for iOS.

Type	Description
Error status	These are the return values when the API for each class is executed. For details, refer to Error Status List (p.37).
Printer status	Status of the printer when print data was sent. The printer status can be acquired only when sendData (p.107) is executed. For details, refer to Printer Status List (p.38).
Battery status	Status of the printer's remaining battery power. For details, refer to Battery Status (p.39).

Error Status List

Error statuses are defined in each API-executing class.

Error status	Cause
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed. <Example> • An invalid parameter such as null was passed. • A value outside the supported range was specified.
EPOS_OC_ERR_OPEN	Open processing failed. <Example> Could not connect to the designated printer.
EPOS_OC_ERR_CONNECT	Failed to connect to device. <Example> Failed to send the data to the printer.
EPOS_OC_ERR_TIMEOUT	The specified timeout time was exceeded. <Example> Could not send all the data within the specified time.
EPOS_OC_ERR_MEMORY	Could not allocate the necessary memory for processing.
EPOS_OC_ERR_ILLEGAL	Illegal method used. <Example> When the printer was not opened, an API for sending a command to the printer was called.
EPOS_OC_ERR_PROCESSING	Could not execute process. <Example> Could not execute the process because an identical process is being executed in another thread.
EPOS_OC_ERR_UNSUPPORTED	An unsupported model name or language specification was specified.
EPOS_OC_ERR_OFF_LINE	The printer is offline.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Printer Status List

Printer Status	Cause
EPOS_OC_ST_NO_RESPONSE (0x00000001)	No response from the printer
EPOS_OC_ST_PRINT_SUCCESS (0x00000002)	Printing is successfully completed
<Other than TM-P60, TM-P60II, TM-P80> EPOS_OC_ST_DRAWER_KICK (0x00000004)	Status of the 3rd pin of the drawer kick-out connector = "H"
<TM-P60, TM-P60II, TM-P80> EPOS_OC_ST_BATTERY_OFFLINE (0x00000004)	Battery offline status
EPOS_OC_ST_OFF_LINE 0x00000008)	Offline
EPOS_OC_ST_COVER_OPEN (0x00000020)	The cover is open
EPOS_OC_ST_PAPER_FEED (0x00000040)	Paper is being fed by a paper feed switch operation
EPOS_OC_ST_WAIT_ON_LINE (0x00000100)	Waiting to be brought back online
EPOS_OC_ST_PANEL_SWITCH (0x00000200)	The paper feed switch is being pressed (ON)
EPOS_OC_ST_MECHANICAL_ERR (0x00000400)	A mechanical error occurred
EPOS_OC_ST_AUTOCUTTER_ERR (0x00000800)	An autocutter error occurred
EPOS_OC_ST_UNRECOVER_ERR (0x00002000)	An unrecoverable error occurred
EPOS_OC_ST_AUTORECOVER_ERR (0x00004000)	An automatically recoverable error occurred
EPOS_OC_ST_RECEIPT_NEAR_END (0x00020000)	No paper in roll paper near end sensor
EPOS_OC_ST_RECEIPT_END (0x00080000)	No paper in roll paper end sensor
EPOS_OC_ST_BUZZER (0x01000000)	• A buzzer is on (only for applicable devices) • Waiting for label to be removed (only for applicable devices)

Battery Status

The battery status consists of the following 16 bits (0x0000).

Bit	Description
Upper 8 bits	Common battery status For details, refer to Common battery status (upper 8 bits) (p.39) .
Lower 8 bits	Battery status exclusive by model For details, refer to Printer specifications (p.155).

"0x0000" is returned if the battery status cannot be acquired or if the model does not support the battery status.

Common battery status (upper 8 bits)

Battery Status	Cause
0x30	The AC adapter is connected
0x31	The AC adapter is not connected

API Reference

This chapter describes the APIs provided in the ePOS-Print SDK for iOS.

ePOS-Print API

The ePOS-Print APIs are APIs for creating and printing print documents. The following classes are available.

❏ EposBuilder class (p. 41)

❏ EposPrint class (p. 43)

The APIs that you can use and the settings that you can designate vary based on the printer. For details, refer to Printer specifications (p.155).

EposBuilder class

This class creates print documents for printer control commands such as character strings to print, graphic printing, and paper cutting. The following APIs are available.

API		Description	Page
initWithPrinterModel		Initialize an EposBuilder class instance.	p. 44
Clearing command buffers	clearCommandBuffer	Clears the command buffers added by APIs.	p. 46
Text	addTextAlign	Adds a tag for the text alignment setting.	p. 47
	addTextLineSpace	Adds a tag for the line feed space setting.	p. 48
	addTextRotate	Adds a tag for the text rotation setting.	p. 49
	addText	Adds a tag for printing text.	p. 50
	addTextLang	Adds a tag for the target language setting.	p. 51
	addTextFont	Adds a tag for the text font setting.	p. 52
	addTextSmooth	Adds a tag for the text smoothing setting.	p. 53
	addTextDouble	Adds a tag for specifying the double-sized text setting.	p. 54
	addTextSize	Adds a tag for the text scale setting.	p. 55
	addTextStyle	Adds a tag for the text style setting.	p. 56
	addTextPosition	Adds a tag for specifying the print position of text.	p. 58
Paper Feed	addFeedUnit	Adds a tag for paper feeding (in dots).	p. 59
	addFeedLine	Adds a tag for paper feeding (in lines).	p. 60
	addPaperFeedPosi-tion	Adds a tag for label / black mark paper feed-ing.	p. 97

API		Description	Page
Graphic	addImage (For multiple tone printing)	Adds multiple tone raster image printing to the command buffer.	p. 61
	addImage	Adds a tag for a raster image to be printed.	p. 64
	addLogo	Adds a tag for an NV logo to be printed.	p. 66
Barcode	addBarcode	Adds a tag for a bar code to be printed.	p. 67
	addSymbol	Adds a tag for a 2D-Code to be printed.	p. 73
Pagemode	addPageBegin	Adds a tag for switching to page mode.	p. 79
	addPageEnd	Adds a tag for finishing page mode.	p. 80
	addPageArea	Adds a tag for specifying the print area in page mode.	p. 81
	addPageDirection	Adds a tag for specifying the print direction in page mode.	p. 83
	addPagePosition	Adds a tag for specifying the print position in page mode.	p. 85
	addPageLine	Adds a tag for drawing a line in page mode.	p. 87
	addPageRectangle	Adds a tag for drawing a rectangle in page.	p. 89
Cut	addCut	Adds a tag for paper cut.	p. 91
Drawer kick-out	addPulse	Adds a tag for the drawer kick-out.	p. 92
Buzzer	addSound (For setting cycle buzzer)	Sets the buzzer sounding cycle and adds it to the command buffer.	p. 93
	addSound	Adds a tag for turning on the buzzer.	p. 95
Paper Layout	addLayout	Adds a tag for paper layout information.	p. 98
Send Command	addCommand	Adds a tag for inserting commands.	p. 100

EposPrint class

Controls the printer by sending a print document created using the EposBuilder class, and monitors the transmission result and the communication status.

API	Description	Page
init	Initialize an EposPrint class instance.	p. 101
openPrinter (For acquiring printer status)	Starts communications with the printer and monitoring of the printer status.	p. 102
openPrinter	Start communication with the printer.	p. 104
closePrinter	End communication with the printer.	p. 106
sendData	Sends a command to the printer.	p. 107
sendData (For acquiring battery status)	Sends a command to the printer.	p. 109
setStatusChangeEventCallback	Registers the printer status callback method.	p. 111
setOnlineEventCallback	Registers the online event callback method.	p. 113
setOfflineEventCallback	Registers the offline event callback method.	p. 115
setPowerOffEventCallback	Registers the power off event callback method.	p. 116
setCoverOkEventCallback	Registers the cover close event callback method.	p. 118
setCoverOpenEventCallback	Registers the cover open event callback method.	p. 120
setPaperOkEventCallback	Registers the paper OK event callback method.	p. 122
setPaperNearEndEventCallback	Registers the paper near end event callback method.	p. 124
setPaperEndEventCallback	Registers the paper end event callback method.	p. 126
setDrawerClosedEventCallback	Registers the drawer close event callback method.	p. 128
setDrawerOpenEventCallback	Registers the drawer open event callback method.	p. 130
setBatteryLowEventCallback	Registers the battery low event notification destination	p. 132
setBatteryOkEventCallback	Registers the battery OK event notification destination	p. 134
setBatteryStatusChangeEventCallback	Registers the battery status callback method.	p. 136

initWithPrinterModel

Initializes an EposBuilder class instance.

Syntax

```
- (id) initWithPrinterModel:(NSString *)printerModel
                            Lang:(int)lang;
```

Parameter

- printerModel : Specifies the model name for the target printer.

Set value	Description
"TM-T88V"	TM-T88V
"TM-T70"	TM-T70
"TM-T70II"	TM-T70II
"TM-U220"	TM-U220
"TM-P60"	TM-P60
"TM-P60II"	TM-P60II
"TM-P80"	TM-P80
"TM-T20"	TM-T20
"TM-T20II"	TM-T20II
"TM-T81II"	TM-T81II
"TM-T82"	TM-T82
"TM-T82II"	TM-T82II
"TM-T90II"	TM-T90II

- lang : Specifies the language specifications for the printer.

Set value	Description
EPOS_OC_MODEL_ANK	ANK model
EPOS_OC_MODEL_JAPANESE	Japanese model
EPOS_OC_MODEL_CHINESE	Simplified Chinese model
EPOS_OC_MODEL_TAIWAN	Traditional Chinese model
EPOS_OC_MODEL_KOREAN	Korean model
EPOS_OC_MODEL_THAI	Thai model
EPOS_OC_MODEL_SOUTHASIA	South Asian model

Return Value

If processing succeeds, the initialized EposBuilder class instance is returned.

If processing failed, "nil" is returned. The following reasons can cause processing to fail.

- An invalid parameter was specified.
- Could not acquire the necessary memory
- An unsupported model name or language specification was specified.

Example

If you are initializing the command buffer for the TM-T88V ANK model:

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    ///Process///
    [builder release];
}
```

clearCommandBuffer

Clears command buffers used by APIs of the EposBuilder class.

Syntax

```
- (int) clearCommandBuffer;
```

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.

Example

If you are clearing the command buffer:

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus;
    ///Process///
    errorStatus = [builder clearCommandBuffer];
    ///Process///
    [builder release];
}
```

addTextAlign

Adds the text alignment setting to the command buffer.

✎	This API setting also applies to barcodes/2D-Code.
⚠	When the page mode is selected for the print mode, use addPagePosition (p.85) instead of this API to set the alignment.

Syntax

```
- (int) addTextAlign:(int)align;
```

Parameter

- align : Specifies the text alignment.

Set value	Description
EPOS_OC_ALIGN_LEFT (default)	Alignment to the left
EPOS_OC_ALIGN_CENTER	Alignment to the center
EPOS_OC_ALIGN_RIGHT	Alignment to the right

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

To set alignment to the center:

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addTextAlign: EPOS_OC_ALIGN_CENTER];
    ///Process///
}
```

addTextLineSpace

Adds the line feed space setting to the command buffer.

Syntax

```
- (int) addTextLineSpace:(long)linespc;
```

Parameter

- linespc : Specifies the line feed space (in dots). Specifies an integer from 0 to 255.

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

To set the line feed space to 30 dots:

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addTextLineSpace: 30];
    ///Process///
}
```

addTextRotate

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 (p.83) instead of this API function.

Syntax

```
- (int) addTextRotate:(int)rotate;
```

Parameter

- rotate : Specifies whether to rotate text.

Set value	Description
EPOS_OC_TRUE	Specifies rotated printing of text.
EPOS_OC_FALSE (default)	Cancels rotated printing of text.

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

To set text rotation:

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addTextRotate: EPOS_OC_TRUE];
    ///Process///
}
```

addText

Adds the printing of text to the command buffer.

⚠ After printing text, to print content other than text, execute line feed or paper feed.

Syntax

```
- (int) addText:(NSString *)data;
```

Parameter

- data : 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 Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

To add character strings:

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addText: @"Hello,\t"];
    errorStatus = [builder addText: @"World!\n"];
    ///Process///
}
```

addTextLang

Adds the language setting to a command buffer. Encodes the string specified by addText (p.50) according to the language information specified by this API.

This API is an API to be called before calling addText (p.50).

Syntax

```
- (int) addTextLang:(int)lang;
```

Parameter

- lang : Specifies the target language.

Set value	Language
EPOS_OC_LANG_EN(default)	English(ANK)
EPOS_OC_LANG_JA	Japanese
EPOS_OC_LANG_ZH_CN	Simplified Chinese
EPOS_OC_LANG_ZH_TW	Traditional Chinese
EPOS_OC_LANG_KO	Korean
EPOS_OC_LANG_TH	Thai (South Asia specifications)
EPOS_OC_LANG_VI	Vietnamese (South Asia specifications)

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

To set the language as English:

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addTextLang: EPOS_OC_LANG_EN];
    ///Process///
}
```

addTextFont

Adds the text font setting to the command buffer.

Syntax

```
- (int) addTextFont:(int)font;
```

Parameter

- font : Specifies the font.

Set value	Language
EPOS_OC_FONT_A (default)	Font A
EPOS_OC_FONT_B	Font B
EPOS_OC_FONT_C	Font C

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

To set the font B:

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addTextFont: EPOS_OC_FONT_B];
    ///Process///
}
```

addTextSmooth

Adds the smoothing setting to the command buffer.

Syntax

```
- (int) addTextSmooth:(int)smooth;
```

Parameter

- smooth : Specifies whether to enable smoothing.

Set value	Description
EPOS_OC_TRUE	Specifies smoothing.
EPOS_OC_FALSE (default)	Cancels smoothing

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

To enable smoothing:

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addTextSmooth: EPOS_OC_TRUE];
    ///Process///
}
```

addTextDouble

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

Syntax

```
- (int) addTextDouble:(int)dw Dh:(int)dh;
```

Parameter

- dw : Specifies the double-sized width.

Set value	Description
EPOS_OC_TRUE	Specifies the double-sized width.
EPOS_OC_FALSE (default)	Cancels the double-sized width
EPOS_OC_PARAM_UNSPECIFIED	Retains the current setting.

- dh : Specifies the double-sized height.

Set value	Description
EPOS_OC_TRUE	Specifies the double-sized height
EPOS_OC_FALSE (default)	Cancels the double-sized height
EPOS_OC_PARAM_UNSPECIFIED	Retains the current setting.

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

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

To set the size as double width and height:

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addTextDouble: EPOS_OC_TRUE
        Dh: EPOS_OC_TRUE];
    ///Process///
}
```

addTextSize

Adds the text scale setting to the command buffer.

Syntax

```
    - (int) addTextSize:(long)width Height:(long)height;
```

Parameter

- width : Specifies the horizontal scale of text.

Set value	Description
Integer from 1 to 8	Horizontal scale (default : 1)
EPOS_OC_PARAM_UNSPECIFIED	Retains the current setting.

- height : Specifies the vertical scale of text.

Set value	Description
Integer from 1 to 8	Vertical scale (default : 1)
EPOS_OC_PARAM_UNSPECIFIED	Retains the current setting.

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

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

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addTextSize: 4 Height: 4];
    ///Process///
}
```

4

addTextStyle

Adds the text style setting to the command buffer.

Syntax

```
- (int) addTextStyle:(int)reverse Ul:(int)ul Em:(int)em
                     Color:(int)color;
```

Parameter

- reverse : Specifies inversion of black and white for text.

Set value	Description
EPOS_OC_TRUE	Specifies the inversion of black and white parts of characters.
EPOS_OC_FALSE (default)	Cancels the inversion of black and white parts of characters.
EPOS_OC_PARAM_UNSPECIFIED	Retains the current setting.

- ul : Specifies the underline style.

Set value	Description
EPOS_OC_TRUE	Specifies underlining.
EPOS_OC_FALSE (default)	Cancels underlining.
EPOS_OC_PARAM_UNSPECIFIED	Retains the current setting.

- em : Specifies the bold style.

Set value	Description
EPOS_OC_TRUE	Specifies emphasized printing of characters.
EPOS_OC_FALSE (default)	Cancels emphasized printing of characters.
EPOS_OC_PARAM_UNSPECIFIED	Retains the current setting.

- color : Specifies the color.

Set value	Description
EPOS_OC_COLOR_NONE	Characters are not printed.
EPOS_OC_COLOR_1 (default)	First color
EPOS_OC_COLOR_2	Second color
EPOS_OC_COLOR_3	Third color
EPOS_OC_COLOR_4	Fourth color
EPOS_OC_PARAM_UNSPECIFIED	Retains the current color setting

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

To set the underline style:

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addTextStyle: EPOS_OC_PARAM_UNSPECIFIED
        Ul: EPOS_OC_TRUE Em: EPOS_OC_PARAM_UNSPECIFIED
        Color: EPOS_OC_PARAM_UNSPECIFIED];
    ///Process///
}
```

addTextPosition

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

Syntax

```
- (int) addTextPosition:(long)x;
```

Parameter

- x :	Specifies the horizontal print start position (in dots).
	Specifies an integer from 0 to 65535.

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

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

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addTextPosition: 120];
    ///Process///
}
```

addFeedUnit

Adds paper feeding in dots to the command buffer.

Syntax

```
- (int) addFeedUnit:(long)unit;
```

Parameter

- unit : Specifies the paper feed space (in dots). Specifies an integer from 0 to 255.

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

To feed paper by 30 dots:

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addFeedUnit: 30];
    ///Process///
}
```

addFeedLine

Adds paper feeding in lines to the command buffer.

Syntax

```
- (int) addFeedLine:(long)line;
```

Parameter

- unit : Specifies the paper feed space (in lines). Specifies an integer from 0 to 255.

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

To feed paper by 3 lines:

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addFeedLine: 3];
    ///Process///
}
```

addImage(For multiple tone printing)

Adds raster image printing to the command buffer. Prints the graphic rendered by the UIImage class.
Out of the UI Image class graphics, the specified scope is converted to raster image data according to this API setting. 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 EPOS_OC_ALIGN_LEFT for the addTextAlign (p.47), and specify a multiple of 8 not exceeding the printer's paper width for the width parameter of this API.

Syntax

```
- (int) addImage:(UIImage *)data X:(long)x Y:(long)y
                 Width:(long)width Height:(long)height
                 Color:(int)color Mode:(int)mode
                 Halftone:(int)halftone
                 Brightness:(double)brightness;
```

Parameter

- data : Specifies an instance of the UIImage class.
- x : Specifies the horizontal start position in the print area.
  Specifies an integer from 0 to 65534.
- y : Specifies the vertical start position in the print area.
  Specifies an integer from 0 to 65534.
- width : Specifies the width of the print area. Specifies an integer from 1 to 65535.
- height : Specifies the height of the print area. Specifies an integer from 1 to 65535.
- color : Specifies the color.

Set value	Description
EPOS_OC_COLOR_NONE	Characters are not printed.
EPOS_OC_COLOR_1	First color
EPOS_OC_COLOR_2	Second color
EPOS_OC_COLOR_3	Third color
EPOS_OC_COLOR_4	Fourth color
EPOS_OC_PARAM_DEFAULT	First color

If the area defined in the x/y parameters and the width/height parameters do not fit in the image size defined by the data parameter, EPOS_OC_ERR_PARAM will be returned for the return value.

- mode : Specify the color mode.

Set value	Description	TM printer-separate setting									
		TM-T88V	TM-T70	TM-T70II	TM-U220	TM-P60/ TM-P60II	TM-P80	TM-T20/ TM-T20II	TM-T81II	TM-T82/ TM-T82II	TM-T90II
EPOS_OC_MODE_MONO	Monochrome (2 tone)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
EPOS_OC_MODE_GRAY16	Multiple tone (16 tone)	✔	-	✔	-	-	-	-	-	-	✔
EPOS_OC_PARAM_DEFAULT	Specify the half tone treatment method. (Monochrome (2 tone))	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔

- halftone : Specify the half tone treatment method.

Set value	Description
EPOS_OC_HALFTONE_DITHER	Dither (This is suitable for graphic printing).
EPOS_OC_HALFTONE_ERROR_DIFFUSION	Error diffusion (This is suitable for mixed printing or characters and graphics).
EPOS_OC_HALFTONE_THRESHOLD	Threshold value (This is suitable for printing of characters).
EPOS_OC_PARAM_DEFAULT	Default value (dither) selection

⚠ In the case of multiple tone (16 tone), this is disregarded.

- brightness : Specify the correction value for brightness.

Set value	Description
Actual figure from 0.1 to 10.0	Brightness correction value (gamma value)
EPOS_OC_PARAM_DEFAULT	Select the default value (1.0)

⚠ If you specify a value other than 1.0, the printing speed will become slower.

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

```
UIImage * imageData = Nil;

id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    ///Process///
    errorStatus = [builder addImage: imageData X: 0 Y: 0 Width: 256
        Height: 256 Color: EPOS_OC_PARAM_DEFAULT Mode: EPOS_OC_MODE_MONO
        Halftone: EPOS_OC_HALFTONE_DITHER Brightness: 1.0];
    ///Process///
}
```

To print an image 256 dots wide and 256 dots high in page mode:

```
UIImage * imageData = Nil;

id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    ///Process///
    errorStatus = [builder addPageBegin];
    errorStatus = [builder addPagePosition: 0 Y: 255];
    errorStatus = [builder addImage: imageData X: 0 Y: 0 Width: 256
        Height: 256 Color: EPOS_OC_PARAM_DEFAULT Mode: EPOS_OC_MODE_MONO
        Halftone: EPOS_OC_HALFTONE_DITHER Brightness: 1.0];
    errorStatus = [builder addPageEnd];
    ///Process///
}
```

addImage

Adds raster image printing to the command buffer. Prints the graphic rendered by the UIImage class.
In the UIImage class graphic, changes the specified range to binary value with dither processing, and converts it to raster image data. 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.

- When printing in multiple tone, use addImage(For multiple tone printing) (p.61).
- To print a raster image at high speed, specify EPOS_OC_ALIGN_LEFT for the addTextAlign (p.47), and specify a multiple of 8 not exceeding the printer's paper width for the width parameter of this API.

Syntax

```
- (int) addImage:(UIImage *)data X:(long)x Y:(long)y
                  Width:(long)width Height:(long)height
                  Color:(int)color;
```

Parameter

- data : Specifies an instance of the UIImage class.
- x : Specifies the horizontal start position in the print area.
  Specifies an integer from 0 to 65534.
- y : Specifies the vertical start position in the print area.
  Specifies an integer from 0 to 65534.
- width : Specifies the width of the print area. Specifies an integer from 1 to 65535.
- height : Specifies the height of the print area. Specifies an integer from 1 to 65535.
- color : Specifies the color.

Set value	Description
EPOS_OC_COLOR_NONE	Characters are not printed.
EPOS_OC_COLOR_1	First color
EPOS_OC_COLOR_2	Second color
EPOS_OC_COLOR_3	Third color
EPOS_OC_COLOR_4	Fourth color
EPOS_OC_PARAM_DEFAULT	First color

If the area defined in the x/y parameters and the width/height parameters do not fit in the image size defined by the data parameter, EPOS_OC_ERR_PARAM will be returned for the return value.

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

```
UIImage * imageData = Nil;

id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    ///Process///
    errorStatus = [builder addImage: imageData X: 0 Y: 0 Width: 256
        Height: 256 Color: EPOS_OC_PARAM_DEFAULT];
    ///Process///
}
```

To print an image 256 dots wide and 256 dots high in page mode:

```
UIImage * imageData = Nil;

id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    ///Process///
    errorStatus = [builder addPageBegin];
    errorStatus = [builder addPagePosition: 0 Y: 255];
    errorStatus = [builder addImage: imageData X: 0 Y: 0 Width: 256
        Height: 256 Color: EPOS_OC_PARAM_DEFAULT];
    errorStatus = [builder addPageEnd];
    ///Process///
}
```

addLogo

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

Register a logo in advance into the printer using the following utilities:
- Model-dedicated Utility
- TM Flash Logo Setup Utility

Syntax

```
- (int) addLogo:(long)key1 Key2:(long)key2;
```

Parameter

• key1 :	Specifies the key code 1 of an NV logo. Specifies an integer from 32 to 126.
• key2 :	Specifies the key code 2 of an NV logo. Specifies an integer from 32 to 126.

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

To print a NV logo with the key code parameters specified as 48, 48:

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addLogo: 48 Key2: 48];
    ///Process///
}
```

addBarcode

Adds barcode printing to the command buffer.

Syntax

```
- (int) addBarcode:(NSString *)data Type:(int)type
                   Hri:(int)hri Font:(int)font
                   Width:(long)width
                   Height:(long)height;
```

Parameter

- data : Specifies the barcode data as a string.

Specify a string that follows the barcode standard specified by the type parameter. If the specified string does not conform to the standard, a barcode will not be printed.

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.
EAN13	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.
JAN13	
EAN8	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.
JAN8	
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.
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.

Barcode type	Description
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 : Specifies the barcode type.

Set value	Barcode type
EPOS_OC_BARCODE_UPC_A	UPC-A
EPOS_OC_BARCODE_UPC_E	UPC-E
EPOS_OC_BARCODE_EAN13	EAN13
EPOS_OC_BARCODE_JAN13	JAN13
EPOS_OC_BARCODE_EAN8	EAN8
EPOS_OC_BARCODE_JAN8	JAN8
EPOS_OC_BARCODE_CODE39	CODE39
EPOS_OC_BARCODE_ITF	ITF
EPOS_OC_BARCODE_CODABAR	CODABAR
EPOS_OC_BARCODE_CODE93	CODE93
EPOS_OC_BARCODE_CODE128	CODE128
EPOS_OC_BARCODE_GS1_128	GS1-128
EPOS_OC_BARCODE_GS1_ DATABAR_OMNIDIRECTIONAL	GS1 DataBar Omnidirectional
EPOS_OC_BARCODE_GS1_DATABAR_TRUNCATED	GS1 DataBar Truncated
EPOS_OC_BARCODE_GS1_DATABAR_LIMITED	GS1 DataBar Limited
EPOS_OC_BARCODE_GS1_DATABAR_EXPANDED	GS1 Databar Expanded

- hri : Specifies the HRI position.

Set value	Description
EPOS_OC_HRI_NONE(default)	HRI not printed
EPOS_OC_HRI_ABOVE	Above the bar code
EPOS_OC_HRI_BELOW	Below the bar code
EPOS_OC_HRI_BOTH	Both above and below the bar code
EPOS_OC_PARAM_UNSPECIFIED	Retains the current setting.

- font : Specifies the HRI font.

Set value	Description
EPOS_OC_FONT_A(default)	Font A
EPOS_OC_FONT_B	Font B
EPOS_OC_FONT_C	Font C
EPOS_OC_PARAM_UNSPECIFIED	Retains the current setting.

- width : Specifies the width of each module in dots. Specifies an integer from 2 to 6.

Set value	Description
Integer from 2 to 6	The width of each module. (Unit:dot)
EPOS_OC_PARAM_UNSPECIFIED	Retains the current setting.

- height : Specifies the barcode height in dots. Specifies an integer from 1 to 255.

Set value	Description
Integer from 1 to 255	The barcode height. (Unit:dot)
EPOS_OC_PARAM_UNSPECIFIED	Retains the current setting.

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

To print barcodes:

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
     Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
     int errorStatus = EPOS_OC_SUCCESS;
     errorStatus = [builder addBarcode: @"01234567890"
         Type: EPOS_OC_BARCODE_UPC_A Hri: EPOS_OC_HRI_BELOW
         Font: EPOS_OC_PARAM_UNSPECIFIED Width: 2 Height: 64];
     errorStatus = [builder addBarcode: @"01234500005"
         Type: EPOS_OC_BARCODE_UPC_E Hri: EPOS_OC_PARAM_UNSPECIFIED
         Font: EPOS_OC_PARAM_UNSPECIFIED Width: EPOS_OC_PARAM_UNSPECIFIED
         Height: EPOS_OC_PARAM_UNSPECIFIED];
     errorStatus = [builder addBarcode: @"201234567890"
         Type: EPOS_OC_BARCODE_EAN13 Hri: EPOS_OC_PARAM_UNSPECIFIED
         Font: EPOS_OC_PARAM_UNSPECIFIED Width: EPOS_OC_PARAM_UNSPECIFIED
         Height: EPOS_OC_PARAM_UNSPECIFIED];
     errorStatus = [builder addBarcode: @"201234567890"
         Type: EPOS_OC_BARCODE_JAN13 Hri: EPOS_OC_PARAM_UNSPECIFIED
         Font: EPOS_OC_PARAM_UNSPECIFIED Width: EPOS_OC_PARAM_UNSPECIFIED
         Height: EPOS_OC_PARAM_UNSPECIFIED];
     errorStatus = [builder addBarcode: @"2012345" Type:
         EPOS_OC_BARCODE_EAN8
         Hri: EPOS_OC_PARAM_UNSPECIFIED Font: EPOS_OC_PARAM_UNSPECIFIED
         Width: EPOS_OC_PARAM_UNSPECIFIED Height:
         EPOS_OC_PARAM_UNSPECIFIED];
     errorStatus = [builder addBarcode: @"2012345" Type:
         EPOS_OC_BARCODE_JAN8
         Hri: EPOS_OC_PARAM_UNSPECIFIED Font: EPOS_OC_PARAM_UNSPECIFIED
         Width: EPOS_OC_PARAM_UNSPECIFIED Height:
         EPOS_OC_PARAM_UNSPECIFIED];
     errorStatus = [builder addBarcode: @"ABCDE" Type:
         EPOS_OC_BARCODE_CODE39
         Hri: EPOS_OC_PARAM_UNSPECIFIED Font: EPOS_OC_PARAM_UNSPECIFIED
         Width: EPOS_OC_PARAM_UNSPECIFIED Height:
         EPOS_OC_PARAM_UNSPECIFIED];
     errorStatus = [builder addBarcode: @"012345" Type: EPOS_OC_BARCODE_ITF
         Hri: EPOS_OC_PARAM_UNSPECIFIED Font: EPOS_OC_PARAM_UNSPECIFIED
         Width: EPOS_OC_PARAM_UNSPECIFIED Height:
         EPOS_OC_PARAM_UNSPECIFIED];
     errorStatus = [builder addBarcode: @"A012345A"
         Type: EPOS_OC_BARCODE_CODABAR Hri: EPOS_OC_PARAM_UNSPECIFIED
         Font: EPOS_OC_PARAM_UNSPECIFIED Width: EPOS_OC_PARAM_UNSPECIFIED
         Height: EPOS_OC_PARAM_UNSPECIFIED];
     errorStatus = [builder addBarcode: @"ABCDE" Type:
         EPOS_OC_BARCODE_CODE93
         Hri: EPOS_OC_PARAM_UNSPECIFIED Font: EPOS_OC_PARAM_UNSPECIFIED
         Width: EPOS_OC_PARAM_UNSPECIFIED Height:
         EPOS_OC_PARAM_UNSPECIFIED];
     errorStatus = [builder addBarcode: @"{Babcde"
         Type: EPOS_OC_BARCODE_CODE128 Hri: EPOS_OC_PARAM_UNSPECIFIED
         Font: EPOS_OC_PARAM_UNSPECIFIED Width: EPOS_OC_PARAM_UNSPECIFIED
         Height: EPOS_OC_PARAM_UNSPECIFIED];
```

```
    errorStatus = [builder addBarcode: @"(01)201234567890*"
        Type: EPOS_OC_BARCODE_GS1_128 Hri: EPOS_OC_PARAM_UNSPECIFIED
        Font: EPOS_OC_PARAM_UNSPECIFIED Width: EPOS_OC_PARAM_UNSPECIFIED
        Height: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addBarcode: @"0201234567890"
        Type: EPOS_OC_BARCODE_GS1_DATABAR_OMNIDIRECTIONAL
        Hri: EPOS_OC_PARAM_UNSPECIFIED Font: EPOS_OC_PARAM_UNSPECIFIED
        Width: EPOS_OC_PARAM_UNSPECIFIED Height:
        EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addBarcode: @"0201234567890"
        Type: EPOS_OC_BARCODE_GS1_DATABAR_TRUNCATED
        Hri: EPOS_OC_PARAM_UNSPECIFIED Font: EPOS_OC_PARAM_UNSPECIFIED
        Width: EPOS_OC_PARAM_UNSPECIFIED Height:
        EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addBarcode: @"0201234567890"
        Type: EPOS_OC_BARCODE_GS1_DATABAR_LIMITED
        Hri: EPOS_OC_PARAM_UNSPECIFIED Font: EPOS_OC_PARAM_UNSPECIFIED
        Width: EPOS_OC_PARAM_UNSPECIFIED Height:
        EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addBarcode: @"(01)2012345678903"
        Type: EPOS_OC_BARCODE_GS1_DATABAR_EXPANDED
        Hri: EPOS_OC_PARAM_UNSPECIFIED Font: EPOS_OC_PARAM_UNSPECIFIED
        Width: EPOS_OC_PARAM_UNSPECIFIED Height:
        EPOS_OC_PARAM_UNSPECIFIED];
    ///Process///
}
```

addSymbol

Adds 2D-Code printing to the command buffer.

Syntax

```
- (int) addSymbol:(NSString *)data Type:(int)type
                  Level:(int)level Width:(long)width
                  Height:(long)height Size:(long)size;
```

Parameter

- data : Specifies 2D-Code data as a character string.

2D-Code type	Description
Standard PDF417	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.
Truncated PDF417	
QR Code Model 1	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
QR Code Model 2	

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 (: {( ): {)
Aztec Code Full-Range mode	After converting the character string to UTF-8, conduct the escape sequence and encode. Up to 3,067 characters of text, 3,832 numerical figures and 1,914 bytes of binary data can be specified.

2D-Code type	Description
Aztec Code Compact mode	After converting the character string to UTF-8, conduct the escape sequence and encode. Up to 89 characters of text, 110 numerical figures and 53 bytes of binary data can be specified.
DataMatrix square	After converting the character string to UTF-8, conduct the escape sequence and encode. The symbol is either a square ranging in size from 10 lines x 10 rows ~ 144 lines ~ 144 rows, or a rectangle comprising 8 lines, 12 lines or 16 lines. Up to 2,335 alphanumerical, 3,116 numerical figures and 1,556 bytes of binary data can be specified.
DataMatrix rectangle, 8 lines	
DataMatrix rectangle, 12 lines	
DataMatrix rectangle, 16 lines	

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

String	Description
\xnn	Control code
\\	Back slash

- type : Specifies the 2D-Code type.

Set value	2D-Code type
EPOS_OC_SYMBOL_PDF417_STANDARD	Standard PDF417
EPOS_OC_SYMBOL_PDF417_TRUNCATED	Truncated PDF417
EPOS_OC_SYMBOL_QRCODE_MODEL_1	QR Code Model 1
EPOS_OC_SYMBOL_QRCODE_MODEL_2	QR Code Model 2
EPOS_OC_SYMBOL_MAXICODE_MODE_2	MaxiCode Mode 2
EPOS_OC_SYMBOL_MAXICODE_MODE_3	MaxiCode Mode 3
EPOS_OC_SYMBOL_MAXICODE_MODE_4	MaxiCode Mode 4
EPOS_OC_SYMBOL_MAXICODE_MODE_5	MaxiCode Mode 5
EPOS_OC_SYMBOL_MAXICODE_MODE_6	MaxiCode Mode 6
EPOS_OC_SYMBOL_GS1_DATABAR_STACKED	GS1 DataBar Stacked
EPOS_OC_SYMBOL_GS1_DATABAR_STACKED_ OMNIDIRECTIONAL	GS1 DataBar Stacked Omnidirectional
EPOS_OC_SYMBOL_GS1_DATABAR_EXPANDED _STACKED	GS1 DataBar Expanded Stacked
EPOS_OC_SYMBOL_AZTECCODE_FULLRANGE	Aztec Code Full-Range mode
EPOS_OC_SYMBOL_AZTECCODE_COMPACT	Aztec Code Compact mode
EPOS_OC_SYMBOL_DATAMATRIX_SQUARE	DataMatrix square
EPOS_OC_SYMBOL_DATAMATRIX_RECTANGLE_ 8	DataMatrix rectangle, 8 lines

Set value	2D-Code type
EPOS_OC_SYMBOL_DATAMATRIX_RECTANGLE_12	DataMatrix rectangle, 12 lines
EPOS_OC_SYMBOL_DATAMATRIX_RECTANGLE_16	DataMatrix rectangle, 16 lines

- level : Specifies the error correction level.

Set value	Description
EPOS_OC_LEVEL_0	PDF417 error correction level 0
EPOS_OC_LEVEL_1	PDF417 error correction level 1
EPOS_OC_LEVEL_2	PDF417 error correction level 2
EPOS_OC_LEVEL_3	PDF417 error correction level 3
EPOS_OC_LEVEL_4	PDF417 error correction level 4
EPOS_OC_LEVEL_5	PDF417 error correction level 5
EPOS_OC_LEVEL_6	PDF417 error correction level 6
EPOS_OC_LEVEL_7	PDF417 error correction level 7
EPOS_OC_LEVEL_8	PDF417 error correction level 8
EPOS_OC_LEVEL_L	QR Code error correction level L
EPOS_OC_LEVEL_M	QR Code error correction level M
EPOS_OC_LEVEL_Q	QR Code error correction level Q
EPOS_OC_LEVEL_H	QR Code error correction level H
Integer from 5 to 95	Aztec Code error correction level (percent unit)
EPOS_OC_LEVEL_DEFAULT	Default level
EPOS_OC_PARAM_UNSPECIFIED	Retains the current setting.

- Select the level according to the 2D-Code type.
- MaxiCode/two-dimensional GS1 DataBar/DataMatrix, select EPOS_OC_LEVEL_DEFAULT.

- width : Specifies the module width.

Set value	Description
Integer from 1 to 255	Module width
EPOS_OC_PARAM_UNSPECIFIED	Retains the current setting.

MaxiCode is ignored.

- height : Specifies the module height.

Set value	Description
Integer from 1 to 255	Module height
EPOS_OC_PARAM_UNSPECIFIED	Retains the current setting.

⚠ QR Code/MaxiCode/two-dimensional GS1 DataBar/Aztec Code/DataMatrix are ignored.

- size : Specifies the 2D-Code maximum size.

Set value	Description
Integer from 0 to 65535	2D-Code maximum size
EPOS_OC_PARAM_UNSPECIFIED	Retains the current setting.

⚠ QR Code/MaxiCode/Aztec Code/DataMatrix are ignored.

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

To print 2D-Code:

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
     Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
     int errorStatus = EPOS_OC_SUCCESS;
     errorStatus = [builder addSymbol: @"ABCDE"
         Type: EPOS_OC_SYMBOL_PDF417_STANDARD Level:
         EPOS_OC_PARAM_UNSPECIFIED Width: EPOS_OC_PARAM_UNSPECIFIED
         Height: EPOS_OC_PARAM_UNSPECIFIED Size: EPOS_OC_PARAM_UNSPECIFIED];
     errorStatus = [builder addSymbol: @"ABCDE"
          Type: EPOS_OC_SYMBOL_QRCODE_MODEL_2 Level: EPOS_OC_LEVEL_Q
          Width: EPOS_OC_PARAM_UNSPECIFIED Height: EPOS_OC_PARAM_UNSPECIFIED
          Size: EPOS_OC_PARAM_UNSPECIFIED];
     errorStatus = [builder addSymbol: @"908063840\\x1d850\\x1d001\\x1d\\x04"
          Type: EPOS_OC_SYMBOL_MAXICODE_MODE_2 Level: EPOS_OC_PARAM_UNSPECIFIED
          Width: EPOS_OC_PARAM_UNSPECIFIED Height: EPOS_OC_PARAM_UNSPECIFIED
          Size: EPOS_OC_PARAM_UNSPECIFIED];
     errorStatus = [builder addSymbol: @"0201234567890"
          Type: EPOS_OC_SYMBOL_GS1_DATABAR_STACKED
          Level: EPOS_OC_PARAM_UNSPECIFIED Width: EPOS_OC_PARAM_UNSPECIFIED
          Height: EPOS_OC_PARAM_UNSPECIFIED Size: EPOS_OC_PARAM_UNSPECIFIED];
     errorStatus = [builder addSymbol: @"0201234567890"
          Type: EPOS_OC_SYMBOL_GS1_DATABAR_STACKED_OMNIDIRECTIONAL
          Level: EPOS_OC_PARAM_UNSPECIFIED Width: EPOS_OC_PARAM_UNSPECIFIED
          Height: EPOS_OC_PARAM_UNSPECIFIED Size: EPOS_OC_PARAM_UNSPECIFIED];
     errorStatus = [builder addSymbol: @"(01)02012345678903"
          Type: EPOS_OC_SYMBOL_GS1_DATABAR_EXPANDED_STACKED
          Level: EPOS_OC_PARAM_UNSPECIFIED Width: EPOS_OC_PARAM_UNSPECIFIED
          Height: EPOS_OC_PARAM_UNSPECIFIED Size: EPOS_OC_PARAM_UNSPECIFIED];
    ///Process///
}
```

addPageBegin

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

Use this API function with addPageEnd (p.80).

Syntax

```
- (int) addPageBegin;
```

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

To print the characters "ABCDE" in page mode:

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
     Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
     int errorStatus = EPOS_OC_SUCCESS;
     errorStatus = [builder addPageBegin];
     errorStatus = [builder addText: @"ABCDE"];
     errorStatus = [builder addPageEnd];
    ///Process///
}
```

addPageEnd

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

Use this API function with addPageBegin (p.79).

Syntax

```
- (int) addPageEnd;
```

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

To print the characters "ABCDE" in page mode:

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
     Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
     int errorStatus = EPOS_OC_SUCCESS;
     errorStatus = [builder addPageBegin];
     errorStatus = [builder addText: @"ABCDE"];
     errorStatus = [builder addPageEnd];
     ///Process///
}
```

addPageArea

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.79) and addPageEnd (p.80).

Syntax

```
- (int) addPageArea:(long)x Y:(long)y Width:(long)width
                            Height:(long)height;
```

Parameter

- x : 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 : 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 : Specifies the width of the print area (in dots). Specifies an integer from 1 to 65535.
- height : Specifies the height of the print area (in dots). Specifies an integer from 1 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

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

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":

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addPageBegin];
    errorStatus = [builder addPageArea: 100 Y: 50 Width: 200 Height: 30];
    errorStatus = [builder addText: @"ABCDE"];
    errorStatus = [builder addPageEnd];
    ///Process///
}
```

addPageDirection

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.79) and addPageEnd (p.80).

Syntax

```
- (int) addPageDirection:(int)dir;
```

Parameter

- dir : Specifies the print direction in page mode.

Set value	Description
EPOS_OC_DIRECTION_LEFT_TO_RIGHT(default)	Left to right (No rotation.Data is printed from the top left corner to the right.)
EPOS_OC_DIRECTION_BOTTOM_TO_TOP	Bottom to top (Counterclockwise rotation by 90 degrees. Data is printed from the bottom left corner to the top.)
EPOS_OC_DIRECTION_RIGHT_TO_LEFT	Right to left (Rotation by 180 degrees.Data is printed from the bottom right corner to the left.)
EPOS_OC_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

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

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

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
     Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addPageBegin];
    errorStatus = [builder addPageArea: 100 Y: 50 Width: 30 Height: 200];
    errorStatus = [builder addPageDirection:
        EPOS_OC_DIRECTION_TOP_TO_BOTTOM];
    errorStatus = [builder addText: @"ABCDE"];
    errorStatus = [builder addPageEnd];
    ///Process///
}
```

addPagePosition

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.79) and addPageEnd (p.80).

Syntax

```
- (int) addPagePosition:(long)x Y:(long)y;
```

Parameter

• x :	Specifies the horizontal print position (in dots). Specifies an integer from 0 to 65535.
• y :	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 2D-Code:
  Specify the top left of the symbol. This can be omitted when printing from the top left.

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

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

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
     Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addPageBegin];
    errorStatus = [builder addPageArea: 100 Y: 50 Width: 200 Height: 100];
    errorStatus = [builder addPagePosition: 50 Y: 30];
    errorStatus = [builder addText: @"ABCDE"];
    errorStatus = [builder addPageEnd];
    ///Process///
}
```

addPageLine

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.79) and addPageEnd (p.80).

Syntax

```
    - (int) addPageLine:(long)x1 Y1:(long)y1 X2:(long)x2
                    Y2:(long)y2 Style:(int)style;
```

Parameter

- x1 : Specifies the horizontal start position of the line (in dots).
  Specifies an integer from 0 to 65535.
- y1 : Specifies the vertical start position of the line (in dots).
  Specifies an integer from 0 to 65535.
- x2 : Specifies the horizontal end position of the line (in dots).
  Specifies an integer from 0 to 65535.
- y2 : Specifies the vertical end position of the line (in dots).
  Specifies an integer from 0 to 65535.
- style : Specifies the line type.

Set value	Description
EPOS_OC_LINE_THIN	Solid line: Thin
EPOS_OC_LINE_MEDIUM	Solid line: Medium
EPOS_OC_LINE_THICK	Solid line: Thick
EPOS_OC_LINE_THIN_DOUBLE	Double line: Thin
EPOS_OC_LINE_MEDIUM_DOUBLE	Double line: Medium
EPOS_OC_LINE_THICK_DOUBLE	Double line: Thick
EPOS_OC_PARAM_DEFAULT	Solid line: Thin

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

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

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-P60"
     Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addPageBegin];
    errorStatus = [builder addPageRectangle: 100 Y1: 0 X2: 500 Y2: 200
                                   Style: EPOS_OC_LINE_THIN);
    errorStatus = [builder addPageEnd];
    ///Process///
}
```

addPageRectangle

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.79) and addPageEnd (p.80).

Syntax

```
- (int) addPageRectangle:(long)x1 Y1:(long)y1
                          X2:(long)x2 Y2:(long)y2
                          Style:(int)style;
```

Parameter

- x1 : Specifies the horizontal start position of the line (in dots).
  Specifies an integer from 0 to 65535.
- y1 : Specifies the vertical start position of the line (in dots).
  Specifies an integer from 0 to 65535.
- x2 : Specifies the horizontal end position of the line (in dots).
  Specifies an integer from 0 to 65535.
- y2 : Specifies the vertical end position of the line (in dots).
  Specifies an integer from 0 to 65535.
- style : Specifies the line type.

Set value	Description
EPOS_OC_LINE_THIN	Solid line: Thin
EPOS_OC_LINE_MEDIUM	Solid line: Medium
EPOS_OC_LINE_THICK	Solid line: Thick
EPOS_OC_LINE_THIN_DOUBLE	Double line: Thin
EPOS_OC_LINE_MEDIUM_DOUBLE	Double line: Medium
EPOS_OC_LINE_THICK_DOUBLE	Double line: Thick
EPOS_OC_PARAM_DEFAULT	Solid line: Thin

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

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

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-P60"
    Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addPageBegin];
    errorStatus = [builder addPageLine: 100 Y1: 0 X2: 500 Y2: 0
                              Style: EPOS_OC_LINE_THIN];
    errorStatus = [builder addPageEnd];
    ///Process///
}
```

addCut

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

⚠ Not available in page mode.

Syntax

```
- (int) addCut:(int)type;
```

Parameter

- type : Specifies the paper cut type.

Set value	Description
EPOS_OC_CUT_NO_FEED	Cut without feeding (The paper is cut without being fed.)
EPOS_OC_CUT_FEED	Feed cut (The paper is fed to the cut position and then is cut.)
EPOS_OC_CUT_RESERVE	Cut reservation (Printing continues until the cut position is reached, at which the paper is cut.)
EPOS_OC_PARAM_DEFAULT	Feed cut (The paper is fed to the cut position and then is cut.)

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

To perform feed cut operation:

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
     Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addCut: EPOS_OC_CUT_FEED];
    ///Process///
}
```

addPulse

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

```
- (int) addPulse:(int)drawer Time:(int)time;
```

Parameter

- drawer : Specifies the drawer kick connector.

Set value	Description
EPOS_OC_DRAWER_1	Pin 2 of the drawer kick-out connector
EPOS_OC_DRAWER_2	Pin 5 of the drawer kick-out connector
EPOS_OC_PARAM_DEFAULT	Pin 2 of the drawer kick-out connector

- time : Specifies the ON time of the drawer kick signal.

Set value	Description
EPOS_OC_PULSE_100	100 ms
EPOS_OC_PULSE_200	200 ms
EPOS_OC_PULSE_300	300 ms
EPOS_OC_PULSE_400	400 ms
EPOS_OC_PULSE_500	500 ms
EPOS_OC_PARAM_DEFAULT	100 ms

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

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

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
     Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addPulse: EPOS_OC_DRAWER_1
        Time: EPOS_OC_PULSE_100];
    ///Process///
}
```

addSound(For setting cycle buzzer)

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

```
- (int) addSound:(int)pattern Repeat:(long)repeat
                              Cycle:(long)cycle;
```

Parameter

- pattern : Specifies the buzzer pattern.

Set value	Description
EPOS_OC_PATTERN_A	Pattern A
EPOS_OC_PATTERN_B	Pattern B
EPOS_OC_PATTERN_C	Pattern C
EPOS_OC_PATTERN_D	Pattern D
EPOS_OC_PATTERN_E	Pattern E
EPOS_OC_PATTERN_ERROR	Error sound pattern
EPOS_OC_PATTERN_PAPER_END	Pattern when there is no paper
EPOS_OC_PATTERN_1	Pattern 1
EPOS_OC_PATTERN_2	Pattern 2
EPOS_OC_PATTERN_3	Pattern 3
EPOS_OC_PATTERN_4	Pattern 4
EPOS_OC_PATTERN_5	Pattern 5
EPOS_OC_PATTERN_6	Pattern 6
EPOS_OC_PATTERN_7	Pattern 7
EPOS_OC_PATTERN_8	Pattern 8
EPOS_OC_PATTERN_9	Pattern 9
EPOS_OC_PATTERN_10	Pattern 10
EPOS_OC_PARAM_DEFAULT	Pattern A

- repeat : Specifies the number of repeats.

Set value	Description
1 to 255	Number of repeats
EPOS_OC_PARAM_DEFAULT	One time

- cycle : This specifies the buzzer sounding cycle (in units of milliseconds)

Set value	Description
1000 to 25500	1000 to 25500 milliseconds
EPOS_OC_PARAM_DEFAULT	1000 milliseconds

"Pattern A to E"/ "Error sound pattern"/"Pattern when there is no paper" is disregarded.

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

When sounding pattern 1 three times at 1,000 millisecond cycles

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
     Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addSound: EPOS_OC_PATTERN_1 Repeat: 3 Cycle: 1000];
    ///Process///
}
```

addSound

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

If you want to optionally set the buzzer sounding cycle (milliseconds), use addSound(For setting cycle buzzer) (p.93).

- 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

```
- (int) addSound:(int)pattern Repeat:(long)repeat;
```

Parameter

- pattern : Specifies the buzzer pattern.

Set value	Description
EPOS_OC_PATTERN_A	Pattern A
EPOS_OC_PATTERN_B	Pattern B
EPOS_OC_PATTERN_C	Pattern C
EPOS_OC_PATTERN_D	Pattern D
EPOS_OC_PATTERN_E	Pattern E
EPOS_OC_PATTERN_ERROR	Error sound pattern
EPOS_OC_PATTERN_PAPER_END	Pattern when there is no paper
EPOS_OC_PATTERN_1	Pattern 1
EPOS_OC_PATTERN_2	Pattern 2
EPOS_OC_PATTERN_3	Pattern 3
EPOS_OC_PATTERN_4	Pattern 4
EPOS_OC_PATTERN_5	Pattern 5
EPOS_OC_PATTERN_6	Pattern 6
EPOS_OC_PATTERN_7	Pattern 7
EPOS_OC_PATTERN_8	Pattern 8
EPOS_OC_PATTERN_9	Pattern 9
EPOS_OC_PATTERN_10	Pattern 10
EPOS_OC_PARAM_DEFAULT	Pattern A

- repeat : Specifies the number of repeats.

Set value	Description
1 to 255	Number of repeats
EPOS_OC_PARAM_DEFAULT	One time

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

To repeat the sound pattern A three times:

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
     Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addSound: EPOS_OC_PATTERN_A Repeat: 3];
    ///Process///
}
```

addFeedPosition

Adds label / black mark paper feeding to the command buffer.

Syntax

```
- (int) addFeedPosition:(int)position;
```

Parameter

- position : Specifies the feed position.

Set value	Description
EPOS_OC_FEED_PEELING	Feeds to the peeling position.
EPOS_OC_FEED_CUTTING	Feeds to the cutting position.
EPOS_OC_FEED_CURRENT_TOF	Feeds to the top of the current label.
EPOS_OC_FEED_NEXT_TOF	Feeds to the top of the next label.

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

To feed a label paper to the peeling position:

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-P60II"
     Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addFeedPosition: EPOS_OC_FEED_PEELING];
    ///Process///
}
```

addLayout

Adds label / black mark paper layout information to the command buffer.

Syntax

```
- (int) addLayout:(int)type
                 Width:(long)width Height:(long)height
                 MarginTop:(long)marginTop
                 MarginBottom:(long)marginBottom
                 OffsetCut:(long)offsetCut
                 OffsetLabel:(long)offsetLabel;
```

Parameter

- type : Specifies the paper type.

Set value	Description
EPOS_OC_LAYOUT_RECEIPT	Receipt paper (no black mark)
EPOS_OC_LAYOUT_LABEL	Label paper (no black mark)
EPOS_OC_LAYOUT_LABEL_BM	Label paper (with black mark)
EPOS_OC_LAYOUT_RECEIPT_BM	Receipt paper (with black mark)

- width : Specifies paper width (in units of 0.1mm). Specifies an integer from 1 to 10000.
- height : Specifies the distance (in units of 0.1mm) from the standard printing position to the next standard printing position. Specifies an integer from 0 to 10000.
  If "0" is specified, the distance from the standard printing position to the next standard printing position is detected automatically.
- marginTop : Specifies the distance (in units of 0.1mm) from the standard printing position to the top position. Specifies an integer from -9999 to 10000.
- marginBottom :Specifies the distance (in units of 0.1mm) from the standard eject position to the bottom edge of the printable area. Specifies an integer from -9999 to 10000.
- offsetCut : Specifies the distance (in units of 0.1mm) from the standard eject position to the cutting position. Specifies an integer from -9999 to 10000.
- offsetLabel : Specifies the distance (in units of 0.1mm) from the standard eject position to the bottom edge of the label. Specifies an integer from 0 to 10000.

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

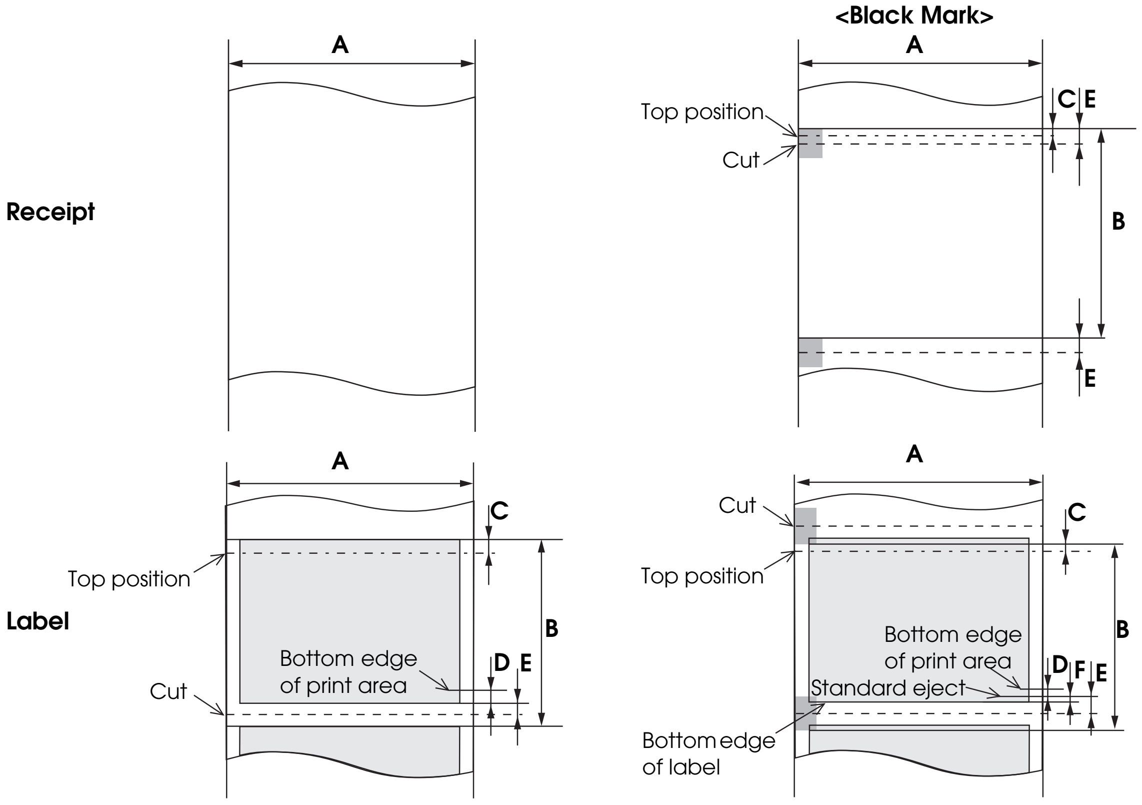
Example

To set 60mm label paper (black mark):

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-P60II"
     Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addLayout: EPOS_OC_PAPER_TYPE_LABEL_BM
                   Width:600 Height:0 MarginTop:15 MarginBottom:-15
                   OffsetCut:15 OffsetLabel:0];
    ///Process///
}
```

Detailed description

❏ See below for the parameters that can be specified for each type of paper, and the positions for those parameters.



Mark	Parameter	Set value			
		Receipt	Receipt (Black mark)	Label	Label (Black mark)
A	width	1 to 10000	1 to 10000	1 to 10000	1 to 10000
B	height	0	0 to 10000	0 to 10000	0 to 10000
C	marginTop	0	-9999 to 10000	0 to 10000	-9999 to 10000
D	marginBottom	0	0	-9999 to 0	-9999 to 10000
E	offsetCut	0	-9999 to 10000	0 to 10000	0 to 10000
F	offsetLabel	0	0	0	0 to 10000

addCommand

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

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

Syntax

```
- (int) addCommand:(NSData *)data;
```

Parameter

- data :   Specifies ESC/POS command as a binary data.

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    NSData* data = Nil;
    ///Process///
    errorStatus = [builder addCommand: data];
}
```

init

Initializes an EposPrint class instance.

Syntax

```
- (id) init;
```

Return Value

The initialized EposPrint class instance is returned.

Example

```
id printer = [[EposPrint alloc] init];
if ( printer != nil) {
    ///Process///
   [printer release];
}
```

openPrinter(For acquiring printer status)

This starts communications with the printer and monitoring of printer status.

✎	• Printer status is notified by the callback method that was registered in the EposPrint class. For details, see Automatic Acquisition of Printer Status (p.35). • If you want to stop monitoring of printer status, call closePrinter (p.106).
⚠	if communication with the printer is not required anymore, be sure to call closePrinter (p.106), closePrinter API, to end communication with the printer.

Syntax

```
- (int) openPrinter:(int)deviceType
                    DeviceName:(NSString *)deviceName
                    Enabled:(int)enabled
                    Interval:(long)interval;
```

Parameter

- deviceType : Specifies the type for the device to start communication.

Set value	Description
EPOS_OC_DEVTYPE_TCP	Wi-Fi/Ethernet device

- deviceName : Specifies the identifier used for identification of the target device.
  Specifies the following for each device type:

Set value	Specified Value
EPOS_OC_DEVTYPE_TCP	IP address (IPv4)

- enabled : This specifies whether printer status monitoring is enabled or disabled.

Set value	Specified Value
EPOS_OC_TRUE	Enabled
EPOS_OC_FALSE	Disabled
EPOS_OC_PARAM_DEFAULT	Select default value (disabled)

- interval : This specifies the interval (in units of milliseconds) for updating printer status.

Set value	Specified Value
1000 to 60000 integer	Interval for updating printer status (in units of milliseconds)
EPOS_OC_PARAM_DEFAULT	Specify the default value (1000)

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_OPEN	Open processing failed.
EPOS_OC_ERR_ILLEGAL	An attempt was made to start communicating with the device with which communication had already started.
EPOS_OC_ERR_PROCESSING	Could not execute process.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

Case where printer status monitoring is enabled and communications are commenced using Wi-Fi/Ethernet and a printer with an IP address of 192.168.192.168

```
id printer = [[EposPrint alloc] init];
if ( printer != nil) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
        Name:@”192.168.192.168” Enabled: EPOS_OC_TRUE
        Interval:EPOS_OC_PARAM_DEFAULT];
    ///Process///
}
```

openPrinter

Starts communication with the printer.

✎	If you want to automatically acquire the printer status, use openPrinter(For acquiring printer status) (p.102).
⚠	if communication with the printer is not required anymore, be sure to call closePrinter (p.106), closePrinter API, to end communication with the printer.

Syntax

```
- (int) openPrinter:(int)deviceType
                    DeviceName:(NSString *)deviceName;
```

Parameter

- deviceType : Specifies the type for the device to start communication.

Set value	Description
EPOS_OC_DEVTYPE_TCP	Wi-Fi/Ethernet device

- deviceName : Specifies the identifier used for identification of the target device.
  Specifies the following for each device type:

Set value	Specified Value
EPOS_OC_DEVTYPE_TCP	IP address (IPv4)

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_OPEN	Open processing failed.
EPOS_OC_ERR_ILLEGAL	An attempt was made to start communicating with the device with which communication had already started.
EPOS_OC_ERR_PROCESSING	Could not execute process.
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

To start communication via Wi-Fi/Ethernet with the printer whose IP address is "192.168.192.168":

```
id printer = [[EposPrint alloc] init];
if ( printer != nil) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
        DeviceName:@"192.168.192.168”];
    ///Process///
}
```

closePrinter

This ends communications with the printer and monitoring of printer status.

Syntax

```
- (int) closePrinter;
```

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_ILLEGAL	This API was called when communication had not started yet.
EPOS_OC_ERR_PROCESSING	Could not execute process.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

```
id printer = [[EposPrint alloc] init];
if ( printer != nil) {
   int errorStatus = EPOS_OC_SUCCESS;
   errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
       DeviceName:@”192.168.192.168”];
    ///Process///
   errorStatus = [printer closePrinter];
}
```

sendData

Sends a print document created using the EposBuilder class.

If you want to acquire the battery status when sending a print document, use sendData(For acquiring battery status) (p.109).

Syntax

```
– (int) sendData:(EposBuilder *)builder
                 Timeout:(long)timeout
                 Status:(unsigned long *)status;
```

Parameter

- builder : Specifies an EposBuilder class instance. For details on the EposBuilder class, refer to EposBuilder class (p.41).
- timeout : Specifies the transmission/reception waiting timeout time. Specifies an integer in the range 0-600000 (in milliseconds).
- status : The printer status when command transmission ended is set. A combination of printer status settings is set. For details, refer to Printer Status List (p.38).

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_ILLEGAL	This API was called when communication had not started yet.
EPOS_OC_ERR_PROCESSING	Could not execute process.
EPOS_OC_ERR_TIMEOUT	Could not send all the data within the specified time.
EPOS_OC_ERR_CONNECT	Connection error occurred
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_OFF_LINE	The printer was offline.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

To send a command to the printer by specifying 10 seconds for its timeout parameter:

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    unsigned long status = 0;

    errorStatus = [builder addText:@"ABCDE"];

    id printer = [[EposPrint alloc] init];

    if ( printer != nil ) {
        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                               DeviceName:@"192.168.192.168"];
        errorStatus = [printer sendData:builder Timeout:10000
                       Status:&status];
        errorStatus = [printer closePrinter];
        [printer release];
    }
    [builder release];
}
```

sendData(For acquiring battery status)

Sends a print document created using the EposBuilder class.

Syntax

```
- (int) sendData:(EposBuilder *)builder
                 Timeout:(long)timeout
                 Status:(unsigned long *)status
                 Battery:(unsigned long *)battery;
```

Parameter

- builder : Specifies an EposBuilder class instance. For details on the EposBuilder class, refer to EposBuilder class (p.41).
- timeout : Specifies the transmission/reception waiting timeout time. Specifies an integer in the range 0-600000 (in milliseconds).
- status : The printer status when command transmission ended is set. A combination of printer status settings is set. For details, refer to Printer Status List (p.38).
- battery : The battery status when command transmission ended is set. For details, refer to Printer specifications (p.155).

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_ILLEGAL	This API was called when communication had not started yet.
EPOS_OC_ERR_PROCESSING	Could not execute process.
EPOS_OC_ERR_TIMEOUT	Could not send all the data within the specified time.
EPOS_OC_ERR_CONNECT	Connection error occurred
EPOS_OC_ERR_MEMORY	Could not allocate memory.
EPOS_OC_ERR_OFF_LINE	The printer was offline.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

To send a command to the printer by specifying 10 seconds for its timeout parameter:

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-P60II"
    Lang: EPOS_OC_MODEL_ANK];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    unsigned long status = 0;
    unsigned long battery = 0;

    errorStatus = [builder addText:@"ABCDE"];

    id printer = [[EposPrint alloc] init];

    if ( printer != nil ) {
        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                                DeviceName:@”192.168.192.168”];
        errorStatus = [printer sendData:builder Timeout:10000
                       Status:&status Battery:&battery];
        errorStatus = [printer closePrinter];
        [printer release];
    }
    [builder release];
}
```

setStatusChangeEventCallback

This registers the callback method for printer status events.

- This API can be executed following execution of openPrinter(For acquiring printer status) (p.102).
- When this API is executed on multiple occasions, the callback method that is specified afterwards is overwritten.

Syntax

```
- (void) setStatusChangeEventCallback: (SEL) method
                                Target: (NSObject*) target;
```

Parameter

• method :	This specifies the callback method selector.
• target :	This specifies the object that has the callback method.

If null is specified for either the method or target, the callback method is nullified.

Definition of Callback Method

```
- (void) Method name: (NSString *)deviceName
                       Status:(NSNumber *)status;
```

Parameter

• deviceName :	The identifier (IPv4 type IP address/MAC address) of the device that is notified of printer status is set.
• status :	Printer status is set.

Example

```
- (void)onStatusChange:(NSString *)deviceName Status:(NSNumber *)status
{
  ///Process///
}

- (void)openPrinter
{
  id printer = [[EposPrint alloc] init];

  if ( printer != nil) {
    int errorStatus = EPOS_OC_SUCCESS;

    [printer setStatusChangeEventCallback @selector(onStatusChange:Status:)
                                                  Target:self];

     errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                            Name:@"192.168.192.168"
                            Enabled: EPOS_OC_TRUE
                            Interval:EPOS_OC_PARAM_DEFAULT];

    ///Process///

  }
}
```

setOnlineEventCallback

This registers the callback method for online events. This is the notification method when printer status is online.

- This API can be executed following execution of openPrinter(For acquiring printer status) (p.102).
- When this API is executed on multiple occasions, the callback method that is specified afterwards is overwritten.

Syntax

```
- (void) setOnlineEventCallback: (SEL) method
                          Target: (NSObject*) target;
```

Parameter

- method : This specifies the callback method selector.
- target : This specifies the object that has the callback method.

If null is specified for either the method or target, the callback method is nullified.

Definition of Callback Method

```
- (void) Method name: (NSString *)deviceName
```

Parameter

- deviceName : The identifier (IPv4 type IP address/MAC address) of the device that is notified of online event is set.

Example

```
- (void)onOnline:(NSString *)deviceName
{
  ///Process///
}

- (void)openPrinter
{
  id printer = [[EposPrint alloc] init];

  if ( printer != nil) {
    int errorStatus = EPOS_OC_SUCCESS;

    [printer setOnlineEventCallback @selector(onOnline:) Target:self];

     errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                               Name:@"192.168.192.168"
                               Enabled: EPOS_OC_TRUE
                               Interval:EPOS_OC_PARAM_DEFAULT];

    ///Process///

  }
}
```

setOfflineEventCallback

This registers the callback method for offline events. This is the notification method when printer is offline.

- This API can be executed following execution of openPrinter(For acquiring printer status) (p.102).
- When this API is executed on multiple occasions, the callback method that is specified afterwards is overwritten.

Syntax

```
- (void) setOfflineEventCallback: (SEL) method
                           Target: (NSObject*) target;
```

Parameter

• method :	This specifies the callback method selector.
• target :	This specifies the object that has the callback method.

If null is specified for either the method or target, the callback method is nullified.

Definition of Callback Method

```
- (void) Method name: (NSString *)deviceName
```

Parameter

- deviceName : The identifier (IPv4 type IP address/MAC address) of the device that is notified of offline event is set.

Example

```
- (void)onOffline:(NSString *)deviceName
{
  ///Process///
}

- (void)openPrinter
{
  id printer = [[EposPrint alloc] init];

  if ( printer != nil) {
    int errorStatus = EPOS_OC_SUCCESS;

    [printer setOfflineEventCallback @selector(onOffline:) Target:self];

     errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                            Name:@"192.168.192.168"
                            Enabled: EPOS_OC_TRUE
                            Interval:EPOS_OC_PARAM_DEFAULT];

    ///Process///
  }
}
```

setPowerOffEventCallback

This registers the callback method for power off events. This is the notification method when there is no response concerning printer status.

- This API can be executed following execution of openPrinter(For acquiring printer status) (p.102).
- When this API is executed on multiple occasions, the callback method that is specified afterwards is overwritten.

Syntax

```
- (void) setPowerOffEventCallback: (SEL) method
                              Target: (NSObject*) target;
```

Parameter

- method : This specifies the callback method selector.
- target : This specifies the object that has the callback method.

If null is specified for either the method or target, the callback method is nullified.

Definition of Callback Method

```
- (void) Method name: (NSString *)deviceName
```

Parameter

- deviceName : The identifier (IPv4 type IP address/MAC address) of the device that is notified of power off event is set.

Example

```
- (void)onPowerOff:(NSString *)deviceName
{
  ///Process///
}

- (void)openPrinter
{
  id printer = [[EposPrint alloc] init];

  if ( printer != nil) {
    int errorStatus = EPOS_OC_SUCCESS;

    [printer setPowerOffEventCallback @selector(onPowerOff:) Target:self];

     errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                            Name:@"192.168.192.168"
                            Enabled: EPOS_OC_TRUE
                            Interval:EPOS_OC_PARAM_DEFAULT];

    ///Process///

  }
}
```

setCoverOkEventCallback

This registers the callback method for cover close events. This is the notification method when printer status indicates cover close.

- This API can be executed following execution of openPrinter(For acquiring printer status) (p.102).
- When this API is executed on multiple occasions, the callback method that is specified afterwards is overwritten.

Syntax

```
- (void) setCoverOkEventCallback: (SEL) method
                                 Target: (NSObject*) target;
```

Parameter

- method : This specifies the callback method selector.
- target : This specifies the object that has the callback method.

If null is specified for either the method or target, the callback method is nullified.

Definition of Callback Method

```
- (void) Method name: (NSString *)deviceName
```

Parameter

- deviceName : The identifier (IPv4 type IP address/MAC address) of the device that is notified of cover ok event is set.

Example

```
- (void)onCoverOk:(NSString *)deviceName
{
  ///Process///
}

- (void)openPrinter
{
  id printer = [[EposPrint alloc] init];

  if ( printer != nil) {
    int errorStatus = EPOS_OC_SUCCESS;

    [printer setCoverOkEventCallback @selector(onCoverOk:) Target:self];

     errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                            Name:@"192.168.192.168"
                            Enabled: EPOS_OC_TRUE
                            Interval:EPOS_OC_PARAM_DEFAULT];

    ///Process///

  }
}
```

setCoverOpenEventCallback

This registers the callback method for cover open events. This is the notification method when printer status indicates cover open.

- This API can be executed following execution of openPrinter(For acquiring printer status) (p.102).
- When this API is executed on multiple occasions, the callback method that is specified afterwards is overwritten.

Syntax

```
- (void) setCoverOpenEventCallback: (SEL) method
                            Target: (NSObject*) target;
```

Parameter

• method :	This specifies the callback method selector.
• target :	This specifies the object that has the callback method.

If null is specified for either the method or target, the callback method is nullified.

Definition of Callback Method

```
- (void) Method name: (NSString *)deviceName
```

Parameter

- deviceName : The identifier (IPv4 type IP address/MAC address) of the device that is notified of cover open event is set.

Example

```
- (void)onCoverOpen:(NSString *)deviceName
{
  ///Process///
}

- (void)openPrinter
{
  id printer = [[EposPrint alloc] init];

  if ( printer != nil) {
    int errorStatus = EPOS_OC_SUCCESS;

    [printer setCoverOpenEventCallback @selector(onCoverOpen:) Target:self];

     errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                             Name:@"192.168.192.168"
                             Enabled: EPOS_OC_TRUE
                             Interval:EPOS_OC_PARAM_DEFAULT];

    ///Process///

  }
}
```

setPaperOkEventCallback

This registers the callback method for paper OK events. This is the notification method when printer status indicates paper ok.

- This API can be executed following execution of openPrinter(For acquiring printer status) (p.102).
- When this API is executed on multiple occasions, the callback method that is specified afterwards is overwritten.

Syntax

```
- (void) setPaperOkEventCallback: (SEL) method
                          Target: (NSObject*) target;
```

Parameter

• method :	This specifies the callback method selector.
• target :	This specifies the object that has the callback method.

If null is specified for either the method or target, the callback method is nullified.

Definition of Callback Method

```
- (void) Method name: (NSString *)deviceName
```

Parameter

- deviceName : The identifier (IPv4 type IP address/MAC address) of the device that is notified of paper ok event is set.

Example

```
- (void)onPaperOk:(NSString *)deviceName
{
  ///Process///
}

- (void)openPrinter
{
  id printer = [[EposPrint alloc] init];

  if ( printer != nil) {
    int errorStatus = EPOS_OC_SUCCESS;

    [printer setPaperOkEventCallback @selector(onPaperOk:) Target:self];

     errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                            Name:@"192.168.192.168"
                            Enabled: EPOS_OC_TRUE
                            Interval:EPOS_OC_PARAM_DEFAULT];

    ///Process///

  }
}
```

setPaperNearEndEventCallback

This registers the callback method for paper near end events. This is the notification method when printer status indicates paper is near the end.

- This API can be executed following execution of openPrinter(For acquiring printer status) (p.102).
- When this API is executed on multiple occasions, the callback method that is specified afterwards is overwritten.

Syntax

```
- (void) setPaperNearEndEventCallback: (SEL) method
                                  Target: (NSObject*) target;
```

Parameter

- method : This specifies the callback method selector.
- target : This specifies the object that has the callback method.

If null is specified for either the method or target, the callback method is nullified.

Definition of Callback Method

```
- (void) Method name: (NSString *)deviceName
```

Parameter

- deviceName : The identifier (IPv4 type IP address/MAC address) of the device that is notified of paper near end event is set.

Example

```
- (void)onPaperNearEnd:(NSString *)deviceName
{
  ///Process///
}

- (void)openPrinter
{
  id printer = [[EposPrint alloc] init];

  if ( printer != nil) {
    int errorStatus = EPOS_OC_SUCCESS;

    [printer setPaperNearEndEventCallback @selector(onPaperNearEnd:) Target:self];

     errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                           Name:@"192.168.192.168"
                           Enabled: EPOS_OC_TRUE
                           Interval:EPOS_OC_PARAM_DEFAULT];

    ///Process///

  }
}
```

setPaperEndEventCallback

This registers the callback method for paper end events. This is the notification method when printer status indicates there is no paper.

- This API can be executed following execution of openPrinter(For acquiring printer status) (p.102).
- When this API is executed on multiple occasions, the callback method that is specified afterwards is overwritten.

Syntax

```
- (void) setPaperEndEventCallback: (SEL) method
                                Target: (NSObject*) target;
```

Parameter

- method : This specifies the callback method selector.
- target : This specifies the object that has the callback method.

If null is specified for either the method or target, the callback method is nullified.

Definition of Callback Method

```
- (void) Method name: (NSString *)deviceName
```

Parameter

- deviceName : The identifier (IPv4 type IP address/MAC address) of the device that is notified of paper end event is set.

Example

```
- (void)onPaperEnd:(NSString *)deviceName
{
  ///Process///
}

- (void)openPrinter
{
  id printer = [[EposPrint alloc] init];

  if ( printer != nil) {
    int errorStatus = EPOS_OC_SUCCESS;

    [printer setPaperEndEventCallback @selector(onPaperEnd:) Target:self];

     errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                            Name:@"192.168.192.168"
                            Enabled: EPOS_OC_TRUE
                            Interval:EPOS_OC_PARAM_DEFAULT];

    ///Process///

  }
}
```

setDrawerClosedEventCallback

This registers the callback method for drawer closed events. This is the notification method when printer status indicates drawer closed.

- This API can be executed following execution of openPrinter(For acquiring printer status) (p.102).
- When this API is executed on multiple occasions, the callback method that is specified afterwards is overwritten.

Syntax

```
- (void) setDrawerClosedEventCallback: (SEL) method
                                  Target: (NSObject*) target;
```

Parameter

- method : This specifies the callback method selector.
- target : This specifies the object that has the callback method.

If null is specified for either the method or target, the callback method is nullified.

Definition of Callback Method

```
- (void) Method name: (NSString *)deviceName
```

Parameter

- deviceName : The identifier (IPv4 type IP address/MAC address) of the device that is notified of drawer closed event is set.

Example

```
- (void)onDrawerClosed:(NSString *)deviceName
{
  ///Process///
}

- (void)openPrinter
{
  id printer = [[EposPrint alloc] init];

  if ( printer != nil) {
    int errorStatus = EPOS_OC_SUCCESS;

    [printer setDrawerClosedEventCallback @selector(onDrawerClosed:) Target:self];

     errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                           Name:@"192.168.192.168"
                           Enabled: EPOS_OC_TRUE
                           Interval:EPOS_OC_PARAM_DEFAULT];

    ///Process///

  }
}
```

setDrawerOpenEventCallback

This registers the callback method for drawer open events. This is the notification method when drawer is open concerning printer status.

- This API can be executed following execution of openPrinter(For acquiring printer status) (p.102).
- When this API is executed on multiple occasions, the callback method that is specified afterwards is overwritten.

Syntax

```
- (void) setDrawerOpenEventCallback: (SEL) method
                                Target: (NSObject*) target;
```

Parameter

- method : This specifies the callback method selector.
- target : This specifies the object that has the callback method.

If null is specified for either the method or target, the callback method is nullified.

Definition of Callback Method

```
- (void) Method name: (NSString *)deviceName
```

Parameter

- deviceName : The identifier (IPv4 type IP address/MAC address) of the device that is notified of drawer open is set.

Example

```
- (void)onDrawerOpen:(NSString *)deviceName
{
  ///Process///
}

- (void)openPrinter
{
  id printer = [[EposPrint alloc] init];

  if ( printer != nil) {
    int errorStatus = EPOS_OC_SUCCESS;

    [printer setDrawerOpenEventCallback @selector(onDrawerOpen:) Target:self];

     errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                            Name:@"192.168.192.168"
                            Enabled: EPOS_OC_TRUE
                            Interval:EPOS_OC_PARAM_DEFAULT];

    ///Process///

  }
}
```

setBatteryLowEventCallback

This registers the callback method for a battery low event. This is the notification method when printer status is offline due to battery.

- This API can be executed following execution of openPrinter(For acquiring printer status) (p.102).
- When this API is executed on multiple occasions, the callback method that is specified afterwards is overwritten.

Syntax

```
- (void) setBatteryLowEventCallback: (SEL) method
                                Target: (NSObject*) target;
```

Parameter

• method :	This specifies the callback method selector.
• target :	This specifies the object that has the callback method.

If null is specified for either the method or target, the callback method is nullified.

Definition of Callback Method

```
- (void) Method name: (NSString *)deviceName
```

Parameter

- deviceName : The identifier (IPv4 format IP address / MAC address) of the device that performed the battery low notification is set.

Example

```
- (void)onBatteryLow:(NSString *)deviceName
{
  ///Process///
}

- (void)openPrinter
{
  id printer = [[EposPrint alloc] init];

  if ( printer != nil) {
    int errorStatus = EPOS_OC_SUCCESS;

    [printer setBatteryLowEventCallback @selector(onBatteryLow:) Target:self];

     errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                            Name:@"192.168.192.168"
                            Enabled: EPOS_OC_TRUE
                            Interval:EPOS_OC_PARAM_DEFAULT];

    ///Process///

  }
}
```

setBatteryOkEventCallback

This registers the callback method for a battery OK event. This is the notification method when the printer status recovers from offline due to remaining battery power.

- This API can be executed following execution of openPrinter(For acquiring printer status) (p.102).
- When this API is executed on multiple occasions, the callback method that is specified afterwards is overwritten.

Syntax

```
- (void) setBatteryOkEventCallback: (SEL) method
                        Target: (NSObject*) target;
```

Parameter

• method :	This specifies the callback method selector.
• target :	This specifies the object that has the callback method.

If null is specified for either the method or target, the callback method is nullified.

Definition of Callback Method

```
- (void) Method name: (NSString *)deviceName
```

Parameter

- deviceName : The identifier (IPv4 format IP address / MAC address) of the device that performed the battery OK event notification is set.

Example

```
- (void)onBatteryOk:(NSString *)deviceName
{
  ///Process///
}

- (void)openPrinter
{
  id printer = [[EposPrint alloc] init];

  if ( printer != nil) {
    int errorStatus = EPOS_OC_SUCCESS;

    [printer setBatteryOkEventCallback @selector(onBatteryOk:) Target:self];

     errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                            Name:@"192.168.192.168"
                            Enabled: EPOS_OC_TRUE
                            Interval:EPOS_OC_PARAM_DEFAULT];

    ///Process///

  }
}
```

setBatteryStatusChangeEventCallback

This registers the callback method for battery status events.

- This API can be executed following execution of openPrinter(For acquiring printer status) (p.102).
- When this API is executed on multiple occasions, the callback method that is specified afterwards is overwritten.

Syntax

```
- (void) setBatteryStatusChangeEventCallback:
                    (SEL) method Target: (NSObject*) target;
```

Parameter

• method :	This specifies the callback method selector.
• target :	This specifies the object that has the callback method.

If null is specified for either the method or target, the callback method is nullified.

Definition of Callback Method

```
- (void) Method name: (NSString *)deviceName
                      Battery:(NSNumber *)battery;
```

Parameter

• deviceName :	The identifier (IPv4 type IP address/MAC address) of the device that is notified of battery status is set.
• battery :	Battery status is set.

Example

```
- (void)onBatteryStatusChange:(NSString *)deviceName Battery:(NSNumber *)battery
{
  ///Process///
}

- (void)openPrinter
{
  id printer = [[EposPrint alloc] init];

  if ( printer != nil) {
    int errorStatus = EPOS_OC_SUCCESS;

    [printer setBatteryStatusChangeEventCallback @selector
                                  (onBatteryStatusChange:Battery:) Target:self];

     errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                            Name:@"192.168.192.168"
                            Enabled: EPOS_OC_TRUE
                            Interval:EPOS_OC_PARAM_DEFAULT];

    ///Process///

  }
}
```

Printer Search API

API to search for printers. The following classes are available.

- ❏ EpsonIoFinder class (p. 138)

EpsonIoFinder class

Class to search for printers. The following APIs are available.

API	Description	Page
start	Starts searching for printers.	p. 138
stop	End communication with the printer.	p. 139
getResult	Getting the printer search result.	p. 140

start

Starts a search for printers of the specified device type.

- If you use this API, be sure to use stop (p.139) to stop the search.
- You cannot call this API when a printer search is already in progress.

Syntax

```
+ (int) start:(int)deviceType
             FindOption:(NSString *)findOption;
```

Parameter

- deviceType : Specifies the device type to search for. The following values can be specified.

deviceType	Description
EPSONIO_OC_DEVTYPE_TCP	Searches for TM devices connected to the network

- findOption : Specifies the setting value when searching for a specific target device.

deviceType	Setting Value
EPSONIO_OC_DEVTYPE_TCP	The broadcast address to search for

Return value

Return value	Description
EPSONIO_OC_SUCCESS	Processing was successful
EPSONIO_OC_ERR_ILLEGAL	This API was called when a search was already in progress
EPSONIO_OC_ERR_PROCESSING	Could not execute process.
EPSONIO_OC_ERR_PARAM	Invalid parameter was passed
EPSONIO_OC_ERR_MEMORY	Could not allocate memory.
EPSONIO_OC_ERR_FAILURE	Unspecified error encountered.

stop

Stops the printer search.

Syntax

```
    + (int) stop;
```

Return value

Return value	Description
EPSONIO_OC_SUCCESS	Processing was successful
EPSONIO_OC_ERR_ILLEGAL	This API was called when a search was not in progress.
EPSONIO_OC_ERR_PROCESSING	Could not execute process.
EPSONIO_OC_ERR_FAILURE	Unspecified error encountered.

getResult

Gets the printer search result until the time when this API was called.

Syntax

```
+ (NSArray *) getResult:(int *)errorStatus;
```

Parameter

- errorStatus :

  Returns the error status.

Return value	Description
EPSONIO_OC_SUCCESS	Processing was successful
EPSONIO_OC_ERR_ILLEGAL	This API was called when a search was not in progress.
EPSONIO_OC_ERR_PROCESSING	Could not execute process.
EPSONIO_OC_ERR_PARAM	Invalid parameter was passed
EPSONIO_OC_ERR_MEMORY	Could not allocate memory.
EPSONIO_OC_ERR_FAILURE	Unspecified error encountered.

Return value

Returns a list of devices found during the search.
Identification information of the found devices is stored as a character string (String type) in the list.
The stored results differ depending on the type of device (deviceType).

deviceType	List to Acquire
EPSONIO_OC_DEVTYPE_TCP	List of IP addresses of printers

Log Setting API

Sets the log output. The following class is available.

❏ EposLog class (p. 141)

EposLog class

Sets the log output function.

API	Description	Page
setLogSettings	Sets the log output function.	p. 141

setLogSettings

Sets the log output function.

Syntax

```
+ (int) setLogSettings:(int) period
                Enabled:(int) enabled
                IpAddress:(NSString *)ipAddress
                Port:(int)port LogSize:(int)logSize
                LogLevel:(int)logLevel;
```

Parameter

- period : Specifies the method of setting the log output function.

Set value	Description
EPOS_OC_LOG_TEMPORARY	The settings of this API are disabled when the application is ended.
EPOS_OC_LOG_PERMANENT	The settings of this API are enabled even after the application is ended.

- enabled : Specifies whether to enable the log output function and the log output destination.

Set value	Description
EPOS_OC_LOG_DISABLE	Disables the log output function.
EPOS_OC_LOG_STORAGE	Outputs log data to the device's storage.
EPOS_OC_LOG_TCP	Outputs log data over TCP.

When "enabled" is set to "EPOS_OC_LOG_STORAGE", enable iTunes file sharing. Follow the steps below.

1. Add "UIFileSharingEnabled" to info.plist of the application.
   It is automatically changed to "Application supports iTunes file sharing".
2. Set "Value" of "Application supports iTunes file sharing" to "YES".

- ipAddress : Specifies the IPv4 IP address for TCP communication.

> If either of the following values is specified for enabled, "nil" can be specified for this parameter.
> ∗ EPOS_OC_LOG_DISABLE
> ∗ EPOS_OC_LOG_STORAGE

- port : Specifies the port number for TCP communication. Specifies an integer from 0 to 65535.

> Even if either of the following values is specified for enabled, specify an integer within the range.
> ∗ EPOS_OC_LOG_DISABLE
> ∗ EPOS_OC_LOG_STORAGE

- logSize : Specifies the maximum size of log data that is saved on the device's storage. Specifies an integer from 1 to 50 (Unit:MB).

> Even if either of the following values is specified for enabled, specify an integer within the range.
> ∗ EPOS_OC_LOG_DISABLE
> ∗ EPOS_OC_LOG_TCP

- logLevel : Specifies the level of log data to be output.

Set value	Description
EPOS_OC_LOG_LOW	Low level

Return Value

Error status	Description
EPOS_OC_SUCCESS	Processing was successful.
EPOS_OC_ERR_PARAM	Invalid parameter was passed.
EPOS_OC_ERR_FAILURE	An unspecified error occurred.

Example

To output log data to port 8080 (IP address: 192.168.192.168) over TCP:

```
errorStatus = [EposLog setLogSettings: EPOS_OC_LOG_PERMANENT
              Enabled:EPOS_OC_LOG_TCP IpAddress:@"192.168.192.168"
              Port:8080 LogSize:10 LogLevel: EPOS_OC_LOG_LOW ];
    ///Process///
}
```

To output log data to the device's storage:

```
errorStatus = [EposLog setLogSettings: EPOS_OC_LOG_PERMANENT
              Enabled: EPOS_OC_LOG_STORAGE IpAddress:nil
              Port:0 LogSize:10 LogLevel: EPOS_OC_LOG_LOW ];
    ///Process///
}
```

To disable the log output function:

```
errorStatus = [EposLog setLogSettings: EPOS_OC_LOG_PERMANENT
              Enabled: EPOS_OC_LOG_DISABLE IpAddress:nil
              Port:0 LogSize:10 LogLevel: EPOS_OC_LOG_LOW ];
    ///Process///
}
```

Command Transmission/Reception

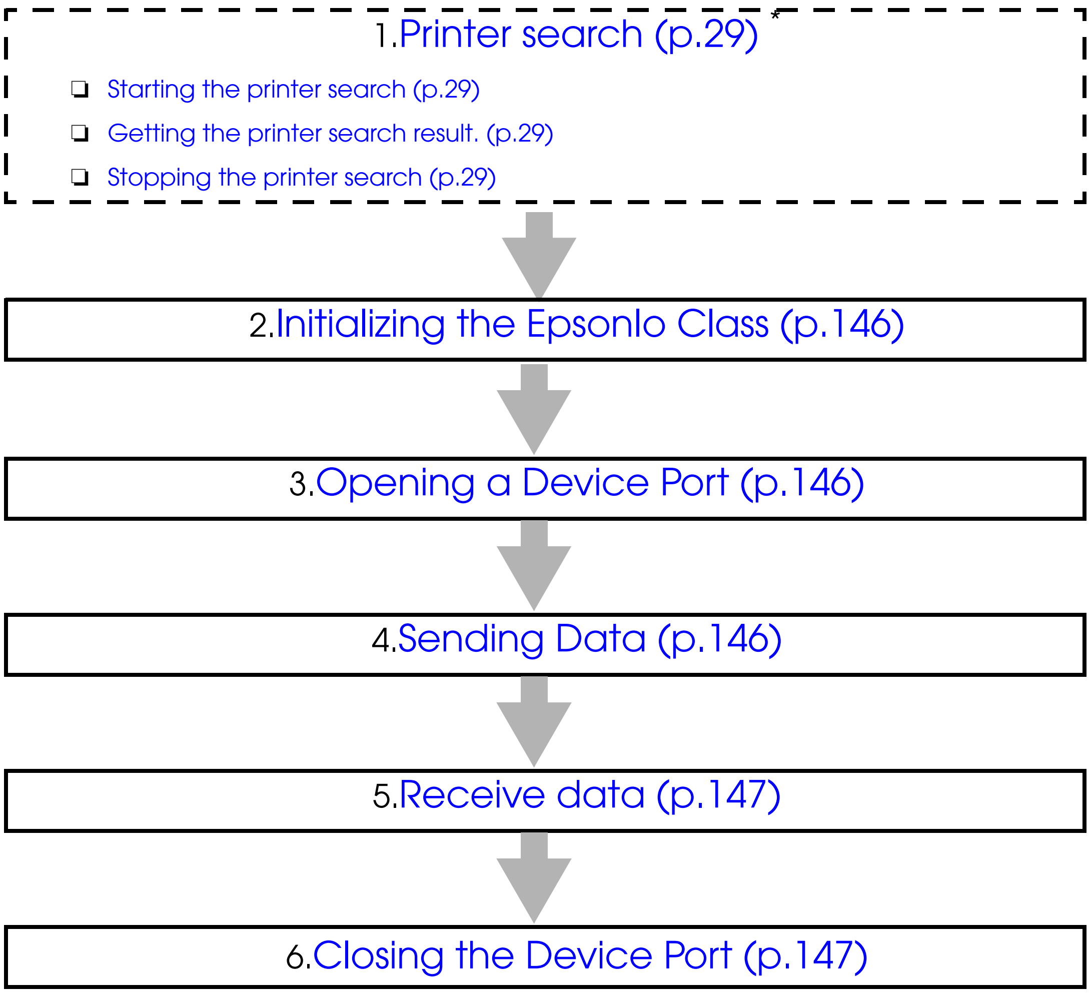
This chapter describes APIs for transmission and reception of commands (ESC/POS commands, etc.).

A command transmission/reception API cannot be used with the EposPrint class (p.43) of ePOS-Print API.

Programming

Programming Flow

Perform programming following this flow.


1.Printer search (p.29) *
❏ Starting the printer search (p.29)
❏ Getting the printer search result. (p.29)
❏ Stopping the printer search (p.29)
2.Initializing the EpsonIo Class (p.146)
3.Opening a Device Port (p.146)
4.Sending Data (p.146)
5.Receive data (p.147)
6.Closing the Device Port (p.147)

*This is optional.

Initializing the EpsonIo Class

Use init (p.149) to initialize the EpsonIo class. Please refer to the following code.

```
//Initializing the EpsonIo Class
id port = [[EpsonIo alloc] init];
if ( port != nil ) {
             -Processing-
    [port release];
}
```

Opening a Device Port

Use the EpsonIo class's open (p.150) to open a device port. Please refer to the following code.

```
//Initializing the EpsonIo Class
id port = [[EpsonIo alloc] init];
if ( port != nil ) {
  int errorStatus = EPSONIO_OC_SUCCESS;
 //Opening a Device Port
  errorStatus = [port open:EPSONIO_OC_DEVTYPE_TCP DeviceName:
   @"192.168.192.168" DeviceSettings:nil];
  if (EPSONIO_OC_SUCCESS == errorStatus ) {
             -Processing-
  }
}
```

Sending Data

Use the EpsonIo class's write (p.152) to send data to the printer. Please refer to the following code.

Printing out "Hello, World!"

```
//Settings for sending
long sizeWritten;
int errStatus;
NSString *str = @"Hello, World!\r\n";
NSData *data = [str dataUsingEncoding:NSASCIIStringEncoding];

//Sending Data
errStatus = [port write:data Offset:0 Size:[data length]
                 Timeout:100 SizeWritten:& sizeWritten];
```

Receive data

Use the EpsonIo class's close (p.151) to receive data from the printer. Please refer to the following code.

```
//Settings for receiving
NSMutableData *data;
long sizeRead;
int errStatus;
data = [[NSMutableData alloc] initWithLength:256];

//Receive data
errStatus =
 [port read:data Offset:0 Size:256 Timeout:100 SizeRead:& sizeRead];
```

Closing the Device Port

Use the EpsonIo class's close (p.151) to close the device port. Please refer to the following code.

```
//Initializing the EpsonIo Class
id port = [[EpsonIo alloc] init];
if ( port != nil ) {
  int errorStatus = EPSONIO_OC_SUCCESS;
 //Opening a Device Port
  errorStatus = [port open:EPSONIO_OC_DEVTYPE_TCP DeviceName:
   @"192.168.192.168" DeviceSettings:nil];
  if (EPSONIO_OC_SUCCESS == errorStatus ) {
              -Processing-
  }
 //Closing the Device Port
  errorStatus = [port close];
}
```

List of Error Values

Error values are defined in the EpsonIo class.

Error Value	Cause
EPSONIO_OC_SUCCESS	Processing was successful
EPSONIO_OC_ERR_PARAM	Invalid parameter was passed. <Example> • An invalid parameter such as Nil was passed. • A value outside the supported range was specified.
EPSONIO_OC_ERR_OPEN,	Open processing failed. <Example> Failed to create a socket for TCP communication..
EPSONIO_OC_ERR_CONNECT	Failed to connect to device. <Example> • Failed to send data to the target device for a reason other than a timeout. • Failed to receive data from the target device for a reason other than a timeout.
EPSONIO_OC_ERR_TIMEOUT	Exceeded the specified timeout period. <Example> • Could not send the specified size of data within the specified period. • Could not receive even a single byte of data within the specified period.
EPSONIO_OC_ERR_MEMORY	Could not allocate the necessary memory for processing.
EPSONIO_OC_ERR_ILLEGAL	Illegal method used. <Example> • The API for sending and receiving data was called when the device port was not open. • The printer search API was called again when a printer search was already in progress.
EPSONIO_OC_ERR_PROCESSING	Could not execute process. <Example> Could not get lock rights to the shared resource because the same process is currently being executed by another thread.
EPSONIO_OC_ERR_FAILURE	Unspecified error encountered.

Command Transmission/Reception API Reference

The following classes are available for command transmission/reception APIs:

EpsonIo class

Class to transmit and receive data. The following APIs are available.

API	Description	Page
init	Initialize an EpsonIo class instance.	p. 149
open	Opens the device port.	p. 150
close	Closes the device port.	p. 151
write	Send data.	p. 152
read	Receive data.	p. 154

init

Initializes an instance of the EpsonIo class that was created.

Syntax

```
- (id) init;
```

Return value

Returns an initialized EpsonIo class instance.

open

Opens the specified device port.

⚠ You can open up to 16 devices ports simultaneously within a single application.

Syntax

```
    - (int) open:(int)deviceType
                  DeviceName:(NSString *)deviceName
                  DeviceSettings:(NSString *)deviceSettings;
```

Parameter

- deviceType : Specifies the device type to open. The following values can be specified.

deviceType	Description
EPSONIO_OC_DEVTYPE_TCP	Specify this when the printer to be opened will connect with Wi-Fi/Ethernet.

- deviceName : Specifies the identifier to locate the target device. The following values can be specified.

deviceType	Setting Value
EPSONIO_OC_DEVTYPE_TCP	IP address (IPv4)

- deviceSettings (Reserved) :
  Specify "nil".

Return value

Return value	Description
EPSONIO_OC_SUCCESS	Processing was successful
EPSONIO_OC_ERR_OPEN	Open processing failed.
EPSONIO_OC_ERR_ILLEGAL	User attempted to open a device that is already open.
EPSONIO_OC_ERR_PROCESSING	Could not execute process.
EPSONIO_OC_ERR_PARAM	Invalid parameter was passed.
EPSONIO_OC_ERR_MEMORY	Could not allocate memory.
EPSONIO_OC_ERR_FAILURE	Unspecified error encountered.

close

Closes the specified device port.

Syntax

```
- (int) close;
```

Return value

Return value	Description
EPSONIO_OC_SUCCESS	Processing was successful
EPSONIO_OC_ERR_ILLEGAL	This API was called when no device port was open.
EPSONIO_OC_ERR_PROCESSING	Could not execute process.
EPSONIO_OC_ERR_FAILURE	Unspecified error encountered.

write

Sends data to a device port.

Syntax

```
- (int) write:(NSData *)data
              Offset:(size_t)offset
              Size:(size_t)size
              Timeout:(long)timeout
              SizeWritten:(size_t *)sizeWritten;
```

Parameter

• data :	The sending data buffer. It stores data to be sent.
• offset :	Specifies the start position for sending data. Please specify the offset value from the top of the sending data buffer.
• size :	Specifies the number of bytes to send.

If "0" is specified for size, no data will be sent.
In such a case, "0" is returned for sizeWritten.

• timeout :	Specifies the time in milliseconds to wait for sending to complete. The maximum value that can be specified is 600000 (which equates to 10 minutes).

- Take the transmission speed and volume of data to be sent into account when specifying the timeout value.
- When the timeout value is short, the sending process will still continue until all the data has been sent, while normal data sending is occurring, even if the timeout value is exceeded.

• sizeWritten :	Stores the number of bytes of data that were sent.

The printer did not necessarily receive the amount of data that sizeWritten returns.

If the amount of time specified in timeout is exceeded, the number of bytes that were sent up to that point is stored in sizeWritten.

Return value

Return value	Description
EPSONIO_OC_SUCCESS	Processing was successful
EPSONIO_OC_ERR_ILLEGAL	This API was called when no device port was open.
EPSONIO_OC_ERR_PROCESSING	Could not execute process.
EPSONIO_OC_ERR_PARAM	Invalid parameter was passed.
EPSONIO_OC_ERR_TIMEOUT	Could not send all data within specified period
EPSONIO_OC_ERR_CONNECT	Connection error occurred
EPSONIO_OC_ERR_MEMORY	Could not allocate memory.
EPSONIO_OC_ERR_FAILURE	Unspecified error encountered.

read

Receives data from a device port.

⚠ This API continues receiving until a receiving error occurs.
However, if not even a single byte of data is received during the period specified in timeout, the process ends.

Syntax

```
- (int) read:(NSMutableData *)data
             Offset:(size_t)offset
             Size:(size_t)size
             Timeout:(long)timeout
             SizeRead:(size_t *)sizeRead;
```

Parameter

- data : The receiving data buffer for storing received data.
- offset : Specifies the point to start storing data in the receiving data buffer.
  Please specify the offset value from the top of the receiving data buffer.
- size : Specifies the number of bytes that can be received.

  If "0" is specified for size, no data will be received.
  In such a case, "0" is returned for sizeRead.

- timeout : Specifies the time in milliseconds to receive data.
  The maximum value that can be specified is 600000 (which equates to 10 minutes).
- sizeRead : Returns the number of bytes that were received.

Return value

Return value	Description
EPSONIO_OC_SUCCESS	Processing was successful
EPSONIO_OC_ERR_ILLEGAL	This API was called when no device port was open.
EPSONIO_OC_ERR_PROCESSING	Could not execute process.
EPSONIO_OC_ERR_PARAM	Invalid parameter was passed.
EPSONIO_OC_ERR_TIMEOUT	Could not receive any data within specified period
EPSONIO_OC_ERR_CONNECT	Connection error occurred
EPSONIO_OC_ERR_MEMORY	Could not allocate memory.
EPSONIO_OC_ERR_FAILURE	Unspecified error encountered.

Appendix

Printer specifications

TM-T88V

		58mm	80mm
Interface		Ethernet, Wi-Fi	
Resolution		180 dpi x 180 dpi (W x H)	
Language		• ANK model • Japanese model • Simplified Chinese model • Traditional Chinese model • Korean model • South Asian model	
Print Width		360 dots	512 dots
Charactners in a Line	Font A	ANK: 30 characters	ANK: 42 characters
	Font B	ANK: 40 characters	ANK: 52 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		360 dots x 831 dots (W x H)	512 dots x 831 dots (W x H)
Page Mode Maximum Area		360 dots x 1662 dots (W x H)	512 dots x 1662 dots (W x H)
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	

	58mm	80mm
Two-Dimensional Code	PDF417, QR Code, MaxiCode, GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional, GS1 DataBar Expanded Stacked (Composite Symbology not supported)	
Paper Cut	Cut, Feed cut	
Drawer Kick-Out	Supported	
Buzzer	Optional	
Battery	Not supported	

List of Supported ePOS-Print APIs

ePOS-Print API	Page	ePOS-Print API	Page
EposBuilder Class			
initWithPrinterModel	44	clearCommandBuffer	46
addTextAlign	47	addTextLineSpace	48
addTextRotate	49	addText	50
addTextLang	51	addTextFont	52
addTextSmooth	53	addTextDouble	54
addTextSize	55	addTextStyle	56
addTextPosition	58	addFeedUnit	59
addFeedLine	60	addImage(For multiple tone printing)	61
addImage	64	addLogo	66
addBarcode	67	addSymbol	73
addPageBegin	79	addPageEnd	80
addPageArea	81	addPageDirection	83
addPagePosition	85	addCut	91
addPulse	92	addSound(For setting cycle buzzer)	93
addSound	95	addCommand	100
EposPrint Class			
init	101	openPrinter (For acquiring printer status)	102
openPrinter	104	closePrinter	106
sendData	107	setStatusChangeEventCallback	111
setOnlineEventCallback	113	setOfflineEventCallback	115
setPowerOffEventCallback	116	setCoverOkEventCallback	118
setCoverOpenEventCallback	120	setPaperOkEventCallback	122
setPaperNearEndEventCallback	124	setPaperEndEventCallback	126
setDrawerClosedEventCallback	128	setDrawerOpenEventCallback	130

All the command transmission/reception APIs are supported.

TM-T70

		80mm
Interface		Ethernet, Wi-Fi
Resolution		180 dpi x 180 dpi (W x H)
Language		• ANK model • Japanese model • Simplified Chinese model • Traditional Chinese model • South Asian model
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)
Bar Code		UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF, CODABAR, CODE93, CODE128
Two-Dimensional Code		QR Code
Paper Cut		Cut, Feed cut
Drawer Kick-Out		Supported
Buzzer		Not supported
Battery		Not supported

List of Supported ePOS-Print APIs

ePOS-Print API	Page	ePOS-Print API	Page
EposBuilder Class			
initWithPrinterModel	44	clearCommandBuffer	46
addTextAlign	47	addTextLineSpace	48
addTextRotate	49	addText	50
addTextLang	51	addTextFont	52
addTextSmooth	53	addTextDouble	54
addTextSize	55	addTextStyle	56
addTextPosition	58	addFeedUnit	59
addFeedLine	60	addImage(For multiple tone printing)	61
addImage	64	addLogo	66
addBarcode	67	addSymbol	73
addPageBegin	79	addPageEnd	80
addPageArea	81	addPageDirection	83
addPagePosition	85	addCut	91
addPulse	92	addCommand	100
EposPrint Class			
init	101	openPrinter (For acquiring printer status)	102
openPrinter	104	closePrinter	106
sendData	107	setStatusChangeEventCallback	111
setOnlineEventCallback	113	setOfflineEventCallback	115
setPowerOffEventCallback	116	setCoverOkEventCallback	118
setCoverOpenEventCallback	120	setPaperOkEventCallback	122
setPaperNearEndEventCallback	124	setPaperEndEventCallback	126
setDrawerClosedEventCallback	128	setDrawerOpenEventCallback	130

All the command transmission/reception APIs are supported.

TM-T70II

		58mm	80mm
Interface		Ethernet, Wi-Fi	
Resolution		ANK: 180 dpi x 180 dpi (W x H) Other: 203 dpi x 203 dpi (W x H)	
Language		• ANK model • Japanese model • Simplified Chinese model • Traditional Chinese model • Korean model • South Asian model	
Print Width		ANK: 360 dots Other: 416 dots	ANK: 512 dots Other: 576 dots
Characters in a Line	Font A	ANK: 30 characters Other: 34 characters	ANK: 42 characters Other: 48 characters
	Font B	ANK: 40 characters Other: 52 characters	ANK: 56 characters Other: 72 characters
Character Size	Font A	12 dots x 24 dots (W x H)	
	Font B	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		ANK: 360 dots x 1662 dots (W x H) Other: 416 dots x 1662 dots (W x H)	ANK: 512 dots x 1662 dots (W x H) Other: 576 dots x 1662 dots (W x H)
Page Mode Maximum Area		ANK: 360 dots x 1662 dots (W x H) Other: 416 dots x 1662 dots (W x H)	ANK: 512 dots x 1662 dots (W x H) Other: 576 dots x 1662 dots (W x H)
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	

	58mm	80mm
Paper Cut	Cut, Feed cut	
Drawer Kick-Out	Supported	
Buzzer	Supported	
Battery	Not supported	

List of Supported ePOS-Print APIs

ePOS-Print API	Page	ePOS-Print API	Page
EposBuilder Class			
initWithPrinterModel	44	clearCommandBuffer	46
addTextAlign	47	addTextLineSpace	48
addTextRotate	49	addText	50
addTextLang	51	addTextFont	52
addTextSmooth	53	addTextDouble	54
addTextSize	55	addTextStyle	56
addTextPosition	58	addFeedUnit	59
addFeedLine	60	addImage(For multiple tone printing)	61
addImage	64	addLogo	66
addBarcode	67	addSymbol	73
addPageBegin	79	addPageEnd	80
addPageArea	81	addPageDirection	83
addPagePosition	85	addCut	91
addPulse	92	addSound(For setting cycle buzzer)	93
addSound	95	addCommand	100
EposPrint Class			
init	101	openPrinter (For acquiring printer status)	102
openPrinter	104	closePrinter	106
sendData	107	setStatusChangeEventCallback	111
setOnlineEventCallback	113	setOfflineEventCallback	115
setPowerOffEventCallback	116	setCoverOkEventCallback	118
setCoverOpenEventCallback	120	setPaperOkEventCallback	122
setPaperNearEndEventCallback	124	setPaperEndEventCallback	126
setDrawerClosedEventCallback	128	setDrawerOpenEventCallback	130

All the command transmission/reception APIs are supported.

TM-T90II

		58mm	80mm
Interface		Ethernet, Wi-Fi	
Resolution		203 dpi x 203 dpi (W x H)	
Language		Japanese model	
Print Width		416 dots	576 dots
Characters in a Line	Font A	35 characters	48 characters
	Font B	42 characters	5772 characters
	Font C	52 characters	72 characters
Character Size	Font A	12 dots x 24 dots (W x H)	
	Font B	9 dots x 17 dots (W x H)	
	Font C	8 dots x 16 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	
	Font C	At the 15th dot from the top of the character	
Default Line Feed Space		30 dots	
Color Specification		First color	
Page Mode Default Area		416 dots x 1662 dots (W x H)	576 dots x 1662 dots (W x H)
Page Mode Maximum Area		416 dots x 1662 dots (W x H)	576 dots x 1662 dots (W x H)
Bar Code		Codabar, Code39, ITF, JAN13(EAN), JAN8(EAN), UPC-A, UPC-E, Code93, Code128, GS1-128, GS1 DataBar Omni-directional, GS1 DataBar Truncated, GS1 DataBar Expanded, GS1 DataBar Limited	
Two-Dimensional Code		PDF417, QRCode, MaxiCode, GS1 DataBar Stacked, GS1 DataBar Stacked Omni-directional, GS1 DataBar Expanded Stacked	
Paper Cut		Cut, Feed cut	Cut, Feed cut
Drawer Kick-Out		Supported	Supported
Buzzer		Supported	Supported
Battery		Not supported	Not supported

List of Supported ePOS-Print APIs

ePOS-Print API	Page	ePOS-Print API	Page
EposBuilder Class			
initWithPrinterModel	44	clearCommandBuffer	46
addTextAlign	47	addTextLineSpace	48
addTextRotate	49	addText	50
addTextLang	51	addTextFont	52
addTextSmooth	53	addTextDouble	54
addTextSize	55	addTextStyle	56
addTextPosition	58	addFeedUnit	59
addFeedLine	60	addImage(For multiple tone printing)	61
addImage	64	addLogo	66
addBarcode	67	addSymbol	73
addPageBegin	79	addPageEnd	80
addPageArea	81	addPageDirection	83
addPagePosition	85	addCut	91
addPulse	92	addCommand	100
EposPrint Class			
init	101	openPrinter (For acquiring printer status)	102
openPrinter	104	closePrinter	106
sendData	107	setStatusChangeEventCallback	111
setOnlineEventCallback	113	setOfflineEventCallback	115
setPowerOffEventCallback	116	setCoverOkEventCallback	118
setCoverOpenEventCallback	120	setPaperOkEventCallback	122
setPaperNearEndEventCallback	124	setPaperEndEventCallback	126
setDrawerClosedEventCallback	128	setDrawerOpenEventCallback	130

All the command transmission/reception APIs are supported.

TM-P60

		58mm	60mm
Interface		Wi-Fi	
Resolution		203 dpi x 203 dpi (W x H)	
Language		ANK model	
Print Width		420 dots	432 dots
Characters in a Line	Font A	ANK: 35 characters	ANK: 36 characters
	Font B	ANK: 42 characters	ANK: 43 characters
	Font C	ANK: 52 characters	ANK: 54 characters
Character Size	Font A	ANK: 12 dots x 24 dots (W x H)	
	Font B	ANK: 10 dots x 24 dots (W x H)	
	Font C	ANK: 8 dots x 16 dots (W x H)	
Character Baseline	Font A	At the 21st dot from the top of the character	
	Font B	At the 21st dot from the top of the character	
	Font C	At the 15th dot from the top of the character	
Default Line Feed Space		30 dots	
Color Specification		First color	
Page Mode Default Area		420 dots x 1200 dots (W x H)	432 dots x 1200 dots (W x H)
Page Mode Maximum Area		420 dots x 1200 dots (W x H)	432 dots x 1200 dots (W x H)
Bar Code		UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF, CODABAR,CODE93, CODE128	
Two-Dimensional Code		Not supported	
Paper Cut		Cut, No cut	
Drawer Kick-Out		Not supported	
Buzzer		Supported	
Battery		Supported	

List of Supported ePOS-Print APIs

ePOS-Print API	Page	ePOS-Print API	Page
EposBuilder Class			
initWithPrinterModel	44	clearCommandBuffer	46
addTextAlign	47	addTextLineSpace	48
addTextRotate	49	addText	50
addTextLang	51	addTextFont	52
addTextSmooth	53	addTextDouble	54
addTextSize	55	addTextStyle	56
addTextPosition	58	addFeedUnit	59
addFeedLine	60	addFeedPosition*	97
addImage(For multiple tone printing)	61	addImage	64
addLogo	66	addBarcode	67
addPageBegin	79	addPageEnd	80
addPageArea	81	addPageDirection	83
addPagePosition	85	addPageLine	87
addPageRectangle	89	addCut	91
addLayout*	98	addCommand	100
EposPrint Class			
init	101	openPrinter (For acquiring printer status)	102
openPrinter	104	closePrinter	106
sendData	107	sendData (For acquiring battery status)	109
setStatusChangeEventCallback	111	setOnlineEventCallback	113
setOfflineEventCallback	115	setPowerOffEventCallback	116
setCoverOkEventCallback	118	setCoverOpenEventCallback	120
setPaperOkEventCallback	122	setPaperNearEndEventCallback	124
setPaperEndEventCallback	126	setBatteryLowEventCallback	132
setBatteryOkEventCallback	134	setBatteryStatusChangeEventCallback	136

*:Only for the peeler model.

All the command transmission/reception APIs are supported.

Battery Status

Upper 8 bits

Battery Status	Cause
0x30	The AC adapter is connected
0x31	The AC adapter is not connected

Lower 8 bits

Battery Status	Cause
0x30	H level
0x31	M level
0x32	L level
0x33	S level
0x34	Battery not installed

If 0x0000 is returned, the battery status cannot be acquired.

TM-P60II

		58mm	60mm
Interface		Wi-Fi	
Resolution		203 dpi x 203 dpi (W x H)	
Country		• North America • Europe	
Print Width		420 dots	432 dots
Characters in a Line	Font A	ANK: 35 characters	ANK: 36 characters
	Font B	ANK: 42 characters	ANK: 43 characters
	Font C	ANK: 52 characters	ANK: 54 characters
Character Size	Font A	ANK: 12 dots x 24 dots (W x H)	
	Font B	ANK: 10 dots x 24 dots (W x H)	
	Font C	ANK: 8 dots x 16 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	
	Font C	At the 15th dot from the top of the character	
Default Line Feed Space		30 dots	
Color Specification		First color	
Page Mode Default Area		420 dots x 831 dots (W x H)	576 dots x 831 dots (W x H)
Page Mode Maximum Area		420 dots x 1662 dots (W x H)	576 dots x 1662 dots (W x H)
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	
Paper Cut		Cut, Feed cut	
Drawer Kick-Out		Not supported	
Buzzer		Optional	
Battery		Supported	

List of Supported ePOS-Print APIs

ePOS-Print API	Page	ePOS-Print API	Page
EposBuilder Class			
initWithPrinterModel	44	clearCommandBuffer	46
addTextAlign	47	addTextLineSpace	48
addTextRotate	49	addText	50
addTextLang	51	addTextFont	52
addTextSmooth	53	addTextDouble	54
addTextSize	55	addTextStyle	56
addTextPosition	58	addFeedUnit	59
addFeedLine	60	addFeedPosition*	97
addImage(For multiple tone printing)	61	addImage	64
addLogo	66	addBarcode	67
addSymbol	73	addPageBegin	79
addPageEnd	80	addPageArea	81
addPageDirection	83	addPagePosition	85
addPageLine	87	addPageRectangle	89
addCut	91	addSound(For setting cycle buzzer)	93
addSound	95	addLayout*	98
addCommand	100		
EposPrint Class			
init	101	openPrinter (For acquiring printer status)	102
openPrinter	104	closePrinter	106
sendData	107	sendData (For acquiring battery status)	109
setStatusChangeEventCallback	111	setOnlineEventCallback	113
setOfflineEventCallback	115	setPowerOffEventCallback	116
setCoverOkEventCallback	118	setCoverOpenEventCallback	120
setPaperOkEventCallback	122	setPaperNearEndEventCallback	124
setPaperEndEventCallback	126	setBatteryLowEventCallback	132
setBatteryOkEventCallback	134	setBatteryStatusChangeEventCallback	136

*:Only for the peeler model.

All the command transmission/reception APIs are supported.

Battery Status

Upper 8 bits

Battery Status	Cause
0x30	The AC adapter is connected
0x31	The AC adapter is not connected

Lower 8 bits

Battery Status	Cause
0x30	Battery amount 0 (real end)
0x31	Battery amount 1 (near end)
0x32	Battery amount 2
0x33	Battery amount 3
0x34	Battery amount 4
0x35	Battery amount 5
0x36	Battery amount 6

If 0x0000 is returned, the battery status cannot be acquired.

TM-P80

		80mm
Interface		Wi-Fi
Resolution		203 dpi x 203 dpi (W x H)
Language		ANK model
Print Width		576 dots
Characters in a Line	Font A	ANK: 48 characters
	Font B	ANK: 64 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		576 dots x 1662 dots (W x H)
Page Mode Maximum Area		576 dots x 1662 dots (W x H)
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, Data Matrix, Aztec Code, GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional, GS1 DataBar Expanded Stacked, Composite Symbology
Paper Cut		Feed
Drawer Kick-Out		Not supported
Buzzer		Optional
Battery		Supported

List of Supported ePOS-Print APIs

All the command transmission/reception APIs are supported.

Battery Status

Upper 8 bits

Battery Status	Cause
0x30	The AC adapter is connected
0x31	The AC adapter is not connected

Lower 8 bits

Battery Status	Cause
0x30	Battery amount 0 (real end)
0x31	Battery amount 1 (near end)
0x32	Battery amount 2
0x33	Battery amount 3
0x34	Battery amount 4
0x35	Battery amount 5
0x36	Battery amount 6

If 0x0000 is returned, the battery status cannot be acquired.

TM-U220

		76mm	69.5mm	57.5mm
Interface		Ethernet, Wi-Fi		
Resolution		160 dpi x 72 dpi (W x H)		
Language		• ANK model • Japanese model • Simplified Chinese model • Traditional Chinese model • Korean model • Thai model • South Asian model		
Print Width		400 or 385*1 dots	360 dots	300 or 297*1 dots
Characters in a Line	Font A	ANK: 40 characters	ANK: 36 characters	ANK: 30 characters
	Font B	ANK: 33 characters	ANK: 30 characters	ANK: 25 characters
Character Size	Font A	ANK: 7 dots x 9 dots (W x H)		
	Font B	ANK: 9 dots x 9 dots (W x H)		
Character Baseline	Font A	-		
	Font B	-		
Default Line Feed Space		12 dots		
Color Specification		First color		
Page Mode Default Area		-		
Page Mode Maximum Area		-		
Bar Code		Not supported		
Two-Dimensional Code		Not supported		
Paper Cut		Cut, No cut		
Drawer Kick-Out		Supported		
Buzzer		Not supported		
Battery		Not supported		

*1: DipSW2-1 = ON

List of Supported ePOS-Print APIs

ePOS-Print API	Page	ePOS-Print API	Page
EposBuilder Class			
initWithPrinterModel	44	clearCommandBuffer	46
addTextAlign	47	addTextLineSpace	48
addTextRotate	49	addText	50
addTextLang	51	addTextFont	52
addTextStyle	56	addFeedUnit	59
addFeedLine	60	addCut	91
addPulse	92	addCommand	100
EposPrint Class			
init	101	openPrinter (For acquiring printer status)	102
openPrinter	104	closePrinter	106
sendData	107	setStatusChangeEventCallback	111
setOnlineEventCallback	113	setOfflineEventCallback	115
setPowerOffEventCallback	116	setCoverOkEventCallback	118
setCoverOpenEventCallback	120	setPaperOkEventCallback	122
setPaperNearEndEventCallback	124	setPaperEndEventCallback	126
setDrawerClosedEventCallback	128	setDrawerOpenEventCallback	130

All the command transmission/reception APIs are supported.

TM-T20

		58mm	80mm
Interface		Ethernet	
Resolution		203 dpi x 203 dpi (W x H)	
Language		• ANK model • Japanese model	
Print Width		420 dots	576 dots
Characters in a Line	Font A	ANK: 35 characters	ANK: 48 characters
	Font B	ANK: 46 characters	ANK: 64 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		420 dots x 831 dots (W x H)	576 dots x 831 dots (W x H)
Page Mode Maximum Area		420 dots x 1662 dots (W x H)	576 dots x 1662 dots (W x H)
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	
Paper Cut		Cut, Feed cut	
Drawer Kick-Out		Supported	
Buzzer		Not supported	
Battery		Not supported	

List of Supported ePOS-Print APIs

ePOS-Print API	Page	ePOS-Print API	Page
EposBuilder Class			
initWithPrinterModel	44	clearCommandBuffer	46
addTextAlign	47	addTextLineSpace	48
addTextRotate	49	addText	50
addTextLang	51	addTextFont	52
addTextSmooth	53	addTextDouble	54
addTextSize	55	addTextStyle	56
addTextPosition	58	addFeedUnit	59
addFeedLine	60	addImage(For multiple tone printing)	61
addImage	64	addLogo	66
addBarcode	67	addSymbol	73
addPageBegin	79	addPageEnd	80
addPageArea	81	addPageDirection	83
addPagePosition	85	addCut	91
addPulse	92	addCommand	100
EposPrint Class			
init	101	openPrinter (For acquiring printer status)	102
openPrinter	104	closePrinter	106
sendData	107	setStatusChangeEventCallback	111
setOnlineEventCallback	113	setOfflineEventCallback	115
setPowerOffEventCallback	116	setCoverOkEventCallback	118
setCoverOpenEventCallback	120	setPaperOkEventCallback	122
setPaperNearEndEventCallback	124	setPaperEndEventCallback	126
setDrawerClosedEventCallback	128	setDrawerOpenEventCallback	130

All the command transmission/reception APIs are supported.

TM-T20II

		58mm	80mm
Interface		Ethernet	
Resolution		203 dpi x 203 dpi (W x H)	
Language		• ANK model	
Print Width		420 dots	576 dots
Characters in a Line	Font A	ANK: 35 characters	ANK: 48 characters
	Font B	ANK: 46 characters	ANK: 64 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		420 dots x 831 dots (W x H)	576 dots x 831 dots (W x H)
Page Mode Maximum Area		420 dots x 1662 dots (W x H)	576 dots x 1662 dots (W x H)
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	
Paper Cut		Cut, Feed cut	
Drawer Kick-Out		Supported	
Buzzer		Optional	
Battery		Not supported	

List of Supported ePOS-Print APIs

ePOS-Print API	Page	ePOS-Print API	Page
EposBuilder Class			
initWithPrinterModel	44	clearCommandBuffer	46
addTextAlign	47	addTextLineSpace	48
addTextRotate	49	addText	50
addTextLang	51	addTextFont	52
addTextSmooth	53	addTextDouble	54
addTextSize	55	addTextStyle	56
addTextPosition	58	addFeedUnit	59
addFeedLine	60	addImage(For multiple tone printing)	61
addImage	64	addLogo	66
addBarcode	67	addSymbol	73
addPageBegin	79	addPageEnd	80
addPageArea	81	addPageDirection	83
addPagePosition	85	addCut	91
addPulse	92	addSound(For setting cycle buzzer)	93
addSound	95	addCommand	100
EposPrint Class			
init	101	openPrinter (For acquiring printer status)	102
openPrinter	104	closePrinter	106
sendData	107	setStatusChangeEventCallback	111
setOnlineEventCallback	113	setOfflineEventCallback	115
setPowerOffEventCallback	116	setCoverOkEventCallback	118
setCoverOpenEventCallback	120	setPaperOkEventCallback	122
setPaperNearEndEventCallback	124	setPaperEndEventCallback	126
setDrawerClosedEventCallback	128	setDrawerOpenEventCallback	130

All the command transmission/reception APIs are supported.

TM-T81II

		80mm
Interface		Ethernet
Resolution		203 dpi x 203 dpi (W x H)
Language		Simplified Chinese model
Print Width		576 dots
Characters in a Line	Font A	ANK: 48 characters
	Font B	ANK: 64 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		576 dots x 831 dots (W x H)
Page Mode Maximum Area		576 dots x 1662 dots (W x H)
Bar Code		UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF,CODABAR, CODE93, CODE128
Two-Dimensional Code		PDF417, QR Code
Paper Cut		Cut, Feed cut
Drawer Kick-Out		Supported
Buzzer		Not supported
Battery		Not supported

List of Supported ePOS-Print APIs

ePOS-Print API	Page	ePOS-Print API	Page
EposBuilder Class			
initWithPrinterModel	44	clearCommandBuffer	46
addTextAlign	47	addTextLineSpace	48
addTextRotate	49	addText	50
addTextLang	51	addTextFont	52
addTextSmooth	53	addTextDouble	54
addTextSize	55	addTextStyle	56
addTextPosition	58	addFeedUnit	59
addFeedLine	60	addImage(For multiple tone printing)	61
addImage	64	addLogo	66
addBarcode	67	addSymbol	73
addPageBegin	79	addPageEnd	80
addPageArea	81	addPageDirection	83
addPagePosition	85	addCut	91
addPulse	92	addCommand	100
EposPrint Class			
init	101	openPrinter (For acquiring printer status)	102
openPrinter	104	closePrinter	106
sendData	107	setStatusChangeEventCallback	111
setOnlineEventCallback	113	setOfflineEventCallback	115
setPowerOffEventCallback	116	setCoverOkEventCallback	118
setCoverOpenEventCallback	120	setPaperOkEventCallback	122
setPaperNearEndEventCallback	124	setPaperEndEventCallback	126
setDrawerClosedEventCallback	128	setDrawerOpenEventCallback	130

All the command transmission/reception APIs are supported.

TM-T82

		58mm	80mm
Interface		Ethernet	
Resolution		203 dpi x 203 dpi (W x H)	
Language		• Simplified Chinese model • South Asian model	
Print Width		420 dots	576 dots
Characters in a Line	Font A	ANK: 35 characters	ANK: 48 characters
	Font B	ANK: 46 characters	ANK: 64 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		420 dots x 831 dots (W x H)	576 dots x 831 dots (W x H)
Page Mode Maximum Area		420 dots x 1662 dots (W x H)	576 dots x 1662 dots (W x H)
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	
Paper Cut		Cut, Feed cut	
Drawer Kick-Out		Supported	
Buzzer		Optional	
Battery		Optional	

List of Supported ePOS-Print APIs

ePOS-Print API	Page	ePOS-Print API	Page
EposBuilder Class			
initWithPrinterModel	44	clearCommandBuffer	46
addTextAlign	47	addTextLineSpace	48
addTextRotate	49	addText	50
addTextLang	51	addTextFont	52
addTextSmooth	53	addTextDouble	54
addTextSize	55	addTextStyle	56
addTextPosition	58	addFeedUnit	59
addFeedLine	60	addImage(For multiple tone printing)	61
addImage	64	addLogo	66
addBarcode	67	addSymbol	73
addPageBegin	79	addPageEnd	80
addPageArea	81	addPageDirection	83
addPagePosition	85	addCut	91
addPulse	92	addSound(For setting cycle buzzer)	93
addSound	95	addCommand	100
EposPrint Class			
init	101	openPrinter (For acquiring printer status)	102
openPrinter	104	closePrinter	106
sendData	107	setStatusChangeEventCallback	111
setOnlineEventCallback	113	setOfflineEventCallback	115
setPowerOffEventCallback	116	setCoverOkEventCallback	118
setCoverOpenEventCallback	120	setPaperOkEventCallback	122
setPaperNearEndEventCallback	124	setPaperEndEventCallback	126
setDrawerClosedEventCallback	128	setDrawerOpenEventCallback	130

All the command transmission/reception APIs are supported.

TM-T82II

		58mm	80mm
Interface		Ethernet	
Resolution		203 dpi x 203 dpi (W x H)	
Language		• ANK model • Simplified Chinese model • Traditional Chinese model • South Asian model	
Print Width		420 dots	576 dots
Characters in a Line	Font A	ANK: 35 characters	ANK: 48 characters
	Font B	ANK: 46 characters	ANK: 64 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		420 dots x 831 dots (W x H)	576 dots x 831 dots (W x H)
Page Mode Maximum Area		420 dots x 1662 dots (W x H)	576 dots x 1662 dots (W x H)
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	
Paper Cut		Cut, Feed cut	
Drawer Kick-Out		Supported	
Buzzer		Optional	
Battery		Not supported	

List of Supported ePOS-Print APIs

All the command transmission/reception APIs are supported.