EPSON
EXCEED YOUR VISION

ePOS-Print SDK for Android User's Manual

M00048915
Rev.P

Cautions

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

Trademarks

EPSON is a registered trademark of Seiko Epson Corporation.

Exceed Your Vision and ESC/POS are registered trademarks or trademarks of Seiko Epson Corporation.

Android™ is trademark of Google Inc. in the United States and other countries.

Java™ is a registered trademark of Oracle Corporation, its subsidiaries, and affiliates in the U.S. and other countries.

Wi-Fi® is a registered trademark of the Wi-Fi Alliance®.

The *Bluetooth*® word mark and logos are registered trademarks owned by Bluetooth SIG, Inc. and any use of such marks by Seiko Epson Corporation is under license.

Eclipse® is a trademark or registered trademark of Eclipse Foundation, Inc.

QR Code® is a registered trademark of DENSO Wave Incorporated.

All other trademarks are the property of their respective owners and used for identification purpose only.

ESC/POS® Command System

EPSON ESC/POS is a proprietary POS printer command system that includes patented or patent-pending commands. ESC/POS is compatible with most EPSON POS printers and displays.

ESC/POS is designed to reduce the processing load on the host computer in POS environments. It comprises a set of highly functional and efficient commands and also offers the flexibility to easily make future upgrades.

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

Programming Guide 29

API Reference 55

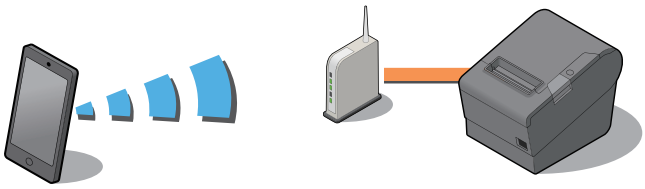
Command Transmission/Reception 167

Appendix 177

Overview

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

Overview of ePOS-Print SDK



The ePOS-Print SDK for Android is an SDK aimed at development engineers who are developing Android™ 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 iOS" for iOS 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, Print class. For details on the command transmission/reception APIs, refer to Command Transmission/Reception (p.167).

Features

- ❏ Allows printing to TM printers from Android applications.
- ❏ Allows acquisition of TM printer status from Android 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 Support Information by Printer (p.178).)
- ❏ 2D-Code printing
  (For 2D-Code that can be printed by each model, refer to Support Information by Printer (p.178).)
- ❏ 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

Printer Easy Select API

- ❏ Select a printer easily
  (You can select a printer easily by using NFC or QR code.)

Log Setting API

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

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

Operating Environment

Android Version

- ❏ Android Version 2.3.3 to 2.3.7
- ❏ Android Version 3.1 to 3.2.2
- ❏ Android Version 4.0 to 4.4
- ❏ Android Version 5.0 to 5.1

- USB is supported for Android Version 3.1 and over.
- For the latest version, refer to the README file.

Android Device

Device that supports ARMv5TE

Printer

TM Printer	Interface			
	Wired LAN	Wi-Fi®	*Bluetooth*®	USB
TM-m10	✔	-	-	✔
TM-P20	-	✔	✔	✔
TM-P60(Receipt) Wi-Fi	-	✔	-	-
TM-P60(Receipt) *Bluetooth*	-	-	✔	-
TM-P60(Peeler) Wi-Fi	-	✔	-	-
TM-P60(Peeler) *Bluetooth*	-	-	✔	-
TM-P60II(Receipt) Wi-Fi	-	✔	-	✔
TM-P60II(Receipt) *Bluetooth*	-	-	✔	✔
TM-P60II(Peeler) Wi-Fi	-	✔	-	✔
TM-P60II(Peeler) *Bluetooth*	-	-	✔	✔
TM-P80 Wi-Fi	-	✔	-	✔
TM-P80 *Bluetooth*	-	-	✔	✔
TM-T20	✔	-	-	✔
TM-T20II	✔	-	✔	✔
TM-T70	✔	✔	-	✔
TM-T70II	✔	✔	✔	✔
TM-T81II	✔	-	-	✔
TM-T82	✔	-	-	✔
TM-T82II	✔	-	-	✔
TM-T83II	✔	-	-	✔
TM-T88V	✔	✔	✔	✔
TM-T90II	✔	✔	-	✔
TM-U220 Series	✔	✔	-	✔
TM-U330 Series	✔	✔	-	✔

In the TM printer settings, set only Receive Buffer Full for the Busy Condition.
Regarding the settings, see the Technical Reference Guide for the printer.

Development Environment

The following are necessary to develop an Android application.

- Android SDK r15 or later
- Java Development Kit 6 or later

Contents in the Package

Package

File	Description
ePOS-Print.jar	Compiled Java class file, archived into a jar format file to allow APIs to be used from Java programs.
ePOSEasySelect.jar	A Java class file for selecting a printer easily.
libeposprint.so	Library for function execution. (ARMv5TE supported)
libeposeasyselect.so	Library for the ePOSEasySelect function execution. (ARMv5TE supported)
ePOS-Print_Sample_Android.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_Android_en_revx.pdf	This manual.
ePOS-Print_SDK_Android_ja_revx.pdf	The Japanese-language edition of this manual.
ePOS-Print_SDK_Android_ AppDevGuide_en_revx.pdf	Describes the procedure for building a development environment.
ePOS-Print_SDK_Android_ AppDevGuide_ja_revx.pdf	Describes the procedure for building a development environment. (The Japanese-language edition)

Manual

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

❏ ePOS-Print SDK for Android User's Manual (This Document)

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

Sample Program

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

❏ ePOS-Print_Sample_Android.zip

- Basic function sample (ePOSPrintSample)
- Easy Select sample (ePOSEasySelectSample)
- Receipt print sample (ePOSReceiptPrintSample)

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.57) and command transmission/reception API (p.167) 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.
- When the screen display rotates, Activity may be discarded. To retain a Print instance using Activity, closePrinter of the Print class should be called before Activity is discarded.
- If the device goes into sleep mode while communicating with a printer via *Bluetooth*, the connection will be lost.

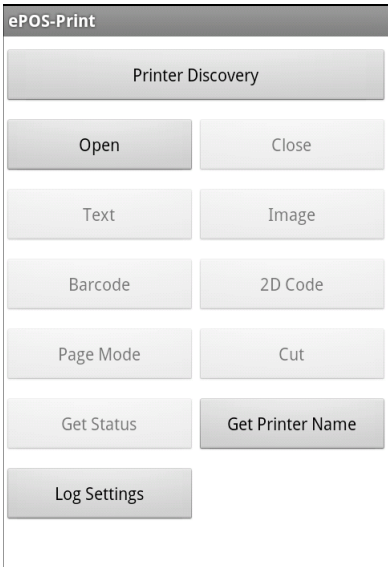
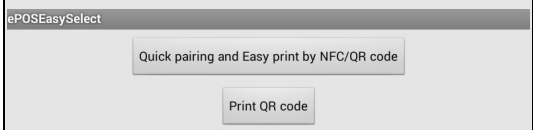
Sample Program

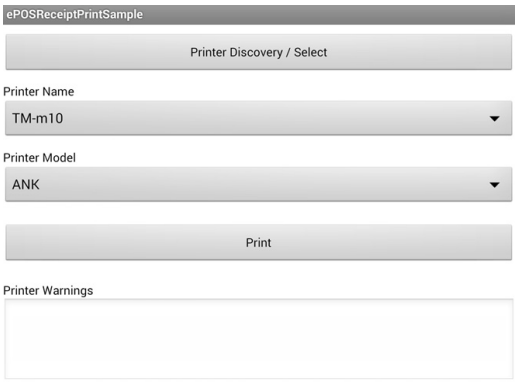
This chapter describes how to use the sample program.

- The sample program is provided as an Android application project for use with Eclipse, including the Java source files.
- For an Android application for TM printers developed using ePOS-Print SDK, the following program is available.

Overview

The Sample Program has the following functionality.

Sample Program	Description
<ePOSPrintSample> 	The following functions are implemented in the sample program: (The sample program does not contain a functionality for turning text / images / barcodes / etc.) • 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
<ePOSEasySelectSample> 	Connects to a printer easily by using NFC or QR code. • Obtain printer information using NFC. • Obtain printer information using QR code. • Analyze the obtained printer information. • Open a port using the results of analysis. • Create printer information QR code from printer search results.

Sample Program	Description
<ePOSReceiptPrintSample>  	Prints the sample receipt data. • Searching for printers • Receipt printing

Usage Environment

Development Environment

- Android SDK r16
- Java Development Kit 6
- Eclipse
- ADT Plugin for Eclipse

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

Printer

- TM printer supported in ePOS-Print SDK.

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. In Eclipse, go to (File)-(Import), select (General)-(Existing Project into Workspace), and then click (Next).

3. The Import Projects window will be displayed. Make the settings shown below and click (Finish).

Item	Setting
Select root directory	Specify the directory where you extracted the sample program zip file.
Copy projects into workspace	Check this option.

4. In the "Package Explorer" View, right-click the "ePOSPrintSample" project, the "ePOSEasySelectSample" or the "ePOSReceiptPrintSample" project, and then select (Properties).

5. "Properties for ePOSPrintSample" or "Properties for ePOSEasySelectSample" or "Properties for ePOSReceiptPrintSample" is displayed. Set the following and click (OK).

Item	Setting
Android	Select the Android OS version of the target device.
Java Build Path	Ensure that a path to ePOS-Print.jar or ePOS-Pairing.jar inside the project copied to the project workspace is set in Libraries.

6. In the "Package Explorer" View, right-click the "ePOSPrintSample" project, the "ePOSEasySelectSample" project or the "ePOSReceiptPrintSample" project, and then select (Run As Android Application).

7. The sample program will be installed to the target Android 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:

❏ ePOSPrintSample

- Search for printers and printing (p.19)
- Acquisition of Printer Model Name (p.26)

❏ ePOSEasySelectSample

- Printer Selection Using NFC or QR Code (p.27)
- QR Code Printing (p.27)

❏ ePOSReceiptPrintSample

- Sample receipt data Printing (p.28)

Search for printers and printing

Use the sample program as follows:

1 Start the sample program. For details, refer to Environmental Construction (p.18).

2 Search for printers. Tap (Printer Discovery) on the main screen.
Select (Device Type) to display a list of IP addresses/Mac addresses/USB device nodes for the printers retrieved on the (Printer List).

3 Tap the printer to use from (Printer List) displayed.

4 Open the printer's port. Tap (Open) on the main screen.
The "Device Type" and "IP Address/ Mac Address/ Device Node" 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.21).
Graphic printing	Tap (Image) on the main screen. For details, refer to Graphic printing (p.21).
Barcode printing	Tap (Barcode) on the main screen. For details, refer to Barcode printing (p.22).
2D-Code printing	Tap (2D Code) on the main screen. For details, refer to 2D-Code printing (p.22).
Printing in page mode	Tap (Page Mode) on the main screen. For details, refer to Printing in page mode (p.23).
Paper cutting	Tap (Cut) on the main screen. For details, refer to Paper cutting (p.23).
Log output setting	Tap (Log Settings) on the main screen. For details, refer to Log output setting (p.23).
Printer status acquisition	Tap (Get Status) on the main screen.

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.24).
- Method (API) execution error
  For details, refer to Method (API) execution error (p.25).

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] is supported by models that support multiple tones. For details, see addImage (p.78).

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.
5. After printing, check the log file.
   For details, refer to setLogSettings (p.163).

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
BUZZER	Buzzer is sounding (compatible devices only)

- Battery Status: The following will be displayed.

String displayed	Description
0xnnnn	Battery status value For details, refer to Battery Status (p.53).

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.167).

Use the following procedure:

1. Start the sample program. For details, refer to Environmental Construction (p.18).
2. Search for printers. Tap (Printer Discovery) on the main screen.
   Select (Device Type) to display a list of IP addresses/ Mac addresses/ Device nodes/ Printer names for the printers retrieved on the (Printer List).
3. Tap the printer to use from (Printer List) displayed.
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.

Printer Selection Using NFC or QR Code

Use the following procedure:

1. Start the sample program. For details, refer to Environmental Construction (p.18).
2. Tap (Quick pairing and Easy print by NFC/QR code) on the main screen.
3. Select a printer by using the following method:
   - Hold the smart device to the NFC device.
   - Read the QR code using the camera.
     Put the QR code inside the red frame in the camera preview.
4. Tap (Print) to print.

QR Code Printing

Use the following procedure:

1. Start the sample program. For details, refer to Environmental Construction (p.18).
2. Tap (Print QR code) on the main screen.
3. Tap (Find) in the QR Code Printing window.
   In (Printer List), the detected printers are displayed in list form.
4. Select the printer you want to use.
5. Tap (Print) to print.

Sample receipt data Printing

Use the following procedure:

1. Start the sample program. For details, refer to Environmental Construction (p.18).
2. Search for printers. Tap (Printer Discovery / Select) on the main screen.
   Select (Interface Type) to display a list of IP addresses/ Mac addresses/ Device nodes/ Printer names for the printers retrieved on the (Printer List).
3. Tap the printer to use from (Printer List) displayed.
4. Print the sample receipt data.
   Select (Printer Name) and (Printer Model), then tap (Print) to print.
5. When printing fails, the action to take will be displayed.
   Depending on the printer status, a message is displayed in (Printer Warnings).

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 Android applications that use ePOS-Print SDK for Android, please refer to the "ePOS-Print SDK for Android Application Development - Setup Guide".

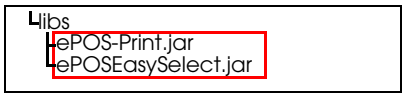
How to Incorporate the ePOS-Print SDK for Android

This section explains how to incorporate the ePOS-Print SDK for Android.

> This explanation uses Eclipse. If you are using another development environment, please make the appropriate changes.

Incorporate the SDK using following procedures.

1. Create a new project in Eclipse.
2. Copy provided JAR file (ePOS-Print.jar and ePOSEasySelect.jar) into following path:



> ePOSEasySelect.jar is required only when Printer Easy Select is used.

3. In Libraries tab of the target project's properties, confirm that the JAR file you added (ePOS-Print.jar) is registered in (Java Build Path).
If it has not been added, add the JAR file into build path using (Add Jars...).
4. Copy the library file (libeposprint.so and libeposeasyselect.so) into following path:

5. Select the project in Eclipse's Package Explorer, right click on it, and press (Refresh).

6 Write the package import declaration in the *.java source file(s) of the application you would like to use this SDK in as follows:

```
import com.epson.eposprint.*;
import com.epson.epsonio.*;
import com.epson.easyselect.*;
```

There is no need to define the easyselect package if Printer Easy Select is not used.

7 Confirm that the target project's "/libs" folder is in the Source tab of the target project's properties. If not, add "libs" to the build path using (Add Folder...).

8 With the target project selected from Eclipse's Package Explorer, select (Preferences) in the (Window) menu.

9 The (Preferences) screen is displayed. From the list on the left, select (Java)-(Compiler).

10 The (Compiler) screen is displayed. Set the (Compiler compliance level:) to "1.6", and click (Apply). After that, click (OK).

11 Double-click (AndroidManifest.xml) from Eclipse's Package Explorer.

12 Select the (Permissions) tab.

13 The (Android Manifest Permissions) screen is displayed. Click the (Add) button.

14 Select (Uses Permission), and click the (OK) button.

15 (Uses Permission) is added to (Permissions). Select the permissions of functions attached to the added (Uses Permission) from the (Name) under (Attributes for Uses Permission).

Functionality	(Name) setting
Wi-Fi	android.permission.INTERNET
Bluetooth	android.permission.*BLUETOOTH*
	android.permission.*BLUETOOTH*_ADMIN
USB	android.permission.USB

There is one setting of permissions for function that can be attached per [Uses Permission] in [Permissions]. For using the *Bluetooth* function and all functions, you must repeat settings from procedures 13 to 15.

16 Save "AndroidManifest.xml".

ePOS-Print SDK

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. Advance Preparations (USB Connection) (p.32) *

2. Printer Selection (p.33) *

<Select printer using search function>
- Selecting a printer using the search function (p.33)
- Getting the printer search result. (p.33)
- Stopping the printer search (p.34)

<Select printer using NFC>
- Selecting a printer using NFC (p.34)

<Select printer using QR code>
- Selecting a printer using QR code (p.35)

3. Print Document Creation (p.36)

- ❏ To create a text print document: (p.36)
- ❏ To create a graphic print document: (p.37)
- ❏ To create a page mode print document (p.38)

4 Transmission of Print Document (p.39)

* 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.41).

Advance Preparations (USB Connection)

For a USB interface, we recommend obtaining access permission to USB devices in the application.

To open a port using the openPrinter method without obtaining access permission to USB devices in advance, note the following:

- When [OK] is pressed in the dialog box for access permission acquisition, it takes about 10 seconds to open a port.
- When [Cancel] is pressed in the dialog box for access permission acquisition, the state of waiting for the timeout lasts for 30 seconds.

How to obtain access permission in the application is as follows:

1 Add the following code to AndroidManifest.xml.

```
<manifest ...>
    <application>
        <activity ...>
            <intent-filter>
                <action android:name
                        ="android.hardware.usb.action.USB_DEVICE_ATTACHED" />
            </intent-filter>
            <meta-data android:name
                        ="android.hardware.usb.action.USB_DEVICE_ATTACHED"
                         android:resource="@xml/device_filter" />
        </activity>
    </application>
</manifest>
```

2 Add res/xml/device_filter.xml to the source file.

3 Write the following code in the device_filter.xml file.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <usb-device vendor-id="1208" />
</resources>
```

4 A dialog box appears when obtaining access permission. Press (OK).

Printer Selection

Selecting a printer using the search function

Use the Finder class's start (p.152) to start searching for printers. Please refer to the following code.

```
int errStatus = IoStatus.SUCCESS;

//Start search
try {

    switch(deviceType){
    //Wi-Fi/Ethernet device
    case DevType.TCP:
        Finder.start(getBaseContext(), DevType.TCP, "255.255.255.255");
        break;
    //Bluetooth device
    case DevType.BLUETOOTH:
        Finder.start(getBaseContext(), DevType.BLUETOOTH, "null");
        break;
    //USB device
    case DevType.USB:
        Finder.start(getBaseContext(), DevType.USB, "null");
        break;
    default:
        Finder.start(getBaseContext(), DevType.TCP, "255.255.255.255");
        break;
    }

//Exception handling
} catch ( EpsonIoException e ) {
    errStatus = e.getStatus();
}
```

Getting the printer search result.

Use the Finder class's getDeviceinfolist (p.154) to get the result of the printer search.
Please refer to the following code. Use the obtained results in openPrinter (p.119).

```
int errStatus = IoStatus.SUCCESS;
DeviceInfo[] mList = null;

//Get device list
try {
    mList = getDeviceInfoList(FilterOption.PARAM_DEFAULT);
//Exception handling
} catch ( EpsonIoException e ) {
    errStatus = e.getStatus();
}
```

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

Stopping the printer search

Use the Finder class's stop (p.153) to stop searching for printers. Please refer to the following code.

```
int errStatus = IoStatus.SUCCESS;

//Stop search
try {
    Finder.stop();
//Exception handling
} catch ( EpsonIoException e ) {
    errStatus = e.getStatus();
}
```

Selecting a printer using NFC

Use parseNFC (p.159) in the EasySelect class to analyze the NFC tag.
Use the programming example below for your reference. Use the obtained results in openPrinter (p.119).

```
@Override
protected void onNewIntent(Intent intent) {

    //Receive NFC tag by onNewIntent.

    EasySelect easySelect = new EasySelect();

    Tag tag = (Tag)intent.getParcelableExtra( NfcAdapter.EXTRA_TAG );

    //Analyze NFC tag
    ArrayList<EasySelectInfo> easySelectInfoArray = null;
    easySelectInfoArray = easySelect.parseNFC( tag, PARSE_NFC_TIMEOUT_DEFAULT );
    if ((null == easySelectInfoArray) || (easySelectInfoArray.size() == 0)){
        //If it is not NFC for EasySelect
        return ;
    }

    //Obtain the first NFC data
    EasySelectInfo easySelectInfo = easySelectInfoArray[0];

    String printerName = easySelectInfo.printerName;
    if(printerName == null || printerName.equals("")){
        printerName = "TM-T88V"  //Name of the printer to use
    }

    String macAddress = easySelectInfo.macAddress;
    if(macAddress == null || macAddress.equals("")){
        //Could not obtain the macAddress
    }

    try {
        Print printer = new Print();

        //Open the printer using analyzed data
        printer.openPrinter(easySelectInfo.deviceType, easySelectInfo.macAddress);

        //Create an Builder class instance by using the analyzed data.
        Builder builder = new Builder(printerName,  Builder.MODEL_ANK);

        ///Printing process///

        printer.closePrinter();

    } catch (EposException e) {
        ///Process///
    }
}
```

This function is used for NFC supported models only.

Selecting a printer using QR code

Use parseQR (p.160) in the EasySelect class to analyze the QR code.
Use the programming example below for your reference. Use the obtained results in openPrinter (p.119).

```
EasySelect easySelect = new EasySelect();
String data;

//Store the QR code data obtained from the camera image.

//Analyze the QR code
EasySelectInfo easySelectInfo = easySelect.parseQR(data);
if (null == easySelectInfo) {
    //If it is not QR code for EasySelect
    return ;
}

try {
    Print printer = new Print();

    //Open the printer using analyzed data
    printer.openPrinter(easySelectInfo.deviceType, easySelectInfo.macAddress);

    //Create an Builder class instance by using the analyzed data.
    Builder builder = new Builder(easySelectInfo.printerName, Builder.MODEL_JAPANESE);

    printer.closePrinter();

//Exception handling
} catch (EposException e) {
    ///Process///
}
```

How to Create Printer Easy Select QR code

❏ For models that can automatically print Printer Easy Select QR code

Use the QR code for dynamic status sheets.
For details on how to print dynamic status sheets, refer to the Technical Reference Guide of each model.

❏ For models that cannot automatically print Printer Easy Select QR code

Create QR code using createQR (p.161).
Refer to QR code creation in the sample program.

Print Document Creation

Create a print document using the Builder class (p.55).
Create a Builder class using the constructor for it and create a print document using APIs of the Builder class. Use the programming example below for your reference.

```
try {
    //Initialize a Builder class instance
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    //Create a print document
    builder.addTextLang(Builder.LANG_EN);
    builder.addTextSmooth(Builder.TRUE);
    builder.addTextFont(Builder.FONT_A);
    builder.addTextSize(3, 3);
    builder.addText("Hello,\t");
    builder.addText("World!\n");
    builder.addCut(Builder.CUT_FEED);
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

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.

Make the language settings based on the language of the characters you are printing.
For details, refer to addTextLang (p.68).

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

```
try {
    //Initialize a Builder class instance
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);

    //Create a print document
    //<Configure the print character settings>
    builder.addTextLang(Builder.LANG_EN);
    builder.addTextSmooth(Builder.TRUE);
    builder.addTextFont(Builder.FONT_A);
    builder.addTextSize(4, 4);
    builder.addTextStyle(Builder.FALSE, Builder.FALSE, Builder.TRUE,
                         Builder.PARAM_UNSPECIFIED);

    //<Specify the print data>
    builder.addText("Hello,\t");
    builder.addText("World!\n");
    builder.addCut(Builder.CUT_FEED);
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

To create a graphic print document:

To create a graphic print document, for graphics, store the android.graphics.Bitmap class in the command buffers with addImage (p.78) of the Builder class.
Use the programming example below for your reference.

```
import android.content.res.Resources;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
try {
     //Initialize a Builder class instance
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);

     //Create a print document
    Bitmap bmp = BitmapFactory.decodeResource(getResources(),R.drawable.background);
    builder.addImage(bmp, 0, 0, 8, 48, Builder.PARAM_DEFAULT);
    builder.addCut(Builder.CUT_FEED);
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

For ways of graphic printing, you can also print the graphics registered in the printer's NV memory. For details, refer to addLogo (p.86).

To create a page mode print document

The page mode starts by storing addPageBegin (p.97) of the Builder class into a command buffer. Store the print area (addPageArea (p.99)) and the print start position (addPagePosition (p.102)) 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.98) in a command buffer. Use the programming example below for your reference.

Make the language settings based on the language of the characters you are printing.
For details, refer to addTextLang (p.68).

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

```
try {
    //Initialize a Builder class instance
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);

    //Create a print document
    //<The page mode starts>
    builder.addPageBegin();
    builder.addPageArea(100, 50, 200, 100);
    builder.addPagePosition(0, 42);
    //<Configure the print character settings>
    builder.addTextLang(Builder.LANG_EN);
    builder.addTextSmooth(Builder.TRUE);
    builder.addTextFont(Builder.FONT_A);
    builder.addTextSize(2, 2);
    builder.addTextStyle(Builder.FALSE, Builder.FALSE, Builder.TRUE,
                         Builder.PARAM_UNSPECIFIED);
    //<Specify the print data>
    builder.addText("Hello,\t");
    builder.addText("World!\n");
    //<The page mode ends>
    builder.addPageEnd();
    builder.addCut(Builder.CUT_FEED);
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

Transmission of Print Document

Send a print document using the Print class (p.57). Create a Print class using the constructor for it, use sendData to specify the Builder class instance that stores the command buffers for the print document, and send the document.
The command buffers stored in the Builder class instance will be retained until clearCommandBuffer (p.63) is executed. Execute clearCommandBuffer after the success of sendData (p.127).

If you want to print the same document repeatedly, you don't have to execute clearCommandBuffer.

Use the programming example below for your reference.

```
//Initialize a Print class instance
Print printer = new Print();

int[] status = new int[1];
status[0] = 0;

try {
    //Initialize a Builder class instance
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);

    //Create a print document
    //<The page mode starts>
    builder.addTextLang(Builder.LANG_EN);
    builder.addTextSmooth(Builder.TRUE);
    builder.addTextFont(Builder.FONT_A);
    builder.addTextSize(4, 4);
    builder.addTextStyle(Builder.FALSE, Builder.FALSE, Builder.TRUE, Builder.PARAM_UNSPECIFIED);

    //<Specify the print data>
    builder.addText("Hello,\t");
    builder.addText("World!\n");
    builder.addCut(Builder.CUT_FEED);

    //Send a print document
    //<Start communication with the printer>
    ////Wi-Fi/Ethernet device
    printer.openPrinter(mList.getDeviceType(), mList.getDeviceName());
    ////USB device
    printer.openPrinter(mList.getDeviceType(), mList.getDeviceName(), "null", Print.TRUE,
                        Print.PARAM_DEFAULT);
    ////USB device
    printer.openPrinter(mList.getDeviceType(), mList.getDeviceName(), getApplicationContext(),
                        Print.TRUE, Print.PARAM_DEFAULT);

    //<Send data>
    printer.sendData(builder, 10000, status);
    //<Delete the command buffers>
    if((status[0] & Print.ST_PRINT_SUCCESS) == Print.ST_PRINT_SUCCESS)
    {
        builder.clearCommandBuffer();
    }
    //<End communication with the printer>
    printer.closePrinter();

} catch (EposException e) {
    int errStatus = e.getErrorStatus();
    status[0] = e.getPrinterStatus();
    printer.closePrinter();
}
```

Effective range of command buffers for setting

The effective range of addXXX in the Builder class instance used for setting is from the time when addXXX is set until sendData is executed. The set value is initialized each time sendData is executed.
Refer to the following:

Example:

```
Print printer = new Print();
Builder builder = null;

Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
builder.addText("Hello, World!\n");          String for which the addTextFont setting is disabled
builder.addTextFont(Builder.FONT_A);
builder.addText("Hello, World!\n");          String for which the addTextFont setting is enabled (FONT_A)
printer.sendData(builder, 10000, status);
builder.addText("Hello, World!\n");          String for which the addTextFont setting is disabled
builder.addTextFont(Builder.FONT_B);
builder.addText("Hello, World!\n");          String for which the addTextFont setting is enabled (FONT_B)
printer.sendData(builder, 10000, status);
```

Printing After Checking the Printer Status

To ensure successful print operation, print after checking the printer status.
Acquire the printer status in getStatus (p.147), and print it out when the printer is online.
Use the programming example below for your reference.

```
//<Send data for confirmation>
Builder builder = null;
Print printer = null;

int retry = 0;

//<Create a print document>                                          (1)
try {
    builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addText("Hello, World!\n");
    builder.addCut(Builder.CUT_FEED);
}
catch (EposException e) {
    builder = null;
    return;
}

for (retry = 0; retry < 3; retry++) {
    int errStatus = 0;
    int[] status = new int[1];
    int[] battery = new int[1];
    status[0] = 0;
    battery[0] = 0;

    try {
        //Initialize an EposPrint class instance
        printer = new Print();

        //Start communication with the printer
        printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168");
    }
    catch (EposException e) {
        printer = null;
        continue;
    }

    try {
        //<Get printer status>                                       (2)
        printer.getStatus(status, battery);

        if ((status[0] & Print.ST_OFF_LINE) != Print.ST_OFF_LINE) {  (3)
            //<Send print data>
            printer.sendData(builder, 10000, status, battery);
        }

        else if ((status[0] & Print.ST_OFF_LINE) == Print.ST_OFF_LINE) {  (4)
            ;
        }
        else {
            ;
        }
    }
    catch (EposException e) {
        errStatus = e.getErrorStatus();
        status[0] = e.getPrinterStatus();
        battery[0] = e.getBatteryStatus();
    }
    finally {
        //<End communication with the printer>
        printer.closePrinter();
        printer = null;
    }

    if (errStatus != EposException.ERR_CONNECT) {
        break;
    }
}
builder.clearCommandBuffer();
builder = null;
```

1. Create print data.
2. Acquire the printer status.
3. When the printer status is online, send the print data you created in step 1.
4. When the printer status is offline, clear the factor that is making the printer status offline. (Such as cover open and no paper.)

Automatic Acquisition of Printer Status

In ePOS-Print SDK, listener is used to automatically notify printer status to the application. Refer to the following.

```
//Registration of StatusChangeEventListener for giving notification of printer status   (1)
public class SampleActivity extends Activity implements OnClickListener,
                                                         StatusChangeEventListener {

  ///Process///

    //Implement the StatusChangeEventListener method                                    (2)/(5)
    private void onStatusChangeEvent(String deviceName, int status) {

      ///Process///
    }

    private void openPrinter() {
        //Initialize the print class instance
        Print printer = new Print();

        //Register the notification destination of printer status changes               (3)
        printer.setStatusChangeEventCallback(this);

        try {
        //Start communications with the printer and monitoring of the printer status   (4)
          printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                              Print.PARAM_DEFAULT);

           ///Process///

        } catch (EposException e) {
          int errStatus = e.getErrorStatus();
          printer.closePrinter();
        }
    }
}
```

1 Define the listener interface for acquiring the printer status.

> Above, StatusChangeEventListener, which notifies printer status at the intervals specified in openPrinter (p.119), is defined.
> ePOS-Print has listener interfaces 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 Listener Interface List (p.44) for the listener interface that can be used with ePOS-Print.

2 Implement the notification destination method when events occur.

3 Register the printer status notification destination.

4 Use openPrinter (p.119) to start monitoring of the printer status.

5 Notify the printer status to the event implemented in (2).

> When printer status notification is ended, it ends on the closePrinter (p.126) of the Print class.

Listener Interface List

For details on the listener interface, refer to API Reference (p.55), which explains the notification destination registration API.

Function	Event listener Notification destination method Notification destination registration API
Printer status notification	public interface StatusChangeEventListener extends EventListener
	void onStatusChangeEvent(String deviceName, int status)
	setStatusChangeEventCallback (p.133)
Online notification	public interface OnlineEventListener extends EventListener
	void onOnlineEvent(String deviceName)
	setOnlineEventCallback (p.134)
Offline notification	public interface OfflineEventListener extends EventListener
	void onOfflineEvent(String deviceName)
	setOfflineEventCallback (p.135)
Power off notification	public interface PowerOffEventListener extends EventListener
	void onPowerOffEvent(String deviceName)
	setPowerOffEventCallback (p.136)
Cover close notification	public interface CoverOkEventListener extends EventListener
	void onCoverOkEvent(String deviceName)
	setCoverOkEventCallback (p.137)
Cover open notification	public interface CoverOpenEventListener extends EventListener
	void onCoverOpenEvent(String deviceName)
	setCoverOpenEventCallback (p.138)
Paper OK notification	public interface PaperOkEventListener extends EventListener
	void onPaperOkEvent(String deviceName)
	setPaperOkEventCallback (p.139)
Paper near end notification	public interface PaperNearEndEventListener extends EventListener
	void onPaperNearEndEvent(String deviceName)
	setPaperNearEndEventCallback (p.140)
Paper end notification	public interface PaperEndEventListener extends EventListener
	void onPaperEndEvent(String deviceName)
	setPaperEndEventCallback (p.141)
Drawer close notification	public interface DrawerClosedEventListener extends EventListener
	void onDrawerClosedEvent(String deviceName)
	setDrawerClosedEventCallback (p.142)
Drawer open notification	public interface DrawerOpenEventListener extends EventListener
	void onDrawerOpenEvent(String deviceName)
	setDrawerOpenEventCallback (p.143)
Battery low notification	public interface BatteryLowEventListener extends EventListener
	void onBatteryLowEvent(String deviceName)
	setBatteryLowEventCallback (p.144)

Function	Event listener
	Notification destination method
	Notification destination registration API
Battery OK notification	public interface BatteryOkEventListener extends EventListener
	void onBatteryOkEvent(String deviceName)
	setBatteryOkEventCallback (p.145)
Battery status notification	public interface BatteryStatusChangeEventListener extends EventListener
	void onBatteryStatusChangeEvent(String deviceName, int battery)
	setBatteryStatusChangeEventCallback (p.146)

Exception handling

In ePOS-Print SDK for Android, it is designed that when an error occurs, a propriety exception with an integer (int) type parameter is generated to notify the calling side of such an error. The ePOS-Print API acquires information using the EposException class (p.58), and the search API acquires information using the EposException class (p.58). The following errors are sent:

Type	Description
Error status	Cause of error when each class's API was executed. For details, refer to Error Statuses and Actions to Take (p.49).
Printer status	Status of the printer when print data was sent. The printer status can be acquired only when sendData (Previous format) (p.129) is executed. For details, refer to Printer Statuses and Actions to Take (p.51).
Battery status	Status of the printer's battery. For details, refer to Battery Status (p.53).

Steps for Handling

ePOS-Print API

Acquire the error status using the EposException class getErrorStatus (p.148), the printer status using getPrinterStatus (p.149), and the battery status using getBatteryStatus (p.150).
Use the programming example below for your reference.

```
//<Send data for confirmation>
Print printer = new Print();

int[] status = new int[1];
int[] battery = new int[1]
status[0] = 0;
battery[0] = 0;

try {
     //Create a print document
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK, getApplicationContext());
    builder.addText("Hello,\t");
    builder.addText("World!\n");
    builder.addCut(Builder.CUT_FEED);

     //<Send print data>
     ////Wi-Fi/Ethernet device
    printer.openPrinter(mList.getDeviceType(), mList.getDeviceName(), Print.TRUE,
                        Print.PARAM_DEFAULT);
     ////Bluetooth device
    printer.openPrinter(Print.DEVTYPE_BLUETOOTH, "00:00:12:34:56:78", Print.TRUE,
                        Print.PARAM_DEFAULT);
     ////USB device
    printer.openPrinter(Print.DEVTYPE_USB, "/dev/bus/usb/001/002", Print.TRUE,
                        Print.PARAM_DEFAULT);
    printer.sendData(builder, 10000, status, battery);
    printer.closePrinter();

} catch (EposException e) {
     //Acquire the error status
    int errStatus = e.getErrorStatus();
     //Acquire the printer status
    status[0] = e.getPrinterStatus();
     //Acquire the battery status
    battery[0] = e.getBatteryStatus();
    printer.closePrinter();
}
```

Search API

Acquire the error status using getStatus (p.157) of the EpsonIoException class.
Use the programming example below for your reference.

```
int errStatus = IoStatus.SUCCESS;
DeviceInfo[] mList = null;

//Acquire a list of devices
try {

////Wi-Fi/Ethernet device
    Finder.start(getBaseContext(), DevType.TCP, "255.255.255.255");
////Bluetooth device
    Finder.start(getBaseContext(), DevType.BLUETOOTH, "null");
////USB device
    Finder.start(getBaseContext(), DevType.USB, "null");

    mList = getDeviceInfoList(FilterOption.PARAM_DEFAULT);
//Exception processing
} catch ( EpsonIoException e ) {
    errStatus = e.getStatus();
}
```

Error Statuses and Actions to Take

Error statuses are defined in each API-executing class.
This section describes the details of error statuses and actions to take, so that error handling can be performed in your application.

Error status	Cause	Action to Take
ERR_PARAM	Invalid parameter was passed. <Example> • An invalid parameter such as null was passed. • A value outside the supported range was specified.	The parameter was specified incorrectly. Check the parameter.
ERR_OPEN	Open processing failed. <Example> • Could not connect to the designated printer. • When attempting to connect via USB, the USB cable was not connected	Check the Android device and the printer. (The printer's power condition, communication condition, etc.)
ERR_CONNECT	Failed to connect to device. <Example> Failed to send the data to the printer.	Execute the closePrinter method and then the openPrinter method to restore the communication between the Android device and the printer. When *Bluetooth* is selected for the interface, the Android OS may automatically establish connection again. Try to acquire the status for 20 seconds. Then, if "ERR_CONNECT" is still received continuously, restore the communication between the Android device and the printer.
ERR_TIMEOUT	The specified timeout time was exceeded. <Example> Could not transmit all the data in the specified time.	Check the timeout time. Set a value for the timeout time longer than the time required for printing.
ERR_MEMORY	Could not allocate the necessary memory for processing.	End the unneeded applications.
ERR_ILLEGAL	Illegal method used. <Example> • When the printer was not opened, an API for sending a command to the printer was called. • When attempting to connect via USB, a constructor with a context parameter was not used	Use the API in a proper way. Refer to Programming Flow (p.31).

Error status	Cause	Action to Take
ERR_PROCESSING	Could not execute process. <Example> Could not execute the process because an identical process is being executed in another thread.	Review the application processing timing so that processes do not overlap each other.
ERR_UNSUPPORTED	An unsupported model name or language specification was specified.	Cannot be used for unsupported models.
ERR_OFF_LINE	The printer is offline.	Eliminate the cause that makes the printer offline. (Such as cover open and no paper.)
ERR_FAILURE	An unspecified error occurred.	• Check the communication settings of the Android device. (Wi-Fi connection setting, *Bluetooth* connection, USB connection setting, etc.) • Check that there is no problem with the execution environment.

Printer Statuses and Actions to Take

Printer statuses vary depending on the TM printer's model.
This section describes the details of printer statuses and actions to take, so that error handling can be performed in your application.

Printer Status	Cause	Action to Take
Print.ST_NO_RESPONSE (0x00000001)	No response from the printer	Check the printer status including the power condition and cable, and the communication status.
Print.ST_PRINT_SUCCESS (0x00000002)	Printing is successfully completed	-
<Other than TM-P20, TM-P60, TM-P60II, TM-P80> Print.ST_DRAWER_KICK (0x00000004)	Status of the 3rd pin of the drawer kick-out connector = "H"	-
<TM-P20, TM-P60, TM-P60II, TM-P80> Print.ST_BATTERY_OFFLINE (0x00000004)	Battery offline status	Charge the battery.
Print.ST_OFF_LINE (0x00000008)	Offline	Eliminate the cause that makes the printer offline. (Such as cover open and no paper.)
Print.ST_COVER_OPEN (0x00000020)	The cover is open	Close the printer's cover.
Print.ST_PAPER_FEED (0x00000040)	Paper is being fed by a paper feed switch operation	-
Print.ST_PANEL_SWITCH (0x00000200)	The paper feed switch is being pressed (ON)	-
Print.ST_MECHANICAL_ERR (0x00000400)	A mechanical error occurred	Eliminate the cause of the error and turn the printer on again.
Print.ST_AUTOCUTTER_ERR (0x00000800)	An autocutter error occurred	Turn the printer off immediately.
Print.ST_UNRECOVER_ERR (0x00002000)	An unrecoverable error occurred	Turn the printer off immediately.
Print.ST_AUTORECOVER_ERR (0x00004000)	An automatically recoverable error occurred	The error status is automatically canceled when the temperature of the head drops as the time passes.
Print.ST_RECEIPT_NEAR_END (0x00020000)	No paper in roll paper near end sensor	Feed paper into the printer.
Print.ST_RECEIPT_END (0x00080000)	No paper in roll paper end sensor	Feed paper into the printer.
Print.ST_BUZZER (0x01000000)	A buzzer is on (only for applicable devices)	-
	Waiting for label to be removed (only for applicable devices)	Remove the label.
Print.ST_HEAD_OVERHEAT * (0x10000000)	The head temperature increased, causing an automatically recoverable error.	The error status is automatically canceled when the temperature of the head drops as time passes.

Printer Status	Cause	Action to Take
Print.ST_MOTOR_OVERHEAT * (0x20000000)	The motor driver IC temperature increased, causing an automatically recoverable error.	The error status is automatically canceled when the temperature of the motor driver IC drops as time passes.
Print.ST_BATTERY_OVERHEAT * (0x40000000)	The battery temperature increased, causing an automatically recoverable error.	The error status is automatically canceled when the temperature of the battery drops as time passes.
Print.ST_WRONG_PAPER * (0x00001000)	The inserted paper is different from the layout settings.	Set the proper paper to match the layout settings in the printer.

* Cannot be acquired using sendData.

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.53).
Lower 8 bits	Battery status exclusive by model For details, refer to Support Information by Printer (p.178).

"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 Android.

ePOS-Print API

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

- ❏ Builder class (p. 55)
- ❏ Print class (p. 57)
- ❏ EposException class (p. 58)

The APIs that you can use and the settings that you can designate vary based on the printer. For details, refer to List of Supported APIs for Each Printer Model (p.177) and Support Information by Printer (p.178).

Builder 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
Constructor		Initialize a Builder class instance.	59
Constructor (Previous format)		Initialize a Builder class instance. (The log output function cannot be used.)	61
Clearing command buffers	clearCommandBuffer	Clears the command buffers added by APIs.	63
Text	addTextAlign	Adds a tag for the text alignment setting.	64
	addTextLineSpace	Adds a tag for the line feed space setting.	65
	addTextRotate	Adds a tag for the text rotation setting.	66
	addText	Adds a tag for printing text.	67
	addTextLang	Adds a tag for the target language setting.	68
	addTextFont	Adds a tag for the text font setting.	69
	addTextSmooth	Adds a tag for the text smoothing setting.	70
	addTextDouble	Adds a tag for specifying the double-sized text setting.	71
	addTextSize	Adds a tag for the text scale setting.	72
	addTextStyle	Adds a tag for the text style setting.	73
	addTextPosition	Adds a tag for specifying the print position of text.	75
Paper Feed	addFeedUnit	Adds a tag for paper feeding (in dots).	76
	addFeedLine	Adds a tag for paper feeding (in lines).	77
	addFeedPosition	Adds a tag for label / black mark paper feeding.	113

API		Description	Page
Graphic	addImage	Adds multiple tone raster image printing to the command buffer. Compresses image data and adds them to the command buffer. (*Bluetooth* interface)	78
	addImage (Previous format)	Adds multiple tone raster image printing to the command buffer. (Image data compression cannot be used (*Bluetooth* interface).)	81
	addImage (Previous format)	Adds a tag for a raster image to be printed. (Image data compression cannot be used (*Bluetooth* interface). Multiple tones cannot be printed.)	84
	addLogo	Adds a tag for an NV logo to be printed.	86
Barcode	addBarcode	Adds a tag for a barcode to be printed.	87
	addSymbol	Adds a tag for a 2D-Code to be printed.	92
Page mode	addPageBegin	Adds a tag for switching to page mode.	97
	addPageEnd	Adds a tag for finishing page mode.	98
	addPageArea	Adds a tag for specifying the print area in page mode.	99
	addPageDirection	Adds a tag for specifying the print direction in page mode.	100
	addPagePosition	Adds a tag for specifying the print position in page mode.	102
	addPageLine	Adds a tag for drawing a line in page mode.	103
	addPageRectangle	Adds a tag for drawing a rectangle in page mode.	105
Cut	addCut	Adds a tag for paper cut.	107
Drawer kick-out	addPulse	Adds a tag for the drawer kick-out.	108
Buzzer	addSound	Adds a tag for turning on the buzzer.	109
	addSound (Previous format)	Adds a tag for turning on the buzzer. (The buzzer sounding cycle cannot be set.)	111
Paper Layout	addLayout	Adds a tag for paper layout information.	114
Send Command	addCommand	Adds a tag for inserting commands.	116

Print class

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

API	Description	Page
Constructor	Initialize a Print class instance. This constructor is for the log output function.	117
Constructor (Previous format)	Initialize a Print class instance. The log output function cannot be used and communications via a USB connection cannot be made.	118
openPrinter	Starts communications with the printer and monitoring of the printer status	119
openPrinter (Previous format)	Starts communications with the printer and monitoring of the printer status (Timeout cannot be set.)	121
openPrinter (Previous format)	Start communication with the printer. (The printer status acquisition and timeout cannot be set.)	124
closePrinter	End communication with the printer.	126
sendData	Sends a command to the printer.	127
sendData (Previous format)	Sends a command to the printer. (The battery status cannot be acquired.)	129
beginTransaction	Starts transaction.	131
endTransaction	Finishes transaction.	132
setStatusChangeEventCallback	Registers the printer status notification destination	133
setOnlineEventCallback	Registers the online event notification destination	134
setOfflineEventCallback	Registers the offline event notification destination	135
setPowerOffEventCallback	Registers the power off event notification destination	136
setCoverOkEventCallback	Registers the cover close event notification destination	137
setCoverOpenEventCallback	Registers the cover open event notification destination	138
setPaperOkEventCallback	Registers the paper OK event notification destination	139
setPaperNearEndEventCallback	Registers the paper near end event notification destination	140
setPaperEndEventCallback	Registers the paper end event notification destination	141
setDrawerClosedEventCallback	Registers the drawer close event notification destination	142
setDrawerOpenEventCallback	Registers the drawer open event notification destination	143
setBatteryLowEventCallback	Registers the battery low event notification destination	144
setBatteryOkEventCallback	Registers the battery OK event notification destination	145
setBatteryStatusChangeEventCallback	Registers the battery status notification destination	146
getStatus	Acquires the printer status and the battery status.	147

EposException class

Acquires the status during slowdown when an exception occurs in printing error or an API execution error occurs.

API	Description	Page
getErrorStatus	Acquires the error status.	148
getPrinterStatus	Acquires the printer status.	149
getBatteryStatus	Acquires the battery status.	150

Builder class (Constructor)

Constructor for the Builder class. Initializes a Builder class instance.
Use this constructor to use the log output function.

Syntax

```
public Builder(String printerModel, int lang,
               Context context) throws EposException
```

Parameter

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

Set value	Description
"TM-m10"	TM-m10
"TM-P20"	TM-P20
"TM-P60"	TM-P60
"TM-P60II"	TM-P60II
"TM-P80"	TM-P80
"TM-T20"	TM-T20
"TM-T20II"	TM-T20II
"TM-T70"	TM-T70
"TM-T70II"	TM-T70II
"TM-T81II"	TM-T81II
"TM-T82"	TM-T82
"TM-T82II"	TM-T82II
"TM-T88V"	TM-T88V
"TM-T90II"	TM-T90II
"TM-U220"	TM-U220

- lang : Specifies the language specifications for the printer.

Set value	Printer model	TM printer-separate setting																
		TM-m10	TM-P20	TM-P60	TM-P60II	TM-P80	TM-T20	TM-T20II	TM-T81II	TM-T82	TM-T82II	TM-T83II	TM-T70	TM-T70II	TM-T88V	TM-T90II	TM-U220	TM-U330
Builder. MODEL_ANK	ANK	✔	✔	✔	✔	✔	✔	✔	-	-	✔	-	✔	✔	✔	-	✔	-
Builder. MODEL_JAPANESE	Japanese	✔	✔	-	-	-	✔	-	-	-	-	-	✔	✔	✔	✔	✔	-
Builder. MODEL_CHINESE	Simplified Chinese	-	✔	-	-	-	-	-	✔	-	✔	-	✔	✔	✔	-	✔	✔
Builder. MODEL_TAIWAN	Traditional Chinese	✔	✔	-	✔	✔	-	-	-	-	✔	-	✔	✔	✔	-	✔	-
Builder. MODEL_KOREAN	Korean	-	-	-	-	-	-	-	-	-	-	✔	-	✔	✔	-	✔	-
Builder. MODEL_THAI	Thai	-	-	-	-	-	-	-	-	✔	✔	-	✔	✔	✔	-	✔	-
Builder. MODEL_SOUTHASIA	South Asian	-	✔	-	-	-	-	-	-	✔	✔	-	✔	✔	✔	-	✔	-

- context : Specifies the context of the application.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_UNSUPPORTED	An unsupported model name or unsupported language specifications were specified.
ERR_FAILURE	An unspecified error occurred.

Example

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

```
import android.content.Context;

try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK,
                                  getApplicationContext());
  ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

Builder class (Constructor) (Previous format)

Constructor for the Builder class. Initializes a Builder class instance.

The log output function cannot be used.
To use the log output function, use the Builder class (Constructor) (p.59).

Syntax

```
public Builder(String printerModel, int lang)
 throws EposException
```

Parameter

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

Set value	Description
"TM-m10"	TM-m10
"TM-P20"	TM-P20
"TM-P60"	TM-P60
"TM-P60II"	TM-P60II
"TM-P80"	TM-P80
"TM-T20"	TM-T20
"TM-T20II"	TM-T20II
"TM-T70"	TM-T70
"TM-T70II"	TM-T70II
"TM-T81II"	TM-T81II
"TM-T82"	TM-T82
"TM-T82II"	TM-T82II
"TM-T88V"	TM-T88V
"TM-T90II"	TM-T90II
"TM-U220"	TM-U220

- lang : Specifies the language specifications for the printer.

Set value	Printer model	TM printer-separate setting																
		TM-m10	TM-P20	TM-P60	TM-P60II	TM-P80	TM-T20	TM-T20II	TM-T70	TM-T70II	TM-T81II	TM-T82	TM-T82II	TM-T83II	TM-T88V	TM-T90II	TM-U220	TM-U330
Builder. MODEL_ANK	ANK	✔	✔	✔	✔	✔	✔	✔	✔	✔	-	-	✔	-	✔	-	✔	-
Builder. MODEL_JAPANESE	Japanese	✔	✔	-	-	-	✔	-	✔	✔	-	-	-	-	✔	✔	✔	-
Builder. MODEL_CHINESE	Simplified Chinese	-	✔	-	-	-	-	-	✔	✔	✔	-	✔	-	✔	-	✔	✔
Builder. MODEL_TAIWAN	Traditional Chinese	✔	✔	-	✔	✔	-	-	✔	✔	-	-	✔	-	✔	-	✔	-
Builder. MODEL_KOREAN	Korean	-	-	-	-	-	-	-	-	✔	-	-	-	✔	✔	-	✔	-
Builder. MODEL_THAI	Thai	-	-	-	-	-	-	-	✔	✔	-	✔	✔	-	✔	-	✔	-
Builder. MODEL_SOUTHASIA	South Asian	-	✔	-	-	-	-	-	✔	✔	-	✔	✔	-	✔	-	✔	-

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_UNSUPPORTED	An unsupported model name or unsupported language specifications were specified.
ERR_FAILURE	An unspecified error occurred.

Example

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

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
  ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

clearCommandBuffer

Clears command buffers used by APIs of the Builder class.
The command buffers stored in the Builder class instance will be retained until this API is executed.

Syntax

```
public void clearCommandBuffer()
```

Example

If you are clearing the command buffer:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    ///Process///
    builder.clearCommandBuffer();
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

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, use addPagePosition (p.102) instead of this API to set the alignment.

Syntax

```
public void addTextAlign(int align) throws EposException
```

Parameter

- align : Specifies the text alignment.

Set value	Description
Builder.ALIGN_LEFT (default)	Alignment to the left
Builder.ALIGN_CENTER	Alignment to the center
Builder.ALIGN_RIGHT	Alignment to the right

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To set alignment to the center:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addTextAlign(Builder.ALIGN_CENTER);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addTextLineSpace

Adds the line feed space setting to the command buffer.

Syntax

```
public void addTextLineSpace(int linespc)
 throws EposException
```

Parameter

- linespc : Specifies the line feed space (in dots). Specifies an integer from 0 to 255. (Default value: Refer to Support Information by Printer (p.178).)

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To set the line feed space to 30 dots:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addTextLineSpace(30);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

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.100) instead of this API function.

Syntax

```
public void addTextRotate(int rotate)
 throws EposException
```

Parameter

- rotate : Specifies whether to rotate text.

Set value	Description
Builder.TRUE	Specifies rotated printing of text.
Builder.FALSE (default)	Cancels rotated printing of text.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To set text rotation:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addTextRotate(Builder.TRUE);
     ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

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.
(Example: After printing text, an attempt was made to perform graphic printing, but nothing was printed.)

Syntax

```
public void addText(String data) throws EposException
```

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

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To add character strings:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addText("Hello,\t");
    builder.addText("World!\n");
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addTextLang

Adds the language setting to a command buffer. Encodes the string specified by addText (p.67) according to the language information specified by this API.
Specify the value according to the language specifications set for Builder class (Constructor) (p.59).

Syntax

```
public void addTextLang(int lang) throws EposException
```

Parameter

- lang : Specifies the target language.

Set value	Language
Builder.LANG_EN(default)	English(ANK)
Builder.LANG_JA	Japanese
Builder.LANG_ZH_CN	Simplified Chinese
Builder.LANG_ZH_TW	Traditional Chinese
Builder.LANG_KO	Korean
Builder.LANG_TH	Thai (South Asia specifications)
Builder.LANG_VI	Vietnamese (South Asia specifications)

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To set the language as English:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addTextLang(Builder.LANG_EN);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addTextFont

Adds the text font setting to the command buffer.

Syntax

```
public void addTextFont(int font) throws EposException
```

Parameter

- font : Specifies the font.

Set value	Description	TM printer-separate setting												
		TM-m10	TM-P20	TM-P60/TM-P60II	TM-P80	TM-T20/TM-T20II	TM-T70/TM-T70II	TM-T81II	TM-T82/TM-T82II	TM-T83II	TM-T88V	TM-T90II	TM-U220	TM-U330
Builder.FONT_A (default)	Font A	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Builder.FONT_B	Font B	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Builder.FONT_C	Font C	✔	✔	✔	-	-	-	-	-	-	-	✔	-	-
Builder.FONT_D	Font D	-	✔	-	-	-	-	-	-	-	-	-	-	-
Builder.FONT_E	Font E	-	✔	-	-	-	-	-	-	-	-	-	-	-

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To set the font B:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addTextFont(Builder.FONT_B);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addTextSmooth

Adds the smoothing setting to the command buffer.

Syntax

```
public void addTextSmooth(int smooth)
 throws EposException
```

Parameter

- smooth : Specifies whether to enable smoothing.

Set value	Description
Builder.TRUE	Specifies smoothing.
Builder.FALSE (default)	Cancels smoothing

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To enable smoothing:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addTextSmooth(Builder.TRUE);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addTextDouble

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

Syntax

```
public void addTextDouble(int dw, int dh)
 throws EposException
```

Parameter

- dw : Specifies the double-sized width.

Set value	Description
Builder.TRUE	Specifies the double-sized width.
Builder.FALSE (default)	Cancels the double-sized width
Builder.PARAM_UNSPECIFIED	Retains the current setting.

- dh : Specifies the double-sized height.

Set value	Description
Builder.TRUE	Specifies the double-sized height
Builder.FALSE (default)	Cancels the double-sized height
Builder.PARAM_UNSPECIFIED	Retains the current setting.

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

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To set the size as double width and height:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addTextDouble(Builder.TRUE, Builder.TRUE);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addTextSize

Adds the text scale setting to the command buffer.

Syntax

```
public void addTextSize(int width, int height)
 throws EposException
```

Parameter

- width : Specifies the horizontal scale of text.

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

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

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

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addTextSize(4, 4);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addTextStyle

Adds the text style setting to the command buffer.

Syntax

```
public void addTextStyle(int reverse, int ul, int em
, int color) throws EposException
```

Parameter

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

Set value	Description
Builder.TRUE	Specifies the inversion of black and white parts of characters.
Builder.FALSE (default)	Cancels the inversion of black and white parts of characters.
Builder.PARAM_UNSPECIFIED	Retains the current setting.

- ul : Specifies the underline style.

Set value	Description
Builder.TRUE	Specifies underlining.
Builder.FALSE (default)	Cancels underlining.
Builder.PARAM_UNSPECIFIED	Retains the current setting.

- em : Specifies the bold style.

Set value	Description
Builder.TRUE	Specifies emphasized printing of characters.
Builder.FALSE (default)	Cancels emphasized printing of characters.
Builder.PARAM_UNSPECIFIED	Retains the current setting.

- color : Specifies the color.

Set value	Description
Builder.COLOR_NONE	Characters are not printed.
Builder.COLOR_1 (default)	First color
Builder.PARAM_UNSPECIFIED	Retains the current color setting

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To set the underline style:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addTextStyle(Builder.PARAM_UNSPECIFIED, Builder.TRUE,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addTextPosition

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

After executing this API, you cannot use addTextAlign (p.64) or addTextRotate (p.66).

Syntax

```
public void addTextPosition(int x) throws EposException
```

Parameter

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

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

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

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addTextPosition(120);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addFeedUnit

Adds paper feeding in dots to the command buffer.

Syntax

```
public void addFeedUnit(int unit) throws EposException
```

Parameter

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

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To feed paper by 30 dots:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addFeedUnit(30);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addFeedLine

Adds paper feeding in lines to the command buffer.

Syntax

```
public void addFeedLine(int line) throws EposException
```

Parameter

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

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To feed paper by 3 lines:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addFeedLine(3);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addImage

Adds raster image printing to the command buffer.
Prints the graphic in the android.graphics.Bitmap class.
Out of the android.graphics.Bitmap class graphics, the specified scope is converted to raster image data according to this API setting. 1 pixel of the image corresponds to 1 dot of the printer. If transparent shading is included, it is regarded as white.

- Set image compression only for a *Bluetooth* interface.
- To print a raster image at high speed, specify Builder.ALIGN_LEFT for the addTextAlign (p.64), and specify a multiple of 8 not exceeding the printer's paper width for the width parameter of this API.
- When printing transmission images, the printing speed may become slower.
- Multiple tone printing is not supported in Page Mode. Multiple tone graphic printing is supported in Standard Mode only.
- Image compression is not supported in Page Mode.

Syntax

```
public void addImage(Bitmap data, int x, int y
                   , int width, int height, int color
                   , int mode, int halftone
                   , double brightness, int compress)
                    throws EposException
```

Parameter

- data : Specifies an instance of the android.graphics.Bitmap 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.

If the area specified by the x/y parameters and the width/height parameters extends beyond the image size specified by the data parameter, an EposException with ERR_PARAM contained in its error status occurs.

- color : Specifies the color.

Set value	Description
Builder.COLOR_NONE	Characters are not printed.
Builder.COLOR_1	First color
Builder.PARAM_DEFAULT	First color

- mode : Specify the color mode.

Set value	Description	TM printer-separate setting												
		TM-m10	TM-P20	TM-P60/TM-P60II	TM-P80	TM-T20/TM-T20II	TM-T70	TM-T70II	TM-T81II	TM-T82/TM-T82II	TM-T83II	TM-T88V	TM-T90II	TM-U220/TM-U330
Builder.MODE_MONO	Monochrome (2 tone)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Builder.MODE_GRAY16	Multiple tone (16 tone)	✔	-	-	-	-	-	✔	-	-	-	✔	✔	-
Builder.PARAM_DEFAULT	Specify the half tone treatment method. (Monochrome (2 tone))	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔

- halftone : Specify the half tone treatment method.

Set value	Description
Builder.HALFTONE_DITHER	Dither (This is suitable for graphic printing).
Builder.HALFTONE_ERROR_DIFFUSION	Error diffusion (This is suitable for mixed printing or characters and graphics).
Builder.HALFTONE_THRESHOLD	Threshold value (This is suitable for printing of characters).
Builder.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)
Builder.PARAM_DEFAULT	Select the default value (1.0)

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

- compress : Specifies image compression.
  Specify Builder.COMPRESS_DEFLATE only when *Bluetooth* is selected for the interface.

Set value	Description	TM printer-separate setting													
		TM-m10	TM-P20	TM-P60/TM-P60II	TM-P80	TM-T20	TM-T20II	TM-T70	TM-T70II	TM-T81II	TM-T82/TM-T82II	TM-T83II	TM-T88V	TM-T90II	TM-U220/TM-U330
Builder.COMPRESS_DEFLATE	Image compression is carried out	✔	✔	-	-	-	✔	-	✔	-	-	-	✔	-	-
Builder.COMPRESS_NONE	Image compression is not carried out	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Builder.PARAM_DEFAULT	Specify the half tone treatment method. (Image compression is not carried out)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔

For TCP or USB communication, specify Builder.PARAM_DEFAULT.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

```
try {
    Bitmap imageData = null;
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    ///Process///
    builder.addImage(imageData, 0, 0, 256, 256, Builder.PARAM_DEFAULT,
                     Builder.MODE_MONO, Builder.HALFTONE_DITHER, 1.0,
                     Builder.PARAM_DEFAULT);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addImage (Previous format)

Adds raster image printing to the command buffer. When the *Bluetooth* interface is used, white streaks may appear because printing by image data compression is not possible.
Prints the graphic in the android.graphics.Bitmap class.
Out of the android.graphics.Bitmap class graphics, the specified scope is converted to raster image data according to this API setting. 1 pixel of the image corresponds to 1 dot of the printer. If transparent shading is included, it is regarded as white.

- To print a raster image at high speed, specify Builder.ALIGN_LEFT for the addTextAlign (p.64), and specify a multiple of 8 not exceeding the printer's paper width for the width parameter of this API.
- When printing transmission images, the printing speed may become slower.
- Multiple tone printing is not supported in Page Mode. Multiple tone graphic printing is supported in Standard Mode only.

Syntax

```
public void addImage(Bitmap data, int x, int y
                   , int width, int height, int color
                   , int mode, int halftone
                   , double brightness)
                    throws EposException
```

Parameter

- data : Specifies an instance of the android.graphics.Bitmap 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.

If the area specified by the x/y parameters and the width/height parameters extends beyond the image size specified by the data parameter, an EposException with ERR_PARAM contained in its error status occurs.

- color : Specifies the color.

Set value	Description
Builder.COLOR_NONE	Characters are not printed.
Builder.COLOR_1	First color
Builder.PARAM_DEFAULT	First color

- mode : Specify the color mode.

Set value	Description	TM printer-separate setting												
		TM-m10	TM-P20	TM-P60/TM-P60II	TM-P80	TM-T20/TM-T20II	TM-T70	TM-T70II	TM-T81II	TM-T82/TM-T82II	TM-T83II	TM-T88V	TM-T90II	TM-U220/TM-U330
Builder.MODE_MONO	Monochrome (2 tone)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Builder.MODE_GRAY16	Multiple tone (16 tone)	✔	-	-	-	-	-	✔	-	-	-	✔	✔	-
Builder.PARAM_DEFAULT	Specify the half tone treatment method. (Monochrome (2 tone))	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔

- halftone : Specify the half tone treatment method.

Set value	Description
Builder.HALFTONE_DITHER	Dither (This is suitable for graphic printing).
Builder.HALFTONE_ERROR_DIFFUSION	Error diffusion (This is suitable for mixed printing or characters and graphics).
Builder.HALFTONE_THRESHOLD	Threshold value (This is suitable for printing of characters).
Builder.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)
Builder.PARAM_DEFAULT	Select the default value (1.0)

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

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

```
try {
    Bitmap imageData = null;
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    ///Process///
    builder.addImage(imageData, 0, 0, 256, 256, Builder.PARAM_DEFAULT,
                     Builder.MODE_MONO, Builder.HALFTONE_DITHER, 1.0);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

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

```
try {
    Bitmap imageData = null;
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    ///Process///
    builder.addPageBegin();
    builder.addPagePosition(0, 255);
    builder.addImage(imageData, 0, 0, 256, 256, Builder.PARAM_DEFAULT,
                     Builder.MODE_MONO, Builder.HALFTONE_DITHER, 1.0);
    builder.addPageEnd();
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addImage (Previous format)

Adds raster image printing to the command buffer. Multiple tones cannot be printed.
When a *Bluetooth* device is connected, white streaks may appear because printing by image data compression is not possible.
Prints the graphic in the android.graphics.Bitmap class.
Of the graphics in the android.graphics.Bitmap class, makes the specified range into binary with the dither processing, and converts it into raster image data. 1 pixel of the image corresponds to 1 dot of the printer. If transparent shading is included, it is regarded as white.

- When printing in multiple tone, use addImage (p.78).
- To print a raster image at high speed, specify Builder.ALIGN_LEFT for the addTextAlign (p.64), and specify a multiple of 8 not exceeding the printer's paper width for the width parameter of this API.
- When printing transmission images, the printing speed may become slower.

Syntax

```
public void addImage(Bitmap data, int x, int y
                     , int width, int height, int color)
                      throws EposException
```

Parameter

- data : Specifies an instance of the android.graphics.Bitmap 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
Builder.COLOR_NONE	Characters are not printed.
Builder.COLOR_1	First color
Builder.PARAM_DEFAULT	First color

If the area specified by the x/y parameters and the width/height parameters extends beyond the image size specified by the data parameter, an EposException with ERR_PARAM contained in its error status occurs.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

```
try {
    Bitmap imageData = null;
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    ///Process///
    builder.addImage(imageData, 0, 0, 256, 256, Builder.PARAM_DEFAULT);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

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

```
try {
    Bitmap imageData = null;
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    ///Process///
    builder.addPageBegin();
    builder.addPagePosition(0, 255);
    builder.addImage(imageData, 0, 0, 256, 256, Builder.PARAM_DEFAULT);
    builder.addPageEnd();
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

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
- Multiple tone printing is not supported in Page Mode. Multiple tone graphic printing is supported in Standard Mode only.

Syntax

```
public void addLogo(int key1, int key2)
 throws EposException
```

Parameter

- key1 : Specifies the key code 1 of an NV logo. Specifies an integer from 0 to 255.
- key2 : Specifies the key code 2 of an NV logo. Specifies an integer from 0 to 255.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

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

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addLogo(48, 48);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addBarcode

Adds barcode printing to the command buffer.

Syntax

```
public void addBarcode
(String data, int type, int hri, int font, int width
, int height) throws EposException
```

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.
CODE93	Start and stop characters are automatically added. A check digit is automatically calculated and added.

Barcode type	Description
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, a check digit, and a stop character are automatically added. FNC1 is automatically added to the start of the data. It is not added half way through the data. 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 GS1 DataBar Truncated GS1 DataBar Limited	Specify a 13-digit global trade item number (GTIN) not including an application identifier (AI) or a check digit.

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

- hri : Specifies the HRI position.

Set value	Description
Builder.HRI_NONE(default)	HRI not printed
Builder.HRI_ABOVE	Above the barcode
Builder.HRI_BELOW	Below the barcode
Builder.HRI_BOTH	Both above and below the barcode
Builder.PARAM_UNSPECIFIED	Retains the current setting.

- font : Specifies the HRI font.

Set value	Description
Builder.FONT_A(default)	Font A
Builder.FONT_B	Font B
Builder.FONT_C	Font C
Builder.FONT_D	Font D
Builder.FONT_E	Font E
Builder.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 1 to 6	The width of each module. (Unit:dot)
Builder.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)
Builder.PARAM_UNSPECIFIED	Retains the current setting.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To print barcodes:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addBarcode("01234567890", Builder.BARCODE_UPC_A,
            Builder.HRI_BELOW, Builder.PARAM_UNSPECIFIED, 2, 64);
    builder.addBarcode("01234500005", Builder.BARCODE_UPC_E,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addBarcode("201234567890", Builder.BARCODE_EAN13,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addBarcode("201234567890", Builder.BARCODE_JAN13,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addBarcode("2012345", Builder.BARCODE_EAN8,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addBarcode("2012345", Builder.BARCODE_JAN8,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addBarcode("ABCDE", Builder.BARCODE_CODE39,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addBarcode("012345", Builder.BARCODE_ITF,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addBarcode("A012345A", Builder.BARCODE_CODABAR,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addBarcode("ABCDE", Builder.BARCODE_CODE93,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addBarcode("{Babcde", Builder.BARCODE_CODE128,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addBarcode("(01)201234567890*", Builder.BARCODE_GS1_128,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addBarcode("0201234567890",
            Builder.BARCODE_GS1_DATABAR_OMNIDIRECTIONAL,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addBarcode("0201234567890",
            Builder.BARCODE_GS1_DATABAR_TRUNCATED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addBarcode("0201234567890",
            Builder.BARCODE_GS1_DATABAR_LIMITED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED);
    builder.addBarcode("(01)2012345678903",
            Builder.BARCODE_GS1_DATABAR_EXPANDED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addSymbol

Adds 2D-Code printing to the command buffer.

Syntax

```
public void addSymbol
(String data, int type, int level, int width,
 int height, int size)
 throws EposException
```

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 MaxiCode Mode 3 MaxiCode Mode 4 MaxiCode Mode 5 MaxiCode Mode 6	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)
GS1 DataBar Stacked GS1 DataBar Stacked Omnidirectional	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 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.
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.

2D-Code type	Description
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 to 144 lines x 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
Builder.SYMBOL_PDF417_STANDARD	Standard PDF417
Builder.SYMBOL_PDF417_TRUNCATED	Truncated PDF417
Builder.SYMBOL_QRCODE_MODEL_1	QR Code Model 1
Builder.SYMBOL_QRCODE_MODEL_2	QR Code Model 2
Builder.SYMBOL_MAXICODE_MODE_2	MaxiCode Mode 2
Builder.SYMBOL_MAXICODE_MODE_3	MaxiCode Mode 3
Builder.SYMBOL_MAXICODE_MODE_4	MaxiCode Mode 4
Builder.SYMBOL_MAXICODE_MODE_5	MaxiCode Mode 5
Builder.SYMBOL_MAXICODE_MODE_6	MaxiCode Mode 6
Builder.SYMBOL_GS1_DATABAR_STACKED	GS1 DataBar Stacked
Builder.SYMBOL_GS1_DATABAR_STACKED_OMNIDIRECTIONAL	GS1 DataBar Stacked Omnidirectional
Builder.SYMBOL_GS1_DATABAR_EXPANDED_STACKED	GS1 DataBar Expanded Stacked
Builder.SYMBOL_AZTECCODE_FULLRANGE	Aztec Code Full-Range mode
Builder.SYMBOL_AZTECCODE_COMPACT	Aztec Code Compact mode
Builder.SYMBOL_DATAMATRIX_SQUARE	DataMatrix square
Builder.SYMBOL_DATAMATRIX_RECTANGLE_8	DataMatrix rectangle, 8 lines
Builder.SYMBOL_DATAMATRIX_RECTANGLE_12	DataMatrix rectangle, 12 lines
Builder.SYMBOL_DATAMATRIX_RECTANGLE_16	DataMatrix rectangle, 16 lines

- level : Specifies the error correction level.

Set value	Description
Builder.LEVEL_0	PDF417 error correction level 0
Builder.LEVEL_1	PDF417 error correction level 1
Builder.LEVEL_2	PDF417 error correction level 2
Builder.LEVEL_3	PDF417 error correction level 3
Builder.LEVEL_4	PDF417 error correction level 4
Builder.LEVEL_5	PDF417 error correction level 5
Builder.LEVEL_6	PDF417 error correction level 6
Builder.LEVEL_7	PDF417 error correction level 7
Builder.LEVEL_8	PDF417 error correction level 8
Builder.LEVEL_L	QR Code error correction level L
Builder.LEVEL_M	QR Code error correction level M
Builder.LEVEL_Q	QR Code error correction level Q
Builder.LEVEL_H	QR Code error correction level H
Builder.LEVEL_DEFAULT	Default level
5 to 95 integer	Aztec Code error correction level (percent unit)
Builder.PARAM_UNSPECIFIED	Retains the current setting.

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

- width : Specifies the module width.

Set value	Description
Integer from 0 to 255	Module width
Builder.PARAM_UNSPECIFIED	Retains the current setting.

MaxiCode is ignored.

- height : Specifies the module height.

Set value	Description
Integer from 0 to 255	Module height
Builder.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
Builder.PARAM_UNSPECIFIED	Retains the current setting.

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

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To print 2D-Code:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addSymbol("ABCDE", Builder.SYMBOL_PDF417_STANDARD,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addSymbol("ABCDE", Builder.SYMBOL_QRCODE_MODEL_2,
            Builder.LEVEL_Q, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addSymbol("908063840\\x1d850\\x1d001\\x1d\\x04",
            Builder.SYMBOL_MAXICODE_MODE_2, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED);
    builder.addSymbol("0201234567890", Builder.SYMBOL_GS1_DATABAR_STACKED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addSymbol("0201234567890",
            Builder.SYMBOL_GS1_DATABAR_STACKED_OMNIDIRECTIONAL,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.addSymbol("(01)02012345678903",
            Builder.SYMBOL_GS1_DATABAR_EXPANDED_STACKED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addPageBegin

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

Use this API function with addPageEnd (p.98).

Syntax

```
public void addPageBegin() throws EposException
```

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To print the characters "ABCDE" in page mode:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addPageBegin();
    builder.addText("ABCDE");
    builder.addPageEnd();
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addPageEnd

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

Use this API function with addPageBegin (p.97).

Syntax

```
public void addPageEnd() throws EposException
```

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To print the characters "ABCDE" in page mode:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addPageBegin();
    builder.addText("ABCDE");
    builder.addPageEnd();
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

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.97) and addPageEnd (p.98).

Syntax

```
public void addPageArea(int x, int y, int width,
                        int height) throws EposException
```

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 0 to 65535.
- height : Specifies the height of the print area (in dots). Specifies an integer from 0 to 65535.

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

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
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":

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addPageBegin();
    builder.addPageArea(100, 50, 200, 30);
    builder.addText("ABCDE");
    builder.addPageEnd();
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

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.97) and addPageEnd (p.98).

Syntax

```
public void addPageDirection(int dir)
 throws EposException
```

Parameter

- dir : Specifies the print direction in page mode.

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

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

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

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addPageBegin();
    builder.addPageArea(100, 50, 30, 200);
    builder.addPageDirection(Builder.DIRECTION_TOP_TO_BOTTOM);
    builder.addText("ABCDE");
    builder.addPageEnd();
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

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.97) and addPageEnd (p.98).

Syntax

```
public void addPagePosition(int x, int y)
 throws EposException
```

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.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
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":

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addPageBegin();
    builder.addPageArea(100, 50, 200, 100);
    builder.addPagePosition(50, 30);
    builder.addText("ABCDE");
    builder.addPageEnd();
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

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.97) and addPageEnd (p.98).

Syntax

```
public void addPageLine
(int x1, int y1, int x2, int y2, int style)
 throws EposException
```

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
Builder.LINE_THIN	Solid line: Thin
Builder.LINE_MEDIUM	Solid line: Medium
Builder.LINE_THICK	Solid line: Thick
Builder.LINE_THIN_DOUBLE	Double line: Thin
Builder.LINE_MEDIUM_DOUBLE	Double line: Medium
Builder.LINE_THICK_DOUBLE	Double line: Thick
Builder.PARAM_DEFAULT	Solid line: Thin

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
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):

```
try {
    Builder builder = new Builder("TM-P60", Builder.MODEL_ANK);
    builder.addPageBegin();
    builder.addPageLine(100, 0, 500, 0, Builder.LINE_THIN);
    builder.addPageEnd();
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

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.97) and addPageEnd (p.98).

Syntax

```
public void addPageRectangle
(int x1, int y1, int x2, int y2, int style)
 throws EposException
```

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
Builder.LINE_THIN	Solid line: Thin
Builder.LINE_MEDIUM	Solid line: Medium
Builder.LINE_THICK	Solid line: Thick
Builder.LINE_THIN_DOUBLE	Double line: Thin
Builder.LINE_MEDIUM_DOUBLE	Double line: Medium
Builder.LINE_THICK_DOUBLE	Double line: Thick
Builder.PARAM_DEFAULT	Solid line: Thin

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

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

```
try {
    Builder builder = new Builder("TM-P60", Builder.MODEL_ANK);
    builder.addPageBegin();
    builder.addPageRectangle(100, 0, 500, 200, Builder.LINE_THIN);
    builder.addPageEnd();
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addCut

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

Not available in page mode.

Syntax

```
public void addCut(int type) throws EposException
```

Parameter

- type : Specifies the paper cut type.

Set value	Description
Builder.CUT_NO_FEED	Cut without feeding (The paper is cut without being fed.))
Builder.CUT_FEED	Feed cut (The paper is fed to the cut position and then is cut.)
Builder.CUT_RESERVE	Cut reservation (Printing continues until the cut position is reached, at which the paper is cut.)
Builder.PARAM_DEFAULT	Feed cut (The paper is fed to the cut position and then is cut.)

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To perform feed cut operation:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addCut(Builder.CUT_FEED);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

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

```
public void addPulse(int drawer, int time)
 throws EposException
```

Parameter

- drawer : Specifies the drawer kick connector.

Set value	Description
Builder.DRAWER_1	Pin 2 of the drawer kick-out connector
Builder.DRAWER_2	Pin 5 of the drawer kick-out connector
Builder.PARAM_DEFAULT	Pin 2 of the drawer kick-out connector

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

Set value	Description
Builder.PULSE_100	100 ms
Builder.PULSE_200	200 ms
Builder.PULSE_300	300 ms
Builder.PULSE_400	400 ms
Builder.PULSE_500	500 ms
Builder.PARAM_DEFAULT	100 ms

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

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

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addPulse(Builder.DRAWER_1, Builder.PULSE_100);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addSound

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

```
public void addSound(int pattern, int repeat, int cycle)
 throws EposException
```

Parameter

- pattern : Specifies the buzzer pattern.

Set value	Description
Builder.PATTERN_A	Pattern A (Optional Buzzer)
Builder.PATTERN_B	Pattern B (Optional Buzzer)
Builder.PATTERN_C	Pattern C (Optional Buzzer)
Builder.PATTERN_D	Pattern D (Optional Buzzer)
Builder.PATTERN_E	Pattern E (Optional Buzzer)
Builder.PATTERN_ERROR	Error sound pattern (Optional Buzzer)
Builder.PATTERN_PAPER_END	Pattern when there is no paper (Optional Buzzer)
Builder.PATTERN_1	Pattern 1 (Internal Buzzer)
Builder.PATTERN_2	Pattern 2 (Internal Buzzer)
Builder.PATTERN_3	Pattern 3 (Internal Buzzer)
Builder.PATTERN_4	Pattern 4 (Internal Buzzer)
Builder.PATTERN_5	Pattern 5 (Internal Buzzer)
Builder.PATTERN_6	Pattern 6 (Internal Buzzer)
Builder.PATTERN_7	Pattern 7 (Internal Buzzer)
Builder.PATTERN_8	Pattern 8 (Internal Buzzer)
Builder.PATTERN_9	Pattern 9 (Internal Buzzer)
Builder.PATTERN_10	Pattern 10 (Internal Buzzer)
Builder.PARAM_DEFAULT	Pattern A

- repeat : Specifies the number of repeats.

Set value	Description
1 to 255	Number of repeats
Builder.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
Builder.PARAM_DEFAULT	1000 milliseconds

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

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

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

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addSound(Builder.PATTERN_1, 3, 1000);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addSound (Previous format)

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

- You cannot set the buzzer sounding cycle. If you want to optionally set the buzzer sounding cycle (milliseconds), use addSound (p.109).
- 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

```
public void addSound(int pattern, int repeat)
 throws EposException
```

Parameter

- pattern : Specifies the buzzer pattern.

Set value	Description
Builder.PATTERN_A	Pattern A (Optional Buzzer)
Builder.PATTERN_B	Pattern B (Optional Buzzer)
Builder.PATTERN_C	Pattern C (Optional Buzzer)
Builder.PATTERN_D	Pattern D (Optional Buzzer)
Builder.PATTERN_E	Pattern E (Optional Buzzer)
Builder.PATTERN_ERROR	Error sound pattern (Optional Buzzer)
Builder.PATTERN_PAPER_END	Pattern when there is no paper (Optional Buzzer)
Builder.PATTERN_1	Pattern 1 (Internal Buzzer)
Builder.PATTERN_2	Pattern 2 (Internal Buzzer)
Builder.PATTERN_3	Pattern 3 (Internal Buzzer)
Builder.PATTERN_4	Pattern 4 (Internal Buzzer)
Builder.PATTERN_5	Pattern 5 (Internal Buzzer)
Builder.PATTERN_6	Pattern 6 (Internal Buzzer)
Builder.PATTERN_7	Pattern 7 (Internal Buzzer)
Builder.PATTERN_8	Pattern 8 (Internal Buzzer)
Builder.PATTERN_9	Pattern 9 (Internal Buzzer)
Builder.PATTERN_10	Pattern 10 (Internal Buzzer)
Builder.PARAM_DEFAULT	Pattern A

- repeat : Specifies the number of repeats.

Set value	Description
1 to 255	Number of repeats
Builder.PARAM_DEFAULT	One time

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To repeat the sound pattern A three times:

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addSound(Builder.PATTERN_A, 3);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addFeedPosition

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

Syntax

```
public void addFeedPosition(int position)
 throws EposException
```

Parameter

- position : Specifies the feed position.

Set value	Description
Builder.FEED_PEELING	Feeds to the peeling position.
Builder.FEED_CUTTING	Feeds to the cutting position.
Builder.FEED_CURRENT_TOF	Feeds to the top of the current label.
Builder.FEED_NEXT_TOF	Feeds to the top of the next label.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

To feed a label paper to the peeling position:

```
try {
    Builder builder = new Builder("TM-P60II", Builder.MODEL_ANK);
    builder.addFeedPosition(Builder.FEED_PEELING);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

addLayout

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

Syntax

```
public void addLayout(int type, int width, int height,
                      int marginTop, int marginBottom,
                      int offsetCut, int offsetLabel)
                      throws EposException
```

Parameter

- type : Specifies the paper type.

Set value	Description
Builder.LAYOUT_RECEIPT	Receipt paper (no black mark)
Builder.LAYOUT_LABEL	Label paper (no black mark)
Builder.LAYOUT_LABEL_BM	Label paper (with black mark)
Builder.LAYOUT_RECEIPT_BM	Receipt paper (with black mark)

- width : Specifies paper width (in units of 0.1 mm). Specifies an integer from 1 to 10000.
- height : Specifies the distance (in units of 0.1 mm) 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 0.1 mm) from the standard printing position to the top position. Specifies an integer from -9999 to 10000.
- marginBottom :Specifies the distance (in units of 0.1 mm) 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 0.1 mm) from the standard eject position to the cutting position. Specifies an integer from -9999 to 10000.
- offsetLabel : Specifies the distance (in units of 0.1 mm) from the standard eject position to the bottom edge of the label. Specifies an integer from 0 to 10000.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

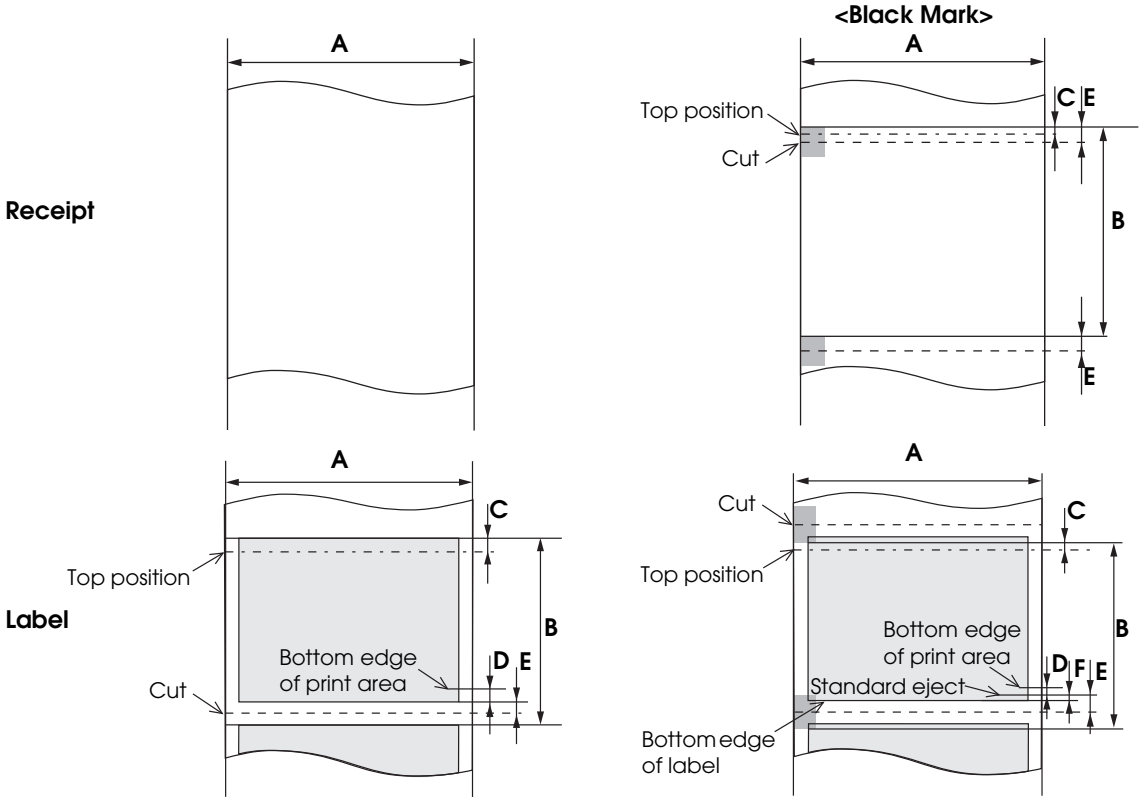
Example

To set 60 mm label paper (black mark):

```
try {
    Builder builder = new Builder("TM-P60II", Builder.MODEL_ANK);
    builder.addLayout(Builder.PAPER_TYPE_LABEL_BM, 600, 0,
                      15, -15, 15, 0);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

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.

Refer to the following URL for details of the ESC/POS command.
https://reference.epson-biz.com/modules/ref_escpos/index.php?content_id=2

Syntax

```
public void addCommand(byte[] data) throws EposException
```

Parameter

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

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

```
try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    byte[] data = null;
    ///Process///
    builder.addCommand(data);
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

Print class (Constructor)

Constructor for the Print class. Initializes a Print class instance.
This constructor is used when using the log output function and when communicating via a USB connection.

Syntax

```
public Print(Context context)
```

Parameter

- context : Specifies the context of the application.

Example

```
import android.content.Context;

Print printer = new Print(getApplicationContext());

    ///Process///
```

Print class (Constructor)(Previous format)

Constructor for the Print class. Initializes a Print class instance.
The log output function cannot be used and communications via a USB connection cannot be made.

Use Print class (Constructor) (p.117) when using the log output function and when communicating via a USB connection.

Syntax

```
public Print()
```

Example

```
Print printer = new Print();

    ///Process///
```

openPrinter

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

	if communication with the printer is not required anymore, be sure to call closePrinter (p.126), closePrinter API, to end communication with the printer.
	• Printer status is notified to the events registered in the print class. For details, see Automatic Acquisition of Printer Status (p.43). • If you want to stop monitoring of printer status, call closePrinter (p.126). • If you use the printer from multiple mobile terminals, see the Cautions (p.206).

Syntax

```
public void openPrinter
(int deviceType, String deviceName, int enabled,
 int interval, int timeout) throws EposException
```

Parameter

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

Set value	Description
Print.DEVTYPE_TCP	Wi-Fi/Ethernet device
Print.DEVTYPE_*BLUETOOTH*	*Bluetooth* device
Print.DEVTYPE_USB	USB device

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

deviceType	Specified Value
Print.DEVTYPE_TCP	One of the following can be specified. • IPv4 IP address (Example: "192.168.192.168") • MAC address (Example: "01:23:45:67:89:AB") • Printer host name (Arbitrary string)
Print.DEVTYPE_*BLUETOOTH*	BD address (Example: "01:23:45:67:89:AB")
Print.DEVTYPE_USB	Device node

	• When a printer's IP address is set as DHCP, specify a MAC address or printer host name for deviceName. • When Print.DEVTYPE_TCP is selected for deviceType, and a printer host name is specified for deviceName, use in an environment in which it is possible to search for a printer host name from the DNS server.

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

Set value	Specified Value
Print.TRUE	Enabled
Print.FALSE	Disabled
Print.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)
Print.PARAM_DEFAULT	Specify the default value (1000)

- timeout : This specifies the maximum waiting time (in milliseconds) for establishing communication with the printer.

Set value	Specified Value
1000 to 300000 integer	Maximum waiting time until an error is returned.
Print.PARAM_DEFAULT	Specify the default value (15000)

- If the specified device does not exist, an error is returned immediately.
- When Print.DEVTYPE_TCP is specified for deviceType, if the specified device is already used, an attempt is made to execute this API until the timeout time.
- For *Bluetooth* communication, specify Print.PARAM_DEFAULT.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_OPEN	The port open process failed.
ERR_TIMEOUT	The device specified was already being used, and communication with the printer could not be established within the timeout time.
ERR_ILLEGAL	An attempt was made to start communicating with the device with which communication had already started.
ERR_PROCESSING	Could not execute process.
ERR_MEMORY	Could not allocate memory.
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

```
Print printer = new Print();
try {
    printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                        Print.PARAM_DEFAULT, Print.PARAM_DEFAULT);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

openPrinter (Previous format)

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

if communication with the printer is not required anymore, be sure to call closePrinter (p.126), closePrinter API, to end communication with the printer.

- The timeout time for this API cannot be set. If you want to set the timeout time for this API, use openPrinter (p.119).
- Printer status is notified to the events registered in the print class.
  For details, see Automatic Acquisition of Printer Status (p.43).
- If you want to stop monitoring of printer status, call closePrinter (p.126).
- If another application opened the printer, depending on the connection method, care should be taken about the following:
  - TCP connection:
    Retry this API for 15 seconds. After 15 seconds, ERR_OPEN will be returned.
  - *Bluetooth* connection:
    When an attempt is made to start communication using this API, its result may not be returned.
- If you use the printer from multiple mobile terminals, see the Cautions (p.206).

Syntax

```
public void openPrinter
(int deviceType, String deviceName, int enabled,
 int interval) throws EposException
```

Parameter

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

Set value	Description
Print.DEVTYPE_TCP	Wi-Fi/Ethernet device
Print.DEVTYPE_*BLUETOOTH*	*Bluetooth* device
Print.DEVTYPE_USB	USB device

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

deviceType	Specified Value
Print.DEVTYPE_TCP	One of the following can be specified. • IPv4 IP address (Example: "192.168.192.168") • MAC address (Example: "01:23:45:67:89:AB") • Printer host name (Arbitrary string)
Print.DEVTYPE_*BLUETOOTH*	BD address (Example: "01:23:45:67:89:AB")
Print.DEVTYPE_USB	Device node

- When a printer's IP address is set as DHCP, specify a MAC address or printer host name for deviceName.
- When Print.DEVTYPE_TCP is selected for deviceType, and a printer host name is specified for deviceName, use in an environment in which it is possible to search for a printer host name from the DNS server.

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

Set value	Specified Value
Print.TRUE	Enabled
Print.FALSE	Disabled
Print.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)
Print.PARAM_DEFAULT	Specify the default value (1000)

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_OPEN	• The port open process failed. • The printer was already in use.
ERR_ILLEGAL	An attempt was made to start communicating with the device with which communication had already started.
ERR_PROCESSING	Could not execute process.
ERR_MEMORY	Could not allocate memory.
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

```
Print printer = new Print();
try {
    printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                        Print.PARAM_DEFAULT);
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

openPrinter (Previous format)

Starts communication with the printer.
Printer status cannot be acquired.

	if communication with the printer is not required anymore, be sure to call closePrinter (p.126), closePrinter API, to end communication with the printer.
	• The timeout time for this API cannot be set. If you want to set the timeout time for this API, use openPrinter (p.119). • If you want to automatically acquire the printer status, use openPrinter (p.119). • If another application opened the printer, depending on the connection method, care should be taken about the following: ∗ TCP connection: Retry this API for 15 seconds. After 15 seconds, ERR_OPEN will be returned. ∗ *Bluetooth* connection: When an attempt is made to start communication using this API, its result may not be returned. • If you use the printer from multiple mobile terminals, see the Cautions (p.206).

Syntax

```
public void openPrinter
(int deviceType, String deviceName) throws EposException
```

Parameter

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

Set value	Description
Print.DEVTYPE_TCP	Wi-Fi/Ethernet device
Print.DEVTYPE_*BLUETOOTH*	*Bluetooth* device
Print.DEVTYPE_USB	USB device

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

deviceType	Specified Value
Print.DEVTYPE_TCP	One of the following can be specified. • IPv4 IP address (Example: "192.168.192.168") • MAC address (Example: "01:23:45:67:89:AB") • Printer host name (Arbitrary string)
Print.DEVTYPE_*BLUETOOTH*	BD address (Example: "01:23:45:67:89:AB")
Print.DEVTYPE_USB	Device node

	• When a printer's IP address is set as DHCP, specify a MAC address or printer host name for deviceName. • When Print.DEVTYPE_TCP is selected for deviceType, and a printer host name is specified for deviceName, use in an environment in which it is possible to search for a printer host name from the DNS server.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_OPEN	• The port open process failed. • The printer was already in use.
ERR_ILLEGAL	An attempt was made to start communicating with the device with which communication had already started.
ERR_PROCESSING	Could not execute process.
ERR_MEMORY	Could not allocate memory.
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":

```
Print printer = new Print();
try {
    printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168");
    ///Process///
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

closePrinter

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

Syntax

```
public void closePrinter() throws EposException
```

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_ILLEGAL	This API was called when communication had not started yet.
ERR_PROCESSING	Could not execute process.
ERR_FAILURE	An unspecified error occurred.

Example

```
Print printer = new Print();
try {
    printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168");
    ///Process///
    printer.closePrinter();
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
}
```

sendData

Sends a print document created using the Builder class.

- If you are using a *Bluetooth* connection, it may not be able to detect the offline status, and timeout errors may occur.
- If you use the printer from multiple mobile terminals, see the Cautions (p.206).

Syntax

```
public void sendData
        (Builder builder, int timeout, int[] status,
         int[] battery) throws EposException
```

Parameter

- builder : Specifies a Builder class instance. For details on the Builder class, refer to Builder class (p.55).
- timeout : Specifies the transmission/reception waiting timeout time.
  Adjust the timeout time according to the specifications for the model, communication interface, and transmission data size.
  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 Statuses and Actions to Take (p.51).
- battery : The battery status when command transmission ended is set.
  For details, refer to Support Information by Printer (p.178).

If an exception occurs, use getPrinterStatus (p.149) with exception handling to acquire the printer status, and getBatteryStatus (p.150) for the battery status.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_ILLEGAL	This API was called when communication had not started yet.
ERR_PROCESSING	Could not execute process.
ERR_TIMEOUT	Could not send all the data within the specified time.
ERR_CONNECT	Connection error occurred
ERR_MEMORY	Could not allocate memory.
ERR_OFF_LINE	The printer was offline.
ERR_FAILURE	An unspecified error occurred.

Example

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

```
Print printer = new Print();
int[] status = new int[1];
int[] battery = new int[1];
status[0] = 0;
battery[0] = 0;

try {
    Builder builder = new Builder("TM-P60II", Builder.MODEL_ANK);
    builder.addText("ABCDE");

    printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168");
    printer.sendData(builder, 10000, status, battery);
    printer.closePrinter();
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
    status[0] = e.getPrinterStatus();
    battery[0] = e.getBatteryStatus();
}
```

sendData (Previous format)

Sends a print document created using the Builder class. The battery status cannot be acquired.

- If you want to acquire the battery status when sending a print document, use sendData (p.127).
- The battery status cannot be acquired. If you are using a *Bluetooth* connection, it may not be able to detect the offline status, and timeout errors may occur.
- If you use the printer from multiple mobile terminals, see the Cautions (p.206).

Syntax

```
public void sendData
(Builder builder, int timeout, int[] status)
 throws EposException
```

Parameter

- builder : Specifies a Builder class instance. For details on the Builder class, refer to Builder class (p.55).
- timeout : Specifies the transmission/reception waiting timeout time.
  Adjust the timeout time according to the specifications for the model, communication interface, and transmission data size.
  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 Statuses and Actions to Take (p.51).

In an exception occurs, use the getPrinterStatus (p.149) with exception processing to get the printer status.

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_ILLEGAL	This API was called when communication had not started yet.
ERR_PROCESSING	Could not execute process.
ERR_TIMEOUT	Could not send all the data within the specified time.
ERR_CONNECT	Connection error occurred
ERR_MEMORY	Could not allocate memory.
ERR_OFF_LINE	The printer was offline.
ERR_FAILURE	An unspecified error occurred.

Example

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

```
Print printer = new Print();
int[] status = new int[1];
status[0] = 0;

try {
    Builder builder = new Builder("TM-T88V", Builder.MODEL_ANK);
    builder.addText("ABCDE");

    printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168");
    printer.sendData(builder, 10000, status);
    printer.closePrinter();
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
    status[0] = e.getPrinterStatus();
}
```

beginTransaction

Starts transaction.
Transaction indicates a set of print processing operations, such as printing a sheet of receipt or a coupon. The operation from just after calling this API to when the transaction finishes using endTransaction (p.132)is handled as one set of print processing operations.

For details about specifying a transaction, see the To specify a transaction (p.208).

Syntax

```
public void beginTransaction() throws EposException
```

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_ILLEGAL	This API was called when communication had not started yet. Transaction has already started using this function.
ERR_FAILURE	An unspecified error occurred.

Example

Multiple or one print processing is transacted.:

```
int[] status = new int[1];
status[0] = 0;

try {
    // For printerModel and lang, specify the model you are using.
    Builder builder = new Builder(printerModel, lang);
    Builder builder2 = new Builder(printerModel, lang);

    builder.addText("ABCDE");

    builder2.addText("12345");
    builder2.addCut(Builder.CUT_FEED);


    Print printer = new Print();
    printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168");
    printer.beginTransaction();

    printer.sendData(builder, 10000, status);

    printer.sendData(builder2, 10000, status);

    printer.endTransaction();
    printer.closePrinter();
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
    status[0] = e.getPrinterStatus();
}
```

endTransaction

Finishes transaction.
Transaction indicates a set of print processing operations, such as printing a sheet of receipt or a coupon. The operation from just after calling beginTransaction (p.131) to when the transaction finishes using this API is handled as one set of print processing operations.

For details about specifying a transaction, see the To specify a transaction (p.208).

Syntax

```
public void endTransaction() throws EposException
```

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_ILLEGAL	This API was called when communication had not started yet. This API was called when transaction had not started.
ERR_FAILURE	An unspecified error occurred.

Example

Multiple or one print processing is transacted.:

```
int[] status = new int[1];
status[0] = 0;

try {
    // For printerModel and lang, specify the model you are using.
    Builder builder = new Builder(printerModel, lang);
    Builder builder2 = new Builder(printerModel, lang);

    builder.addText("ABCDE");

    builder2.addText("12345");
    builder2.addCut(Builder.CUT_FEED);


    Print printer = new Print();
    printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168");
    printer.beginTransaction();

    printer.sendData(builder, 10000, status);

    printer.sendData(builder2, 10000, status);

    printer.endTransaction();
    printer.closePrinter();
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
    status[0] = e.getPrinterStatus();
}
```

setStatusChangeEventCallback

This registers the notification destination of printer status.

- This API can be executed following execution of openPrinter (p.119).
- When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten.

Syntax

```
public void setStatusChangeEventCallback
            (StatusChangeEventListener target)
```

Parameter

- target : This specifies the object (listener interface) that has the notification destination method (listener registration method). If null is specified for either the method or target, the notification destination registration is nullified.

Listener Interface

```
public interface StatusChangeEventListener
                 extends EventListener
```

Listener Registration Method

```
void onStatusChangeEvent(String deviceName, int status)
```

Parameter

- deviceName : The identifier (IPv4 type IP address/ BD address/ Device node/ Printer host name) of the device that is notified of printer status is set.
- status : Printer status is set.

Example

```
public class SampleActivity extends Activity implements
                                    OnClickListener, StatusChangeEventListener {
    ///Process///
    private void onStatusChangeEvent(String deviceName, int status) {
        ///Process///
    }
    private void openPrinter() {
        Print printer = new Print();
        printer.setStatusChangeEventCallback(this);
        try {
            printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                    Print.PARAM_DEFAULT);
            ///Process///
        } catch (EposException e) {
            int errStatus = e.getErrorStatus();
        }
    }
}
```

setOnlineEventCallback

This registers the notification destination of online events. This refers to events that are notified when printer status is online.

- This API can be executed following execution of openPrinter (p.119).
- When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten.

Syntax

```
public void setOnlineEventCallback
            (OnlineEventListener target)
```

Parameter

- target : This specifies the object (listener interface) that has the notification destination method (listener registration method). If null is specified for either the method or target, the notification destination registration is nullified.

Listener Interface

```
public interface OnlineEventListener extends
                 EventListener
```

Listener Registration Method

```
void onOnlineEvent(String deviceName)
```

Parameter

- deviceName : The identifier (IPv4 type IP address/ BD address/ Device node/ Printer host name) of the device that is notified of online event is set.

Example

```
public class SampleActivity extends Activity implements
                                    OnClickListener, OnlineEventListener {
    ///Process///
    private void onOnlineEvent(String deviceName) {
        ///Process///
    }
    private void openPrinter() {
        Print printer = new Print();
        printer.setOnlineEventCallback(this);
        try {
            printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                    Print.PARAM_DEFAULT);
            ///Process///
        } catch (EposException e) {
            int errStatus = e.getErrorStatus();
        }
    }
}
```

setOfflineEventCallback

This registers the notification destination of offline events. This is the notification method when printer is offline concerning printer status.

- This API can be executed following execution of openPrinter (p.119).
- When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten.

Syntax

```
public void setOfflineEventCallback
            (OfflineEventListener target)
```

Parameter

- target : This specifies the object (listener interface) that has the notification destination method (listener registration method). If null is specified for either the method or target, the notification destination registration is nullified.

Listener Interface

```
public interface OfflineEventListener extends
                 EventListener
```

Listener Registration Method

```
void onOfflineEvent(String deviceName)
```

Parameter

- deviceName : The identifier (IPv4 type IP address/ BD address/ Device node/ Printer host name) of the device that is notified of offline event is set.

Example

```
public class SampleActivity extends Activity implements
                                    OnClickListener, OfflineEventListener {
    ///Process///
    private void onOfflineEvent(String deviceName) {
        ///Process///
    }
    private void openPrinter() {
        Print printer = new Print();
        printer.setOfflineEventCallback(this);
        try {
            printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                    Print.PARAM_DEFAULT);
            ///Process///
        } catch (EposException e) {
            int errStatus = e.getErrorStatus();
        }
    }
}
```

setPowerOffEventCallback

This registers the notification destination of power off events. This refers to events that are notified when there is no response concerning printer status.

- This API can be executed following execution of openPrinter (p.119).
- When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten.

Syntax

```
public void setPowerOffEventCallback
            (PowerOffEventListener target)
```

Parameter

- target : This specifies the object (listener interface) that has the notification destination method (listener registration method). If null is specified for either the method or target, the notification destination registration is nullified.

Listener Interface

```
public interface PowerOffEventListener extends
                 EventListener
```

Listener Registration Method

```
void onPowerOffEvent(String deviceName)
```

Parameter

- deviceName : The identifier (IPv4 type IP address/ BD address/ Device node/ Printer host name) of the device that is notified of power off event is set.

Example

```
public class SampleActivity extends Activity implements
                                    OnClickListener, PowerOffEventListener {
    ///Process///
    private void onPowerOffEvent(String deviceName) {
        ///Process///
    }
    private void openPrinter() {
        Print printer = new Print();
        printer.setPowerOffEventCallback(this);
        try {
            printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                    Print.PARAM_DEFAULT);
            ///Process///
        } catch (EposException e) {
            int errStatus = e.getErrorStatus();
        }
    }
}
```

setCoverOkEventCallback

This registers the notification destination of cover close events. This refers to events that are notified when printer status indicates cover close.

- This API can be executed following execution of openPrinter (p.119).
- When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten.

Syntax

```
public void setCoverOkEventCallback
            (CoverOkEventListener target)
```

Parameter

- target : This specifies the object (listener interface) that has the notification destination method (listener registration method). If null is specified for either the method or target, the notification destination registration is nullified.

Listener Interface

```
public interface CoverOkEventListener extends
                 EventListener
```

Listener Registration Method

```
void onCoverOkEvent(String deviceName)
```

Parameter

- deviceName : The identifier (IPv4 type IP address/ BD address/ Device node/ Printer host name) of the device that is notified of cover close event is set.

Example

```
public class SampleActivity extends Activity implements
                                    OnClickListener, CoverOkEventListener {
    ///Process///
    private void onCoverOkEvent(String deviceName) {
        ///Process///
    }
    private void openPrinter() {
        Print printer = new Print();
        printer.setCoverOkEventCallback(this);
        try {
            printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                    Print.PARAM_DEFAULT);
            ///Process///
        } catch (EposException e) {
            int errStatus = e.getErrorStatus();
        }
    }
}
```

setCoverOpenEventCallback

This registers the notification destination of cover open events. This refers to events that are notified when the cover is open concerning printer status.

- This API can be executed following execution of openPrinter (p.119).
- When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten.

Syntax

```
public void setCoverOpenEventCallback
            (CoverOpenEventListener target)
```

Parameter

- target : This specifies the object (listener interface) that has the notification destination method (listener registration method). If null is specified for either the method or target, the notification destination registration is nullified.

Listener Interface

```
public interface CoverOpenEventListener extends
                 EventListener
```

Listener Registration Method

```
void onCoverOpenEvent(String deviceName)
```

Parameter

- deviceName : The identifier (IPv4 type IP address/ BD address/ Device node/ Printer host name) of the device that is notified of cover open event is set.

Example

```
public class SampleActivity extends Activity implements
                                    OnClickListener, CoverOpenEventListener {
    ///Process///
    private void onCoverOpenEvent(String deviceName) {
        ///Process///
    }
    private void openPrinter() {
        Print printer = new Print();
        printer.setCoverOpenEventCallback(this);
        try {
            printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                    Print.PARAM_DEFAULT);
            ///Process///
        } catch (EposException e) {
            int errStatus = e.getErrorStatus();
        }
    }
}
```

setPaperOkEventCallback

This registers the notification destination of paper OK events. This refers to events that are notified when printer status indicates paper OK.

- This API can be executed following execution of openPrinter (p.119).
- When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten.

Syntax

```
public void setPaperOkEventCallback
            (PaperOkEventListener target)
```

Parameter

- target : This specifies the object (listener interface) that has the notification destination method (listener registration method). If null is specified for either the method or target, the notification destination registration is nullified.

Listener Interface

```
public interface PaperOkEventListener extends
                 EventListener
```

Listener Registration Method

```
void onPaperOkEvent(String deviceName)
```

Parameter

- deviceName : The identifier (IPv4 type IP address/ BD address/ Device node/ Printer host name) of the device that is notified of paper ok event is set.

Example

```
public class SampleActivity extends Activity implements
                                    OnClickListener, PaperOkEventListener {
    ///Process///
    private void onPaperOkEvent(String deviceName) {
        ///Process///
    }
    private void openPrinter() {
        Print printer = new Print();
        printer.setPaperOkEventCallback(this);
        try {
            printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                    Print.PARAM_DEFAULT);
            ///Process///
        } catch (EposException e) {
            int errStatus = e.getErrorStatus();
        }
    }
}
```

setPaperNearEndEventCallback

This registers the notification destination of paper near end events. This refers to events that are notified when printer status indicates paper is near the end.

- This API can be executed following execution of openPrinter (p.119).
- When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten.

Syntax

```
public void setPaperNearEndEventCallback
            (PaperNearEndEventListener target)
```

Parameter

- target : This specifies the object (listener interface) that has the notification destination method (listener registration method). If null is specified for either the method or target, the notification destination registration is nullified.

Listener Interface

```
public interface PaperNearEndEventListener extends
                 EventListener
```

Listener Registration Method

```
void onPaperNearEndEvent(String deviceName)
```

Parameter

- deviceName : The identifier (IPv4 type IP address/ BD address/ Device node/ Printer host name) of the device that is notified of paper near end event is set.

Example

```
public class SampleActivity extends Activity implements
                                    OnClickListener, PaperNearEndEventListener {
    ///Process///
    private void onPaperNearEndEvent(String deviceName) {
        ///Process///
    }
    private void openPrinter() {
        Print printer = new Print();
        printer.setPaperNearEndEventCallback(this);
        try {
            printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                    Print.PARAM_DEFAULT);
            ///Process///
        } catch (EposException e) {
            int errStatus = e.getErrorStatus();
        }
    }
}
```

setPaperEndEventCallback

This registers the notification destination of paper end events. This refers to events that are notified when printer status indicates there is no paper.

- This API can be executed following execution of openPrinter (p.119).
- When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten.

Syntax

```
public void setPaperEndEventCallback
            (PaperEndEventListener target)
```

Parameter

- target : This specifies the object (listener interface) that has the notification destination method (listener registration method). If null is specified for either the method or target, the notification destination registration is nullified.

Listener Interface

```
public interface PaperEndEventListener extends
                 EventListener
```

Listener Registration Method

```
void onPaperEndEvent(String deviceName)
```

Parameter

- deviceName : The identifier (IPv4 type IP address/ BD address/ Device node/ Printer host name) of the device that is notified of paper end event is set.

Example

```
public class SampleActivity extends Activity implements
                                    OnClickListener, PaperEndEventListener {
    ///Process///
    private void onPaperEndEvent(String deviceName) {
        ///Process///
    }
    private void openPrinter() {
        Print printer = new Print();
        printer.setPaperEndEventCallback(this);
        try {
            printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                    Print.PARAM_DEFAULT);
            ///Process///
        } catch (EposException e) {
            int errStatus = e.getErrorStatus();
        }
    }
}
```

setDrawerClosedEventCallback

This registers the notification destination of drawer closed events. This refers to events that are notified when printer status indicates the drawer is closed.

- This API can be executed following execution of openPrinter (p.119).
- When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten.

Syntax

```
public void setDrawerClosedEventCallback
            (DrawerClosedEventListener target)
```

Parameter

- target : This specifies the object (listener interface) that has the notification destination method (listener registration method). If null is specified for either the method or target, the notification destination registration is nullified.

Listener Interface

```
public interface DrawerClosedEventListener extends
                 EventListener
```

Listener Registration Method

```
void onDrawerClosedEvent(String deviceName)
```

Parameter

- deviceName : The identifier (IPv4 type IP address/ BD address/ Device node/ Printer host name) of the device that is notified of drawer closed event is set.

Example

```
public class SampleActivity extends Activity implements
                                    OnClickListener, DrawerClosedEventListener {
    ///Process///
    private void onDrawerClosedEvent(String deviceName) {
        ///Process///
    }
    private void openPrinter() {
        Print printer = new Print();
        printer.setDrawerClosedEventCallback(this);
        try {
            printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                    Print.PARAM_DEFAULT);
            ///Process///
        } catch (EposException e) {
            int errStatus = e.getErrorStatus();
        }
    }
}
```

setDrawerOpenEventCallback

This registers the notification destination of drawer open events. This refers to events that are notified when printer status is drawer open.

- This API can be executed following execution of openPrinter (p.119).
- When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten.

Syntax

```
public void setDrawerOpenEventCallback
            (DrawerOpenEventListener target)
```

Parameter

- target : This specifies the object (listener interface) that has the notification destination method (listener registration method). If null is specified for either the method or target, the notification destination registration is nullified.

Listener Interface

```
public interface DrawerOpenEventListener extends
                 EventListener
```

Listener Registration Method

```
void onDrawerOpenEvent(String deviceName)
```

Parameter

- deviceName : The identifier (IPv4 type IP address/ BD address/ Device node/ Printer host name) of the device that is notified of drawer open event is set.

Example

```
public class SampleActivity extends Activity implements
                                    OnClickListener, DrawerOpenEventListener {
    ///Process///
    private void onDrawerOpenEvent(String deviceName) {
        ///Process///
    }
    private void openPrinter() {
        Print printer = new Print();
        printer.setDrawerOpenEventCallback(this);
        try {
            printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                    Print.PARAM_DEFAULT);
            ///Process///
        } catch (EposException e) {
            int errStatus = e.getErrorStatus();
        }
    }
}
```

setBatteryLowEventCallback

This registers the notification destination of a battery low event. This refers to events that are notified when printer status is battery offline.

- This API can be executed following execution of openPrinter (p.119).
- When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten.

Syntax

```
public void setBatteryLowEventCallback
            (BatteryLowEventListener target)
```

Parameter

- target : This specifies the object (listener interface) that has the notification destination method (listener registration method). If null is specified for either the method or target, the notification destination registration is nullified.

Listener Interface

```
public interface BatteryLowEventListener extends
                 EventListener
```

Listener Registration Method

```
void onBatteryLowEvent(String deviceName)
```

Parameter

- deviceName : The identifier (IPv4 format IP address / BD address/ Device node/ Printer host name) of the device that performed battery low event notification is set.

Example

```
public class SampleActivity extends Activity implements
                                    OnClickListener, BatteryLowEventListener {
    ///Process///
    private void onBatteryLowEvent(String deviceName) {
        ///Process///
    }
    private void openPrinter() {
        Print printer = new Print();
        printer.setBatteryLowEventCallback(this);
        try {
            printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                    Print.PARAM_DEFAULT);
            ///Process///
        } catch (EposException e) {
            int errStatus = e.getErrorStatus();
        }
    }
}
```

setBatteryOkEventCallback

This registers the notification destination of a battery OK event. This refers to events that are notified when the printer status recovers from offline due to remaining battery power.

- This API can be executed following execution of openPrinter (p.119).
- When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten.

Syntax

```
public void setBatteryOkEventCallback
            (BatteryOkEventListener target)
```

Parameter

- target : This specifies the object (listener interface) that has the notification destination method (listener registration method). If null is specified for either the method or target, the notification destination registration is nullified.

Listener Interface

```
public interface BatteryOkEventListener extends
                 EventListener
```

Listener Registration Method

```
void onBatteryOkEvent(String deviceName)
```

Parameter

- deviceName : The identifier (IPv4 format IP address / BD address/ Device node/ Printer host name) of the device that performed battery OK event notification is set.

Example

```
public class SampleActivity extends Activity implements
                              OnClickListener, BatteryOkEventListener {
    ///Process///
    private void onBatteryOkEvent(String deviceName) {
        ///Process///
    }
    private void openPrinter() {
        Print printer = new Print();
        printer.setBatteryOkEventCallback(this);
        try {
            printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                    Print.PARAM_DEFAULT);
            ///Process///
        } catch (EposException e) {
            int errStatus = e.getErrorStatus();
        }
    }
}
```

setBatteryStatusChangeEventCallback

This registers the notification destination of battery status.

- This API can be executed following execution of openPrinter (p.119).
- When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten.

Syntax

```
public void setBatteryStatusChangeEventCallback
            (BatteryStatusChangeEventListener target)
```

Parameter

- target : This specifies the object (listener interface) that has the notification destination method (listener registration method). If null is specified for either the method or target, the notification destination registration is nullified.

Listener Interface

```
public interface BatteryStatusChangeEventListener
                 extends EventListener
```

Listener Registration Method

```
void onBatteryStatusChangeEvent
                (String deviceName, int battery)
```

Parameter

- deviceName : The identifier (IPv4 type IP address/ BD address/ Device node/ Printer host name) of the device that is notified of battery status is set.
- battery : Battery status is set.

Example

```
public class SampleActivity extends Activity implements
                        OnClickListener, BatteryStatusChangeEventListener {
    ///Process///
    private void onBatteryStatusChangeEvent(String deviceName, int battery) {
        ///Process///
    }
    private void openPrinter() {
        Print printer = new Print();
        printer.setBatteryStatusChangeEventCallback(this);
        try {
            printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168", Print.TRUE,
                    Print.PARAM_DEFAULT);
            ///Process///
        } catch (EposException e) {
            int errStatus = e.getErrorStatus();
        }
    }
}
```

getStatus

Acquires the printer status and the battery status.
In addition to the printer statuses acquired by sendData (p.127), this API can acquire the following printer statuses.

- Head temporary overheat error
- Motor driver IC temporary overheat error
- Battery temporary overheat error
- Paper error

Syntax

```
public void getStatus(int[] status, int[] battery)
                        throws EposException
```

Parameter

- status : This sets the printer status at the time this API was executed.
  A combination of printer status settings is set. For details, refer to Printer Statuses and Actions to Take (p.51).
- battery : This sets the battery status at the time this API was executed.
  For details, refer to Support Information by Printer (p.178).

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_ILLEGAL	This API was called when communication had not started yet.
ERR_MEMORY	Could not allocate memory.
ERR_FAILURE	An unspecified error occurred.

Example

```
Print printer = new Print();
int[] status = new int[1];
int[] battery = new int[1];
try {
    printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168");
    printer.getStatus(status, battery);
    ///Process///
} catch (EposException e) {
    int errSratus = e.getErrorStatus();
}
```

getErrorStatus

Acquire the error status from an exception.

Syntax

```
public int getErrorStatus()
```

Return value

Returns the error status set by the API in which an exception occurred.

Example

To acquire the error status from EposException.

```
try {
    printer.openPrinter(Print.DEVTYPE_TCP, "192.168.192.168");
} catch (EposException e) {
    int errStatus = e.getErrorStatus();
    if (errStatus == EposException.ERR_OPEN) {
       ///Process///
    }
}
```

getPrinterStatus

Acquires the printer status from an exception that occurred in sendData (p.127).

Syntax

```
public int getPrinterStatus()
```

Return value

Returns the printer status. A combination of printer status settings is set.
For details, refer to Printer Statuses and Actions to Take (p.51).

Example

To acquire the printer status from EposException.

```
int[] printerStatus = new int[1];
printerStatus[0] = 0;
int timeout = 1000;

try {
    printer.sendData(builder, timeout, printerStatus);
} catch (EposException e) {
    int errSratus = e.getErrorStatus();
    if (errStatus == EposException.ERR_TIMEOUT) {
        printerStatus[0] = e.getPrinterStatus();
    }
}

if ((printerStatus[0] & Print.ST_PRINT_SUCCESS) == Print.ST_PRINT_SUCCESS)
{
    ///Process///
}
```

getBatteryStatus

Acquires the battery status from an exception that occurred in sendData (p.127).

Syntax

```
public int getBatteryStatus()
```

Return value

Returns the battery status. For details, refer to Support Information by Printer (p.178).

Example

To acquire the battery status from EposException.

```
int[] printerStatus = new int[1];
int[] batteryStatus = new int[1];
printerStatus[0] = 0;
batteryStatus[0] = 0;
int timeout = 1000;

try {
    printer.sendData(builder, timeout, printerStatus, batteryStatus);
} catch (EposException e) {
    int errSratus = e.getErrorStatus();
    if (errStatus == EposException.ERR_TIMEOUT) {
        printerStatus[0] = e.getPrinterStatus();
        batteryStatus[0] = e.getBatteryStatus();
    }
}

if ((printerStatus[0] & Print.ST_PRINT_SUCCESS) == Print.ST_PRINT_SUCCESS)
{
    ///Process///
}
```

Printer Search API

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

- ❏ Finder class (p. 151)
- ❏ EpsonIoException class (p. 151)

Finder class

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

API	Description	Page
start	Starts searching for printers.	152
stop	End communication with the printer.	153
getDeviceInfoList	Getting the printer search result.	156
getResult (Previous format)		156

EpsonIoException class

This class notifies you of the exception error value that occurred during the API calling of the Finder class and the EpsonIo class (p.171).
The following APIs are available.

API	Description	Page
getStatus	Acquires an error value of an exception	157

start

Starts a search for printers of the specified device type.

If you use this API, be sure to use stop (p.153) to stop the search.

You cannot call this API when a printer search is already in progress.

Syntax

```
public static synchronized void start
            (Context context, int deviceType,
             String findOption)
             throws EpsonIoException
```

Parameter

- context : Set a Context class instance of caller.
  (Example: Set the Context acquired by getBaseContext() in Activity.)
- deviceType : Specifies the device type to search for. The following values can be specified.

deviceType	Description
DevType.TCP	Searches for TM devices connected to the network
DevType.*BLUETOOTH*	Searches for *Bluetooth* devices that have a device class of Printer or Uncategorized.
DevType.USB	Searches for USB devices from the PID and VID. Search condition Searches for TM device connected to the USB.

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

deviceType	Setting Value
DevType.TCP	The broadcast address to search for
DevType.*BLUETOOTH*	"null"
DevType.USB	"null"

Exceptions

When processing fails, EpsonIoException is thrown with one of the following error values.

Error Value	Description
IoStatus.ERR_ILLEGAL	This API was called when a search was already in progress
IoStatus.ERR_PROCESSING	Could not execute process.
IoStatus.ERR_PARAM	Invalid parameter was passed.
IoStatus.ERR_MEMORY	Could not allocate memory.
IoStatus.ERR_FAILURE	An unspecified error occurred.

stop

Stops the printer search.

Syntax

```
public static synchronized void stop()
                throws EpsonIoException
```

Exceptions

When processing fails, EpsonIoException is thrown with one of the following error values.

Error Value	Description
IoStatus.ERR_ILLEGAL	This API was called when a search was not in progress.
IoStatus.ERR_PROCESSING	Could not execute process.
IoStatus.ERR_FAILURE	An unspecified error occurred.

getDeviceinfolist

Acquires the results of device search until this API is called.

This API cannot acquire *Bluetooth* devices or USB devices that are already open.

Syntax

```
public static synchronized final DeviceInfo[]
                getDeviceInfoList(int filterOption)
                throws EpsonIoException
```

Parameter

- filterOption : This specifies the filtering method for Epson printers. Specify one of the following values:

Set value	Description
FilterOption.FILTER_NONE	Do not filter
FilterOption.FILTER_NAME	Filter in the printer name.
FilterOption.PARAM_DEFAULT	Filter in the printer name.

For TCP or USB connection, only Epson printers are searched for regardless of the FitlerOption setting.

Return value

The device information list (DeviceInfo[]) of devices found during search is returned.
Device information is stored in the list as a DeviceInfo-type array.

Information to be stored varies depending on the device type.

deviceType	DeviceInfo	Information to be obtained
DevType.TCP	getDeviceType()	DevType.TCP(Fixed)
	getPrinterName()	Printer model name
	getDeviceName()	• DHCP disabled: IP address • DHCP enabled: MAC address
	getIpAddress()	IP Address
	getMacAddress()	MAC Address
DevType.*BLUETOOTH*	getDeviceType()	DevType.*BLUETOOTH*(Fixed)
	getPrinterName()	*Bluetooth* device name
	getDeviceName()	BD Address (the same format as the MAC address format)
	getIpAddress()	"" (Empty character)
	getMacAddress()	"" (Empty character)
DevType.USB	getDeviceType()	DevType.USB(Fixed)
	getPrinterName()	"TMPrinter" (Fixed)
	getDeviceName()	Device node
	getIpAddress()	"" (Empty character)
	getMacAddress()	"" (Empty character)

Exceptions

When processing fails, EpsonIoException is thrown with one of the following error values.

Error Value	Description
IoStatus.ERR_ILLEGAL	This API was called when a search was already in progress
IoStatus.ERR_PROCESSING	Could not execute process.
IoStatus.ERR_PARAM	Invalid parameter was passed.
IoStatus.ERR_MEMORY	Could not allocate memory.
IoStatus.ERR_FAILURE	An unspecified error occurred.

getResult (Previous format)

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

This API cannot acquire *Bluetooth* devices or USB devices that are already open.

Syntax

```
public static synchronized final String[] getResult()
                throws EpsonIoException
```

Return value

The list of devices found during search is returned.
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
DevType.TCP	List of IP addresses of printers
DevType.*BLUETOOTH*	List of BD addresses of *Bluetooth* devices
DevType.USB	List of device node of USB devices

Exceptions

When processing fails, EpsonIoException is thrown with one of the following error values.

Error Value	Description
IoStatus.ERR_ILLEGAL	This API was called when a search was not in progress.
IoStatus.ERR_PROCESSING	Could not execute process.
IoStatus.ERR_MEMORY	Could not allocate memory.
IoStatus.ERR_FAILURE	An unspecified error occurred.

getStatus

Gets the error value of the exception.

Syntax

```
public int getStatus();
```

Return value

Returns the error value that is thrown with the exception. Error values are defined in the IoStatus class.

Error Value	Cause
IoStatus.ERR_PARAM	Invalid parameter was passed.
IoStatus.ERR_MEMORY	Could not allocate the necessary memory for processing.
IoStatus.ERR_ILLEGAL	Illegal method used.
IoStatus.ERR_PROCESSING	Could not execute process.
IoStatus.ERR_FAILURE	An unspecified error occurred.

Printer Easy Select API

The Printer Easy Select APIs are APIs for selecting a printer using NFC or QR Code. They convert data obtained from NFC or QR code to a format that can be passed to openPrinter.
The following classes are available.

- ❏ EasySelect class (p. 158)
- ❏ EasySelectInfo class (p. 158)

EasySelect class

Analyzes NFC data and QR code data. The following APIs are available.

API	Description	Page
parseNFC	Analyzes NFC tag data.	159
parseNFC (Previous format)	Analyzes NFC tag data. (Not supported in products that have both Ethernet and Wi-Fi interfaces.)	160
parseQR	Analyzes QR code data.	160
createQR	Creates QR code print data for Easy Select.	161

EasySelectInfo class

This class stores data analyzed by an EasySelect class instance and converts it into a variable to be passed to openPrinter. The following member variables are available.

Member Variable	Description	Page
deviceType	Device type in the analysis result	162
printerName	Printer name in the analysis result	162
macAddress	MAC address or BD address in the analysis result	162

parseNFC

Analyzes NFC tag data.

Syntax

```
public ArrayList<EasySelectInfo>
            parseNFC(Tag tag, int timeout);
```

Parameter

- tag : Specifies NFC tag data.

Analysis Tag	Description
Data for Network	Locally-defined data (for Network)
BTSSP	NFC standard specification (for *Bluetooth*)

- timeout : Specifies the analysis waiting time for NFC tag.
  If the specified value is out of range, PARSE_NFC_TIMEOUT_DEFAULT will be applied.

Set value	Specified Value
0 to 60000 integer	Analysis waiting time (in units of milliseconds)
PARSE_NFC_TIMEOUT_DEFAULT	Specify the default value (500)

Return value

Returns the result of NFC analysis. Stores it into an ArrayList<EasySelectInfo> class instance.
If analysis fails, returns null.

Among the EasySelectInfo members, null or "" will be specified to the information that could not be obtained from tag. The causes and countermeasures are described below.

- When EasySelectInfo.printerName is null or ""

  The printer name is not included in NFC. Specify the printer name in advance or take any other countermeasure.

- When EasySelectInfo.macAddress is ""

  In products that have both Ethernet and Wi-Fi interfaces, if the network communication is disabled or if there is no response from the printer's macAddress within the specified timeout limit, the macAddress is "". Check if the printer is turned on and if the network connection has been established.

parseNFC (Previous format)

Analyzes NFC tag data.
Not supported in products that have both Ethernet and Wi-Fi interfaces.

Syntax

```
public EasySelectInfo parseNFC(Tag tag)
```

Parameter

- tag : Specifies NFC tag data.

Analysis Tag	Description
Wi-Fi data	Locally-defined data (for Wi-Fi)
BTSSP	NFC standard specification (for *Bluetooth*)

Return value

Returns the result of NFC analysis. Stores it into an EasySelectInfo class instance.
If analysis fails, returns null.

Among the EasySelectInfo members, null or "" will be specified to the information that could not be obtained from tag. The causes and countermeasures are described below.

- When EasySelectInfo.printerName is null or ""

  The printer name is not included in NFC. Specify the printer name in advance or take any other countermeasure.

- When EasySelectInfo.macAddress is ""

  When using a product that has both Ethernet and Wi-Fi interfaces, the macAddress is "". Use parseNFC (p.159)

parseQR

Analyzes QR code string data.

Syntax

```
public EasySelectInfo parseQR(String data)
```

Parameter

- data : Specifies QR code string data.

Return value

Returns the result of QR code string data analysis. Stores it into an EasySelectInfo class instance.
If analysis fails, returns null.

createQR

Creates QR code print data for Easy Select.

Syntax

```
public String createQR(String printerName,
                       int deviceType,
                       String macAddress)
```

Parameter

- printerName : Specifies the printer name.
- deviceType : Specifies the device type. Set either of the following:

Set value	Description
Print.DEVTYPE_TCP	Wi-Fi/ Ethernet device
Print.DEVTYPE_*BLUETOOTH*	*Bluetooth* device

- macAddress : Specifies the BD address.
  BD addresses support the following formats:

Format	Description
00:11:22:33:44:55	Separated by a colon ":".
00-11-22-33-44-55	Separated by a hyphen "-".
001122334455	Not separated.

Return value

Returns QR code print data for Easy Select. If print data creation fails, returns null.

deviceType

Stores the device type in the analysis result.

Stored data	Description
Print.DEVTYPE_TCP	Wi-Fi/ Ethernet device
Print.DEVTYPE_*BLUETOOTH*	NFC standard specification (for *Bluetooth*)

Format

```
int deviceType;
```

printerName

Stores the printer name in the analysis result.

Format

```
String printerName;
```

macAddress

Stores the BD address in the analysis result.

Format

```
String macAddress;
```

Log Setting API

Sets the log output. The following class is available.

❏ Log class (p. 163)

Log class

Sets the log output function.

API	Description	Page
setLogSettings	Sets the log output function.	163

setLogSettings

Sets the log output function.

Syntax

```
public static void setLogSettings(Context context,
            int period, int enabled, String ipAddress,
            int port, int logSize, int logLevel)
             throws EposException
```

Parameter

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

Set value	Description
Log.LOG_TEMPORARY	The settings of this API are disabled when the application is ended.
Log.LOG_PERMANENT	The settings of this API are enabled even after the application is ended.

To specify period for the Log.LOG_PERMANENT, set the permissions for the application to access the storage.

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

Set value	Description
Log.LOG_DISABLE	Disables the log output function.
Log.LOG_STORAGE	Outputs log data to the device's storage.
Log.LOG_TCP	Outputs log data over TCP.

- To specify enabled for the Log.LOG_STORAGE, set the permissions for the application to access the storage.
- To specify enabled for the Log.LOG_TCP, set the permissions for the application to access the network.

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

If either of the following values is specified for enabled, "null" can be specified for this parameter.
- Log.LOG_DISABLE
- Log.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.
- Log.LOG_DISABLE
- Log.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.
- Log.LOG_DISABLE
- Log.LOG_TCP

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

Set value	Description
LOG_LOW	Low level

Exceptions

When processing fails, EposException is thrown with one of the following error values.

Error status	Description
ERR_PARAM	Invalid parameter was passed.
ERR_FAILURE	An unspecified error occurred.

Example

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

```
try {
  Log.setLogSettings(getApplicationContext(), Log.LOG_PERMANENT,
                     Log.LOG_TCP, "192.168.192.168", 8080, 10, Log.LOG_LOW);
} catch (EposException e) {
int errStatus = e.getErrorStatus();
    status[0] = e.getPrinterStatus();
}
```

To output log data to the device's storage:

```
try {
  Log.setLogSettings(getApplicationContext(), Log.LOG_PERMANENT,
                     Log.LOG_STORAGE, null, 0, 10, Log.LOG_LOW);
} catch (EposException e) {
int errStatus = e.getErrorStatus();
    status[0] = e.getPrinterStatus();
}
```

To disable the log output function:

```
try {
  Log.setLogSettings(getApplicationContext(), Log.LOG_PERMANENT,
                     Log.LOG_DISABLE, null, 0, 10, Log.LOG_LOW);
} catch (EposException e) {
int errStatus = e.getErrorStatus();
    status[0] = e.getPrinterStatus();
}
```

How to Extract a Log File

Save destination

Android Version	Save destination
Android Version 4.1 or earlier	/(storage path)/Android/data/(application package name)/files/EposLog <Example:> /storage/sdcard0/Android/data/com.example/files/EposLog
Android Version 4.2 or later	/storage/emulated/(index referring to user)/Android/data/(application package name)/files/EposLog <Example:> /storage/emulated/0/Android/data/com.example/files/EposLog

File name

❏ EposLog.xx

How to read a log

Log format

A log record is configured in the following format:

<< date and time, process ID: thread ID, input and output layer, input and output direction, input and output data >>

Item	Description
Date and time	In yyyy/mm/dd,h:mm:ss.000 format.
Process ID: thread ID	ID of each process
Input and output layer	Layer at which data is input and output • APIIO: Interface layer called by the application • IOCM/DEVIO: Layer for communication with devices
Input and output direction	Direction in which data is input and output • ->: Input from a layer • <-: Output from a layer
Input and output data	Called API, parameter, and communication data

Each item is separated by a comma (,).

Output example

To call the addCut method from the application:

```
2014/07/28,20:12:35.836,00002ae9:00006008,APIIO,->,0x687bc5d8,,addCut,1
2014/07/28,20:12:35.836,00002ae9:00006008,APIIO,<-,0x687bc5d8,0,addCut}
```

Command Transmission/Reception

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

⚠	The APIs for command transmission and reception described in this chapter are intended for customers who understand ESC/POS commands very well.
✎	A command transmission/reception API cannot be used with the Print class (p.57) of ePOS-Print API.

Programming

Programming Flow

Perform programming following this flow.

1. Advance Preparations (USB Connection) (p.32) *

2. Printer Selection (p.33) *

<Select printer using search function>
- Selecting a printer using the search function (p.33)
- Getting the printer search result. (p.33)
- Stopping the printer search (p.34)

<Select printer using NFC>
- Selecting a printer using NFC (p.34)

<Select printer using QR code>
- Selecting a printer using QR code (p.35)

3. Opening a Device Port (p.168)

4. Sending Data (p.168)

5. Receiving Data (p.169)

6. Closing the Device Port (p.169)

* This is optional.

Opening a Device Port

Use the EpsonIo class's open (p.171) to open a device port. Please refer to the following code.

```
//Initialize the EpsonIo class
EpsonIo mPort = new EpsonIo();
int errStatus = IoStatus.SUCCESS;

//Open the device port
try {

////Wi-Fi/Ethernet device
    mPort.open(DevType.TCP, "192.168.192.168", null, null);
////Bluetooth device
    mPort.open(DevType.BLUETOOTH, "00:00:12:34:56:78", null, null);
////USB device
    mPort.open(DevType.USB, "/dev/bus/usb/001/002", null, getApplicationContext());

//Exception handling
} catch ( EpsonIoException e ) {
    errStatus = e.getStatus();
}
```

Sending Data

Use the EpsonIo class's write (p.175) to send data to the printer. Please refer to the following code.

Printing out "Hello, World!"

```
//Settings for sending
String str = "Hello, World!\r\n";
byte[] data = str.getBytes();
int offset = 0;
int size = data.length;
int timeout = 5000;
int sizeWritten = 0;
int errStatus = IoStatus.SUCCESS;

try {
//Send data
    sizeWritten = mPort.write(data, offset, size, timeout);
//Exception handling
} catch ( EpsonIoException e ) {
    errStatus = e.getStatus();
}
```

Receiving Data

Use the EpsonIo class's read (p.176) to receive data from the printer. Please refer to the following code.

```
//Settings for receiving
byte[] data = new byte[256];
int offset = 0;
int size = 256;
int timeout = 5000;
int sizeRead = 0;
int errStatus = IoStatus.SUCCESS;

//Receive data
try {
    sizeRead = mPort.read(data, offset, size, timeout);
//Exception handling
} catch ( EpsonIoException e ) {
    errStatus = e.getStatus();
}
```

Closing the Device Port

Use the EpsonIo class's close (p.174) to close the device port. Please refer to the following code.

```
int errStatus = IoStatus.SUCCESS;

//Close the device port
try {
    mPort.close();
//Exception handling
} catch ( EpsonIoException e ) {
    errStatus = e.getStatus();
}
```

Exception handling

A command transmission/reception API generates a propriety exception with an integer (int) type parameter when an error occurs and notify the calling side of such an error.

Steps for Handling

Use the EpsonIoException class's getStatus (p.157) to get the error value. Please refer to the following code.

```
String str = "Hello, World!\r\n";
byte[] data = str.getBytes();
int offset = 0;
int size = data.length;
int timeout = 5000;
int sizeWritten = 0;
int errStatus = IoStatus.SUCCESS;

try {
    sizeWritten = mPort.write(data, offset, size, timeout);
} catch ( EpsonIoException e ) {
//Get error value
    errStatus = e.getStatus();
}
```

List of Error Values

Error values are defined in the IoStatus class.

Error Value	Cause
IoStatus.ERR_PARAM	Invalid parameter was passed. <Example> • An invalid parameter such as null was passed. • A value outside the supported range was specified.
IoStatus.ERR_OPEN	Open processing failed.
IoStatus.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.
IoStatus.ERR_MEMORY	Could not allocate the necessary memory for processing.
IoStatus.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.
IoStatus.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.
IoStatus.ERR_FAILURE	An unspecified error occurred.

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
open	Opens the device port.	171
open(Previous format)	Opens the device port. (The communications via a USB connection cannot be made.)	173
close	Closes the device port.	174
write	Send data.	175
read	Receive data.	176

open

Opens the specified device port.

Syntax

```
public void open
            (int deviceType, String deviceName,
             String deviceSettings, Context context)
             throws EpsonIoException
```

Parameter

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

Set value	Description
DevType.TCP	Specify this when the printer to be opened will connect with Wi-Fi/Ethernet.
DevType.*BLUETOOTH*	Specify this when the printer to be opened will connect with *Bluetooth*.
DevType.USB	Specify this when the printer to be opened will connect with USB.

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

deviceType	Specified Value
DevType.TCP	One of the following can be specified. • IPv4 IP address (Example: "192.168.192.168") • MAC address (Example: "01:23:45:67:89:AB") • Printer host name (Arbitrary string)
DevType.*BLUETOOTH*	BD address (Example: "01:23:45:67:89:AB")
DevType.USB	Device node

- deviceSettings :
  Specify "null".

- context : Specifies the context of the application.

deviceType	Specified Value
DevType.TCP	"null"
DevType.*BLUETOOTH*	"null"
DevType.USB	context of the application

Exceptions

When processing fails, EpsonIoException is thrown with one of the following error values.

Error Value	Description
IoStatus.ERR_OPEN	Open processing failed.
IoStatus.ERR_ILLEGAL	User attempted to open a device that is already open.
IoStatus.ERR_PROCESSING	Could not execute process.
IoStatus.ERR_PARAM	Invalid parameter was passed.
IoStatus.ERR_MEMORY	Could not allocate memory.
IoStatus.ERR_FAILURE	An unspecified error occurred.

open(Previous format)

Opens the specified device port.

Use open (p.171) when communicating via a USB connection.

Syntax

```
public void open
            (int deviceType, String deviceName,
             String deviceSettings)
             throws EpsonIoException
```

Parameter

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

deviceType	Description
DevType.TCP	Specify this when the printer to be opened will connect with Wi-Fi/Ethernet.
DevType.*BLUETOOTH*	Specify this when the printer to be opened will connect with *Bluetooth*.

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

deviceType	Specified Value
DevType.TCP	One of the following can be specified. • IPv4 IP address (Example: "192.168.192.168") • MAC address (Example: "01:23:45:67:89:AB") • Printer host name (Arbitrary string)
DevType.*BLUETOOTH*	BD address (Example: "01:23:45:67:89:AB")

- deviceSettings :

  Specify "null".

Exceptions

When processing fails, EpsonIoException is thrown with one of the following error values.

Error Value	Description
IoStatus.ERR_OPEN	Open processing failed.
IoStatus.ERR_ILLEGAL	User attempted to open a device that is already open.
IoStatus.ERR_PROCESSING	Could not execute process.
IoStatus.ERR_PARAM	Invalid parameter was passed.
IoStatus.ERR_MEMORY	Could not allocate memory.
IoStatus.ERR_FAILURE	An unspecified error occurred.

close

Closes the specified device port.

Syntax

```
public void close() throws EpsonIoException
```

Exceptions

When processing fails, EpsonIoException is thrown with one of the following error values.

Error Value	Description
IoStatus.ERR_ILLEGAL	This API was called when no device port was open.
IoStatus.ERR_PROCESSING	Could not execute process.
IoStatus.ERR_FAILURE	An unspecified error occurred.

write

Sends data to a device port.

Syntax

```
public int write
            (byte[] data, int offset, int size,
             int timeout)
             throws EpsonIoException
```

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, the return value will be "0".

- 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 too 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.
- With a *Bluetooth* device, there is a chance that the sending process will be blocked. In such a case, processing will not complete even if the specified timeout value elapses.

Return value

Returns the number of bytes of data that were sent.

- The printer did not necessarily receive the amount of data that the return value shows.
- If the amount of time specified in timeout is exceeded, the returned return value is the number of bytes that were sent up to that point.

Exceptions

When processing fails, EpsonIoException is thrown with one of the following error values.

Error Value	Description
IoStatus.ERR_ILLEGAL	This API was called when no device port was open.
IoStatus.ERR_PROCESSING	Could not execute process.
IoStatus.ERR_PARAM	Invalid parameter was passed.
IoStatus.ERR_CONNECT	Connection error occurred
IoStatus.ERR_MEMORY	Could not allocate memory.
IoStatus.ERR_FAILURE	An unspecified error occurred.

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

```
public int read
            (byte[] data, int offset, int size,
             int timeout)
             throws EpsonIoException
```

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, the return value will be "0".

- timeout : Specifies the time in milliseconds to receive data. The maximum value that can be specified is 600000 (which equates to 10 minutes).

Return value

Returns the number of bytes that were received.

Exceptions

When processing fails, EpsonIoException is thrown with one of the following error values.

Error Value	Description
IoStatus.ERR_ILLEGAL	This API was called when no device port was open.
IoStatus.ERR_PROCESSING	Could not execute process.
IoStatus.ERR_PARAM	Invalid parameter was passed.
IoStatus.ERR_CONNECT	Connection error occurred
IoStatus.ERR_MEMORY	Could not allocate memory.
IoStatus.ERR_FAILURE	An unspecified error occurred.

Appendix

List of Supported APIs for Each Printer Model

API	TM-m10	TM-P20	TM-P60	TM-P60(Peeler)	TM-P60II	TM-P60II(Peeler)	TM-P80	TM-T20	TM-T20II	TM-T70	TM-T70II	TM-T81II	TM-T82	TM-T82II	TM-T83II	TM-T88V	TM-T90II	TM-U220	TM-U330
addTextAlign (p.64)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
addTextLineSpace (p.65)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
addTextRotate (p.66)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
addText (p.67)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
addTextLang (p.68)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
addTextFont (p.69)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
addTextSmooth (p.70)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	-	-
addTextDouble (p.71)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	-	-
addTextSize (p.72)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	-	-
addTextStyle (p.73)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
addTextPosition (p.75)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	-	✔
addFeedUnit (p.76)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
addFeedLine (p.77)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
addImage (p.78)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
addImage (Previous format) (p.81)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
addImage (Previous format) (p.84)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
addLogo (p.86)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	-	-
addBarcode (p.87)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	-	-
addSymbol (p.92)	✔	✔	-	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	-	-
addPageBegin (p.97)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	-	-
addPageEnd (p.98)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	-	-
addPageArea (p.99)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	-	-
addPageDirection (p.100)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	-	-
addPagePosition (p.102)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	-	-
addPageLine (p.103)	✔	✔	-	✔	✔	✔	✔	-	-	-	-	-	-	-	-	-	-	-	-
addPageRectangle (p.105)	✔	✔	-	✔	✔	✔	✔	-	-	-	-	-	-	-	-	-	-	-	-
addCut (p.107)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
addPulse (p.108)	✔	-	-	-	-	-	-	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
addSound (p.109)	✔	✔	-	-	✔	✔	✔	-	✔	-	✔	-	✔	✔	✔	✔	-	-	✔
addSound (Previous format) (p.111)	✔	✔	-	-	✔	✔	✔	-	✔	-	✔	-	✔	✔	✔	✔	-	-	✔
addFeedPosition (p.113)	-	✔	-	✔	-	✔	✔	-	-	-	-	-	-	-	-	-	-	-	✔
addLayout (p.114)	-	✔	-	✔	-	✔	✔	-	-	-	-	-	-	-	-	-	-	-	-
addCommand (p.116)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔

Support Information by Printer

TM-m10

		58 mm
Resolution		203 dpi x 203 dpi (W x H)
Country		• ANK model • Japanese model • Traditional Chinese model
Print Width		420 dots
Characters in a Line	Font A	ANK: 35 characters Kanji *: 17 characters
	Font B	ANK: 42 characters Kanji *: 21 characters
	Font C	ANK: 46 characters
Character Size	Font A	ANK: 12 dots x 24 dots (W x H) Kanji *: 24 dots x 24 dots (W x H)
	Font B	ANK: 10 dots x 24 dots (W x H) Kanji *: 20 dots x 24 dots (W x H)
	Font C	ANK: 9 dots x 17 dots (W x H)
Character Baseline	Font A	ANK; At the 21st dot from the top of the character Kanji *: At the 21st dot from the top of the character
	Font B	ANK; At the 21st dot from the top of the character Kanji *: At the 21st dot from the top of the character
	Font C	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 1200 dots (W x H)
Page Mode Maximum Area		420 dots x 1200 dots (W x H)
Barcode		UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF,CODABAR, CODE93, CODE128, GS1-128, GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Limited, GS1 DataBar Expanded
Two-Dimensional Code		PDF417, QR Code, MaxiCode, GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional, GS1 DataBar Expanded Stacked (Composite Symbology not supported)
Paper Cut		Cut, Feed cut

	58 mm
Drawer Kick-Out	Supported
Buzzer	Option (Pattern A ~ Pattern E, Error, No paper, Stop)
Battery	Not supported

* Only for Multi-language model

TM-P20 (ANK model / Multi-language model)

		58 mm
Resolution		203 dpi x 203 dpi (W x H)
Country		• ANK model • Japanese model • Simplified Chinese model • Traditional Chinese model • South Asian model
Print Width		384 dots
Characters in a Line	Font A	ANK: 32 characters Kanji *: 16 characters
	Font B	ANK: 42 characters
	Font C	ANK: 42 characters
	Font D	ANK: 38 characters
	Font E	ANK: 48 characters
Character Size	Font A	ANK: 12 dots x 24 dots (W x H) Kanji *: 24 dots x 24 dots (W x H)
	Font B	ANK: 9 dots x 24 dots (W x H)
	Font C	ANK: 9 dots x 17 dots (W x H)
	Font D	ANK: 10 dots x 24 dots (W x H)
	Font E	ANK: 8 dots x 16 dots (W x H)
Character Baseline	Font A	ANK: At the 21st dot from the top of the character Kanji *: 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 16th dot from the top of the character
	Font D	At the 21st dot from the top of the character
	Font E	At the 15th dot from the top of the character
Default Line Feed Space		30 dots
Color Specification		First color
Page Mode Default Area		384 dots x 2400 dots (W x H)
Page Mode Maximum Area		384 dots x 2400 dots (W x H)
Barcode		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

	58 mm
Two-Dimensional Code	PDF417, QR Code, MaxiCode, GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional, GS1 DataBar Expanded Stacked, Aztec Code, Data Matrix, (Composite Symbology not supported)
Paper Cut	Feed cut (Feeds paper to cutting position)
Drawer Kick-Out	Not supported
Buzzer	Support (Pattern 1 ~ Pattern 10, Stop)
Battery	Supported

* Only for Multi-language model

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-P60

		58 mm	60 mm
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)
Barcode		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	

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/ TM-P60II with Peeler (ANK model / Multi-language model)

		Receipt	Die-cut label
Resolution		203 dpi x 203 dpi (W x H)	
Language		• ANK model • Traditional Chinese model	
Print Width		432 dots	160 dots ~ 400 dots
Characters in a Line	Font A	ANK: 36 characters Kanji *: 18 characters	ANK: 33 characters Kanji *: 16 characters
	Font B	ANK: 43 characters	ANK: 40 characters
	Font C	ANK: 54 characters	ANK: 50 characters
Character Size	Font A	ANK: 12 dots x 24 dots (W x H) Kanji *: 24 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	ANK: At the 21st dot from the top of the character Kanji *: 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		432 dots x 1624 dots (W x H)	400 dots x 1624 dots (W x H)
Page Mode Maximum Area		432 dots x 1624 dots (W x H)	400 dots x 1624 dots (W x H)
Barcode		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, Aztec Code, DataMatrix (Composite Symbology not supported)	
Paper Cut	TM-P60II	Cut, Feed cut	
	TM-P60II with Peeler	Feed cut (Feeds paper to cutting position)	
Drawer Kick-Out		Not supported	
Buzzer		Support (Pattern 1 ~ Pattern 10, Stop)	

	Receipt	Die-cut label
Battery	Supported	

* Only for Multi-language model

Paper Layout

Paper type	Receipt paper (without black mark)	Receipt paper (with black mark)	Die-cut label paper (without black mark)	Die-cut label paper (with black mark)
width (sf)	290 to 600	290 to 600	290 to 600	290 to 600
height (sa)	0	0, 284 to 1550	0, 284 to 1550	0, 284 to 1550
marginTop (sb)	0	-130 to 1500	0 to 1500	-15 to 1500
marginBottom (se)	0	0	-15 to 0	-15 to 15
offsetCut (sc)	0	-256 to 50	0 to 50	0 to 50
offsetLabel (sd)	0	0	0	0 to 15

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 (ANK model / Multi-language model)

		80 mm
Resolution		203 dpi x 203 dpi (W x H)
Language		• ANK model • Traditional Chinese model
Print Width		576 dots
Characters in a Line	Font A	ANK: 48 characters Kanji *: 24 characters
	Font B	ANK: 64 characters
Character Size	Font A	ANK: 12 dots x 24 dots (W x H) Kanji *: 24 dots x 24 dots (W x H)
	Font B	ANK: 9 dots x 17 dots (W x H)
Character Baseline	Font A	ANK: At the 21st dot from the top of the character Kanji *: 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)
Barcode		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 not supported)
Paper Cut	manualcutter model	Feed cut (Feeds paper to cutting position)
	autocutter model	Cut, Feed cut
Drawer Kick-Out		Not supported
Buzzer		Support (Pattern 1 ~ Pattern 10, Stop)
Battery		Supported

* Only for Multi-language model

Paper Layout

Paper type	Receipt paper (without black mark)	Receipt paper (with black mark)
width (sf)	800	800
height (sa)	0	0, 284 to 3100
marginTop (sb)	0	-98 to 3100
marginBottom (se)	0	0
offsetCut (sc)	0	-173 to 50
offsetLabel (sd)	0	0

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-T20

		58 mm	80 mm
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)
Barcode		UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF,CODABAR, CODE93, CODE128, GS1-128, GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Limited, GS1 DataBar Expanded	
Two-Dimensional Code		PDF417, QR Code, MaxiCode, GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional, GS1 DataBar Expanded Stacked (Composite Symbology not supported)	
Paper Cut		Cut, Feed cut	
Drawer Kick-Out		Supported	
Buzzer		Option (Pattern A ~ Pattern E, Error, No paper, Stop)	
Battery		Not supported	

TM-T20II

			58 mm	80 mm
Resolution			203 dpi x 203 dpi (W x H)	
Language			ANK model	
Print Width	Normal mode		420 dots	576 dots
	42 Column Mode		378 dots	546 dots
Characters in a Line	Font A	Normal mode	ANK: 35 characters	ANK: 48 characters
		42 Column Mode	ANK: 42 characters	ANK: 42 characters
	Font B	Normal mode	ANK: 46 characters	ANK: 64 characters
		42 Column Mode	ANK: 31 characters	ANK: 60 characters
Character Size	Font A	Normal mode	ANK: 12 dots x 24 dots (W x H)	
		42 Column Mode	ANK: 9 dots x 17 dots (W x H)	ANK: 13 dots x 24 dots (W x H)
	Font B	Normal mode	ANK: 9 dots x 17 dots (W x H)	
		42 Column Mode	ANK: 12 dots x 24 dots (W x H)	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)
Barcode			UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF,CODABAR, CODE93, CODE128, GS1-128, GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Limited, GS1 DataBar Expanded	
Two-Dimensional Code			PDF417, QR Code, MaxiCode, GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional, GS1 DataBar Expanded Stacked (Composite Symbology not supported)	
Paper Cut			Cut, Feed cut	
Drawer Kick-Out			Supported	
Buzzer			Option (Pattern A ~ Pattern E, Error, No paper, Stop)	
Battery			Not supported	

TM-T70 (ANK model)

		80 mm
Resolution		180 dpi x 180 dpi (W x H)
Language		ANK 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 character
	Font B	At the 16th dot from the top of the character
Default Line Feed Space		30 dots
Color Specification		First color
Page Mode Default Area		512 dots x 831 dots (W x H)
Page Mode Maximum Area		512 dots x 1662 dots (W x H)
Barcode		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

TM-T70 (Multi-language model)

		80 mm
Resolution		203 dpi x 203 dpi (W x H)
Language		• Japanese model • Simplified Chinese model • Traditional Chinese model • South Asian model
Print Width		576 dots
Characters in a Line	Font A	ANK: 48 characters Kanji: 24 characters
	Font B	ANK: 64 characters
Character Size	Font A	ANK: 12 dots x 24 dots (W x H) Kanji: 24 dots x 24 dots (W x H)
	Font B	ANK: 9 dots x 17 dots (W x H)
Character Baseline	Font A	ANK: At the 21st dot from the top of the character Kanji: 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)
Barcode		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

TM-T70II (ANK model)

		80 mm
Resolution		180 dpi x 180 dpi (W x H)
Language		ANK model
Print Width		512 dots
Characters in a Line	Font A	ANK: 42 characters
	Font B	ANK: 56 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 character
	Font B	At the 16th dot from the top of the character
Default Line Feed Space		30 dots
Color Specification		First color
Page Mode Default Area		512 dots x 1662 dots (W x H)
Page Mode Maximum Area		512 dots x 1662 dots (W x H)
Barcode		UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF, CODABAR, CODE93, CODE128, GS1-128, GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Limited, GS1 DataBar Expanded
Two-Dimensional Code		PDF417, QR Code, MaxiCode, GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional, GS1 DataBar Expanded Stacked (Composite Symbology not supported)
Paper Cut		Cut, Feed cut
Drawer Kick-Out		Supported
Buzzer		Option (Pattern A ~ Pattern E, Error, No paper, Stop)
Battery		Not supported

TM-T70II (Multi-language model)

		58 mm	80 mm
Resolution		203 dpi x 203 dpi (W x H)	
Language		• Japanese model • Simplified Chinese model • Traditional Chinese model • Korean model • South Asian model	
Print Width		416 dots	576 dots
Characters in a Line	Font A	ANK: 34 characters Kanji *2: 17 characters	ANK: 48 characters Kanji *2: 24 characters
	Font B	ANK *1: 52 characters ANK: 46 characters Kanji *1: 26 characters	ANK *1: 72 characters ANK: 64 characters Kanji *1: 36 characters
	Special font A *2	48 characters	
	Special font B *2	64 characters	
Character Size	Font A	ANK: 12 dots x 24 dots (W x H) Kanji *2: 24 dots x 24 dots (W x H)	
	Font B	ANK *1: 8 dots x 16 dots (W x H) ANK: 9 dots x 17 dots (W x H) Kanji *1: 16 dots x 16 dots (W x H)	
	Special font A *2	12 dots x 24 dots (W x H)	
	Special font B *2	9 dots x 24 dots (W x H)	
Character Baseline	Font A	ANK: At the 21st dot from the top of the character Kanji *2: At the 21st dot from the top of the character	
	Font B	ANK *1: At the 15th dot from the top of the character ANK: At the 16th dot from the top of the character Kanji *1: At the 15th dot from the top of the character	
	Special font A *2	At the 21st dot from the top of the character	
	Special font B *2	At the 21st 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)

	58 mm	80 mm
Barcode	UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF, CODABAR, CODE93, CODE128, GS1-128, GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Limited, GS1 DataBar Expanded	
Two-Dimensional Code	PDF417, QR Code, MaxiCode, GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional, GS1 DataBar Expanded Stacked (Composite Symbology not supported)	
Paper Cut	Cut, Feed cut	
Drawer Kick-Out	Supported	
Buzzer	Option (Pattern A ~ Pattern E, Error, No paper, Stop)	
Battery	Not supported	

*1 Only for Japanese model

*2 Differs depending on the Multilingual Model specifications.

TM-T81II

		80 mm
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)
Barcode		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

TM-T82

		58 mm	80 mm
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)
Barcode		UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF,CODABAR, CODE93, CODE128, GS1-128, GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Limited, GS1 DataBar Expanded	
Two-Dimensional Code		PDF417, QR Code, MaxiCode, GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional, GS1 DataBar Expanded Stacked (Composite Symbology not supported)	
Paper Cut		Cut, Feed cut	
Drawer Kick-Out		Supported	
Buzzer		Optional	
Battery		Not supported	

TM-T82II (ANK model / Multi-language model)

			80 mm
Resolution			203 dpi x 203 dpi (W x H)
Language			• ANK model • Simplified Chinese model • Traditional Chinese model • South Asian model
Print Width	Normal mode		576 dots
	42 Column Mode		546 dots
Characters in a Line	Font A	Normal mode	ANK: 48 characters Kanji *: 24 characters
		42 Column Mode	ANK: 42 characters Kanji *: 21 characters
	Font B	Normal mode	ANK: 64 characters
		42 Column Mode	ANK: 60 characters
Character Size	Font A	Normal mode	ANK: 12 dots x 24 dots (W x H) Kanji *: 24 dots x 24 dots (W x H)
		42 Column Mode	ANK: 13 dots x 24 dots (W x H) Kanji *: 26 dots x 24 dots (W x H)
	Font B	Normal mode	ANK: 9 dots x 17 dots (W x H)
		42 Column Mode	ANK: 9 dots x 17 dots (W x H)
Character Baseline	Font A		ANK: At the 21st dot from the top of the character Kanji *: 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)
Barcode			UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF,CODABAR, CODE93, CODE128, GS1-128, GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Limited, GS1 Databar Expanded
Two-Dimensional Code			PDF417, QR Code, MaxiCode, GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional, GS1 DataBar Expanded Stacked (Composite Symbology not supported)
Paper Cut			Cut, Feed cut

	80 mm
Drawer Kick-Out	Supported
Buzzer	Option (Pattern A ~ Pattern E, Error, No paper, Stop)
Battery	Not supported

* Only for Multi-language model

TM-T83II

		80 mm
Resolution		180 dpi x 180 dpi (W x H)
Language		Korean model
Print Width		512 dots
Characters in a Line	Font A	ANK: 42 characters Kanji: 21 characters
	Font B	ANK: 56 characters Kanji: 32 characters
Character Size	Font A	ANK: 12 dots x 24 dots (W x H) Kanji: 24 dots x 24 dots (W x H)
	Font B	ANK: 9 dots x 17 dots (W x H) Kanji: 16 dots x 16 dots (W x H)
Character Baseline	Font A	ANK: At the 21st dot from the top of the character Kanji: At the 21st dot from the top of the character
	Font B	ANK: At the 16th dot from the top of the character Kanji: 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)
Barcode		UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF,CODABAR, CODE93, CODE128, GS1-128, GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Limited, GS1 DataBar Expanded
Two-Dimensional Code		PDF417, QR Code, MaxiCode, GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional, GS1 DataBar Expanded Stacked (Composite Symbology not supported)
Paper Cut		Cut, Feed cut
Drawer Kick-Out		Supported
Buzzer		Option (Pattern A ~ Pattern E, Error, No paper, Stop)
Battery		Not supported

TM-T88V (ANK model / Multi-language model)

		58 mm	80 mm
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
Characters in a Line	Font A	ANK: 30 characters Kanji *: 15 characters	ANK: 42 characters Kanji *: 21 characters
	Font B	ANK: 40 characters Kanji *: 22 characters	ANK: 56 characters Kanji *: 32 characters
	Special font A*	30 characters	42 characters
	Special font B*	40 characters	56 characters
Character Size	Font A	ANK: 12 dots x 24 dots (W x H) Kanji *: 24 dots x 24 dots (W x H)	
	Font B	ANK: 9 dots x 17 dots (W x H) Kanji *: 16 dots x 16 dots (W x H)	
	Special font A*	12 dots x 24 dots (W x H)	
	Special font B*	9 dots x 24 dots (W x H)	
Character Baseline	Font A	ANK: At the 21st dot from the top of the character Kanji *: At the 21st dot from the top of the character	
	Font B	ANK: At the 16th dot from the top of the character Kanji *: At the 15th dot from the top of the character	
	Special font A*	At the 20th dot from the top of the character	
	Special font B*	At the 20th 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)
Barcode		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	

	58 mm	80 mm
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	Option (Pattern A ~ Pattern E, Error, No paper, Stop)	
Battery	Not Supported	

* Differs depending on the Multilingual Model specifications.

TM-T90II

		58 mm	80 mm
Resolution		203 dpi x 203 dpi (W x H)	
Language		Japanese model	
Print Width		420 dots	576 dots
Characters in a Line	Font A	35 characters	48 characters
	Font B	42 characters	57 characters
	Font C	52 characters	72 characters
Character Size	Font A	12 dots x 24 dots (W x H)	
	Font B	10 dots x 24 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 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 1662 dots (W x H)	576 dots x 1662 dots (W x H)
Page Mode Maximum Area		420 dots x 1662 dots (W x H)	576 dots x 1662 dots (W x H)
Barcode		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	
Paper Cut		Cut, Feed cut	
Drawer Kick-Out		Supported	
Buzzer		Supported	
Battery		Not supported	

TM-U220

		76 mm	69.5 mm	57.5 mm
Resolution	Single-density	80 dpi x 72 dpi (W x H)		
	Double-density	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	Single-density	200 dots	180 dots	150 dots
	Double-density	400 or 385[*1] half dots	360 half dots	300 or 297[*1] half dots
Characters in a Line	Font A	ANK: 33 characters Kanji [*2]: 25 characters	ANK: 30 characters Kanji [*2]: 22 characters	ANK: 25 characters Kanji [*2]: 18 characters
	Font B	ANK: 40 characters	ANK: 36 characters	ANK: 30 characters
Character Size	Font A	ANK: 4.5 dots x 9 dots (W x H) Kanji [*2]: 16 dots x 16 dots (W x H)		
	Font B	ANK: 3.5 dots x 9 dots (W x H)		
Character Baseline	Font A	ANK: Bottom of the characters Kanji [*2]: At the 15th dot from the top of the character		
	Font B	ANK: Bottom of the characters		
Default Line Feed Space		12 dots		
Color Specification		First color		
Page Mode Default Area		-		
Page Mode Maximum Area		-		
Barcode		Not supported		
Two-Dimensional Code		Not supported		
Paper Cut		Cut, Feed cut		
Drawer Kick-Out		Supported		
Buzzer		Not supported		
Battery		Not supported		

*1: DipSW2-1 = ON

*2: Differs depending on the Multilingual Model specifications.

addTextStyle (p.73) has the following restrictions.

- reverse parameter: Not supported

TM-U330

		76 mm	69.5 mm	57.5 mm
Resolution	Single-density	80 dpi x 72 dpi (W x H)		
	Double-density	160 dpi x 72 dpi (W x H)		
Language		Simplified Chinese model		
Print Width	120 dpi base	300 dots	270 dots	225 dots
	240 dpi base	600 dots	540 dots	450 dots
	180 dpi base	450 dots	405 dots	337 dots
Characters in a Line	Font A	ANK: 33 characters	ANK: 30 characters	ANK: 25 characters
	Font B	ANK: 42 characters	ANK: 38 characters	ANK: 32 characters
	Chinese (180/90 dpi)	16 characters	15 characters	12 characters
	Chinese (80 dpi)	22 characters	20 characters	16 characters
Character Size	Font A	ANK: 9 dots x 24 dots (W x H)		
	Font B	ANK: 7 dots x 24 dots (W x H)		
	Chinese	Kanji: 24 dots x 24 dots (W x H)		
Character Baseline	Font A	-		
	Font B	-		
	Chinese	-		
Default Line Feed Space		12 dots		
Color Specification		First color		
Page Mode Default Area		-		
Page Mode Maximum Area		-		
Barcode		Not supported		
Two-Dimensional Code		Not supported		
Paper Cut		Cut, No cut		
Drawer Kick-Out		Supported		
Buzzer		Not supported		
Battery		Not supported		

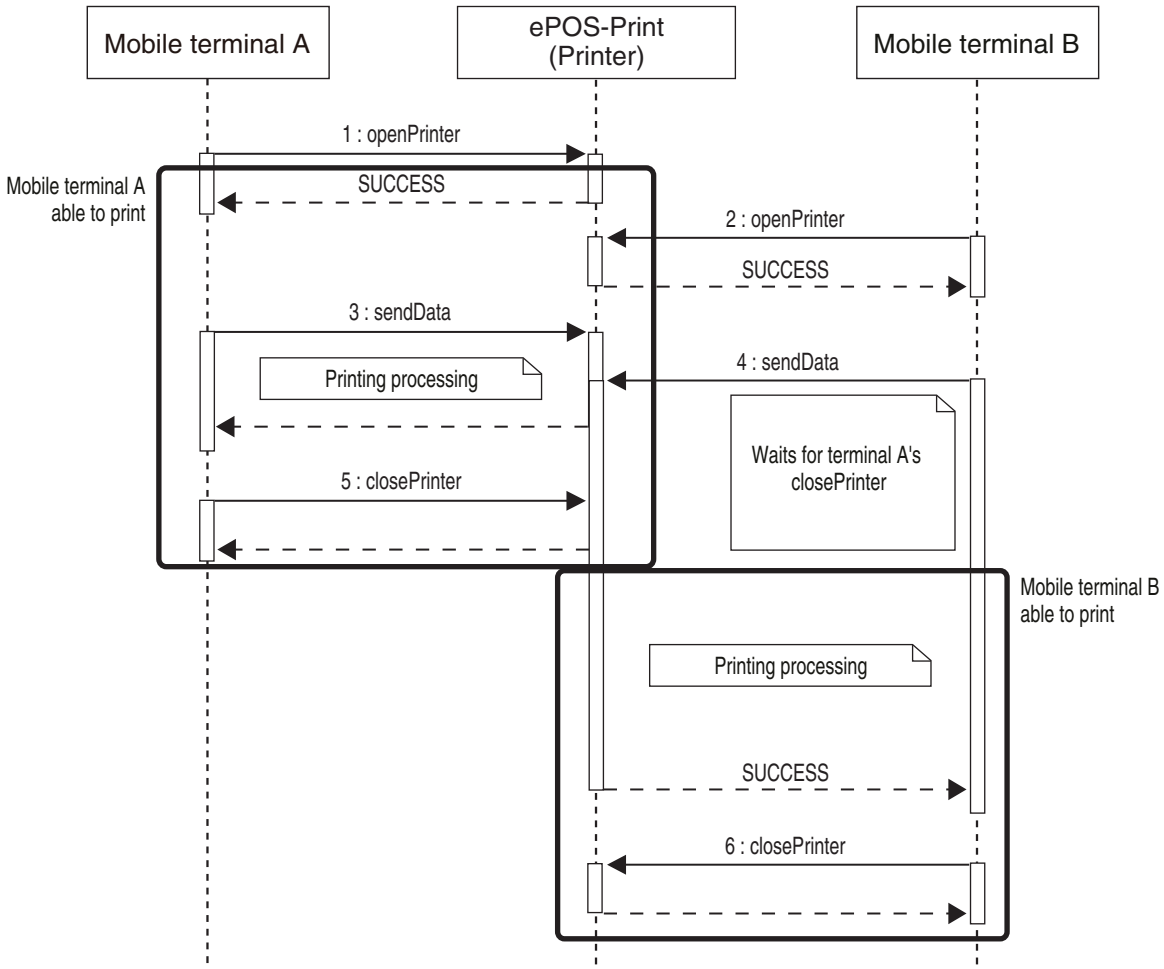
Cautions

If you Use the Printer from Multiple Mobile Terminals

If you use the printer from multiple mobile terminals, while you are using a particular terminal it will not be possible to print from the other ones. With Version 1.6.0 and later, if openPrinter processing has been initiated on one terminal when the printer is being used by another terminal, the openPrinter processing will wait for the other terminal's processing to end.
The chart below shows the flow of processing when a single printer is used from mobile terminal A and mobile terminal B.

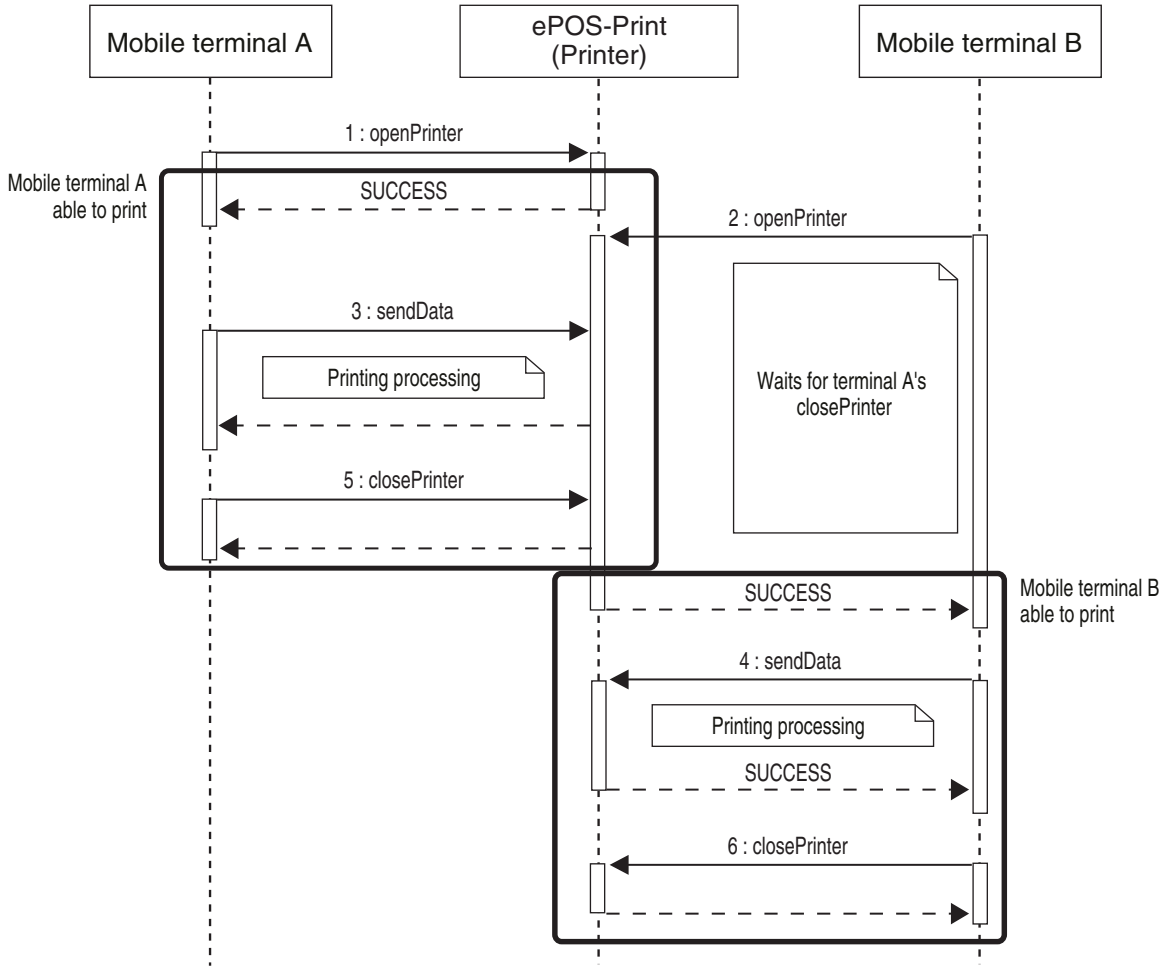
Version 1.5.0 and earlier

With Version 1.5.0 and earlier, mobile terminal B will wait for mobile terminal A's closePrinter processing to end before executing sendData processing.



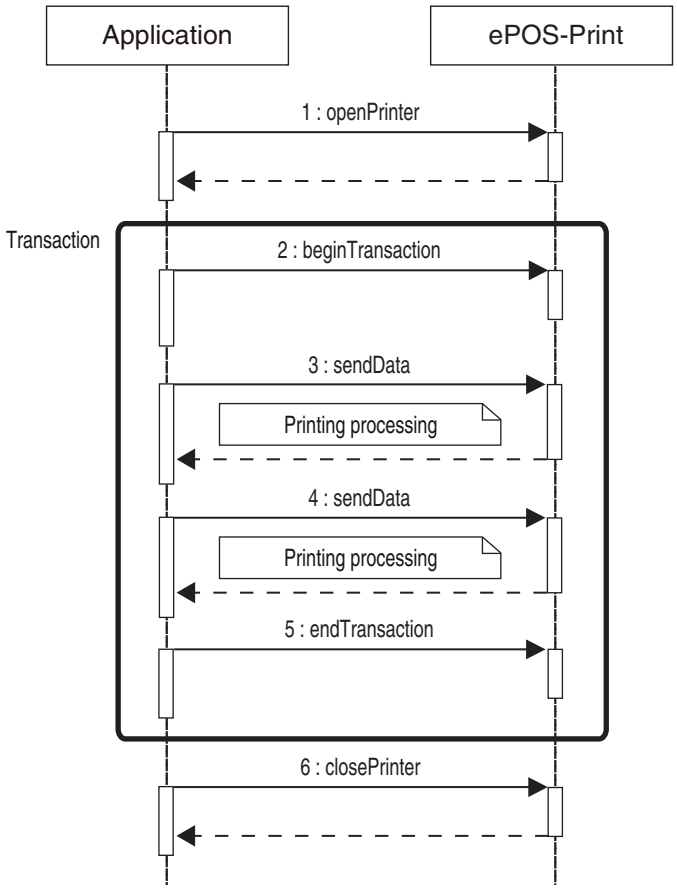
Version 1.6.0 and later

With Version 1.6.0 and later, mobile terminal B will wait for mobile terminal A's closePrinter processing to end before executing openPrinter processing.



To specify a transaction

Put the set of print processing to be carried out consecutively (such as a single receipt or a single coupon) between beginTransaction (p.131) and endTransaction (p.132).



Open Source Software License Agreement

In addition to proprietary Epson software, ePOS-Print SDK for Android uses open source software. For information about the open source software used by ePOS-Print SDK for Android, see the following URL:

ZXing(https://github.com/zxing/zxing)

ZXing is licensed based on Apache 2.0 license (http://www.apache.org/licenses/LICENSE-2.0.html).