EPSON
EXCEED YOUR VISION

ePOS-Print SDK for Windows Store apps User's Manual

Overview

Describes the features and development environment.

Sample Program

Describes how to use the sample program.

Programming Guide

Describes how to write programs in application development.

API Reference

Describes the APIs provided in ePOS-Print SDK for Windows Store apps.

Command Transmission/Reception

Describes the APIs for transmitting and receiving commands.

Appendix

Describes the specifications for printers used for the ePOS-Print SDK for Windows Store apps.

M00075005
Rev.F

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.

Microsoft®, Windows®, Visual Studio®, Visual C#®, and Visual Basic® are registered trademarks or trademarks of Microsoft Corporation in the United States 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.

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

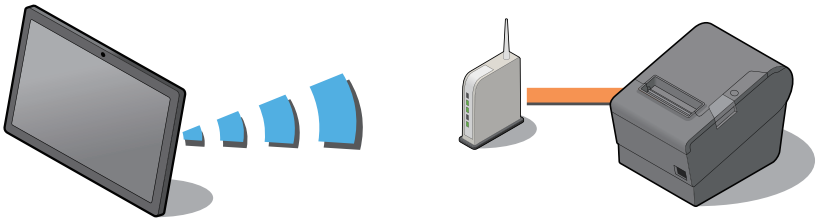
The manual is made up of the following sections:

Contents

Overview

This chapter describes the features of and the specifications for ePOS-Print SDK for Windows Store apps.

Overview of ePOS-Print SDK



The ePOS-Print SDK for Windows Store apps is an SDK aimed at development engineers who are developing Windows Store apps for printing on an Epson TM printer. Applications are developed using the APIs provided by ePOS-Print SDK.

ePOS-Print SDK provides ePOS-Print SDK for iOS for iOS applications as well as ePOS-Print SDK for Android for Android applications.

> APIs for transmitting/receiving commands to/from TM printers are also provided.
> A command transmission/reception API cannot be used with the ePOS-Print API, Print class. For details on the command transmission/reception APIs, refer to Command Transmission/Reception (p.191).

Features

- ❏ Allows printing to TM printers from Windows Store apps.
- ❏ Allows acquisition of TM printer status from Windows Store apps.

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.208).)
- ❏ 2D-Code printing
  (For 2D-Code that can be printed by each model, refer to Support Information by Printer (p.208).)
- ❏ Drawer kick function
- ❏ Buzzer function
- ❏ Paper layout setting
- ❏ Label / black mark paper feed setting
- ❏ ESC/POS command transmission
- ❏ Acquisition of response from printer (printing result / printer status / battery status)
- ❏ Compatible with Asian languages (simplified Chinese, traditional Chinese, Korean, Thai, Vietnamese)

Printer Search API

- ❏ Search for printers

Log Setting API

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

Operating Environment

OS

- ❏ Windows 10 (32 bit/64 bit)
- ❏ Windows 8.1 (32 bit/64 bit)

- For the latest version, refer to the README file.
- Windows RT (ARM) is not supported.

Printer

TM Printer	Interface		
	Wired LAN	Wi-Fi®	*Bluetooth*®
TM-m10 Ethernet	✔	-	-
TM-m10 Wi-Fi *	✔	✔	-
TM-m10 *Bluetooth*®	-	-	✔
TM-m30 Standard model *	✔	✔	-
TM-m30 *Bluetooth*® model *	✔	✔	✔
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-T88V	✔	✔	✔
TM-T90II	✔	✔	-
TM-U220 Series	✔	✔	-
TM-U330 Series	✔	✔	-

*: The Wi-Fi interface is disabled when the Ethernet cable is connected.

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

Development Environment

The following are necessary to develop a Windows Store apps.

❑ Visual Studio 2013

Development Language

- Visual C#
- Visual Basic .NET

The following development languages are not supported:

* Visual C++ (C++/CX)
* JavaScript

Contents in the Package

Package

File	Description
LibEposPrint.vsix	The ePOS-Print library. (Installer)
ePOS-Print_Sample_WinStoreApps.zip	A sample program file.
README.en.txt	A readme file.
README.jp.txt	A readme file. (The Japanese-language edition)
EULA.en.txt	A software license to use.
EULA.jp.txt	A software license to use. (The Japanese-language edition)
ePOS-Print_SDK_WinStoreApps_en_revx.pdf	This manual. Describes the ePOS-Print SDK for Windows Store app programming methods and APIs.
ePOS-Print_SDK_WinStoreApps_ja_revx.pdf	The Japanese-language edition of this manual. Describes the ePOS-Print SDK for Windows Store app programming methods and APIs.

Manual

The following manuals are available for ePOS-Print SDK for Windows Store apps.

❏ ePOS-Print SDK for Windows Store apps User's Manual (This Document)

❏ ePOS-Print SDK for Windows Store apps Application Development Setup Guide

Sample Program

For a Windows Store apps for TM printers developed using ePOS-Print SDK for Windows Store apps, the following program is available.

❏ ePOS-Print_Sample_WinStoreApps.zip

- Basic function sample (ePOSPrintSample)
- 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

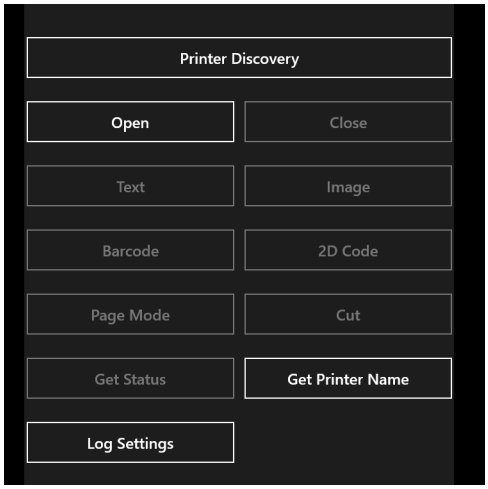
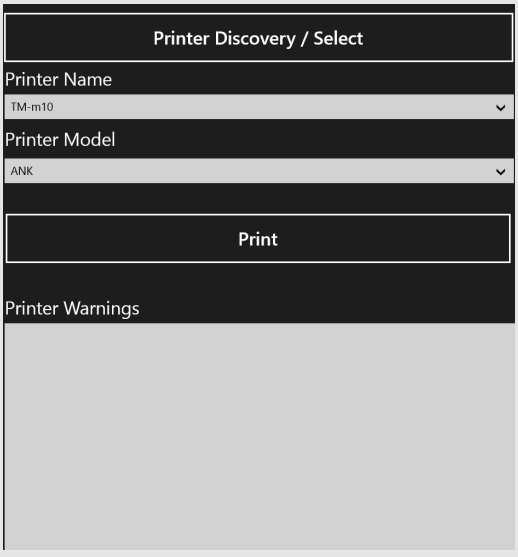
- A communication API (p.53) and command transmission/reception API (p.191) in the ePOS-Print APIs cannot be used for the same device at the same time.
- A maximum of 16 device ports can be opened in the same application at the same time.
- More than one port cannot be opened for the same device at the same time.
- ePOS-Print SDK for Windows Store apps cannot co-exist with the EPSON Advanced Printer Driver or OPOS driver:
- In TCP connection, if communication is performed immediately after the printer is powered off, an error may not occur.
- Cannot be used in an environment where multiple network adapters are enabled.
- If the device goes into sleep mode while communicating with a printer via *Bluetooth*®, the connection will be lost. Recovery from that state will be made automatically if reconnection is possible.
- If the application becomes inactive while communicating with a printer via *Bluetooth*®, the connection may be lost. Recovery from that state will be made automatically if reconnection is possible.
- You may be requested to enter a passcode during *Bluetooth*® pairing. In that case, enter "0000" or "4254".

Sample Program

This chapter describes how to use the sample program.

- For an Windows Store apps for TM printers developed using ePOS-Print SDK, the following program is available.
- The sample program is provided as a Windows Store apps project for use with Visual Studio 2013, including the Visual C#/ Visual Basic .NET source files.

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

Usage Environment

Development Environment

- Visual Studio 2013

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

Printer

- TM printer supported in ePOS-Print SDK.

Target device

- Device with a developer license

You need to have a Microsoft account to get a developer license.

Environmental Construction

Follow the procedures below to use the sample program.

How to start the sample program by sideloading it on the target terminal is described here.

1. Install ePOS-Print SDK for Windows Store apps (LibEposPrint.vsix).
2. Explode the sample program zip file into an arbitrary directory.
3. Select the development language for the sample program and start the sln file.
4. Select (Add Reference) in the (PROJECT) menu.
5. The Reference Manager screen appears. Select (Windows)-(Extensions).
6. Select the checkboxes for the following SDKs and click (OK).
   - Epson ePOS-Print SDK
   - Microsoft Visual C++ 2013 Runtime Package for Windows
7. Select as (Store)-(Store Create App Packages) in the (PROJECT) menu.

To run the sample program in the development environment, execute [Rebuild Solution] and [Deploy Solution] in the [BUILD] menu, and start the sample program from the start menu on Windows.

8. The "Create App Packages" screen appears. Select (No) and click (Next).
9. The "Select and Configure Packages" screen appears.
   Set the package to create and click (Create).

ePOS-Print SDK for Windows Store apps is developed using C++/CS (unmanaged code). For that reason, be sure to specify either "x86" or "x64" for [Select the packages to create and the solution configuration mappings].

10 The "Package Creation Completed" screen appears.
Check the output location and click (OK).

11 Open the output location directory and copy the folder into an arbitrary location in the target terminal.

12 In the target terminal, select the "Add-AppDevPackage.ps1" file in the copied folder and select (Run with PowerShell) from Context Menu.

When Windows Power Shell executes, a message appears, prompting you to get a developer license. Use your Microsoft account to get a developer license.

13 As instructed by the script, sideload the sample application onto the target terminal.

14 Select a project name from the start menu on Windows of the target terminal.
The sample program starts.

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.21)
- Acquisition of Printer Model Name (p.28)

❏ ePOSReceiptPrintSample

- Sample receipt data Printing (p.29)

Search for printers and printing

Use the sample program as follows:

1. Start the sample program. For details, refer to Environmental Construction (p.19).
2. Search for printers. Push (Printer Discovery) on the main screen.
   Select (Device Type) to display a list of IP addresses/Mac addresses for the printers retrieved on the (Printer List).
3. Push the IP address/Mac address of the printer in use from the (Printer List) displayed.
4. Open the printer's port. Push (Open) on the main screen.
   The "Device Type" and "IP Address/Mac Address" 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. Push (Open).

7 Execute the following processes:

Process	Description
Text printing	Push (Text) on the main screen. For details, refer to Text printing (p.23).
Graphic printing	Push (Image) on the main screen. For details, refer to Graphic printing (p.23).
Barcode printing	Push (Barcode) on the main screen. For details, refer to Barcode printing (p.24).
2D-Code printing	Push (2D Code) on the main screen. For details, refer to 2D-Code printing (p.24).
Printing in page mode	Push (Page Mode) on the main screen. For details, refer to Printing in page mode (p.25).
Paper cutting	Push (Cut) on the main screen. For details, refer to Paper cutting (p.25).
Log output setting	Push (Log Settings) on the main screen. For details, refer to Log output setting (p.25).
Printer status acquisition	Push (Get Status) on the main screen.

8 The following execution results will be displayed. Check the execution results and click (OK).

- Process execution result (error status / printer status / battery status)
  For details, refer to Process execution result (p.26).
- Method (API) execution error
  For details, refer to Method (API) execution error (p.27).

9 When all processing is finished, Push (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. Push (Print) to print.

Graphic printing

Execute the graphic printing according to the following procedure:

1. Push (Select Image) to select an image file to print.
2. Push (Color Mode) to select the tone.

[Gray 16] is supported by models that support multiple tones. For details, see AddImageAsync (p.81).

3. Push (Halftone Method) to select the halftone treatment method.
4. Push (Brightness) and input a value to specify brightness.
5. Push (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 Push (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 Push (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. Push (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. Push (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. Push (Setting) to enable the log output settings.
5. After printing, check the log file.
   For details, refer to SetLogSettings (p.186).

Execution result

Process execution result

Any of the following will be displayed:

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

String displayed	Description
HR_S_OK	Succeeded.
HR_E_INVALIDARG	• An invalid parameter was passed. • An unsupported model or language of use has been specified.
HR_E_ACCESSDENIED	• The open process failed. • Failed to connect to the device. • Offline.
HR_E_ABORT	The process was timed out.
HR_E_PENDING	Failed to execute the process.
HR_E_OUTOFMEMORY	Could not secure the memory required for the process.
HR_E_FAIL	• Used in an illegal manner. • 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-P20, TM-P60, TM-P60II, TM-P80)
BATTERY_OFFLINE	Battery offline (TM-P20, 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.49) .

Method (API) execution error

Any of the following will be displayed:

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

String displayed	Description
HR_E_INVALIDARG	• An invalid parameter was passed. • An unsupported model or language of use has been specified.
HR_E_ACCESSDENIED	• The open process failed. • Failed to connect to the device. • Printer is offline.
HR_E_PENDING	Failed to execute the process.
HR_E_ABORT	All data couldn't be sent during the specified time.
HR_E_OUTOFMEMORY	Could not secure the memory required for the process.
HR_E_FAIL	• Used in an illegal manner. • 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.191).

Use the following procedure:

1. Start the sample program. For details, refer to Environmental Construction (p.19).
2. Search for printers. Push (Printer Discovery) on the main screen.
   Select (Device Type) to display a list of IP addresses/Mac addresses for the printers retrieved on the (Printer List).
3. Push the IP address/Mac address of the printer in use from the (Printer List) displayed.
4. Push (Get Printer Name) on the main screen.
5. Push (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.

Sample receipt data Printing

Use the following procedure:

1. Start the sample program. For details, refer to Environmental Construction (p.19).
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.

- Descriptions made using source code in this chapter are based on Visual C#. For any language other than the above, read such descriptions in ways that suit the relevant language.
- For ways to construct a development environment for Windows Store apps that use ePOS-Print SDK for Windows Store apps, please refer to the "ePOS-Print SDK for Windows Store apps Application Development Setup Guide".

How to Incorporate the ePOS-Print SDK for Windows Store apps

This section explains how to incorporate the ePOS-Print SDK for Windows Store apps.

Explanations are made based on Visual Studio 2013.

Incorporate the SDK using following procedures.

1 Create a new project in Visual Studio 2013.

2 Declare a communication function in the "Package.appxmanifest" of the created project.

- TCP declaration
  1. Select (View Designer) in the context menu for "Package.appxmanifest" to display Package.appxmanifest.
  2. Select (Capabilities) to display the "Capabilities" screen.
  3. Select the checkboxes for "Internet (Client)" and "Private Networks (Client & Server)", and save the data.
- *Bluetooth*® declaration
  1. Select (View Designer) in the context menu for "Package.appxmanifest" to display Package.appxmanifest.
  2. Add the following bold-character portions between the <Capabilities> tags and save the data.

```
<Capabilities>
  <Capability Name="internetClient" />
  <DeviceCapability Name="proximity" />
  <m2:DeviceCapability Name="bluetooth.rfcomm">
    <m2:Device Id="any">
      <m2:Function Type="name:serialPort" />
    </m2:Device>
  </m2:DeviceCapability>
</Capabilities>
```

3 Select (Add Reference) in the (PROJECT) menu.

4 The Reference Manager screen appears. Select (Windows)-(Extensions).

5 Select the checkboxes for the following SDKs and click (OK).

- Epson ePOS-Print SDK
- Microsoft Visual C++ 2013 Runtime Package for Windows

Install Epson ePOS-Print SDK using LibEposPrint.vsix.
For details, see "ePOS-Print SDK for Windows Store apps Application Development Setup Guide".

6 Select (Configuration Manager) in the (BUILD) menu.

7 The "Configuration Manager" screen appears. Specify "x86" or "x64" for the (Active solution platform) setting and click (Close).

ePOS-Print SDK for Windows Store apps is developed using C++/CS (unmanaged code).
For that reason, be sure to specify either "x86" or "x64" for [Active solution platform].

8 Write the import definition in the source file for the application you want to use. See the following:

- Visual C#: **`using LibEposPrint;`**
- Visual Basic: **`Imports LibEposPrint`**

ePOS-Print SDK

Print Mode

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

Standard mode

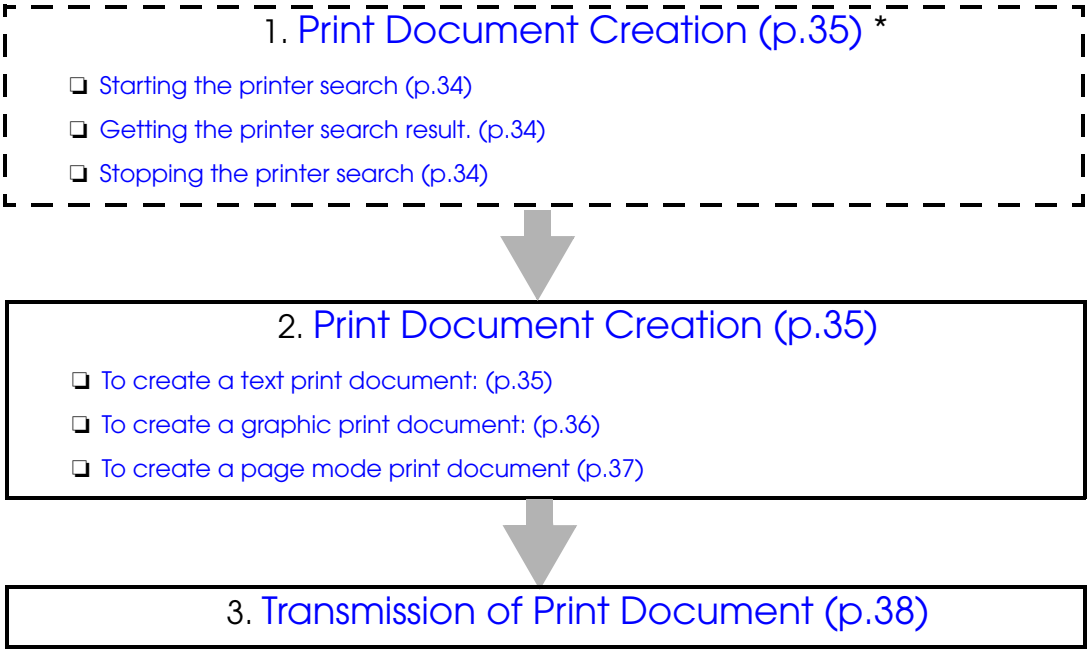
In standard mode, characters are printed line by line. The line feed space is adjusted based on the font size and the height of images, barcodes, etc. This mode is suitable for the type of printing such as printing receipts that requires the paper length to change according to the print space.

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.



* 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.40).

Printer Selection

Starting the printer search

Use the Finder class's StartAsync (p.180) to start searching for printers. Please refer to the following code.

```
//Start search
try
{
    //Wi-Fi/Ethernet device
    await Finder.StartAsync(IoDevType.TCP, "255.255.255.255");
    //Bluetooth device
    await Finder.StartAsync(IoDevType.BLUETOOTH, "");
}
//Exception handling
catch (Exception ex)
{
    if (ex.HResult == IoHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

Getting the printer search result.

Use the Finder class's GetDeviceInfoList (p.183) to get the result of the printer search.
Please refer to the following code.

```
DeviceInfo[] list = null;

//Get device list
try
{
    list = Finder.GetDeviceInfoList(IoFilterOption.DEFAULT);
}
//Exception handling
catch (Exception ex)
{
    if (ex.HResult == IoHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

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 the StartAsync.

Stopping the printer search

Use the Finder class's StopAsync (p.182) to stop searching for printers. Please refer to the following code.

```
//Stop search
try
{
    await Finder.StopAsync();
}
//Exception handling
catch (Exception ex)
{
    if (ex.HResult == IoHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

Print Document Creation

Create a print document using the Builder class (p.51).
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.

```
Builder builder = null;

try
{
    //Initialize a Builder class instance
    builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    //Create a print document
    builder.AddTextLang(Lang.LANG_EN);
    builder.AddTextSmooth(Smooth.SMOOTH_TRUE);
    builder.AddTextFont(Font.FONT_A);
    builder.AddTextSize(3, 3);
    builder.AddText("Hello World!\n");
    builder.AddCut(Cut.CUT_FEED);
}
catch (Exception ex)
{
    int errCode = ex.HResult;
}
```

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.65).

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

```
Builder builder = null;

try
{
    //Initialize a Builder class instance
    builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    //Create a print document
    //<Configure the print character settings>
    builder.AddTextLang(Lang.LANG_EN);
    builder.AddTextSmooth(Smooth.SMOOTH_TRUE);
    builder.AddTextFont(Font.FONT_A);
    builder.AddTextSize(4, 4);
    builder.AddTextStyle(Reverse.REVERSE_FALSE, Underline.UNDERLINE_FALSE,
                         Emphasis.EMPHASIS_TRUE, Color.UNSPECIFIED);
    //<Specify the print data>
    builder.AddText("Hello World!\n");
    builder.AddCut(Cut.CUT_FEED);
}
catch (Exception ex)
{
    int errCode = ex.HResult;
}
```

To create a graphic print document:

To create a graphic print document, for graphics, store the Windows.Grapics.Imaging.BitmapDecoder class in the command buffers with AddImageAsync (Previous format) (p.85) of the Builder class.
Use the programming example below for your reference.

```
using Windows.Storage;
using Windows.Storage.Streams;
using Windows.Graphics.Imaging;

Builder builder = null;

private IRandomAccessStreamWithContentType m_selectStream = null;
Builder builder = null;

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

    //Create a print document
    StorageFile imageFile = await Windows.ApplicationModel.Package.Current.
                            InstalledLocation.GetFileAsync("File name of the graphic");
    IRandomAccessStreamWithContentType stream = await imageFile.OpenReadAsync();
    BitmapDecoder imageData = await BitmapDecoder.CreateAsync(stream);

    await builder.AddImageAsync(imageData, 0, 0, (int)imageData.PixelWidth,
                                (int)imageData.PixelHeight, Color.DEFAULT, Mode.DEFAULT,
                                Halftone.DEFAULT, Builder.PARAM_DEFAULT);
    builder.AddCut(Cut.CUT_FEED);
}
catch (Exception ex)
{
    int errCode = ex.HResult;
}
```

For ways of graphic printing, you can also print the graphics registered in the printer's NV memory. For details, refer to AddLogo (p.90).

To create a page mode print document

The page mode starts by storing AddPageBegin (p.105) of the Builder class into a command buffer. Store the print area (AddPageArea (p.109)) and the print start position (AddPagePosition (p.113)) 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.107) 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.65).

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

```
Builder builder = null;

try
{
    //Initialize a Builder class instance
    builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_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(Lang.LANG_EN);
    builder.AddTextSmooth(Smooth.SMOOTH_TRUE);
    builder.AddTextFont(Font.FONT_A);
    builder.AddTextSize(2, 2);
    builder.AddTextStyle(Reverse.REVERSE_FALSE, Underline.UNDERLINE_FALSE,
                         Emphasis.EMPHASIS_TRUE, Color.UNSPECIFIED);
    //<Specify the print data>
    builder.AddText("Hello World!\n");
    //<The page mode ends>
    builder.AddPageEnd();
    builder.AddCut(Cut.CUT_FEED);
}
catch (Exception ex)
{
    int errCode = ex.HResult;
}
```

Transmission of Print Document

Send a print document using the Print class (p.53). Create a Print class using the constructor for it, use SendDataAsync 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.57) is executed. Execute clearCommandBuffer after the success of SendDataAsync (p.141).

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

Use the programming example below for your reference.

```
Print printer = null;
Builder builder = null;
try
{
    //Initialize a Print class instance
    printer = new Print();

    //Initialize a Builder class instance
    builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);

    //Create a print document
    //<Configure the print character settings>
    builder.AddTextLang(Lang.LANG_EN);
    builder.AddTextFont(Font.FONT_A);
    builder.AddTextSize(4, 4);
    builder.AddTextStyle(Reverse.REVERSE_FALSE, Underline.UNDERLINE_FALSE,
                         Emphasis.EMPHASIS_TRUE, Color.UNSPECIFIED);

    //<Specify the print data>
    builder.AddText("Hello World!\n");
    builder.AddCut(Cut.CUT_FEED);

    //Start communication with the printer
    //<Wi-Fi/Ethernet device>
    await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, , "192.168.192.168",
                                   Monitoring.MONITORING_FALSE, Print.PARAM_DEFAULT);
    //<Bluetooth device>
    await printer.OpenPrinterAsync(DevType.DEVTYPE_BLUETOOTH, , "00:00:12:34:56:78",
                                   Monitoring.MONITORING_FALSE, Print.PARAM_DEFAULT);

    //Send a print document
    PrinterStatus sendStatus = await printer.SendDataAsync(builder, 100000);

    //End communication with the printer
    await printer.ClosePrinterAsync();
}
catch(Exception ex){

    //Acquired the error code
    int errCode = ex.HResult;
    //Acquires the printer status at the time of error occurred
    PrinterStatus errStatus = printer.GetPrinterStatus(ex.HResult);

    //Exception processing
}
```

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 SendDataAsync is executed. The set value is initialized each time SendDataAsync is executed. Refer to the following:

Example:

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

Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
builder.AddText("Hello, World!\n");                    String for which the addTextFont setting is disabled
builder.AddTextFont(Font.FONT_A);
builder.AddText("Hello, World!\n");                    String for which the addTextFont setting is enabled (FONT_A)
PrinterStatus sendStatus = await printer.SendDataAsync(builder, 100000);
builder.AddText("Hello, World!\n");                    String for which the addTextFont setting is disabled
builder.AddTextFont(Font.FONT_B);
builder.AddText("Hello, World!\n");                    String for which the addTextFont setting is enabled (FONT_B)
sendStatus = await printer.SendDataAsync(builder, 100000);
```

Printing After Checking the Printer Status

To ensure successful print operation, print after checking the printer status.
Acquire the printer status in GetStatusAsync (p.176), and print it out when the printer is online.
Use the programming example below for your reference.

```
int retry = 0;
Builder builder = null;

try
{
     //Create a print document
    builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    builder.AddText("Hello World!\n");
    builder.AddCut(Cut.CUT_FEED);
}                                                                              (1)
catch (Exception ex)
{
    builder = null;
    return;
}

for (retry = 0; retry < 3; retry++)
{
    int errCode = 0;
    Print printer = null;
    PrinterStatus status;

    try {
         //Initialize a Print class instance
        printer = new Print();
         //Start communication with the printer
        await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                                       Monitoring.MONITORING_FALSE, Print.PARAM_DEFAULT);
    }
    catch (Exception ex) {
        printer = null;
        continue;
    }

    try {
         //<Get printer status>                                              (2)
        status = await printer.GetStatusAsync();

        if ((status.printerStatus & Print.ST_OFF_LINE) != Print.ST_OFF_LINE) {
            //Send a print document                                          (3)
             status = await printer.SendDataAsync(builder, 10000);
        }

        else if ((status.printerStatus & Print.ST_OFF_LINE) == Print.ST_OFF_LINE) {
            ;
        }
        else {                                                             (4)
            ;
        }
    }
    catch (Exception ex) {
         //<Acquired the error code>
        errCode = ex.HResult;
         //Acquires the printer status at the time of error occurred
        PrinterStatus errStatus = printer.GetPrinterStatus(ex.HResult);
         //Exception processing
    }

    try {
        await printer.ClosePrinterAsync();
        printer = null;
    }
    catch (Exception ex) {
        ;
    }
    if (errCode != EposHResult.HR_E_ACCESSDENIED) {
        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 the ePOS-Print SDK, the printer status can be automatically notified to the application by means of callback. Refer to the following.

```
//Registration of callback method for giving notification of printer status
private void Status_Change_Event(string deviceName, int status)
{
  ///Process///
}

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

    try {
    //Register the notification destination of printer status changes
       StatusChangeEvent statusChangeEvent = new StatusChangeEvent(Status_Change_Event);
       printer.SetStatusChangeEventCallback(statusChangeEvent);
    //Start communications with the printer and monitoring of the printer status
       await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, , "192.168.192.168",
                                      Monitoring.MONITORING_TURE, Print.PARAM_DEFAULT);
          ///Process///

    }
    catch(Exception ex)
    {
      //Acquired the error code
      int errCode = ex.HResult;

      //Exception processing
    }

}
```

(1)/(4)

(2)

(3)

1 Implement the notification destination method when events occur.

> In the above description, the callback method, which notifies the printer status at the intervals specified in OpenPrinterAsync (p.133), is defined.
> ePOS-Print SDK has callback method according to each printer status, for example, events such as cover open and drawer open. Use these according to the desired purpose of use. See the Event List (p.43) for the callback method that can be used with ePOS-Print SDK.

2 Register the printer status notification destination.

3 Use OpenPrinterAsync (p.133) to start monitoring of the printer status.

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

> When printer status notification is ended, it ends on the ClosePrinterAsync (p.139) of the Print class.

Event List

For details on the callback method, refer to API Reference (p.51), which explains the callback method registration API.

Event	Callback method registration API
Printer status notification	SetStatusChangeEventCallback (p.148)
Online notification	SetOnlineEventCallback (p.150)
Offline notification	SetOfflineEventCallback (p.152)
Power off notification	SetPowerOffEventCallback (p.154)
Cover close notification	SetCoverOkEventCallback (p.156)
Cover open notification	SetCoverOpenEventCallback (p.158)
Paper OK notification	SetPaperOkEventCallback (p.160)
Paper near end notification	SetPaperNearEndEventCallback (p.162)
Paper end notification	SetPaperEndEventCallback (p.164)
Drawer close notification	SetDrawerClosedEventCallback (p.166)
Drawer open notification	SetDrawerOpenEventCallback (p.168)
Battery low notification	SetBatteryLowEventCallback (p.170)
Battery OK notification	SetBatteryOkEventCallback (p.172)
Battery status notification	SetBatteryStatusChangeEventCallback (p.174)

Exception handling

In ePOS-Print SDK for Windows Store apps, a Windows system exception is generated when an error occurs to notify the calling side of such an error. And if an exception occurs when print data is sent, the printer status will be acquired.
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.46).
Printer status	Status of the printer when print data was sent. For details, refer to Printer Statuses and Actions to Take (p.47).
Battery status	Status of the printer's battery when print data was sent. For details, refer to Battery Status (p.49).

The printer status and the battery status can be automatically sent to the application through the callback method also when any status change occurs. For details, refer to Automatic Acquisition of Printer Status (p.42).

Steps for Handling

ePOS-Print API

Use the programming example below for your reference.

```
Print printer = null;
Builder builder = null;
try
{
    //Initialize a Print class instance
    printer = new Print();

    //Initialize a Builder class instance
    builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    //Create a print document
    builder.AddText("Hello World!\n");
    builder.AddCut(Cut.CUT_FEED);

    //Send a print document
    //<Wi-Fi/Ethernet device>
    await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, , "192.168.192.168",
                                   Monitoring.MONITORING_FALSE, Print.PARAM_DEFAULT);
    //<Bluetooth device>
    await printer.OpenPrinterAsync(DevType.DEVTYPE_BLUETOOTH, , "00:00:12:34:56:78",
                                   Monitoring.MONITORING_FALSE, Print.PARAM_DEFAULT);

    //<Send a data>
    PrinterStatus sendStatus = await printer.SendDataAsync(builder, 100000);

    //End communication with the printer
    await printer.ClosePrinterAsync();
}
catch(Exception ex){

    //Acquired the error code
    int errCode = ex.HResult;
    //Acquires the printer status at the time of error occurred
    PrinterStatus errStatus = printer.GetPrinterStatus(ex.HResult);

    //Exception processing
}
```

Search API

Use the programming example below for your reference.

```
DeviceInfo[] list = null;

//Start search
try
{
    //<Wi-Fi/Ethernet device>
    await Finder.StartAsync(IoDevType.TCP, "255.255.255.255");
    //<Bluetooth device>
    await Finder.StartAsync(IoDevType.BLUETOOTH, "");

    //Acquire a list of devices
    list = Finder.GetDeviceInfoList(IoFilterOption.DEFAULT);

}
catch (Exception ex)
{
    //Exception processing
    if (ex.HResult == IoHResult.HR_E_FAIL)
    {
     ///Process///
    }
}
```

Error Statuses and Actions to Take

In ePOS-Print SDK for Windows Store apps, a Windows system exception with any of the following error statuses is returned.
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
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed. <Example> • An invalid parameter such as null was passed. • A value outside the supported range was specified. • An unsupported model name or language specification was specified.	• The parameter was specified incorrectly. Check the parameter. • Cannot be used for unsupported models.
HR_E_ACCESSDENIED (0x80070005)	• Open processing failed. • Failed to connect to device. • The printer is offline. <Example> • Could not connect to the designated printer. • Failed to send the data to the printer.	• Check the Windows device and the printer. (The printer's power condition, communication condition, etc.) • Eliminate the cause that makes the printer offline. (Such as cover open and no paper.)
HR_E_ABORT (0x80004004)	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.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate the necessary memory for processing.	End the unneeded applications.
HR_E_PENDING (0x8000000A)	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.
HR_E_FAIL (0x80004005)	Illegal method used. An unspecified error occurred. <Example> When the printer was not opened, an API for sending a command to the printer was called.	• Use the API in a proper way. Refer to Programming Flow (p.33). • Check the communication settings of the Windows device. (Wi-Fi connection setting, *Bluetooth*® connection setting, etc.) • Check that there is no problem with the execution environment.

Printer Statuses and Actions to Take

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 SendDataAsync.

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.49) .
Lower 8 bits	Battery status exclusive by model For details, refer to Support Information by Printer (p.208).

"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 Windows Store apps.

ePOS-Print API

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

- Builder class (p. 51)
- Print class (p. 53)

The APIs that you can use and the settings that you can designate vary based on the printer. For details, refer to Support Information by Printer (p.208).

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.	54
Clearing command buffers	ClearCommandBuffer	Clears the command buffers added by APIs.	57
Text	AddTextAlign	Adds a tag for the text alignment setting.	58
	AddTextLineSpace	Adds a tag for the line feed space setting.	60
	AddTextRotate	Adds a tag for the text rotation setting.	61
	AddText	Adds a tag for printing text.	63
	AddTextLang	Adds a tag for the target language setting.	65
	AddTextFont	Adds a tag for the text font setting.	67
	AddTextSmooth	Adds a tag for the text smoothing setting.	69
	AddTextDouble	Adds a tag for specifying the double-sized text setting.	71
	AddTextSize	Adds a tag for the text scale setting.	73
	AddTextStyle	Adds a tag for the text style setting.	75
	AddTextPosition	Adds a tag for specifying the print position of text.	77
Paper Feed	AddFeedUnit	Adds a tag for paper feeding (in dots).	79
	AddFeedLine	Adds a tag for paper feeding (in lines).	80
	AddFeedPosition	Adds a tag for label / black mark paper feeding.	125
Graphic	AddImageAsync (For image compression)	Compresses image data and adds them to the command buffer.	81
	AddImageAsync	Adds multiple tone raster image printing to the command buffer.	85
	AddLogo	Adds a tag for an NV logo to be printed.	90
Barcode	AddBarcode	Adds a tag for a barcode to be printed.	92
	AddSymbol	Adds a tag for a 2D-Code to be printed.	98

	API	Description	Page
Page mode	AddPageBegin	Adds a tag for switching to page mode.	105
	AddPageEnd	Adds a tag for finishing page mode.	107
	AddPageArea	Adds a tag for specifying the print area in page mode.	109
	AddPageDirection	Adds a tag for specifying the print direction in page mode.	111
	AddPagePosition	Adds a tag for specifying the print position in page mode.	113
	AddPageLine	Adds a tag for drawing a line in page mode.	115
	AddPageRectangle	Adds a tag for drawing a rectangle in page mode.	117
Cut	AddCut	Adds a tag for paper cut.	119
Drawer kick-out	AddPulse	Adds a tag for the drawer kick-out.	121
Buzzer	AddSound	Sets the buzzer sounding cycle and adds it to the command buffer.	123
Paper Layout	AddLayout	Adds a tag for paper layout information.	127
Send Command	AddCommand	Adds a tag for inserting commands.	130

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.	132
OpenPrinterAsync	Starts communications with the printer and monitoring of the printer status	133
OpenPrinterAsync(Previous format)	Starts communications with the printer and monitoring of the printer status (Timeout cannot be set.)	136
ClosePrinterAsync	End communication with the printer.	139
SendDataAsync	Sends a command to the printer.	141
BeginTransactionAsync	Starts transaction.	144
EndTransactionAsync	Finishes transaction.	146
SetStatusChangeEventCallback	Registers the printer status notification destination	148
SetOnlineEventCallback	Registers the online event notification destination	150
SetOfflineEventCallback	Registers the offline event notification destination	152
SetPowerOffEventCallback	Registers the power off event notification destination	154
SetCoverOkEventCallback	Registers the cover close event notification destination	156
SetCoverOpenEventCallback	Registers the cover open event notification destination	158
SetPaperOkEventCallback	Registers the paper OK event notification destination	160
SetPaperNearEndEventCallback	Registers the paper near end event notification destination	162
SetPaperEndEventCallback	Registers the paper end event notification destination	164
SetDrawerClosedEventCallback	Registers the drawer close event notification destination	166
SetDrawerOpenEventCallback	Registers the drawer open event notification destination	168
SetBatteryLowEventCallback	Registers the battery low event notification destination	170
SetBatteryOkEventCallback	Registers the battery OK event notification destination	172
SetBatteryStatusChangeEventCallback	Registers the battery status notification destination	174
GetStatusAsync	Acquires the printer status and the battery status.	176
GetPrinterStatus	When an exception occurs and is thrown, acquires the printer status and the battery status	178

Builder class (Constructor)

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

When multiple Builder class instances are created, a large amount of memory is consumed. Create a minimum number of instances and use them while clearing command buffers by using ClearCommandBuffer (p.57).

Syntax

Visual C#

```
public Builder(System.String printerModel,
               LibEposPrint.ModelLang lang);
```

Visual Basic .NET

```
Public Sub Builder(printerModel As String,
                   lang As LibEposPrint.ModelLang)
```

Parameter

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

Set value	Description
"TM-m10"	TM-m10
"TM-m30"	TM-m30
"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-T90II"	TM-T90II
"TM-T88V"	TM-T88V
"TM-U220"	TM-U220
"TM-U330"	TM-U330

- lang : Specifies the language specifications for the printer.

Set value	Printer model	TM printer-separate setting																
		TM-m10	TM-m30	TM-P20	TM-P60	TM-P60II	TM-P80	TM-T20	TM-T20II	TM-T70	TM-T70II	TM-T81II	TM-T82	TM-T82II	TM-T88V	TM-T90II	TM-U220	TM-U330
ModelLang. MODEL_LANG_ANK	ANK	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	-	-	✔	✔	-	✔	-
ModelLang. MODEL_LANG_JAPANESE	Japanese	✔	✔	✔	-	-	-	✔	-	✔	✔	-	-	-	✔	✔	✔	-
ModelLang. MODEL_LANGCHINESE	Simplified Chinese	-	✔	✔	-	-	-	-	-	✔	✔	✔	✔	✔	✔	-	✔	✔
ModelLang. MODEL_LANG_TAIWAN	Traditional Chinese	✔	✔	✔	-	✔	✔	-	-	✔	✔	-	-	✔	✔	-	✔	-
ModelLang. MODEL_LANG_KOREAN	Korean	-	-	-	-	-	-	-	-	-	✔	-	-	-	✔	-	✔	-
ModelLang. MODEL_LANG_THAI	Thai	-	-	-	-	-	-	-	-	✔	✔	-	✔	✔	✔	-	✔	-
ModelLang. MODEL_LANG_SOUTHASIA	South Asian	-	-	✔	-	-	-	-	-	✔	✔	-	✔	✔	✔	-	✔	-

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	• Invalid parameter was passed. • An unsupported model name or unsupported language specifications were specified.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	An unspecified error occurred.

Example

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

- Visual C#

```
try
{
    Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    ///Process///
}
catch (Exception ex)
{
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim builder As Builder = New Builder("TM-T88V",ModelLang.MODEL_LANG_ANK)
    `///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       `///Process///
    End If
    `///Process///
End Try
```

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

Visual C#

```
public void ClearCommandBuffer();
```

Visual Basic .NET

```
Public Sub ClearCommandBuffer()
```

Example

If you are clearing the command buffer:

- Visual C#

```
Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
///Process///

try
{
    builder.ClearCommandBuffer();
    builder = null;
}
catch (Exception ex)
{
    builder = null;
}
```

- Visual Basic .NET

```
Dim builder As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
'///Process///

Try
    builder.ClearCommandBuffer()
    builder = Nothing
Catch ex As Exception
    builder = Nothing
End Try
```

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.113) instead of this API to set the alignment.

Syntax

Visual C#

```
public void AddTextAlign(LibEposPrint.Align align);
```

Visual Basic .NET

```
Public Sub AddTextAlign(align As LibEposPrint.Align)
```

Parameter

- align : Specifies the text alignment.

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

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	An unspecified error occurred.

Example

To set alignment to the center:

- Visual C#

```
try
{
    Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    builder.AddTextAlign(Align.ALIGN_CENTER);
    ///Process///
}
catch (Exception ex)
{
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim builder As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
    builder.AddTextAlign(Align.ALIGN_CENTER)
    `///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       `///Process///
    End If
    `///Process///
End Try
```

AddTextLineSpace

Adds the line feed space setting to the command buffer.

Syntax

Visual C#

```
public void AddTextLineSpace(int linespc);
```

Visual Basic .NET

```
Public Sub AddTextLineSpace(linespc As Integer)
```

Parameter

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

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	An unspecified error occurred.

Example

To set the line feed space to 30 dots:

- Visual C#

```
try
{
    Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    builder.AddTextLineSpace(30);
    ///Process///
}
catch (Exception ex)
{
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim builder As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
    builder.AddTextLineSpace(30)
    `///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
        `///Process///
    End If
    `///Process///
End Try
```

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.111) instead of this API function.

Syntax

Visual C#

```
public void AddTextRotate(LibEposPrint.Rotate rotate);
```

Visual Basic .NET

```
Public Sub AddTextRotate(rotate As LibEposPrint.Rotate)
```

Parameter

- rotate : Specifies whether to rotate text.

Set value	Description
Rotate.ROTATE_TRUE	Specifies rotated printing of text.
Rotate.ROTATE_FALSE (default)	Cancels rotated printing of text.

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	An unspecified error occurred.

Example

To set text rotation:

- Visual C#

```
try
{
    Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    builder.AddTextRotate(Rotate.ROTATE_TRUE);
    ///Process///
}
catch (Exception ex)
{
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim builder As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
    builder.AddTextRotate(Rotate.ROTATE_TRUE)
    `///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       `///Process///
    End If
    `///Process///
End Try
```

AddText

Adds the printing of text to the command buffer.

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

Syntax

Visual C#

```
public void AddText(System.String data);
```

Visual Basic .NET

```
Public Sub AddText(data As String)
```

Parameter

- data : Specify a character string to be printed.
  For the horizontal tab/line feed, Specifies the following:
  For the horizontal tab/line feed, use the following escape sequences:

String	Description
• Visual C# (Escape Sequence) \t • Visual Basic vbTab	Horizontal tab (HT)
• Visual C# (Escape Sequence) \n • Visual Basic vbCrLf	Line feed (CR+LF)
• Visual C# (Escape Sequence) \\ • Visual Basic \	Carriage return

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	An unspecified error occurred.

Example

To add character strings:

- Visual C#

```
try
{
    Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    builder.AddText("Hello,\t");
    builder.AddText("World\n");
    ///Process///
}
catch (Exception ex)
{
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim builder As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
    builder.AddText("Hello, " + vbTab)
    builder.AddText("World" + vbCrLF)
    `///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       `///Process///
    End If
    `///Process///
End Try
```

AddTextLang

Adds the language setting to a command buffer. Encodes the string specified by AddText (p.63) according to the language information specified by this API.
Specify the value according to the language specifications set for Builder class (Constructor) (p.54).

Syntax

Visual C#

```
public void AddTextLang(LibEposPrint.Lang lang);
```

Visual Basic .NET

```
Public Sub AddTextLang(lang As LibEposPrint.Lang)
```

Parameter

- lang : Specifies the target language.

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

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	An unspecified error occurred.

Example

To set the language as English:

- Visual C#

```
try
{
    Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    builder.AddTextLang(Lang.LANG_EN);
    ///Process///
}
catch (Exception ex)
{
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim builder As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
    builder.AddTextLang(Lang.LANG_EN)
    `///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       `///Process///
    End If
    `///Process///
End Try
```

AddTextFont

Adds the text font setting to the command buffer.

Syntax

Visual C#

```
public void AddTextFont(LibEposPrint.Font font);
```

Visual Basic .NET

```
Public Sub AddTextFont(font As LibEposPrint.Font)
```

Parameter

- font : Specifies the font.

Set value	Description	TM printer-separate setting												
		TM-m10	TM-m30	TM-P20	TM-P60/TM-P60II	TM-P80	TM-T20/TM-T20II	TM-T70/TM-T70II	TM-T81III	TM-T82/TM-T82II	TM-T88V	TM-T90II	TM-U220	TM-U330
Font.FONT_A (default)	Font A	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Font.FONT_B	Font B	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Font.FONT_C	Font C	✔	✔	✔	✔	-	-	-	-	-	-	✔	-	-
Font.FONT_D	Font D	-	-	✔	-	-	-	-	-	-	-	-	-	-
Font.FONT_E	Font E	-	-	✔	-	-	-	-	-	-	-	-	-	-

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	An unspecified error occurred.

Example

To set the font B:

- Visual C#

```
try
{
    Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    builder.AddTextFont(Font.FONT_B);
    ///Process///
}
catch (Exception ex)
{
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim builder As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
    builder.AddTextFont(Font.FONT_B)
    `///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       `///Process///
    End If
    `///Process///
End Try
```

AddTextSmooth

Adds the smoothing setting to the command buffer.

Syntax

Visual C#

```
public void AddTextSmooth(LibEposPrint.Smooth smooth);
```

Visual Basic .NET

```
Public Sub AddTextSmooth(smooth As LibEposPrint.Smooth)
```

Parameter

- smooth : Specifies whether to enable smoothing.

Set value	Description
Smooth.SMOOTH_TRUE	Specifies smoothing.
Smooth.SMOOTH_FALSE (default)	Cancels smoothing

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	An unspecified error occurred.

Example

To enable smoothing:

- Visual C#

```
try
{
    Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    builder.AddTextSmooth(Smooth.SMOOTH_TRUE);
    ///Process///
}
catch (Exception ex)
{
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim builder As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
    builder.AddTextSmooth(Smooth.SMOOTH_TRUE)
    `///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       `///Process///
    End If
    `///Process///
End Try
```

AddTextDouble

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

Syntax

Visual C#

```
public void AddTextDouble(LibEposPrint.Double dw,
                          LibEposPrint.Double dh);
```

Visual Basic .NET

```
Public Sub AddTextDouble(dw As LibEposPrint.DoubleSize,
                         dh As LibEposPrint.DoubleSize)
```

Parameter

- dw : Specifies the double-sized width.

Set value	Description
DoubleSize.DOUBLE_TRUE	Specifies the double-sized width.
DoubleSize.DOUBLE_FALSE (default)	Cancels the double-sized width
DoubleSize.UNSPECIFIED	Retains the current setting.

- dh : Specifies the double-sized height.

Set value	Description
DoubleSize.DOUBLE_TRUE	Specifies the double-sized height
DoubleSize.DOUBLE_FALSE (default)	Cancels the double-sized height
DoubleSize.UNSPECIFIED	Retains the current setting.

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

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	An unspecified error occurred.

Example

To set the size as double width and height:

- Visual C#

```
try
{
    Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    builder.AddTextDouble(DoubleSize.DOUBLE_TRUE, DoubleSize.DOUBLE_TRUE);
    ///Process///
}
catch (Exception ex)
{
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim builder As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
    builder.AddTextDouble([DoubleSize].DOUBLE_TRUE,[DoubleSize].DOUBLE_TRUE)
    `///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       `///Process///
    End If
    `///Process///
End Try
```

AddTextSize

Adds the text scale setting to the command buffer.

Syntax

Visual C#

```
public void AddTextSize(int width, int height);
```

Visual Basic .NET

```
Public Sub AddTextSize(width As Integer,
                       height As Integer)
```

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

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	An unspecified error occurred.

Example

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

- Visual C#

```
try
{
    Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    builder.AddTextSize(4, 4);
    ///Process///
}
catch (Exception ex)
{
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim builder As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
    builder.AddTextSize(4,4)
    `///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       `///Process///
    End If
    `///Process///
End Try
```

AddTextStyle

Adds the text style setting to the command buffer.

Syntax

Visual C#

```
public void AddTextStyle(LibEposPrint.Reverse reverse,
                         LibEposPrint.Underline ul,
                         LibEposPrint.Emphasis em,
                         LibEposPrint.Color color);
```

Visual Basic .NET

```
Public Sub AddTextStyle(reverse As LibEposPrint.Reverse,
                        ul As LibEposPrint.Underline,
                        em As LibEposPrint.Emphasis,
                        color As LibEposPrint.Color)
```

Parameter

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

Set value	Description
Reverse.REVERSE_TRUE	Specifies the inversion of black and white parts of characters.
Reverse.REVERSE_FALSE (default)	Cancels the inversion of black and white parts of characters.
Reverse.UNSPECIFIED	Retains the current setting.

- ul : Specifies the underline style.

Set value	Description
Underline.UNDERLINE_TRUE	Specifies underlining.
Underline.UNDERLINE_FALSE (default)	Cancels underlining.
Underline.UNSPECIFIED	Retains the current setting.

- em : Specifies the bold style.

Set value	Description
Emphasis.EMPHASIS_TRUE	Specifies emphasized printing of characters.
Emphasis.EMPHASIS_FALSE (default)	Cancels emphasized printing of characters.
Emphasis.UNSPECIFIED	Retains the current setting.

- color : Specifies the color.

Set value	Description
Color.COLOR_NONE	Characters are not printed.
Color.COLOR_1 (default)	First color
Color.COLOR_2	Second color
Color.COLOR_3	Third color
Color.COLOR_4	Fourth color
Color.UNSPECIFIED	Retains the current color setting

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	An unspecified error occurred.

Example

To set the underline style:

- Visual C#

```
try
{
    Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    builder.AddTextStyle(Reverse.UNSPECIFIED, Underline.UNDERLINE_TRUE,
                         Emphasis.UNSPECIFIED, Color.UNSPECIFIED);
    ///Process///
}
catch (Exception ex)
{
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim builder As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
    builder.AddTextStyle(Reverse.UNSPECIFIED, Underline.UNDERLINE_TRUE,
                         Emphasis.UNSPECIFIED, Color.UNSPECIFIED)
    `///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       `///Process///
    End If
    `///Process///
End Try
```

AddTextPosition

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

After executing this API, you cannot use AddTextAlign (p.58) or AddTextRotate (p.61).

Syntax

Visual C#

```
public void AddTextPosition(int x);
```

Visual Basic .NET

```
Public Sub AddTextPosition(x As Integer)
```

Parameter

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

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	An unspecified error occurred.

Example

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

- Visual C#

```
try
{
    Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    builder.AddTextPosition(120);
    ///Process///
}
catch (Exception ex)
{
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim builder As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
    builder.AddTextPosition(120)
    `///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       `///Process///
    End If
    `///Process///
End Try
```

AddFeedUnit

Adds paper feeding in dots to the command buffer.

Syntax

Visual C#

```
public void AddFeedUnit(int unit);
```

Visual Basic .NET

```
Public Sub AddFeedUnit(unit As Integer)
```

Parameter

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

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	An unspecified error occurred.

Example

To feed paper by 30 dots:

- Visual C#

```
try
{
    Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    builder.AddFeedUnit(30);
    ///Process///
}
catch (Exception ex)
{
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim builder As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
    builder.AddFeedUnit(30)
    `///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       `///Process///
    End If
    `///Process///
End Try
```

AddFeedLine

Adds paper feeding in lines to the command buffer.

Syntax

Visual C#

```
public void AddFeedLine(int line);
```

Visual Basic .NET

```
Public Sub AddFeedLine(line As Integer)
```

Parameter

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

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	An unspecified error occurred.

Example

To feed paper by 3 lines:

- Visual C#

```
try
{
    Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    builder.AddFeedLine(3);
    ///Process///
} catch (Exception ex) {
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim builder As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
    builder.AddFeedLine(3)
    `///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       `///Process///
    End If
    `///Process///
End Try
```

AddImageAsync

Adds raster image printing to the command buffer.
Prints the graphic in the Windows.Grapics.Imaging.BitmapDecoder class.
Out of the Windows.Grapics.Imaging.BitmapDecoder 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.
- This API is an asynchronous API. Wait until the process by this API ends to execute a new process.
- This API does not support cancellation of asynchronous operation by Windows.Foundation.IAsyncInfo.Cancel().
- To print a raster image at high speed, specify Align.ALIGN_LEFT for the AddTextAlign (p.58), 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

Visual C#

```
public Windows.Foundation.IAsyncAction AddImageAsync
            (Windows.Grapics.Imaging.BitmapDecoder data,
             int x, int y, int width, int height,
             LibEposPrint.Color color,
             LibEposPrint.Mode mode,
             LibEposPrint.Halftone halftone,
             double brightness,
             LibEposPrint.Compress compress);
```

Visual Basic .NET

```
Public Function AddImageAsync
        (data As Windows.Graphics.Imaging.BitmapDecoder,
         x As Integer, y As Integer, width As Integer,
         height As Integer, color As LibEposPrint.Color,
         mode As LibEposPrint.Mode,
         halftone As LibEposPrint.Halftone,
         brightness As Double,
         compress As LibEposPrint.Compress)
         As Windows.Foundation.IAsyncAction
```

Parameter

- data : Specifies an instance of the Windows.Grapics.Imaging.BitmapDecoder 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
Color.COLOR_NONE	Characters are not printed.
Color.COLOR_1	First color
Color.COLOR_2	Second color
Color.COLOR_3	Third color
Color.COLOR_4	Fourth color
Color.DEFAULT	First color

- mode : Specify the color mode.

Set value	Description	TM printer-separate setting												
		TM-m10	TM-m30	TM-P20	TM-P60/ TM-P60II	TM-P80	TM-T20/ TM-T20II	TM-T70	TM-T70II	TM-T81III	TM-T82/ TM-T82II	TM-T88V	TM-T90II	TM-U220/ TM-U330
Mode.MODE_MONO	Monochrome (2 tone)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Mode.MODE_GRAY16	Multiple tone (16 tone)	✔	✔	-	-	-	-	-	✔	-	-	✔	✔	-
Mode.DEFAULT	Monochrome (2 tone)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔

- halftone : Specify the half tone treatment method.

Set value	Description
Halftone.HALFTONE_DITHER	Dither (This is suitable for graphic printing).
Halftone. HALFTONE_ERROR_DIFFUSION	Error diffusion (This is suitable for mixed printing or characters and graphics).
Halftone.HALFTONE_THRESHOLD	Threshold value (This is suitable for printing of characters).
Halftone.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. Set image compression only for a *Bluetooth*® interface.

Set value	Description	TM printer-separate setting													
		TM-m10	TM-m30	TM-P20	TM-P60/ TM-P60II	TM-P80	TM-T20	TM-T20II	TM-T70	TM-T70II	TM-T81II	TM-T82/ TM-T82II	TM-T88V	TM-T90II	TM-U220/ TM-U330
Compress. COMPRESS_ DEFLATE	Image compression is carried out.	✔	✔	✔	-	-	-	✔	-	✔	-	-	✔	-	-
Compress. COMPRESS_NONE	Image compression is not carried out.	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Compress.DEFAULT	Image compression is not carried out.	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	An unspecified error occurred.

Example

- Visual C#

```
using Windows.Graphics.Imaging;

private async void sampleImageFunction()
  try
  {
      BitmapDecoder imageData = null;
      ///Process///
      Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
      await builder.AddImageAsync(imageData, 0, 0, 256, 256, Color.DEFAULT,
                    Mode.MODE_MONO, Halftone.HALFTONE_DITHER, 1.0,
                    Compress.COMPRESS_ DEFLATE);
      ///Process///
  }
  catch (Exception ex)
  {
      if (ex.HResult == EposHResult.HR_E_FAIL)
      {
          ///Process///
      }
      ///Process///
  }
```

- Visual Basic .NET

```
Imports Windows.Graphics.Image

Private Async Sub sampleImageFunction()
  Dim ImageData As BitmapDecoder = Nothing
  `///Process///
  Try
      Dim builder As Builder
                      = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
      Await builder.AddImageAsync(imageData, 0, 0, 256, 256, Color.DEFAULT,
                    Mode.MODE_MONO, Halftone.HALFTONE_DITHER, 1.0,
                    Compress.COMPRESS_ DEFLATE)
      `///Process///
  Catch ex As Exception
      If ex.HResult = EposHResult.HR_E_FAIL Then
         `///Process///
      End If
      `///Process///
  End Try
```

AddImageAsync (Previous format)

Adds raster image printing to the command buffer.
Prints the graphic in the Windows.Grapics.Imaging.BitmapDecoder class.
Out of the Windows.Grapics.Imaging.BitmapDecoder 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.

- This API is an asynchronous API. Wait until the process by this API ends to execute a new process.
- This API does not support cancellation of asynchronous operation by Windows.Foundation.IAsyncInfo.Cancel().
- To print a raster image at high speed, specify Align.ALIGN_LEFT for the AddTextAlign (p.58), 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

Visual C#

```
public Windows.Foundation.IAsyncAction AddImageAsync
            (Windows.Grapics.Imaging.BitmapDecoder data,
             int x, int y, int width, int height,
             LibEposPrint.Color color,
             LibEposPrint.Mode mode,
             LibEposPrint.Halftone halftone,
             double brightness);
```

Visual Basic .NET

```
Public Function AddImageAsync
        (data As Windows.Graphics.Imaging.BitmapDecoder,
         x As Integer, y As Integer, width As Integer,
         height As Integer, color As LibEposPrint.Color,
         mode As LibEposPrint.Mode,
         halftone As LibEposPrint.Halftone,
         brightness As Double)
         As Windows.Foundation.IAsyncAction
```

Parameter

- data : Specifies an instance of the Windows.Grapics.Imaging.BitmapDecoder 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
Color.COLOR_NONE	Characters are not printed.
Color.COLOR_1	First color
Color.COLOR_2	Second color
Color.COLOR_3	Third color
Color.COLOR_4	Fourth color
Color.DEFAULT	First color

- mode : Specify the color mode.

Set value	Description	TM printer-separate setting												
		TM-m10	TM-m30	TM-P20	TM-P60/ TM-P60II	TM-P80	TM-T20/ TM-T20II	TM-T70	TM-T70II	TM-T81II	TM-T82/ TM-T82II	TM-T88V	TM-T90II	TM-U220/ TM-U330
Mode.MODE_MONO	Monochrome (2 tone)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Mode.MODE_GRAY16	Multiple tone (16 tone)	✔	✔	-	-	-	-	-	✔	-	-	✔	✔	-
Mode.DEFAULT	Monochrome (2 tone)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔

- halftone : Specify the half tone treatment method.

Set value	Description
Halftone.HALFTONE_DITHER	Dither (This is suitable for graphic printing).
Halftone.HALFTONE_ERROR_DIFFUSION	Error diffusion (This is suitable for mixed printing or characters and graphics).
Halftone.HALFTONE_THRESHOLD	Threshold value (This is suitable for printing of characters).
Halftone.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

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	An unspecified error occurred.

Example

- Visual C#

```
using Windows.Graphics.Imaging;

private async void sampleImageFunction()
  try
  {
      BitmapDecoder imageData = null;
      ///Process///
      Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
      await builder.AddImageAsync(imageData, 0, 0, 256, 256, Color.DEFAULT,
                        Mode.MODE_MONO, Halftone.HALFTONE_DITHER, 1.0);
      ///Process///
  }
  catch (Exception ex)
  {
      if (ex.HResult == EposHResult.HR_E_FAIL)
      {
          ///Process///
      }
      ///Process///
  }
```

- Visual Basic .NET

```
Imports Windows.Graphics.Image

Private Async Sub sampleImageFunction()
  Dim ImageData As BitmapDecoder = Nothing
  `///Process///
  Try
      Dim builder As Builder
                        = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
      Await builder.AddImageAsync(imageData, 0, 0, 256, 256, Color.DEFAULT,
                        Mode.MODE_MONO, Halftone.HALFTONE_DITHER, 1.0)
      `///Process///
  Catch ex As Exception
      If ex.HResult = EposHResult.HR_E_FAIL Then
         `///Process///
      End If
      `///Process///
  End Try
```

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

- Visual C#

```
try
{
    BitmapDecoder imageData = null;
    ///Process///
    Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    builder.AddPageBegin();
    builder.AddPagePosition(0, 255);
    await builder.AddImageAsync(imageData, 0, 0, 256, 256, Color.DEFAULT,
                Mode.MODE_MONO, Halftone.HALFTONE_DITHER, 1.0);
    builder.AddPageEnd();
    ///Process///
}
catch (Exception ex)
{
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim ImageData As BitmapDecoder = Nothing
    Dim builder As Builder
                    = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
    builder.AddPageBegin()
    builder.AddTextPosition(0, 255)
    Await builder.AddImageAsync(imageData, 0, 0, 256, 256, Color.DEFAULT,
                Mode.MODE_MONO, Halftone.HALFTONE_DITHER, 1.0)
    builder.AddPageEnd()
    `///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       `///Process///
    End If
    `///Process///
End Try
```

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

Visual C#

```
public void AddLogo(int key1, int key2);
```

Visual Basic .NET

```
Public Sub AddLogo(key1 As Integer, key2 As Integer)
```

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

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	An unspecified error occurred.

Example

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

- Visual C#

```
try
{
    Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    builder.AddLogo(48, 48);
    ///Process///
}
catch (Exception ex)
{
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim builder As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
    builder.AddLogo(48, 48)
    `///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       `///Process///
    End If
    `///Process///
End Try
```

AddBarcode

Adds barcode printing to the command buffer.

Syntax

Visual C#

```
public void AddBarcode
     (System.String data, LibEposPrint.Barcode type,
      LibEposPrint.Hri hri, LibEposPrint.Font font,
      int width, int height);
```

Visual Basic .NET

```
Public Sub AddBarcode
    (data As String, type As LibEposPrint.BarcodeType,
     hri As LibEposPrint.Hri, font As LibEposPrint.Font,
     width As Integer, height As Integer)
```

Parameter

- data : Specifies the barcode data as a string.

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

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

Barcode type	Description
CODE93	Start and stop characters are automatically added. A check digit is automatically calculated and added.
CODE128	Specify a start character (CODE A, CODE B, CODE C). A stop character is automatically added. A check digit is automatically calculated and added. To encode each of the following characters, specify two characters starting with the character "{": FNC1: {1 FNC2: {2 FNC3: {3 FNC4: {4 CODE A: {A CODE B: {B CODE C: {C SHIFT: {S {: {{
GS1-128	A start character, 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
BarcodeType.BARCODE_UPC_A	UPC-A
BarcodeType.BARCODE_UPC_E	UPC-E
BarcodeType.BARCODE_EAN13	EAN13
BarcodeType.BARCODE_JAN13	JAN13
BarcodeType.BARCODE_EAN8	EAN8
BarcodeType.BARCODE_JAN8	JAN8
BarcodeType.BARCODE_CODE39	CODE39
BarcodeType.BARCODE_ITF	ITF
BarcodeType.BARCODE_CODABAR	CODABAR
BarcodeType.BARCODE_CODE93	CODE93
BarcodeType.BARCODE_CODE128	CODE128
BarcodeType.BARCODE_GS1_128	GS1-128
BarcodeType.BARCODE_GS1_DATABAR_OMNIDIRECTIONAL	GS1 DataBar Omnidirectional
BarcodeType.BARCODE_GS1_DATABAR_TRUNCATED	GS1 DataBar Truncated
BarcodeType.BARCODE_GS1_DATABAR_LIMITED	GS1 DataBar Limited
BarcodeType.BARCODE_GS1_DATABAR_EXPANDED	GS1 DataBar Expanded

- hri : Specifies the HRI position.

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

- font : Specifies the HRI font.

Set value	Description
Font.FONT_A(default)	Font A
Font.FONT_B	Font B
Font.FONT_C	Font C
Font.FONT_D	Font D
Font.FONT_E	Font E
Font.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

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	An unspecified error occurred.

Example

To print barcodes:

- Visual C#

```
try
{
    Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    builder.AddBarcode("01234567890", BarcodeType.BARCODE_UPC_A,Hri.HRI_BELOW,
             Font.UNSPECIFIED, 2, 64);
    builder.AddBarcode("01234500005", BarcodeType.BARCODE_UPC_E, Hri.UNSPECIFIED,
             Font.UNSPECIFIED, Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.AddBarcode("201234567890", BarcodeType.BARCODE_EAN13, Hri.UNSPECIFIED,
             Font.UNSPECIFIED, Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.AddBarcode("201234567890", BarcodeType.BARCODE_JAN13, Hri.UNSPECIFIED,
             Font.UNSPECIFIED, Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.AddBarcode("2012345", BarcodeType.BARCODE_EAN8, Hri.UNSPECIFIED,
             Font.UNSPECIFIED, Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.AddBarcode("2012345", BarcodeType.BARCODE_JAN8, Hri.UNSPECIFIED,
             Font.UNSPECIFIED, Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.AddBarcode("ABCDE", BarcodeType.BARCODE_CODE39, Hri.UNSPECIFIED,
             Font.UNSPECIFIED, Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.AddBarcode("012345", BarcodeType.BARCODE_ITF, Hri.UNSPECIFIED,
             Font.UNSPECIFIED, Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.AddBarcode("A012345A", BarcodeType.BARCODE_CODABAR, Hri.UNSPECIFIED,
             Font.UNSPECIFIED, Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.AddBarcode("ABCDE", BarcodeType.BARCODE_CODE93, Hri.UNSPECIFIED,
             Font.UNSPECIFIED, Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.AddBarcode("{Babcde", BarcodeType.BARCODE_CODE128, Hri.UNSPECIFIED,
             Font.UNSPECIFIED, Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.AddBarcode("(01)201234567890*", BarocdeType.BARCODE_GS1_128,
             Hri.UNSPECIFIED, Font.UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
             Builder.PARAM_UNSPECIFIED);
    builder.AddBarcode("0201234567890",
             BarcodeType.BARCODE_GS1_DATABAR_OMNIDIRECTIONAL, Hri.UNSPECIFIED,
             Font.UNSPECIFIED, Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.AddBarcode("0201234567890", BarcodeType.BARCODE_GS1_DATABAR_TRUNCATED,
             Hri.UNSPECIFIED, Font.UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
             Builder.PARAM_UNSPECIFIED);
    builder.AddBarcode("0201234567890", BarcodeType.BARCODE_GS1_DATABAR_LIMITED,
             Hri.UNSPECIFIED, Font.UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
             Builder.PARAM_UNSPECIFIED);
    builder.AddBarcode("(01)2012345678903", BarcodeType.BARCODE_GS1_DATABAR_EXPANDED,
             Hri.UNSPECIFIED, Font.UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
             Builder.PARAM_UNSPECIFIED);
    ///Process///
}
catch (Exception ex)
{
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim builder As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
    builder.AddBarcode("01234567890", BarcodeType.BARCODE_UPC_A,Hri.HRI_BELOW,
                       Font.UNSPECIFIED, 2, 64)
    builder.AddBarcode("01234500005", BarcodeType.BARCODE_UPC_E, Hri.UNSPECIFIED,
                       Font.UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
                       Builder.PARAM_UNSPECIFIED)
    builder.AddBarcode("201234567890", BarcodeType.BARCODE_EAN13, Hri.UNSPECIFIED,
                       Font.UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
                       Builder.PARAM_UNSPECIFIED)
    builder.AddBarcode("201234567890", BarcodeType.BARCODE_JAN13, Hri.UNSPECIFIED,
                       Font.UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
                       Builder.PARAM_UNSPECIFIED)
    builder.AddBarcode("2012345", BarcodeType.BARCODE_EAN8, Hri.UNSPECIFIED,
                       Font.UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
                       Builder.PARAM_UNSPECIFIED)
    builder.AddBarcode("2012345", BarcodeType.BARCODE_JAN8, Hri.UNSPECIFIED,
                       Font.UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
                       Builder.PARAM_UNSPECIFIED)
    builder.AddBarcode("ABCDE", BarcodeType.BARCODE_CODE39, Hri.UNSPECIFIED,
                       Font.UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
                       Builder.PARAM_UNSPECIFIED)
    builder.AddBarcode("012345", BarcodeType.BARCODE_ITF, Hri.UNSPECIFIED,
                       Font.UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
                       Builder.PARAM_UNSPECIFIED)
    builder.AddBarcode("A012345A", BarcodeType.BARCODE_CODABAR, Hri.UNSPECIFIED,
                       Font.UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
                       Builder.PARAM_UNSPECIFIED)
    builder.AddBarcode("ABCDE", BarcodeType.BARCODE_CODE93, Hri.UNSPECIFIED,
                       Font.UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
                       Builder.PARAM_UNSPECIFIED)
    builder.AddBarcode("{Babcde", BarcodeType.BARCODE_CODE128, Hri.UNSPECIFIED,
                       Font.UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
                       Builder.PARAM_UNSPECIFIED)
   builder.AddBarcode("(01)201234567890*", BarocdeType.BARCODE_GS1_128, Hri.UNSPECIFIED,
                       Font.UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
                       Builder.PARAM_UNSPECIFIED)
    builder.AddBarcode("0201234567890", BarcodeType.BARCODE_GS1_DATABAR_OMNIDIRECTIONAL,
                       Hri.UNSPECIFIED, Font.UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
                       Builder.PARAM_UNSPECIFIED)
    builder.AddBarcode("0201234567890", BarcodeType.BARCODE_GS1_DATABAR_TRUNCATED,
                       Hri.UNSPECIFIED, Font.UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
                       Builder.PARAM_UNSPECIFIED)
    builder.AddBarcode("0201234567890", BarcodeType.BARCODE_GS1_DATABAR_LIMITED,
                       Hri.UNSPECIFIED, Font.UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
                       Builder.PARAM_UNSPECIFIED)
    builder.AddBarcode("(01)2012345678903", BarcodeType.BARCODE_GS1_DATABAR_EXPANDED,
                       Hri.UNSPECIFIED, Font.UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
                       Builder.PARAM_UNSPECIFIED)
    `///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       `///Process///
    End If
    `///Process///
End Try
```

AddSymbol

Adds 2D-Code printing to the command buffer.

Syntax

Visual C#

```
public void AddSymbol
     (System.String data, LibEposPrint.Symbol type,
      LibEposPrint.Level level1, int level2, int width,
      int height, int size);
```

Visual Basic .NET

```
Public Sub AddSymbol
    (data As String, type As LibEposPrint.SymbolType,
     level1 As LibEposPrint.Level, level2 As Integer,
     width As Integer, height As Integer,
     size As Integer)
```

Parameter

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

Specify a string that follows the 2D-code standard specified by the type parameter. If the specified string does not conform to the standard, a 2D-code will not be printed.

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.
DataMatrix square DataMatrix rectangle, 8 lines DataMatrix rectangle, 12 lines DataMatrix rectangle, 16 lines	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.

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

- level1 : Specifies the error correction level.

Set value	Description
Level.LEVEL_0	PDF417 error correction level 0
Level.LEVEL_1	PDF417 error correction level 1
Level.LEVEL_2	PDF417 error correction level 2
Level.LEVEL_3	PDF417 error correction level 3
Level.LEVEL_4	PDF417 error correction level 4
Level.LEVEL_5	PDF417 error correction level 5
Level.LEVEL_6	PDF417 error correction level 6
Level.LEVEL_7	PDF417 error correction level 7
Level.LEVEL_8	PDF417 error correction level 8
Level.LEVEL_L	QR Code error correction level L
Level.LEVEL_M	QR Code error correction level M
Level.LEVEL_Q	QR Code error correction level Q
Level.LEVEL_H	QR Code error correction level H
Level.LEVEL_DEFAULT	Default level
Level.UNSPECIFIED	Retains the current setting.

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

- level2 : Specifies the error correction level.

Set value	Description
5 to 95 integer	Aztec Code error correction level (percent unit)
Builder.PARAM_UNSPECIFIED	Default level
Builder.PARAM_UNSPECIFIED	Retains the current setting.

- Select the level according to the 2D-Code type.
- For PDF417/ QR Code/ MaxiCode/ two-dimensional GS1 DataBar/ DataMatrix, select Builder.PARAM_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

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	An unspecified error occurred.

Example

To print 2D-Code:

- Visual C#

```
try
{
    Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    builder.AddSymbol("ABCDE", SymbolType.SYMBOL_PDF417_STANDARD,
            Level.UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED);
    builder.AddSymbol("ABCDE", SymbolType.SYMBOL_QRCODE_MODEL_2, Level.LEVEL_Q,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.AddSymbol("908063840\\x1d850\\x1d001\\x1d\\x04",
            SymbolType.SYMBOL_MAXICODE_MODE_2, Level.UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.AddSymbol("0201234567890", SymbolType.SYMBOL_GS1_DATABAR_STACKED,
            Level.UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED);
    builder.AddSymbol("0201234567890",
            SymbolType.SYMBOL_GS1_DATABAR_STACKED_OMNIDIRECTIONAL,
            Level.UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED);
    builder.AddSymbol("(01)02012345678903",
            SymbolType.SYMBOL_GS1_DATABAR_EXPANDED_STACKED, Level.UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.AddSymbol("ABCDE", SymbolType.SYMBOL_AZTECCODE_FULLRANGE,
            Level.UNSPECIFIED, 23, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED);
    builder.AddSymbol("ABCDE", SymbolType.SYMBOL_DATAMATRIX_SQUARE,
            Level.UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
            Builder.PARAM_UNSPECIFIED);
    ///Process///
}
catch (Exception ex)
{
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim builder As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
    builder.AddSymbol("ABCDE", SymbolType.SYMBOL_PDF417_STANDARD,
                      Level.UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
                      Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
                      Builder.PARAM_UNSPECIFIED)
    builder.AddSymbol("ABCDE", SymbolType.SYMBOL_QRCODE_MODEL_2, Level.LEVEL_Q,
                      Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
                      Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED)
    builder.AddSymbol("908063840\\x1d850\\x1d001\\x1d\\x04",
                      SymbolType.SYMBOL_MAXICODE_MODE_2, Level.UNSPECIFIED,
                      Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
                      Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED)
    builder.AddSymbol("0201234567890", SymbolType.SYMBOL_GS1_DATABAR_STACKED,
                      Level.UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
                      Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
                      Builder.PARAM_UNSPECIFIED)
    builder.AddSymbol("0201234567890",
                      SymbolType.SYMBOL_GS1_DATABAR_STACKED_OMNIDIRECTIONAL,
                      Level.UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
                      Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
                      Builder.PARAM_UNSPECIFIED)
    builder.AddSymbol("(01)02012345678903",
                      SymbolType.SYMBOL_GS1_DATABAR_EXPANDED_STACKED,
                      Level.UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
                      Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
                      Builder.PARAM_UNSPECIFIED)
    builder.AddSymbol("ABCDE", SymbolType.SYMBOL_AZTECCODE_FULLRANGE,
                      Level.UNSPECIFIED, 23, Builder.PARAM_UNSPECIFIED,
                      Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED)
    builder.AddSymbol("ABCDE", SymbolType.SYMBOL_DATAMATRIX_SQUARE,
                      Level.UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
                      Builder.PARAM_UNSPECIFIED, Builder.PARAM_UNSPECIFIED,
                      Builder.PARAM_UNSPECIFIED)
    `///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       `///Process///
    End If
    `///Process///
End Try
```

AddPageBegin

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

Use this API function with AddPageEnd (p.107).

Syntax

Visual C#

```
public void AddPageBegin();
```

Visual Basic .NET

```
Public Sub AddPageBegin()
```

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	An unspecified error occurred.

Example

To print the characters "ABCDE" in page mode:

- Visual C#

```
try
{
    Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    builder.AddPageBegin();
    builder.AddText("ABCDE");
    builder.AddPageEnd();
    ///Process///
}
catch (Exception ex)
{
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim builder As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
    builder.AddPageBegin()
    builder.AddText("ABCDE")
    builder.AddPageEnd()
    `///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       `///Process///
    End If
    `///Process///
End Try
```

AddPageEnd

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

Use this API function with AddPageBegin (p.105).

Syntax

Visual C#

```
public void AddPageEnd();
```

Visual Basic .NET

```
Public Sub AddPageEnd()
```

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	An unspecified error occurred.

Example

To print the characters "ABCDE" in page mode:

- Visual C#

```
try
{
    Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    builder.AddPageBegin();
    builder.AddText("ABCDE");
    builder.AddPageEnd();
    ///Process///
}
catch (Exception ex)
{
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim builder As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
    builder.AddPageBegin()
    builder.AddText("ABCDE")
    builder.AddPageEnd()
    `///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       `///Process///
    End If
    `///Process///
End Try
```

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.105) and AddPageEnd (p.107).

Syntax

Visual C#

```
public void AddPageArea(int x, int y,
                        int width, int height);
```

Visual Basic .NET

```
Public Sub AddPageArea
                (x As Integer, y As Integer,
                 width As Integer, height As Integer)
```

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

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	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":

- Visual C#

```
try
{
    Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    builder.AddPageBegin();
    builder.AddPageArea(100, 50, 200, 30);
    builder.AddText("ABCDE");
    builder.AddPageEnd();
    ///Process///
}
catch (Exception ex)
{
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim builder As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
    builder.AddPageBegin()
    builder.AddPageArea(100, 50, 200, 30)
    builder.AddText("ABCDE")
    builder.AddPageEnd()
    `///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       `///Process///
    End If
    `///Process///
End Try
```

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.105) and AddPageEnd (p.107).

Syntax

Visual C#

```
public void AddPageDirection
                        (LibEposPrint.Direction dir);
```

Visual Basic .NET

```
Public Sub AddPageDirection
                        (dir As LibEposPrint.Direction)
```

Parameter

- dir : Specifies the print direction in page mode.

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

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	An unspecified error occurred.

Example

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

- Visual C#

```
try
{
    Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    builder.AddPageBegin();
    builder.AddPageArea(100, 50, 200, 30);
    builder.AddPageDirection(Direction.DIRECTION_TOP_TO_BOTTOM);
    builder.AddText("ABCDE");
    builder.AddPageEnd();
    ///Process///
}
catch (Exception ex)
{
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim builder As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
    builder.AddPageBegin()
    builder.AddPageArea(100, 50, 200, 30)
    builder.AddPageDirection(Direction.DIRECTION_TOP_TO_BOTTOM)
    builder.AddText("ABCDE")
    builder.AddPageEnd()
    `///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
        `///Process///
    End If
    `///Process///
End Try
```

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.105) and AddPageEnd (p.107).

Syntax

Visual C#

```
public void AddPagePosition(int x, int y);
```

Visual Basic .NET

```
Public Sub AddPagePosition(x As Integer, y As Integer)
```

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

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	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":

- Visual C#

```
try
{
    Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    builder.AddPageBegin();
    builder.AddPageArea(100, 50, 200, 100);
    builder.AddPagePosition(50, 30);
    builder.AddText("ABCDE");
    builder.AddPageEnd();
    ///Process///
}
catch (Exception ex)
{
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim builder As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
    builder.AddPageBegin()
    builder.AddPageArea(100, 50, 200, 30)
    builder.AddPagePosition(50, 30)
    builder.AddText("ABCDE")
    builder.AddPageEnd()
    `///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
        `///Process///
    End If
    `///Process///
End Try
```

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.105) and AddPageEnd (p.107).

Syntax

Visual C#

```
public void AddPageLine(int x1, int y1, int x2, int y2,
                        LibEposPrint.Line style);
```

Visual Basic .NET

```
Public Sub AddPageLine(x1 As Integer, y1 As Integer,
                       x2 As Integer, y2 As Integer,
                       style As LibEposPrint.Line)
```

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
Line.LINE_THIN	Solid line: Thin
Line.LINE_MEDIUM	Solid line: Medium
Line.LINE_THICK	Solid line: Thick
Line.LINE_THIN_DOUBLE	Double line: Thin
Line.LINE_MEDIUM_DOUBLE	Double line: Medium
Line.LINE_THICK_DOUBLE	Double line: Thick
Line.DEFAULT	Solid line: Thin

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	An unspecified error occurred.

Example

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

- Visual C#

```
try
{
    Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    builder.AddPageBegin();
    builder.AddPageLine(100, 0, 500, 0, Line.LINE_THIN);
    builder.AddPageEnd();
    ///Process///
}
catch (Exception ex)
{
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim builder As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
    builder.AddPageBegin()
    builder.AddPageLine(100, 0, 500, 0, Line.LINE_THIN)
    builder.AddPageEnd()
    `///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       `///Process///
    End If
    `///Process///
End Try
```

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.105) and AddPageEnd (p.107).

Syntax

Visual C#

```
public void AddPageRectangle
                    (int x1, int y1, int x2,
                     int y2, LibEposPrint.Line style);
```

Visual Basic .NET

```
Public Sub AddPageRectangle
          (x1 As Integer, y1 As Integer, x2 As Integer,
           y2 As Integer, style As LibEposPrint.Line)
```

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
Line.LINE_THIN	Solid line: Thin
Line.LINE_MEDIUM	Solid line: Medium
Line.LINE_THICK	Solid line: Thick
Line.LINE_THIN_DOUBLE	Double line: Thin
Line.LINE_MEDIUM_DOUBLE	Double line: Medium
Line.LINE_THICK_DOUBLE	Double line: Thick
Line.DEFAULT	Solid line: Thin

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	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:

- Visual C#

```
try
{
    Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    builder.AddPageBegin();
    builder.AddPageRectangle(100, 0, 500, 200, Line.LINE_THIN);
    builder.AddPageEnd();
    ///Process///
}
catch (Exception ex)
{
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim builder As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
    builder.AddPageBegin()
    builder.AddPageRectangle(100, 0, 500, 200, Line.LINE_THIN)
    builder.AddPageEnd()
    `///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       `///Process///
    End If
    `///Process///
End Try
```

AddCut

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

Not available in page mode.

Syntax

Visual C#

```
public void AddCut(LibEposPrint.Cut type);
```

Visual Basic .NET

```
Public Sub AddCut(type As LibEposPrint.Cut)
```

Parameter

- type : Specifies the paper cut type.

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

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	An unspecified error occurred.

Example

To perform feed cut operation:

- Visual C#

```
try
{
    Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    ///Process///
    builder.AddCut(Cut.CUT_FEED);
    ///Process///
}
catch
(Exception ex) {
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim builder As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
    `///Process///
    builder.AddCut(Cut.CUT_FEED)
    `///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       `///Process///
    End If
    `///Process///
End Try
```

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

Visual C#

```
public void AddPulse(LibEposPrint.Drawer drawer,
                     LibEposPrint.Pulse time);
```

Visual Basic .NET

```
Public Sub AddPulse(drawer As LibEposPrint.Drawer,
                    time As LibEposPrint.Pulse)
```

Parameter

- drawer : Specifies the drawer kick connector.

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

- time : Specifies the drawer kick signal energizing time.

Set value	Description
Pulse.PULSE_100	100 ms
Pulse.PULSE_200	200 ms
Pulse.PULSE_300	300 ms
Pulse.PULSE_400	400 ms
Pulse.PULSE_500	500 ms
Pulse.DEFAULT	100 ms

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	An unspecified error occurred.

Example

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

- Visual C#

```
try
{
    Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
     ///Process///
    builder.AddPulse(Drawer.DRAWER_1, Pulse.PULSE_100);
     ///Process///
}
catch (Exception ex)
{
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
         ///Process///
    }
     ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim builder As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
    ` ///Process///
    builder.AddPulse(Drawer.DRAWER_1, Pulse.PULSE_100)
    ` ///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       ` ///Process///
    End If
    ` ///Process///
End Try
```

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

Visual C#

```
public void AddSound(LibEposPrint.Pattern pattern,
                     int repeat, int cycle);
```

Visual Basic .NET

```
Public Sub AddSound(pattern As LibEposPrint.Pattern,
                    repeat As Integer, cycle As Integer)
```

Parameter

- pattern : Specifies the buzzer pattern.

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

- 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

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	An unspecified error occurred.

Example

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

- Visual C#

```
try
{
    Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    ///Process///
    builder.AddSound(Pattern.PATTERN_1, 3, 1000);
    ///Process///
}
catch (Exception ex)
{
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim builder As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
    `///Process///
    builder.AddSound(Pattern.PATTERN_1, 3, 1000)
    `///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       `///Process///
    End If
    `///Process///
End Try
```

AddFeedPosition

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

Syntax

Visual C#

```
public void AddFeedPosition
                        (LibEposPrint.Feed position);
```

Visual Basic .NET

```
Public Sub AddFeedPosition
                        (position As LibEposPrint.Feed)
```

Parameter

- position : Specifies the feed position.

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

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	An unspecified error occurred.

Example

To feed a label paper to the peeling position:

- Visual C#

```
try
{
    Builder builder = new Builder("TM-P60II", ModelLang.MODEL_LANG_ANK);
    ///Process///
    builder.AddFeedPosition(Feed.FEED_PEELING);
    ///Process///
}
catch (Exception ex)
{
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim builder As Builder = New Builder("TM-P60II", ModelLang.MODEL_LANG_ANK)
    `///Process///
    builder.AddFeedPosition(Feed.FEED_PEELING)
    `///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       `///Process///
    End If
    `///Process///
End Try
```

AddLayout

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

Syntax

Visual C#

```
public void AddLayout(LibEposPrint.Layout type,
                      int width, int height,
                      int marginTop, int marginBottom,
                      int offsetCut, int offsetLabel);
```

Visual Basic .NET

```
Public Sub AddLayout
                (type As LibEposPrint.Layout,
                 width As Integer, height As Integer,
                 marginTop As Integer,
                 marginBottom As Integer,
                 offsetCut As Integer,
                 offsetLabel As Integer)
```

Parameter

- type : Specifies the paper type.

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

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	An unspecified error occurred.

Example

To set 60 mm label paper (black mark):

- Visual C#

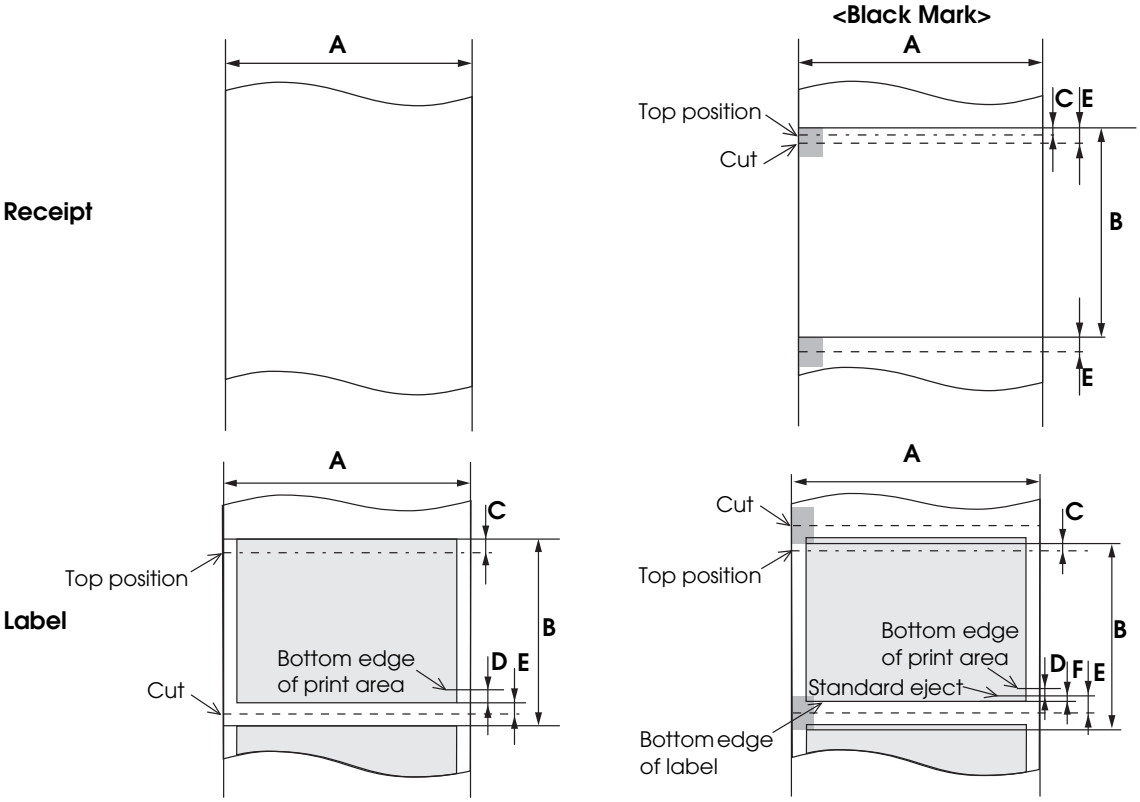
```
try
{
    Builder builder = new Builder("TM-P60II", ModelLang.MODEL_LANG_ANK);
    builder.AddLayout(Layout.LAYOUT_LABEL_BM, 600, 0, 15, -15, 15, 0);
    ///Process///
}
catch (Exception ex)
{
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim builder As Builder = New Builder("TM-P60II", ModelLang.MODEL_LANG_ANK)
    builder.AddLayout(Layout.LAYOUT_LABEL_BM, 600, 0, 15, -15, 15, 0)
    `///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       `///Process///
    End If
    `///Process///
End Try
```

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

Visual C#

```
public void AddCommand(byte[] data);
```

Visual Basic .NET

```
Public Sub AddCommand(data() As Byte)
```

Parameter

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

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	An unspecified error occurred.

Example

- Visual C#

```
try
{
    Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    byte[] data = null;
    ///Process///
    builder.AddCommand(data);
    ///Process///
}
catch (Exception ex)
{
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim builder As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
    Dim data As Byte() = Nothing
    `///Process///
    builder.AddCommand(data)
    `///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       `///Process///
    End If
    `///Process///
End Try
```

Print class (Constructor)

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

Syntax

Visual C#

```
public Print()
```

Visual Basic .NET

```
Public Sub Print()
```

Example

- Visual C#

```
try
{
    Print printer = new Print();
    ///Process///
}
catch (Exception ex)
{
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim printer As Print = New Print()
    `///Process///
Catch ex As Exception
    `///Process///
End Try
```

OpenPrinterAsync

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

if communication with the printer is not required anymore, be sure to call ClosePrinterAsync (p.139), ClosePrinterAsync API, to end communication with the printer.

- This API is an asynchronous API. Wait until the process by this API ends to execute a new process.
- This API does not support cancellation of asynchronous operation by Windows.Foundation.IAsyncInfo.Cancel().
- When a printer is connected, a connection confirmation screen may appear. Execute this API from a UI thread.
- Printer status is notified to the events registered in the print class. For details, see Automatic Acquisition of Printer Status (p.42).
- If you want to stop monitoring of printer status, call ClosePrinterAsync (p.139).
- For connection via *Bluetooth*®, execute this API when devices are paired.
- If you use the printer from multiple mobile terminals, see the Cautions (p.237).

Syntax

Visual C#

```
public Windows.Foundation.IAsyncAction OpenPrinterAsync
          (LibEposPrint.DevType deviceType,
           System.String deviceName,
           LibEposPrint.Monitoring enabled,
           int interval, int timeout);
```

Visual Basic .NET

```
Public Function OpenPrinterAsync
         (deviceType As LibEposPrint.DevType,
          deviceName As String,
          enabled As LibEposPrint.Monitoring,
          interval As Integer, timeout As Integer)
          As Windows.Foundation.IAsyncAction
```

Parameter

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

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

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

deviceType	Specified Value
DevType.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.DEVTYPE_*BLUETOOTH*	BD address (Example: "01:23:45:67:89:AB")

- When a printer's IP address is set as DHCP, specify a MAC address or printer host name for deviceName.
- When DevType.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
Monitoring. MONITORING_TRUE	Enabled
Monitoring. MONITORING_FALSE	Disabled
Monitoring.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 DevType.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

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_ACCESSDENIED (0x80070005)	Open processing failed.
HR_E_ABORT (0x80004004)	The device specified was already being used, and communication with the printer could not be established within the timeout time.
HR_E_PENDING (0x8000000A)	Could not execute process.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	• An attempt was made to start communicating with the device with which communication had already started. • 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:

- Visual C#

```
try
{
    Print printer = new Print();
    await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                  Monitoring.MONITORING_TRUE, Print.PARAM_DEFAULT,
                  Print.PARAM_DEFAULT);
    ///Process///
}
catch (Exception ex)
{
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim printer As Print = New Print()
    Await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                  Monitoring.MONITORING_TRUE, Print.PARAM_DEFAULT,
                  Print.PARAM_DEFAULT)
    `///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       `///Process///
    End If
    `///Process///
End Try
```

OpenPrinterAsync(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 ClosePrinterAsync (p.139), ClosePrinterAsync 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 OpenPrinterAsync (p.133). • This API is an asynchronous API. Wait until the process by this API ends to execute a new process. • This API does not support cancellation of asynchronous operation by Windows.Foundation.IAsyncInfo.Cancel(). • When a printer is connected, a connection confirmation screen may appear. Execute this API from a UI thread. • Printer status is notified to the events registered in the print class. For details, see Automatic Acquisition of Printer Status (p.42). • If you want to stop monitoring of printer status, call ClosePrinterAsync (p.139). • For connection via *Bluetooth*®, execute this API when devices are paired.] • If you use the printer from multiple mobile terminals, see the Cautions (p.237).

Syntax

Visual C#

```
public Windows.Foundation.IAsyncAction OpenPrinterAsync
          (LibEposPrint.DevType deviceType,
           System.String deviceName,
           LibEposPrint.Monitoring enabled,
           int interval);
```

Visual Basic .NET

```
Public Function OpenPrinterAsync
          (deviceType As LibEposPrint.DevType,
           deviceName As String,
           enabled As LibEposPrint.Monitoring,
           interval As Integer)
           As Windows.Foundation.IAsyncAction
```

Parameter

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

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

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

deviceType	Specified Value
DevType.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.DEVTYPE_*BLUETOOTH*	BD address (Example: "01:23:45:67:89:AB")

- When a printer's IP address is set as DHCP, specify a MAC address or printer host name for deviceName.
- When DevType.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
Monitoring. MONITORING_TRUE	Enabled
Monitoring. MONITORING_FALSE	Disabled
Monitoring.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

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_ACCESSDENIED (0x80070005)	Open processing failed.
HR_E_PENDING (0x8000000A)	Could not execute process.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	• An attempt was made to start communicating with the device with which communication had already started. • 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:

- Visual C#

```
try
{
    Print printer = new Print();
    await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                  Monitoring.MONITORING_TRUE, Print.PARAM_DEFAULT);
    ///Process///
}
catch (Exception ex)
{
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim printer As Print = New Print()
    Await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                  Monitoring.MONITORING_TRUE, Print.PARAM_DEFAULT)
    `///Process///
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       `///Process///
    End If
    `///Process///
End Try
```

ClosePrinterAsync

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

- This API is an asynchronous API. Wait until the process by this API ends to execute a new process.
- This API does not support cancellation of asynchronous operation by Windows.Foundation.IAsyncInfo.Cancel().

Syntax

Visual C#

```
public Windows.Foundation.IAsyncAction
                                    ClosePrinterAsync();
```

Visual Basic .NET

```
Public Function ClosePrinterAsync()
                    As Windows.Foundation.IAsyncAction
```

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_PENDING (0x8000000A)	Could not execute process.
HR_E_FAIL (0x80004005)	• This API was called when communication had not started yet. • An unspecified error occurred.

Example

- Visual C#

```
try
{
    Print printer = new Print();
    await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                Monitoring.MONITORING_TRUE, Print.PARAM_DEFAULT);
     ///Process///
    await printer.ClosePrinterAsync();
}
catch (Exception ex)
{
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
         ///Process///
    }
     ///Process///
}
```

- Visual Basic .NET

```
Try
    Dim printer As Print = New Print()
    Await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                  Monitoring.MONITORING_TRUE, Print.PARAM_DEFAULT)
    ` ///Process///
    Await printer.ClosePrinterAsync()
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       ` ///Process///
    End If
    ` ///Process///
End Try
```

SendDataAsync

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.
- This API is an asynchronous API. Wait until the process by this API ends to execute a new process.
- This API does not support cancellation of asynchronous operation by Windows.Foundation.IAsyncInfo.Cancel().
- If you use the printer from multiple mobile terminals, see the Cautions (p.237).

Syntax

Visual C#

```
public Windows.Foundation.IAsyncOperation
        <PrinterStatus> SendDataAsync
        (LibEposPrint.Builder builder, int timeout);
```

Visual Basic .NET

```
Public Function SendDataAsync
  (builder As LibEposPrint.Builder, timeout As Integer)
   As Windows.Foundation.IAsyncOperation
  (Of LibEposPrint.PrinterStatus)
```

Parameter

- builder : Specifies a Builder class instance. For details on the Builder class, refer to Builder class (p.51).
- 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).

The timeout time specified for timeout is a period of time between print document transmission and printer status acquisition.

Return value

The printer status and the battery status return.

- The printer status returns to PrinterStatus.printerStatus.
  For details on the sprinter status, see Printer Statuses and Actions to Take (p.47).
- The battery status returns to PrinterStatus.batteryStatus.
  For details on the battery status, see Battery Status (p.49).

The printer status and battery status to be returned vary depending on the printer status monitoring setting for OpenPrinterAsync (p.133).

Printer status monitoring setting	Description
MONITORING_TRUE	The printer status and battery status monitored last return.
MONITORING_FALSE	The printer status when the print document has been sent returns. If the printer status fails to be acquired, ST_NO_RESPONSE returns as the printer status and 0x0000 returns as the battery status.

When an exception occurs, the printer status and the battery status are acquired using GetPrinterStatus (p.178) during exception processing.

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_PENDING (0x8000000A)	Could not execute process.
HR_E_ABORT (0x80004004)	Could not send all the data within the specified time.
HR_E_ACCESSDENIED (0x80070005)	• Connection error occurred. • The printer was offline.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	• This API was called when communication had not started yet. • An unspecified error occurred.

If HR_E_ACCESSDENIED occurs, whether the printer is offline is identified by checking the printer status. When "ST_NO_RESPONSE" is not set and is set "ST_OFF_LINE" as the printer status, the printer is offline.

Example

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

- Visual C#

```
Print printer = new Print();
PrinterStatus status;
status.printerStatus = 0;
status.batteryStatus = 0;

try
{
    Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    builder.AddText("ABCDE\n");

    await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                Monitoring.MONITORING_TRUE, Print.PARAM_DEFAULT);

    status = await printer.SendDataAsync(builder, 10000);

    ///Process///

    await printer.ClosePrinterAsync();

}
catch (Exception ex)
{
    PrinterStatus errStatus = printer.GetPrinterStatus(ex.HResult);
    ///Process///
}
```

- Visual Basic .NET

```
Dim printer As Print = New Print()
Dim status As PrinterStatus
status.printerStatus = 0
status.batteryStatus = 0

Try

    Dim builder As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
    builder.AddText("ABCDE" + vbCrLF)

    status = Awiat printer.OpenPrinterAsync(DevType.DEVTYPE_TCP,
                            "192.168.192.168", Monitoring.MONITORING_TRUE,
                            Print.PARAM_DEFAULT)
    status = Await printer.SendDataAsync(builder, 10000)

    `///Process///

    Await printer.ClosePrinterAsync()
Catch ex As Exception
    Dim errStatus As PrinterStatus = printer.GetPrinterStatus(ex.HResult)
    `///Process///
End Try
```

BeginTransactionAsync

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 EndTransactionAsync (p.146)is handled as one set of print processing operations.

- For details about specifying a transaction, see the To specify a transaction (p.239).
- This API is an asynchronous API. Wait until the process by this API ends to execute a new process.
- This API does not support cancellation of asynchronous operation by Windows.Foundation.IAsyncInfo.Cancel().

Syntax

Visual C#

```
public Windows.Foundation.IAsyncAction
        BeginTransactionAsync();
```

Visual Basic .NET

```
Public Function BeginTransactionAsync() As
  Windows.Foundation.IAsyncAction
```

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_PENDING (0x8000000A)	This API was called when communication had not started yet. Transaction has already started using this function.
HR_E_FAIL (0x80004005)	An unspecified error occurred.

Example

Multiple or one print processing is transacted:

- Visual C#

```
Print printer = new Print();
PrinterStatus status;
status.printerStatus = 0;
status.batteryStatus = 0;

try {
  Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
  Builder builder2 = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);

  builder.AddText("ABCDE\n");

  builder2.AddText("12345\n");
  builder2.AddCut(CUT_FEED);

  await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                                 Monitoring.MONITORING_TRUE, Print.PARAM_DEFAULT);

  await printer.BeginTransactionAsync();

  status = await printer.SendDataAsync(builder, 10000);
  status = await printer.SendDataAsync(builder2, 10000);

  await printer.EndTransactionAsync();

  ///Process///
  await printer.ClosePrinterAsync();
} catch(Exception ex) {
  PrinterStatus errStatus = printer.GetPrinterStatus(ex.HResult);
  ///Process///
}
```

- Visual Basic .NET

```
Dim printer As Print = New Print()
Dim status As PrinterStatus
status.printerStatus = 0
status.batteryStatus = 0
Try
  Dim builder As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
  Dim builder2 As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)

  builder.AddText("ABCDE" + vbCrLf)

  builder2.AddText("12345" + vbCrLf)
  builder2.AddCut(CUT_FEED)

  Await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                                 Monitoring.MONITORING_TRUE, Print.PARAM_DEFAULT)
  Await printer.BeginTransactionAsync()
  status = Await printer.SendDataAsync(builder, 10000)
  status = Await printer.SendDataAsync(builder2, 10000)
  Await printer.EndTransactionAsync()

  '///Process///
  Await printer.ClosePrinterAsync()
Catch ex As Exception
  Dim errStatus As PrinterStatus = printer.GetPrinterStatus(ex.HResult)
  '///Process///
End Try
```

EndTransactionAsync

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 BeginTransactionAsync (p.144) 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.239).
- This API is an asynchronous API. Wait until the process by this API ends to execute a new process.
- This API does not support cancellation of asynchronous operation by Windows.Foundation.IAsyncInfo.Cancel().

Syntax

Visual C#

```
public Windows.Foundation.IAsyncAction
        EndTransactionAsync();
```

Visual Basic .NET

```
Public Function EndTransactionAsync() As
  Windows.Foundation.IAsyncAction
```

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_PENDING (0x8000000A)	This API was called when communication had not started yet. This API was called when transaction had not started.
HR_E_FAIL (0x80004005)	An unspecified error occurred.

Example

Multiple or one print processing is transacted:

- Visual C#

```
Print printer = new Print();
PrinterStatus status;
status.printerStatus = 0;
status.batteryStatus = 0;

try {
  Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
  Builder builder2 = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);

  builder.AddText("ABCDE\n");

  builder2.AddText("12345\n");
  builder2.AddCut(CUT_FEED);

  await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                                 Monitoring.MONITORING_TRUE, Print.PARAM_DEFAULT);

  await printer.BeginTransactionAsync();

  status = await printer.SendDataAsync(builder, 10000);
  status = await printer.SendDataAsync(builder2, 10000);

  await printer.EndTransactionAsync();

  ///Process///
  await printer.ClosePrinterAsync();
} catch(Exception ex) {
  PrinterStatus errStatus = printer.GetPrinterStatus(ex.HResult);
  ///Process///
}
```

- Visual Basic .NET

```
Dim printer As Print = New Print()
Dim status As PrinterStatus
status.printerStatus = 0
status.batteryStatus = 0
Try
  Dim builder As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
  Dim builder2 As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)

  builder.AddText("ABCDE" + vbCrLf)

  builder2.AddText("12345" + vbCrLf)
  builder2.AddCut(CUT_FEED)

  Await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                                 Monitoring.MONITORING_TRUE, Print.PARAM_DEFAULT)
  Await printer.BeginTransactionAsync()
  status = Await printer.SendDataAsync(builder, 10000)
  status = Await printer.SendDataAsync(builder2, 10000)
  Await printer.EndTransactionAsync()

  '///Process///
  Await printer.ClosePrinterAsync()
Catch ex As Exception
  Dim errStatus As PrinterStatus = printer.GetPrinterStatus(ex.HResult)
  '///Process///
End Try
```

SetStatusChangeEventCallback

This registers the notification destination of printer status.

- This API can be executed following execution of OpenPrinterAsync (p.133).
- When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten.

Syntax

Visual C#

```
public void SetStatusChangeEventCallback
            (LibEposPrint.StatusChangeEvent target);
```

Visual Basic .NET

```
Public Sub SetStatusChangeEventCallback
            (target As LibEposPrint.StatusChangeEvent)
```

Parameter

- target : This specifies the object that has the notification destination method (definition of call-back method).
  To cancel the registration of the notification destination, specify as follows:

Development Language	Code
Visual C#	null
Visual Basic .NET	Nothing

Definition of Callback Method

Visual C#

```
void Method name(String deviceName, int status)
```

Visual Basic .NET

```
Sub Method name(deviceName As String, status As Integer)
```

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

- Visual C#

```
private void Status_Change_Event(string deviceName, int status)
{
    ///Process///
}

private async void openPrinter()
{
    Print printer = new Print();
    StatusChangeEvent statusChangeEvent
                                    = new StatusChangeEvent(Status_Change_Event);
    printer.SetStatusChangeEventCallback(statusChangeEvent);

    try
    {
        await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                                       Monitoring.MONITORING_TRUE,
                                       Print.PARAM_DEFAULT);
        ///Process///
    }
    catch (Exception ex)
    {
        if (ex.HResult == EposHResult.HR_E_FAIL)
        {
            ///Process///
        }
    }
}
```

- Visual Basic .NET

```
Private Sub Status_Change_Event(deviceName As String, status As Integer)
  '///Process///
End Sub

Private Async Sub openPrinter()
    Dim printer As Print = New Print()

    Dim statusChangeEvent As StatusChangeEvent
                             = New StatusChangeEvent(AddressOf statusChangeEvent)
    printer.SetStatusChangeEventCallback(statusChangeEvent)

    Try
        Await printer.OpenPrintAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                         Monitoring.MONITORING_TRUE, Print.PARAM_DEFAULT)
        '///Process///
    Catch ex As Exception
        if ex.HResult = EposHResult.HR_E_FAIL Then
            '///Process///
        End If
      '///Process///
    End Try
End Sub
```

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 OpenPrinterAsync (p.133).
- When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten.

Syntax

Visual C#

```
public void SetOnlineEventCallback
            (LibEposPrint.OnlineEvent target);
```

Visual Basic .NET

```
Public Sub SetOnlineEventCallback
            (target As LibEposPrint.OnlineEvent)
```

Parameter

- target : This specifies the object that has the notification destination method (definition of call-back method).
  To cancel the registration of the notification destination, specify as follows:

Development Language	Code
Visual C#	null
Visual Basic .NET	Nothing

Definition of Callback Method

Visual C#

```
void Method name(String deviceName)
```

Visual Basic .NET

```
Sub Method name(deviceName As String)
```

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.

Example

- Visual C#

```
private void Online_Event(string deviceName)
{
    ///Process///
}

private async void openPrinter()
{
    Print printer = new Print();

    OnlineEvent onlineEvent = new OnlineEvent(Online_Event);
    printer.SetOnlineEventCallback(onlineEvent);

    try
    {
        await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                                       Monitoring.MONITORING_TRUE,
                                       Print.PARAM_DEFAULT);
        ///Process///
    }
    catch (Exception ex)
    {
        if (ex.HResult == EposHResult.HR_E_FAIL)
        {
            ///Process///
        }
    }
}
```

- Visual Basic .NET

```
Private Sub Online_Event(deviceName As String)
  '///Process///
End Sub

Private Async Sub openPrinter()
    Dim printer As Print = New Print()

    Dim onlineEvent As OnlineEvent = New OnlineEvent(AddressOf Online_Event)
    printer.SetOnlineEventCallback(onlineEvent)

    Try
        Await printer.OpenPrintAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                        Monitoring.MONITORING_TRUE, Print.PARAM_DEFAULT)
        '///Process///
    Catch ex As Exception
        if ex.HResult = EposHResult.HR_E_FAIL Then
            '///Process///
        End If
      '///Process///
    End Try
End Sub
```

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 OpenPrinterAsync (p.133).
- When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten.

Syntax

Visual C#

```
public void SetOfflineEventCallback
            (LibEposPrint.OfflineEvent target);
```

Visual Basic .NET

```
Public Sub SetOfflineEventCallback
            (target As LibEposPrint.OfflineEvent)
```

Parameter

- target : This specifies the object that has the notification destination method (definition of call-back method).
  To cancel the registration of the notification destination, specify as follows:

Development Language	Code
Visual C#	null
Visual Basic .NET	Nothing

Definition of Callback Method

Visual C#

```
void Method name(String deviceName)
```

Visual Basic .NET

```
Sub Method name(deviceName As String)
```

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.

Example

- Visual C#

```
private void Offline_Event(string deviceName)
{
    ///Process///
}

private async void openPrinter()
{
    Print printer = new Print();

    OfflineEvent offlineEvent = new OfflineEvent(Offline_Event);
    printer.SetOfflineEventCallback(offlineEvent);

    try
    {
        await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                                       Monitoring.MONITORING_TRUE,
                                       Print.PARAM_DEFAULT);
        ///Process///
    }
    catch (Exception ex)
    {
        if (ex.HResult == EposHResult.HR_E_FAIL)
        {
            ///Process///
        }
    }
}
```

- Visual Basic .NET

```
Private Sub Offline_Event(deviceName As String)
  '///Process///
End Sub

Private Async Sub openPrinter()
    Dim printer As Print = New Print()

    Dim offlineEvent As OfflineEvent = New OfflineEvent(AddressOf Offline_Event)
    printer.SetOfflineEventCallback(offlineEvent)

    Try
        Await printer.OpenPrintAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                        Monitoring.MONITORING_TRUE, Print.PARAM_DEFAULT)
        '///Process///
    Catch ex As Exception
        if ex.HResult = EposHResult.HR_E_FAIL Then
            '///Process///
        End If
      '///Process///
    End Try
End Sub
```

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 OpenPrinterAsync (p.133).
- When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten.

Syntax

Visual C#

```
public void SetPowerOffEventCallback
            (LibEposPrint.PowerOffEvent target);
```

Visual Basic .NET

```
Public Sub SetPowerOffEventCallback
            (target As LibEposPrint.PowerOffEvent)
```

Parameter

- target : This specifies the object that has the notification destination method (definition of call-back method).
  To cancel the registration of the notification destination, specify as follows:

Development Language	Code
Visual C#	null
Visual Basic .NET	Nothing

Definition of Callback Method

Visual C#

```
void Method name(String deviceName)
```

Visual Basic .NET

```
Sub Method name(deviceName As String)
```

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.

Example

- Visual C#

```
private void Power_Off_Event(string deviceName)
{
    ///Process///
}

private async void openPrinter()
{
    Print printer = new Print();

    PowerOffEvent powerOffEvent = new PowerOffEvent(Power_Off_Event);
    printer.SetPowerOffEventCallback(powerOffEvent);

    try
    {
        await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                                       Monitoring.MONITORING_TRUE,
                                       Print.PARAM_DEFAULT);
        ///Process///
    }
    catch (Exception ex)
    {
        if (ex.HResult == EposHResult.HR_E_FAIL)
        {
            ///Process///
        }
    }
}
```

- Visual Basic .NET

```
Private Sub Power_Off_Event(deviceName As String)
  '///Process///
End Sub

Private Async Sub openPrinter()
    Dim printer As Print = New Print()

    Dim powerOffEvent As PowerOffEvent
                              = New PowerOffEvent(AddressOf Power_Off_Event)
    printer.SetPowerOffEventCallback(powerOffEvent)

    Try
        Await printer.OpenPrintAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                         Monitoring.MONITORING_TRUE, Print.PARAM_DEFAULT)
        '///Process///
    Catch ex As Exception
        if ex.HResult = EposHResult.HR_E_FAIL Then
            '///Process///
        End If
      '///Process///
    End Try
End Sub
```

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 OpenPrinterAsync (p.133).
- When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten.

Syntax

Visual C#

```
public void SetCoverOkEventCallback
            (LibEposPrint.CoverOkEvent target);
```

Visual Basic .NET

```
Public Sub SetCoverOkEventCallback
            (target As LibEposPrint.CoverOkEvent)
```

Parameter

- target : This specifies the object that has the notification destination method (definition of call-back method).
  To cancel the registration of the notification destination, specify as follows:

Development Language	Code
Visual C#	null
Visual Basic .NET	Nothing

Definition of Callback Method

Visual C#

```
void Method name(String deviceName)
```

Visual Basic .NET

```
Sub Method name(deviceName As String)
```

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.

Example

- Visual C#

```
private void Cover_Ok_Event(string deviceName)
{
    ///Process///
}

private async void openPrinter()
{
    Print printer = new Print();

    CoverOkEvent coverOkEvent = new CoverOkEvent(Cover_Ok_Event);
    printer.SetCoverOkEventCallback(coverOkEvent);

    try
    {
        await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                                       Monitoring.MONITORING_TRUE,
                                       Print.PARAM_DEFAULT);
        ///Process///
    }
    catch (Exception ex)
    {
        if (ex.HResult == EposHResult.HR_E_FAIL)
        {
            ///Process///
        }
    }
}
```

- Visual Basic .NET

```
Private Sub Cover_Ok_Event(deviceName As String)
  '///Process///
End Sub

Private Async Sub openPrinter()
    Dim printer As Print = New Print()

    Dim coverOkEvent As CoverOkEvent = New CoverOkEvent(AddressOf Cover_Ok_Event)
    printer.SetCoverOkEventCallback(coverOkEvent)

    Try
        Await printer.OpenPrintAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                        Monitoring.MONITORING_TRUE, Print.PARAM_DEFAULT)
        '///Process///
    Catch ex As Exception
        if ex.HResult = EposHResult.HR_E_FAIL Then
            '///Process///
        End If
      '///Process///
    End Try
End Sub
```

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 OpenPrinterAsync (p.133).
- When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten.

Syntax

Visual C#

```
public void SetCoverOpenEventCallback
            (LibEposPrint.CoverOpenEvent target);
```

Visual Basic .NET

```
Public Sub SetCoverOpenEventCallback
           (target As LibEposPrint.CoverOpenEvent)
```

Parameter

- target : This specifies the object that has the notification destination method (definition of call-back method).
  To cancel the registration of the notification destination, specify as follows:

Development Language	Code
Visual C#	null
Visual Basic .NET	Nothing

Definition of Callback Method

Visual C#

```
void Method name(String deviceName)
```

Visual Basic .NET

```
Sub Method name(deviceName As String)
```

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.

Example

- Visual C#

```
private void Cover_Open_Event(string deviceName)
{
    ///Process///
}

private async void openPrinter()
{
    Print printer = new Print();

    CoverOpenEvent coverOpenEvent = new CoverOpenEvent(Cover_Open_Event);
    printer.SetCoverOpenEventCallback(coverOpenEvent);

    try
    {
        await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                                       Monitoring.MONITORING_TRUE,
                                       Print.PARAM_DEFAULT);
        ///Process///
    }
    catch (Exception ex)
    {
        if (ex.HResult == EposHResult.HR_E_FAIL)
        {
            ///Process///
        }
    }
}
```

- Visual Basic .NET

```
Private Sub Cover_Open_Event(deviceName As String)
  '///Process///
End Sub

Private Async Sub openPrinter()
    Dim printer As Print = New Print()

    Dim coverOpenEvent As CoverOpenEvent
                              = New CoverOpenEvent(AddressOf Cover_Open_Event)
    printer.SetCoverOpenEventCallback(coverOpenEvent)

    Try
        Await printer.OpenPrintAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                         Monitoring.MONITORING_TRUE, Print.PARAM_DEFAULT)
        '///Process///
    Catch ex As Exception
        if ex.HResult = EposHResult.HR_E_FAIL Then
            '///Process///
        End If
      '///Process///
    End Try
End Sub
```

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 OpenPrinterAsync (p.133).
- When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten.

Syntax

Visual C#

```
public void SetPaperOkEventCallback
            (LibEposPrint.PaperOkEvent target);
```

Visual Basic .NET

```
Public Sub SetPaperOkEventCallback
            (target As LibEposPrint.PaperOkEvent)
```

Parameter

- target : This specifies the object that has the notification destination method (definition of call-back method).
  To cancel the registration of the notification destination, specify as follows:

Development Language	Code
Visual C#	null
Visual Basic .NET	Nothing

Definition of Callback Method

Visual C#

```
void Method name(String deviceName)
```

Visual Basic .NET

```
Sub Method name(deviceName As String)
```

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.

Example

- Visual C#

```
private void Paper_Ok_Event(string deviceName)
{
    ///Process///
}

private async void openPrinter()
{
    Print printer = new Print();

    PaperOkEvent paperOkEvent = new PaperOkEvent(Paper_Ok_Event);
    printer.SetPaperOkEventCallback(paperOkEvent);

    try
    {
        await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                                       Monitoring.MONITORING_TRUE,
                                       Print.PARAM_DEFAULT);
        ///Process///
    }
    catch (Exception ex)
    {
        if (ex.HResult == EposHResult.HR_E_FAIL)
        {
            ///Process///
        }
    }
}
```

- Visual Basic .NET

```
Private Sub Paper_Ok_Event(deviceName As String)
  '///Process///
End Sub

Private Async Sub openPrinter()
    Dim printer As Print = New Print()

    Dim paperOkEvent As PaperOkEvent = New PaperOkEvent(AddressOf Paper_Ok_Event)
    printer.SetPaperOkEventCallback(paperOkEvent)

    Try
        Await printer.OpenPrintAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                        Monitoring.MONITORING_TRUE, Print.PARAM_DEFAULT)
        '///Process///
    Catch ex As Exception
        if ex.HResult = EposHResult.HR_E_FAIL Then
            '///Process///
        End If
      '///Process///
    End Try
End Sub
```

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 OpenPrinterAsync (p.133).
- When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten.

Syntax

Visual C#

```
public void SetPaperNearEndEventCallback
            (LibEposPrint.PaperNearEndEvent target);
```

Visual Basic .NET

```
Public Sub SetPaperNearEndEventCallback
            (target As LibEposPrint.PaperNearEndEvent)
```

Parameter

- target : This specifies the object that has the notification destination method (definition of call-back method).
  To cancel the registration of the notification destination, specify as follows:

Development Language	Code
Visual C#	null
Visual Basic .NET	Nothing

Definition of Callback Method

Visual C#

```
void Method name(String deviceName)
```

Visual Basic .NET

```
Sub Method name(deviceName As String)
```

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.

Example

- Visual C#

```
private void Paper_Near_End_Event(string deviceName)
{
    ///Process///
}

private async void openPrinter()
{
    Print printer = new Print();

    PaperNearEndEvent paperNearEndEvent
                                        = new PaperNearEndEvent(Paper_Near_End_Event);
    printer.SetPaperNearEndEventCallback(paperNearEndEvent);

    try
    {
        await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                                       Monitoring.MONITORING_TRUE,
                                       Print.PARAM_DEFAULT);
        ///Process///
    }
    catch (Exception ex)
    {
        if (ex.HResult == EposHResult.HR_E_FAIL)
        {
            ///Process///
        }
    }
}
```

- Visual Basic .NET

```
Private Sub Paper_Near_End_Event(deviceName As String)
  '///Process///
End Sub

Private Async Sub openPrinter()
    Dim printer As Print = New Print()

    Dim paperNearEndEvent As PaperNearEndEvent
                          = New PaperNearEndEvent(AddressOf Paper_Near_End_Event)
    printer.SetPaperNearEndEventCallback(paperNearEndEvent)

    Try
        Await printer.OpenPrintAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                         Monitoring.MONITORING_TRUE, Print.PARAM_DEFAULT)
        '///Process///
    Catch ex As Exception
        if ex.HResult = EposHResult.HR_E_FAIL Then
            '///Process///
        End If
      '///Process///
    End Try
End Sub
```

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 OpenPrinterAsync (p.133).
- When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten.

Syntax

Visual C#

```
public void SetPaperEndEventCallback
            (LibEposPrint.PaperEndEvent target);
```

Visual Basic .NET

```
Public Sub SetPaperEndEventCallback
            (target As LibEposPrint.PaperEndEvent)
```

Parameter

- target : This specifies the object that has the notification destination method (definition of call-back method).
  To cancel the registration of the notification destination, specify as follows:

Development Language	Code
Visual C#	null
Visual Basic .NET	Nothing

Definition of Callback Method

Visual C#

```
void Method name(String deviceName)
```

Visual Basic .NET

```
Sub Method name(deviceName As String)
```

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.

Example

- Visual C#

```
private void Paper_End_Event(string deviceName)
{
    ///Process///
}

private async void openPrinter()
{
    Print printer = new Print();

    PaperEndEvent paperEndEvent = new PaperEndEvent(Paper_End_Event);
    printer.SetPaperEndEventCallback(paperEndEvent);

    try
    {
        await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                                       Monitoring.MONITORING_TRUE,
                                       Print.PARAM_DEFAULT);
        ///Process///
    }
    catch (Exception ex)
    {
        if (ex.HResult == EposHResult.HR_E_FAIL)
        {
            ///Process///
        }
    }
}
```

- Visual Basic .NET

```
Private Sub Paper_End_Event(deviceName As String)
  '///Process///
End Sub

Private Async Sub openPrinter()
    Dim printer As Print = New Print()

    Dim paperEndEvent As PaperEndEvent
                      = New PaperEndEvent(AddressOf Paper_End_Event)
    printer.SetPaperEndEventCallback(paperEndEvent)

    Try
        Await printer.OpenPrintAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                        Monitoring.MONITORING_TRUE, Print.PARAM_DEFAULT)
        '///Process///
    Catch ex As Exception
        if ex.HResult = EposHResult.HR_E_FAIL Then
            '///Process///
        End If
      '///Process///
    End Try
End Sub
```

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 OpenPrinterAsync (p.133).
- When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten.

Syntax

Visual C#

```
public void SetDrawerClosedEventCallback
            (LibEposPrint.DrawerClosedEvent target);
```

Visual Basic .NET

```
Public Sub SetDrawerClosedEventCallback
            (target As LibEposPrint.DrawerClosedEvent)
```

Parameter

- target : This specifies the object that has the notification destination method (definition of call-back method).
  To cancel the registration of the notification destination, specify as follows:

Development Language	Code
Visual C#	null
Visual Basic .NET	Nothing

Definition of Callback Method

Visual C#

```
void Method name(String deviceName)
```

Visual Basic .NET

```
Sub Method name(deviceName As String)
```

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.

Example

- Visual C#

```
private void Drawer_Closed_Event(string deviceName)
{
    ///Process///
}

private async void openPrinter()
{
    Print printer = new Print();

    DrawerClosedEvent drawerCLosedEvent
                                    = new DrawerClosedEvent(Drawer_Closed_Event);
    printer.SetDrawerClosedEventCallback(drawerCLosedEvent);

    try
    {
        await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                                       Monitoring.MONITORING_TRUE,
                                       Print.PARAM_DEFAULT);
        ///Process///
    }
    catch (Exception ex)
    {
        if (ex.HResult == EposHResult.HR_E_FAIL)
        {
            ///Process///
        }
    }
}
```

- Visual Basic .NET

```
Private Sub Drawer_Closed_Event(deviceName As String)
  '///Process///
End Sub

Private Async Sub openPrinter()
    Dim printer As Print = New Print()

    Dim Dim drawerClosedEvent As DrawerClosedEvent
                        = New DrawerClosedEvent(AddressOf Drawer_Closed_Event)
    printer.SetDrawerClosedEventCallback(drawerClosedEvent)

    Try
        Await printer.OpenPrintAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                        Monitoring.MONITORING_TRUE, Print.PARAM_DEFAULT)
        '///Process///
    Catch ex As Exception
        if ex.HResult = EposHResult.HR_E_FAIL Then
            '///Process///
        End If
      '///Process///
    End Try
End Sub
```

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 OpenPrinterAsync (p.133).
- When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten.

Syntax

Visual C#

```
public void SetDrawerOpenEventCallback
            (LibEposPrint.DrawerOpenEvent target);
```

Visual Basic .NET

```
Public Sub SetDrawerOpenEventCallback
            (target As LibEposPrint.DrawerOpenEvent)
```

Parameter

- target : This specifies the object that has the notification destination method (definition of call-back method).
  To cancel the registration of the notification destination, specify as follows:

Development Language	Code
Visual C#	null
Visual Basic .NET	Nothing

Definition of Callback Method

Visual C#

```
void Method name(String deviceName)
```

Visual Basic .NET

```
Sub Method name(deviceName As String)
```

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.

Example

- Visual C#

```
private void Drawer_Open_Event(string deviceName)
{
    ///Process///
}

private async void openPrinter()
{
    Print printer = new Print();

    DrawerOpenEvent drawerOpenEvent = new DrawerOpenEvent(Drawer_Open_Event);
    printer.SetDrawerOpenEventCallback(drawerOpenEvent);

    try
    {
        await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                                       Monitoring.MONITORING_TRUE,
                                       Print.PARAM_DEFAULT);
        ///Process///
    }
    catch (Exception ex)
    {
        if (ex.HResult == EposHResult.HR_E_FAIL)
        {
            ///Process///
        }
    }
}
```

- Visual Basic .NET

```
Private Sub Drawer_Open_Event(deviceName As String)
  '///Process///
End Sub

Private Async Sub openPrinter()
    Dim printer As Print = New Print()

    Dim drawerOpenEvent As DrawerOpenEvent
                        = New DrawerOpenEvent(AddressOf Drawer_Open_Event)
    printer.SetDrawerOpenEventCallback(drawerOpenEvent)

    Try
        Await printer.OpenPrintAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                         Monitoring.MONITORING_TRUE, Print.PARAM_DEFAULT)
        '///Process///
    Catch ex As Exception
        if ex.HResult = EposHResult.HR_E_FAIL Then
            '///Process///
        End If
      '///Process///
    End Try
End Sub
```

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 OpenPrinterAsync (p.133).
- When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten.

Syntax

Visual C#

```
public void SetBatteryLowEventCallback
            (LibEposPrint.BatteryLowEvent target);
```

Visual Basic .NET

```
Public Sub SetBatteryLowEventCallback
            (target As LibEposPrint.BatteryLowEvent)
```

Parameter

- target : This specifies the object that has the notification destination method (definition of call-back method).
  To cancel the registration of the notification destination, specify as follows:

Development Language	Code
Visual C#	null
Visual Basic .NET	Nothing

Definition of Callback Method

Visual C#

```
void Method name(String deviceName)
```

Visual Basic .NET

```
Sub Method name(deviceName As String)
```

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.

Example

- Visual C#

```
private void Battery_Low_Event(string deviceName)
{
    ///Process///
}

private async void openPrinter()
{
    Print printer = new Print();

    BatteryLowEvent batteryLowEvent = new BatteryLowEvent(Battery_Low_Event);
    printer.SetBatteryLowEventCallback(batteryLowEvent);

    try
    {
        await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                                       Monitoring.MONITORING_TRUE,
                                       Print.PARAM_DEFAULT);
        ///Process///
    }
    catch (Exception ex)
    {
        if (ex.HResult == EposHResult.HR_E_FAIL)
        {
            ///Process///
        }
    }
}
```

- Visual Basic .NET

```
Private Sub Battery_Low_Event(deviceName As String)
  '///Process///
End Sub

Private Async Sub openPrinter()
    Dim printer As Print = New Print()

    Dim batteryLowEvent As BatteryLowEvent
                        = New BatteryLowEvent(AddressOf Battery_Low_Event)
    printer.SetBatteryLowEventCallback(batteryLowEvent)

    Try
        Await printer.OpenPrintAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                        Monitoring.MONITORING_TRUE, Print.PARAM_DEFAULT)
        '///Process///
    Catch ex As Exception
        if ex.HResult = EposHResult.HR_E_FAIL Then
            '///Process///
        End If
      '///Process///
    End Try
End Sub
```

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 OpenPrinterAsync (p.133).
- When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten.

Syntax

Visual C#

```
public void SetBatteryOkEventCallback
            (LibEposPrint.BatteryOkEvent target);
```

Visual Basic .NET

```
Public Sub SetBatteryOkEventCallback
            (target As LibEposPrint.BatteryOkEvent)
```

Parameter

- target : This specifies the object that has the notification destination method (definition of call-back method).
  To cancel the registration of the notification destination, specify as follows:

Development Language	Code
Visual C#	null
Visual Basic .NET	Nothing

Definition of Callback Method

Visual C#

```
void Method name(String deviceName)
```

Visual Basic .NET

```
Sub Method name(deviceName As String)
```

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.

Example

- Visual C#

```
private void Battery_Ok_Event(string deviceName)
{
    ///Process///
}

private async void openPrinter()
{
    Print printer = new Print();

    BatteryOkEvent batteryOkEvent = new BatteryOkEvent(Battery_Ok_Event);
    printer.SetBatteryOkEventCallback(batteryOkEvent);

    try
    {
        await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                                       Monitoring.MONITORING_TRUE,
                                       Print.PARAM_DEFAULT);
        ///Process///
    }
    catch (Exception ex)
    {
        if (ex.HResult == EposHResult.HR_E_FAIL)
        {
            ///Process///
        }
    }
}
```

- Visual Basic .NET

```
Private Sub Battery_Ok_Event(deviceName As String)
  '///Process///
End Sub

Private Async Sub openPrinter()
    Dim printer As Print = New Print()

    Dim batteryOkEvent As BatteryOkEvent
                       = New BatteryOkEvent(AddressOf Battery_Ok_Event)
    printer.SetBatteryOkEventCallback(batteryOkEvent)

    Try
        Await printer.OpenPrintAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                         Monitoring.MONITORING_TRUE, Print.PARAM_DEFAULT)
        '///Process///
    Catch ex As Exception
        if ex.HResult = EposHResult.HR_E_FAIL Then
            '///Process///
        End If
      '///Process///
    End Try
End Sub
```

SetBatteryStatusChangeEventCallback

This registers the notification destination of battery status.

- This API can be executed following execution of OpenPrinterAsync (p.133).
- When this API is executed on multiple occasions, the notification destination that is specified afterwards is overwritten.

Syntax

Visual C#

```
public void SetBatteryStatusChangeEventCallback
      (LibEposPrint.BatteryStatusChangeEvent target);
```

Visual Basic .NET

```
Public Sub SetBatteryStatusChangeEventCallback
       (target As LibEposPrint.BatteryStatusChangeEvent)
```

Parameter

- target : This specifies the object that has the notification destination method (definition of call-back method).
  To cancel the registration of the notification destination, specify as follows:

Development Language	Code
Visual C#	null
Visual Basic .NET	Nothing

Definition of Callback Method

Visual C#

```
void Method name(String deviceName, int battery)
```

Visual Basic .NET

```
Sub Method name(deviceName As String,
                battery As Integer)
```

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.
- battery : Battery status is set.

Example

- Visual C#

```
private void Battery_Status_Change_Event(string deviceName, int battery)
{
     ///Process///
}

private async void openPrinter()
{
    Print printer = new Print();
    BatteryStatusChangeEvent batteryStatusChangeEvent
                  = new BatteryStatusChangeEvent(Battery_Status_Change_Event);
    printer.SetBatteryStatusChangeEventCallback(batteryStatusChangeEvent);

    try
    {
        await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                                       Monitoring.MONITORING_TRUE,
                                       Print.PARAM_DEFAULT);
         ///Process///
    }
    catch (Exception ex)
    {
        if (ex.HResult == EposHResult.HR_E_FAIL)
        {
             ///Process///
        }
    }
}
```

- Visual Basic .NET

```
Private Sub Battery_Status_Change_Event(deviceName As String, battery As Integer)
  '///Process///
End Sub

Private Async Sub openPrinter()
    Dim printer As Print = New Print()

    Dim batteryStatusChangeEvent As BatteryStatusChangeEvent
             = New BatteryStatusChangeEvent(AddressOf Battery_Status_Change_Event)
    printer.SetBatteryStatusChangeEventCallback(batteryStatusChangeEvent)

    Try
        Await printer.OpenPrintAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                         Monitoring.MONITORING_TRUE, Print.PARAM_DEFAULT)
        '///Process///
    Catch ex As Exception
        if ex.HResult = EposHResult.HR_E_FAIL Then
            '///Process///
        End If
      '///Process///
    End Try
End Sub
```

GetStatusAsync

Acquires the printer status and the battery status.
In addition to the printer statuses acquired by SendDataAsync (p.141), this API can acquire the following printer statuses.

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

- This API is an asynchronous API. Wait until the process by this API ends to execute a new process.
- This API does not support cancellation of asynchronous operation by Windows.Foundation.IAsyncInfo.Cancel().

Syntax

Visual C#

```
public Windows.Foundation.IAsyncOperation
                <PrinterStatus> getStatusAsync();
```

Visual Basic .NET

```
Public Function GetStatusAsync() As
                Windows.Foundation.IAsyncOperation
                (Of LibEposPrint.PrinterStatus)
```

Return value

The printer status and the battery status return.

- The printer status returns to PrinterStatus.printerStatus.
  This returns the printer status at the time this API was executed.
  For details on the sprinter status, see Printer Statuses and Actions to Take (p.47).
- The battery status returns to PrinterStatus.batteryStatus.
  This returns the battery status at the time this API was executed.
  For details on the battery status, see Support Information by Printer (p.208).

Exceptions

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

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	• An attempt was made to start communicating with the device with which communication had already started. • An unspecified error occurred.

Example

- Visual C#

```
try
{
  Print printer = new Print();
  PrinterStatus status = new PrinterStatus();
  await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                        Monitoring.MONITORING_TRUE, Print.PARAM_DEFAULT);
  status = await printer.getStatusAsync();
  ///Process///
  await printer.ClosePrinterAsync();
}
catch (Exception ex)
{
  if (ex.HResult == EposHResult.HR_E_FAIL)
    {
      ///Process///
    }
  ///Process///
}
```

- Visual Basic .NET

```
Try
  Dim printer As Print = New Print()
  Dim status As PrinterStatus = New PrinterStatus()
  Await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                       Monitoring.MONITORING_TRUE, Print.PARAM_DEFAULT)
  Status = Await printer.getStatusAsync()
  `///Process///
  Await printer.ClosePrinterAsync()
Catch ex As Exception
  if ex.HResult = EposHResult.HR_E_FAIL Then
    `///Process///
  End If
  `///Process///
End Try
```

GetPrinterStatus

Acquires the printer status and battery status from an exception that occurred in SendDataAsync (p.141).

Syntax

Visual C#

```
public static LibEposPrint.PrinterStatus
              GetPrinterStatus(int hresult);
```

Visual Basic .NET

```
Public Function GetPrinterStatus(hresult As Integer)
                As LibEposPrint.PrinterStatus
```

Parameter

- hresult : This specifies HResult as an exception.

Return value

Returns the printer status and battery status.

- The printer status is returned to PrinterStatus.printerStatus.
  For details, refer to Printer Statuses and Actions to Take (p.47).
- The battery status is returned to PrinterStatus.batteryStatus.
  For details, refer to Battery Status (p.49).

Example

To acquire the printer status from Exception.

- Visual C#

```
Print printer = new Print();
PrinterStatus status;
status.printerStatus = 0;
status.batteryStatus = 0;

try
{
    Builder builder = new Builder("TM-T88V", ModelLang.MODEL_LANG_ANK);
    ///Process///
    await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                Monitoring.MONITORING_TRUE, Print.PARAM_DEFAULT);
    status = await printer.SendDataAsync(builder, 10000);
    ///Process///
    await printer.ClosePrinterAsync();
}
catch (Exception ex)
{
    status = printer.GetPrinterStatus(ex.HResult);
    ///Process///
    if((status.printerStatus & Print.ST_PRINT_SUCCESS) == Print.ST_PRINT_SUCCESS)
    {
        ///Process///
    }
}
```

- Visual Basic .NET

```
Dim printer As Print = New Print()
Dim status As PrinterStatus
status.printerStatus = 0
status.batteryStatus = 0

Try
    Dim builder As Builder = New Builder("TM-T88V", ModelLang.MODEL_LANG_ANK)
    builder.AddText("ABCDE" + vbCrLf)

    Await printer.OpenPrinterAsync(DevType.DEVTYPE_TCP, "192.168.192.168",
                                Monitoring.MONITORING_TRUE, Print.PARAM_DEFAULT)
    status = Await printer.SendDataAsync(builder, 10000)
    Await printer.ClosePrinterAsync()

    `///Process///

Catch ex As Exception
    Dim errStatus As PrinterStatus = printer.GetPrinterStatus(ex.HResult)
    If (status.printerStatus And Print.ST_PRINT_SUCCESS)
                                    = Print.ST_PRINT_SUCCESS Then
        `///Process///
    End If
    `///Process///
End Try
```

Printer Search API

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

❏ Finder class (p. 180)

Finder class

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

API	Description	Page
StartAsync	Starts searching for printers.	180
StopAsync	End communication with the printer.	182
GetDeviceInfoList	Getting the printer search result.	183
GetResult(Previous format)		185

StartAsync

Starts a search for printers of the specified device type.

If you use this API, be sure to use StopAsync (p.182) to stop the search.

- For a TCP device search, if multiple network adapters exist, a search is made for an adapter for which the number of hops becomes minimum.
- You cannot call this API when a printer search is already in progress.
- This API is an asynchronous API. Wait until the process by this API ends to execute a new process.
- This API does not support cancellation of asynchronous operation by Windows.Foundation.IAsyncInfo.Cancel().

Syntax

Visual C#

```
public static Windows.Foundation.IAsyncAction
       StartAsync
       (LibEposPrint.IoDevType deviceType,
        System.String findOption);
```

Visual Basic .NET

```
Public Shared Function StartAsync
                (deviceType As LibEposPrint.IoDevType,
                 findOption As String)
                 As Windows.Foundation.IAsyncAction
```

Parameter

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

deviceType	Description
IoDevType.TCP	Searches for TM devices connected to the network
IoDevType.*BLUETOOTH*	Searches for *Bluetooth*® devices already paired and connected via serial port.

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

deviceType	Setting Value
IoDevType.TCP	The broadcast address to search for
IoDevType.*BLUETOOTH*	"" (Empty characters)

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_PENDING (0x8000000A)	Could not execute process.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	• This API was called when a search was already in progress • An unspecified error occurred.

StopAsync

Stops the printer search.

- This API is an asynchronous API. Wait until the process by this API ends to execute a new process.
- This API does not support cancellation of asynchronous operation by Windows.Foundation.IAsyncInfo.Cancel().

Syntax

Visual C#

```
public static Windows.Foundation.IAsyncAction
       StopAsync();
```

Visual Basic .NET

```
Public Shared Function StopAsync()
                  As Windows.Foundation.IAsyncAction
```

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_PENDING (0x8000000A)	Could not execute process.
HR_E_FAIL (0x80004005)	• This API was called when a search was already in progress • An unspecified error occurred.

GetDeviceInfoList

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

- This API is an asynchronous API. Wait until the process by this API ends to execute a new process.
- This API does not support cancellation of asynchronous operation by Windows.Foundation.IAsyncInfo.Cancel().

Syntax

Visual C#

```
public static DeviceInfo[]
            GetDeviceInfoList(LibEposPrint.IoFilterOpt
            ion filterOption);
```

Visual Basic .NET

```
Public Shared Function GetDeviceInfoList(filterOption As
              LibEposPrint.IoFilterOption) As
              DeviceInfo()
```

Parameter

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

deviceType	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 connection, only Epson printers are searched for regardless of the filterOption setting.

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	deviceType	List to Acquire
IoDevType.TCP	DeviceType	DevType.TCP(Fixed)
	PrinterName	Printer model name
	DeviceName	• DHCP disabled: IP address • DHCP enabled: MAC address
	IpAddress	IP Address
	MacAddress	MAC Address
IoDevType.*BLUETOOTH*	DeviceType	DevType.*BLUETOOTH*(Fixed)
	PrinterName	*Bluetooth*® device name
	DeviceName	BD Address (the same format as the MAC address format)
	IpAddress	"" (Empty character)
	MacAddress	"" (Empty character)

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_PENDING (0x8000000A)	Could not execute process.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	• This API was called when a search was already in progress • An unspecified error occurred.

GetResult(Previous format)

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

- This API is an asynchronous API. Wait until the process by this API ends to execute a new process.
- This API does not support cancellation of asynchronous operation by Windows.Foundation.IAsyncInfo.Cancel().

Syntax

Visual C#

```
public static System.String[] GetResult();
```

Visual Basic .NET

```
Public Shared Function GetResult() As String()
```

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
IoDevType.TCP	List of IP addresses of printers
IoDevType.*BLUETOOTH*	List of BD addresses of *Bluetooth*® devices

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_PENDING (0x8000000A)	Could not execute process.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	• This API was called when a search was already in progress • An unspecified error occurred.

Log Setting API

Sets the log output. The following class is available.

- Log class (p. 186)

Log class

Sets the log output function.

API	Description	Page
SetLogSettings	Sets the log output function.	186

SetLogSettings

Sets the log output function.

> When the log output function is enabled for TCP communication, it may take a longer period of time to process each API depending on the environment.

Syntax

Visual C#

```
public static void SetLogSettings
         (LibEposPrint.LogPeriod period,
          LibEposPrint.LogEnabled enabled,
          System.String ipAddress, int port,
          int logSize, LibEposPrint.LogLevel logLevel);
```

Visual Basic .NET

```
Public Shared Sub SetLogSettings
              (period As LibEposPrint.LogPeriod,
               enabled As LibEposPrint.LogEnabled,
               ipAddress As String, port As Integer,
               logSize As Integer,
               logLevel As LibEposPrint.LogLevel)
```

Parameter

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

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

> To specify period for the LogPeriod.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
LogEnabled.LOG_DISABLE	Disables the log output function.
LogEnabled.LOG_STORAGE	Outputs log data to the device's storage.
LogEnabled.LOG_TCP	Outputs log data over TCP.

- To specify enabled for the LogEnabled.LOG_STORAGE, set the permissions for the application to access the storage.
- To specify enabled for the LogEnabled.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, "" (empty characters) can be specified for this parameter.
- ∗ LogEnabled.LOG_DISABLE
- ∗ LogEnabled.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.
- ∗ LogEnabled.LOG_DISABLE
- ∗ LogEnabled.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.
- ∗ LogEnabled.LOG_DISABLE
- ∗ LogEnabled.LOG_TCP

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

Set value	Description
LogLevel.LOG_LOW	Low level

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_FAIL (0x80004005)	An unspecified error occurred.

Example

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

- Visual C#

```
try
{
    Log.SetLogSettings(LogPeriod.LOG_PERMANENT, LogEnabled.LOG_TCP,
                       "192.168.192.168", 8080, 10, LogLevel.LOG_LOW);
} catch (Exception ex) {
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Log.SetLogSettings(LogPeriod.LOG_PERMANENT, LogEnabled.LOG_TCP,
                       "192.168.192.168", 8080, 10, LogLevel.LOG_LOW)
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       `///Process///
    End If
    `///Process///
End Try
```

To output log data to the device's storage:

- Visual C#

```
try
{
    Log.SetLogSettings(LogPeriod.LOG_PERMANENT, LogEnabled.LOG_STORAGE, "",
                       0, 10, LogLevel.LOG_LOW);
} catch (Exception ex) {
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Log.SetLogSettings(LogPeriod.LOG_PERMANENT, LogEnabled.LOG_STORAGE, "",
                       0, 10, LogLevel.LOG_LOW)
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       `///Process///
    End If
    `///Process///
End Try
```

To disable the log output function:

- Visual C#

```
try
{
    Log.SetLogSettings(LogPeriod.LOG_PERMANENT, LogEnabled.LOG_DISABLE, "",
                       0, 10, LogLevel.LOG_LOW);
} catch (Exception ex) {
    if (ex.HResult == EposHResult.HR_E_FAIL)
    {
        ///Process///
    }
    ///Process///
}
```

- Visual Basic .NET

```
Try
    Log.SetLogSettings(LogPeriod.LOG_PERMANENT, LogEnabled.LOG_DISABLE, "",
                       0, 10, LogLevel.LOG_LOW)
Catch ex As Exception
    If ex.HResult = EposHResult.HR_E_FAIL Then
       `///Process///
    End If
    `///Process///
End Try
```

How to Extract a Log File

Save destination

❑ %LOCALAPPDATA%/Packages/(application package name)/LocalState/EposLog

File name

File name	Description
EposLog.txt	The upper limit of the number of lines in the log file is 5MB.
EposLog.txt.<X>.zip <Example:> EposLog.txt.1.zip	When number of lines in the log file reaches the upper limit, it is compressed in the ZIP format.

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.53) of ePOS-Print SDK for Windows Store apps. • Descriptions made using source code in this chapter are based on Visual C#. For any language other than the above, read such descriptions in ways that suit the relevant language.

Programming

Programming Flow

Perform programming following this flow.

1. Printer Selection (p.34) *
   - ❏ Starting the printer search (p.34)
   - ❏ Getting the printer search result. (p.34)
   - ❏ Stopping the printer search (p.34)

↓

2. Opening a Device Port (p.192)

↓

3. Sending Data (p.192)

↓

4. Receiving Data (p.193)

↓

5. Closing the Device Port (p.193)

* This is optional.

Opening a Device Port

Use the EpsonIo class's OpenAsync (p.196) to open a device port. Please refer to the following code.

```
try {
//Initialize the EpsonIo class
    EpsonIo port = new EpsonIo();

//Open the device port
    //<Wi-Fi/Ethernet device>
    async port.OpenAsync(IoDevType.TCP, "192.168.192.168", "");
    //<Bluetooth device>
    async port.OpenAsync(IoDevType.BLUETOOTH, "00:00:12:34:56:78", "");
}
//Exception processing
catch (Exception ex)
{
    int errCode = ex.HResult;
}
```

Sending Data

Use the EpsonIo class's WriteBytes (p.199) to buffer data and send data using WriteAsync (p.200). Please refer to the following code.

Printing out "Hello, World!"

```
//Settings for sending
System.String str = "Hello World!\n";
byte[] data = System.Text.Encoding.GetEncoding("shift_jis").GetBytes(str);
int offset = 0;
int size = data.length;
int timeout = 5000;
int sizeCopy = 0;
int sizeWritten = 0;

try {
//Buffer data
    sizeCopy = port.WriteBytes(data, offset, size);

//Send data
    sizeWritten = await port.WriteAsync(timeout);
}
//Exception processing
catch (Exception ex)
{
    int errCode = ex.HResult;
}
```

Receiving Data

Use the EpsonIo class's ReadAsync (p.204) to receive data from a printer and read data using ReadBytes (p.202). 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 sizeCopy = 0;

try {
//Receive data
    sizeRead = await port.ReadAsync(size, timeout);

//Read data
    sizeCopy = port.ReadBytes(out data, data, offset, size);
}
//Exception processing
catch (Exception ex)
{
    int errCode = ex.HResult;
}
```

Closing the Device Port

Use the EpsonIo class's CloseAsync (p.198) to close the device port. Please refer to the following code.

```
try {
    EpsonIo port = new EpsonIo();
    async port.OpenAsync(IoDevType.TCP, "192.168.192.168", "");
    //Close the device port
    async port.CloseAsync();
}
//Exception processing
catch (Exception ex)
{
    int errCode = ex.HResult;
}
```

Exception handling

The command transmission/reception API generates a Windows system exception when an error occurs to notify the calling side of such an error.

Steps for Handling

Please refer to the following code.

```
//Settings for sending
System.String str = "Hello World!\n";
byte[] data = System.Text.Encoding.GetEncoding("utf-8").GetBytes(str);
int offset = 0;
int size = data.length;
int timeout = 5000;
int sizeCopy = 0;
int sizeWritten = 0;

try {
//Buffer data
    sizeCopy = port.WriteBytes(data, offset, size);
//Send data
    sizeWritten = await port.WriteAsync(timeout);
}
//Exception processing
catch (Exception ex)
{
    if(ex.HResult == IoHResult.HR_E_FAIL)
    {
        ///Process///
    }
}
```

List of Error Values

In ePOS-Print SDK for Windows Store apps, a Windows system exception with any of the following error statuses is returned.

Error Value	Cause
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed. <Example> • An invalid parameter such as null was passed. • A value outside the supported range was specified.
HR_E_ACCESSDENIED (0x80070005)	• Open processing failed. • Failed to connect to device. <Example> • Failed to create a TCP and *Bluetooth*® communication socket. • Failed to open the device file to which the path is specified. • 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.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate the necessary memory for processing.
HR_E_PENDING (0x8000000A)	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.
HR_E_FAIL (0x80004005)	• Illegal method used. • An unspecified error occurred. <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.

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
OpenAsync	Opens the device port.	196
CloseAsync	Closes the device port.	198
WriteBytes	Writes sending data.	199
WriteAsync	Send data.	200
ReadBytes	Reads receiving data.	202
ReadAsync	Receive data.	204

OpenAsync

Opens the specified device port.

- If communication with the printer is not required anymore, be sure to call CloseAsync (p.198) to end communication with the printer.
- A maximum of 16 device ports can be opened in the same application at the same time.

- This API is an asynchronous API. Wait until the process by this API ends to execute a new process.
- This API does not support cancellation of asynchronous operation by Windows.Foundation.IAsyncInfo.Cancel().
- When a printer is connected, a connection confirmation screen may appear. Execute this API from a UI thread.

Syntax

Visual C#

```
public Windows.Foundation.IAsyncAction OpenAsync
            (LibEposPrint.IoDevType deviceType,
             System.String deviceName,
             System.String deviceSettings);
```

Visual Basic .NET

```
Public Function OpenAsync
       (deviceType As LibEposPrint.IoDevType,
        deviceName As String, deviceSettings As String)
        As Windows.Foundation.IAsyncAction
```

Parameter

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

deviceType	Description
IoDevType.TCP	Specify this when the printer to be opened will connect with Wi-Fi/Ethernet.
IoDevType.*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
IoDevType.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)
IoDevType.*BLUETOOTH*	BD address (Example: "01:23:45:67:89:AB")

- deviceSettings :
  Specifies "" (empty characters).

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_ACCESSDENIED (0x80070005)	Open processing failed.
HR_E_PENDING (0x8000000A)	Could not execute process.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	• User attempted to open a device that is already open. • An unspecified error occurred.

CloseAsync

Closes the specified device port.

- This API is an asynchronous API. Wait until the process by this API ends to execute a new process.
- This API does not support cancellation of asynchronous operation by Windows.Foundation.IAsyncInfo.Cancel().

Syntax

Visual C#

```
public Windows.Foundation.IAsyncAction CloseAsync();
```

Visual Basic .NET

```
Public Function CloseAsync()
                As Windows.Foundation.IAsyncAction
```

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_PENDING (0x8000000A)	Could not execute process.
HR_E_FAIL (0x80004005)	• This API was called when no device port was open. • An unspecified error occurred.

WriteBytes

Writes sending data into the EpsonIo class's buffer.

Syntax

Visual C#

```
public int WriteBytes(byte[] data, int offset,
                      int size);
```

Visual Basic .NET

```
Public Function WriteBytes(data() As Byte,
          offset As Integer, size As Integer) As Integer
```

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 written to the EpsonIo class's buffer.
In such a case, the return value will be "0".

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

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_PENDING (0x8000000A)	Could not execute process.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	• This API was called when no device port was open. • An unspecified error occurred.

If this API is executed while WriteAsync (p.200) is running, a HR_E_PENDING exception occurs due to exclusive control.

WriteAsync

Sends the EpsonIo class's buffer to a device port.

- This API is an asynchronous API. Wait until the process by this API ends to execute a new process.
- This API does not support cancellation of asynchronous operation by Windows.Foundation.IAsyncInfo.Cancel().
- If CloseAsync (p.198) is called before a print process becomes complete on the printer side, data transmission may be suspended.

Syntax

Visual C#

```
public Windows.Foundation.IAsyncOperation<int>
            WriteAsync(int timeout);
```

Visual Basic .NET

```
Public Function WriteAsync(timeout As Integer) As
        Windows.Foundation.IAsyncOperation(Of Integer)
```

Parameter

- 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

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_ACCESSDENIED (0x80070005)	Connection error occurred.
HR_E_PENDING (0x8000000A)	Could not execute process.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	• This API was called when no device port was open. • An unspecified error occurred.

- If this API is executed by the same object at the same time, a HR_E_PENDING exception occurs due to exclusive control for a process without ownership.
- When this API is called within 20 seconds after the connection to the TM printer is lost, no exception occurs because of *Bluetooth*® device control specifications.

ReadBytes

Reads receiving data from the EpsonIo class's buffer.

This API can be executed on multiple occasions as long as data remains in the EpsonIo class's buffer.

Syntax

Visual C#

```
public int ReadBytes(out byte[] outData, byte[] data,
                     int offset, int size);
```

Visual Basic .NET

```
Public Function ReadBytes
          (ByRef outData() As Byte, data() As Byte,
           offset As Integer, size As Integer) As Integer
```

Parameter

- outData : The receiving data buffer for storing received data.
  (A buffer secured by ePOS-Print SDK)
- data : The receiving data buffer for storing received data.
  (A buffer secured by this API)
- 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".

Return value

The byte size of the data read from the EpsonIo class's buffer is returned.

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_ACCESSDENIED (0x80070005)	Connection error occurred.
HR_E_PENDING (0x8000000A)	Could not execute process.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	• This API was called when no device port was open. • An unspecified error occurred.

If this API is executed while ReadAsync (p.204) is running, a HR_E_PENDING exception occurs due to exclusive control.

ReadAsync

Receives data into the Epsonlo class's buffer from a device port.

- This API is an asynchronous API. Wait until the process by this API ends to execute a new process.
- This API does not support cancellation of asynchronous operation by Windows.Foundation.IAsyncInfo.Cancel().
- This API continues to receive data until a reception error occurs; however, if even a single byte of data fails to be received in the period of time specified for timeout, the process ends.

Syntax

Visual C#

```
public Windows.Foundation.IAsyncOperation<int>
            ReadAsync(int size, int timeout);
```

Visual Basic .NET

```
Public Function ReadAsync
       (size As Integer, timeout As Integer) As
        Windows.Foundation.IAsyncOperation(Of Integer)
```

Parameter

- 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).

  - 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 that were received.

When this API is executed on multiple occasions, all the received data is added.

Exceptions

If processing fails, an Exception returning any of the following HResults occurs.

HResult	Description
HR_E_INVALIDARG (0x80070057)	Invalid parameter was passed.
HR_E_PENDING (0x8000000A)	Could not execute process.
HR_E_ACCESSDENIED (0x80070005)	Connection error occurred.
HR_E_OUTOFMEMORY (0x8007000E)	Could not allocate memory.
HR_E_FAIL (0x80004005)	• This API was called when no device port was open. • An unspecified error occurred.

If this API is executed by an identical object at the same time, a HR_E_PENDING exception occurs due to exclusive control for a process without ownership.

Appendix

List of Supported APIs for Each Printer Model

API	TM-m10	TM-m30	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-T88V	TM-T90II	TM-U220	TM-U330
AddTextAlign (p.58)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
AddTextLineSpace (p.60)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
AddTextRotate (p.61)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
AddText (p.63)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
AddTextLang (p.65)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
AddTextFont (p.67)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
AddTextSmooth (p.69)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	-	-
AddTextDouble (p.71)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	-	-
AddTextSize (p.73)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	-	-
AddTextStyle (p.75)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
AddTextPosition (p.77)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	-	✔
AddFeedUnit (p.79)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
AddFeedLine (p.80)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
AddImageAsync (p.81)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
AddImageAsync (Previous format) (p.85)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
AddLogo (p.90)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	-	-
AddBarcode (p.92)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	-	-
AddSymbol (p.98)	✔	✔	✔	-	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	-	-
AddPageBegin (p.105)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	-	-
AddPageEnd (p.107)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	-	-
AddPageArea (p.109)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	-	-
AddPageDirection (p.111)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	-	-
AddPagePosition (p.113)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	-	-
AddPageLine (p.115)	✔	✔	✔	-	✔	✔	✔	✔	-	-	-	-	-	-	-	-	-	-	-
AddPageRectangle (p.117)	✔	✔	✔	-	✔	✔	✔	✔	-	-	-	-	-	-	-	-	-	-	-
AddCut (p.119)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
AddPulse (p.121)	✔	✔	-	-	-	-	-	-	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
AddSound (p.123)	✔	✔	✔	-	-	✔	✔	✔	-	✔	-	✔	-	✔	✔	✔	-	-	✔
AddFeedPosition (p.125)	-	-	✔	-	✔	-	✔	✔	-	-	-	-	-	-	-	-	-	-	✔
AddLayout (p.127)	-	-	✔	-	✔	-	✔	✔	-	-	-	-	-	-	-	-	-	-	-
AddCommand (p.130)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔

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 *1: 17 characters
	Font B	ANK: 42 characters Kanji *2: 21 characters
	Font C	ANK: 46 characters
Character Size	Font A	ANK: 12 dots x 24 dots (W x H) Kanji *1: 24 dots x 24 dots (W x H)
	Font B	ANK: 10 dots x 24 dots (W x H) Kanji *2: 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 *1: At the 21st dot from the top of the character
	Font B	ANK; At the 21st dot from the top of the character Kanji *2: 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 2400 dots (W x H)
Page Mode Maximum Area		420 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
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		Cut, Feed cut

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

*1 Differs depending on the Multilingual model specifications.

*2 Only for Japanese model.

TM-m30

		58 mm	80 mm
Resolution		203 dpi x 203 dpi (W x H)	
Country		• ANK model • Japanese model • Simplified Chinese model • Traditional Chinese model	
Print Width		420 dots	576 dots
Characters in a Line	Font A	ANK: 35 characters Kanji *1: 17 characters	ANK: 48 characters Kanji *1: 24 characters
	Font B	ANK: 42 characters Kanji *2: 21 characters	ANK: 57 characters Kanji *2: 28 characters
	Font C	ANK: 46 characters	ANK: 64 characters
Character Size	Font A	ANK: 12 dots x 24 dots (W x H) Kanji *1: 24 dots x 24 dots (W x H)	
	Font B	ANK: 10 dots x 24 dots (W x H) Kanji *2: 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 *1: At the 21st dot from the top of the character	
	Font B	ANK; At the 21st dot from the top of the character Kanji *2: 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 2400 dots (W x H)	576 dots x 2400 dots (W x H)
Page Mode Maximum Area		420 dots x 2400 dots (W x H)	576 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	
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		Cut, Feed cut	
Drawer Kick-Out		Supported	

	58 mm	80 mm
Buzzer	Option (Pattern A ~ Pattern E, Error, No paper, Stop)	
Battery	Not supported	

*1 Differs depending on the Multilingual model specifications.

*2 Only for Japanese 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)	
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)

		58 mm	80 mm
Resolution		180 dpi x 180 dpi (W x H)	
Language		ANK model	
Print Width		360 dots	512 dots
Characters in a Line	Font A	ANK: 30 characters	ANK: 42 characters
	Font B	ANK: 40 characters	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 15th dot from the top of the character	
Default Line Feed Space		30 dots	
Color Specification		First color	
Page Mode Default Area		360 dots x 1662 dots (W x H)	512 dots x 1662 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	
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)
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	

*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-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		Codabar, Code39, ITF, JAN13(EAN), JAN8(EAN), UPC-A, UPC-E, Code93, Code128, GS1-128, GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Expanded, GS1 DataBar Limited	
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.75) 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		

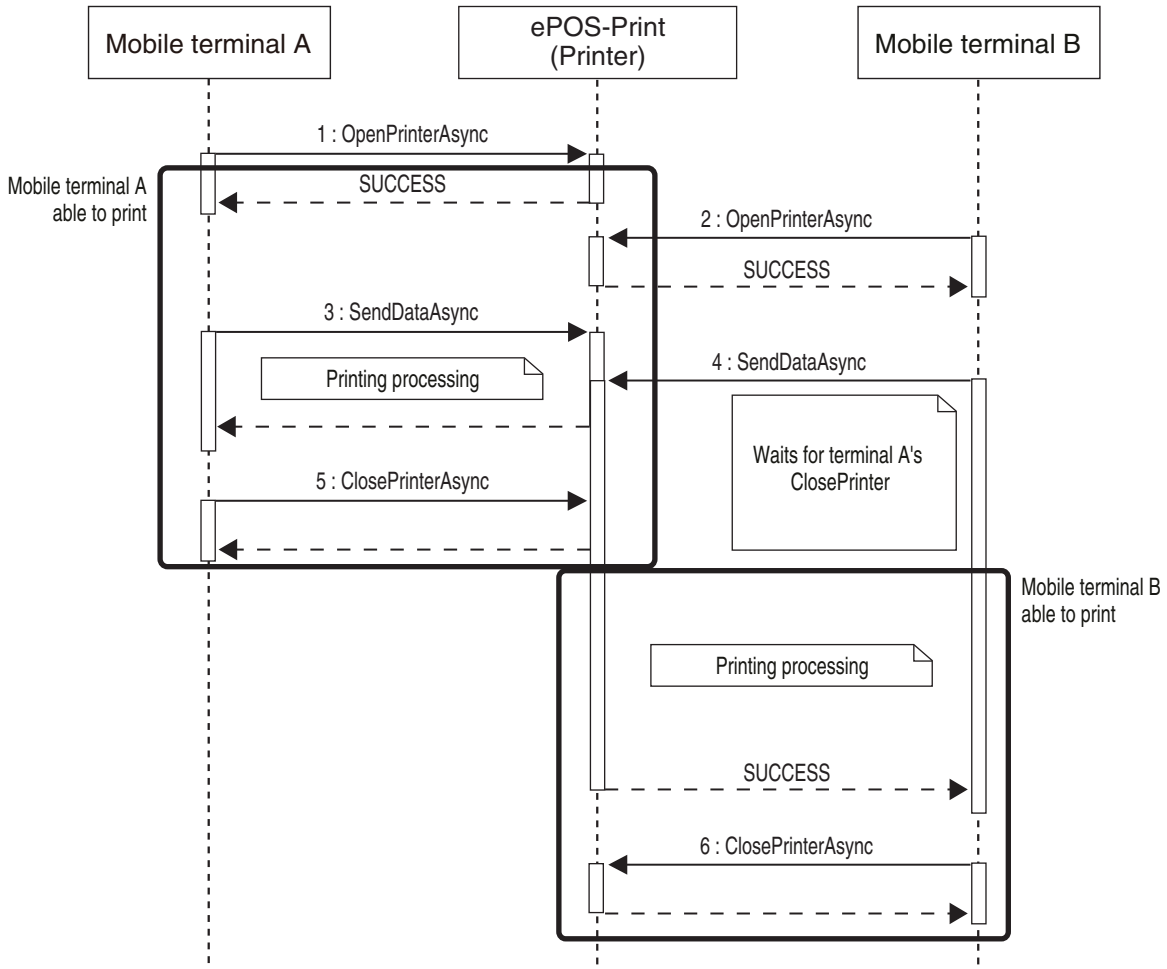
Cautions

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 OpenPrinterAsync processing has been initiated on one terminal when the printer is being used by another terminal, the OpenPrinterAsync 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.

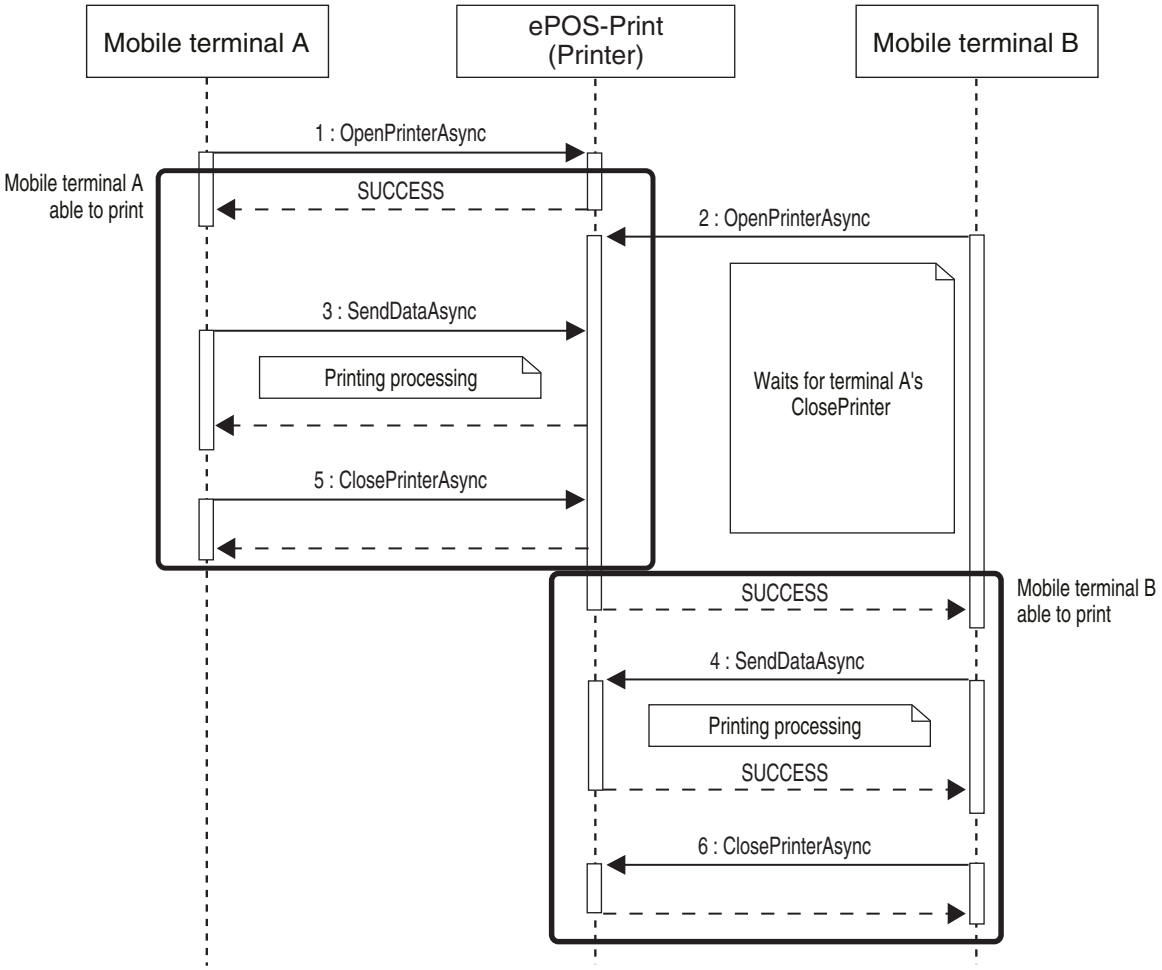
Version 1.5.0 and earlier

With Version 1.5.0 and earlier, mobile terminal B will wait for mobile terminal A's ClosePrinterAsync processing to end before executing SendDataAsync processing.



Version 1.6.0 and later

With Version 1.6.0 and later, mobile terminal B will wait for mobile terminal A's ClosePrinterAsync processing to end before executing OpenPrinterAsync 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 BeginTransactionAsync (p.144) and EndTransactionAsync (p.146).