



## SII POS for .NET Service Object Application Programmer's Guide

Rev.07

[Products]

MP-B30 Series

Seiko Instruments Inc.

Rev.01	January 2019
Rev.02	November 2019
Rev.03	September 2020
Rev.04	March 2021
Rev.05	December 2021
Rev.06	October 2022
Rev.07	July 2023


Copyright© 2019-2023 by Seiko Instruments Inc.  
All rights reserved.

Microsoft® and Windows® are registered trademarks of Microsoft Corporation in the U.S., Japan, and other countries.

Bluetooth® is a registered trademark of Bluetooth SIG, Inc.

Seiko Instruments Inc. (hereinafter referred to as "SII") has prepared this manual for use by SII personnel, licensees, and customers. The information contained herein is the property of SII and shall not be reproduced in whole or in part without the prior written approval of SII.

SII reserves the right to make changes without notice to the specifications and materials contained herein and shall not be responsible for any damages (including consequential) caused by reliance on the materials presented, including but not limited to typographical, arithmetic, or listing errors.

**SII**  is a trademark of Seiko Instruments Inc.

---

# Introduction

---

This manual describes "SII POS for .NET Service Object" (hereinafter referred to as the "software") provided by Seiko Instruments Inc. (hereinafter referred to as "SII").

## Target Products

---

The products supported by this manual are listed below.

	Device Name	Description in This Manual
PosPrinter	MP-B30 POS Printer	Printer

Also see "UnifiedPOS Retail Peripheral Architecture Version 1.12" (hereinafter "UPOS V1.12") and "Microsoft Point of Service for .NET - POS for .NET v1.12 SDK Documentation" when using this software.

## Notation in This Manual

---

The notation in this manual is described.

## Operation and Display

In principle, this manual is written on the basis of the following conditions:

- Screenshots and display layouts of Windows 10
- Operating instructions with a mouse and a keyboard

## Operating System Abbreviations

The operating system abbreviations used in this manual are listed below.

Operating System	Abbreviation
General Microsoft® Windows®	Windows
Microsoft® Windows® 11	Windows 11
Microsoft® Windows® 11 IoT Enterprise	
Microsoft® Windows® 10	Windows 10
Microsoft® Windows® 10 IoT Enterprise	

## Terms

The terms used in this manual are defined as below.

Term	Description
Configuration program	The program that executes addition and setting change of devices for PosPrinter provided by this software. When installing this software, it will be installed as [SII POS for .NET Utility] on the computer.
Default	The value immediately after satisfying the availability condition.
Line spacing	The height of each print line (total value of the printed line height and the whitespace between each pair of lines).
Printer command	Command for controlling the printer described in "MP-B30 SERIES THERMAL PRINTER TECHNICAL REFERENCE".

## Symbols

The symbols used in this manual are described below.

### Caution

- ◆ Notes and limitations are described.

### Reference

- Supplemental information and related matters are described.

---

# Table of Contents

---

## Chapter 1 Overview 1-1

---

1.1	Configuration .....	1-1
1.1.1	Structural Diagram.....	1-1
1.2	Operating Environment .....	1-2
1.2.1	System Environment.....	1-2
1.3	Printer Settings .....	1-2
1.4	Limitations .....	1-3
1.4.1	General .....	1-3
1.4.2	PosPrinter .....	1-3

## Chapter 2 Installation 2-1

---

2.1	Installation .....	2-1
2.2	Uninstallation .....	2-4

## Chapter 3 How to Operate Configuration Program 3-1

---

3.1	Startup.....	3-1
3.2	Screen Layout.....	3-2
3.2.1	Menu Bar .....	3-3
3.2.2	Tool Bar .....	3-3
3.2.3	Device View .....	3-3
3.2.4	Setting View .....	3-4
3.3	Functions.....	3-6
3.3.1	Addition of Device.....	3-6
3.3.2	Changing Device Settings.....	3-10
3.3.3	Deletion of Device .....	3-11
3.3.4	Device Interactive Test .....	3-11
3.3.5	Log Setting .....	3-13

4.1 PosPrinter .....	4-1
4.1.1 Summary.....	4-1
4.1.2 Data Characters and Escape Sequences.....	4-8
4.1.3 Common Properties.....	4-12
CapCompareFirmwareVersion Property.....	4-12
CapPowerReporting Property .....	4-12
CapStatisticsReporting Property .....	4-12
CapUpdateFirmware Property .....	4-13
CapUpdateStatistics Property .....	4-13
CheckHealthText Property.....	4-13
Claimed Property.....	4-14
DeviceDescription Property .....	4-14
DeviceEnabled Property R/W .....	4-14
DeviceName Property.....	4-15
FreezeEvents Property R/W .....	4-15
OutputId Property .....	4-15
PowerNotify Property R/W .....	4-16
PowerState Property.....	4-16
ServiceObjectDescription Property .....	4-17
ServiceObjectVersion Property .....	4-17
State Property .....	4-17
SynchronizingObject Property.....	4-18
4.1.4 Specific Properties.....	4-19
AsyncMode Property R/W.....	4-19
CapCharacterSet Property.....	4-19
CapCoverSensor Property.....	4-19
CapMapCharacterSet Property .....	4-20
CapRec2Color Property.....	4-20
CapRecBarCode Property .....	4-20
CapRecBitmap Property .....	4-20
CapRecBold Property .....	4-21
CapRecCartridgeSensor Property.....	4-21
CapRecColor Property.....	4-21
CapRecDHigh Property .....	4-21
CapRecDWide Property.....	4-22
CapRecDWideDHigh Property.....	4-22
CapRecEmptySensor Property .....	4-22
CapRecItalic Property.....	4-22
CapRecLeft90 Property .....	4-23
CapRecMarkFeed Property .....	4-23
CapRecNearEndSensor Property .....	4-23
CapRecPageMode Property .....	4-24
CapRecPaperCut Property .....	4-24
CapRecPresent Property .....	4-24
CapRecRight90 Property .....	4-24
CapRecRotate180 Property.....	4-25

	CapRecStamp Property.....	4-25
	CapRecUnderline Property.....	4-25
	CapTransaction Property.....	4-25
	CartridgeNotify Property R/W.....	4-26
	CharacterSet Property R/W .....	4-26
	CharacterSetList Property.....	4-27
	CoverOpen Property.....	4-27
	ErrorLevel Property .....	4-27
	ErrorStation Property.....	4-28
	ErrorString Property.....	4-28
	FlagWhenIdle Property R/W .....	4-29
	FontTypefaceList Property.....	4-29
	MapCharacterSet Property R/W.....	4-29
	MapMode Property R/W .....	4-30
	PageModeArea Property .....	4-31
	PageModeDescriptor Property.....	4-31
	PageModeHorizontalPosition Property R/W .....	4-32
	PageModePrintArea Property R/W .....	4-33
	PageModePrintDirection Property R/W .....	4-33
	PageModeStation Property R/W .....	4-36
	PageModeVerticalPosition Property R/W .....	4-36
	RecBarCodeRotationList Property .....	4-37
	RecBitmapRotationList Property .....	4-37
	RecCartridgeState Property.....	4-37
	RecCurrentCartridge Property R/W.....	4-38
	RecEmpty Property .....	4-38
	RecLetterQuality Property R/W .....	4-38
	RecLineChars Property R/W .....	4-39
	RecLineCharsList Property .....	4-39
	RecLineHeight Property R/W .....	4-40
	RecLineSpacing Property R/W .....	4-40
	RecLinesToPaperCut Property .....	4-41
	RecLineWidth Property.....	4-41
	RecNearEnd Property.....	4-41
	RecSidewaysMaxChars Property.....	4-42
	RecSidewaysMaxLines Property.....	4-42
	RotateSpecial Property R/W .....	4-43
4.1.5	Common Methods .....	4-44
	CheckHealth Method .....	4-44
	Claim Method .....	4-44
	ClearOutput Method .....	4-45
	Close Method .....	4-45
	CompareFirmwareVersion Method .....	4-45
	DirectIO Method .....	4-45
	Open Method.....	4-47
	Release Method .....	4-47
	ResetStatistic(string) Method .....	4-47
	ResetStatistics() Method.....	4-47
	ResetStatistics(StatisticCategories) Method.....	4-48

	ResetStatistics(string[]) Method .....	4-48
	RetrieveStatistic(string) Method .....	4-48
	RetrieveStatistics() Method .....	4-48
	RetrieveStatistics(StatisticCategories) Method .....	4-48
	RetrieveStatistics(string[]) Method .....	4-49
	UpdateFirmware Method .....	4-49
	UpdateStatistic Method.....	4-49
	UpdateStatistics(Statistic[]) Method.....	4-49
	UpdateStatistics(StatisticCategories, Object) Method.....	4-49
4.1.6	Specific Methods .....	4-50
	ClearPrintArea Method .....	4-50
	MarkFeed Method .....	4-50
	PageModePrint Method .....	4-51
	PrintBarCode Method .....	4-53
	PrintBitmap Method .....	4-65
	PrintImmediate Method.....	4-66
	PrintMemoryBitmap Method.....	4-67
	PrintNormal Method.....	4-67
	RotatePrint Method.....	4-68
	SetBitmap Method .....	4-69
	SetLogo Method .....	4-70
	TransactionPrint Method.....	4-70
	ValidateData Method .....	4-71
4.1.7	Events .....	4-72
	DirectIOEvent Event .....	4-72
	ErrorEvent Event .....	4-72
	OutputCompleteEvent Event.....	4-72
	StatusUpdateEvent Event.....	4-73

## Appendix A Exceptions

**A-1**

A.1	PosPrinter Exception Error List.....	A-1
-----	--------------------------------------	-----

## Appendix B Statistics

**B-1**



---

# Chapter 1 Overview

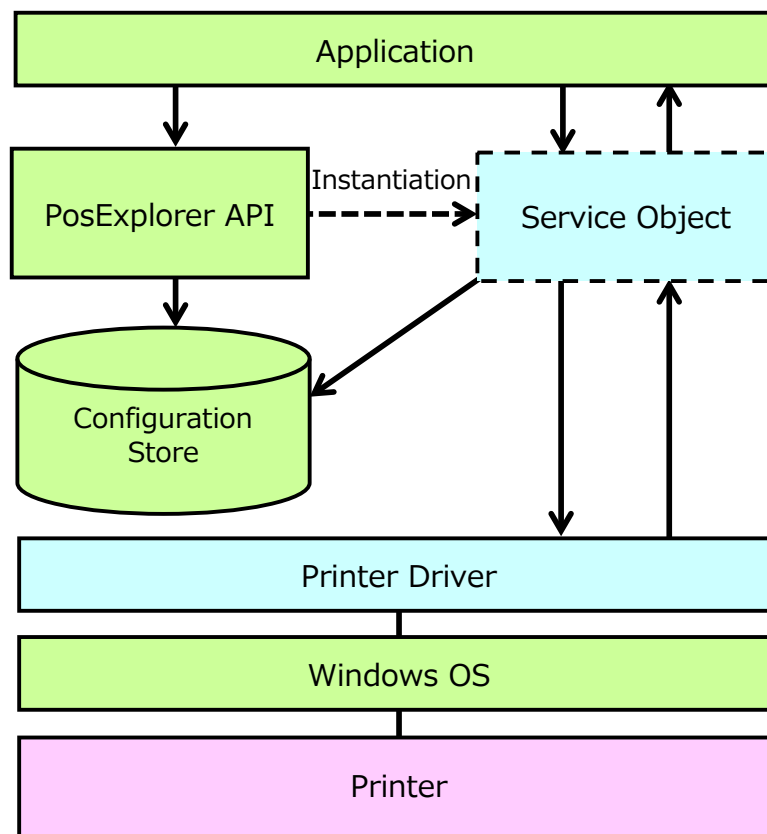
---

This chapter describes the overview of this software.

## 1.1 Configuration

### 1.1.1 Structural Diagram

The structure of the software is as follows, and the scope of this manual is indicated by dashed lines.



## 1.2 Operating Environment

### 1.2.1 System Environment

The system environment of this software is shown below.

Item	Specifications
Operating System	Microsoft® Windows® 11 (64 bits) Microsoft® Windows® 11 IoT Enterprise (64 bits) Microsoft® Windows® 10 (32 bits and 64 bits) Microsoft® Windows® 10 IoT Enterprise (32 bits and 64 bits)
.NET Framework* <sup>1</sup>	.NET Framework 3.5 or .NET Framework 4.0
Microsoft POS for .NET SDK* <sup>1</sup>	POS for .NET 1.12 or POS for .NET 1.14
Printer driver	"SII Printer Driver for Windows" for MP-B30 series

\*1: Before installing this software, it is required to install .NET Framework 3.5 and Microsoft POS for .NET 1.12 or .NET Framework 4.0 and Microsoft POS for .NET 1.14 in advance.

## 1.3 Printer Settings

The memory switches of the printer are set to [Value] in the following table forcibly by executing **Claim** when using the software.

See "MP-B30 SERIES Thermal Printer USER'S GUIDE" for details about the memory switches.

MS	Function	Value	Note
1-2	Mark Mode Selection (Mark Mode)	0 : Enable 1 : Disable	Either one of [Value] on the left can be set by [Mark Mode] in the configuration program.
1-6	Data Discard Selection When Error Occurs (Error Through)	1 : Disable	-
1-8	Data Discard Selection When Output Buffer Full Occurs (Response Data Discarding)	1 : Disable	
2-2	Realtime Command Selection (Realtime Command)	0 : Enable	
2-3 to 2-4	Print Quality Selection (Print Quality)	01B : Quality 2 10B : Quality 1 11B : Standard	Any one of [Value] on the left can be set by [Print Speed] in the configuration program.
9-1	Automatic Status Response Selection (Auto Status Back)	0 : Enable	-
9-2	Initialized Response Selection (Init. Response)	0 : Enable	

## 1.4 Limitations

The limitations of this software are described.

### 1.4.1 General

When using 1 printer simultaneously from multiple computers via TCP/IP connections, use **TransactionPrint** to prevent print data from other computers from interrupting.

### 1.4.2 PosPrinter

This software is based on UnifiedPOS Specification Version 1.12, and all interfaces of PosPrinter device are provided with the following limitations.

- (a) The method and property settings related to journal and slip prints are not supported.
- (b) The following functions are not supported.
  - Paper cut
  - Feed and Paper cut
  - Feed, Paper cut and Stamp
  - Stamp
  - Feed reverse
  - Font typeface selection
  - Italic
  - Alternate color (Custom)
  - Red color
  - Shading
  - RBG Color
  - Sub Script
  - Super Script
  - Strike-through
- (c) All the following methods always return *ErrorCode.Illegal* after they are enabled.
  - **BeginInsertion**
  - **BeginRemoval**
  - **ChangePrintSide**
  - **CutPaper**
  - **EndInsertion**
  - **EndRemoval**
  - **PrintTwoNormal**
- (d) **DirectIOEvent** (device-specific event) is not supported.
- (e) When [Process Completion Timing] is set as "Data printing" in the configuration program, the printer command "Execution Response Request" is used inside this software for controlling the print operation. Therefore, an unexpected behavior may occur when sending "Execution Response Request" by the "Pass through embedded data" escape sequence (ESC|#E).
- (f) When an error occurs, the printer command "Hardware Reset" is sent from PosPrinter to cancel printing in the printer; however, printing may be performed a bit before PosPrinter stops printing in the printer.
- (g) For Bluetooth connection, when the device becomes a recoverable error during printing, it may take some time to return to normal status from the error cancellation. The first processing after error cancelation should be done after about 10 seconds from the error cancellation.

---

## Chapter 2 Installation

---

This chapter describes installation/uninstallation of the software.

It is necessary to install the printer driver before installing this software.

For the installation procedure of the printer driver, see the installation part of "SII Printer Driver for Windows User's Guide" for MP-B30 series.

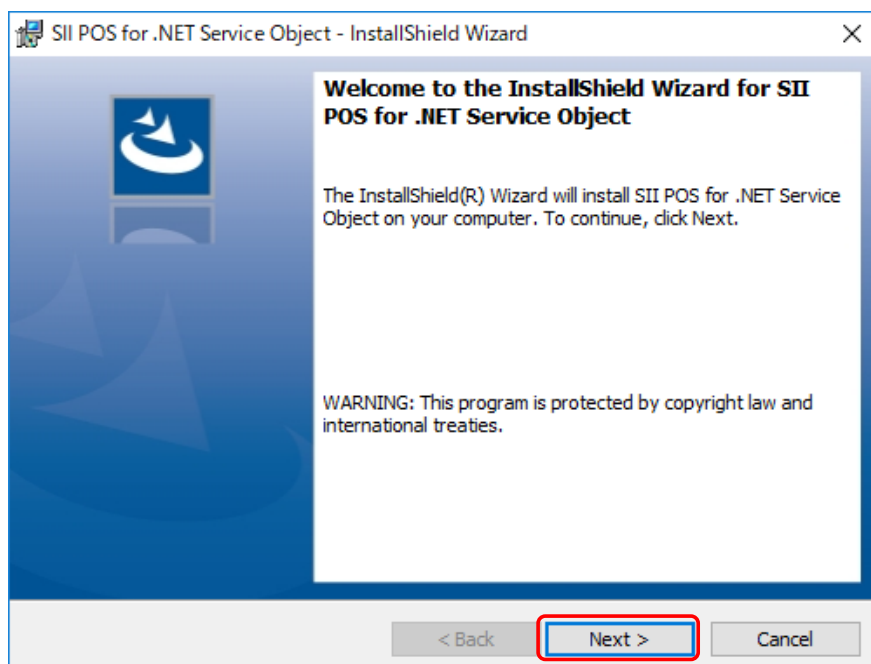
### Caution

- ◆ This installation requires logon to the computer with administrator privileges.

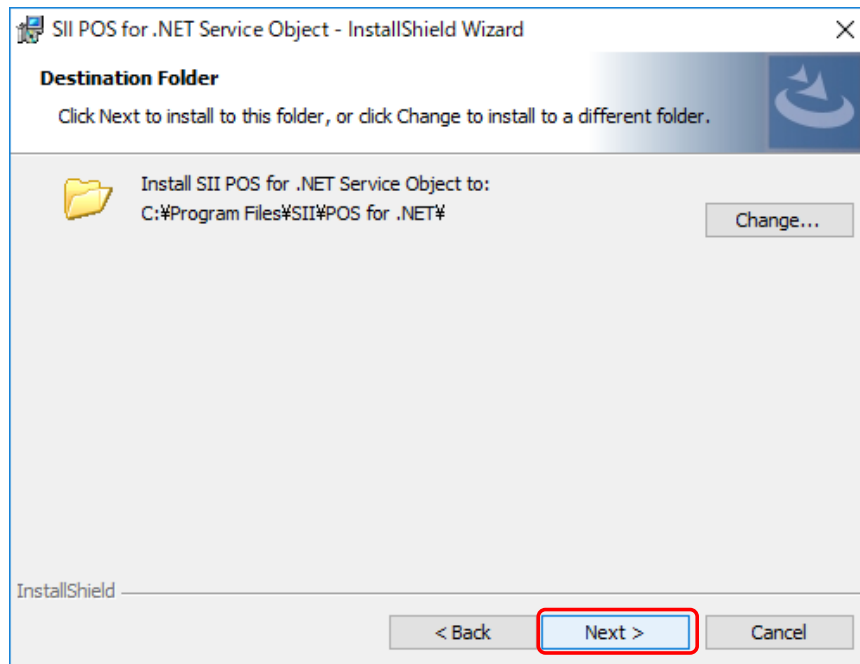
## 2.1 Installation

The installation procedure of this software is described below.

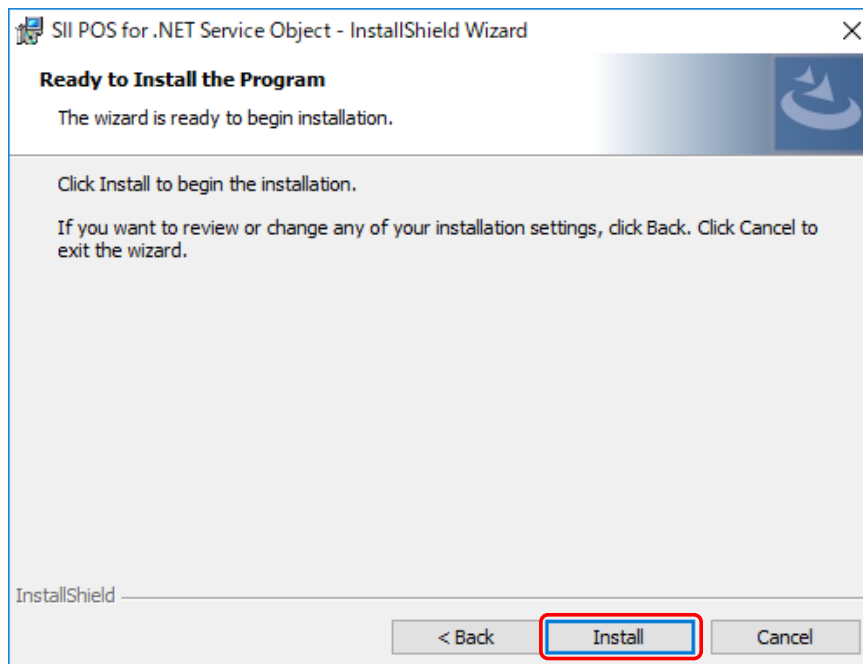
- (1) Start the setup program.  
For 32-bit OS : SetupPosNet.exe  
For 64-bit OS : SetupPosNet64.exe
- (2) When the installer starts up, click the [Next >] button.



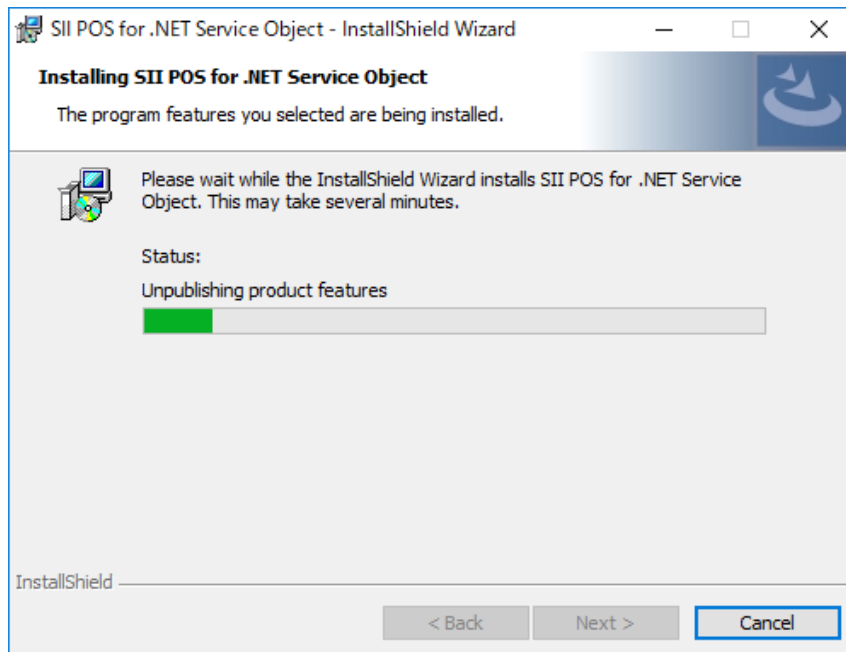
(3) Specify the destination folder and click the [Next >] button.



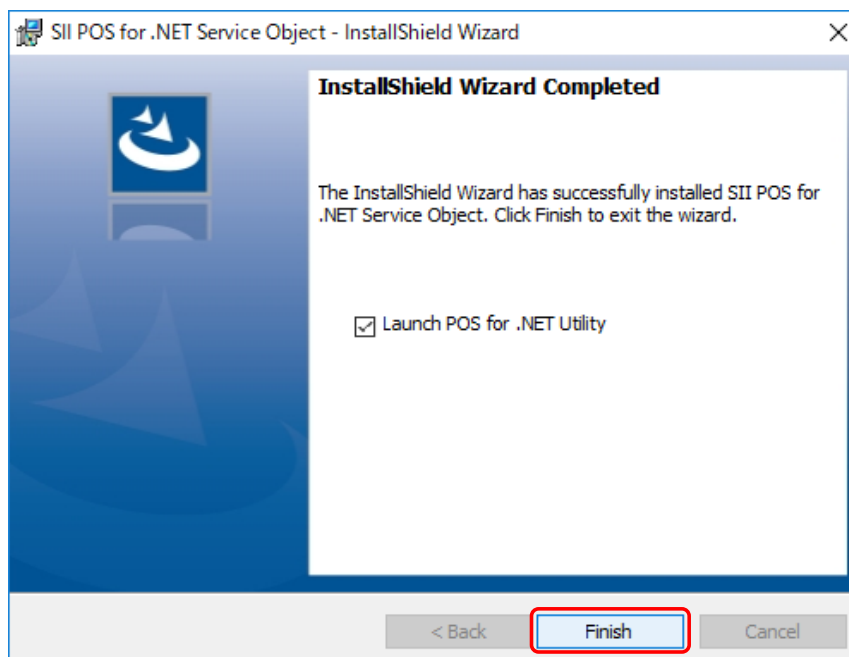
(4) Click the [Install] button.



- (5) The installation starts, and the progress status bar is displayed.



- (6) Click the [Finish] button. When clicking the [Finish] button with the "Launch POS for .NET Utility" checkbox on, the setup program ends and the configuration program (POS for .NET Utility) starts up.



## **2.2 Uninstallation**

When the software is no longer used, click "Uninstall a program" in [Programs and Features] in the Control Panel. When the [Uninstall or change a program] window is displayed, select SII POS for .NET Service Object, and click the [Uninstall] button.

---

## Chapter 3 How to Operate Configuration Program

---

This chapter describes the configuration program provided by this software.

### 3.1 Startup

The startup procedure of the configuration program is described.

- For Windows 11:  
Select [All apps] - [POS for .NET Utility] from the Start menu, and then the configuration program starts up.
- For Windows 10:  
Select [SII POS for .NET] - [POS for .NET Utility] from the Start menu, and then the configuration program starts up.

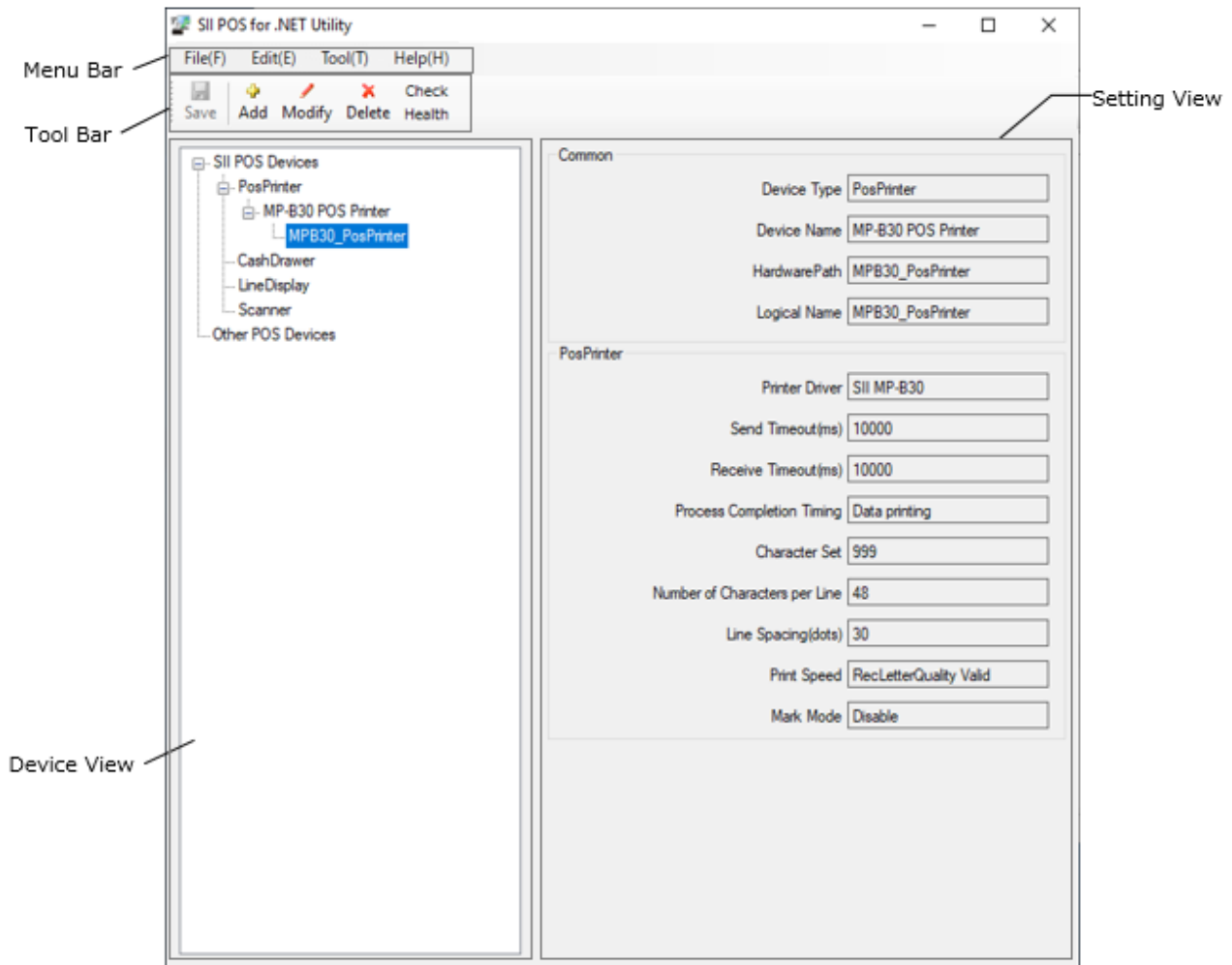
### Caution

- ◆ This software operates using a printer driver. The printer driver is required to be installed on the computer for using this software.
- ◆ Using this software requires logon to the computer with administrator privileges.



## 3.2 Screen Layout

The screen layout of the configuration program is described.



Item	Description
Menu Bar	The menu bar of the configuration program. See "3.2.1 Menu Bar" for items in the menu bar.
Tool Bar	The tool bar of the configuration program. See "3.2.2 Tool Bar" for items in the tool bar.
Device View	The type, the name, and the logical name of the device registered in the system are displayed in a tree.
Setting View	Displays setting contents of the device selected in "Device View". See "3.2.4(1) PosPrinter setting items" for items of each device.

### 3.2.1 Menu Bar

Item		Description
File(F)	Save(S)	Saves the data being edited in configuration.xml.
	Restore(R)	Discards the data being edited and re-reads the data saved in configuration.xml.
	End(E)	Ends the configuration program.
Edit(E)	Add(A)	Adds a new device.
	Modify(M)	Changes the setting contents for the device being selected.
	Delete(D)	Deletes the device being selected.
Tool(T)	CheckHealth	Executes an interactive test on the device being selected.
	LogSetting	Performs common log settings for all devices. See "3.3.5 Log Setting" for details of log settings.
Help(H)	About SII POS for .NET Utility(A)	Displays the version information of the configuration program.
	Language Mode(L)	Japanese(J) Displays the configuration program in Japanese. English(E) Displays the configuration program in English.

### 3.2.2 Tool Bar

Item	Description
Save	Performs the same process as Menu Bar - [File(F)] - [Save(S)].
Add	Performs the same process as Menu Bar - [Edit(E)] - [Add(A)].
Modify	Performs the same process as Menu Bar - [Edit(E)] - [Modify(M)].
Delete	Performs the same process as Menu Bar - [Edit(E)] - [Delete(D)].
CheckHealth	Performs the same process as Menu Bar - [Tool(T)] - [CheckHealth].

### 3.2.3 Device View

Name	Description
SII POS Devices	Displays SII devices. When the logical name is selected in "Device View", setting contents of the device can be changed or deleted.
Other POS Devices	Displays devices other than SII devices. Device settings cannot be changed or deleted.

### 3.2.4 Setting View

#### (1) PosPrinter setting items

The items displayed in "Setting View" when MP-B30 POS Printer is selected and setting contents are described below.

The screenshot shows the 'SII POS for .NET Utility' application window. On the left is a tree view under 'SII POS Devices' with 'MPB30\_PosPrinter' selected. The main area is divided into two sections: 'Common' and 'PosPrinter'. The 'Common' section contains fields for Device Type (PosPrinter), Device Name (MP-B30 POS Printer), HardwarePath (MPB30\_PosPrinter), and Logical Name (MPB30\_PosPrinter). The 'PosPrinter' section contains fields for Printer Driver (SII MP-B30), Send Timeout(ms) (10000), Receive Timeout(ms) (10000), Process Completion Timing (Data printing), Character Set (999), Number of Characters per Line (48), Line Spacing(dots) (30), Print Speed (RecLetterQuality Valid), and Mark Mode (Disable).

Item	Description	Setting Content (" " : Default)
Common		
Device Type	Device type	[PosPrinter]
Device Name	Device name	MP-B30 POS Printer
HardwarePath	Set automatically. It cannot be changed.	-
Logical Name	Any logical name entered	-

Item	Description	Setting Content (" " : Default)
PosPrinter		
Printer Driver	Printer driver used for communication with the printer	-
Send Timeout(ms)	Send timeout value in communication with the printer (milliseconds)	3000 to 60000 (10000)
Receive Timeout(ms)	Receive timeout value in communication with the printer (milliseconds)	3000 to 60000 (10000)
Process Completion Timing	Timing of method completion	Data transmission Data printing
Character Set	Character set type <b>CharacterSet</b> is initialized with this value. See <b>CharacterSet</b> for details.	437 737 850 852 855 857 858 860 863 865 866 932*1 999*2 1250 1251 1252 1253 1254
Number of Characters per Line	Number of 1-byte characters per line <b>RecLineChars</b> is initialized with this value.	36,41,44,48,57,64,72
Line Spacing(dots)	Line spacing per line (dot)  Settable range: The settable minimum value differs depending on the selected values of effective dots and characters per line. <b>RecLineSpacing</b> is initialized with this value.	Settable range: 24 to 255 (30)
		Number of Characters per line: 36,41,44,48
		Settable range: 16 to 255 Number of Characters per line: 57,64,72
Print Speed	Print speed of the printer It is decided by <b>RecLetterQuality</b> when selecting <b>RecLetterQuality Valid</b> .	<b>RecLetterQuality Valid</b> Standard*3 Quality 1*3 Quality 2*3
Mark Mode	Mark detection mode Selects whether to enable or disable mark detection.	<b>Disable</b> : Mark detection disable <b>Enable</b> : Mark detection enable

\*1: Default for Japanese

\*2: Default for English

\*3: See "MP-B30 SERIES Thermal Printer USER'S GUIDE" for details of the print speed.

### 3.3 Functions

The functions of the configuration program are described.

#### 3.3.1 Addition of Device

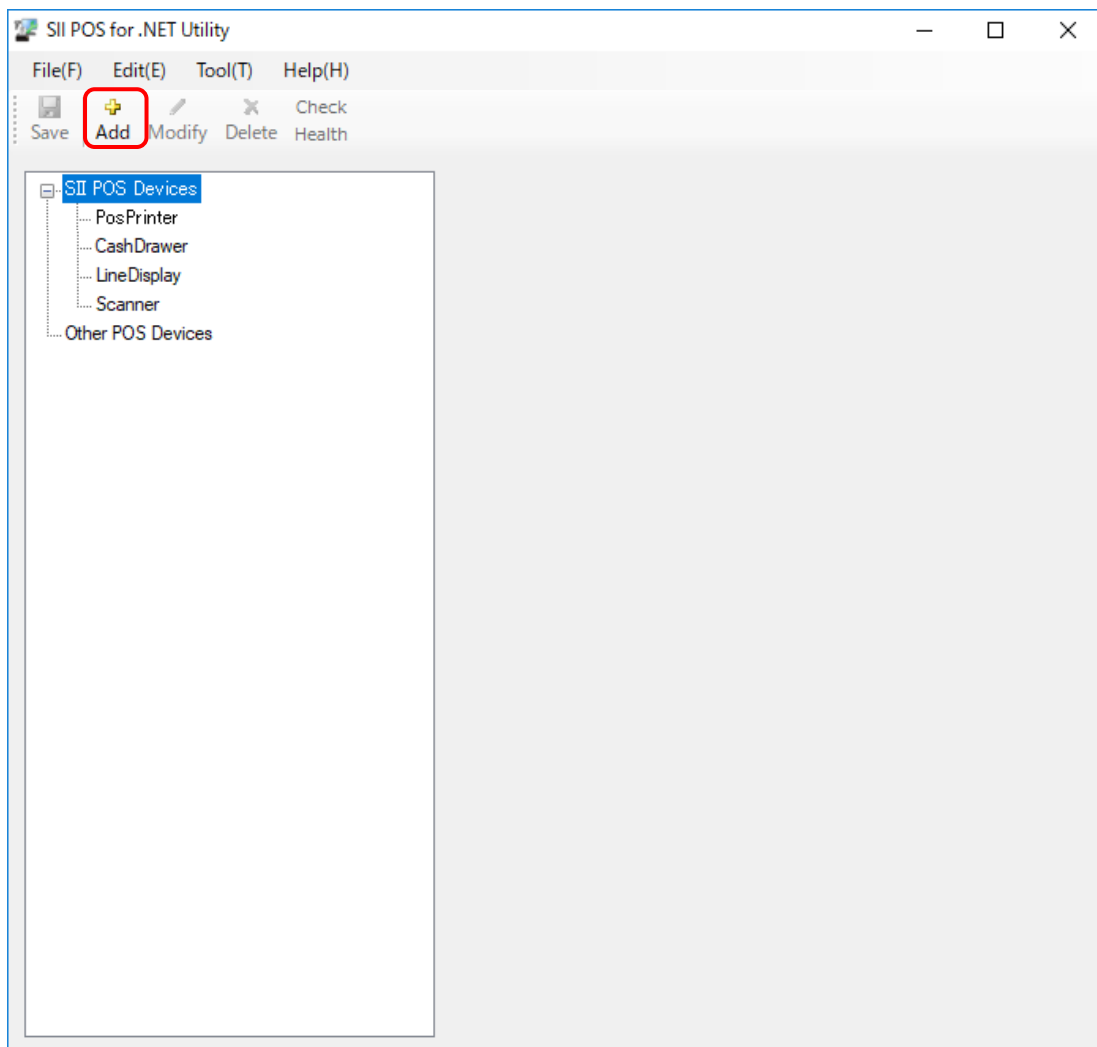
The procedure for adding a device is described.

When the configuration program is started up immediately after installing this software, a device needs to be added since no device has been added.

When adding a new PosPrinter, it is necessary to install the printer driver for the communication port to be used in advance. See "SII Printer Driver for Windows User's Guide" for MP-B30 series for installation of the printer driver.

##### (1) Addition of PosPrinter

(a) When the configuration program starts, the following window is displayed. Click the [Add] button.



- (b) Select "PosPrinter" for [Device Type] and "MP-B30 POS Printer" for [Device Name], then click the [Next] button.

The screenshot shows a dialog box titled "Add PosDevice" with a close button (X) in the top right corner. The main area is labeled "Select Device". It contains two dropdown menus: "Device Type" with "PosPrinter" selected, and "Device Name" with "MP-B30 POS Printer" selected. At the bottom right, there are two buttons: "Next" and "Cancel". The "Next" button is highlighted with a red rectangular border.

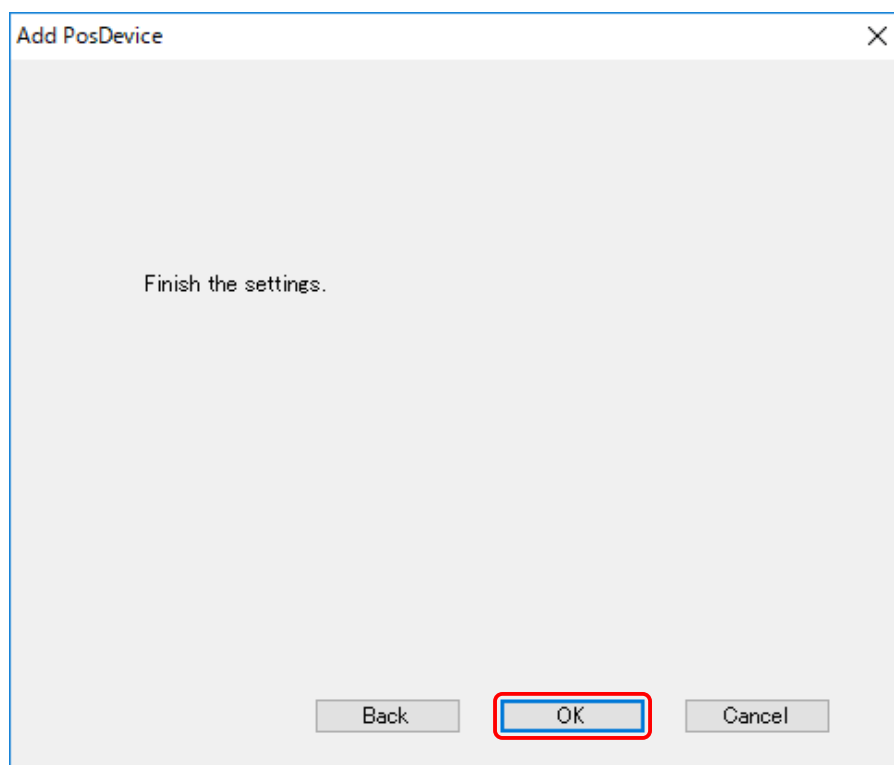
- (c) Enter or select settings of the printer, and then click the [Next] button.

The screenshot shows the same "Add PosDevice" dialog box, but now it is titled "Set PosPrinter properties". It contains several configuration fields and dropdown menus: "Logical Name" (text box with "MPB30\_PosPrinter"), "Printer Driver" (dropdown with "SII MP-B30"), "Send Timeout(ms)" (text box with "10000"), "Receive Timeout(ms)" (text box with "10000"), "Process Completion Timing" (dropdown with "Data printing"), "Character Set" (dropdown with "999"), "Number of Characters per Line" (dropdown with "48"), "Line Spacing(dots)" (text box with "30"), "Print Speed" (dropdown with "RecLetterQuality Valid"), and "Mark Mode" (dropdown with "Disable"). At the bottom, there are three buttons: "Back", "Next", and "Cancel". The "Next" button is highlighted with a red rectangular border.

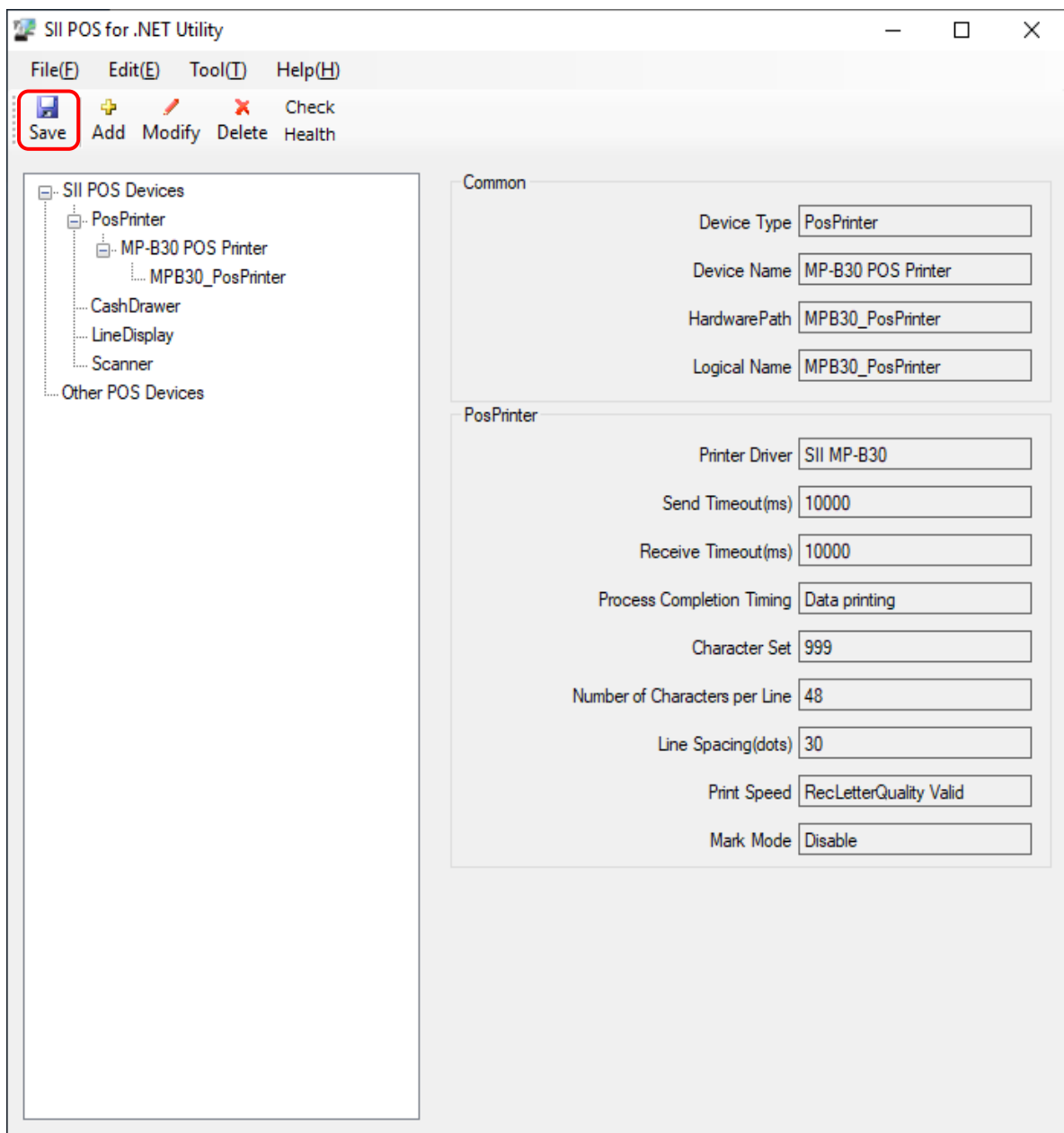
## Caution

- ◆ The same logical name cannot be set for multiple service objects.

(d) Click the [OK] button.



(e) Confirm the contents in "Setting View", and click the [Save] button.

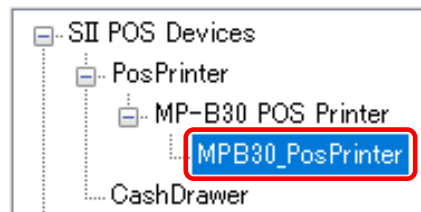




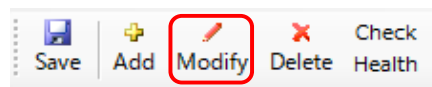
### 3.3.2 Changing Device Settings

The settings of the added device can be changed with the [Modify] button.

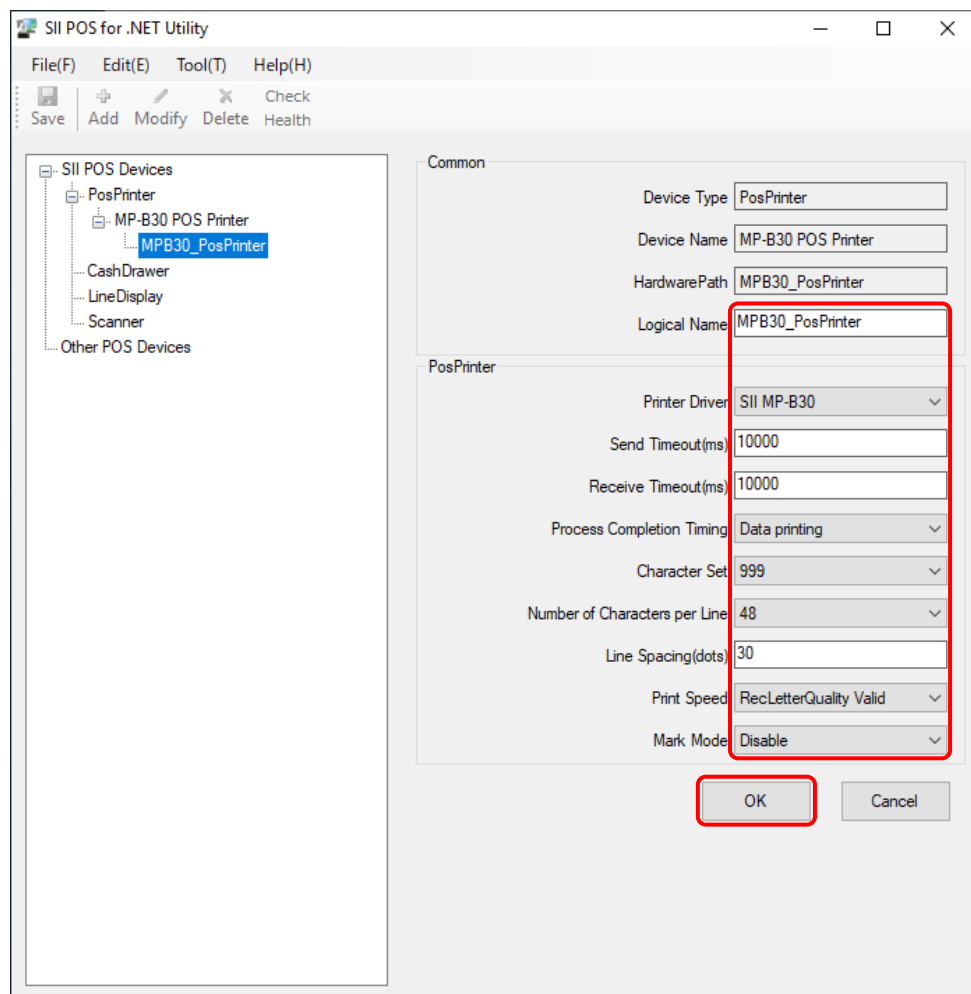
- (a) Select the logical name of PosPrinter to change from "Device View".



- (b) Click the [Modify] button in "Tool Bar".



- (c) "Setting View" is displayed in editable state. Click the [OK] button after changing the contents.



- (d) Click the [Save] button in "Tool Bar".

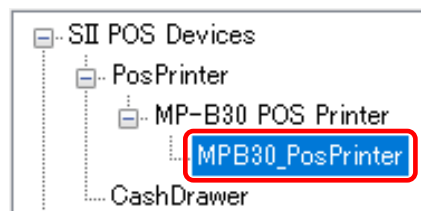
### 3.3.3 Deletion of Device

The added device can be deleted with the [Delete] button.  
Select the target logical name, and click the [Delete] button.

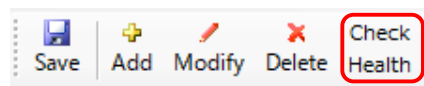
### 3.3.4 Device Interactive Test

In the configuration program, an interactive test can be performed on the device selected in "Device View".  
The procedure of the interactive test is described below.

- (a) Select the logical name of PosPrinter for interactive test from "Device View".



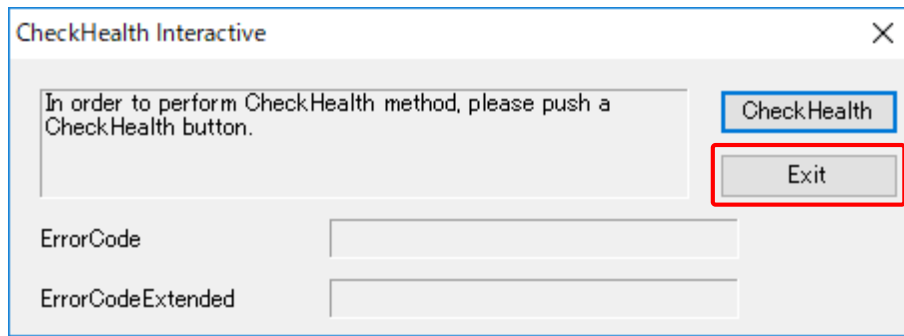
- (b) Click the [CheckHealth] button in "Tool Bar".



- (c) The preparation for the interactive test is started.

[When the preparation for the interactive test succeeded]

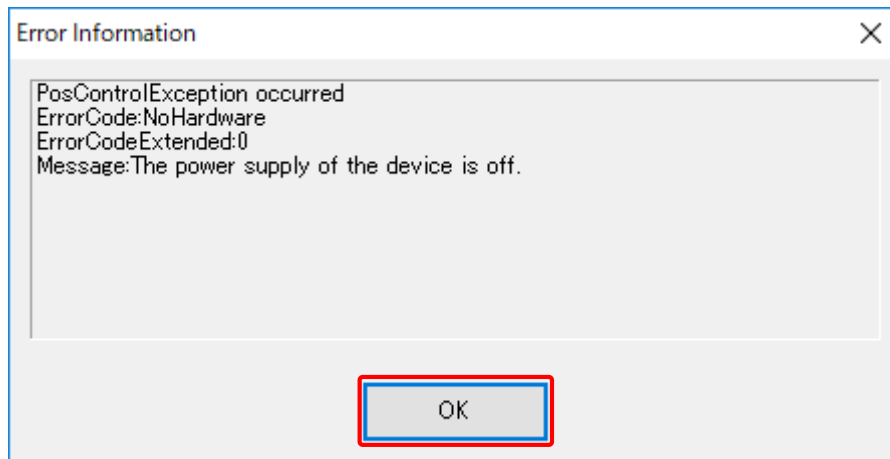
- (d) The CheckHealth Interactive dialogue to perform the interactive test is displayed.



To start the interactive test, click the [CheckHealth] button.  
To exit the interactive test, click the [Exit] button.

[When the preparation for the interactive test failed]

- (d) The Error Information dialogue is displayed.

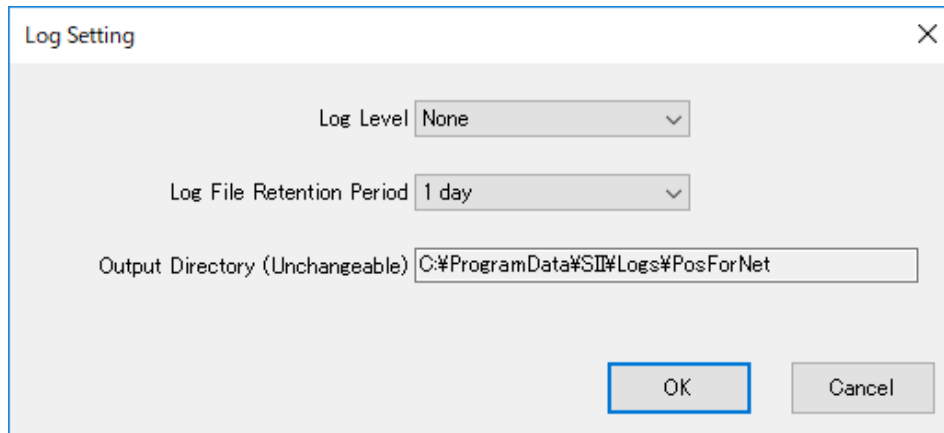


Confirm ErrorCode displayed in the dialogue. See "Appendix A Exceptions" for ErrorCode.  
Click the [OK] button after confirming ErrorCode.

### 3.3.5 Log Setting

In the configuration program, common log settings can be made for all devices.

Select [Tool] – [LogSetting] from "Menu Bar" to display the following window.



The image shows a 'Log Setting' dialog box with a title bar containing a close button (X). Inside the dialog, there are three settings:

- Log Level:** A dropdown menu currently set to 'None'.
- Log File Retention Period:** A dropdown menu currently set to '1 day'.
- Output Directory (Unchangeable):** A text field containing the path 'C:\ProgramData\SII\Logs\PosForNet'.

At the bottom right of the dialog are two buttons: 'OK' and 'Cancel'.

The level of the log and the contents to be output are as follows.

Item	Description (" " : Default)	
Log Level	None	No logs are output.
	Error	The following logs are output. • Error at execution
	Info	The following logs are output. • Error at execution • Highlighted event at execution
	Debug	The following logs are output. • Error at execution • Highlighted event at execution • More detailed information for debugging
Log File Retention Period	Select the retention period for log files. • 1 day • 3 days • 10 days • 30 days • 90 days  Log files past the retention period are deleted when logs are output. The actual retention period may be longer by 1 day at maximum. The maximum size of a log file is 32 MB. When the log file exceeds the maximum size, a new log file is created and stored up to the retention period.	
Output Directory (Unchangeable)	Log output directory. The log output directory and file name are as follows. Output Directory: <System Drive>\ProgramData\SII\Logs\PosForNet The output directory cannot be changed. File Name: <yyyyMMdd>.log However, when the log file exceeds the maximum size, the file name is changed to <yyyyMMdd_hhmmssfff>.log, and a new <yyyyMMdd>.log is created.*1	

\*1: Meanings of the symbols used for the file name are as follows. Each value comes from System Clock of Windows.

yyyy : Year  
MM : Month  
dd : Day  
hh : Hour  
mm : Minute  
ss : Second  
fff : Millisecond

(1) Log setting procedure

The log setting procedure is described below.

- (a) Select [Tool] – [LogSetting] from "Menu Bar".
- (b) Select the log level to output from [Log Level].
- (c) Select the log file retention period from [Log File Retention Period], and click the [OK] button.
- (d) Click the [Save] button in the main screen. The log settings will be applied from the next **Open**.

---

## Chapter 4 Properties, Methods, and Events

---

This chapter describes properties, methods, and events implemented in this software.

### 4.1 PosPrinter

#### 4.1.1 Summary

(1) Common Properties

Property Name	Type	Access	Availability Condition	Default
CapCompareFirmwareVersion	bool	R	Open	<i>false</i>
CapPowerReporting	PowerReporting	R	Open	<i>Standard</i>
CapStatisticsReporting	bool	R	Open	<i>true</i>
CapUpdateFirmware	bool	R	Open	<i>false</i>
CapUpdateStatistics	bool	R	Open	<i>true</i>
CheckHealthText	string	R	Open	<i>""</i>
Claimed	bool	R	Open	<i>false</i>
DeviceDescription	string	R	Open	"SII MP-B30 POS Printer"
DeviceEnabled	bool	R/W	Open & Claim	<i>false</i>
DeviceName	string	R	Open	"MP-B30 POS Printer"
FreezeEvents	bool	R/W	Open & Claim	<i>false</i>
OutputId	int	R	Open	0
PowerNotify	PowerNotification	R/W	Open	<i>Disabled</i>
PowerState	PowerState	R	Open	<i>Unknown</i>
ServiceObjectDescription	string	R	Open	"SII MP-B30 POS Printer Service Object, Copyright(C) 20xx Seiko Instruments Inc."
ServiceObjectVersion	Version	R	Open	1.12.x.x
State	ControlState	R	--	<i>Idle</i>
SynchronizingObject	System.ComponentModel.ISynchronizeInvoke	R/W	Open	Depends on the application.

## (2) Specific Properties

(When RecLineWidth=576, RecLineChars=48, RecLineSpacing=30, CharacterSet=999)

Property Name	Type	Access	Availability Condition	Default
AsyncMode	bool	R/W	Open	false
CapCharacterSet	CharacterSetCapability	R	Open	Kanji
CapCoverSensor	bool	R	Open	true
CapMapCharacterSet	bool	R	Open	false
CapRec2Color	bool	R	Open	false
CapRecBarCode	bool	R	Open	true
CapRecBitmap	bool	R	Open	true
CapRecBold	bool	R	Open	true
CapRecCartridgeSensor	PrinterCartridgeSensors	R	Open	None
CapRecColor	PrinterColors	R	Open	Primary
CapRecDHigh	bool	R	Open	true
CapRecDWide	bool	R	Open	true
CapRecDWideDHigh	bool	R	Open	true
CapRecEmptySensor	bool	R	Open	true
CapRecItalic	bool	R	Open	false
CapRecLeft90	bool	R	Open	true
CapRecMarkFeed	PrinterMarkFeeds	R	Open	None <sup>*1</sup>
CapRecNearEndSensor	bool	R	Open	false
CapRecPageMode	bool	R	Open	true
CapRecPaperCut	bool	R	Open	false
CapRecPresent	bool	R	Open	true
CapRecRight90	bool	R	Open	true
CapRecRotate180	bool	R	Open	true
CapRecStamp	bool	R	Open	false
CapRecUnderline	bool	R	Open	true
CapTransaction	bool	R	Open	true
CartridgeNotify	PrinterCartridgeNotify	R/W <sup>*2</sup>	Open	Disabled
CharacterSet	int	R/W	Open, Claim, & Enable	999 <sup>*1</sup>
CharacterSetList	int[]	R	Open	{437, 737, 850, 852, 855, 857, 858, 860, 863, 865, 866, 932, 999, 1250, 1251, 1252, 1253, 1254}
CoverOpen	bool	R	Open, Claim, & Enable	Depends on printer status.
ErrorLevel	PrinterErrorLevel	R	Open	None
ErrorStation	PrinterStation	R	Open	None
ErrorString	string	R	Open	""
FlagWhenIdle	bool	R/W	Open	false
FontTypefaceList	string[]	R	Open	[0]

Property Name	Type	Access	Availability Condition	Default
MapCharacterSet	bool	R/W <sup>*2</sup>	Open	<i>false</i>
MapMode	MapMode	R/W	Open	<i>Dots</i>
PageModeArea	System.Drawing.Point	R	Open	{0, 0}
PageModeDescriptor	PageModeDescriptors	R	Open	<i>None</i>
PageModeHorizontalPosition	int	R/W	Open	0
PageModePrintArea	System.Drawing.Rectangle	R/W	Open	{0, 0, 0, 0}
PageModePrintDirection	PageModePrintDirection	R/W	Open	<i>None</i>
PageModeStation	PrinterStation	R/W	Open	<i>None</i>
PageModeVerticalPosition	int	R/W	Open	0
RecBarCodeRotationList	Rotation[]	R	Open	{Normal, Left90, Right90, Rotate180}
RecBitmapRotationList	Rotation[]	R	Open	{Normal, Left90, Right90, Rotate180}
RecCartridgeState	PrinterCartridgeStates	R	Open, Claim, & Enable	<i>Unknown</i>
RecCurrentCartridge	PrinterColors	R/W <sup>*2</sup>	Open, Claim, & Enable	<i>Primary</i>
RecEmpty	bool	R	Open, Claim, & Enable	Depends on printer status.
RecLetterQuality	bool	R/W <sup>*2</sup>	Open, Claim, & Enable	<i>false</i>
RecLineChars	int	R/W	Open, Claim, & Enable	48 <sup>*2</sup>
RecLineCharsList	int[]	R	Open	{36,41,44,48,57,64,72}
RecLineHeight	int	R/W	Open, Claim, & Enable	24 <sup>*3</sup>
RecLineSpacing	int	R/W	Open, Claim, & Enable	30 <sup>*1</sup>
RecLinesToPaperCut	int	R	Open, Claim, & Enable	2 <sup>*3</sup>
RecLineWidth	int	R	Open, Claim, & Enable	576
RecNearEnd	bool	R	Open, Claim, & Enable	<i>false</i>
RecSidewaysMaxChars	int	R	Open, Claim, & Enable	200 <sup>*3</sup>
RecSidewaysMaxLines	int	R	Open, Claim, & Enable	19 <sup>*3</sup>
RotateSpecial	Rotation	R/W	Open	<i>Normal</i>

\*1: Can be modified by the configuration program.

\*2: Cannot be rewritten.

\*3: Automatically modified by the configuration program.



The following specific properties are provided but the operation is not supported.

Property Name	Type	Access	Availability Condition	Default
CapConcurrentJrnRec	bool	R	Open	false
CapConcurrentJrnSlp	bool	R	Open	false
CapConcurrentPageMode	bool	R	Open	false
CapConcurrentRecSlp	bool	R	Open	false
CapJrn2Color	bool	R	Open	false
CapJrnBold	bool	R	Open	false
CapJrnCartridgeSensor	PrinterCartridgeSensors	R	Open	None
CapJrnColor	PrinterColors	R	Open	None
CapJrnDHigh	bool	R	Open	false
CapJrnDWide	bool	R	Open	false
CapJrnDWideDHigh	bool	R	Open	false
CapJrnEmptySensor	bool	R	Open	false
CapJrnItalic	bool	R	Open	false
CapJrnNearEndSensor	bool	R	Open	false
CapJrnPresent	bool	R	Open	false
CapJrnUnderline	bool	R	Open	false
CapSlp2Color	bool	R	Open	false
CapSlpBarCode	bool	R	Open	false
CapSlpBitmap	bool	R	Open	false
CapSlpBold	bool	R	Open	false
CapSlpBothSidesPrint	bool	R	Open	false
CapSlpCartridgeSensor	PrinterCartridgeSensors	R	Open	None
CapSlpColor	PrinterColors	R	Open	None
CapSlpDHigh	bool	R	Open	false
CapSlpDWide	bool	R	Open	false
CapSlpDWideDHigh	bool	R	Open	false
CapSlpEmptySensor	bool	R	Open	false
CapSlpFullSlip	bool	R	Open	false
CapSlpItalic	bool	R	Open	false
CapSlpLeft90	bool	R	Open	false
CapSlpNearEndSensor	bool	R	Open	false
CapSlpPageMode	bool	R	Open	false
CapSlpPresent	bool	R	Open	false
CapSlpRight90	bool	R	Open	false
CapSlpRotate180	bool	R	Open	false
CapSlpUnderline	bool	R	Open	false
JrnCartridgeState	PrinterCartridgeStates	R	Open, Claim, & Enable	Unknown
JrnCurrentCartridge	PrinterColors	R/W	Open, Claim, & Enable	None

Property Name	Type	Access	Availability Condition	Default
JrnEmpty	bool	R	Open, Claim, & Enable	<i>false</i>
JrnLetterQuality	bool	R/W	Open, Claim, & Enable	<i>false</i>
JrnLineChars	int	R/W	Open, Claim, & Enable	0
JrnLineCharsList	int[]	R	Open	[0]
JrnLineHeight	int	R/W	Open, Claim, & Enable	0
JrnLineSpacing	int	R/W	Open, Claim, & Enable	0
JrnLineWidth	int	R	Open, Claim, & Enable	0
JrnNearEnd	bool	R	Open, Claim, & Enable	<i>false</i>
SlpBarCodeRotationList	Rotation[]	R	Open	[0]
SlpBitmapRotationList	Rotation[]	R	Open	[0]
SlpCartridgeState	PrinterCartridgeStates	R	Open, Claim, & Enable	<i>Unknown</i>
SlpCurrentCartridge	PrinterColors	R/W	Open, Claim, & Enable	<i>None</i>
SlpEmpty	bool	R	Open, Claim, & Enable	<i>false</i>
SlpLetterQuality	bool	R/W	Open, Claim, & Enable	<i>false</i>
SlpLineChars	int	R/W	Open, Claim, & Enable	0
SlpLineCharsList	int[]	R	Open	[0]
SlpLineHeight	int	R/W	Open, Claim, & Enable	0
SlpLinesNearEndToEnd	int	R	Open, Claim, & Enable	0
SlpLineSpacing	int	R/W	Open, Claim, & Enable	0
SlpLineWidth	int	R	Open, Claim, & Enable	0
SlpMaxLines	int	R	Open, Claim, & Enable	0
SlpNearEnd	bool	R	Open, Claim, & Enable	<i>false</i>
SlpPrintSide	PrinterSide	R	Open, Claim, & Enable	<i>Unknown</i>
SlpSidewaysMaxChars	int	R	Open, Claim, & Enable	0
SlpSidewaysMaxLines	int	R	Open, Claim, & Enable	0

(3) Common Methods

Method Name	Availability Condition
CheckHealth	Open, Claim, & Enable
Claim	Open
ClearOutput	Open & Claim
Close	Open
CompareFirmwareVersion	Open, Claim, & Enable
DirectIO	Open, Claim, & Enable
Open	-
Release	Open & Claim
ResetStatistic(string)	Open, Claim, & Enable
ResetStatistics()	Open, Claim, & Enable
ResetStatistics(StatisticCategories)	Open, Claim, & Enable
ResetStatistics(string[])	Open, Claim, & Enable
RetrieveStatistic(string)	Open, Claim, & Enable
RetrieveStatistics()	Open, Claim, & Enable
RetrieveStatistics(StatisticCategories)	Open, Claim, & Enable
RetrieveStatistics(string[])	Open, Claim, & Enable
UpdateFirmware	Open, Claim, & Enable
UpdateStatistic	Open, Claim, & Enable
UpdateStatistics(Statistic[])	Open, Claim, & Enable
UpdateStatistics(StatisticCategories, Object)	Open, Claim, & Enable

(4) Specific Methods

Method Name	Availability Condition
BeginInsertion	Open, Claim, & Enable
BeginRemoval	Open, Claim, & Enable
ChangePrintSide	Open, Claim, & Enable
ClearPrintArea	Open, Claim, & Enable
CutPaper	Open, Claim, & Enable
EndInsertion	Open, Claim, & Enable
EndRemoval	Open, Claim, & Enable
MarkFeed	Open, Claim, & Enable
PageModePrint	Open, Claim, & Enable
PrintBarCode	Open, Claim, & Enable
PrintBitmap	Open, Claim, & Enable
PrintImmediate	Open, Claim, & Enable
PrintMemoryBitmap	Open, Claim, & Enable
PrintNormal	Open, Claim, & Enable
PrintTwoNormal	Open, Claim, & Enable
RotatePrint	Open, Claim, & Enable

Method Name	Availability Condition
<b>SetBitmap</b>	Open, Claim, & Enable
<b>SetLogo</b>	Open, Claim, & Enable
<b>TransactionPrint</b>	Open, Claim, & Enable
<b>ValidateData</b>	Open, Claim, & Enable

(5) Events

Event Name	Availability Condition
<b>DirectIOEvent</b>	Open, Claim, & Enable <sup>*1</sup>
<b>ErrorEvent</b>	Open, Claim, & Enable
<b>OutputCompleteEvent</b>	Open, Claim, & Enable
<b>StatusUpdateEvent</b>	Open, Claim, & Enable

<sup>\*1</sup>: The availability condition differs from that of UPOS V 1.12.

#### 4.1.2 Data Characters and Escape Sequences

(1) Escape Sequence operated when specified

Name	Data	Remarks
Paper cut	ESC [#]P	Not supported.
Feed and Paper cut	ESC [#]fP	Not supported.
Feed, Paper cut, and Stamp	ESC [#]sP	Not supported.
Print bitmap	ESC #B	<ul style="list-style-type: none"> <li>Prints the pre-stored bitmap. The placeholder '#' is replaced by the bitmap number. A value from 1 to 20 can be specified for '#'. If values other than 1 to 20 are specified for '#', they are ignored. If the character '#' is omitted, the data is regarded as print data instead of an escape sequence.</li> </ul>
Print top logo	ESC tL	<ul style="list-style-type: none"> <li>Prints the pre-stored top logo.</li> </ul>
Print bottom logo	ESC bL	<ul style="list-style-type: none"> <li>Prints the pre-stored bottom logo.</li> </ul>
Fire stamp	ESC sL	Not supported.
Feed lines	ESC [#]lF	<ul style="list-style-type: none"> <li>Feeds the paper forward by lines. The placeholder '#' is replaced by an ASCII decimal string indicating the number of lines to be fed. A value from 0 to 255 can be specified for '#'. If '#' exceeds this range, the maximum supported number of 255 lines are fed. If '#' is omitted, then one line is fed.</li> <li>This is ignored during rotated 90° right/left mode by <b>RotatePrint</b> or during Page Mode by <b>PageModePrint</b>.</li> </ul>
Feed units	ESC [#]uF	<ul style="list-style-type: none"> <li>Feeds the paper forward by units in <b>MapMode</b>. If <b>MapMode</b> is set to <i>MapMode.Dots</i>, a value from 1 to 255 can be specified for '#'. The placeholder '#' is replaced by an ASCII decimal string indicating the number of units to be fed. If '#' is omitted, then one unit is fed. If '#' exceeds this range, the maximum supported number of 255 units is fed.</li> <li>This is ignored during rotated 90° right/left mode by <b>RotatePrint</b> or during Page Mode by <b>PageModePrint</b>.</li> </ul>
Feed reverse	ESC [#]rF	Not supported.

Name	Data	Remarks
Pass through embedded data	ESC #E	<ul style="list-style-type: none"> <li>Sends the characters following "#E" through to the printer without modifying any of them. The placeholder '#' is replaced by an ASCII decimal string indicating the number of bytes following the escape sequence that should be passed through as-is to the printer. A value from 1 to 65535 can be specified for '#'. If '#' exceeds this range, transmission of embedded data is not executed. If the print data of the number of bytes specified by '#' is not set after the escape sequence is specified, only the transmittable print data is sent. (Example: If ESC 2Ea is specified, only "a" is sent since only one byte is set for the character string.) If '#' is omitted, the data is regarded as print data instead of an escape sequence.</li> <li>During rotated 90° right/left mode by <b>RotatePrint</b>, the width cannot be calculated exactly because data string specified by transmission of embedded data is not counted as character string. Therefore, make an appropriate adjustment by inserting blanks.</li> </ul>
Print in-line barcode	ESC #R	<ul style="list-style-type: none"> <li>Prints a barcode. The placeholder '#' is replaced by an ASCII decimal string indicating the number of characters of the string following R (definition of the barcode characteristics). If '#' is omitted, the data is regarded as print data instead of an escape sequence.</li> <li>If the number of characters specified by '#' does not match the number of bytes following R, all the data within the range specified by '#' is discarded.</li> <li>During rotated 90° right/left mode by <b>RotatePrint</b>, the width cannot be calculated exactly because data string specified by transmission of barcode printing is not counted as character string. Therefore, make an appropriate adjustment by inserting blanks.</li> </ul>

- In-Line Barcode Printing**

The application can print barcodes along with other print data by using the "Print in-line barcode" escape sequence (ESC|#R). The placeholder '#' is replaced by the number of characters of the string (definition of the barcode characteristics) following R.

The string following R specifies the barcode characteristics using lowercase alphabet letters and numbers. The available numbers are the constant values defined for **PrintBarCode**.

The characters indicating the attributes are as follows:

- s: symbology (barcode type)
- h: height (barcode height)
- w: width (barcode width)
- a: alignment (position of barcode)
- t: text position (position of HRI string)
- d: start of data (start position of barcode data)
- e: end of data (end position of barcode data)

Attributes must be written in the above order.

Every attribute is mandatory. If one of these two conditions is not obeyed or a value outside of the range is specified for the number following each attribute, it may cause unpredictable print results.

Below is an example of printing UPC-A under the condition of center, HRI string printed below the barcode, 200 dots height, and 400 dots width.

ESC|33Rs101h200w400a-2t-13d123456789012e

For the barcode quiet zone, see the description of **PrintBarCode**.

(2) Escape Sequence valid until changed

Name	Data	Remarks
Font typeface	ESC #T	Not supported.

(3) Escape Sequence reset by end of print method or "Normal" escape sequence

Name	Data	Remarks
Bold	ESC !bC	<ul style="list-style-type: none"> <li>Prints in bold.</li> <li>If '!' is specified, bold is disabled.</li> </ul>
Underline	ESC !#uC	<ul style="list-style-type: none"> <li>Prints with underline.</li> <li>The placeholder '#' is replaced by an ASCII decimal string indicating the thickness of the underline in printer dot units. The available thickness is from 0 to 2.</li> <li>If '#' is 3 or larger, then a thickness of 2 is used for the underline.</li> <li>If '#' is omitted, then a thickness of 1 is used.</li> <li>If '!' is specified, underline is disabled.</li> </ul>
Italic	ESC !iC	Not supported.
Alternate color (Custom)	ESC [#]rC	Not supported.
Red color	ESC rC	Not supported.
Reverse video	ESC !rvC	<ul style="list-style-type: none"> <li>Prints in a reverse video format.</li> <li>If '!' is specified, reverse video is disabled.</li> </ul>
Shading	ESC [#]sC	Not supported.
Single high and wide	ESC 1C	<ul style="list-style-type: none"> <li>Prints normal size.</li> </ul>
Double wide	ESC 2C	<ul style="list-style-type: none"> <li>Prints double-wide characters.</li> </ul>
Double high	ESC 3C	<ul style="list-style-type: none"> <li>Prints double-high characters.</li> </ul>
Double high and wide	ESC 4C	<ul style="list-style-type: none"> <li>Prints double-high / double-wide characters.</li> </ul>
Scale horizontally	ESC #hC	<ul style="list-style-type: none"> <li>A supported value for the placeholder '#' is 1 to 8.</li> <li>If a value less than 1 is specified for '#', print in 1 scale.</li> <li>If a value greater than 8 is specified for '#', print in 8 scale.</li> <li>If '#' is omitted, the data is regarded as print data instead of an escape sequence.</li> </ul>
Scale vertically	ESC #vC	<ul style="list-style-type: none"> <li>A supported value for the placeholder '#' is 1 to 8.</li> <li>If a value less than 1 is specified for '#', print in 1 scale.</li> <li>If a value greater than 8 is specified for '#', print in 8 scale.</li> <li>If '#' is omitted, the data is regarded as print data instead of an escape sequence.</li> </ul>
RGB Color	ESC [#]fC	Not supported.

Name	Data	Remarks
Center	ESC cA	<ul style="list-style-type: none"> <li>Aligns the text after ESC cA in the center. This must be specified at the head of the line. If not, this is invalid. Also, if there is a linefeed on the print data, the center is valid after linefeed.</li> <li>This specification is ignored during rotated 90° right/left mode by <b>RotatePrint</b> or during Page Mode by <b>PageModePrint</b>.</li> </ul>
Right justify	ESC rA	<ul style="list-style-type: none"> <li>Aligns the text after ESC rA to the right. This must be specified at the head of the line. If not, this is invalid. Also, if there is a linefeed on the print data, the right justify is valid after linefeed.</li> <li>This specification is ignored during rotated 90° right/left mode by <b>RotatePrint</b> or during Page Mode by <b>PageModePrint</b>.</li> </ul>
Left justify	ESC lA	<ul style="list-style-type: none"> <li>Aligns the text after ESC lA to the left. This must be specified at the head of the line. If not, this is invalid. Also, if there is a linefeed on the print data, the left justify is valid after linefeed.</li> <li>This specification is ignored during rotated 90° right/left mode by <b>RotatePrint</b> or during Page Mode by <b>PageModePrint</b>.</li> </ul>
Normal	ESC N	<ul style="list-style-type: none"> <li>Restores printer characteristics to normal condition.</li> </ul>
SubScript	ESC [!]tbC	Not supported.
SuperScript	ESC [!]tpC	Not supported.
Strike-through	ESC [!][#]stC	Not supported.



### 4.1.3 Common Properties

This section describes the details of the common properties for PosPrinter.  
For details of the thrown exception errors, see "Appendix A Exceptions".

#### CapCompareFirmwareVersion Property

Type **bool**

Description Gets a Boolean value that indicates whether the Service Object/device supports comparing the firmware version in the physical device against that of a firmware file.  
The following table shows the valid property values.

Value	Meaning
<i>false</i>	The function that compares firmware versions is not supported.

This property is initialized to *false* by **Open**.

#### CapPowerReporting Property

Type **PowerReporting**

Description Gets the power reporting capabilities of the device.  
The following table shows the valid property values.

Value	Meaning
<i>PowerReporting.Standard</i>	The following 2 types of power states can be determined and reported. <ul style="list-style-type: none"><li>• <i>PowerState.OffOffline</i> (power off or offline)</li><li>• <i>PowerState.Online</i></li></ul>

This property is initialized to *PowerReporting.Standard* by **Open**.

#### CapStatisticsReporting Property

Type **bool**

Description Gets a Boolean value that indicates whether the device can accumulate and can provide various statistics regarding usage.  
The following table shows the valid property values.

Value	Meaning
<i>true</i>	The device accumulates and can provide various statistics regarding usage. The information accumulated and reported is device specific, and is retrieved using <b>RetrieveStatistic(s)</b> .

This property is initialized to *true* by **Open**.

## CapUpdateFirmware Property

Type **bool**

Description Gets a Boolean value that indicates whether the device's firmware can be updated through **UpdateFirmware**.

The following table shows the valid property values.

Value	Meaning
<i>false</i>	Firmware update is not supported.

This property is initialized to *false* by **Open**.

## CapUpdateStatistics Property

Type **bool**

Description Gets a Boolean value that indicates whether some or all the device statistics can be reset to 0 using **ResetStatistic(s)**.

The following table shows the valid property values.

Value	Meaning
<i>true</i>	The device statistics, or some of the statistics, can be reset to 0 using <b>ResetStatistic(s)</b> .

This property is initialized to *true* by **Open**.

## CheckHealthText Property

Type **string**

Description Gets a string that indicates the health of the device.

This property is updated by the Service Object when the application calls **CheckHealth**.

The following examples show the results of diagnosis.

Method Parameter	Method Result	CheckHealthText
<i>HealthCheckLevel.External</i>	Success	"External HCheck: Successful"
	Fail	"External HCheck: Failure"
<i>HealthCheckLevel.Interactive</i> <sup>*1</sup>	Success	"Interactive HCheck: Successful"
	Fail	"Interactive HCheck: Failure"
<i>HealthCheckLevel.Internal</i>	Success	"Internal HCheck: Successful"
	Fail	"Internal HCheck: Failure"

<sup>\*1</sup>: In the case of *HealthCheckLevel.Interactive*, if the dialog box is closed without testing after execution, "Interactive HCheck: Canceled" is set.

This property is initialized to empty string by **Open**.

## Claimed Property

Type **bool**

Description Gets a Boolean value that indicates whether the device is claimed for exclusive access. The following table shows the valid property values.

Value	Meaning
<i>false</i>	The device is released for sharing with other applications.
<i>true</i>	The exclusive access to the device is obtained.

This property is initialized to *false* by **Open**.

## DeviceDescription Property

Type **string**

Description Gets a string identifying the device and the company that manufactured it. This property depends on **DeviceName**. This property is initialized to the following values by **Open**.

DeviceName	Value
"MP-B30 POS Printer"	"SII MP-B30 POS Printer"

## DeviceEnabled Property R/W

Type **bool**

Description Gets or sets a Boolean value that indicates whether the device has been placed in an operational state. The following table shows the valid property values.

Value	Meaning
<i>false</i>	The device has been disabled. If changed to <i>false</i> , then the device is physically disabled when possible, any subsequent input will be discarded, and output operations are disallowed.
<i>true</i>	The device is in an operational state. If changed to <i>true</i> , then the device is brought to an operational state.

The application must set this property to *true* before using the device.

If **State** is other than *ControlState.Idle*, **DeviceEnabled** cannot be changed from *true* to *false*.

This property is initialized to *false* by **Open**.

## DeviceName Property

Type **string**

Description Gets a short string identifying the device and any pertinent information about it.  
This property is initialized to the following values by **Open**.

Printer	Value
MP-B30	"MP-B30 POS Printer"

## FreezeEvents Property R/W

Type **bool**

Description Gets or sets a Boolean value that indicates whether the application has requested that the Service Object not deliver events.  
The following table shows the valid property values.

Value	Meaning
<i>false</i>	The application allows events to be delivered. If some events have been held while events were frozen and all other conditions are correct for delivering the events, changing the <b>FreezeEvents</b> property to <i>false</i> allows these events to be delivered.
<i>true</i>	The application has requested that the Service Object not deliver events. Events will be queued by the Service Object but not delivered until the application changes the <b>FreezeEvents</b> property to <i>false</i> .

An application may choose to freeze events for a specific sequence of code where interruption by an event is not desirable.

If an error occurs while a print method such as **PrintNormal** is operated under **AsyncMode** is *true*, **ErrorEvent** is frozen and **State** turns to *ControlState.Busy*. In this case, discard the frozen event by **ClearOutput** or set **FreezeEvents** to *false* to cause **ErrorEvent**, and then execute **Close**, since the Service Object cannot be closed under this circumstance.

This property is initialized to *false* by **Open**.

## OutputId Property

Type **int**

Description Holds the identifier of the most recently started asynchronous output (call to an asynchronous method when **AsyncMode** is set to *true*).

When a method successfully initiates an asynchronous output, the Service Object assigns an identifier to the request. When the output completes, the Service Object will fire an **OutputCompleteEvent** passing this output ID as a parameter.

**OutputId** is allocated automatically within the range of **int**.

This property is initialized to 0 by **Open**.

## PowerNotify Property R/W

Type **PowerNotification**

Description Gets or sets the type of power notification selection made by the application.  
The following table shows the valid property values.

Value	Meaning
<i>PowerNotification.Disabled</i>	The Service Object will not provide any power notifications to the application. No power notification <b>StatusUpdateEvents</b> will be fired, and <b>PowerState</b> may not be set.
<i>PowerNotification.Enabled</i>	When <b>DeviceEnabled</b> is set to <i>true</i> , the Service Object will fire the power notification <b>StatusUpdateEvents</b> and update <b>PowerState</b> . The level of functionality depends on the value of <b>CapPowerReporting</b> .

**PowerNotify** can be set only while the device is disabled; that is, while **DeviceEnabled** is *false*.

This property is initialized to *PowerNotification.Disabled* by **Open**.

## PowerState Property

Type **PowerState**

Description Gets the current power condition of the device.  
The following table shows the valid property values.

Value	Meaning
<i>PowerState.OffOffline</i>	The device is powered off or offline.
<i>PowerState.Online</i>	The device is powered on and ready for use.
<i>PowerState.Unknown</i>	Cannot determine the device's power state due to one of the following reasons. <ul style="list-style-type: none"><li>• <b>PowerNotify</b> is <i>PowerNotification.Disabled</i>.</li><li>• <b>DeviceEnabled</b> is <i>false</i>.</li></ul>

This property is initialized to *PowerState.Unknown* by **Open**.

## ServiceObjectDescription Property

Type            **string**

Description    Gets a string identifying the Service Object that supports the device and the company that produced it.

This property is initialized to the following values by **Open**.

DeviceName	Value
"MP-B30 POS Printer"	"SII MP-B30 POS Printer Service Object, Copyright (C) 20xx Seiko Instruments Inc."

## ServiceObjectVersion Property

Type            **Version**

Description    Gets the Service Object version number.

Version numbers consist of four integers; Major, Minor, Build, and Revision.

The Major and Minor version numbers correspond to the UPOS version that the Service Object implements.

When Build version is A, Revision version is B, this property is initialized to 1.12.A.B by **Open**.

## State Property

Type            **ControlState**

Description    Gets the current state of the device.

The following table shows the valid property values.

Value	Meaning
<i>ControlState.Busy</i>	The device is in a normal state and is busy executing output.
<i>ControlState.Closed</i>	The device is closed.
<i>ControlState.Error</i>	An error has been reported, and the application must recover the Control to a normal state before normal I/O can resume. This state is only possible inside the <b>ErrorEvent</b> event handler.
<i>ControlState.Idle</i>	The device is in a normal state and is not busy.

This property is always readable.

This property is initialized to *ControlState.Idle* by **Open**.

## SynchronizingObject Property

Type	<b>System.ComponentModel.ISynchronizeInvoke</b>
Description	<p>This property holds an object that implements the .NET Framework class <b>ISynchronizeInvoke</b> that is used to marshal events to a particular thread. If <b>SynchronizingObject</b> is set to <i>null</i>, events are raised on a system thread owned by the Service Object.</p> <p>The application using a Windows form sets the <i>this</i> pointer of the <b>Form</b> class of the main form to <b>SynchronizationObject</b>, so that events are notified to the main application thread as required by the <b>Form</b> class.</p>

#### 4.1.4 Specific Properties

This section describes the details of the specific properties for PosPrinter.

For exception errors of specific properties that are not supported, see "Appendix A Exceptions".

##### AsyncMode Property R/W

Type **bool**

Description Gets or sets a Boolean value that indicates whether certain print methods will be performed asynchronously.

The following table shows the valid property values.

Value	Meaning
<i>false</i>	<b>PrintNormal, PrintBarCode, PrintBitmap, PrintMemoryBitmap, MarkFeed, RotatePrint, TransactionPrint, and PageModePrint</b> print methods are executed synchronously.
<i>true</i>	The methods are executed asynchronously.

This property is initialized to *false* by **Open**.

##### CapCharacterSet Property

Type **CharacterSetCapability**

Description Indicates the printable character setting.

The following table shows the valid property values.

Value	Meaning
<i>CharacterSetCapability.Kanji</i>	The character setting supports Code Page 932, including ASCII characters 0x20 through 0x7F and the one-byte katakana characters 0xA1 through 0xDF. It also includes the Shift-JIS code characters defined in JIS 1st and 2nd levels.

This property is initialized to *CharacterSetCapability.Kanji* by **Open**.

##### CapCoverSensor Property

Type **bool**

Description Gets a Boolean value that indicates whether the printer has a "cover open" sensor function for the receipt.

The following table shows the valid property values.

Value	Meaning
<i>true</i>	The printer has a "cover open" sensor.

This property is initialized to *true* by **Open**.



## CapMapCharacterSet Property

Type **bool**

Description Gets a Boolean value that indicates that the Service Object is able to map the characters of the application to a character set defined by **CharacterSetList**.  
The following table shows the valid property values.

Value	Meaning
<i>false</i>	The Service Object cannot exactly map the characters to the character sets defined in <b>CharacterSetList</b> .

This property is initialized to *false* by **Open**.

## CapRec2Color Property

Type **bool**

Description Gets a Boolean value that indicates whether the receipt can print dark plus an alternate color.  
The following table shows the valid property values.

Value	Meaning
<i>false</i>	Two color printing of the receipt is not supported.

This property is initialized to *false* by **Open**.

## CapRecBarCode Property

Type **bool**

Description Gets a Boolean value that indicates whether the receipt has barcode printing capability.  
The following table shows the valid property values.

Value	Meaning
<i>true</i>	The receipt has barcode printing capability.

This property is initialized to *true* by **Open**.

## CapRecBitmap Property

Type **bool**

Description Gets a Boolean value that indicates whether the receipt can print bitmaps.  
The following table shows the valid property values.

Value	Meaning
<i>true</i>	The receipt can print bitmaps.

This property is initialized to *true* by **Open**.

## CapRecBold Property

Type **bool**

Description Gets a Boolean value that indicates whether the receipt can print bold characters. The following table shows the valid property values.

Value	Meaning
<i>true</i>	The receipt can print bold characters.

This property is initialized to *true* by **Open**.

## CapRecCartridgeSensor Property

Type **PrinterCartridgeSensors**

Description Gets a value that indicates the presence of receipt cartridge monitoring sensors. The following table shows the valid property values.

Value	Meaning
<i>PrinterCartridgeSensors.None</i>	Receipt cartridge monitoring sensors are not supported.

This property is initialized to *PrinterCartridgeSensors.None* by **Open**.

## CapRecColor Property

Type **PrinterColors**

Description Gets a value that indicates available receipt color cartridges. The following table shows the valid property values.

Value	Meaning
<i>PrinterColors.Primary</i>	Receipt supports primary color (Black).

This property is initialized to *PrinterColors.Primary* by **Open**.

## CapRecDHigh Property

Type **bool**

Description Gets a Boolean value that indicates whether the receipt can print double high characters. The following table shows the valid property values.

Value	Meaning
<i>true</i>	The receipt can print double high characters.

This property is initialized to *true* by **Open**.

## CapRecDWide Property

Type **bool**

Description Gets a Boolean value that indicates whether the receipt can print double wide characters. The following table shows the valid property values.

Value	Meaning
<i>true</i>	The receipt can print double wide characters.

This property is initialized to *true* by **Open**.

## CapRecDWideDHigh Property

Type **bool**

Description Gets a Boolean value that indicates whether the receipt can print double high/double wide characters. The following table shows the valid property values.

Value	Meaning
<i>true</i>	The receipt can print double high/double wide characters.

This property is initialized to *true* by **Open**.

## CapRecEmptySensor Property

Type **bool**

Description Gets a Boolean value that indicates whether the receipt has an "out-of-paper" sensor. The following table shows the valid property values.

Value	Meaning
<i>true</i>	The receipt has an "out-of-paper" sensor.

This property is initialized to *true* by **Open**.

## CapRecItalic Property

Type **bool**

Description Gets a Boolean value that indicates whether the receipt can print italic characters. The following table shows the valid property values.

Value	Meaning
<i>false</i>	The receipt cannot print Italic characters.

This property is initialized to *false* by **Open**.

## CapRecLeft90 Property

Type **bool**

Description Gets a Boolean value that indicates whether the receipt can print in a rotated 90° left mode.

The following table shows the valid property values.

Value	Meaning
<i>true</i>	The receipt can print in a rotated 90° left mode.

This property is initialized to *true* by **Open**.

## CapRecMarkFeed Property

Type **PrinterMarkFeeds**

Description Indicates the type of mark sensed paper handling available.  
Either a logical OR of the following values is set in this property.

Value	Meaning
<i>PrinterMarkFeeds.None</i>	Control function for mark sensed paper is disabled.
<i>PrinterMarkFeeds.Takeup</i>	After detecting the mark, feeds the paper to the paper take-up position.
<i>PrinterMarkFeeds.Cutter</i>	After detecting the mark, feeds the paper to the cutting position. (Feeds the paper to the same position as <i>PrinterMarkFeeds.Takeup</i> .)

The default of this property can be changed at the setting in the configuration program.

This property is initialized to *PrinterMarkFeeds.None* by **Open**.

## CapRecNearEndSensor Property

Type **bool**

Description Gets a Boolean value that indicates whether the receipt has a low-paper sensor.  
The following table shows the valid property values.

Value	Meaning
<i>false</i>	The printer does not have a "paper-near-end" sensor.

This property is initialized to *false* by **Open**.

## CapRecPageMode Property

Type **bool**

Description Gets a Boolean value that indicates whether the printer can support Page Mode for the receipt station.

The following table shows the valid property values.

Value	Meaning
<i>true</i>	The printer can support Page Mode for the receipt station.

This property is initialized to *true* by **Open**.

## CapRecPaperCut Property

Type **bool**

Description Gets a Boolean value that indicates whether the receipt can perform paper cuts.

The following table shows the valid property values.

Value	Meaning
<i>false</i>	The receipt does not have a paper cut capability.

This property is initialized to *false* by **Open**.

## CapRecPresent Property

Type **bool**

Description Gets a Boolean value that indicates whether the receipt print station is present.

The following table shows the valid property values.

Value	Meaning
<i>true</i>	The receipt print station is present.

This property is initialized to *true* by **Open**.

## CapRecRight90 Property

Type **bool**

Description Gets a Boolean value that indicates whether the receipt can print in a rotated 90° right mode.

The following table shows the valid property values.

Value	Meaning
<i>true</i>	The receipt can print in a rotated 90° right mode.

This property is initialized to *true* by **Open**.

## CapRecRotate180 Property

Type **bool**

Description Gets a Boolean value that indicates whether the receipt can print in a rotated upside down mode.

The following table shows the valid property values.

Value	Meaning
<i>true</i>	The receipt can print in a rotated upside down mode.

This property is initialized to *true* by **Open**.

## CapRecStamp Property

Type **bool**

Description Gets a Boolean value that indicates whether the receipt has a stamp capability.

The following table shows the valid property values.

Value	Meaning
<i>false</i>	The receipt does not have a stamp capability.

This property is initialized to *false* by **Open**.

## CapRecUnderline Property

Type **bool**

Description Gets a Boolean value that indicates whether the receipt can print underlined characters.

The following table shows the valid property values.

Value	Meaning
<i>true</i>	The receipt can print underlined characters.

This property is initialized to *true* by **Open**.

## CapTransaction Property

Type **bool**

Description Gets a Boolean value that indicates whether receipt station supports printer transactions.

The following table shows the valid property values.

Value	Meaning
<i>true</i>	The receipt station supports printer transactions.

This property is initialized to *true* by **Open**.

## CartridgeNotify Property R/W

Type **PrinterCartridgeNotify**

Description Gets or sets the type of printer cartridge state notification the application wants to receive. The following table shows the valid property values.

Value	Meaning
<i>PrinterCartridgeNotify.Disabled</i>	Cartridge-state notifications are not available.

This property cannot be rewritten.

This property is initialized to *PrinterCartridgeNotify.Disabled* by **Open**.

## CharacterSet Property R/W

Type **int**

Description Gets or sets the numeric value that indicates the character set that the application wants to use for printing characters.

The following table shows the valid property values.

Value	Meaning
437	Selects Code Page 437 character set.
737	Selects Code Page 737 character set.
850	Selects Code Page 850 character set.
852	Selects Code Page 852 character set.
855	Selects Code Page 855 character set.
857	Selects Code Page 857 character set.
858	Selects Code Page 858 character set.
860	Selects Code Page 860 character set.
863	Selects Code Page 863 character set.
865	Selects Code Page 865 character set.
866	Selects Code Page 866 character set.
932	Selects Katakana as Code Page 932 character set (Shift-JIS Code).
999	Selects Windows ANSI character set.*1
1250	Selects Code Page 1250 character set.
1251	Selects Code Page 1251 character set.
1252	Selects Code Page 1252 character set.*1
1253	Selects Code Page 1253 character set.
1254	Selects Code Page 1254 character set.

\*1: Windows ANSI character set is equal to Code Page 1252 character set.

The default of this property can be changed at the setting in the configuration program. This property is initialized to the value of character set which is set in [Character Set] of the configuration program by **Open**.

## CharacterSetList Property

Type            **int[]**

Description    Gets the list of character set numbers supported for printing.  
This property is initialized to {437, 737, 850, 852, 855, 857, 858, 860, 863, 865, 866, 932, 999, 1250, 1251, 1252, 1253, 1254} by **Open**.

## CoverOpen Property

Type            **bool**

Description    Gets a Boolean value that indicates whether the printer's cover is open.  
The following table shows the valid property values.

Value	Meaning
<i>false</i>	The printer's cover is closed.
<i>true</i>	The printer's cover is open.

This property is set and kept current by the Service Object while the device is enabled.

## ErrorLevel Property

Type            **PrinterErrorLevel**

Description    Gets the severity of the most recent error condition.  
The following table shows the valid property values.

Value	Meaning
<i>PrinterErrorLevel.Fatal</i>	A non-recoverable error has occurred.
<i>PrinterErrorLevel.None</i>	No error condition is present.
<i>PrinterErrorLevel.Recoverable</i>	A recoverable error has occurred.

This property is set by the Service Object only before delivering an **ErrorEvent**. When the error is cleared, the Service Object changes **ErrorLevel** to *PrinterErrorLevel.None*.  
This property is initialized to *PrinterErrorLevel.None* by **Open**.



## ErrorStation Property

Type           **PrinterStation**

Description    Gets the station that was printing when an error was detected.  
The following table shows the valid property values.

Value	Meaning
<i>PrinterStation.None</i>	The error was not detected.
<i>PrinterStation.Receipt</i>	The error is detected at the receipt station.

This property is set by the Service Object only before delivering an **ErrorEvent** to the application. When the error is cleared, the Service Object changes **ErrorStation** to *PrinterStation.None*.

This property is initialized to *PrinterStation.None* by **Open**

## ErrorString Property

Type           **string**

Description    Holds a vendor-supplied description of the current error.  
The following table shows the valid property values.

Setting Priority	ErrorCode	ErrorCodeExtended	String
1	<i>ErrorCode.NoHardware</i>		The power supply of the device is off.
2	<i>ErrorCode.Extended</i>	<i>ExtendedErrorFatal</i> (1010)	Unrecoverable error occurred.
3	<i>ErrorCode.Extended</i>	<i>ExtendedErrorBattery</i> (1013)	Battery error occurred.
4	<i>ErrorCode.Extended</i>	<i>ExtendedErrorHeadTemp</i> (1005)	Head temperature error.
5	<i>ErrorCode.Extended</i>	<i>ExtendedErrorVpPower</i> (1001)	Vp power error occurred.
6	<i>ErrorCode.Extended</i>	<i>ExtendedErrorCoverOpen</i> (201)	The cover is open.
7	<i>ErrorCode.Extended</i>	<i>ExtendedErrorReceiptEmpty</i> (203)	Out of receipt form.
8	<i>ErrorCode.Extended</i>	<i>ExtendedErrorMarkPaperJam</i> (1014)	Mark paper jam error occurred.
9	<i>ErrorCode.Failure</i>		Communication error occurred.
			Windows system error occurred.
			Time out.

The values in the above table are described in descending order of priority. When multiple errors occur simultaneously, the higher-priority value is set.

This property is set **ErrorString** only before delivering an **ErrorEvent** to the application. When the error is cleared, the Service Object changes **ErrorString** to an empty string.

This property is initialized to empty string by **Open**.

## FlagWhenIdle Property R/W

- Type **bool**
- Description Gets or sets a Boolean value that indicates whether or not to notify that **State** turns to *ControlState.Idle*.  
The following table shows the valid property values.

Value	Meaning
<i>false</i>	<b>StatusUpdateEvent</b> is not notified.
<i>true</i>	<b>StatusUpdateEvent</b> will be sent when <b>State</b> is <i>ControlState.Idle</i> .

**FlagWhenIdle** is automatically reset to *false* when **StatusUpdateEvent** is notified after **FlagWhenIdle** is set to *true*.

By using **FlagWhenIdle** and **StatusUpdateEvent**, the application can know when all outstanding asynchronous outputs are finished. The event will be notified if the outputs are completed successfully or the outputs are deleted by **ClearOutput** or the event handler that receives **ErrorEvent**.

If **State** is already set to *ControlState.Idle* when **FlagWhenIdle** is set to *true*, then a **StatusUpdateEvent** is notified immediately. The application can therefore use this event without regard to the disagreement between the finish of asynchronous output and the setting of this flag.

This property is initialized to *false* by **Open**.

## FontTypefaceList Property

- Type **string[]**
- Description Gets a string array that specifies the fonts and typefaces supported by the printer.  
An empty array indicates that only the default font is supported.  
This property is initialized to an empty string array by **Open**.

## MapCharacterSet Property R/W

- Type **bool**
- Description Gets a Boolean value that indicates whether character mapping is supported or not.  
The following table shows the valid property values.

Value	Meaning
<i>false</i>	No mapping is supported.

This property cannot be rewritten.

This property is initialized to *false* by **Open**.

## MapMode Property R/W

Type **MapMode**

Description Holds the mapping mode of the printer.  
The mapping mode defines the unit of measure used for other properties, such as line heights and line spacings.

The following mapping modes are supported.  
The values in () indicate the values converted into dot.

Parameter	Meaning
<i>MapMode.Dots</i>	Printer's dot width 0.125 mm (1 dot)
<i>MapMode.English</i>	0.001 inch (0.203 dots)
<i>MapMode.Metric</i>	0.01 mm (0.08 dots)
<i>MapMode.Twips</i>	1/1440 of an inch (0.1411 dots)

For each mapping mode, the unit is converted using one of the following calculation formulae.

Parameter	Conversion
<i>MapMode.Dots</i>	No conversion
<i>MapMode.English</i>	$k = 1/1000$ ■ <i>MapMode.Dots</i> to <i>MapMode.English</i> conversion $\text{english} = \text{dot} / (\text{dpi} \times k)$ ■ <i>MapMode.English</i> to <i>MapMode.Dots</i> conversion $\text{dot} = \text{english} \times \text{dpi} \times k$
<i>MapMode.Metric</i>	$k = 1/100$ , $\text{mmpi} = 25.4$ ■ <i>MapMode.Dots</i> to <i>MapMode.Metric</i> conversion $\text{metric} = (\text{mmpi} \times \text{dot}) / (\text{dpi} \times k)$ ■ <i>MapMode.Metric</i> to <i>MapMode.Dots</i> conversion $\text{dot} = (\text{metric} \times \text{dpi} \times k) / \text{mmpi}$
<i>MapMode.Twips</i>	$k = 1/1440$ ■ <i>MapMode.Dots</i> to <i>MapMode.Twips</i> conversion $\text{twips} = \text{dot} / (\text{dpi} \times k)$ ■ <i>MapMode.Twips</i> to <i>MapMode.Dots</i> conversion $\text{dot} = \text{twips} \times \text{dpi} \times k$

**MapMode** only changes the unit of each property for display, and all internal processings are executed in dots regardless of **MapMode**.

Therefore, the rounding errors of values do not accumulate.

When converting a dot value to a map mode value, the value is rounded up to an integer.  
When converting from a map mode value to a dot value, the decimal part is truncated.

Setting this property may also change **RecLineSpacing**, **RecLineWidth**, **RecLineHeight**, **PageModeArea**, **PageModePrintArea**, **PageModeHorizontalPosition**, and **PageModeVerticalPosition**.

This property is initialized to *MapMode.Dots* when the device is first enabled following **Open**.

## PageModeArea Property

Type **System.Drawing.Point**

Description Gets the page area for the selected **PageModeStation** expressed in the unit of measure given by **MapMode**.  
Specify *PrinterStation.Receipt* for **PageModeStation** before accessing this property.  
When *PrinterStation.Receipt* is specified for **PageModeStation**, the following values are set to this property.

RecLineWidth	Value When MapMode = <i>MapMode.Dots</i>
576	<i>Point.X</i> = 576, <i>Point.Y</i> = 2400

This property is initialized to {*Point.X* = 0, *Point.Y* = 0} by **Open**.

## PageModeDescriptor Property

Type **PageModeDescriptors**

Description Gets the basic Page Mode functionality of the printer for the selected **PageModeStation**. This property is indicated by OR of the following values.  
Specify *PrinterStation.Receipt* for **PageModeStation** before accessing this property.  
When *PrinterStation.Receipt* is specified for **PageModeStation**, OR of *PageModeDescriptors.Bitmap*, *PageModeDescriptors.BitmapRotate*, *PageModeDescriptors.Barcode*, and *PageModeDescriptors.BarcodeRotate* is set to this property.

Value	Meaning
<i>PageModeDescriptors.Barcode</i>	Printing of barcodes on the <b>PageModeStation</b> is supported
<i>PageModeDescriptors.BarcodeRotate</i>	Rotation of barcodes on the <b>PageModeStation</b> is supported
<i>PageModeDescriptors.Bitmap</i>	Printing of bitmaps on the <b>PageModeStation</b> is supported
<i>PageModeDescriptors.BitmapRotate</i>	Rotation of bitmaps on the <b>PageModeStation</b> is supported

This property is initialized to *PageModeDescriptors.None* by **Open**.

## PageModeHorizontalPosition Property R/W

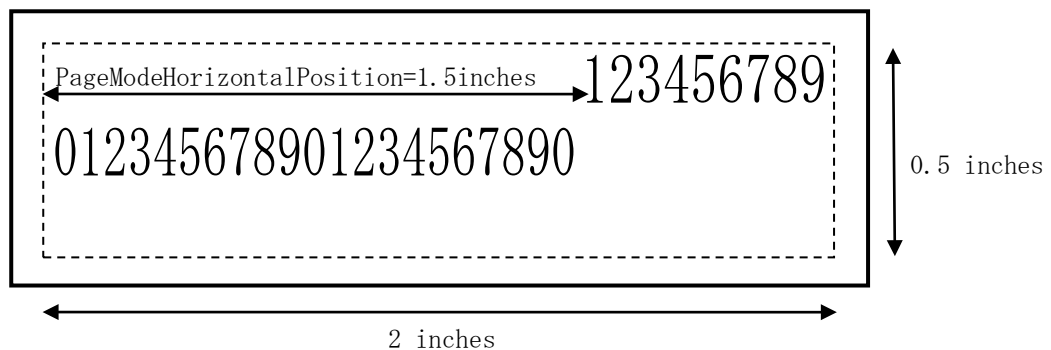
Type            **int**

Description    Gets or sets the horizontal start position offset within the print area for the print station specified by **PageModeStation**.  
The property is expressed in the unit indicated by **MapMode**.  
The horizontal direction is the same as the actual **PageModePrintDirection**.  
A read/get on this property will return the horizontal position offset set by the last write/set and not the current position.  
**PageModeStation** must be set to *PrinterStation.Receipt* before accessing this property, otherwise the value 0 is returned.

The following code sample shows the usage of **PageModeHorizontalPosition**.

```
myptr.MapMode=MapMode.English;  
myptr.PageModeStation=PrinterStation.Receipt;  
myptr.PageModePrint(PageModePrintControl.PageMode);  
// Set print area to 2 inches by 0.5 inches  
myptr.PageModePrintArea=new System.Drawing.Point(0,0,2000,500);  
myptr.PageModePrintDirection=PageModePrintDirection.LeftToRight;  
myptr.PageModeHorizontalPosition=1500;  
myptr.PrintNormal(PrinterStation.Receipt,"123456789012345678901234567890\n");  
myptr.PageModePrint(PageModePrintControl.Normal);
```

The code sample above will generate the following receipt.



This property is initialized to 0 by **Open**.

## PageModePrintArea Property R/W

Type **System.Drawing.Rectangle**

Description Holds the Page Mode print area for the selected **PageModeStation** expressed in the unit specified for **MapMode**.

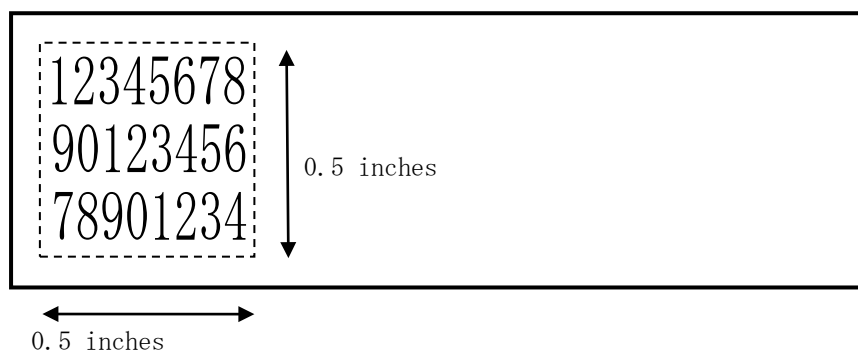
The maximum possible print area of **PageModePrintArea** is the page area of **PageModeArea**.

**PageModeStation** must be set to *PrinterStation.Receipt* before accessing this property, otherwise *Rectangle.Empty* is returned.

The following code sample shows the usage of **PageModePrintArea**.

```
myptr.MapMode=MapMode.English;
myptr.PageModeStation=PrinterStation.Receipt;
myptr.PageModePrint(PageModePrintControl.PageMode);
// Set print area to half inch square block
myptr.PageModePrintArea=new System.Drawing.Point(0,0,500,500);
myptr.PageModePrintDirection=PageModePrintDirection.LeftToRight;
myptr.PrintNormal(PrinterStation.Receipt,"123456789012345678901234\n");
myptr.PageModePrint(PageModePrintControl.Normal);
```

The code sample above will generate the following receipt.



This property is initialized to {*Rectangle.x* = 0, *Rectangle.y* = 0, *Rectangle.width* = 0, *Rectangle.height* = 0} by **Open**.

## PageModePrintDirection Property R/W

Type **PageModePrintDirection**

Description Gets the print direction, as specified by the **PageModePrintDirection** enumeration. **PageModeStation** must be set to *PrinterStation.Receipt* before accessing this property, otherwise *PageModePrintDirection.None* is returned.

The following table shows the valid property values.

Value	Meaning
<i>PageModePrintDirection.BottomToTop</i>	Rotated left 90° printing. Print from bottom to top, starting at the bottom left corner of the Page Mode print area.
<i>PageModePrintDirection.LeftToRight</i>	Normal direction printing. Print from left to right, starting at the top left corner of the Page Mode print area.
<i>PageModePrintDirection.None</i>	The print direction is not specified.
<i>PageModePrintDirection.RightToLeft</i>	Upside down printing. Print from right to left, starting at the bottom right corner of the Page Mode print area.
<i>PageModePrintDirection.TopToBottom</i>	Rotated right 90° printing. Print from top to bottom, starting at the top right corner of the Page Mode print area.

Changing this property may also change the correction direction of the print start point indicated by **PageModeHorizontalPosition** and **PageModeVerticalPosition**. Changing this property is only effective for the current print area. By changing the print areas, it is possible to generate a receipt with text printed in multiple rotations.

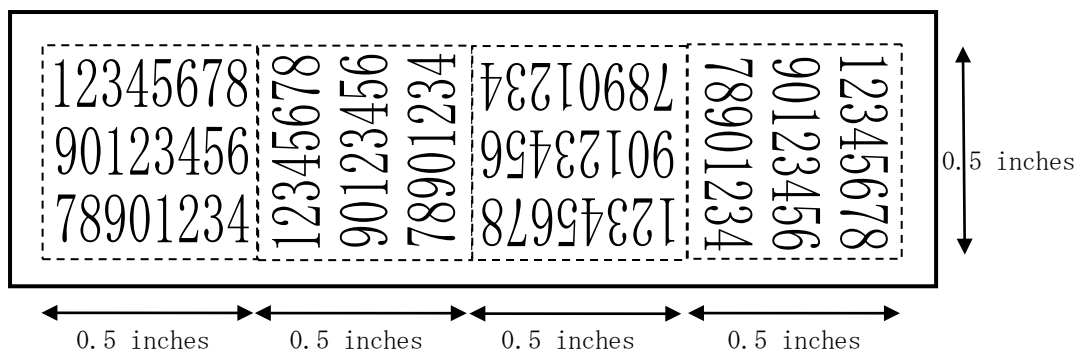
The following code sample shows the usage of **PageModePrintDirection**.

```

myptr.MapMode=MapMode.English;
myptr.PageModeStation=PrinterStation.Receipt;
myptr.PageModePrint(PageModePrintControl.PageMode);
// Set print area to half inch square block
myptr.PageModePrintArea=new System.Drawing.Point(0,0,500,500);
myptr.PageModePrintDirection=PageModePrintDirection.LeftToRight;
myptr.PrintNormal(PrinterStation.Receipt,"123456789012345678901234\n");
myptr.PageModePrintArea=new System.Drawing.Point(500,0,500,500);
myptr.PageModePrintDirection=PageModePrintDirection.BottomToTop;
myptr.PrintNormal(PrinterStation.Receipt,"123456789012345678901234\n");
myptr.PageModePrintArea=new System.Drawing.Point(1000,0,500,500);
myptr.PageModePrintDirection=PageModePrintDirection.RightToLeft;
myptr.PrintNormal(PrinterStation.Receipt,"123456789012345678901234\n");
myptr.PageModePrintArea=new System.Drawing.Point(1500,0,500,500);
myptr.PageModePrintDirection=PageModePrintDirection.TopToBottom;
myptr.PrintNormal(PrinterStation.Receipt,"123456789012345678901234\n");
myptr.PageModePrint(PageModePrintControl.Normal);

```

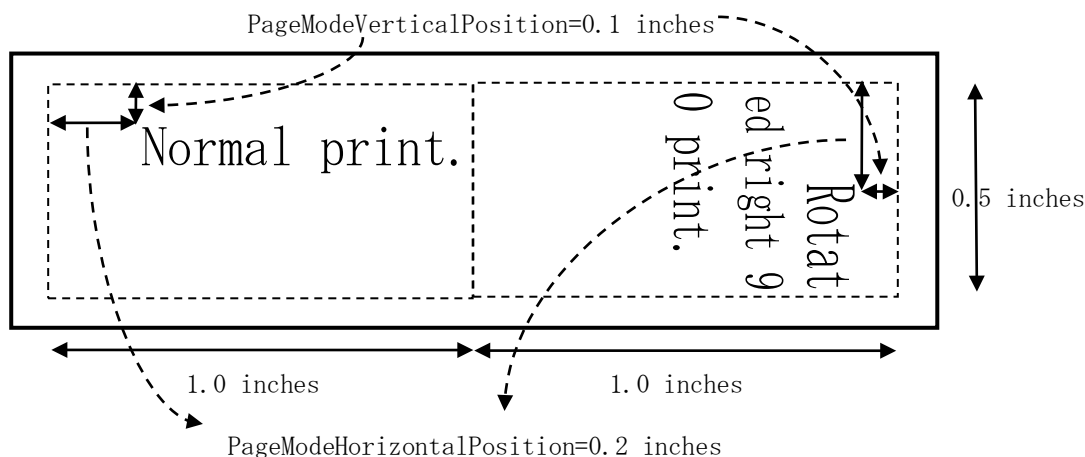
The code sample above will generate the following receipt.



It is also possible to generate rotated text.

```
myptr.MapMode=MapMode.English;
myptr.PageModeStation=PrinterStation.Receipt;
myptr.PageModePrint(PageModePrintControl.PageMode);
myptr.PageModeVerticalPosition=100;
myptr.PageModeHorizontalPosition=200;
myptr.PageModePrintArea=new System.Drawing.Point(0,0,1000,500);
myptr.PageModePrintDirection=PageModePrintDirection.LeftToRight;
myptr.PrintNormal(PrinterStation.Receipt,"Normal print.\n");
myptr.PageModePrintArea=new System.Drawing.Point(1000,0,1000,500);
myptr.PageModePrintDirection=PageModePrintDirection.TopToBottom;
myptr.PrintNormal(PrinterStation.Receipt,"Rotated right 90 print.\n");
myptr.PageModePrint(PageModePrintControl.Normal);
```

The code sample above will generate the following receipt.



This property is initialized to *PageModePrintDirection.None* by **Open**.

And this property is set to *PageModePrintDirection.LeftToRight* when a valid station is specified.



## PageModeStation Property R/W

Type	<b>PrinterStation</b>
Description	<p>Gets or sets the printer station, as specified by the <b>PrinterStation</b> enumeration.</p> <p>Be sure to specify <i>PrinterStation.Receipt</i> for this property before accessing the Page Mode properties or methods.</p> <p>This property is initialized to <i>PrinterStation.None</i> by <b>Open</b>.</p>

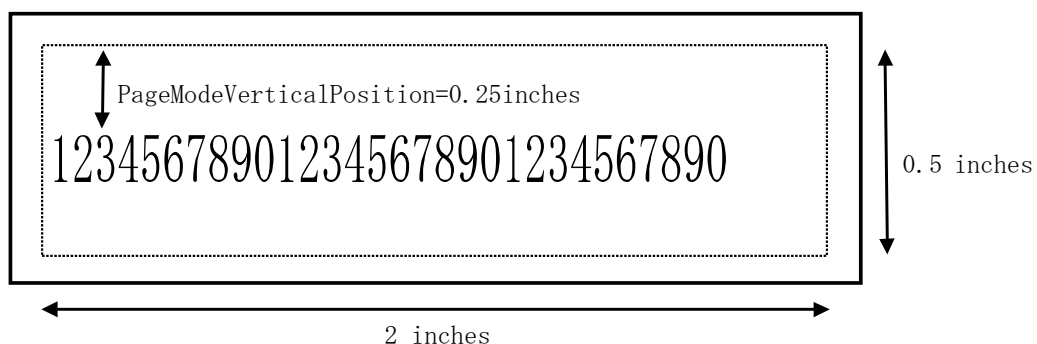
## PageModeVerticalPosition Property R/W

Type	<b>int</b>
Description	<p>Gets or sets the vertical start position offset within the print area for the print station specified by <b>PageModeStation</b>.</p> <p>The property is expressed in the unit indicated by <b>MapMode</b>.</p> <p>The vertical direction is perpendicular to the direction specified in the actual <b>PageModePrintDirection</b>. A read/get on this property will return the vertical position offset set by the last write/set and not the current position.</p> <p>Specify <i>PrinterStation.Receipt</i> for <b>PageModeStation</b> before accessing this property, otherwise the value 0 is returned.</p>

The following code sample shows the usage of **PageModeVerticalPosition**.

```
myptr.MapMode=MapMode.English;
myptr.PageModeStation=PrinterStation.Receipt;
myptr.PageModePrint(PageModePrintControl.PageMode);
// Set print area to 2 inches by 0.5 inches
myptr.PageModePrintArea=new System.Drawing.Point(0,0,2000,500);
myptr.PageModePrintDirection=PageModePrintDirection.LeftToRight;
myptr.PageModeVerticalPosition=250;
myptr.PrintNormal(PrinterStation.Receipt,"123456789012345678901234567890\n");
myptr.PageModePrint(PageModePrintControl.Normal);
```

The code sample above will generate the following receipt.



This property is initialized to 0 by **Open**.

## RecBarcodeRotationList Property

Type **Rotation[]**

Description Gets a list of the directions in which a receipt barcode can be rotated.  
The following table shows the valid property values.

Value	Meaning
<i>Rotation.Left90</i>	Barcode may be printed in a rotated 90° to the left.
<i>Rotation.Normal</i>	Barcode may be printed in the normal orientation.
<i>Rotation.Right90</i>	Barcode may be printed in a rotated 90° to the right.
<i>Rotation.Rotate180</i>	Barcode may be rotated 180° upside down.

This property is initialized to {*Rotation.Normal*, *Rotation.Right90*, *Rotation.Left90*, *Rotation.Rotate180*} by **Open**.

## RecBitmapRotationList Property

Type **Rotation[]**

Description Gets a list of the directions in which a receipt bitmap can be rotated.  
The following table shows the valid property values.

Value	Meaning
<i>Rotation.Left90</i>	Bitmap may be printed in a rotated 90° to the left.
<i>Rotation.Normal</i>	Bitmap may be printed in the normal orientation.
<i>Rotation.Right90</i>	Bitmap may be printed in a rotated 90° to the right.
<i>Rotation.Rotate180</i>	Bitmap may be rotated 180° upside down.

This property is initialized to {*Rotation.Normal*, *Rotation.Right90*, *Rotation.Left90*, *Rotation.Rotate180*} by **Open**.

## RecCartridgeState Property

Type **PrinterCartridgeStates**

Description Gets the status of the selected receipt cartridge.  
The following table shows the valid property values.

Value	Meaning
<i>PrinterCartridgeStates.Unknown</i>	Device does not support cartridge state reporting.

This property is initialized to *PrinterCartridgeStates.Unknown* when the device is first enabled following the **Open** call.

## RecCurrentCartridge Property R/W

Type **PrinterColors**

Description Specifies the currently selected receipt cartridge.  
The following table shows the valid property values.

Value	Meaning
<i>PrinterColors.Primary</i>	Supports primary color.

This property cannot be rewritten.

This property is initialized to *PrinterColors.Primary* when the device is first enabled following the **Open** call.

## RecEmpty Property

Type **bool**

Description Gets a Boolean value that indicates whether the receipt is out of paper.  
The following table shows the valid property values.

Value	Meaning
<i>false</i>	The receipt paper is present.
<i>true</i>	The receipt is out of paper.

This property is initialized and kept current while the device is enabled.

When **CoverOpen** is *true*, **RecEmpty** is not updated.

## RecLetterQuality Property R/W

Type **bool**

Description Gets a Boolean value that indicates whether the printer prints in high-quality mode.  
The following table shows the valid property values.

Value	Meaning
<i>false</i>	Prints in high-speed mode.
<i>true</i>	Prints in high-quality mode.

If [Print Speed] in the configuration program is [RecLetterQuality Valid], this property is enabled and determines the print speed of the printer according to print mode. For print speed settings in the configuration program other than that, see "MP-B30 SERIES Thermal Printer USER'S GUIDE".

This property is initialized to *false* when the device is enabled.

## RecLineChars Property R/W

Type **int**

Description Gets or sets the number of characters that the application wants to print on a receipt line. This property is set to one of the values which **RecLineCharsList** has. Depending on this property, the printer prints in the following font.

When **RecLineWidth** is 576:

Value	RecLineChars	Print Font (H × W)	Character Space	RecLineHeight
1 to 36	36	24 dots × 12 dots	4 dots	24
37 to 41	41		2 dots	
42 to 44	44		1 dot	
45 to 48	48		0 dots	
49 to 57	57	16 dots × 8 dots	2 dots	16
58 to 64	64		1 dot	
65 to 72	72		0 dots	

If the setting value is not supported, then an error is returned.

Setting **RecLineChars** may also update **RecLineHeight**, **RecLineSpacing**, **RecSidewaysMaxChars**, and **RecSidewaysMaxLines**.

The default of this property can be changed at the setting in the configuration program. This property is initialized to the value set in [Number of Characters per Line] of the configuration program when the device is enabled.

## RecLineCharsList Property

Type **int[]**

Description Gets a collection of the line widths (characters per line) supported by the receipt station.

RecLineWidth	Value
576	36,41,44,48,57,64,72

This property is initialized to {36,41,44,48,57,64,72} by **Open**.

## RecLineHeight Property R/W

Type            **int**

Description    Gets or sets the receipt print line height.  
The property is expressed in the unit indicated by **MapMode**.  
This property cannot be rewritten. The value is automatically set by **RecLineChars**.

When **RecLineChars** is changed, **RecLineHeight** is updated to the height of the print font that corresponds to the set **RecLineChars**. For the relation between **RecLineChars** and the print font, see **RecLineChars**.

The relation between **RecLineHeight** and the print font is as follows.  
The values in the following table are at the time of **MapMode** is *MapMode.Dots*.

Print Font (H × W)	RecLineHeight
Font A (24 × 12 dots)	24
Font B (16 × 8 dots)	16

This property is initialized to one value in the above values by the value set at the [Number of Characters per Line] of the configuration program when the device is enabled.

## RecLineSpacing Property R/W

Type            **int**

Description    Gets or sets the spacing of each single-high print line. This includes both the printed line height and the whitespace between each pair of lines.  
The property is expressed in the unit indicated by **MapMode**.  
The values in the following table are at the time of **MapMode** is *MapMode.Dots*.

The relation between **RecLineHeight** and **RecLineCpacing** is as follows.  
When a value out of the range is specified, *ErrorCode.Illegal* is thrown and the property is not set.

RecLineHeight	RecLineCpacing	Default
24	24 to 255	30
16	16 to 255	

The default of this property can be changed at the setting in the configuration program.  
This property is initialized to the value of line spacing set in [Line Spacing (dots)] of the configuration program, when the device is enabled.

## RecLinesToPaperCut Property

Type            **int**

Description    Gets the number of lines that must be advanced before cutting the receipt paper.  
The value obtained by dividing the distance between the print head and the tear bar of the printer by line spacing indicated by **RecLineSpacing** is set. Therefore, this property changes when **RecLineSpacing** is changed.

[Calculation formula]

**RecLinesToPaperCut** = 48 / **RecLineSpacing**

Example:

When **RecLineSpacing** is 31 (**MapMode**=*MapMode.Dots*)

**RecLinesToPaperCut** = 48 / 31 = 1.548... = 2

(The decimal part is rounded up.)

This property is initialized to the value based on the above calculation using [Line Spacing (dots)] of the configuration program by **Open**.

## RecLineWidth Property

Type            **int**

Description    Gets the width of a line for the number of characters indicated by **RecLineChars**.  
The property is expressed in the unit indicated by **MapMode**. The values in the following table are at the time of **MapMode** is *MapMode.Dots*.

Value
576

This property is initialized to 576 when the device is enabled.

## RecNearEnd Property

Type            **bool**

Description    Gets a Boolean value that indicates whether the receipt paper is low.  
This property always shows *false*.

## RecSidewaysMaxChars Property

Type            **int**

Description    Gets the maximum number of one-byte characters that may be printed on each line in sideways mode (rotated 90° to the left or right).

[Calculation formula]

**RecSidewaysMaxChars**

= Maximum height of **PageModeArea** / (Print font width / 2 + Character space)

Example:

When **PageModeArea** is "576,2400", and **RecLineChars** is 48

**RecSidewaysMaxChars** =  $2400 / ((24 / 2) + 0) = 200$

(The decimal part is rounded down.)

This property is initialized to the value based on the above calculation depending on **PageModeArea** and the print font, when the device is first enabled following **Open**.

## RecSidewaysMaxLines Property

Type            **int**

Description    Gets the maximum number of lines that can be printed in sideways mode (rotated 90° to the left or right).

[Calculation formula]

**RecSidewaysMaxLines**

=  $(\text{RecLineWidth} - \text{RecLineHeight}) / \text{RecLineSpacing} + 1$

Example:

When **RecLineWidth** is 576, **RecLineHeight** is 24, and **RecLineSpacing** is 30

**RecSidewaysMaxLines** =  $(576 - 24) / 30 + 1 = 19$

(The decimal part is rounded down.)

This property is initialized to the value based on the above calculation depending on **RecLineWidth**, **RecLineSpacing**, and **RecLineHeight**, when the device is first enabled following **Open**.

## RotateSpecial Property R/W

Type            **Rotation**

Description    Gets or sets the rotation orientation for barcodes.  
The following table shows the valid property values.  
If *rotation* contains *PrintRotation.Barcode* in **RotatePrint**, the rotating direction of the *rotation* is selected.

Value	Meaning
<i>Rotation.Left90</i>	Specifies rotated 90° left (counter-clockwise).
<i>Rotation.Normal</i>	Prints the next barcode in the normal print direction.
<i>Rotation.Right90</i>	Specifies rotated 90° right (clockwise).
<i>Rotation.Rotate180</i>	Specifies rotated 180° printing (upside down printing).

This property is initialized to *Rotation.Normal* by **Open**.



#### 4.1.5 Common Methods

This section describes the details of the common methods for PosPrinter.  
For details of the thrown exception errors, see "Appendix A Exceptions".

##### CheckHealth Method

Syntax `string CheckHealth(HealthCheckLevel level);`

Parameter	Meaning
<i>level</i>	Specifies the type of health check to be executed on the device.

· Values of *level*

Value	Meaning
<i>HealthCheckLevel.External</i>	Executes a complete test using the device. ROM version ID of the printer, <b>ServiceObjectVersion</b> , and <b>DeviceName</b> are printed on the printer.
<i>HealthCheckLevel.Interactive</i>	Executes an interactive test of the device. Displays a modal dialog box to execute a complete test using the device and display results.
<i>HealthCheckLevel.Internal</i>	Execute a health check without using the device physically.

Description Tests the status of the device.  
A text description of the results of this method is placed in **CheckHealthText**.  
**CheckHealth** is always executed synchronously.

##### Claim Method

Syntax `void Claim(int timeout);`

Parameter	Meaning
<i>timeout</i>	Specifies the maximum waiting time (in milliseconds) for exclusive access. If it is 0, the method returns the result immediately even if exclusive access of the device cannot be obtained. If <i>WaitForever</i> (-1) is set, the method waits until exclusive access is obtained.

Description Requests exclusive access to the device.  
The PosPrinter device cannot be used until the exclusive access is obtained.  
When it is successful, **Claimed** is set to *true*.  
When the power is OFF or the cable is not connected, **Claim** is not available.

## ClearOutput Method

Syntax      **void ClearOutput();**

Description      Clears all the buffered device outputs.  
Any output error events that were queued (usually waiting for **FreezeEvents** to be set to *false*) are also cleared.

## Close Method

Syntax      **void Close();**

Description      Releases the device and its resources.  
If **DeviceEnabled** is *true*, the device is first disabled.  
If **Claimed** is *true*, exclusive access to the device is first released.  
Do not execute this while the event is in progress (or in the event handler).

## CompareFirmwareVersion Method

This method is not supported. For the thrown exception errors, see "Appendix A Exceptions".

Syntax      **CompareFirmwareResult CompareFirmwareVersion(string firmwareFileName);**

## DirectIO Method

Syntax      **DirectIOData DirectIO(int command, int data, object obj);**

Parameter	Meaning
<i>command</i>	Command number. Specific values assigned by the Service Object.
<i>data</i>	Additional numeric data. Specific values vary by command number and Service Object.
<i>obj</i>	Additional data provided by the Service Object. The value varies depending on the command number and what the Service Object sends.

Description      The following functions are supported:  
· Remaining memory capacity response  
· International character selection  
· Status response  
**DirectIO** is always executed synchronously.

- **Remaining memory capacity response**

Issues the printer command "Send NV Graphics Memory Remaining Capacity" and returns its response as a numeric value.

The response data is stored in *DirectIOData.data*.

Parameter	Description
<i>command</i>	3
<i>data</i>	<i>null</i>
<i>obj</i>	<i>null</i>

- **International character selection**

Selects the international character.

To change the international character, select the international character with this method after setting **CharacterSet**.

When changing **CharacterSet** to 932 after changing the international character, the international character is set to Japan.

When changing **CharacterSet** to other than 932 after changing the international character, the international character is set to USA.

Parameter	Description
<i>command</i>	201
<i>data</i>	International character number $n$ $0 \leq n \leq 12$ Country names available for $n$ are as follows: 0: USA 1: France 2: Germany 3: United Kingdom 4: Denmark I 5: Sweden 6: Italy 7: Spain I 8: Japan 9: Norway 10: Denmark II 11: Spain II 12: Latin America
<i>obj</i>	<i>null</i>

- **Status response**

Returns the "out-of-paper" sensor status in a numeric value.

Parameter	Description
<i>command</i>	501
<i>data</i>	1
<i>obj</i>	<i>null</i>

The response data is stored in *DirectIOData.data*.

Value	State
0	Paper is ready
1	No paper

## Open Method

Syntax      **void Open();**

Description      Opens the device.  
When **Open** is successful, the common properties and other class-specific properties are initialized.

## Release Method

Syntax      **void Release();**

Description      Releases exclusive access to the device.  
If **DeviceEnabled** is *true*, and the device is an exclusive-use device, then the device is first disabled.  
Do not execute this while the event is in progress (or in the event handler).

## ResetStatistic(string) Method

Syntax      **void ResetStatistic(string statistic);**

Description      Resets the specified statistics to 0.  
For the statistics that can be reset, see "Appendix B Statistics".  
**ResetStatistic** is always executed synchronously.

## ResetStatistics() Method

Syntax      **void ResetStatistics();**

Description      Resets all the statistics to 0.  
For the statistics that can be reset, see "Appendix B Statistics".  
**ResetStatistics** is always executed synchronously.

## ResetStatistics(StatisticCategories) Method

Syntax	<b>void ResetStatistics(StatisticCategories <i>statistics</i>);</b>
Description	Resets all the statistics of the specified category to 0. For the statistics that can be reset, see "Appendix B Statistics". <b>ResetStatistics</b> is always executed synchronously.

## ResetStatistics(string[]) Method

Syntax	<b>void ResetStatistics(string[] <i>statistics</i>);</b>
Description	Resets the specified statistics to 0. For the statistics that can be reset, see "Appendix B Statistics". <b>ResetStatistics</b> is always executed synchronously.

## RetrieveStatistic(string) Method

Syntax	<b>string RetrieveStatistic(string <i>statistic</i>);</b>
Description	Retrieves the specified device statistics. For <i>statistic</i> , specify the statistics to retrieve. When it is successful, <b>RetrieveStatistic</b> returns the XML string of the statistics. For the statistics that are retrieved, see "Appendix B Statistics". <b>RetrieveStatistic</b> is always executed synchronously.

## RetrieveStatistics() Method

Syntax	<b>string RetrieveStatistics();</b>
Description	Retrieves all the device statistics. When it is successful, <b>RetrieveStatistics</b> returns the XML string of the statistics. For the statistics that are retrieved, see "Appendix B Statistics". <b>RetrieveStatistics</b> is always executed synchronously.

## RetrieveStatistics(StatisticCategories) Method

Syntax	<b>string RetrieveStatistics(StatisticCategories <i>statistics</i>);</b>
Description	Retrieves the statistics of the specified category. <i>statistics</i> stores the category of the statistics to be retrieved by the application. When it is successful, <b>RetrieveStatistics</b> returns the XML string of the statistics. For the statistics that are retrieved, see "Appendix B Statistics". <b>RetrieveStatistics</b> is always executed synchronously.

## RetrieveStatistics(string[]) Method

- Syntax**      **string** RetrieveStatistics(**string**[] *statistics*);
- Description**      Retrieves the specified device statistics.  
For *statistic*, specify the statistics to retrieve.  
When it is successful, **RetrieveStatistics** returns the XML string of the statistics.  
For the statistics that are retrieved, see "Appendix B Statistics".  
**RetrieveStatistics** is always executed synchronously.

## UpdateFirmware Method

This method is not supported. For the thrown exception errors, see "Appendix A Exceptions".

- Syntax**      **void** UpdateFirmware(**string** *firmwareFileName*);

## UpdateStatistic Method

This method is not supported. For the thrown exception errors, see "Appendix A Exceptions".

- Syntax**      **void** UpdateStatistic(**string** *name*, **object** *value*);

## UpdateStatistics(Statistic[]) Method

This method is not supported. For the thrown exception errors, see "Appendix A Exceptions".

- Syntax**      **void** UpdateStatistics(**Statistic**[] *statistics*);

## UpdateStatistics(StatisticCategories, Object) Method

This method is not supported. For the thrown exception errors, see "Appendix A Exceptions".

- Syntax**      **void** UpdateStatistics(**StatisticCategories** *statistics*, **object** *value*);

#### 4.1.6 Specific Methods

This section describes the details of the specific methods for PosPrinter.

For exception errors of specific methods that are not supported, see "Appendix A Exceptions".

##### ClearPrintArea Method

Syntax      **void ClearPrintArea();**

Description      Clears the print data on the Page Mode print area defined by **PageModePrintArea**. To clear the entire Page Mode area, specify **PageModePrintArea** to the area equivalent to the one indicated by **PageModeArea**, and then call **ClearPrintArea**. Specify *PrinterStation.Receipt* for **PageModeStation** before calling this method.

##### MarkFeed Method

Syntax      **MarkFeed(MarkFeedType type);**

Parameter	Meaning
<i>type</i>	Indicates the control type of paper indicated mark.

·Values of *type*

Value	Meaning
<i>PrinterMarkFeeds.Takeup</i>	After detecting the mark, feeds the paper to the paper take-up position. The paper feed length is the length of the memory switches MS 21 to 22 (Mark Position Correction) of the printer. The default of the paper feed is 160 dots (20 mm).
<i>PrinterMarkFeeds.Cutter</i>	After detecting the mark, feeds the paper to the cutting position. (Feeds the paper to the same position as <i>PrinterMarkFeeds.Takeup</i> ).

Description      This method is used to utilize the paper that is available for the mark detection.

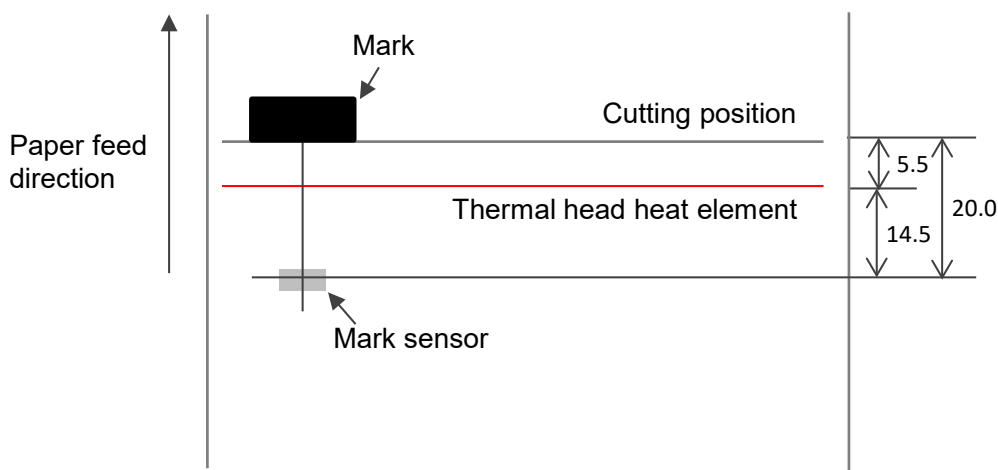
This method is executed synchronously if **AsyncMode** is *false*, and asynchronously if **AsyncMode** is *true*.

This method is available when the setting of [Mark Mode] in the configuration program is set to Enable.

See "MP-B30 SERIES Thermal Printer USER'S GUIDE" for the details of the memory switch of the printer.

The memory switches of the printer can be changed in the "SII Printer Setting Utility for Windows" for MP-B30 series that is to be expanded at the installation of "SII Printer Driver for Windows" for MP-B30 series.

The relation between the sensor position and the defaults of memory switches MS 21 to 22 (Mark Position Correction) of the printer is shown in the following figure.



Unit : mm

**Notes** The paper feed is not performed when this method is executed at the form feed position of the mark sensed paper.

## PageModePrint Method

**Syntax** `PageModePrint(PageModePrintControl control);`

Parameter	Meaning
<i>control</i>	Specifies the control type of Page Mode.

•Values of *control*

Value	Meaning
<i>PageModePrintControl.Cancel</i>	Clears the page and exits Page Mode without any printing of any print area.
<i>PageModePrintControl.Normal</i>	Prints the print area and destroys the canvas and exits Page Mode
<i>PageModePrintControl.PageMode</i>	Enters Page Mode.
<i>PageModePrintControl.PrintSave</i>	Prints the print data of the Page Mode print area and saves the data. This is used for repeated printings.

**Description** Enters or exits Page Mode for the station specified in **PageModeStation**.

If *PageModePrintControl.PageMode* is specified for *control*, then Page Mode is started. Subsequently, the print data can be buffered with **PrintNormal**, **PrintBarCode**, **PrintBitmap**, or **PrintMemoryBitmap** until **PageModePrint** is called by specifying *PageModePrintControl.PrintSave*, *PageModePrintControl.Normal*, or *PageModePrintControl.Cancel*. (Methods called during this time only buffer the print data and they do not initiate printing. Also, the setting of **AsyncMode** does not affect the Page Mode function: No **OutputId** will be assigned, nor will **OutputCompleteEvent** be notified for each operation.)

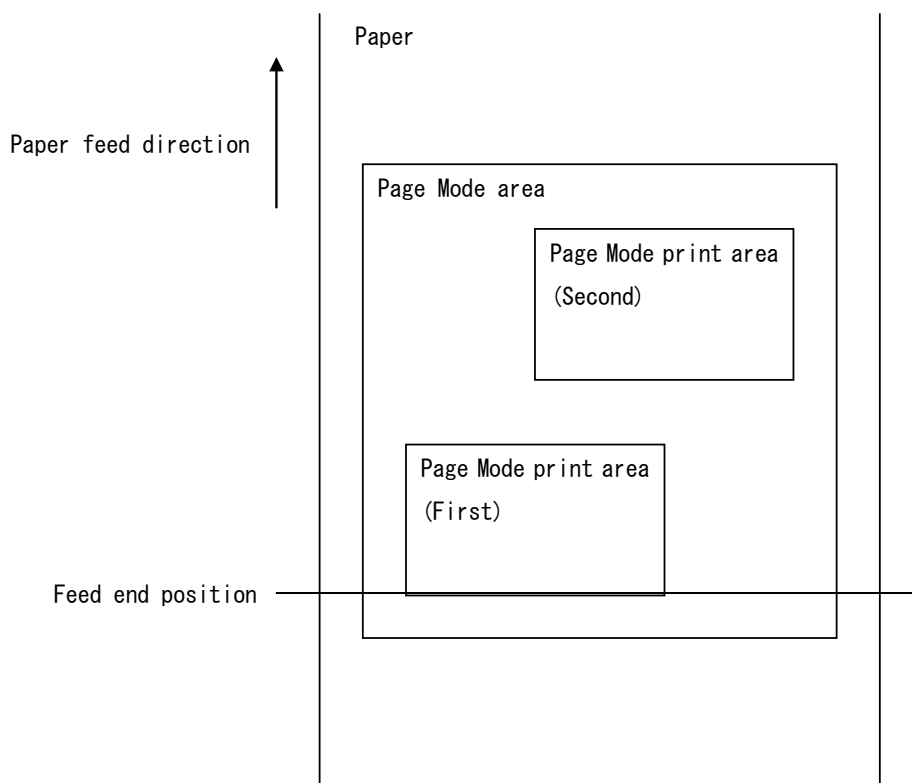


If *PageModePrintControl.PrintSave* is specified for *control*, then Page Mode is continued. If some print data is buffered by one of **PrintNormal**, **PrintBarCode**, **PrintBitmap**, and **PrintMemoryBitmap**, then the data is saved and printed. This control is used to print the same page layout with additional print items inside of the page.

If *PageModePrintControl.Normal* is specified for *control*, then Page Mode is exited to return to the normal state. If some print data is buffered by one of **PrintNormal**, **PrintBarCode**, **PrintBitmap**, and **PrintMemoryBitmap**, then the data is printed. The buffered data will not be saved.

If *PageModePrintControl.Cancel* is specified for *control*, then Page Mode is exited to return to the normal state. If some print data is buffered by one of **PrintNormal**, **PrintBarCode**, **PrintBitmap**, and **PrintMemoryBitmap**, then the data is not printed or saved.

When **PageModePrint** is called while *PageModePrintControl.Normal* or *PageModePrintControl.PrintSave* is specified for *control*, all of the print data on the Page Mode print area defined by **PageModePrintArea** will be printed, and the paper is fed to the end of the area. If more than one Page Mode print area is defined, then after **PageModePrint** is called, all of the data that is to be printed in the respective Page Mode print area(s) will be printed, and the paper will be fed to the end of the Page Mode print area located the farthest "down" the sheet of paper. (See the figure below.)



This method is executed asynchronously if **AsyncMode** is *true*, or synchronously if **AsyncMode** is *false*.

Calling **ClearOutput** cancels Page Mode to return to the normal state. The buffered print data is also cleared. Page Mode can be used within a transaction print, but not within a rotate print.

Specify *PrinterStation.Receipt* for **PageModeStation** before calling this method.

## PrintBarcode Method

Syntax

```
void PrintBarcode(PrinterStation station,
                 string data,
                 BarCodeSymbology symbology,
                 int height,
                 int width,
                 int alignment,
                 BarCodeTextPosition textPosition);
```

Parameter	Meaning
<i>station</i>	Specifies the station to be used. Specify <i>PrinterStation.Receipt</i> .
<i>data</i>	Specifies the string of the barcode.
<i>symbology</i>	Specifies the barcode type to be used. See values below.
<i>height</i>	Specifies the height of the barcode. Expressed in the unit given by <b>MapMode</b> . When <b>MapMode</b> is <i>MapMode.Dots</i> , specify the height from 1 to 255. <i>height</i> of the following barcodes are ignored and set automatically by <i>width</i> , Specify from 1 to 255. <ul style="list-style-type: none"> <li>• QR Code</li> <li>• GS1 Databar Omni-directional</li> <li>• GS1 Databar Expanded</li> <li>• GS1 Databar Expanded Stacked</li> <li>• GS1 Databar Limited</li> <li>• GS1 Databar Truncated</li> </ul> <i>height</i> of the following barcodes are ignored and set with a fixed value. Specify from 1 to 255. <ul style="list-style-type: none"> <li>• GS1 Databar Stacked Omni-directional</li> <li>• GS1 Databar Stacked</li> </ul> During Page Mode by <b>PageModePrint</b> , specify the value within the range of print area specified by <b>PageModePrintArea</b> and <b>PageModeVerticalPosition</b> .
<i>width</i>	Specifies the width of the barcode. Expressed in the unit given by <b>MapMode</b> . The width of the barcode actually printed is the best fit within the width specified by <i>width</i> . When <b>MapMode</b> is <i>MapMode.Dots</i> , specify the width from 1 to <b>RecLineWidth</b> for upright position. When rotating the barcode 90° right/left by <b>RotateSpecial</b> or <b>RotatePrint</b> , specify in the range not exceeding the maximum value of the printer. (See <b>PageModeArea</b> for the maximum value of the printer.) During Page Mode by <b>PageModePrint</b> , specify the value within the range of print area specified by <b>PageModePrintArea</b> and <b>PageModeHorizontalPosition</b> .
<i>alignment</i>	Specifies the position of the barcode. See values below.
<i>textPosition</i>	Specifies the position of the text printed in the barcode. See values below.

•Values of *symbology*

Value	Meaning
<i>BarCodeSymbology.Codabar</i>	Codabar (NW-7)
<i>BarCodeSymbology.Code128</i>	Code128
<i>BarCodeSymbology.Code128Parsed</i>	Code128 Parsed
<i>BarCodeSymbology.Code39</i>	Code39
<i>BarCodeSymbology.Code93</i>	Code93
<i>BarCodeSymbology.Ean13S</i>	EAN13 (JAN13) with supplemental barcode
<i>BarCodeSymbology.EanJan13</i>	EAN13 (JAN13)
<i>BarCodeSymbology.EanJan8</i>	EAN8 (JAN8)
<i>BarCodeSymbology.Gs1DataBar</i>	GS1 Databar Omni-directional
<i>BarCodeSymbology.Gs1DataBarExpanded</i>	GS1 Databar Expanded
<i>BarCodeSymbology.Gs1DataBarStackedOmniDirectional</i> or <i>BarCodeSymbology.Other + 8</i>	GS1 Databar Stacked Omni-directional
<i>BarCodeSymbology.Gs1DataBarExpandedStacked</i> or <i>BarCodeSymbology.Other + 9</i>	GS1 Databar Expanded Stacked
<i>BarCodeSymbology.Itf</i>	Interleaved 2 of 5
<i>BarCodeSymbology.Other + 5</i>	QR Code (Mixed mode)
<i>BarCodeSymbology.Other + 6</i>	GS1 Databar Limited
<i>BarCodeSymbology.Other + 7</i>	GS1 Databar Stacked
<i>BarCodeSymbology.Other + 10</i>	GS1 Databar Truncated
<i>BarCodeSymbology.Pdf417</i>	PDF417
<i>BarCodeSymbology.Upca</i>	UPC-A
<i>BarCodeSymbology.Upce</i>	UPC-E

•Values of *alignment*

Value	Meaning
<i>PrinterBarCodeCenter</i>	Printed with center.
<i>PrinterBarCodeLeft</i>	Printed with left justify.
<i>PrinterBarCodeRight</i>	Printed with right justify.
Other values	Printed with the left margin of the specified value. Expressed in the unit given by <b>MapMode</b> .

When rotation 90° right/left is specified by **RotateSpecial**, **RotatePrint**, and during Page Mode by **PageModePrint**, the setting of *alignment* is invalid and the data is always printed with left justify.

•Values of *textPosition*

Value	Meaning
<i>BarCodeTextPosition.Above</i>	Prints the text above the barcode.
<i>BarCodeTextPosition.Below</i>	Prints the text below the barcode.
<i>BarCodeTextPosition.None</i>	Does not print the text.

Call this method when printing the barcode on the specified printer.

This method is executed synchronously if **AsyncMode** is *false*, and asynchronously if **AsyncMode** is *true*.

If **RotateSpecial** indicates that the barcode is rotated, the barcode is printed in rotated mode. *height*, *width*, and *textPosition* are applied to the barcode before it is rotated.

For example, when specifying *Rotation.Left90* for **RotateSpecial** and calling this method specifying *BarCodeTextPosition.Below* for *textPosition* of this method, the text is placed below the barcode, and then the text and barcode are rotated 90° to the left and printed.

The barcode quiet zone is not secured. Verify that the barcode can be read with your actual device beforehand.

However, the following barcode quiet zones are unnecessary or secured:

- GS1 Databar Omni-directional
- GS1 Databar Expanded
- GS1 Databar Stacked Omni-directional
- GS1 Databar Expanded Stacked
- GS1 Databar Limited
- GS1 Databar Stacked
- GS1 Databar Truncated

The limitations for each barcode are described below.

[Codabar (NW-7)]

Parameter	Limitation
<i>data</i>	The head and end of line must be one of 'A' to 'D'. Other data must be at least one of '0' to '9', '\$', '+', ':', '-', '.', and '/'.
<i>width</i>	When <b>MapMode</b> = <i>MapMode.Dots</i> : $width = (((6 \times X) + (2 \times X \times N)) \times D) + ((X \times N - X) \times D') + (-1 \times X)$ $20 \times D + 2 \times D' - 2 \leq width \leq 72 \times D + 12 \times D' - 6$ D: the number of barcode characters D': the number of data characters (the number of 'A' to 'D', '+', ':', '/', '.' included in barcode data) X: fine element width $2 \leq X \leq 6$ N: ratio of wide element width to fine element width (Set to 2, 2.5, or 3) X and N are automatically set according to <i>width</i> .

[Code128]

Parameter	Limitation
<i>data</i>	Specify any value consisting of decimal numbers from 0 to 105. Each numeric value is treated as the corresponding character shown in the table below. The first letter must be a decimal number 103, 104, or 105, and the barcode data of at least one letter must follow it.
<i>width</i>	When <b>MapMode</b> = <i>MapMode.Dots</i> : $width = X \times (((D + 2) \times 11) + 2)$ $22 \times D + 48 \leq width \leq 66 \times D + 144$ D: the number of barcode characters (including start code) X: fine element width $2 \leq X \leq 6$ X is automatically set according to <i>width</i> .

· Character set of Code128

Number	Character			Number	Character		
	Code A	Code B	Code C		Code A	Code B	Code C
0	SPACE <sup>*1</sup>	SPACE <sup>*1</sup>	00	53	U	U	53
1	!	!	01	54	V	V	54
2	"	"	02	55	W	W	55
3	#	#	03	56	X	X	56
4	\$	\$	04	57	Y	Y	57
5	%	%	05	58	Z	Z	58
6	&	&	06	59	[	[	59
7	'	'	07	60	\	\	60
8	(	(	08	61	]	]	61
9	)	)	09	62	^	^	62
10	*	*	10	63	_	_	63
11	+	+	11	64	NULL	`	64
12	,	,	12	65	SOH	a	65
13	-	-	13	66	STX	b	66
14	.	.	14	67	ETX	c	67
15	/	/	15	68	EOT	d	68
16	0	0	16	69	ENG	e	69
17	1	1	17	70	ACK	f	70
18	2	2	18	71	BEL	g	71
19	3	3	19	72	BS	h	72
20	4	4	20	73	HT	i	73
21	5	5	21	74	LF	j	74
22	6	6	22	75	VT	k	75

Number	Character			Number	Character		
	Code A	Code B	Code C		Code A	Code B	Code C
23	7	7	23	76	FF	l	76
24	8	8	24	77	CR	m	77
25	9	9	25	78	SO	n	78
26	:	:	26	79	SI	o	79
27	;	;	27	80	DLE	p	80
28	<	<	28	81	DC1	q	81
29	=	=	29	82	DC2	r	82
30	>	>	30	83	DC3	s	83
31	?	?	31	84	DC4	t	84
32	@	@	32	85	NAK	u	85
33	A	A	33	86	SYN	v	86
34	B	B	34	87	ETB	w	87
35	C	C	35	88	CAN	x	88
36	D	D	36	89	EM	y	89
37	E	E	37	90	SUB	z	90
38	F	F	38	91	ESC	{	91
39	G	G	39	92	FS		92
40	H	H	40	93	GS	}	93
41	I	I	41	94	RS	~	94
42	J	J	42	95	US	DEL	95
43	K	K	43	96	FNC3	FNC3	96
44	L	L	44	97	FNC2	FNC2	97
45	M	M	45	98	SHIFT	SHIFT	98
46	N	N	46	99	CODE C	CODE C	99
47	O	O	47	100	CODE B	FNC4	CODE B
48	P	P	48	101	FNC4	CODE A	CODE A
49	Q	Q	49	102	FNC1	FNC1	FNC1
50	R	R	50	103	START(CODE A)		
51	S	S	51	104	START(CODE B)		
52	T	T	52	105	START(CODE C)		

\*1: Input a space.

[Code128 Parsed]

Parameter	Limitation
<i>data</i>	<p>The head of line must be the special code (CODE A, CODE B, or CODE C) for the code set to use, and the barcode data of at least one letter must follow it. See "Code128 Special Code Table" for the special code. See "Input example of <i>data</i>" for input of <i>data</i>. The effective range of <i>data</i> differs by code set.</p> <ul style="list-style-type: none"> <li>· Code A : 0x00 to 0x5f, FNC1, FNC2, FNC3, FNC4, SHIFT, CODE B, CODE C</li> <li>· Code B : 0x20 to 0x7f, FNC1, FNC2, FNC3, FNC4, SHIFT, CODE A, CODE C</li> <li>· Code C : 0x30 to 0x39, FNC1, CODE A, CODE B</li> </ul>
<i>width</i>	<p>When <b>MapMode</b> = <i>MapMode.Dots</i>:  <math>width = X \times (((D + 2) \times 11) + 2)</math>  <math>1^{*1} \leq width \leq 66 \times D + 144</math>  D: the number of barcode characters (including start code)  X: fine element width  <math>2 \leq X \leq 6</math>  X is automatically set according to <i>width</i>.</p>

\*1: When  $1$  to  $22 \times D + 48$  is set, *width* is set to  $22 \times D + 48$ .

#### · Code128 Special Code Table

<i>data</i>	Special Code
"{S"	SHIFT
"{A"	CODE A
"{B"	CODE B
"{C"	CODE C
"{1"	FNC1
"{2"	FNC2
"{3"	FNC3
"{4"	FNC4
"{"	'{'

#### Input example of *data*

*data* is comprised of ASCII characters, which the service maps to the corresponding value for the selected code set. In Code A and Code B, this will be a one to one mapping. In Code C, each pair of digits is converted to a single Code C data character in the range 0x00 through 0x63. (If the Code C data contains an odd number of digits, then a leading 0 digit is added by the service before conversion.) A sentinel character, the left curly bracket "{", followed by a certain value, is used to indicate a special character.

When creating a barcode of the barcode character "0123", the input of *data* is as follows according to the code set selected.

Selecting Code A : *data="{A0123"*  
Selecting Code B : *data="{B0123"*  
Selecting Code C : *data="{C0123"* or *data="{C123"*

[Code39]

Parameter	Limitation
<i>data</i>	At least one of '0' to '9', 'A' to 'Z', ' ', '\$', '%', '+', '-', '!', '/' must be specified.
<i>width</i>	When <b>MapMode</b> = <i>MapMode.Dots</i> : $width = (((X \times 7) + (X \times N \times 3)) \times (D + 2)) + (-1 \times X)$ $26 \times D + 50 \leq width \leq 96 \times D + 186$ D: the number of barcode characters X: fine element width $2 \leq X \leq 6$ N: ratio of wide element width to fine element width (Set to 2, 2.5, or 3) X and N are automatically set according to <i>width</i> .

[Code93]

Parameter	Limitation
<i>data</i>	Specify any value consisting of decimal numbers from 0 to 46. Each numeric value is treated as the corresponding character shown in the table below.
<i>width</i>	When <b>MapMode</b> = <i>MapMode.Dots</i> : $width = X \times ((D + 2 + 2) \times 9) + 1$ $18 \times D + 74 \leq width \leq 54 \times D + 222$ D: the number of barcode characters X: fine element width $2 \leq X \leq 6$ X is automatically set according to <i>width</i> .

· Character set of Code93

Number	Character	Number	Character	Number	Character	Number	Character
0	0	12	C	24	O	36	-
1	1	13	D	25	P	37	.
2	2	14	E	26	Q	38	SPACE*1
3	3	15	F	27	R	39	\$
4	4	16	G	28	S	40	/
5	5	17	H	29	T	41	+
6	6	18	I	30	U	42	%
7	7	19	J	31	V	43	(\$)
8	8	20	K	32	W	44	(%)



Number	Character	Number	Character	Number	Character	Number	Character
9	9	21	L	33	X	45	(/)
10	A	22	M	34	Y	46	(+)
11	B	23	N	35	Z		

\*1: Input a space.

[EAN13 (JAN13) with supplemental barcode]

Parameter	Limitation
<i>data</i>	Specify 14, 15, 17, or 18 letters consisting of '0' to '9'. When 15 letters or 18 letters are entered, the 13th character does not affect the printing data.
<i>width</i>	When <b>MapMode</b> = <i>MapMode.Dots</i> : <ul style="list-style-type: none"> <li>When 14 or 15 letters are specified  <math>width = 122 \times X</math>  <math>244 \leq width \leq 732</math> </li> <li>When 17 or 18 letters are specified  <math>width = 149 \times X</math>  <math>298 \leq width \leq 894</math> </li> </ul> X: fine element width $2 \leq X \leq 6$ X is automatically set according to <i>width</i> .

[EAN13 (JAN13)]

Parameter	Limitation
<i>data</i>	Specify 12 or 13 letters consisting of '0' to '9'. The 13th letter does not affect the barcode printing data.
<i>width</i>	When <b>MapMode</b> = <i>MapMode.Dots</i> : $width = 95 \times X$ $190 \leq width \leq 570$ X: fine element width $2 \leq X \leq 6$ X is automatically set according to <i>width</i> .

[EAN8 (JAN8)]

Parameter	Limitation
<i>data</i>	Specify 7 or 8 letters consisting of '0' to '9'. The 8th letter does not affect the barcode printing data.
<i>width</i>	When <b>MapMode</b> = <i>MapMode.Dots</i> : $width = 67 \times X$ $134 \leq width \leq 402$ X: fine element width $2 \leq X \leq 6$ X is automatically set according to <i>width</i> .

[GS1 Databar Omni-directional]

Parameter	Limitation
<i>data</i>	Specify 13 letters consisting of '0' to '9'.
<i>width</i>	When <b>MapMode</b> = <i>MapMode.Dots</i> : $width = 96 \times X$ $1^{*1} \leq width \leq 576$ X: module width $2 \leq X \leq 6$ X is automatically set according to <i>width</i> .

1\*: When 1 to 287 is set, *width* is set to 192.

[GS1 Databar Expanded]

Parameter	Limitation
<i>data</i>	Specify '0' to '9', 'A' to 'Z', 'a' to 'z', space, '!', '"', '%', '&', '(', ')', '*', '+', ',', '-', '.', '/', ':', ';', '<', '>', '=', '?', '_' as 2 or more letters. Input "{1" for FNC1. Be sure to input the check digit since it is not automatically calculated by the printer.
<i>width</i>	Input the value other than 0.

[GS1 Databar Stacked Omni-directional]

Parameter	Limitation
<i>data</i>	Specify 13 letters consisting of '0' to '9'.
<i>height</i>	Input the value other than 0. The value is set as <i>height</i> = 138 regardless of the input value.
<i>width</i>	When <b>MapMode</b> = <i>MapMode.Dots</i> : Input the value other than 0. The value is set as <i>width</i> = 100 regardless of the input value.

Module width is fixed at 2.

[GS1 Databar Expanded Stacked]

Parameter	Limitation
<i>data</i>	Two or more letters of '0' to '9', 'A' to 'Z', 'a' to 'z', space, '!', '"', '%', '&', '(', ')', '*', '+', ',', '-', '.', '/', ':', ';', '<', '>', '=', '?', '_' must be specified. Input "{1" for FNC1. Be sure to input the check digit since it is not automatically calculated by the printer.
<i>width</i>	Input the value other than 0.

Module width is fixed at 2.

[Interleaved 2 of 5]

Parameter	Limitation
<i>data</i>	Specify any value consisting of '0' to '9'. Note that the number of specified letters must be an even number except for 0.
<i>width</i>	When <b>MapMode</b> = <i>MapMode.Dots</i> : $width = ((D \times 2 + 1) \times X \times N) + ((D \times 3 + 6) \times X)$ $14 \times D + 16 \leq width \leq 54 \times D + 54$ D: the number of barcode characters X: fine element width $2 \leq X \leq 6$ N: ratio of wide element width to fine element width (Set to 2, 2.5, or 3) X and N are automatically set according to <i>width</i> .

[QR Code]

Parameter	Limitation
<i>data</i>	Specify characters of the following range: ASCII characters 8 bits Latin/Katakana characters based on JIS X 0201 Shift-JIS code based on JIS X 0208
<i>width</i>	$width = (4V + 17) \times M$ $42 \leq width$ V: version of QR Code (1 to 40) M: module size (2 to 16) For version, the smallest value that input data can be converted to barcode is selected. For module size, the maximum size that does not exceed <i>width</i> is selected after the version is determined.

QR Code model is fixed at 2 and the error correction level is fixed at M. Printing size is based on *width*, and *height* is ignored since QR Code is a square.

If data other than the printable characters is specified, *ErrorCode.Illegal* is thrown.

[GS1 Databar Limited]

Parameter	Limitation
<i>data</i>	Specify 13 letters consisting of '0' to '9'.
<i>width</i>	When <b>MapMode</b> = <i>MapMode.Dots</i> : The effective range of <i>height</i> depends on <i>width</i> . $width = 79 \times X$ $1^{*1} \leq width \leq 474$ X: module width $2 \leq X \leq 6$ X is automatically set according to <i>width</i> .

\*1: When 1 to 236 is set, *width* is set to 158.

[GS1 Databar Stacked]

Parameter	Limitation
<i>data</i>	Specify 13 letters consisting of '0' to '9'.
<i>height</i>	Input the value other than 0. The value is set as <i>height</i> = 26 regardless of the input value.
<i>width</i>	When <b>MapMode</b> = <i>MapMode.Dots</i> : Input the value other than 0. The value is set as <i>width</i> = 100 regardless of the input value.

Module width is fixed at 2.

[GS1 Databar Truncated]

Parameter	Limitation
<i>data</i>	Specify 13 letters consisting of '0' to '9'.
<i>width</i>	When <b>MapMode</b> = <i>MapMode.Dots</i> : The effective range of <i>height</i> depends on <i>width</i> . $width = 96 \times X$ $1^{*1} \leq width \leq 576$ X: module width $2 \leq X \leq 6$ X is automatically set according to <i>width</i> .

\*1: When 1 to 287 is set, *width* is set to 192.

[PDF417]

Parameter	Limitation
<i>data</i>	It must be a character string in which 0x00 to 0x7F follow the ASCII code and 0x80 to 0xFF follow the extended character set of PC437 English list.
<i>width</i> <i>height</i>	$width = (17 \times C + 69) \times X$ $180 \leq width$ $height = R \times Y$ $14 \leq height \leq 255$ X: module width (2 to 4) Y: module height (2 to 127) C: the number of vertical columns (1 to 30) R: the number of rows (3 to 90) For the number of rows and the number of vertical columns, the smallest value that input data can be converted to barcode is selected. For module width and module height, the maximum size that does not exceed <i>width</i> and <i>height</i> is selected after the number of rows and the number of vertical columns are determined.

Print mode is the normal mode and the error correction level is fixed to 4.

[UPC-A]

Parameter	Limitation
<i>data</i>	Specify 11 or 12 letters consisting of '0' to '9'. The 12th letter does not affect the barcode printing data.
<i>width</i>	When <b>MapMode</b> = <i>MapMode.Dots</i> : $width = 95 \times X$ $190 \leq width \leq 570$ X: fine element width $2 \leq X \leq 6$ X is automatically set according to <i>width</i> .

[UPC-E]

Parameter	Limitation
<i>data</i>	Specify 11 or 12 letters consisting of '0' to '9'. The 12th letter does not affect the barcode printing data.
<i>width</i>	When <b>MapMode</b> = <i>MapMode.Dots</i> : $width = 51 \times X$ $102 \leq width \leq 306$ X: fine element width $2 \leq X \leq 6$ X is automatically set according to <i>width</i> .

Additionally, the allowable character must follow the rules below.

1. The 1st letter is "0".
2. The UPC-A left code indicates the 2nd to the 6th characters, the UPC-A right code indicates the 7th to the 11th characters, and the code to be abbreviated is actually printed as UPC-E. If the specified UPC-A initial character is other than 0 or a character not included in the following list is specified, *ErrorCode.Illegal* is thrown.

Maker Code UPC-A Left Code					Item Code UPC-A Right Code					Abbreviated Code					
F1	F2	F3	F4	F5	A1	A2	A3	A4	A5	Z1	Z2	Z3	Z4	Z5	Z6
0-9	0-9	0	0	0	0	0	0-9	0-9	0-9	F1	F2	A3	A4	A5	0
0-9	0-9	1	0	0	0	0	0-9	0-9	0-9	F1	F2	A3	A4	A5	1
0-9	0-9	2	0	0	0	0	0-9	0-9	0-9	F1	F2	A3	A4	A5	2
0-9	0-9	3-9	0	0	0	0	0-9	0-9	0-9	F1	F2	F3	A4	A5	3
0-9	0-9	0-9	1-9	0	0	0	0	0	0-9	F1	F2	F3	F4	A5	4
0-9	0-9	0-9	0-9	1-9	0	0	0	0	5-9	F1	F2	F3	F4	F5	A5

## PrintBitmap Method

Syntax

```
void PrintBitmap(PrinterStation station, string fileName, int width, int alignment);
```

Parameter	Meaning
<i>station</i>	Specifies the station to be used. Specify <i>PrinterStation.Receipt</i> .
<i>fileName</i>	Specifies the name of bitmap file. For the supported image file, see below.
<i>width</i>	Specifies the print width of bitmap. See values below.
<i>alignment</i>	Specifies the print position of bitmap. See values below.

·Supported bitmap file

Item	Specifications
Extension	bmp
Format	Windows Bitmap
Color	1, 4, 8, 24, or 32 bits
Compression format	Uncompressed only

·Values of *width*

Value	Meaning
<i>PrinterBitmapAsIs</i> (-11)	Prints the bitmap with 1 pixel per printer dot.
Other values	Expresses the bitmap width in the unit defined by <b>MapMode</b> . If <b>MapMode</b> is <i>MapMode.Dots</i> , specify the value from 1 to <b>RecLineWidth</b> . When printing bitmap in a rotated 90° to right/left mode by <b>RotatePrint</b> is executed, specify the value from 1 to 2400. During Page Mode by <b>PageModePrint</b> , specify the value within the range of print area defined by <b>PageModePrintArea</b> and <b>PageModeHorizontalPosition</b> .

The value is rounded up to a multiple of 8 inside the Service Object.

·Values of *alignment*

Value	Meaning
<i>PrinterBitmapCenter</i>	Printed with center.
<i>PrinterBitmapLeft</i>	Printed with left justify.
<i>PrinterBitmapRight</i>	Printed with right justify.
Other values	Printed with the left margin of the specified value. Expressed in the unit given by <b>MapMode</b> .

When rotation 90° right/left is specified by **RotatePrint**, and during Page Mode by **PageModePrint**, the setting of *alignment* is invalid and the data is always printed with left justify.

**Description** Prints a bitmap on the specified station.

The highest performance cannot be achieved since the bitmap data is transferred to the printer after **PrintBitmap** is called. It is recommended to print the bitmap data using **SetBitmap** and escape sequence.

If any character data is already sent but not yet printed, that character data is printed first, a linefeed is automatically added, and then the bitmap is printed on the next print line. Any character data sent after **PrintBitmap** is printed on the print line following the bitmap. This method is executed synchronously if **AsyncMode** is *false*, and asynchronously if **AsyncMode** is *true*.

*width* controls the transformation of bitmap data. If *width* is *PrinterBitmapAsIs*, then no transformation is executed. The bitmap is printed with 1 pixel per printer dot.

If *width* is not 0, then the bitmap will be transformed by stretching or compressing the bitmap such that its width is the specified width and the aspect ratio is unchanged.

## PrintImmediate Method

**Syntax** `void PrintImmediate(PrinterStation station, string data);`

Parameter	Meaning
<i>station</i>	Specifies the station to be used. Specify <i>PrinterStation.Receipt</i> .
<i>data</i>	Specifies the characters to be printed. Consists of printable characters, escape sequences, carriage returns (CR), and line feeds (LF).

**Description** This method is used for printing immediately during asynchronous output.

When this method is specified during asynchronous output, processing of this method is executed before the next asynchronous output, after asynchronous output currently being processed.

**PrintImmediate** is intended for use in exception conditions when asynchronous output is not resolved, for example, in an error event handler.

The print data that exceeds the maximum number of characters per line is printed on the next print line.

If there is data remaining unprinted in the printer buffer, printing is executed after all the buffered data is printed.

The values and meanings of special characters within *data* are as follows.

Symbol	Operation
LF	Prints data in the buffer, and feeds to the next line.
CR	Replaceable with the same operation as line feed (LF).
LF & CR	Carriage return (CR) is replaceable with the same operation as line feed (LF). Therefore, operation of line feed (LF) is executed twice.
CR & LF	Carriage return (CR) is ignored. Operation of line feed (LF) is executed once.

## PrintMemoryBitmap Method

Syntax      **void PrintMemoryBitmap(PrinterStation** *station*,  
                 **Bitmap** *data*,  
                 **int** *width*,  
                 **int** *alignment*);

Parameter	Meaning
<i>station</i>	Specifies the station to be used. Specify <i>PrinterStation.Receipt</i> .
<i>data</i>	Specifies the byte array holding the bitmap data. For the supported image file, see <b>PrintBitmap</b> .
<i>width</i>	Specifies the print width of bitmap. See <b>PrintBitmap</b> for values.
<i>alignment</i>	Specifies the print position of bitmap. See <b>PrintBitmap</b> for values.

Description      Prints the bitmap on the specified station.  
For the operation specifications, see **PrintBitmap**.  
This method is executed synchronously if **AsyncMode** is *false*, and asynchronously if **AsyncMode** is *true*.

## PrintNormal Method

Syntax      **void PrintNormal(PrinterStation** *station*, **string** *data*);

Parameter	Meaning
<i>station</i>	Specifies the station to be used. Specify <i>PrinterStation.Receipt</i> .
<i>data</i>	Specifies the characters to be printed. Consists of printable characters, escape sequences, carriage returns (CR), and line feeds (LF).

Description      Prints *data* on the printer.  
The print data that exceeds the maximum number of characters per line is printed on the next print line.  
If there is data remaining unprinted in the printer buffer, printing is executed after all the buffered data is printed.  
This method is executed synchronously if **AsyncMode** is *false*, and asynchronously if **AsyncMode** is *true*.

The values and meanings of special characters within *data* are as follows.

Symbol	Operation
LF	Prints data in the buffer, and feed to the next line.
CR	Replaceable with the same operation as line feed (LF).
LF & CR	Carriage return (CR) is replaceable with the same operation as line feed (LF). Therefore, operation of line feed (LF) is executed twice.



Symbol	Operation
CR & LF	Carriage return (CR) is ignored. Operation of line feed (LF) is executed once.

## RotatePrint Method

Syntax

```
void RotatePrint(PrinterStation station, PrintRotation rotation);
```

Parameter	Meaning
<i>station</i>	Specifies the station to be used. Specify <i>PrinterStation.Receipt</i> .
<i>rotation</i>	Specifies the rotation direction. See values below.

· Values of *rotation*

Value	Meaning
<i>PrintRotation.Right90</i>	Starts rotated 90° right (clockwise) printing.
<i>PrintRotation.Left90</i>	Starts rotated 90° left (counterclockwise) printing.
<i>PrintRotation.Rotate180</i>	Starts rotated 180° printing (upside down printing).
<i>PrintRotation.Barcode</i>	Starts rotated barcode printing. This value is ORed with one of the above start rotated print values.
<i>PrintRotation.Bitmap</i>	Starts rotated bitmap printing. This value is ORed with one of the above start rotated print values.
<i>PrintRotation.Normal</i>	Ends rotated printing.

Description

Executes the rotated printing.

This method is executed synchronously if **AsyncMode** is *false*, and asynchronously if **AsyncMode** is *true*.

When *rotation* contains *PrintRotation.Rotate180*, the upside down print mode is started. Subsequent calls to **PrintNormal** or **PrintImmediate** will print the data upside down until **RotatePrint** is called with *rotation* set to *PrintRotation.Normal*. Lines are printed in the order that they are sent to the Service Object, with the start of each line justified at the right margin of the printer. When *rotation* does not contain *PrintRotation.Barcode* or *PrintRotation.Bitmap*, only the print methods **PrintNormal** and **PrintImmediate** are used during the upside down print mode.

When *rotation* contains *PrintRotation.Right90* or *PrintRotation.Left90*, the sideways print mode is started. Until **RotatePrint** is called with *rotation* set to *PrintRotation.Normal*, the data called by **PrintNormal** is buffered. The value of **AsyncMode** does not affect the operation. In other words, no **OutputId** will be assigned to the request, nor will **OutputCompleteEvent** be notified.

Each print line is rotated by 90°. If all lines do not have the same length, the start positions of the lines are aligned. When *rotation* does not contain *PrintRotation.Barcode* or *PrintRotation.Bitmap*, only **PrintNormal** is used in the sideways print mode.

When *rotation* contains *PrintRotation.Normal*, the rotated print mode is exited. If some data is buffered by **PrintNormal** while the sideways rotated print mode is in effect, the buffered data is printed. The whole block of rotated lines is treated as one message.

When *rotation* contains *PrintRotation.Barcode* or *PrintRotation.Bitmap*, all of barcodes (printed by **PrintBarCode** or the "Print in-line barcode" escape sequence (ESC|#R)) and bitmaps (printed by **PrintBitmap** or the "Print bitmap" escape sequence (ESC|#B)) can be printed in a rotated mode by **RotatePrint**. The rotation direction of barcodes and bitmaps are limited by **RecBarCodeRotationList** and **RecBitmapRotationList** respectively. When *rotation* contains *PrintRotation.Barcode*, the contents of **RotateSpecial** are ignored. Calling **ClearOutput** cancels the rotated print mode. Any buffered lines of sideways rotated print are also cleared.

The Service Object calculates so that the width in the sideways print mode becomes best size. The maximum width is 2400 dots in the sideways print mode. If the print data per line exceeds this range, non-printed data is printed by feeding to the next print line. If the bitmap print or barcode print by the "Print in-line barcode" escape sequence (ESC|#R) or the "Print bitmap" escape sequence (ESC|#B) is specified on **PrintNormal** during the rotation mode, the print data rotates regardless of whether or not *PrintRoatation.Bitmap* and *PrintRotation.Barcode* is specified in *rotation* with OR.

## SetBitmap Method

Syntax      **void SetBitmap**(int *bitmapNumber*,  
                  **PrinterStation** *station*,  
                  **string** *fileName*,  
                  **int** *width*,  
                  **int** *alignment*);

Parameter	Meaning
<i>bitmapNumber</i>	Specifies the number to be assigned to this bitmap. Valid values are 1 to 20.
<i>station</i>	Specifies the station to be used. Specify <i>PrinterStation.Receipt</i> .
<i>fileName</i>	Specifies the name of bitmap file. If an empty string is set, the bitmap setting is canceled. For the supported image file, see <b>PrintBitmap</b> .
<i>width</i>	Specifies the print width of bitmap. See <b>PrintBitmap</b> for values.
<i>alignment</i>	Specifies the print position of bitmap. See <b>PrintBitmap</b> for values.

Description      Saves information of a bitmap to be printed.  
 The bitmap is printed by calling **PrintNormal** or **PrintImmediate** with the "Print Bitmap" escape sequence (ESC|#B) in the print data. The "Print Bitmap" escape sequence (ESC|#B) usually contains the character strings for printing the start and end process headers.

If any character data was sent before the "Print Bitmap" escape sequence (ESC|#B) and has not been printed, that character data is printed first, a linefeed is automatically placed, and then the bitmap is printed. Any character data sent after the "Print Bitmap" escape sequence (ESC|#B) is printed on the line next to the bitmap.

The Service Object prepares printing with downloading bitmap data in the NV graphics area of the printer. When bitmap print is specified by escape sequence, only command which conducts printing is transmitted to provide better performance.

## SetLogo Method

Syntax **void SetLogo(PrinterLogoLocation *location*, string *data*);**

Parameter	Meaning
<i>location</i>	Specifies the logo to be set.
<i>data</i>	Specifies the characters that produce the logo. Consists of printable characters, escape sequences, carriage returns (CR), and line feeds (LF).

• Values of *location*

Value	Meaning
<i>PrinterLogoLocation.Bottom</i>	Produces a bottom logo.
<i>PrinterLogoLocation.Top</i>	Produces a top logo.

Description Saves a data string as the top or bottom logo.  
The logo can be printed by calling **PrintNormal** or **PrintImmediate** with the "Print Top Logo" escape sequence (ESC|tL) or "Print Bottom Logo" escape sequence (ESC|bL) in the print data.

## TransactionPrint Method

Syntax **void TransactionPrint(PrinterStation *station*, PrinterTransactionControl *control*);**

Parameter	Meaning
<i>station</i>	Specifies the station to be used. Specify <i>PrinterStation.Receipt</i> .
<i>control</i>	Specifies the type of the transaction. See below for values.

• Values of *control*

Value	Meaning
<i>PrinterTransactionControl.Normal</i>	Ends a transaction by printing the buffered data.
<i>PrinterTransactionControl.Transaction</i>	Starts a transaction.

**Description**      Calls this method to enter or exit transaction mode.

If *control* is *PrinterTransactionControl.Transaction*, then transaction mode is entered. Subsequent calls to **PrintNormal**, **RotatePrint**, **PrintBarCode**, **PrintBitmap**, or **PageModePrint** will buffer the print data until **TransactionPrint** is called with *control* set to *PrinterTransactionControl.Normal*.

The value of **AsyncMode** does not affect the operation. In other words, no **OutputId** will be assigned to the request, nor will **OutputCompleteEvent** be notified.

If *control* is *PrinterTransactionControl.Normal*, then transaction mode is exited. If some data was buffered by calls to **PrintNormal**, **RotatePrint**, **PrintBarCode**, **PrintBitmap**, or **PageModePrint**, then the buffered data is printed. The entire transaction is treated as one message.

This method is executed synchronously if **AsyncMode** is *false*, and asynchronously if **AsyncMode** is *true*.

Calling **ClearOutput** cancels transaction mode. Any buffered print lines are also cleared.

## ValidateData Method

**Syntax**              **void ValidateData(PrinterStation station, string data);**

Parameter	Meaning
<i>station</i>	Specifies the station to be used. Specify <i>PrinterStation.Receipt</i> .
<i>data</i>	Specifies the data to be validated. Consists of printable characters, escape sequences, carriage returns (CR), and line feeds (LF).

**Description**      Before calling **PrintNormal** or **PrintImmediate**, this method is called to determine whether a data sequence, possibly including one or more escape sequences, is valid for the specified station.

This method does not cause any printing but is used to determine the capability of the station.

When not valid, the exception error is thrown. For details about the thrown exception errors, see "Appendix A Exceptions".

#### 4.1.7 Events

This section describes the details of PosPrinter events.

##### DirectIOEvent Event

This event is not supported.

Syntax      **DirectIOEventHandler DirectIOEvent;**

##### ErrorEvent Event

Syntax      **DeviceErrorEventHandler ErrorEvent;**

Description      This event is notified when an error occurs and **State** of the Service Object enters the error state.

When *DeviceErrorEventArgs.ErrorCode* is *ErrorCode.Extended*,  
*DeviceErrorEventArgs.ErrorCodeExtended* is set to one of the following values:

Value	Meaning
<i>ExtendedErrorCoverOpen</i> (201)	The printer cover is open.
<i>ExtendedErrorRecEmpty</i> (203)	The receipt is out of paper.
<i>ExtendedErrorVpPower</i> (1001)	Vp voltage error has occurred.
<i>ExtendedErrorHeadTemp</i> (1005)	Head-temperature error has occurred.
<i>ExtendedErrorFatal</i> (1010)	Unrecoverable error has occurred.
<i>ExtendedErrorBattery</i> (1013)	Battery error has occurred.
<i>ExtendedErrorMarkPaperJam</i> (1014)	Paper jam error has occurred at mark detection.

The *DeviceErrorEventArgs.ErrorResponse* can be set to either of the following values by the application. The default is *ErrorResponse.Retry*.

Value	Meaning
<i>ErrorResponse.Clear</i>	Exits the error state and clears the asynchronous output.
<i>ErrorResponse.Retry</i>	Exits the error state and retries the asynchronous output.

##### OutputCompleteEvent Event

Syntax      **OutputCompleteEventHandler OutputCompleteEvent;**

Description      This event is notified when the previously started asynchronous output request is completed successfully.

**OutputId** indicates the ID number of the completed asynchronous output request.

## StatusUpdateEvent Event

Syntax **StatusUpdateEventHandler StatusUpdateEvent;**

Description The status of the device is notified when the device encounters an important status change.

The Service Object notifies the first **StatusUpdateEvent** when the device is enabled.

*StatusUpdateEventArgs.Status* is set to one of the following values:

Value	Meaning
<i>StatusCoverOpen</i> (11)	The printer cover is open.
<i>StatusCoverOK</i> (12)	The printer cover is closed.
<i>StatusReceiptEmpty</i> (24)	The receipt is out of paper.
<i>StatusReceiptPaperOK</i> (26)	The receipt is ready.
<i>StatusIdle</i> (1001)	All the asynchronous outputs finished either successfully or by being cleared. <b>State</b> is now <i>ControlState.Idle</i> . <b>FlagWhenIdle</b> must be <i>true</i> for this event to be notified. And the Service Object automatically resets the property to <i>false</i> before the event is notified.
<i>StatusPowerOnline</i> (2001)* <sup>1</sup>	The device is powered on and ready.
<i>StatusPowerOffOffline</i> (2004)* <sup>1</sup>	The device is powered off or offline.

\*1: Notified when **PowerNotify** = *PowerNotification.Enabled*.

---

## Appendix A Exceptions

---

### A.1 PosPrinter Exception Error List

#### (1) Property

ErrorCode	ErrorCode Extended	Meaning
Disabled	0	Not enabled. Call this after setting <b>DeviceEnabled</b> to <i>true</i> .
Illegal	0	This property is not supported. Parameter has an error.
NotClaimed	0	Exclusive access is not available. Call <b>Claim</b> to gain exclusive access.

#### (2) Method

Method	ErrorCode	ErrorCode Extended	Meaning
<b>BeginInsertion</b> <b>BeginRemoval</b> <b>ChangePrintSide</b> <b>CutPaper</b> <b>CompareFirmwareVersion</b> <b>EndInsertion</b> <b>EndRemoval</b> <b>PrintTwoNormal</b> <b>UpdateFirmware</b> <b>UpdateStatistic(s)</b>	Illegal	0	This method is not supported.
<b>CheckHealth</b>	Busy	0	Cannot perform while output is in progress or an error occurs.
	Disabled	0	Not enabled. Call this after setting <b>DeviceEnabled</b> to <i>true</i> .
	Illegal	0	Parameter has an error.
	NotClaimed	0	Exclusive access is not available. Call <b>Claim</b> to gain exclusive access.

Method	ErrorCode	ErrorCode Extended	Meaning
Claim	Claimed	0	Attempt was made to access a device that is exclusively accessed by another process.
	Failure	0	Communication with the printer failed.
	Illegal	0	Parameter has an error.
	NoHardware	0	The printer is powered off or the cable is not connected.
	Timeout	0	Another application has exclusive access to the device and the <i>timeout</i> (in milliseconds) has elapsed before the device is released. Or the device did not become available even though the <i>timeout</i> (in milliseconds) has elapsed.
ClearOutput	NotClaimed	0	Exclusive access is not available. Call <b>Claim</b> to gain exclusive access.
ClearPrintArea	Disabled	0	Not enabled. Call this after setting <b>DeviceEnabled</b> to <i>true</i> .
	NotClaimed	0	Exclusive access is not available. Call <b>Claim</b> to gain exclusive access.
Close	Busy	0	<b>State</b> is set to <i>ControlState.Busy</i> . This means that the device is busy and cannot be stopped.
	Closed	0	The device is already closed.
Open	Illegal	0	The device is already open.
DirectIO MarkFeed PageModePrint PrintBarCode PrintBitmap PrintMemoryBitmap PrintNormal PrintImmediate RotatePrint SetBitmap TransactionPrint	Busy	0	Cannot perform while output is in progress or an error occurs.
	Disabled	0	Not enabled. Call this after setting <b>DeviceEnabled</b> to <i>true</i> .
	Extended	201	The printer cover is open.
	Extended	203	The receipt is out of paper.
	Extended	1001	A Vp voltage error occurred.
	Extended	1005	A head temperature error occurred.
	Extended	1010	An unrecoverable error occurred.
	Extended	1013	A battery error occurred.
	Extended	1014	Paper jam error has occurred at mark detection.
	Failure	0	Communication with the printer failed.
	Illegal	0	Parameter has an error. <b>PageModeStation</b> is not specified. ( <b>PageModePrint</b> )
	NoHardware	0	The printer is powered off or the cable is not connected.
	NotClaimed	0	Exclusive access is not available. Call <b>Claim</b> to gain exclusive access.
	Timeout	0	Data transmit timeout or data receive timeout has occurred.
Release	NotClaimed	0	The device is not exclusive.



Method	ErrorCode	ErrorCode Extended	Meaning
<b>ValidateData</b>	Disabled	0	Not enabled. Call this after setting <b>DeviceEnabled</b> to <i>true</i> .
	Failure	0	At least one of the escape sequences is not supported. No alternatives can be selected.
	Illegal	0	At least one of the escape sequences is out of the range. However, the Service Object can select valid alternatives. Also, this value is placed when the escape sequence is not supported by the Page Mode or rotated 90° left or right print mode.
	Illegal	0	Parameter has an error.
	NotClaimed	0	Exclusive access is not available. Call <b>Claim</b> to gain exclusive access.

*ErrorCode.Illegal* is thrown for **ValidateData** in the following cases:

Escape Sequence	Condition
Print bitmap	No printable bitmap exists.
Feed lines	It is in one of the following states. <ul style="list-style-type: none"> <li>• The number of lines '#' is not correct.</li> <li>• Not supported during rotated 90° right/left mode by <b>RotatePrint</b>.</li> <li>• Not supported during Page Mode by <b>PageModePrint</b>.</li> </ul>
Feed units	It is in one of the following states. <ul style="list-style-type: none"> <li>• The amount of feed '#' is not precisely supported due to occurrence of rounding error of one dot depending on the setting of <b>MapMode</b>.</li> <li>• The amount of feed '#' is not correct.</li> <li>• Not supported during rotated 90° right/left mode by <b>RotatePrint</b>.</li> <li>• Not supported during Page Mode by <b>PageModePrint</b>.</li> </ul>
Pass through embedded data	The number of bytes of embedded data '#' is not correct.
Print in-line barcode	The character string following ESC #R is not correct.
Underline	The thickness '#' is not correct.
Scale vertically	The scale factor '#' is not correct.
Scale horizontally	The scale factor '#' is not correct.
Left justify	It is in one of the following states. <ul style="list-style-type: none"> <li>• Not supported during rotated 90° right/left mode by <b>RotatePrint</b>.</li> <li>• Not supported during Page Mode by <b>PageModePrint</b>.</li> </ul>
Center	It is in one of the following states. <ul style="list-style-type: none"> <li>• Not supported during rotated 90° right/left mode by <b>RotatePrint</b>.</li> <li>• Not supported during Page Mode by <b>PageModePrint</b>.</li> </ul>
Right justify	It is in one of the following states. <ul style="list-style-type: none"> <li>• Not supported during rotated 90° right/left mode by <b>RotatePrint</b>.</li> <li>• Not supported during Page Mode by <b>PageModePrint</b>.</li> </ul>

*ErrorCode.Failure* is thrown for **ValidateData** in the following cases:

Escape Sequence	Condition
Paper cut	Not supported.
Feed and Paper cut	Not supported.
Feed, Paper cut, and Stamp	Not supported.
Print bitmap	The bitmap number '#' is not correct.
Print stamp	Not supported.
Feed reverse	Not supported.
Font typeface	Not supported.
Italic	Not supported.
Custom color	Not supported.
Red color	Not supported.
Shading character	Not supported.
Color option	Not supported.
SubScript	Not supported.
SuperScript	Not supported.
Strike-through	Not supported.

---

## Appendix B Statistics

---

(1) StatisticCategories.Upos

XML Definition Name	Response	Can Be Reset
JournalCoverOpenCount	0	-
ReceiptLineFeedCount	Number of receipt line feeds performed (unit: 100 dot-line)	✓
PrintSideChangeCount	0	-
ReceiptCharacterPrintedCount	0	-
ReceiptCoverOpenCount	0	-
ManufactureDate	Unknown	-
PaperCutCount	0	-
UnifiedPOSVersion	1.12	-
SlipCoverOpenCount	0	-
HoursPoweredCount	Number of hours powered on (unit: hour)	✓
FirmwareRevision	Firmware version	-
SerialNumber	Unknown	-
ReceiptLinePrintedCount	0	-
InstallationDate	Unknown	-
MechanicalRevision	24	-
FailedPaperCutCount	0	-
StampFiredCount	0	-
FailedPrintSideChangeCount	0	-
JournalCharacterPrintedCount	0	-
SlipCharacterPrintedCount	0	-
ManufacturerName	Seiko Instruments Inc.	-
PrinterFaultCount	0	-
MaximumTempReachedCount	0	-

XML Definition Name	Response	Can Be Reset
ModelName	MP-B30	-
CommunicationErrorCount	0	-
JournalLinePrintedCount	0	-
SlipLineFeedCount	0	-
HomeErrorCount	0	-
FormInsertionCount	0	-
Interface	Unknown	-
DeviceCategory	POSPrinter	-
BarcodePrintedCount	0	-
NVRAMWriteCount	0	-
SlipLinePrintedCount	0	-

(2) StatisticCategories.Manufacturer

XML Definition Name	Response	Can Be Reset
HoursPoweredCount_Accumulated	Number of hours powered on (unit: hour) (accumulated)	-
ReceiptLineFeedCount_Accumulated	Number of receipt line feeds performed (unit: 100 dot-line) (accumulated)	-