



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

U00131659710

[Products]

RP-D10 Series

RP-E1x Series

Seiko Instruments Inc.

U00131659700	November 2013
U00131659701	December 2013
U00131659702	October 2014
U00131659703	April 2015
U00131659704	June 2017
U00131659705	March 2019
U00131659706	November 2019
U00131659707	October 2020
U00131659708	December 2021
U00131659709	October 2022
U00131659710	July 2023


Copyright© 2013-2023 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 registered trademarks 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 document describes the specifications, functions, and operating instructions of the SII POS for .NET Service Object (hereinafter referred to as "the software") provided by SII for the POS printers and cash drawers that are connected to the POS printers.

This document is basically written on the basis of the following conditions:

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

Target Printers

This section lists the printers (products) supported by the software.

	Description in This Manual	Printer
POS printer	RP-D10	RP-D10
	RP-E10	RP-E10
		RP-E11

Refer to "UnifiedPOS Retail Peripheral Architecture Version 1.12" (hereinafter "UPOS V 1.12") and "Microsoft Point of Service for .NET - POS for .NET v1.12 SDK Documentation" before using the software.

Operating System Abbreviations

This section lists the operating system abbreviations used in this document.

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

Definition	Description
Default	The value immediately after satisfying the availability condition.
Line spacing	In the software, this means the height of each single-high print line (total value of the printed line height and the whitespace between each pair of lines).

Table of Contents

Chapter 1 Overview 1-1

1.1	Configuration	1-1
1.1.1	Structural Diagram.....	1-1
1.2	Printer Settings	1-2
1.3	Limitation	1-3
1.3.1	General	1-3
1.3.2	PosPrinter	1-3
1.3.3	CashDrawer	1-4

Chapter 2 Operating Environment 2-1

2.1	System Requirements.....	2-1
-----	--------------------------	-----

Chapter 3 Installation 3-1

3.1	Installation	3-1
3.2	Configuration Program	3-4
3.2.1	Item and Detail of Configuration Program	3-4
3.2.2	Startup Configuration Program	3-7
3.2.3	Configuration Program Function	3-7
3.3	Uninstallation	3-15

Chapter 4 Properties, Methods, and Events 4-1

4.1	PosPrinter	4-1
4.1.1	Summary.....	4-1
4.1.2	Data Characters and Escape Sequences.....	4-9
4.1.3	Common Properties.....	4-13
	CapCompareFirmwareVersion Property	4-13
	CapPowerReporting Property	4-13

	CapStatisticsReporting Property	4-13
	CapUpdateFirmware Property	4-14
	CapUpdateStatistics Property	4-14
	CheckHealthText Property	4-14
	Claimed Property	4-15
	DeviceDescription Property	4-15
	DeviceEnabled Property R/W	4-15
	DeviceName Property	4-16
	FreezeEvents Property R/W	4-16
	OutputId Property	4-17
	PowerNotify Property R/W	4-17
	PowerState Property	4-18
	ServiceObjectDescription Property	4-18
	ServiceObjectVersion Property	4-18
	State Property	4-19
	SynchronizingObject Property	4-19
4.1.4	Specific Properties	4-20
	AsyncMode Property R/W	4-21
	CapCharacterSet Property	4-21
	CapCoverSensor Property	4-21
	CapMapCharacterSet Property	4-22
	CapRec2Color Property	4-22
	CapRecBarCode Property	4-22
	CapRecBitmap Property	4-22
	CapRecBold Property	4-23
	CapRecCartridgeSensor Property	4-23
	CapRecColor Property	4-23
	CapRecDHigh Property	4-23
	CapRecDWide Property	4-24
	CapRecDWideDHigh Property	4-24
	CapRecEmptySensor Property	4-24
	CapRecItalic Property	4-24
	CapRecLeft90 Property	4-25
	CapRecMarkFeed Property	4-25
	CapRecNearEndSensor Property	4-25
	CapRecPageMode Property	4-26
	CapRecPaperCut Property	4-26
	CapRecPresent Property	4-26
	CapRecRight90 Property	4-26
	CapRecRotate180 Property	4-27
	CapRecStamp Property	4-27
	CapRecUnderline Property	4-27
	CapTransaction Property	4-27
	CartridgeNotify Property R/W	4-28
	CharacterSet Property R/W	4-28
	CharacterSetList Property	4-29
	CoverOpen Property	4-29

ErrorLevel Property	4-29
ErrorStation Property.....	4-30
ErrorString Property.....	4-30
FlagWhenIdle Property R/W	4-31
FontTypefaceList Property.....	4-31
MapCharacterSet Property R/W	4-31
MapMode Property R/W	4-32
PageModeArea Property	4-33
PageModeDescriptor Property.....	4-33
PageModeHorizontalPosition Property R/W.....	4-34
PageModePrintArea Property R/W	4-35
PageModePrintDirection Property R/W	4-36
PageModeStation Property R/W	4-38
PageModeVerticalPosition Property R/W.....	4-38
RecBarCodeRotationList Property	4-39
RecBitmapRotationList Property	4-39
RecCartridgeState Property.....	4-39
RecCurrentCartridge Property R/W.....	4-40
RecEmpty Property	4-40
RecLetterQuality Property R/W.....	4-40
RecLineChars Property R/W.....	4-41
RecLineCharsList Property	4-42
RecLineHeight Property R/W.....	4-43
RecLineSpacing Property R/W	4-44
RecLinesToPaperCut Property	4-45
RecLineWidth Property.....	4-45
RecNearEnd Property	4-46
RecSidewaysMaxChars Property	4-46
RecSidewaysMaxLines Property	4-47
RotateSpecial Property R/W	4-47
4.1.5 Common Methods	4-48
CheckHealth Method	4-48
Claim Method	4-48
ClearOutput Method	4-49
Close Method	4-49
CompareFirmwareVersion Method	4-49
DirectIO Method	4-49
Open Method	4-51
Release Method	4-51
ResetStatistic(string) Method	4-51
ResetStatistics() Method.....	4-51
ResetStatistics(StatisticCategories) Method.....	4-52
ResetStatistics(string[]) Method	4-52
RetrieveStatistic(string) Method	4-52
RetrieveStatistics() Method.....	4-52
RetrieveStatistics(StatisticCategories) Method.....	4-52
RetrieveStatistics(string[]) Method	4-53

	UpdateFirmware Method	4-53
	UpdateStatistic Method	4-53
	UpdateStatistics(Statistic[]) Method	4-53
	UpdateStatistics(StatisticCategories, Object) Method.....	4-53
4.1.6	Specific Methods	4-54
	ClearPrintArea Method	4-55
	CutPaper Method	4-55
	PageModePrint Method	4-56
	PrintBarCode Method	4-58
	PrintBitmap Method	4-67
	PrintImmediate Method	4-69
	PrintMemoryBitmap Method	4-70
	PrintNormal Method.....	4-70
	RotatePrint Method	4-71
	SetBitmap Method.....	4-73
	SetLogo Method.....	4-73
	TransactionPrint Method.....	4-74
	ValidateData Method	4-75
4.1.7	Events	4-76
	DirectIOEvent Event.....	4-76
	ErrorEvent Event.....	4-76
	OutputCompleteEvent Event	4-76
	StatusUpdateEvent Event.....	4-77
4.2	CashDrawer	4-78
4.2.1	Summary.....	4-78
4.2.2	Common Properties.....	4-80
	CapCompareFirmwareVersion Property	4-80
	CapPowerReporting Property	4-80
	CapStatisticsReporting Property	4-80
	CapUpdateFirmware Property	4-81
	CapUpdateStatistics Property.....	4-81
	CheckHealthText Property.....	4-81
	Claimed Property.....	4-82
	DeviceDescription Property	4-82
	DeviceEnabled Property R/W	4-82
	DeviceName Property	4-83
	FreezeEvents Property R/W	4-83
	PowerNotify Property R/W	4-84
	PowerState Property	4-84
	ServiceObjectDescription Property	4-85
	ServiceObjectVersion Property.....	4-85
	State Property	4-85
	SynchronizingObject Property	4-86
4.2.3	Specific Properties.....	4-87
	CapStatus Property	4-87
	CapStatusMultiDrawerDetect Property.....	4-87
	DrawerOpened Property.....	4-87

4.2.4	Common Methods	4-88
	CheckHealth Method	4-88
	Claim Method	4-88
	Close Method	4-89
	CompareFirmwareVersion Method	4-89
	DirectIO Method	4-89
	Open Method	4-89
	Release Method	4-89
	ResetStatistic(string) Method	4-89
	ResetStatistics() Method	4-90
	ResetStatistics(StatisticCategories) Method	4-90
	ResetStatistics(string[]) Method	4-90
	RetrieveStatistic(string) Method	4-90
	RetrieveStatistics() Method	4-90
	RetrieveStatistics(StatisticCategories) Method	4-90
	RetrieveStatistics(string[]) Method	4-90
	UpdateFirmware Method	4-91
	UpdateStatistic Method	4-91
	UpdateStatistics(Statistic[]) Method	4-91
	UpdateStatistics(StatisticCategories, Object) Method	4-91
4.2.5	Specific Methods	4-92
	OpenDrawer Method	4-92
	WaitForDrawerClose Method	4-92
4.2.6	Events	4-93
	StatusUpdateEvent Event	4-93

Appendix A	Exceptions	A-1
-------------------	-------------------	------------

A.1	PosPrinter Exception Error List	A-1
A.2	CashDrawer Exception Error List	A-6

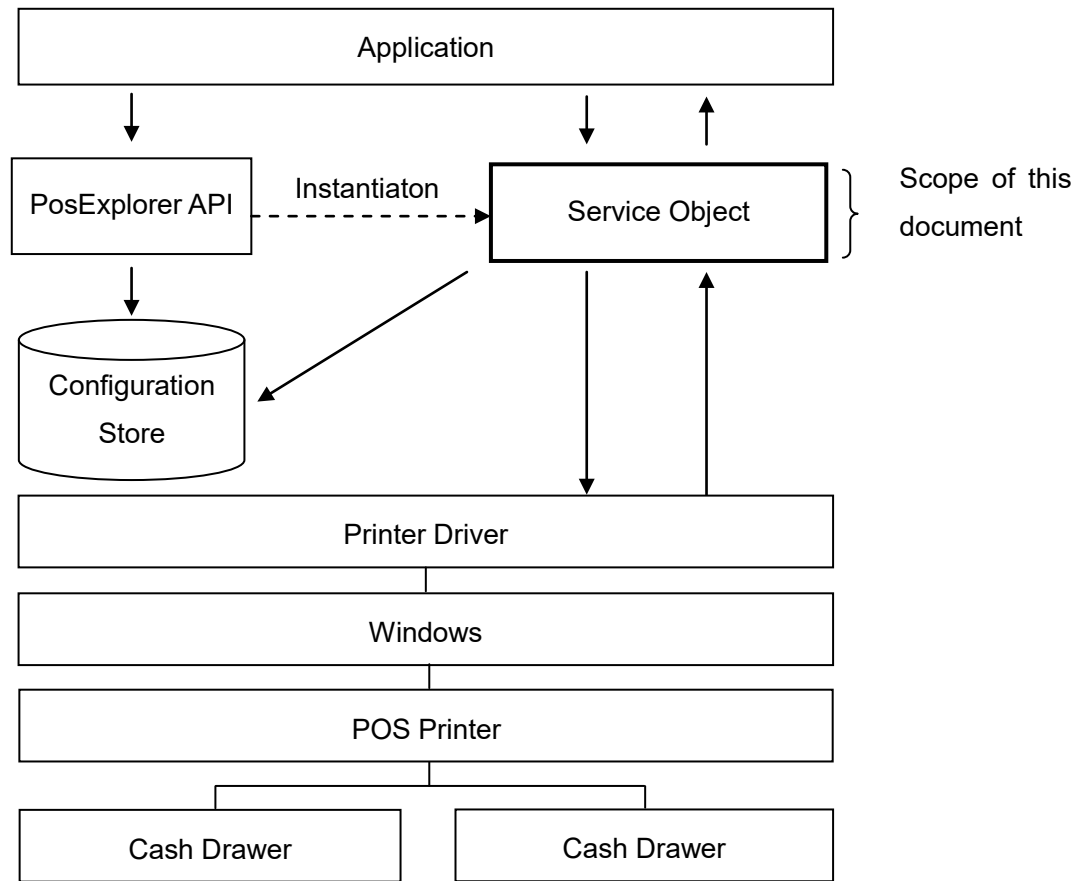
Appendix B	Statistics	B-1
-------------------	-------------------	------------

Chapter 1 Overview

1.1 Configuration

1.1.1 Structural Diagram

Structure of the software and the scope of this document are as follows.



1.2 Printer Settings

The memory switches of the printer are set to [Value] in the following table when using the software. See "RP-D10 SERIES THERMAL PRINTER TECHNICAL REFERENCE" or "RP-E10 SERIES THERMAL PRINTER TECHNICAL REFERENCE" for details about the memory switches.

When in the PosPrinter control, the execution of **Claim** method sets [Value] in the following table forcibly. When in the CashDrawer control, the [Value] is set to the following table forcibly by setting *true* in **DeviceEnabled** property or executing **Claim** method.

MS	Function	Value	Note
1-3	Mark Mode Selection ^{*1} (Mark Mode)	0 : Disable	-
1-6	Near-end Sensor Function Selection ^{*1} (Near End Sensor)	0 : Disable 1 : Enable	Either one [Value] on the left can be set by [NearEnd Sensor] in the configuration program.
4-4	Paper Width Selection (Paper Width)	0 : 58 mm ^{*2} 1 : 80 mm ^{*3}	Either one [Value] on the left is set forcibly by [Number of Effective Dots(dots)] in the configuration program.
4-5	Number of Effective Dots Selection (Number of Effective Dots)	0 : 360dots/512dots 1 : 432dots/576dots	Either one [Value] on the left can be set by [Number of Effective Dots(dots)] in the configuration program.
4-7 to 4-8	Maximum Print Speed Selection (Print Speed)	00B : Low 01B : Middle (Quality) 10B : Middle (Silent) 11B : High	Any one of [Value] on the left can be set by [Print Speed] in the configuration program.
5-1	Automatic Status Response Selection (Auto Status Back)	0 : Enable	-
5-2	Initialized Response Selection (Init. Response)	0 : Enable	
5-3	Data Discard Selection When an Error Occurs (Error Through)	1 : Disable	
5-5	Paper-near-end Sensor Error Selection ^{*1} (Near End Error)	1 : Disable	
13-3	Realtime Command Selection ^{*4} (Realtime Command)	1 : Enable	

*1: In the case of RP-E10.

*2: This value is set when [Number of Effective Dots(dots)] in the configuration program is 360dots/512dots.

*3: This value is set when [Number of Effective Dots(dots)] in the configuration program is 432dots/576dots.

*4: In the case of RP-D10. In the case of RP-E10, the printer command "Enable/Disable Realtime Command" is executed as "Enable".

1.3 Limitation

1.3.1 General

Use a shared printer when all of the following are applied. (If a shared printer is not used, transmission data may interrupt.)

- a. When using 1 printer from multiple computers via TCP/IP connections
- b. When dividing the transmission data into multiple transmissions

For information on how to install and configure shared printers, see the "RP-D10 Series Printer Driver User's Guide" or "RP-E10 Series Printer Driver User's Guide".

1.3.2 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.
 - 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** Method
 - **EndInsertion** Method
 - **BeginRemoval** Method
 - **EndRemoval** Method
 - **ChangePrintSide** Method
 - **MarkFeed** Method
 - **PrintTwoNormal** Method
- (d) The **DirectIOEvent** event (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 operation may occur when "Execution Response Request" is sent 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) When the Bluetooth model is used, if the recoverable error has occurred during printout, it takes time before device is ready for use after recovering from the error.
Wait at least 10 seconds after recovering from the error.

1.3.3 CashDrawer

All interfaces of CashDrawer device are provided with the following limitations.

- (a) All the following methods always return *ErrorCode.Illegal* after they are enabled.
 - **DirectIO** Method
- (b) The **DirectIOEvent** event (device-specific event) is not supported.

Chapter 2 Operating Environment

2.1 System Requirements

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 ^{*1}	.NET Framework 3.5 or .NET Framework 4.0
Microsoft POS for .NET SDK ^{*1}	POS for .NET 1.12 or POS for .NET 1.14
Printer driver ^{*2}	For RP-D10: RP-D10 series printer driver For RP-E10: RP-E10 series printer driver

*1: Before installing the 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

*2: Before installing the software, it is required to install the printer driver for model used in advance. Use dedicated installer for each model to install the printer driver. See "RP-D10 Series Printer Driver User's Guide" or "RP-E10 Series Printer Driver User's Guide" of each model for the installation procedure.

Chapter 3 Installation

3.1 Installation

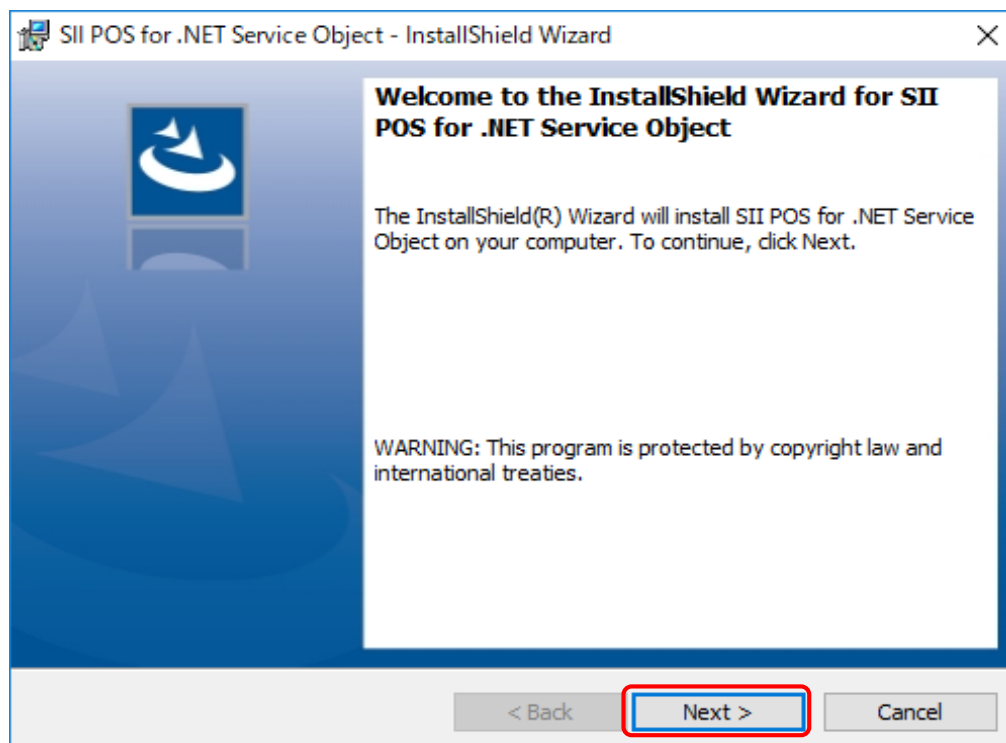
The installation procedure of this software is described below.

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

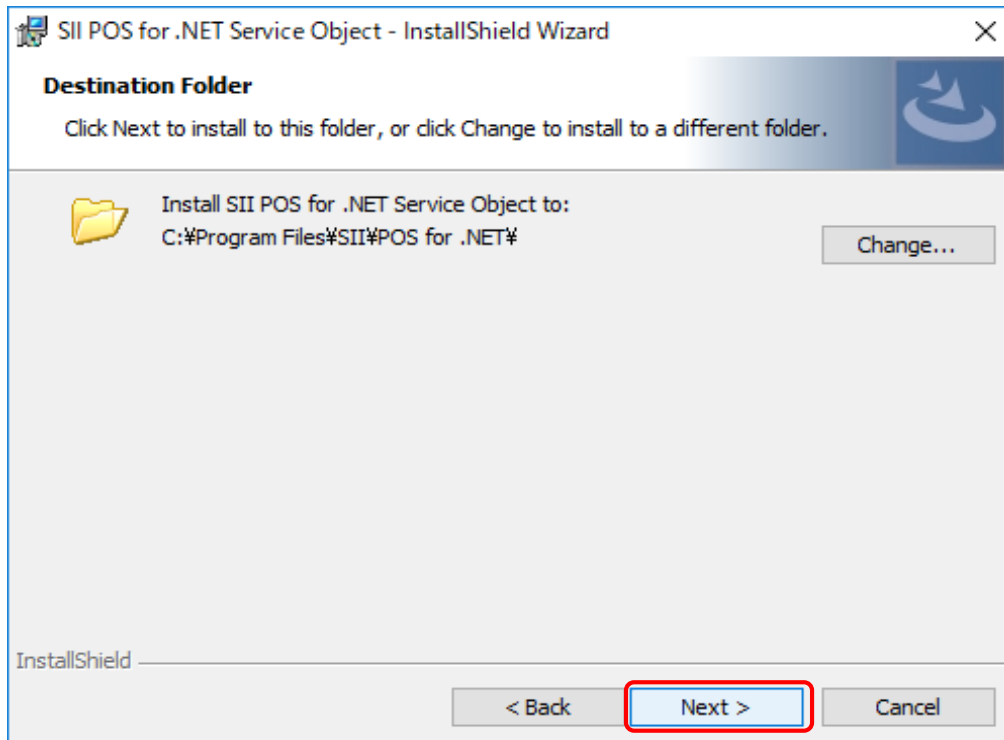
For the installation procedure of the printer driver, see "RP-D10 Series Printer Driver User's Guide" or "RP-E10 Series Printer Driver User's Guide" for the installation procedure.

This installation requires logon to the computer with administrator privileges.

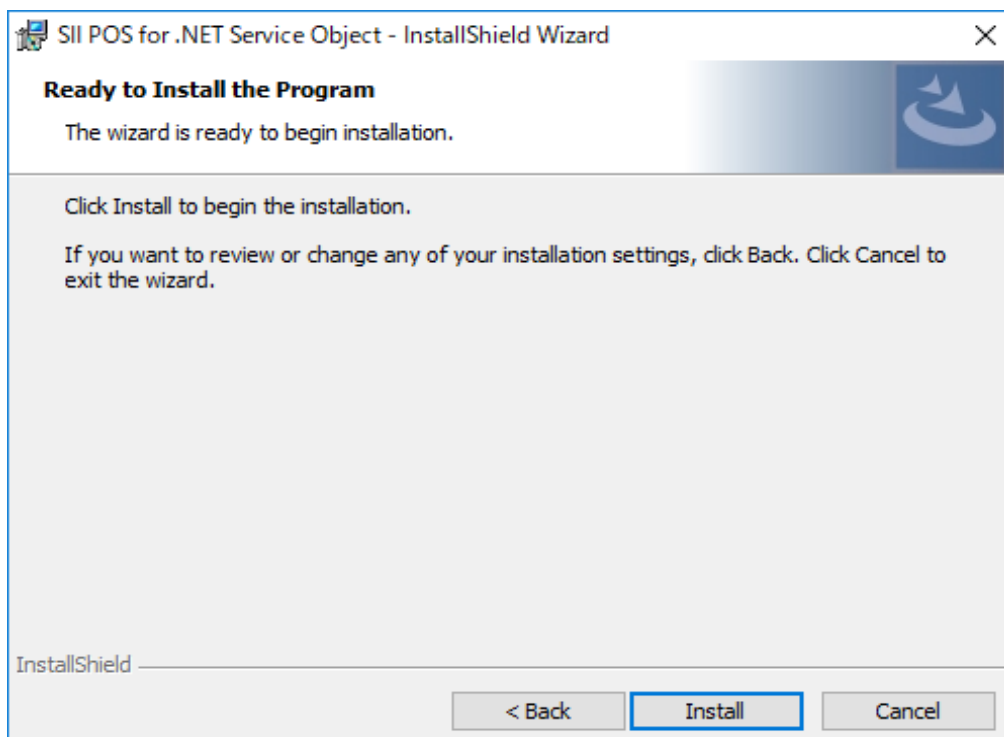
- (1) Start the setup program.
For 32-bit OS : SetupPosNet.exe
For 64-bit OS : SetupPosNet64.exe
- (2) When the installer starts up, click the [Next >] button.



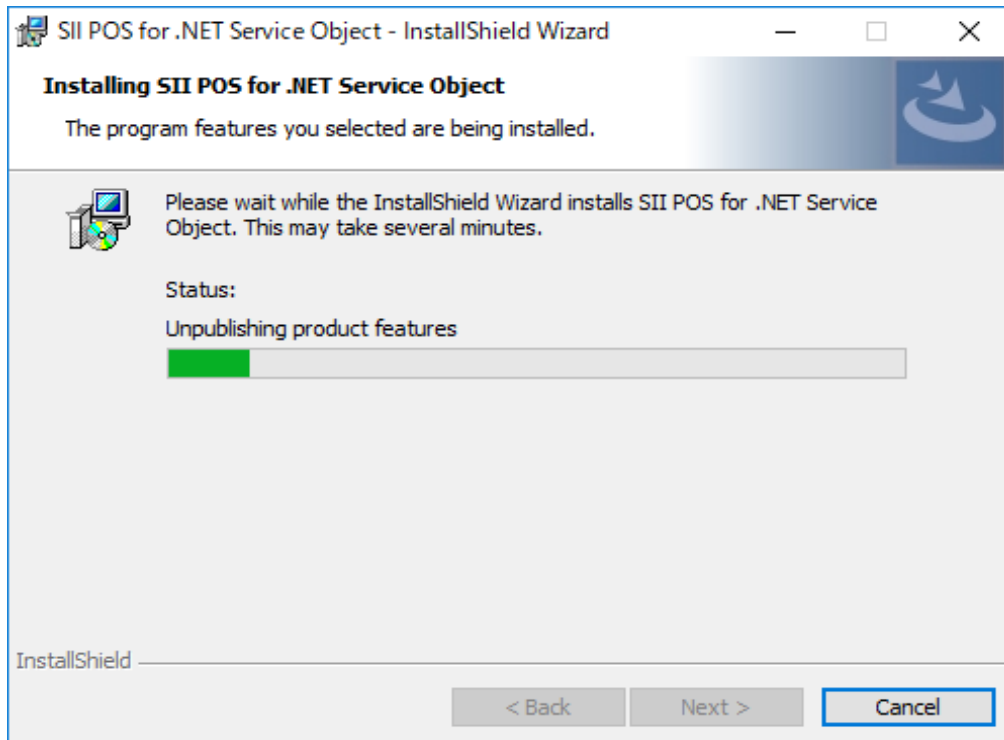
(3) Specify the destination folder and click the [Next >] button.



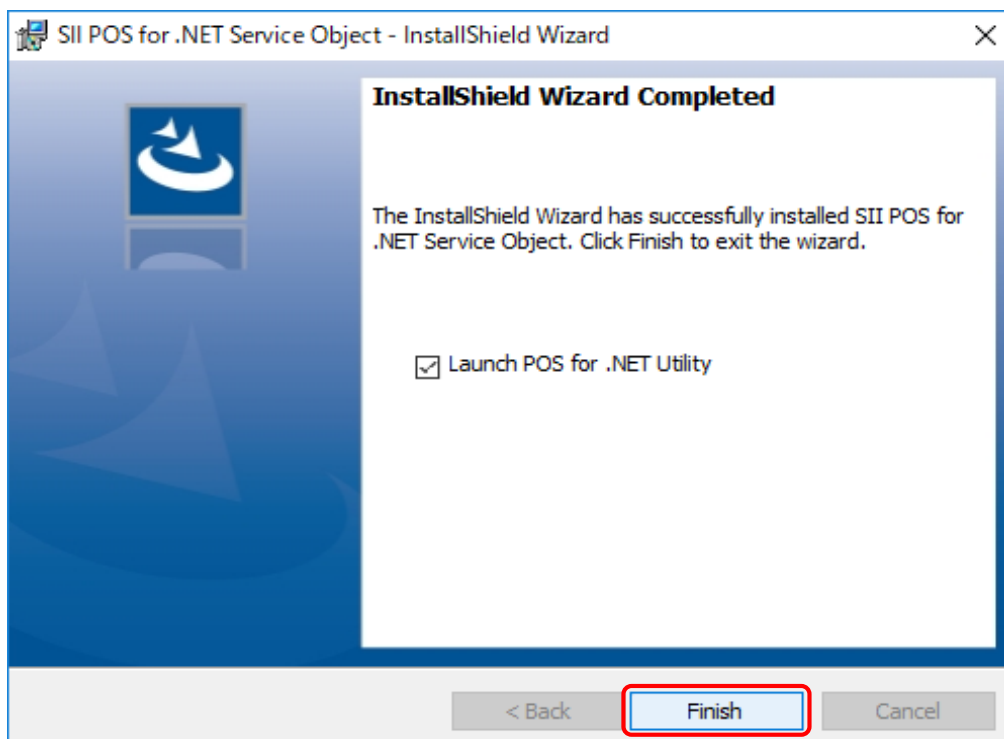
(4) Click the [Install] button.



- (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.



3.2 Configuration Program

This section describes the configuration program (SII POS for .NET Utility) provided by the software.

3.2.1 Item and Detail of Configuration Program

It is possible to set the following items by the configuration program.

Item	Detail
Device Type	Selects the type of the POS device. Either [PosPrinter] or [CashDrawer] is available.
Device Name	Indicates the name of the POS device supported by the software. By selecting the [Device Type], specify the following name depending on the using printer type. [PosPrinter]: RP-D10: [RP-D10 POS Printer] RP-E10: [RP-E10 POS Printer] [CashDrawer]: Cash drawer connected to RP-D10: [RP-D10 Cash Drawer] Cash drawer connected to RP-E10: [RP-E10 Cash Drawer]
HardwarePath	Indicates the hardware path. It is impossible to modify.
Logical Name	Sets the logical name. It is possible to set an arbitrary name. However, it is impossible to set the same logical name for more than one Service Object.
Printer Driver	Selects the printer driver. It is impossible to set the printer driver which is already set in another PosPrinter device.
Send Timeout(ms)	Sets the communication send timeout values with the POS printer. It is possible to set a value from 3000 to 60000 (msec).
Receive Timeout(ms)	Sets the communication receive timeout values with the POS printer. It is possible to set a value from 3000 to 60000 (msec).
Process Completion Timing	Selects the timing for print completion of the POS printer. It is possible to select either [Data transmission] or [Data printing].
Character Set	Selects the type of character set of the POS printer. It is possible to select any of the following character sets. [Code Page437] [Code Page932] [Code Page850] [CharacterSetAnsi] [Code Page852] [Code Page1250] [Code Page858] [Code Page1251] [Code Page860] [Code Page1252] [Code Page863] [Code Page1253] [Code Page865] [Code Page1254] CharacterSet is initialized with this value. See " CharacterSet Property" for details.
Number of Effective Dots(dots)	Selects the number of dots per line of the POS printer. It is possible to select the number of effective dots from any of the following: 576, 512, 432 or 360. RecLineWidth is initialized with this value.

Item	Detail
Number of Characters per Line	<p>Selects the number of 1-byte character per line of the POS printer.</p> <p>It is possible to select any of the following values, depending on the number of effective dots.</p> <p>Number of effective dots, 576: 36,41,44,48,57,64,72</p> <p>Number of effective dots, 432: 27,30,33,36,43,48,54</p> <p>Number of effective dots, 512: 42,56</p> <p>Number of effective dots, 360: 30,40</p> <p>RecLineCharsList is initialized with this value.</p>
Line Spacing(dots)	<p>Selects the line spacing of the POS printer.</p> <p>A value from 16 to 255 is available.</p> <p>The possible minimum value depends on the number of effective dots and the number of characters per line.</p> <ul style="list-style-type: none"> The possible minimum value is 24 when: <p>Number of effective dots, 576:</p> <p>Number of characters per line: 36,41,44,48</p> <p>Number of effective dots, 432:</p> <p>Number of characters per line: 27,30,33,36</p> <p>Number of effective dots, 512:</p> <p>Number of characters per line: 42</p> <p>Number of effective dots, 360:</p> <p>Number of characters per line: 30</p> The possible minimum value is 16 when: <p>Number of effective dots, 576:</p> <p>Number of characters per line: 57,64,72</p> <p>Number of effective dots, 432:</p> <p>Number of characters per line: 43,48,54</p> <p>Number of effective dots, 512:</p> <p>Number of characters per line: 56</p> <p>Number of effective dots, 360:</p> <p>Number of characters per line: 40</p> <p>RecLineSpacing is initialized with this value.</p>
NearEnd Sensor	<p>Selects enable/disable for near end sensor of the POS printer. It is possible to select enable only when the RP-E10 POS Printer is selected for Device Name.</p>
Print Speed	<p>Selects the speed setting of the POS printer.</p> <p>The speed setting is available to select from the followings.</p> <ul style="list-style-type: none"> RecLetterQuality Enable High Middle(Quality) Middle(Silent) Low <p>If RecLetterQuality Enable is selected, the speed setting of the POS printer is defined by RecLetterQuality.</p> <p>If other than RecLetterQuality Enable is selected, RecLetterQuality is ignored and specified speed setting is defined.</p>
Drawer Number	<p>Shows the cash drawer number connected to the POS printer.</p> <p>The number cannot be changed.</p>
POS printer Logical Name	<p>Indicates the logical name of POS printer which connects Drawer.</p> <p>Two cash drawers can be connect to one POS printer.</p>
Sensor status when drawer is open	<p>Selects the drawer sensor status when the drawer is open.</p> <p>It is possible to select either "High" or "Low".</p>

Item	Detail
Pulse time(ms)	Sets the pulse time for the cash drawer signal. A value from 100 to 800 (msec) is available.

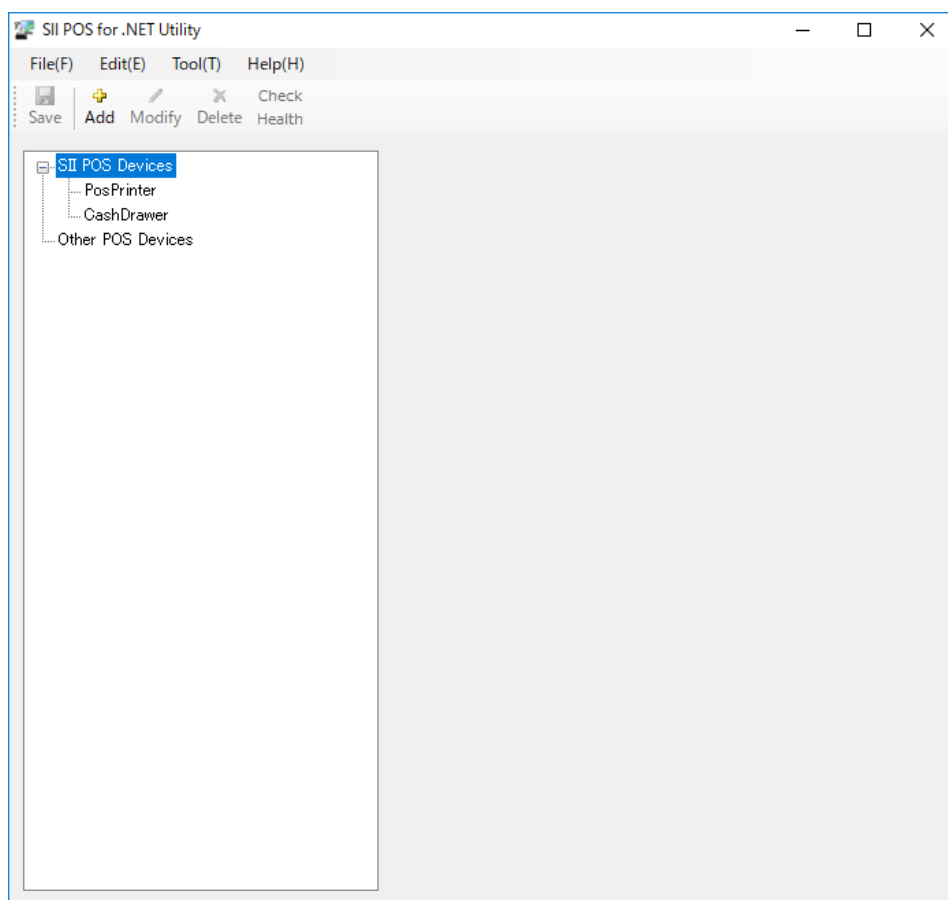
3.2.2 Startup Configuration Program

The startup procedure of the configuration program is described.

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

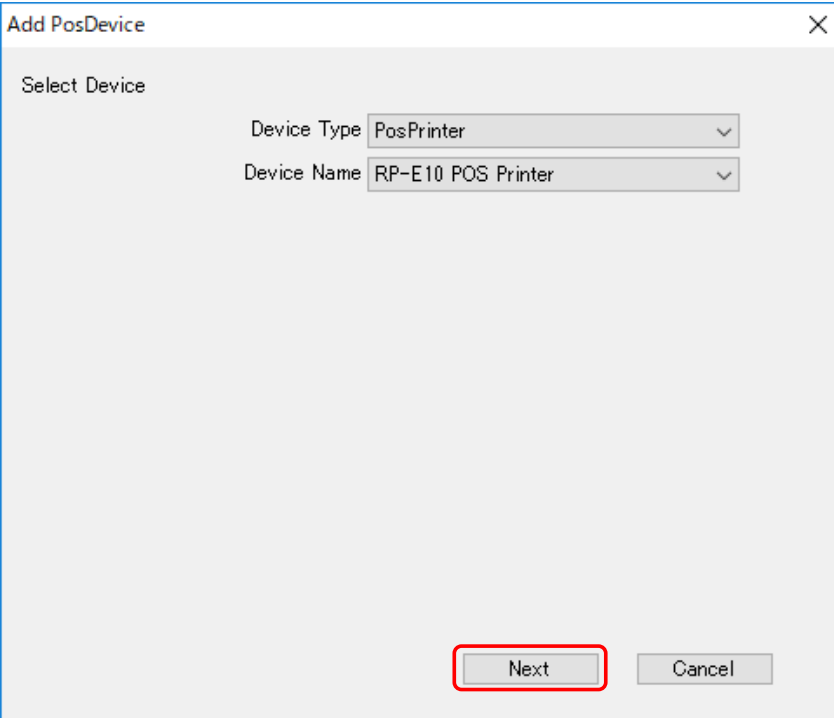
3.2.3 Configuration Program Function

When the configuration program starts, the following window is displayed.



(1) Addition of device

- 1) When [Add/Modify of Device] wizard is displayed, select [Device Type] and [Device Name], then click the [Next] button. [Device Name] drop-down list changes depending on the selected [Device Type]. See "Device Correspondence Table for PosPrinter and CashDrawer".

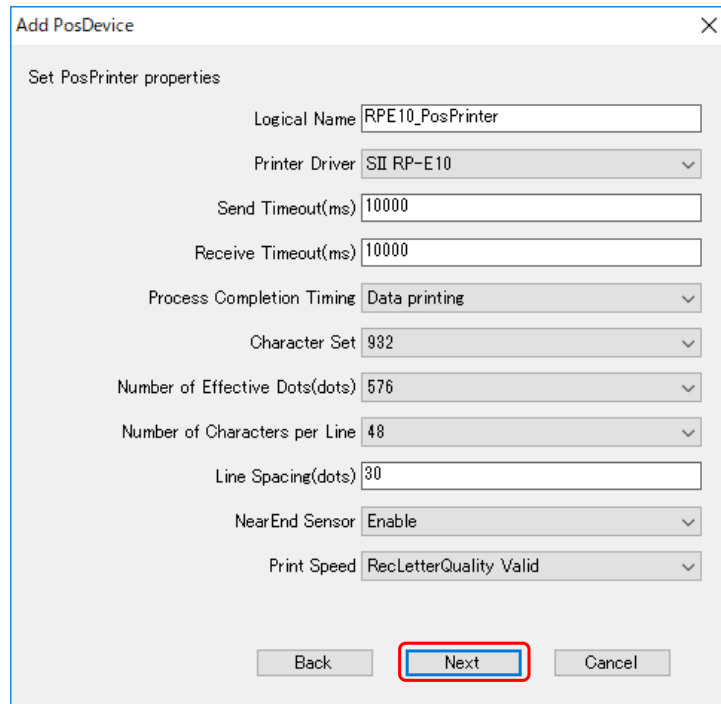


Device Correspondence Table for PosPrinter and CashDrawer

Device Type	RP-D10	RP-E10
PosPrinter	RP-D10 POS Printer	RP-E10 POS Printer
CashDrawer	RP-D10 Cash Drawer	RP-E10 Cash Drawer

2) Enter or select settings.

- (a) When [Device Type] is [PosPrinter]:
Enter or select settings, and then click the [Next] button.



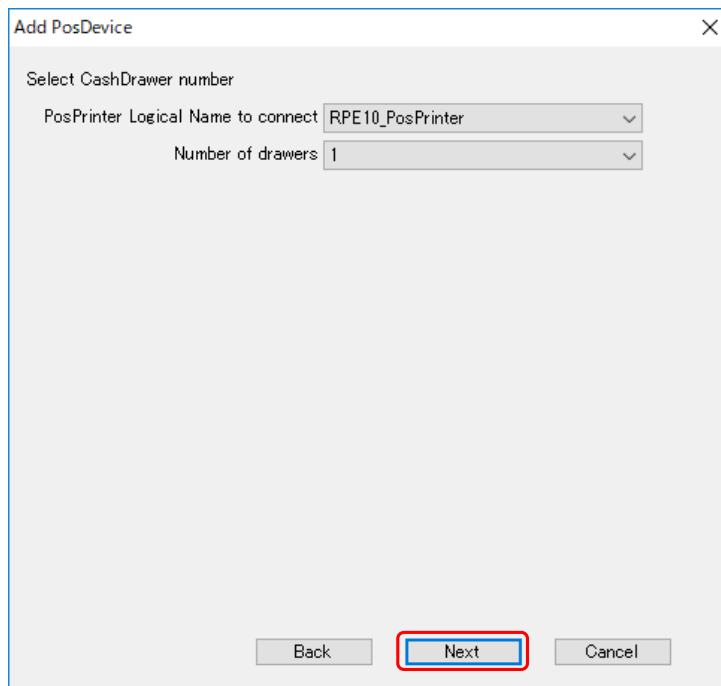
The screenshot shows the 'Add PosDevice' dialog box with the title bar 'Add PosDevice' and a close button. The main section is titled 'Set PosPrinter properties'. It contains the following fields and controls:

- Logical Name: Text box containing 'RPE10_PosPrinter'.
- Printer Driver: Dropdown menu showing 'SII RP-E10'.
- Send Timeout(ms): Text box containing '10000'.
- Receive Timeout(ms): Text box containing '10000'.
- Process Completion Timing: Dropdown menu showing 'Data printing'.
- Character Set: Dropdown menu showing '932'.
- Number of Effective Dots(dots): Dropdown menu showing '576'.
- Number of Characters per Line: Dropdown menu showing '48'.
- Line Spacing(dots): Text box containing '30'.
- NearEnd Sensor: Dropdown menu showing 'Enable'.
- Print Speed: Dropdown menu showing 'RecLetterQuality Valid'.

At the bottom, there are three buttons: 'Back', 'Next' (highlighted with a red rectangle), and 'Cancel'.

(b) When [Device Type] is [CashDrawer]:

- When adding 1 CashDrawer
Select "1" for [Number of drawers], and click the [Next] button.
- When adding 2 CashDrawers
Select "2" for [Number of drawers], and click the [Next] button.



The screenshot shows the 'Add PosDevice' dialog box with the title bar 'Add PosDevice' and a close button. The main section is titled 'Select CashDrawer number'. It contains the following fields and controls:

- PosPrinter Logical Name to connect: Dropdown menu showing 'RPE10_PosPrinter'.
- Number of drawers: Dropdown menu showing '1'.

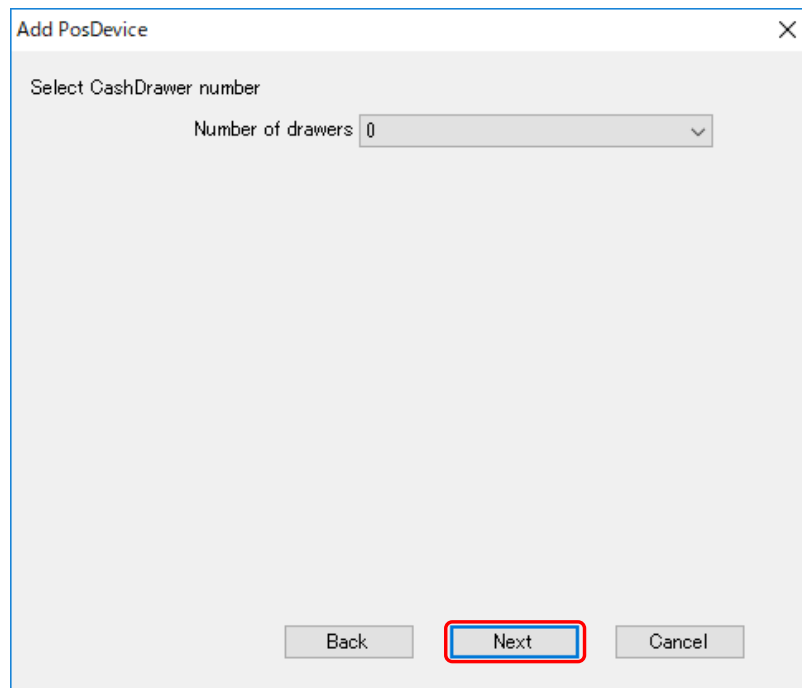
At the bottom, there are three buttons: 'Back', 'Next' (highlighted with a red rectangle), and 'Cancel'.

3) When setting item is displayed, enter or select each setting.

(a) When [Device Type] is [PosPrinter]:

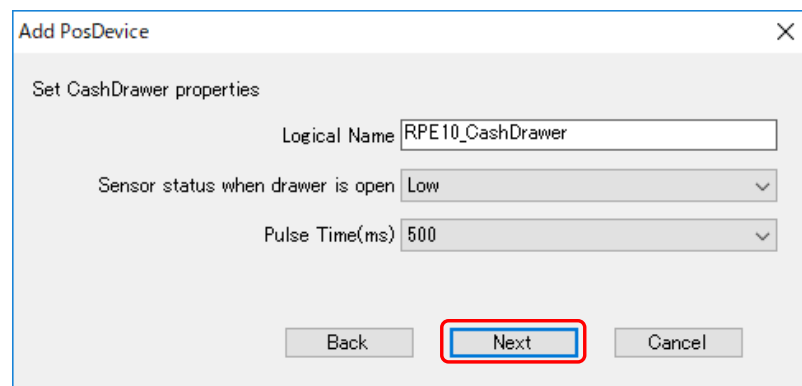
Select the number of CashDrawers to connect to PosPrinter from [Number of drawers].

- When adding PosPrinter only:
Select "0" for [Number of drawers], and click the [Next] button.
- When adding 1 CashDrawer in addition to PosPrinter:
Select "1" for [Number of drawers], and click the [Next] button.
See the description "(b) When [Device Type] is [CashDrawer]" for the subsequent setting.
- When adding 2 CashDrawers in addition to PosPrinter:
Select "2" for [Number of drawers], and click the [Next] button.
See the description "(b) When [Device Type] is [CashDrawer]" for the subsequent setting.

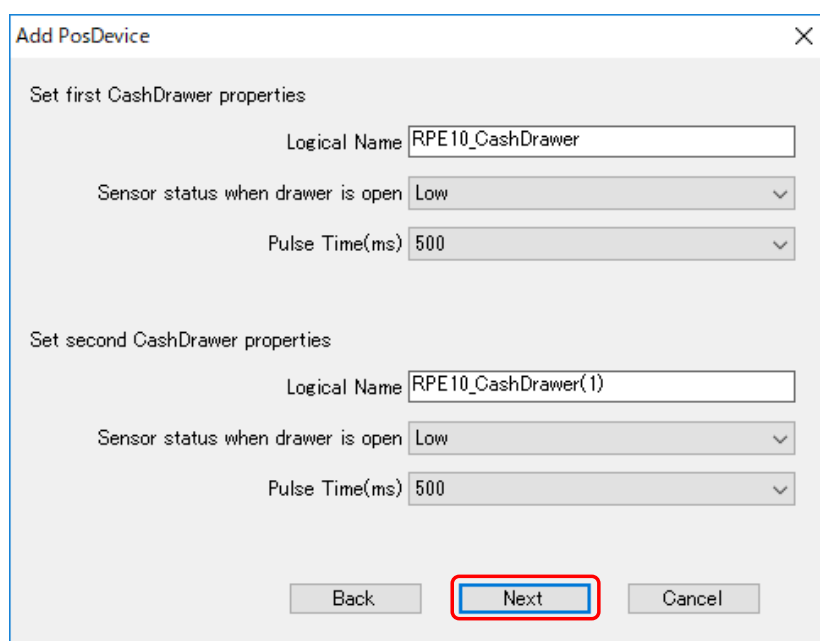


(b) When [Device Type] is [CashDrawer]:

- When adding 1 CashDrawer
Enter or select settings of CashDrawer, and click the [Next] button.

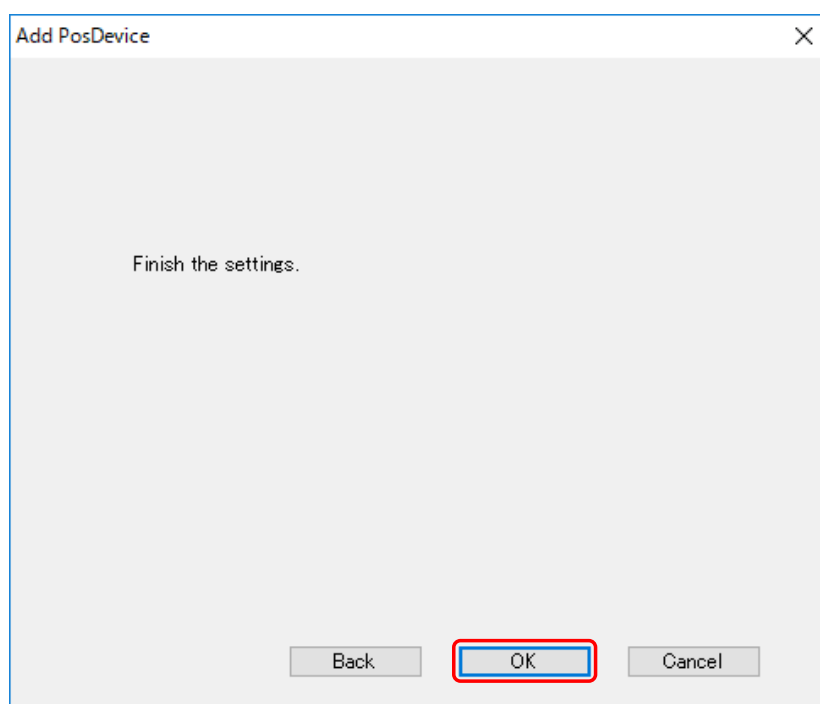


- When adding 2 CashDrawers
Enter or select settings of CashDrawer, and click the [Next] button.



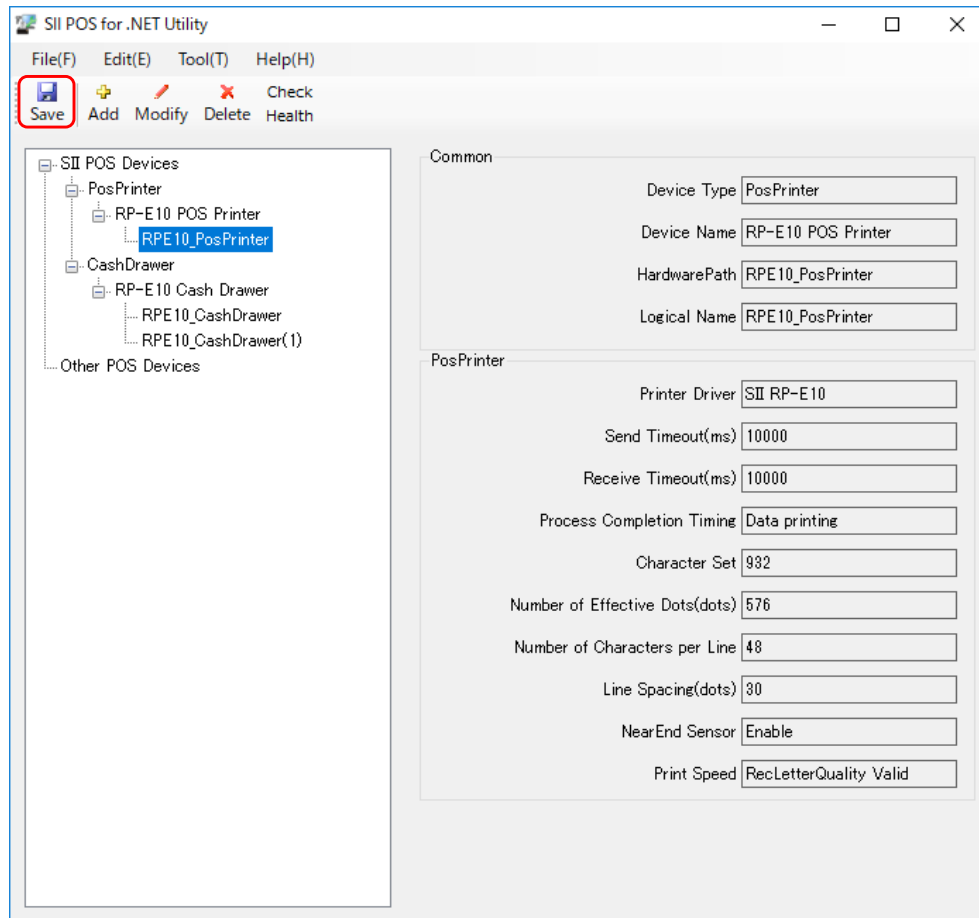
The screenshot shows a dialog box titled "Add PosDevice" with a close button (X) in the top right corner. It contains two sections for configuring cash drawers. The first section, "Set first CashDrawer properties", includes a text field for "Logical Name" with the value "RPE10_CashDrawer", a dropdown for "Sensor status when drawer is open" set to "Low", and a dropdown for "Pulse Time(ms)" set to "500". The second section, "Set second CashDrawer properties", includes a text field for "Logical Name" with the value "RPE10_CashDrawer(1)", a dropdown for "Sensor status when drawer is open" set to "Low", and a dropdown for "Pulse Time(ms)" set to "500". At the bottom, there are three buttons: "Back", "Next", and "Cancel". The "Next" button is highlighted with a red rectangular box.

- 4) Click the [OK] button.



The screenshot shows the same "Add PosDevice" dialog box, but the configuration fields are now disabled. The text "Finish the settings." is displayed in the center of the dialog. At the bottom, the buttons are "Back", "OK", and "Cancel". The "OK" button is highlighted with a red rectangular box.

- 5) Click the [Save] button after specifying each setting.



(2) Modification/Deletion of device

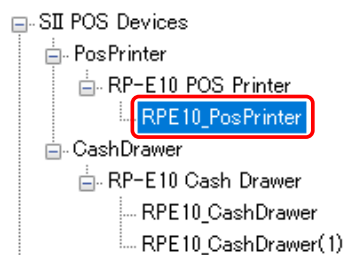
The added device can be modified or deleted by using [Modify] or [Delete] button.

(3) 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.

- 1) Select the logical name of PosPrinter or CashDrawer for interactive test from "Device View".



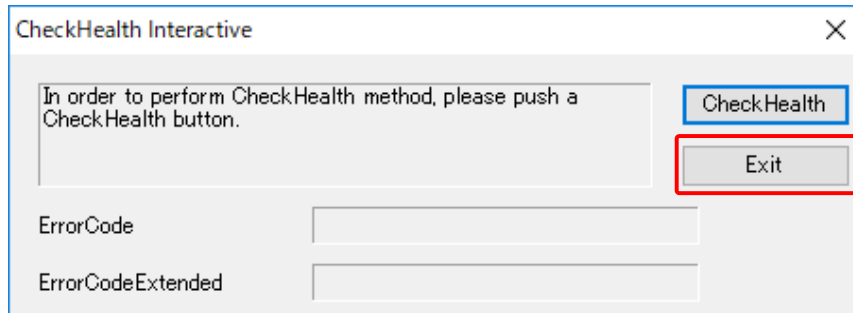
- 2) Click the [CheckHealth] button in "Tool Bar".



- 3) The preparation for the interactive test is started.

[When the preparation for the interactive test succeeded]

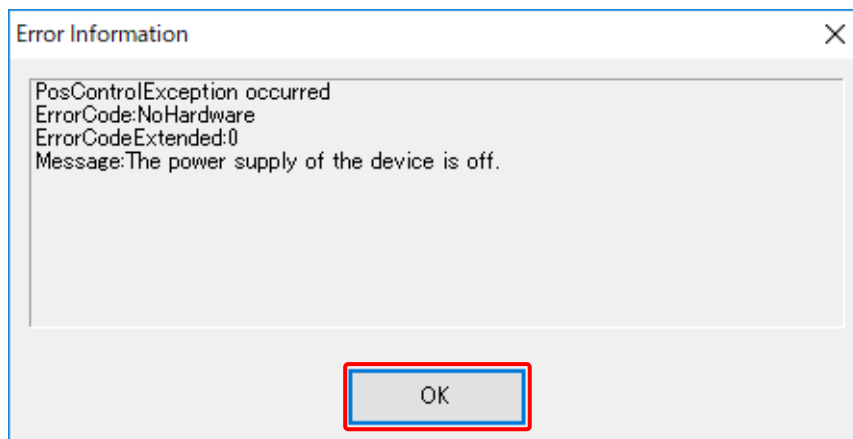
- 4) The CheckHealth Interactive dialogue to perform the interactive test is displayed.



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]

- 4) The Error Information dialogue is displayed.

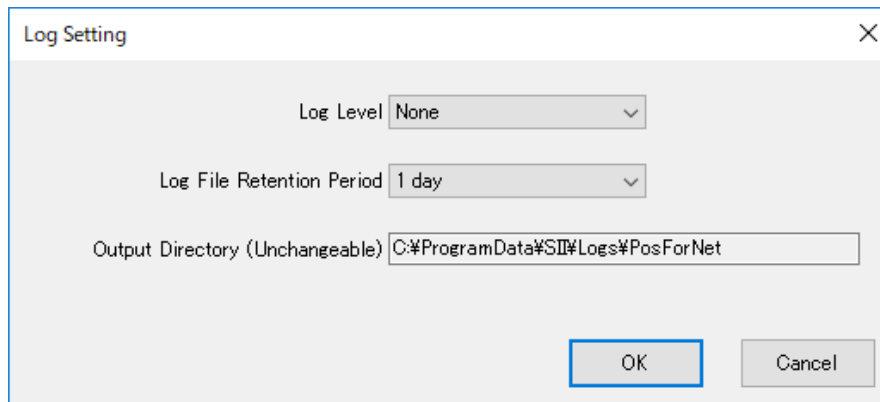


Confirm ErrorCode displayed in the dialogue. See "Appendix A Exceptions" for ErrorCode.
Click the [OK] button after confirming ErrorCode.

(4) 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

- Log setting procedure

The log setting procedure is described below.

1. Select [Tool] – [LogSetting] from "Menu Bar".
2. Select the log level to output from [Log Level].
3. Select the log file retention period from [Log File Retention Period], and click the [OK] button.
4. Click the [Save] button in the main screen. The log settings will be applied from the next **Open**.

3.3 Uninstallation

Use the following procedure to uninstall.

Click "Uninstall a program" in the control panel. When [Programs and Features] is displayed, click the [Uninstall] button for the SII POS for .NET Service Object.

Chapter 4 Properties, Methods, and Events

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	""
Claimed	bool	R	Open	<i>false</i>
DeviceDescription	string	R	Open	See the List of common property default depending on the configuration setting
DeviceEnabled	bool	R/W	Open & Claim	<i>false</i>
DeviceName	string	R	Open	See the List of common property default depending on the configuration setting
FreezeEvents	bool	R/W	Open	<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	See the List of common property default depending on the configuration setting
ServiceObjectVersion	Version	R	Open	1.12.x.x
State	ControlState	R	Open	<i>Idle</i>
SynchronizingObject	System.ComponentModel.ISynchronizeInvoke	R/W	Open	Depends on the application

List of common property default depending on the configuration setting

Property Name	RP-D10 POS Printer	RP-E10 POS Printer
DeviceDescription	"SII RP-D10 POS Printer"	"SII RP-E10 POS Printer"
DeviceName	"RP-D10 POS Printer"	"RP-E10 POS Printer"
ServiceObjectDescription	"SII RP-D10 POS Printer Service Object, Copyright(C) 20xx Seiko Instruments Inc."	"SII RP-E10 POS Printer Service Object, Copyright(C) 20xx Seiko Instruments Inc."

(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
CapConcurrentJrnRec	bool	R	Open	false
CapConcurrentJrnSlp	bool	R	Open	false
CapConcurrentPageMode	bool	R	Open	false
CapConcurrentRecSlp	bool	R	Open	false
CapCoverSensor	bool	R	Open	true
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
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
CapRecNearEndSensor	bool	R	Open	false ^{*1}
				true ^{*2,*3}
CapRecPageMode	bool	R	Open	true
CapRecPaperCut	bool	R	Open	true
CapRecPresent	bool	R	Open	true
CapRecRight90	bool	R	Open	true

Property Name	Type	Access	Availability Condition	Default
CapRecRotate180	bool	R	Open	<i>true</i>
CapRecStamp	bool	R	Open	<i>false</i>
CapRecUnderline	bool	R	Open	<i>true</i>
CapSlp2Color	bool	R	Open	<i>false</i>
CapSlpBarCode	bool	R	Open	<i>false</i>
CapSlpBitmap	bool	R	Open	<i>false</i>
CapSlpBold	bool	R	Open	<i>false</i>
CapSlpBothSidesPrint	bool	R	Open	<i>false</i>
CapSlpCartridgeSensor	PrinterCartridgeSensors	R	Open	<i>None</i>
CapSlpColor	PrinterColors	R	Open	<i>None</i>
CapSlpDHigh	bool	R	Open	<i>false</i>
CapSlpDWide	bool	R	Open	<i>false</i>
CapSlpDWideDHigh	bool	R	Open	<i>false</i>
CapSlpEmptySensor	bool	R	Open	<i>false</i>
CapSlpFullSlip	bool	R	Open	<i>false</i>
CapSlpItalic	bool	R	Open	<i>false</i>
CapSlpLeft90	bool	R	Open	<i>false</i>
CapSlpNearEndSensor	bool	R	Open	<i>false</i>
CapSlpPageMode	bool	R	Open	<i>false</i>
CapSlpPresent	bool	R	Open	<i>false</i>
CapSlpRight90	bool	R	Open	<i>false</i>
CapSlpRotate180	bool	R	Open	<i>false</i>
CapSlpUnderline	bool	R	Open	<i>false</i>
CapTransaction	bool	R	Open	<i>true</i>
CartridgeNotify	PrinterCartridgeNotify	R/W ^{*4}	Open	<i>Disabled</i>
CharacterSet	int	R/W	Open, Claim, & Enable	999 ^{*3}
CharacterSetList	int[]	R	Open	{437, 850, 852, 858, 860, 863, 865, 932, 999, 1250, 1251, 1252, 1253, 1254}
CoverOpen	bool	R	Open, Claim, & Enable	Depends on printer status
ErrorLevel	PrinterErrorLevel	R	Open	<i>None</i>
ErrorStation	PrinterStation	R	Open	<i>None</i>
ErrorString	string	R	Open	""
FlagWhenIdle	bool	R/W	Open	<i>false</i>
FontTypefaceList	string[]	R	Open	[0]
JrnCartridgeState	PrinterCartridgeStates	R	Open, Claim, & Enable	<i>Unknown</i>
JrnCurrentCartridge	PrinterColors	R/W ^{*4}	Open, Claim, & Enable	<i>None</i>
JrnEmpty	bool	R	Open, Claim, & Enable	<i>false</i>

Property Name	Type	Access	Availability Condition	Default
JrnLetterQuality	bool	R/W ^{*4}	Open, Claim, & Enable	<i>false</i>
JrnLineChars	int	R/W ^{*4}	Open, Claim, & Enable	0
JrnLineCharsList	int[]	R	Open	[0]
JrnLineHeight	int	R/W ^{*4}	Open, Claim, & Enable	0
JrnLineSpacing	int	R/W ^{*4}	Open, Claim, & Enable	0
JrnLineWidth	int	R	Open, Claim, & Enable	0
JrnNearEnd	bool	R	Open, Claim, & Enable	<i>false</i>
MapCharacterSet	bool	R/W ^{*4}	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 ^{*4}	Open, Claim, & Enable	<i>Primary</i>
RecEmpty	bool	R	Open, Claim, & Enable	Depends on printer status
RecLetterQuality	bool	R/W ^{*4}	Open, Claim, & Enable	<i>false</i>
RecLineChars	int	R/W	Open, Claim, & Enable	48 ^{*3}
RecLineCharsList	int[]	R	Open	{36,41,44,48,57,64,72} ^{*5}
RecLineHeight	int	R/W	Open, Claim, & Enable	24 ^{*5}
RecLineSpacing	int	R/W	Open, Claim, & Enable	30 ^{*3}
RecLinesToPaperCut	int	R	Open, Claim, & Enable	3 ^{*1,*5}
				4 ^{*2,*5}
RecLineWidth	int	R	Open, Claim, & Enable	576 ^{*3}

Property Name	Type	Access	Availability Condition	Default
RecNearEnd	bool	R	Open, Claim, & Enable	<i>false</i> ^{*1} Depends on printer status ^{*2}
RecSidewaysMaxChars	int	R	Open, Claim, & Enable	200 ^{*5}
RecSidewaysMaxLines	int	R	Open, Claim, & Enable	19 ^{*5}
RotateSpecial	Rotation	R/W	Open	<i>Normal</i>
SlpBarCodeRotationList	Rotation[]	R	Open	[0]
SlpBitmapRotationList	Rotation[]	R	Open	[0]
SlpCartridgeState	PrinterCartridgeStates	R	Open, Claim, & Enable	<i>Unknown</i>
SlpCurrentCartridge	PrinterColors	R/W ^{*4}	Open, Claim, & Enable	<i>None</i>
SlpEmpty	bool	R	Open, Claim, & Enable	<i>false</i>
SlpLetterQuality	bool	R/W ^{*4}	Open, Claim, & Enable	<i>false</i>
SlpLineChars	int	R/W ^{*4}	Open, Claim, & Enable	0
SlpLineCharsList	int[]	R	Open	[0]
SlpLineHeight	int	R/W ^{*4}	Open, Claim, & Enable	0
SlpLinesNearEndToEnd	int	R	Open, Claim, & Enable	0
SlpLineSpacing	int	R/W ^{*4}	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

*1: When the DeviceName="RP-D10 POS Printer".

*2: When the DeviceName="RP-E10 POS Printer".

*3: Can be modified by the configuration program.

*4: Cannot be rewritten.

*5: Automatically modified by the configuration program.

(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
SetBitmap	Open, Claim, & Enable
SetLogo	Open, Claim, & Enable
TransactionPrint	Open, Claim, & Enable
ValidateData	Open, Claim, & Enable

(5) Events

Event Name	Availability Condition
DirectIOEvent	Open, Claim, & Enable ^{*1}
ErrorEvent	Open, Claim, & Enable
OutputCompleteEvent	Open, Claim, & Enable
StatusUpdateEvent	Open, Claim, & Enable

*1: 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	Cuts receipt paper. The placeholder '#' is replaced by an ASCII decimal string indicating the cut percentage. If a value greater than 100 is specified for '#', then a full cut is executed. If a value from 1 to 99 is specified, a partial cut is executed. If '#' is omitted, a full cut is executed. If '#' is 0, it is ignored. This is ignored during rotated 90° right/left mode by RotatePrint method or during page mode by PageModePrint method.
Feed and Paper cut	ESC [#]fP	Cuts receipt paper, after feeding the paper by the RecLinesToPaperCut lines. The placeholder '#' is defined by "Paper cut" escape sequence. This is ignored during rotated 90° right/left mode by RotatePrint method or during page mode by PageModePrint method.
Feed, Paper cut, and Stamp	ESC [#]sP	Not supported.
Print bitmap	ESC #B	Prints the pre-stored bitmap. The placeholder '#' is replaced by the bitmap number. If the character '#' is omitted, the data is regarded as print data instead of an escape sequence. A value from 1 to 20 can be specified for '#'. If values other than 1 to 20 are specified for '#', they are ignored.
Print top logo	ESC tL	Prints the pre-stored top logo.
Print bottom logo	ESC bL	Prints the pre-stored bottom logo.
Fire stamp	ESC sL	Not supported.
Feed lines*	ESC [#]lF	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. A value from 0 to 255 can be specified for '#'. If '#' exceeds this range, the maximum supported number (255) of lines is fed. This is ignored during rotated 90° right/left mode by RotatePrint method or during page mode by PageModePrint method.
Feed units*	ESC [#]uF	Feeds the paper forward by units in MapMode . 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 MapMode is set to <i>MapMode.Dots</i> , '#' is available from 1 to 255. If '#' exceeds this range, the maximum supported number (255) of units is fed. This is ignored during rotated 90° right/left mode by RotatePrint method or during page mode by PageModePrint method.
Feed reverse	ESC [#]rF	Not supported.

Name	Data	Remarks
Pass through embedded data	ESC #E	Sends the characters following "#E" through to the POS 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 POS Printer. If '#' is omitted, the data is regarded as print data instead of an escape sequence. 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 for the number of bytes specified by '#' is not set after the escape sequence is specified, only the available print data to send is sent. (Example: If ESC 2Ea is specified, only "a" is sent since only one byte is set for the character string.) Also, during rotated 90° right/left mode by RotatePrint method, 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.
Print in-line barcode	ESC #R	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. 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. Also, during rotated 90° right/left mode by the RotatePrint method, 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.

*: The line spacing is reduced according to the setting value of the printer function setting "Paper Saving Setting (Paper Saving)".
Therefore, the following printing processes are changed.

- The amount of feed specified by the "Feed lines" escape sequence (ESC|[#]IF) is smaller than the setting value of the **RecLineSpacing** property.
- The amount of feed specified by the "Feed units" escape sequence (ESC|[#]uF) is processed based on the setting value of the paper saving (When Mode1 is selected, paper is not fed at all).

However, the line spacing from last print line to receipt cut position is not reduced because paper is cut after executing the paper feed for saved dot lines.
Cut operation which executes the paper feed for saved dot lines is executed for "Paper cut" escape sequence (ESC|[#]P), "Feed and Paper cut" escape sequence (ESC|[#]fP), and the **CutPaper** method.

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 following R (definition of the barcode characteristics).

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

The following characters are used to indicate the attributes.

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.

Both or either of the two conditions are violated or the value outside of the range is specified for the number subsequent to each attribute, it may cause unpredictable print results.

The following example prints UPC-A with the conditions of centering, HRI string printed below the barcode, 200dots height, and 400dots width.

ESC|33Rs101h200w400a-2t-13d123456789012e

(2) Escape Sequence operated during printing

It has the characteristic that the state is kept until it is changed explicitly.

Name	Data	Remarks
Font typeface	ESC #T	Not supported.

(3) Escape Sequence operated when printing

It has the characteristic that it is reset at the end of each print method or by a "Normal" sequence.

Name	Data	Remarks
Bold	ESC !bC	Prints in bold. If '!' is specified, bold is disabled.
Underline	ESC !#[#]uC	Prints with underline. 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. If '#' is omitted, then a thickness of 1 is used for the underline. If '#' is 3 or larger, then a thickness of 2 is used. If '!' is specified, underline is disabled.
Italic	ESC !iC	Not supported.
Alternate color (Custom)	ESC [#]rC	Not supported.
Red color	ESC rC	Not supported.
Reverse video	ESC !rvC	Prints in a reverse video format. If '!' is specified, reverse video is disabled.
Shading	ESC [#]sC	Not supported.
Single high and wide	ESC 1C	Prints normal size.
Double wide	ESC 2C	Prints double-wide characters.
Double high	ESC 3C	Prints double-high characters.
Double high and wide	ESC 4C	Prints double-high/double-wide characters.
Scale horizontally	ESC #hC	A supported value for the placeholder '#' is 1 to 8. If '#' is omitted, the data is regarded as print data instead of an escape sequence. If values less than 1 are specified for '#', print in 1 scale. If values greater than 8 are specified for '#', print in 8 scale.
Scale vertically	ESC #vC	A supported value for the placeholder '#' is 1 to 8. If '#' is omitted, the data is regarded as print data instead of an escape sequence. If values less than 1 are specified for '#', print in 1 scale. If values greater than 8 are specified for '#', print in 8 scale.
RGB Color	ESC [#]fC	Not supported.

Name	Data	Remarks
Center	ESC cA	Aligns the following texts 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. This specification is ignored during rotated 90° right/left mode by RotatePrint method or during page mode by PageModePrint method.
Right justify	ESC rA	Aligns the subsequent texts 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. This specification is ignored during rotated 90° right/left mode by RotatePrint method or during page mode by PageModePrint method.
Left justify	ESC lA	Aligns the subsequent texts 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. This specification is ignored during rotated 90° right/left mode by RotatePrint method or during page mode by PageModePrint method.
Normal	ESC N	Restores printer characteristics to normal condition.
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 the **Open** method.

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>	Two types of power states, <i>PowerState.OffOffline</i> (power off or offline) and <i>PowerState.Online</i> , can be determined and reported.

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

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 the RetrieveStatistic(s) method.

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

CapUpdateFirmware Property

Type **bool**

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

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 the **Open** method.

CapUpdateStatistics Property

Type **bool**

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

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 the ResetStatistic(s) method.

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

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 the **CheckHealth** method.

The following examples show the results of diagnosis.

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

*1: In the case of *HealthCheckLevel.Interactive*, if the dialog box is closed without testing after the command is executed, "Interactive HCheck: Canceled" is set.

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

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 the **Open** method.

DeviceDescription Property

Type **string**

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

DeviceName Property	Default
"RP-D10 POS Printer"	"SII RP-D10 POS Printer"
"RP-E10 POS Printer"	"SII RP-E10 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 the **State** property is other than *ControlState.Idle*, **DeviceEnabled** property cannot be changed from *true* to *false*.

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

DeviceName Property

Type **string**

Description Gets a short string identifying the device and any pertinent information about it.
This property depends on the [Device Name] setting by the configuration program.
This property is initialized to one of the following values by the **Open** method.

POS Printer	Value
RP-D10	"RP-D10 POS Printer"
RP-E10	"RP-E10 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 FreezeEvents 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 FreezeEvents 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 the **PrintNormal** method is operated under the **AsyncMode** property is true, the **ErrorEvent** event is frozen and the **State** property turns to *ControlState.Busy*. In this case, discard the frozen event by the **ClearOutput** method or set the **FreezeEvents** property to *false* to cause **ErrorEvent**, and then execute the **Close** method, since the Service Object cannot be closed under this circumstance.

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

OutputId Property

Type **int**

Description Holds the identifier of the most recently started asynchronous output (call to an asynchronous method when the **AsyncMode** property 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 Control 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 the **Open** method.

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 StatusUpdateEvents will be fired, and the PowerState property may not be set.
<i>PowerNotification.Enabled</i>	When DeviceEnabled is set to <i>true</i> , the Service Object will fire the power notification StatusUpdateEvents and update the PowerState property. The level of functionality depends on the value of CapPowerReporting .

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

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

PowerState Property

Type **PowerState**

Description Gets the current power condition.
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">•PowerNotify is <i>PowerNotification.Disabled</i>.•DeviceEnabled is <i>false</i>.

When the Bluetooth model is used, it takes about 30 seconds to update **PowerState** to *PowerState.OffOffline* after the power condition of the device is either power off or offline. And it takes about 10 seconds to update **PowerState** to *PowerState.Online* after the power condition of the device is on and ready.

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

ServiceObjectDescription Property

Type **string**

Description Gets a string identifying the Service Object that supports the device and the company that produced it.
This property depends on the **DeviceName** property.
This property is initialized to one of the following values by the **Open** method.

DeviceName Property	Default
"RP-D10 POS Printer"	"SII RP-D10 POS Printer Service Object, Copyright (C) 20xx Seiko Instruments Inc."
"RP-E10 POS Printer"	"SII RP-E10 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 should be set to the UPOS version that the Service Object implements.
For example, when Build version is A, Revision version is B, this property is initialized "1.12.A.B" by the **Open** method.

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 ErrorEvent event handler.
<i>ControlState.Idle</i>	The device is in a good state and is not busy.

This property is always readable.

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

SynchronizingObject Property

Type **System.ComponentModel.ISynchronizeInvoke**

Description Gets or sets the object that is used to marshal the event handler calls issued because of a POS event.
This property holds an object that implements the .NET Framework class **ISynchronizeInvoke** that is used to marshal events to a particular thread. If **SynchronizingObject** is set to *null*, events are raised on a system thread owned by the Service Object.
The application using a Windows form sets the *this* pointer of the **Form** class of the main form to **SynchronizationObject**, so that events are notified to the main application thread as required by the **Form** class.
The **PosExplorer** class initializes the **SynchronizingObject** method to the value of its **SynchronizingObject** property.

4.1.4 Specific Properties

This section describes the details of the specific properties for PosPrinter.
However, the following specific properties for PosPrinter are not supported.
For details of the thrown exception errors, see "Appendix A Exceptions".

CapConcurrentJrnRec,	CapConcurrentJrnSlp,	CapConcurrentPageMode,
CapConcurrentRecSlp,	CapJrn2Color,	CapJrnBold,
CapJrnCartridgeSensor,	CapJrnColor,	CapJrnDHigh,
CapJrnDWide,	CapJrnDWideDHigh,	CapJrnEmptySensor,
CapJrnItalic,	CapJrnNearEndSensor,	CapJrnPresent,
CapJrnUnderline,	CapSlp2Color,	CapSlpBarCode,
CapSlpBitmap,	CapSlpBold,	CapSlpBothSidesPrint,
CapSlpCartridgeSensor,	CapSlpColor ,	CapSlpDHigh,
CapSlpDWide,	CapSlpDWideDHigh,	CapSlpEmptySensor,
CapSlpFullSlip,	CapSlpItalic,	CapSlpLeft90,
CapSlpNearEndSensor,	CapSlpPageMode,	CapSlpPresent,
CapSlpRight90,	CapSlpRotate180,	CapSlpUnderline,
JrnCartridgeState,	JrnCurrentCartridge,	JrnEmpty,
JrnLetterQuality,	JrnLineChars,	JrnLineCharsList,
JrnLineHeight,	JrnLineSpacing,	JrnLineWidth,
JrnNearEnd,	SlpBarCodeRotationList,	SlpBitmapRotationList,
SlpCartridgeState,	SlpCurrentCartridge,	SlpEmpty,
SlpLetterQuality,	SlpLineChars,	SlpLineCharsList,
SlpLineHeight,	SlpLinesNearEndToEnd,	SlpLineSpacing,
SlpLineWidth,	SlpMaxLines,	SlpNearEnd,
SlpPrintSide,	SlpSidewaysMaxChars,	SlpSidewaysMaxLines

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>	PrintNormal , CutPaper , PrintBarCode , PrintBitmap , PrintMemoryBitmap , RotatePrint , TransactionPrint , and PageModePrint print methods are executed synchronously.
<i>true</i>	The methods are executed asynchronously.

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

CapCharacterSet Property

Type **CharacterSetCapability**

Description Indicates the printable character setting of the PosPrinter.

The following table shows the valid property values.

Value	Meaning
<i>CharacterSetCapability.Kanji</i>	The character setting supports Code Page932, including ASCII characters 20H through 7FH and the one-byte katakana characters A1H through DFH. It also includes the Shift-JIS code characters defined by JIS Levels 1 and 2.

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

CapCoverSensor Property

Type **bool**

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

The following table shows the valid property values.

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

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

CapMapCharacterSet Property

Type **bool**

Description Gets a Boolean value that indicates that the Service Object is able to map the characters sent the application to a character set defined by the **CharacterSetList** property.
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 the CharacterSetList property.

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

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 the **Open** method.

CapRecBarCode Property

Type **bool**

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

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

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

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 the **Open** method.

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 the **Open** method.

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 the **Open** method.

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 the **Open** method.

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 the **Open** method.

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 the **Open** method.

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 the **Open** method.

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 the **Open** method.

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 the **Open** method.

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 the **Open** method.

CapRecMarkFeed Property

Type **PrinterMarkFeeds**

Description Gets a value that holds the type of mark-sensed paper handling available.

The following table shows the valid property values.

Value	Meaning
<i>PrinterMarkFeeds.None</i>	Control function for marked thermal paper of the receipt is not supported.

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

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 POS Printer doesn't have a "paper-near-end" sensor.
<i>true</i>	The POS Printer has a "paper-near-end" sensor.

This property depends on the **DeviceName** property.

When **DeviceName** property is "RP-D10 POS Printer":

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

When **DeviceName** property is "RP-E10 POS Printer":

If [Near End Sensor] is set as [Disable] by the configuration program, then this property is initialized to *false* by the **Open** method. In case of other settings, this property is initialized to *true*.

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 the **Open** method.

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>true</i>	The receipt can perform paper cuts.

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

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 the **Open** method.

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 the **Open** method.

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 the **Open** method.

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 the **Open** method.

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 the **Open** method.

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 the **Open** method.

CartridgeNotify Property R/W

Type **PrinterCartridgeNotify**

Description Gets or sets the type of PosPrinter 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 the **Open** method.

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 Page437 character set.
850	Selects Code Page850 character set.
852	Selects Code Page852 character set.
858	Selects Code Page852 character set.
860	Selects Code Page860 character set.
863	Selects Code Page863 character set.
865	Selects Code Page865 character set.
932	Selects Katakana as Code Page932 character set (Shift-JIS Code).
999	Selects Windows ANSI character set.*1
1250	Selects Code Page1250 character set.
1251	Selects Code Page1251 character set.
1252	Selects Code Page1252 character set.*1
1253	Selects Code Page1253 character set.
1254	Selects Code Page1254 character set.

*1: Windows ANSI character set is equal to Code Page1252 character set.

This property is initialized to the setting value of character set in the configuration program by the **Open** method.

CharacterSetList Property

Type **int[]**

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

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 is in the platen close status.
<i>true</i>	The printer is in the platen open status.

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** event to the application. When the error is cleared, the Service Object changes **ErrorLevel** to *PrinterErrorLevel.None*.

This property is initialized to *PrinterErrorLevel.None* by the **Open** method.

ErrorStation Property

Type **PrinterStation**

Description Gets the station that was printing when an error was detected.
The service object sets **ErrorStation** only before delivering an **ErrorEvent** event to the application.
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 initialized to *PrinterStation.None* by the **Open** method

ErrorString Property

Type **string**

Description Gets 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>ExtendedErrorVpPower</i> (1001)	VP power error occurred.
4	<i>ErrorCode.Extended</i>	<i>ExtendedErrorCutterError</i> (1002)	Cutter error.
5	<i>ErrorCode.Extended</i>	<i>ExtendedErrorCoverOpen</i> (201)	The cover is open.
6	<i>ErrorCode.Extended</i>	<i>ExtendedErrorReceiptEmpty</i> (203)	Out of receipt form.
7	<i>ErrorCode.Extended</i>	<i>ExtendedErrorHeadTemp</i> (1005)	Head temperature error.
8	<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.

The Service Object sets **ErrorString** only before delivering an **ErrorEvent** event. If no description is available, **ErrorString** will be empty string. When the error is cleared, the Service Object changes **ErrorString** to empty string.

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

FlagWhenIdle Property R/W

Type **bool**

Description Gets or sets a Boolean value that indicates whether or not to notify that **Status** turns to *ControlState.Idle*.

The following table shows the valid property values.

Value	Meaning
<i>false</i>	StatusUpdateEvent is not notified.
<i>true</i>	StatusUpdateEvent will be sent when the State property 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 the **ClearOutput** method or the event handler that receives **ErrorEvent**.

If the **State** property is already set to *ControlState.Idle* when the **FlagWhenIdle** property 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 the **Open** method.

FontTypefaceList Property

Type **string[]**

Description Gets a string array that specifies the fonts and typefaces supported by the PosPrinter. An empty array indicates that only the default font is supported.

This property is initialized to an empty string array by the **Open** method.

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 the **Open** method.

MapMode Property RW

Type **MapMode**

Description Holds the mapping mode of the POS 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>	POS 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 $\text{dot} = \text{twips} \times \text{dpi} \times k$

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

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** properties.

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

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 the **PageModeStation** property before accessing this property.
When *PrinterStation.Receipt* is specified for **PageModeStation** property, one of the following values is set to this property.

RecLineWidth Property	Value When MapMode = <i>MapMode.Dots</i>
360	<i>Point.X</i> = 360, <i>Point.Y</i> = 2400
432	<i>Point.X</i> = 432, <i>Point.Y</i> = 2400
512	<i>Point.X</i> = 512, <i>Point.Y</i> = 2400
576	<i>Point.X</i> = 576, <i>Point.Y</i> = 2400

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

PageModeDescriptor Property

Type **PageModeDescriptors**

Description Gets the basic Page mode functionality of the printer for the selected **PageModeStation** property.
This property is indicated by OR of the following values.

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

Specify *PrinterStation.Receipt* for the **PageModeStation** property before accessing this property.

When *PrinterStation.Receipt* is specified for **PageModeStation** property, OR of *PageModeDescriptors.Bitmap*, *PageModeDescriptors.BitmapRotate*, *PageModeDescriptors.BarCode*, and *PageModeDescriptors.BarCodeRotate* is set to this property.

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

PageModeHorizontalPosition Property R/W

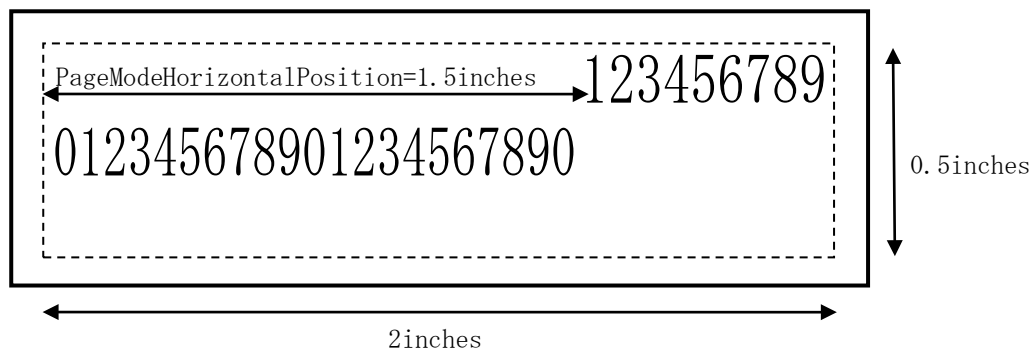
Type **int**

Description Gets or sets the horizontal start position offset within the print area for the print station specified by the **PageModeStation** property.
It expressed in the unit of measure specified by the **MapMode** property.
The horizontal direction is the same as the actual **PageModePrintDirection** property.
A read/get on this property will return the horizontal position offset set by the last write/set and not the current position.
The **PageModeStation** property 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 the **Open** method.

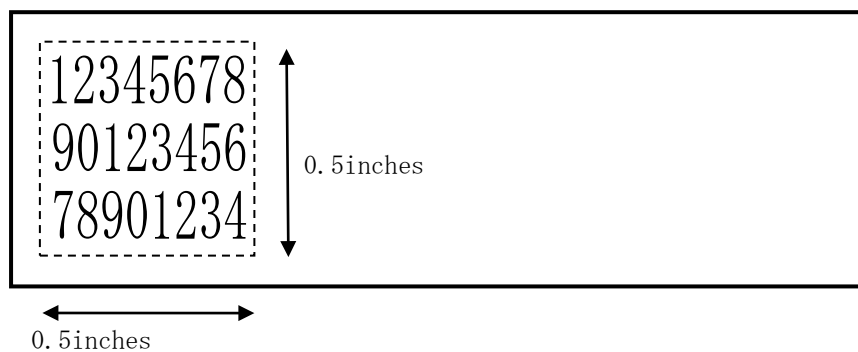
PageModePrintArea Property R/W

Type Holds the Page Mode print area for the selected **PageModeStation** property expressed in the unit specified for **MapMode** property.
The maximum possible print area is the page area.
The **PageModeStation** property 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 the **Open** method.

PageModePrintDirection Property R/W

Type **PageModePrintDirection**

Description Gets the print direction, as specified by the **PageModePrintDirection** enumeration. 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 changes the correction direction of the print start point indicated by the **PageModeHorizontalPosition** and **PageModeVerticalPosition** properties. 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.

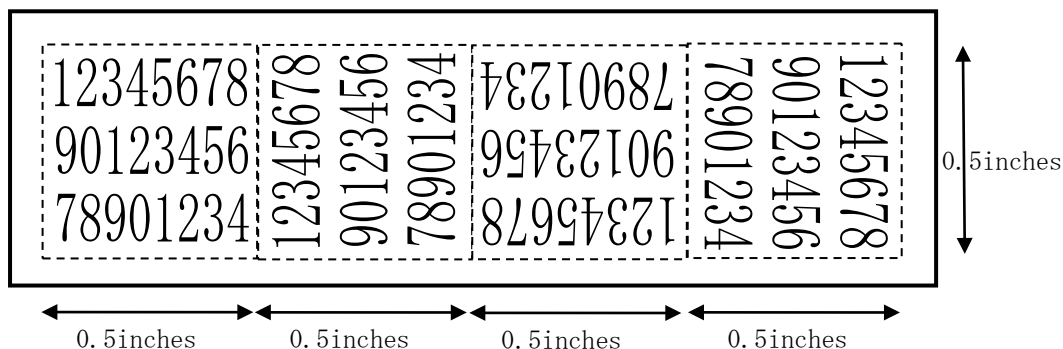
The following code sample shows the usage of **PageModePrintDirection**.

```

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);

```

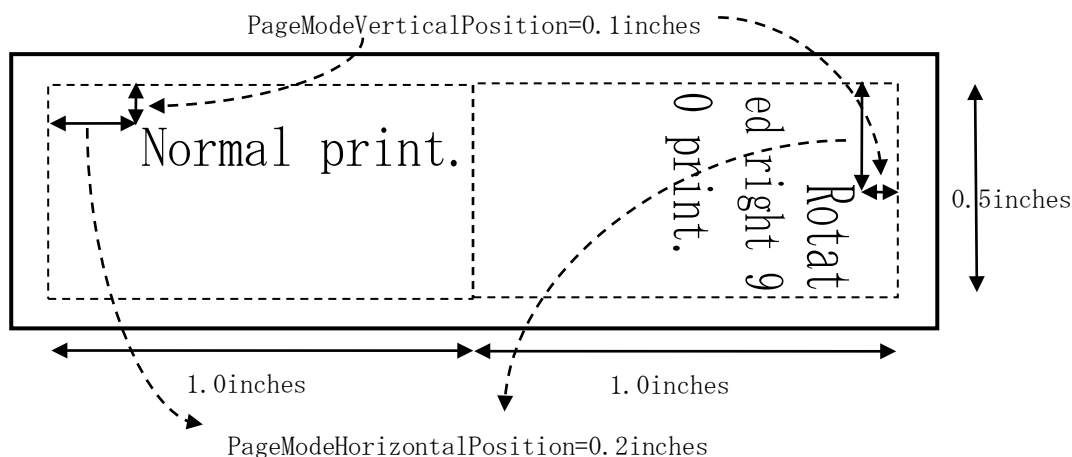
The code sample above will generate the following receipt.



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.



The **PageModeStation** property must be set to *PrinterStation.Receipt* before accessing this property, otherwise the *PageModePrintDirection.None* is returned.

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

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

PageModeStation Property R/W

Type	PrinterStation
Description	<p>Gets or sets the printer station, as specified by the PrinterStation enumeration.</p> <p>The following table shows the valid property values.</p> <p>The available station is <i>PrinterStation.Receipt</i> only.</p> <p>This property is initialized to <i>PrinterStation.None</i> by the Open method.</p> <p>Be sure to specify <i>PrinterStation.Receipt</i> for this property before accessing the property or method of the page mode function.</p>

PageModeVerticalPosition Property R/W

Type	int
Description	<p>Gets or sets the vertical start position offset within the print area for the print station specified by the PageModeStation property.</p> <p>This is expressed in the unit specified for the MapMode property.</p> <p>The vertical direction is perpendicular to the direction specified in the actual PageModePrintDirection property. 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 the PageModeStation property before accessing this property. If <i>PrinterStation.Receipt</i> is not specified, 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 the **Open** method.

RecBarCodeRotationList Property

Type **Rotation[]**

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

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

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

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 the **Open** method.

RecCartridgeState Property

Type **PrinterCartridgeStates**

Description Gets the status of the selected receipt cartridge (ink, ribbon, or toner).
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** method call.

RecCurrentCartridge Property R/W

Type **PrinterColors**

Description Gets 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** method 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 paper is out of paper.

This property is initialized while the device is enabled and keeps the current state.

When the **CoverOpen** property is true, **RecEmpty** property 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>	The prints in high-speed mode.
<i>true</i>	The prints in high-quality mode.

In the configuration setting, if the [Print Speed] is [RecLetterQualityEnable], this property is enabled and defined the [Print Speed] of POS printer according to the print mode. The [Print Speed] for the printer is set to High for high speed mode, Middle(quality) for high quality mode.

For the details about the [Print Speed], see "RP-D10 SERIES THERMAL PRINTER TECHNICAL REFERENCE", or "RP-E10 SERIES THERMAL PRINTER TECHNICAL REFERENCE"

This property is initialized to *false* while 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** property has. Depending on this property, the printer prints in the following font.

When **RecLineWidth** property is 360:

Value	RecLineChars Property	Print Font (H × W)	Character space	RecLineHeight Property
1 to 30	30	24 × 12 dots	0 dots	24
31 to 40	40	16 × 8 dots	1 dot	16

When **RecLineWidth** property is 432:

Value	RecLineChars Property	Print Font (H × W)	Character space	RecLineHeight Property
1 to 27	27	24 × 12 dots	4 dots	24
28 to 30	30	24 × 12 dots	2 dots	24
31 to 33	33	24 × 12 dots	1 dot	24
34 to 36	36	24 × 12 dots	0 dots	24
37 to 43	43	16 × 8 dots	2 dots	16
44 to 48	48	16 × 8 dots	1 dot	16
49 to 54	54	16 × 8 dots	0 dots	16

When **RecLineWidth** property is 512:

Value	RecLineChars Property	Print Font (H × W)	Character space	RecLineHeight Property
1 to 42	42	24 × 12 dots	0 dots	24
43 to 56	56	16 × 8 dots	1 dot	16

When **RecLineWidth** property is 576:

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

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

For example, when **RecLineWidth** property is 360 and "41" is set, and then an error is returned.

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

This property is initialized to the value setting by 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. This property depends on the **RecLineWidth**. This property is initialized to one of the following values by the **Open** method.

RecLineWidth Property	Value
360	30,40
432	27,30,33,36,43,48,54
512	42,56
576	36,41,44,48,57,64,72

RecLineHeight Property R/W

Type **int**

Description Gets or sets the receipt print line height.
 It expressed in the unit of measure indicated by the **MapMode** property.
 This property is automatically set by **RecLineChars**.
 The values in the following table is at the time of **MapMode** is *MapMode.Dots*.

RecLineWidth Property	RecLineChars Property	Value
360	30	24
	40	16
432	27	24
	30	
	33	
	36	
	43	16
	48	
	54	
512	42	24
	56	16
576	36	24
	41	
	44	
	48	
	57	16
	64	
	72	

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 white space between each pair of lines.
 It expressed in the unit of measure indicated by the **MapMode** property setting.
 The values in the following table is at the time of **MapMode** is *MapMode.Dots*.
 The configurable range differs depending on the setting of **RecLineWidth** property and **RecLineChars** property.
 The following table shows the valid configurable ranges.

RecLineWidth Property	RecLineChars Property	Value
360	30	24 to 255
	40	16 to 255
432	27	24 to 255
	30	
	33	
	36	
	43	16 to 255
	48	
	54	
512	42	24 to 255
	56	16 to 255
576	36	24 to 255
	41	
	44	
	48	
	57	16 to 255
	64	
	72	

When the printer function setting of RP-D10 "Paper Saving Setting (Paper Saving)" is enabled, the line spacing and space between line are the specified value in the printer function setting "Paper Saving Setting (Paper Saving)". (The value specified in **RecLineSpacing** is ignored.)

For the printer function setting "Paper Saving Setting (Paper Saving)", see "RP-D10 SERIES THERMAL PRINTER TECHNICAL REFERENCE".

For this property, the default can be changed by setting of 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. This property is determined by the following calculation based on the **DeviceName** property and **RecLineSpacing** property.

[Calculation formula]

When **DeviceName** property is "RP-D10 POS Printer"

RecLinesToPaperCut property = $88 / \text{RecLineSpacing}$ property

When **DeviceName** property is "RP-E10 POS Printer"

RecLinesToPaperCut property = $100 / \text{RecLineSpacing}$ property

Example:

When the **DeviceName** property is "RP-D10 POS Printer" and **RecLineSpacing** property is "31". (**MapMode**=*MapMode.Dots*)

RecLinesToPaperCut property = $88 / 31 = 2.93... = 3$

(The decimal part is rounded up)

This property is initialized when the device is first enabled following the **Open** method.

This property is initialized to the value based on the above calculation.

RecLineWidth Property

Type **int**

Description Gets the width of a line for the number of characters indicated by the **RecLineChars** property.

It expressed in the unit of measure indicated by the **MapMode** property setting. The values in the following table is at the time of **MapMode** is *MapMode.Dots*.

This property is initialized when the device is first enabled following the **Open** method call.

This property is initialized to one of the following values depending on the [Device Name] setting by the configuration program.

Value
360
432
512
576

For this property, the default can be changed by setting of the configuration program.

This property is initialized to one of the values shown above depending on the value set in [Number of Effective Dots (dots)] of the configuration program, when the device is enabled.

RecNearEnd Property

Type **bool**

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

Value	Meaning
<i>false</i>	The receipt paper is not low.
<i>true</i>	The receipt paper is low.

This property is initialized while the device is enabled and keeps the current state.
When the **CapRecNearEndSensor** is *false*, the value of this property is always *false*.
When the **CapRecNearEndSensor** is *true* and the **RecEmpty** property is *true*, **RecNearEnd** property always indicates *true*.

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).
This property is determined by the following calculation based on the **PageModeArea** property and print font.

[Calculation formula]

RecSidewaysMaxChars Property

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

Example:

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

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

(The decimal part is truncated.)

This property is initialized when the device is first enabled following the **Open** method call.
This property is initialized to the value based on the above calculation.

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).

This property is determined by the following calculation based on the **RecLineWidth** property, **RecLineSpacing** property and **RecLineHeight** property.

[Calculation formula]

RecSidewaysMaxLines property

= (**RecLineWidth** property – **RecLineHeight** property) / **RecLineSpacing** property + 1

Example:

When the **RecLineWidth** property is "576", **RecLineHeight** Property is "24" and **RecLineSpacing** property is "30".

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

(The decimal part is rounded down.)

This property is initialized when the device is first enabled following the **Open** method call.

This property is initialized to the value based on the above calculation.

RotateSpecial Property R/W

Type **Rotation**

Description Gets or sets the rotation orientation for bar codes.

The following table shows the valid property values.

If the rotation contains the *PrintRotation.Barcode* in the **RotatePrint** method, 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 the **Open** method.

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. The following values may be specified:

• Values of the *level* parameter

Value	Meaning
<i>HealthCheckLevel.External</i>	Executes a complete test using the device. ROM version ID of the POS printer, ServiceObjectVersion and DeviceName are printed on the printer.
<i>HealthCheckLevel.Interactive</i>	Executes an interactive test of the device. This Service Object displays the modal dialog box and prints the ROM version ID of the POS Printer, ServiceObjectVersion and DeviceName on the station specified by the device.
<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 stored in the **CheckHealthText** property.
The **CheckHealth** method is always executed synchronously.

Claim Method

Syntax **void Claim(int *timeout*);**

Parameter	Meaning
<i>timeout</i>	Specifies the maximum waiting time (in millisecond) 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 the exclusive access to the device.
The PosPrinter device cannot be used until the exclusive access is obtained.
When it is successful, the **Claimed** property 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 that **FreezeEvents** to be set to *false* – are also cleared.

When the Bluetooth model is used, it takes about 10 seconds before device is ready for use after **ClearOutput**.

Close Method

Syntax **void Close();**

Description Releases the device and its resources.
If the **DeviceEnabled** property is true, the device is first disabled.
If the **Claimed** property 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

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

Description This method is not supported.

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 values vary depending on the command number and what the Service Object sends.

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

- **Remaining memory capacity response**

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

Issues the printer command "Remaining User Area Response" and returns its response as a numeric value.

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

- **Status response**

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

Returns the paper sensor status in a numeric value.

Value	RP-D10	RP-E10*1
0	Paper is ready.	Paper is ready./"the paper is low" undetected.
1	No paper.	No paper./"the paper is low" undetected.
2	-	Paper is ready./"the paper is low" detected.
3	-	No paper./"the paper is low" detected.

*1: When the NearEnd Sensor is disabled, the value of "the paper is low" undetected. is always returned.

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

- **International character selection**

Parameter	Description
<i>command</i>	201
<i>data</i>	International character number n $0 \leq n \leq 12$
<i>obj</i>	<i>null</i>

Selects the international character.

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

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

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

The following international characters are supported.

0: USA	7: Spain I
1: France	8: Japan
2: Germany	9: Norway
3: United Kingdom	10: Denmark II
4: Denmark I	11: Spain II
5: Sweden	12: Latin American
6: Italy	

Open Method

Syntax **void Open();**

Description Opens the device.

When the **Open** method is successful, the common property and other class-specific properties are initialized.

When the Bluetooth model is used, wait at least 15 seconds after **Close** method before **Open** method.

Release Method

Syntax **void Release();**

Description Releases the exclusive access to the device.

If the **DeviceEnabled** property 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.

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

ResetStatistics(string[]) Method

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

RetrieveStatistic(string) Method

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

RetrieveStatistics() Method

Syntax	string RetrieveStatistics();
Description	Retrieves all the device statistics. 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.

RetrieveStatistics(StatisticCategories) Method

Syntax	string RetrieveStatistics(StatisticCategories <i>statistics</i>);
Description	Retrieves the statistics of the specified category. The <i>statistics</i> parameter stores the categories of the statistics to be retrieved by the application. 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.

RetrieveStatistics(string[]) Method

Syntax	string RetrieveStatistics(string[] <i>statistics</i>);
Description	Retrieves the specified device statistics. For the <i>statistic</i> parameter, 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

Syntax	void UpdateFirmware(string <i>firmwareFileName</i>);
Description	This method is not supported.

UpdateStatistic Method

Syntax	void UpdateStatistic(string <i>name</i> , object <i>value</i>);
Description	This method is not supported.

UpdateStatistics(Statistic[]) Method

Syntax	void UpdateStatistics(Statistic[] <i>statistics</i>);
Description	This method is not supported.

UpdateStatistics(StatisticCategories, Object) Method

Syntax	void UpdateStatistics(StatisticCategories <i>statistics</i> , object <i>value</i>);
Description	This method is not supported.

4.1.6 Specific Methods

This section describes the details of the specific methods for PosPrinter. However, the following specific methods for PosPrinter are not supported. For details of the thrown exception errors, see "Appendix A Exceptions".

BeginInsertion, EndRemoval,	BeginRemoval, MarkFeed,	ChangePrintSide, PrintTwoNormal	EndInsertion,
--------------------------------	----------------------------	------------------------------------	---------------

ClearPrintArea Method

Syntax **void ClearPrintArea();**

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

CutPaper Method

Syntax **void CutPaper(int *percentage*);**

Parameter	Meaning
<i>percentage</i>	Specifies the percentage of the paper to be cut. The value 100 causes a full paper cut. Other values between 1 and 99 request a partial cut percentage.

Description Cuts the receipt paper.

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

Paper cut can also be executed by using "Paper cut" escape sequence (ESC|[#]P) when calling **PrintNormal** or **PrintImmediate** methods.

If printing data remains in the printer buffer, paper cut is executed after all buffered data is printed.

During rotated 90° right/left mode by **RotatePrint** method and while page mode by **PageModePrint** method is selected, paper cut is not executed even if the method is not successful.

Due to the positions of printer head and cutter, paper cut might be executed at the middle of printing data. To avoid this, call this method after feeding paper for the value of **RecLinesToPaperCut** property.

When the printer function setting "Paper Saving Setting (Ppaer Saving)" is enabled in RP-D10, the value specified in the printer function setting "Paper Saving Setting (Paper Saving)" is applied to the line spacing when the carriage return (CR) or line feed (LF) is executed (The value specified by the **RecLineSpacing** property is ignored).

However, when the "Paper cut" escape sequence (ESC|[#]P), the "Feed and Paper cut" escape sequence (ESC|[#]fP), or the **CutPaper** method is executed after the paper is fed by the carriage return (CR) or line feed (LF), distance from the last print line to the cut position is not reduced because paper is cut after executing the paper feed for saved dot lines.

See "RP-D10 SERIES THERMAL PRINTER TECHNICAL REFERENCE" for details.

PageModePrint Method

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

Parameter	Meaning
<i>control</i>	Specifies the control type of page mode.

·Values of the *control* parameter

Value	Meaning
<i>PageModePrintControl.Cancel</i>	Clear the page and exit the Page Mode without any printing of any print area.
<i>PageModePrintControl.Normal</i>	Print the print area and destroy the canvas and exit Page Mode
<i>PageModePrintControl.PageMode</i>	Enter page mode.
<i>PageModePrintControl.PrintSave</i>	Prints the print data of the page mode print area and save the data. This is used for repeated printings.

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

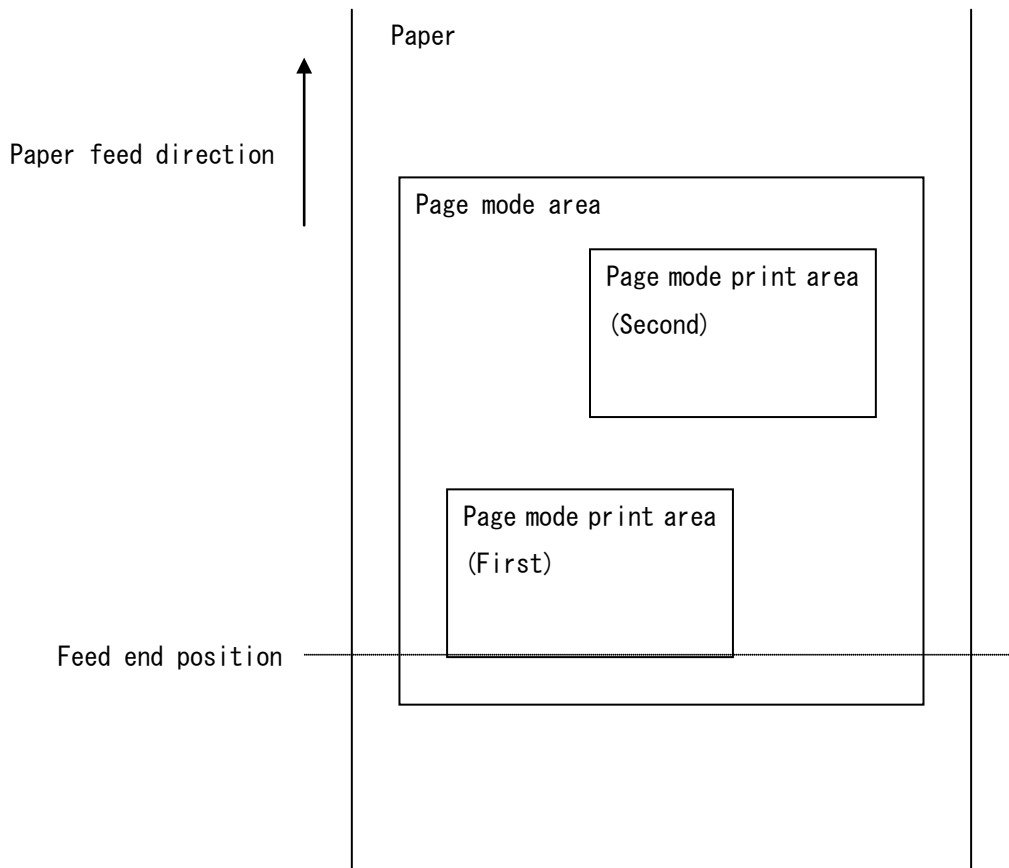
If *PageModePrintControl.PageMode* is specified for *control*, then Page Mode is started. Subsequently, the print data can be buffered with the **PrintNormal**, **PrintBarCode**, **PrintBitmap**, and **PrintMemoryBitmap** methods until the **PageModePrint** method is called by specifying *PageModePrintControl.PrintSave*, *PageModePrintControl.Normal*, or *PageModePrintControl.Cancel*. (Methods called during this time only buffers the print data and they do not initiate printing. Also, the setting of the **AsyncMode** property does not affect the page mode function: No **OutputId** will be assigned and no **OutputCompleteEvent** will 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 the **PrintNormal**, **PrintBarCode**, **PrintBitmap**, and **PrintMemoryBitmap** methods, 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 ended to return to the normal state. If some print data is buffered by one of the **PrintNormal**, **PrintBarCode**, **PrintBitmap**, and **PrintMemoryBitmap** methods, then the data is printed. The buffered data will not be saved.

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

When the **PageModePrint** method 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** is printed and the paper is fed to the end of the area. If more than one page mode print area is defined, then after the **PageModePrint** method is called, all of the data that is to be printed in their respective page mode print area(s) will be printed and the paper will be fed to the end of page of the page mode print area located the farthest "down" the sheet of paper. (See figure below).



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

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

Specify *PrinterStation.Receipt* for the **PageModeStation** property 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. Available station is only <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 MapMode .
<i>width</i>	Specifies the width of the barcode. Expressed in the unit given by MapMode .
<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 the *symbology* parameter

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.Itf</i>	Interleaved 2 of 5
<i>BarcodeSymbology.Other + 5</i>	QR Code (Mixed mode)
<i>BarcodeSymbology.Pdf417</i>	PDF417
<i>BarcodeSymbology.Upca</i>	UPC-A
<i>BarcodeSymbology.Upce</i>	UPC-E

·Values of the *alignment* parameter

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 MapMode .

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

·Values of the *textPosition* parameter

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.

Prints the barcode on the specified printer.

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

If the **RotateSpecial** property indicates that the barcode is rotated, the barcode is printed in rotated mode. The *height*, *width*, and *textPosition* parameters are applied to the barcode before it is rotated. For example, if *BarCodeTextPosition.Below* is specified, the barcode is rotated to the left, and then text is printed to the right of the barcode.

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 MapMode = <i>MapMode.Dots</i> : $width = ((6 \times X + 2 \times X \times N) \times D) + ((X \times N - X) \times D') + (X \times (10 \times 2 - 1))$ $(20 \times D + 2 \times D' + 38) \leq width \leq (72 \times D + 12 \times D' + 114)$ D: the number of barcode character D': the number of data character (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 of the first line must be a decimal number 103, 104, or 105, and at least one letter must follow it.
<i>width</i>	When MapMode = <i>MapMode.Dots</i> : $width = X \times ((10 \times 2) + ((D + 2) \times 11) + 2)$ $(22 \times D + 88) \leq width \leq (66 \times D + 264)$ D: the number of barcode character (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 ^{*1}	SPACE ^{*1}	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

Number	Character			Number	Character		
	CODE A	CODE B	CODE C		CODE A	CODE B	CODE C
22	6	6	22	75	VT	k	75
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"> • Code A : 0x00 to 0x5f, FNC1, FNC2, FNC3, FNC4, SHIFT, CODE B, CODE C • Code B : 0x20 to 0x7f, FNC1, FNC2, FNC3, FNC4, SHIFT, CODE A, CODE C • Code C : 0x30 to 0x39, FNC1, CODE A, CODE B
<i>width</i>	<p>When MapMode = <i>MapMode.Dots</i>: $width = X \times ((10 \times 2) + ((D + 2) \times 11) + 2)$ $1^{*1} \leq width \leq 66 \times D + 264$ 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>.</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 MapMode = <i>MapMode.Dots</i> : $width = (((X \times 7) + (X \times N \times 3)) \times (D + 2)) + ((10 \times 2 - 1) \times X)$ $(26 \times D + 90) \leq width \leq (96 \times D + 306)$ D: the number of barcode character 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 MapMode = <i>MapMode.Dots</i> : $width = X \times ((10 \times 2) + ((D + 2 + 2) \times 9) + 1)$ $(18 \times D + 114) \leq width \leq (54 \times D + 342)$ D: the number of barcode character 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 inputted, the 13th character does not affect the printing data.
<i>width</i>	When MapMode = <i>MapMode.Dots</i> : <ul style="list-style-type: none"> When 14 or 15 letters are specified $width = 138 \times X$ $276 \leq width \leq 828$ When 17 or 18 letters are specified $width = 165 \times X$ $330 \leq width \leq 990$ 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 MapMode = <i>MapMode.Dots</i> : $width = 113 \times X$ $226 \leq width \leq 678$ 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 MapMode = <i>MapMode.Dots</i> : $width = 81 \times X$ $162 \leq width \leq 486$ X: fine element width $2 \leq X \leq 6$ X is automatically set according to <i>width</i> .

[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>	<p>When MapMode = <i>MapMode.Dots</i>:</p> $width = ((D \times 2 + 1) \times X \times N + ((D \times 3) + 6 + (10 \times 2)) \times X)$ $(14 \times D + 56) \leq width \leq (54 \times D + 174)$ <p>D: the number of barcode character 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>.</p>

[PDF417]

Parameter	Limitation
<i>data</i>	00H to 7FH must follow the ASCII code and 80H to FFH must follow the extended character set of PC437 English list.
<i>width</i>	$width = (17 \times C + 69) \times X + (X \times 4)$ $(180 \leq width)$ $Height = R \times Y + (X \times 4)$ $(14 \leq Height \leq 255)$ <p>X: module width (2 to 4) Y: module height (2 to 127) C: the number of vertical column (1 to 30) R: the number of row (3 to 90)</p> <p>For the number of row and the number of vertical column, 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 the <i>width</i> and <i>height</i> parameters is selected after the number of row and the number of vertical column are determined.</p>

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

[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>	$width = (4V+17) \times M + (4M \times 2)$ $(58 \leq width)$ V: Version of QR Code (1 to 40) M: Module size (2 to 11) 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 the <i>width</i> parameter is selected after the version is determined.

QR Code model is fixed at 2 and an 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.

[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 MapMode = <i>MapMode.Dots</i> : $width = 133 \times X$ $226 \leq width \leq 678$ 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 MapMode = <i>MapMode.Dots</i> : $width = 65 \times X$ $130 \leq width \leq 390$ 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 the character which is 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	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. Available station is only <i>PrinterStation.Receipt</i> .
<i>fileName</i>	Specifies the name of bitmap file. For the supported image file, see below.
<i>width</i> ^{*1}	Specifies the print width of bitmap. See values below.
<i>alignment</i> ^{*2}	Specifies the print position of bitmap. See values below.

*1: The value is rounded out to a multiple of 8 by the Service Object.

*2: When rotation 90° right/left is specified by the **RotatePrint** or **PageModePrint** method, the setting of *alignment* parameter is disabled, and the bitmap is always printed with left justify.

· Supported bitmap file

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

•Values of the *width* parameter

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

•Values of the *alignment* parameter

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 MapMode .

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

Description Prints the bitmap on the specified station.
The highest performance cannot be achieved since the bitmap data is transferred to the POS 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 next to the bitmap.
This method is executed synchronously if **AsyncMode** is *false*, and asynchronously if **AsyncMode** is *true*.
The *width* parameter controls the transformation of bitmap data. If *width* is *PrinterBitmapAsIs*, then no transformation is executed. The bitmap is printed with 1 pixel per dot of the POS Printer.
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. Available station is only <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 device immediately.

This method attempts to print the data immediately. That is, it becomes the next printer operation. This method is called when an asynchronous output is not completed.

PrintImmediate is intended to be used under an exception condition where an 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 printing data remains 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 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.
CR & LF	Carriage return (CR) is ignored. Operation of line feed (LF) is executed once.

When the printer function setting "Paper Saving Setting (Paper Saving)" is enabled in RP-D10, the value specified in the printer function setting "Paper Saving Setting (Paper Saving)" is applied to the line spacing when the carriage return (CR) or line feed (LF) is executed (The value specified by the **RecLineSpacing** property is ignored).

However, when any of the "Paper cut" escape sequence (ESC|[#]P), the "Feed and Paper cut" escape sequence (ESC|[#]fP), or the **CutPaper** method is executed after the paper is fed by the carriage return (CR) or line feed (LF), distance from the last print line to the cut position is not reduced because paper is cut after executing the paper feed for saved dot lines.

See "RP-D10 SERIES THERMAL PRINTER TECHNICAL REFERENCE" for details.

PrintMemoryBitmap Method

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

Parameter	Meaning
<i>station</i>	Specifies the station to be used. Available station is only <i>PrinterStation.Receipt</i> .
<i>data</i>	Specifies the byte array holding the bitmap data. For the supported image file, see PrintBitmap .
<i>width</i>	Specifies the print width of bitmap. See PrintBitmap for values.
<i>alignment</i>	Specifies the print position of bitmap. See PrintBitmap 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. Available station is only <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 device.
The print data that exceeds the maximum number of characters per line is printed on the next print line.
If printing data remains 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.
CR & LF	Carriage return (CR) is ignored. Operation of line feed (LF) is executed once.

When the printer function setting "Paper Saving Setting (Paper Saving)" is enabled in RP-D10, the value specified in the printer function setting "Paper Saving (Paper Saving)" is applied to the line spacing when the carriage return (CR) or line feed (LF) is executed (The value specified by the **RecLineSpacing** property is ignored).

However, when any of the "Paper cut" escape sequence (ESC[*#*]P), the "Feed and Paper cut" escape sequence (ESC[*#*]fP), or the **CutPaper** method is executed after the paper is fed by the carriage return (CR) or line feed (LF), distance from the last print line to the cut position is not reduced because paper is cut after executing the paper feed for saved dot lines.

See "RP-D10 SERIES THERMAL PRINTER TECHNICAL REFERENCE" for details.

RotatePrint Method

Syntax

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

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

·Values of the *rotation* parameter

Value	Meaning
<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.Left90</i>	Starts rotated 90° left (counterclockwise) printing.
<i>PrintRotation.Normal</i>	End of rotated printing.
<i>PrintRotation.Right90</i>	Starts rotated 90° right (clockwise) printing.
<i>PrintRotation.Rotate180</i>	Starts rotated 180° printing (upside-down 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 the *rotation* parameter 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 POS 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 horizontal writing mode is started. Until **RotatePrint** is called with the *rotation* parameter set to *PrintRotation.Normal*, the data called by **PrintNormal** is buffered. The value of the **AsyncMode** property does not affect the operation. In other words, no **OutputId** is assigned and no **OutputCompleteEvent** is informed. 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 horizontal writing 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 valid, 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 the **RecBarCodeRotationList** and **RecBitmapRotationList** properties, respectively.

When *rotation* contains *PrintRotation.Barcode*, the **RotateSpecial** setting is ignored. Calling the **ClearOutput** method cancels the rotated print mode. Any buffered lines of sideways rotated print are also cleared. Service Object calculates so that the width in the horizontal writing mode becomes best size. The maximum width is 2400 dots in the horizontal writing 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 and barcode print by the "Print in-line barcode" escape sequence (ESC|#R) or the "Print bitmap" escape sequence (ESC|#B) is specified on the **PrintNormal** method during the rotation mode, the print data rotates regardless of whether or not *PrintRoatation.Bitmap* and *PrintRotation.Barcode* is specified in *rotation* parameter 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.
<i>fileName</i>	Specifies the name of bitmap file. If "(empty string)" is set, the bitmap setting is canceled. For the supported image file, see PrintBitmap .
<i>width</i>	Specifies the print width of bitmap. See PrintBitmap for values.
<i>alignment</i>	Specifies the print position of bitmap. See PrintBitmap for values.

Description Saves the information of the bitmap to be printed.

The bitmap can be printed by calling the **PrintNormal** or **PrintImmediate** method 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 POS 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 the *location* parameter

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 "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. Available station is only <i>PrinterStation.Receipt</i> .
<i>control</i>	Specifies the type of the transaction. See below for values.

·Values of the *control* parameter

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**, **CutPaper**, **RotatePrint**, **PrintBarCode**, **PrintBitmap**, and **PageModePrint** will buffer the print data until **TransactionPrint** is called with the *control* parameter set to *PrinterTransactionControl.Normal*.
The value of **AsyncMode** property does not affect the operation. In other words, no **OutputId** is assigned and no **OutputCompleteEvent** is informed.

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

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

Calling **ClearOutput** method 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. Available station is only <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 the **PrintNormal** or **PrintImmediate** method, this method is called to determine whether a data sequence, which 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 this method is disabled, the exception error is thrown. For details about thrown errors, see "Appendix A Exceptions".

4.1.7 Events

DirectIOEvent Event

Syntax **DirectIOEventHandler DirectIOEvent;**

Description This event is not supported.

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)	A Vp voltage error has occurred.
<i>ExtendedErrorAutocutter</i> (1002)	An autocutter error has occurred.
<i>ExtendedErrorHeadTemp</i> (1005)	A head-temperature error has occurred.
<i>ExtendedErrorFatal</i> (1010)	An unrecoverable error has occurred.

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.

When the Bluetooth model is used, it takes about 10 seconds before device is ready for use after **ErrorResponse** is set *ErrorResponse.Clear*.

OutputCompleteEvent Event

Syntax **OutputCompleteEventHandler OutputCompleteEvent;**

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

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

StatusUpdateEvent Event

Syntax **StatusUpdateEventHandler StatusUpdateEvent;**

Description Notifies the status of the device when the device encounters an important state change. The Service Object notifies the first **StatusUpdateEvent** when the device is enabled.

The *StatusUpdateEventArgs.Status* property 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>StatusReceiptNearEmpty</i> (25)	The receipt paper is low.
<i>StatusReceiptPaperOK</i> (26)	The receipt paper is prepared.
<i>StatusIdle</i> (1001)	All the asynchronous outputs finished either successfully or by being cleared. The State property is now <i>ControlState.Idle</i> . The FlagWhenIdle property must be true for this event to be notified. Then, the Service Object automatically resets the property to <i>false</i> before the event is notified.
<i>StatusPowerOnline</i> (2001)* ¹	The device is powered on and ready.
<i>StatusPowerOffOffline</i> (2004)* ¹	The device is powered off or offline.

*1: This is notified when **PowerNotify** = *PowerNotification.Enabled*.

When the Bluetooth model is used, it takes about 30 seconds to notify the *StatusPowerOffOffline*(2004) after the power condition of the device is either power off or offline.

And it takes about 10 seconds to notify the *StatusPowerOnline*(2001) after the power condition of the device is on and ready.

4.2 CashDrawer

4.2.1 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>false</i>
CapUpdateFirmware	bool	R	Open	<i>false</i>
CapUpdateStatistics	bool	R	Open	<i>false</i>
CheckHealthText	string	R	Open	""
Claimed	bool	R	Open	<i>false</i>
DeviceDescription	string	R	Open	See the List of common property default depending on the configuration setting
DeviceEnabled	bool	R/W	Open	<i>false</i>
DeviceName	string	R	Open	See the List of common property default depending on the configuration setting
FreezeEvents	bool	R/W	Open	<i>false</i>
PowerNotify	PowerNotification	R/W	Open	<i>Disabled</i>
PowerState	PowerState	R	Open	<i>Unknown</i>
ServiceObjectDescription	string	R	Open	See the List of common property default depending on the configuration setting
ServiceObjectVersion	Version	R	Open	1.12.x.x
State	ControlState	R	Open	<i>Idle</i>
SynchronizingObject	System. ComponentModel. ISynchronizeInvoke	R/W	Open	Depends on application

List of common property default depending on the configuration setting

Property Name	RP-D10 Cash Drawer	RP-E10 Cash Drawer
DeviceDescription	"SII RP-D10 Cash Drawer"	"SII RP-E10 Cash Drawer"
DeviceName	"RP-D10 Cash Drawer"	"RP-E10 Cash Drawer"
ServiceObjectDescription	"SII RP-D10 Cash Drawer Service Object, Copyright(C) 20xx Seiko Instruments Inc."	"SII RP-E10 Cash Drawer Service Object, Copyright(C) 20xx Seiko Instruments Inc."

(2) Specific Properties

Property Name	Type	Access	Availability Condition	Default
CapStatus	bool	R	Open	<i>true</i>
CapStatusMultiDrawerDetect	bool	R	Open	<i>false</i>
DrawerOpened	bool	R	Open, & Enable	Depends on cash drawer status

(3) Common Methods

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

(4) Specific Methods

Method Name	Availability Condition
OpenDrawer	Open, & Enable
WaitForDrawerClose	Open, & Enable

(5) Events

Event Name	Availability Condition
StatusUpdateEvent	Open, & Enable

4.2.2 Common Properties

This section describes the details of the common properties for CashDrawer.
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 the **Open** method.

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>	Two types of power states, <i>PowerState.OffOffline</i> (power off or offline) and <i>PowerState.Online</i> , can be determined and reported.

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

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>false</i>	No statistical data regarding the device is available.

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

CapUpdateFirmware Property

Type **bool**

Description Gets a Boolean value that indicates whether the device can accumulate and can provide statistics.

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 the **Open** method.

CapUpdateStatistics Property

Type **bool**

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

The following table shows the valid property values

Value	Meaning
<i>false</i>	None of the statistical data can be reset/updated by the application.

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

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 the **CheckHealth** method.

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> *1	Success	"Interactive HCheck: Successful"
	Fail	"Interactive HCheck: Failure"
<i>HealthCheckLevel.Internal</i>	Success	"Internal HCheck: Successful"
	Fail	"Internal HCheck: Failure"

*1: In the case of *HealthCheckLevel.Interactive*, if the dialog box is closed without testing after the command is executed, "Interactive HCheck: Canceled" is set.

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

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 the **Open** method.

DeviceDescription Property

Type **string**

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

DeviceName Property	Value
"RP-D10 Cash Drawer"	"SII RP-D10 Cash Drawer"
"RP-E10 Cash Drawer"	"SII RP-E10 Cash Drawer"

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.

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

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

DeviceName Property

Type **string**

Description Gets a short string identifying the device and any pertinent information about it.
This property depends on the [Device Name] setting by the configuration program.
This property is initialized to one of the following values by the **Open** method.

POS Printer to Which the Cash Drawer Is Connected	Value
RP-D10	"RP-D10 Cash Drawer"
RP-E10	"RP-E10 Cash Drawer"

FreezeEvents Property R/W

Type **bool**

Description Selects to or not to notify events.
This property indicates the following 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 FreezeEvents 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 FreezeEvents 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.

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

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 StatusUpdateEvents will be fired, and the PowerState property may not be set.
<i>PowerNotification.Enabled</i>	When DeviceEnabled is set to <i>true</i> , the Service Object will fire the power notification StatusUpdateEvents and update the PowerState property. The level of functionality depends on the value of CapPowerReporting .

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

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

PowerState Property

Type **PowerState**

Description Gets the current power condition.
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.
<i>PowerState.Unknown</i>	Cannot determine the device's power state due to one of the following reasons. • PowerNotify = <i>PowerNotification.Disabled</i> • DeviceEnabled = <i>false</i>

When the Bluetooth model is used, it takes about 30 seconds to update **PowerState** to *PowerState.OffOffline* after the power condition of the device is either power off or offline. And it takes about 10 seconds to update **PowerState** to *PowerState.Online* after the power condition of the device is on and ready.

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

ServiceObjectDescription Property

Type **string**

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

This property depends on the **DeviceName** property.

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

DeviceName Property	Value
"RP-D10 Cash Drawer"	"SII RP-D10 Cash Drawer Service Object, Copyright (C) 20xx Seiko Instruments Inc."
"RP-E10 Cash Drawer"	"SII RP-E10 Cash Drawer 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 should be set to the UPOS version that the Service Object implements.

For example, when Build version is A, Revision version is B, this property is initialized "1.12.A.B" by the **Open** method.

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 ErrorEvent 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 the **Open** method.

SynchronizingObject Property

Type	System.ComponentModel.ISynchronizeInvoke
Description	<p>Stores an instance of the ISynchronizeInvoke class. The application can use this property to specify the thread to which events are notified. If SynchronizingObject is set to <i>null</i>, events will be notified to the internal thread owned by the Service Object.</p> <p>The application using a Windows form sets the <i>this</i> pointer of the Form class of the main form to SynchronizationObject, so that events are notified to the main application thread as required by the Form class.</p>

4.2.3 Specific Properties

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

CapStatus Property

Type **bool**

Description Gets the status of the cash drawer status, open or closed.
The following table shows the valid property values.

Value	Meaning
<i>true</i>	The drawer can report its status, open or closed.

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

CapStatusMultiDrawerDetect Property

Type **bool**

Description Gets the Boolean value that indicates the status of multidrawer configurations.
The following table shows the valid property values.

Value	Meaning
<i>false</i>	The statuses unique to each drawer in multidrawer configurations cannot be reported.

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

DrawerOpened Property

Type **bool**

Description Gets the Boolean value that indicates whether the drawer is open.
The following table shows the valid property values.

Value	Meaning
<i>false</i>	The cash drawer is closed.
<i>true</i>	The cash drawer is open.

This property is different depending on the [Sensor status when drawer is open] in the configuration program.

In case of selecting [Low]:

When the sensor status is "Low", this property indicates *true*.

In case of selecting [High]:

When the sensor status is "High", this property indicates *true*.

This property is initialized while the device is enabled and keeps the current state.

4.2.4 Common Methods

This section describes the details of the common methods for CashDrawer.
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. The following values may be specified:

• Values of the *level* parameter

Value	Meaning
<i>HealthCheckLevel.External</i>	Executes a complete test using the device. Attempts to open the cash drawer. When the cash drawer is in open state, "External HCheck: Successful" is returned. This method fails when another application has exclusive access to the device.
<i>HealthCheckLevel.Interactive</i>	Executes an interactive test of the device. The software displays the modal dialog box and opens the specified cash drawer.
<i>HealthCheckLevel.Internal</i>	Executes a health check without using the device physically. "Internal HCheck: Successful" is always returned.

Description Tests the status of the device.
A text description of the results of this method is stored in the **CheckHealthText** property.

Claim Method

Syntax `void Claim(int timeout);`

Parameter	Meaning
<i>timeout</i>	Specifies the maximum waiting time (in millisecond) 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 the exclusive access to the device.
Acquisition of exclusive access is not essential since the Cash Drawer device is a sharable device.
When it is successful, the **Claimed** property is set to true.
When the power is OFF or the cable is not connected, **Claim** is not available.

Close Method

Syntax	void Close();
Description	<p>Releases the device and its resources.</p> <p>If the DeviceEnabled property is <i>true</i>, the device is first disabled.</p> <p>If the Claimed property is <i>true</i>, exclusive access to the device is first released.</p> <p>Do not execute this while the event is in progress (or in the event handler).</p>

CompareFirmwareVersion Method

Syntax	CompareFirmwareResult CompareFirmwareVersion(string firmwareFileName);
Description	This method is not supported.

DirectIO Method

Syntax	DirectIOData DirectIO(int command, int data, object obj);
Description	This method is not supported.

Open Method

Syntax	void Open();
Description	<p>Opens the device.</p> <p>When the Open method is successful, the common property and other class-specific properties are initialized.</p> <p>When the Bluetooth model is used, wait at least 15 seconds after Close method before Open method.</p>

Release Method

Syntax	void Release();
Description	<p>Releases the exclusive access to the device.</p> <p>This method does not change the device enabled state.</p> <p>Do not execute this while the event is in progress (or in the event handler).</p>

ResetStatistic(string) Method

Syntax	void ResetStatistic(string statistic);
Description	This method is not supported.

ResetStatistics() Method

Syntax **void ResetStatistics();**

Description This method is not supported.

ResetStatistics(StatisticCategories) Method

Syntax **void ResetStatistics(StatisticCategories *statistics*);**

Description This method is not supported.

ResetStatistics(string[]) Method

Syntax **void ResetStatistics(string[] *statistics*);**

Description This method is not supported.

RetrieveStatistic(string) Method

Syntax **string RetrieveStatistic(string *statistic*);**

Description This method is not supported.

RetrieveStatistics() Method

Syntax **string RetrieveStatistics();**

Description This method is not supported.

RetrieveStatistics(StatisticCategories) Method

Syntax **string RetrieveStatistics(StatisticCategories *statistics*);**

Description This method is not supported.

RetrieveStatistics(string[]) Method

Syntax **string RetrieveStatistics(string[] *statistics*);**

Description This method is not supported.

UpdateFirmware Method

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

Description This method is not supported.

UpdateStatistic Method

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

Description This method is not supported.

UpdateStatistics(Statistic[]) Method

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

Description This method is not supported.

UpdateStatistics(StatisticCategories, Object) Method

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

Description This method is not supported.

4.2.5 Specific Methods

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

OpenDrawer Method

Syntax	void OpenDrawer();
Description	Opens the cash drawer. This method fails when another application has exclusive access to the device.

WaitForDrawerClose Method

Syntax	void WaitForDrawerClose(int <i>beepTimeout</i>, int <i>beepFrequency</i>, int <i>beepDuration</i>, int <i>beepDelay</i>);
Description	Waits until the cash drawer is closed. This method does not return control to the application until DrawerOpened is <i>false</i> or the printer is powered off. The alert beeper is not supported. This method fails when another application has exclusive access to the device.

4.2.6 Events

StatusUpdateEvent Event

Syntax **StatusUpdateEventHandler StatusUpdateEvent;**

Description This event is notified when the open/close state of the cash drawer is changed.
When the **CapStatus** property is *false*, the device cannot report the state change and this event is not notified.

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

Value	Meaning
<i>StatusClosed</i> (0)	The cash drawer is closed.
<i>StatusOpen</i> (1)	The cash drawer is open.
<i>StatusIdle</i> (1001)	All the asynchronous outputs finished either successfully or by being cleared. The State property is now <i>ControlState.Idle</i> . The FlagWhenIdle property must be <i>true</i> for this event to be notified. Then, the Service Object automatically resets the property to <i>false</i> before the event is notified.
<i>StatusPowerOnline</i> (2001)* ¹	The device is powered on and ready.
<i>StatusPowerOffOffline</i> (2004)* ¹	The device is powered off or offline.

*1: This is notified when **PowerNotify** = *PowerNotification.Enabled*.

When the Bluetooth model is used, it takes about 30 seconds to notify the *StatusPowerOffOffline*(2004) after the power condition of the device is either power off or offline.

And it takes about 10 seconds to notify the *StatusPowerOnline*(2001) after the power condition of the device is on and ready.

Appendix A Exceptions

A.1 PosPrinter Exception Error List

(1) Property

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

(2) Method

Method	ErrorCode	ErrorCode Extended	Meaning
BeginInsertion BeginRemoval ChangePrintSide CompareFirmwareVersion EndInsertion EndRemoval MarkFeed PrintTwoNormal UpdateFirmware UpdateStatistic(s)	Illegal	0	This method is not supported.
CheckHealth	Busy	0	Cannot perform while output is in progress or an error occurs.
	Disabled	0	Not enabled. Call this after setting the DeviceEnabled property to <i>true</i> .
	Illegal	0	Parameter has an error.
	NotClaimed	0	Exclusive access is not available. Call the Claim method to gain exclusive access.
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 millisecond) has elapsed before the device is released. Or, the POS Printer device is not available before the <i>timeout</i> (in millisecond) has elapsed.
ClearOutput	NotClaimed	0	Exclusive access is not available. Call Claim method to gain exclusive access.
ClearPrintArea	Disabled	0	Not enabled. Call this after setting the DeviceEnabled property to <i>true</i> .
	NotClaimed	0	Exclusive access is not available. Call the Claim method to gain exclusive access.
Close	Busy	0	The State property 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.

Method	ErrorCode	ErrorCode Extended	Meaning
CutPaper DirectIO 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 the DeviceEnabled property 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 has occurred.
	Extended	1002	Autocutter error has occurred.
	Extended	1005	A head temperature error occurred.
	Extended	1010	An unrecoverable error occurred.
	Failure	0	Communication with the printer failed.
	Illegal	0	Parameter has an error. Rotated printing or page printing is in progress. (CutPaper) PageModeStation is not specified. (PageModePrint)
	NoHardware	0	The printer is powered off or the cable is not connected.
Release	NotClaimed	0	Exclusive access is not available. Call the Claim method to gain exclusive access.
	Timeout	0	Data transmit timeout or data receive timeout has occurred.
ValidateData	NotClaimed	0	The device is not exclusive.
	Disabled	0	Not enabled. Call this after setting the DeviceEnabled property 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 stored when the escape sequence is not supported by the Page Mode function or rotated 90° left or right print mode.
	Illegal	0	Parameter has an error.
	NotClaimed	0	Exclusive access is not available. Call the Claim method to gain exclusive access.

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

Escape Sequence	Condition
Paper cut	One of the following statuses occurs. -Percentage '#' is not precisely supported. -It is not supported during rotated 90° right/left mode by the RotatePrint method. -It is not supported during page mode by the PageModePrint method.
Feed and Paper cut	One of the following statuses occurs. -Percentage '#' is not precisely supported. -It is not supported during rotated 90° right/left mode by the RotatePrint method. -It is not supported during page mode by the PageModePrint method.
Print bitmap	No printable bitmap exists.
Feed lines	One of the following statuses occurs. -The line number '#' is not correct. - It is not supported during rotated 90° right/left mode by RotatePrint method. - It is not supported during page mode by PageModePrint method.
Feed units	One of the following statuses occurs. - Feed unit number '#' is not precisely supported due to occurrence of rounding error of one dot depending on the setting of the MapMode property. -The feed unit number '#' is not correct. - It is not supported during rotated 90° right/left mode by RotatePrint method. - It is not supported during page mode by PageModePrint method.
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	One of the following statuses occurs. - It is not supported during rotated 90° right/left mode by RotatePrint method. - It is not supported during page mode by PageModePrint method.
Center	One of the following statuses occurs. - It is not supported during rotated 90° right/left mode by RotatePrint method. - It is not supported during page mode by PageModePrint method.
Right justify	One of the following statuses occurs. - It is not supported during rotated 90° right/left mode by RotatePrint method. - It is not supported during page mode by PageModePrint method.

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

Escape Sequence	Condition
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.

A.2 CashDrawer Exception Error List

(1) Property

ErrorCode	ErrorCode Extended	Meaning
Disabled	0	Not enabled. Call this after setting the DeviceEnabled property to <i>true</i> .
Illegal	0	This property is not supported. Parameter has an error.

(2) Method

Method	ErrorCode	ErrorCode Extended	Meaning
CompareFirmwareVersion UpdateFirmware UpdateStatistic(s) ResetStatistic(s) RetrieveStatistic(s)	Illegal	0	This method is not supported.
CheckHealth	Busy	0	Cannot perform while output is in progress or an error occurs.
	Claimed	0	Attempt was made to access a device that is exclusively accessed by another process.
	Disabled	0	Not enabled. Call this after setting the DeviceEnabled property to <i>true</i> .
	Failure	0	Could not confirm that the cash drawer is opened.
	Illegal	0	Parameter has an error.
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 millisecond) has elapsed before the device is released. Or, the POS Printer device is not available before the <i>timeout</i> (in millisecond) has elapsed.
Close	Closed	0	The device is already closed.
Open	Illegal	0	The device is already open.

Method	ErrorCode	ErrorCode Extended	Meaning
OpenDrawer	Claimed	0	Attempt was made to access a device that is exclusively accessed by another process.
	Disabled	0	Not enabled. Call this after setting the DeviceEnabled property to <i>true</i> .
	Failure	0	Communication with the printer failed.
	NoHardware	0	The printer is powered off or the cable is not connected.
	Timeout	0	Data transmit timeout or data receive timeout has occurred.
WaitForDrawerClose	Claimed	0	Attempt was made to access a device that is exclusively accessed by another process.
	Disabled	0	Not enabled. Call this after setting DeviceEnabled to <i>true</i> .
	NoHardware	0	The printer is powered off or the cable is not connected.
Release	NotClaimed	0	The device is not exclusive.

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	Number of autocutter drive times	✓
UnifiedPOSVersion	1.12	-
SlipCoverOpenCount	0	-
HoursPoweredCount	Number of hours powered on (unit: hour)	✓
FirmwareRevision	Firmware version	-
SerialNumber	Unknown	-
ReceiptLinePrintedCount	0	-
InstallationDate	Unknown	-
MechanicalRevision	1C* ¹	-
	1E* ²	
	1A* ³	
FailedPaperCutCount	0	-
StampFiredCount	0	-
FailedPrintSideChangeCount	0	-
JournalCharacterPrintedCount	0	-
SlipCharacterPrintedCount	0	-
ManufacturerName	Seiko Instruments Inc.	-

XML Definition Name	Response	Can Be Reset
PrinterFaultCount	0	-
MaximumTempReachedCount	0	-
ModelName	RP-D10 ^{*4}	-
	RP-E10 ^{*5}	
CommunicationErrorCount	0	-
JournalLinePrintedCount	0	-
SlipLineFeedCount	0	-
HomeErrorCount	0	-
FormInsertionCount	0	-
Interface	Unknown	-
DeviceCategory	POSPrinter	-
BarcodePrintedCount	0	-
NVRAMWriteCount	0	-
SlipLinePrintedCount	0	-

*1: When the device is RP-D10 USB model, Serial model or Ethernet model.

*2: When the device is RP-D10 Bluetooth model.

*3: When the device is RP-E10.

*4: When the **DeviceName** property is "RP-D10 POS Printer".

*5: When the **DeviceName** property is "RP-E10 POS Printer".

(2) StatisticCategories.Manufacturer

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