BIXOLON®

BXL SDK for iOS_UPOS Compliant API Reference Guide

Rev. 1.12

SPP-R210
SPP-R220
SPP-R310
SPP-R410/418
SPP-R200III
SRP-350plusIII/352plusIII
SRP-350III/352III
SRP-275III
SRP-F310II/F312II/F313II
SRP-380/382
SRP-330II/332II
SRP-340II/342II
SRP-S300
SRP-Q300/Q302
SRP-QE300/QE302

http://www.bixolon.com

Table of Contents

1. About This Manual

BXL SDK for iOS_UPOS Compliant complies with UnifiedPOS.

BXL SDK for iOS_UPOS Compliant provides iOS framework that allows applications' software to access BIXOLON printers.

This manual contains instructions, specifications and limitations of BXL SDK for iOS_UPOS Compliant, and it is intended for developers who make application systems using UPOS devices.

You should set the device using UPOS Setup included in BXL SDK for iOS_UPOS Compliant before using the printer.

[Reference Site]
http://monroecs.com/unifiedpos.htm: UnifiedPOS Committee
http://www.bixolon.com: SDK Update
http://developer.apple.com : Apple Developer

BIXOLON is continually improving the functions and quality of products. The specifications of the product and contents of the manual are subject to change without prior notice due to this reason.

2. Operating Environment

2-1 Operating System

This software supports the following operating systems.

- iOS 6.0 and later

2-2 Supported Devices and Interfaces

Models	Interface
SPP-R210	Bluetooth / WLAN
SPP-R220	Bluetooth / WLAN / BLE
SPP-R310	Bluetooth / WLAN
SPP-R410	Bluetooth / WLAN / BLE
SPP-R200III	Bluetooth / WLAN
SRP-350plusIII	Bluetooth / WLAN / Ethernet
SRP-352plusIII	Bluetooth / WLAN / Ethernet
SRP-350III	Ethernet
SRP-352III	Ethernet
SRP-F310II	Bluetooth / WLAN / Ethernet
SRP-F312II	Bluetooth / WLAN / Ethernet
SRP-F313II	Bluetooth / WLAN / Ethernet
SRP-275III	Ethernet
SRP-380	Bluetooth / WLAN / Ethernet
SRP-382	Bluetooth / WLAN / Ethernet
SRP-330II	Ethernet
SRP-332II	Ethernet
SRP-340II	Ethernet
SRP-342II	Ethernet
SRP-S300	Bluetooth / WLAN / Ethernet
SRP-Q300	Bluetooth / WLAN / Ethernet
SRP-Q302	Bluetooth / WLAN / Ethernet
SPP-R418	Bluetooth / WLAN / BLE
SRP-QE300	Ethernet
SRP-QE302	Ethernet

※ BLE : Bluetooth Low Energy

3. Development Environment

3-1 Setting Development Environment

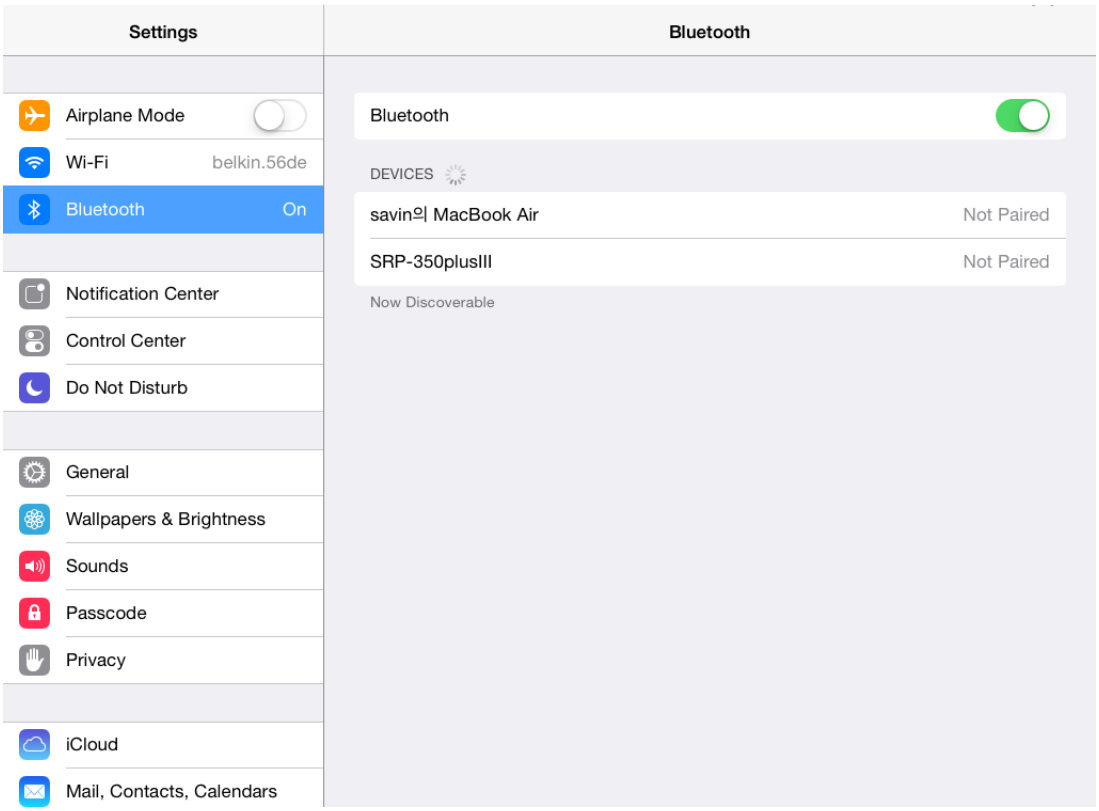
- Xcode 6.0 and later
- iOS SDK
- Reference: http://developer.apple.com/devcenter/ios/index.action

3-2 Connecting iOS Device

The following screen shot was captured from iOS.
Some details and names of specific items could be different depending on the iOS version or device.

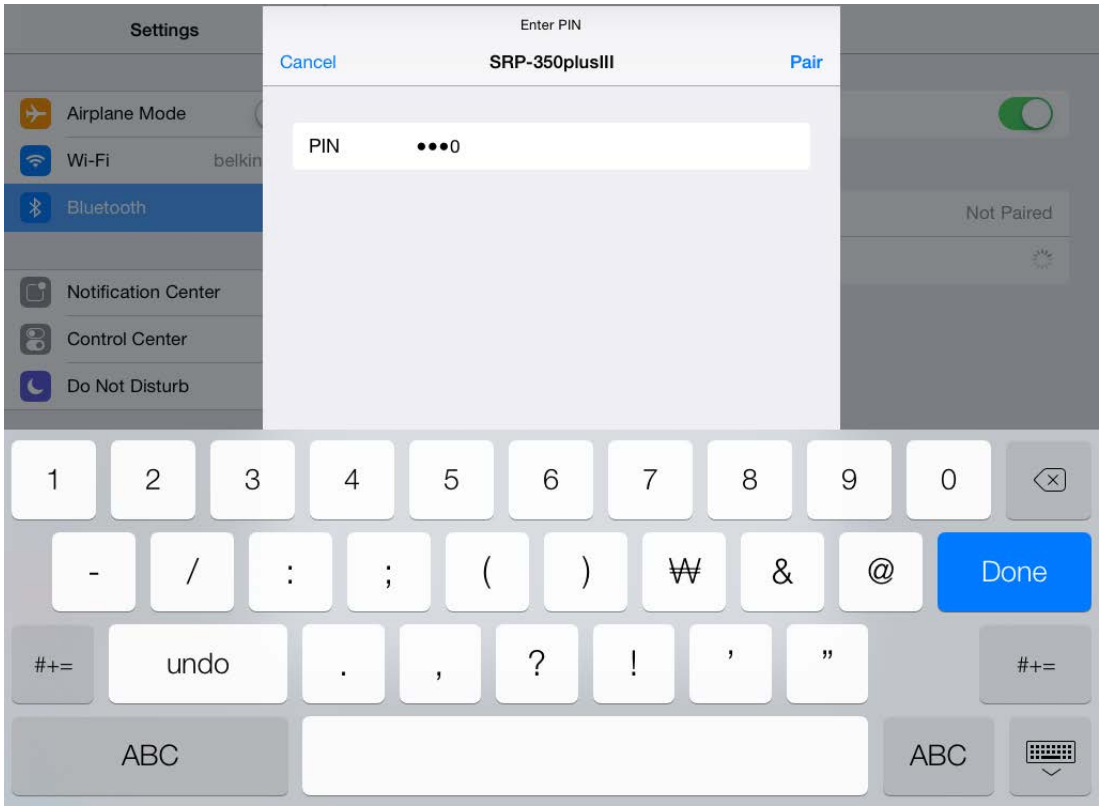
3-2-1 Bluetooth

1. Select [Settings].
2. Bluetooth and printer should be turned on.
3. Select [Bluetooth] for settings.



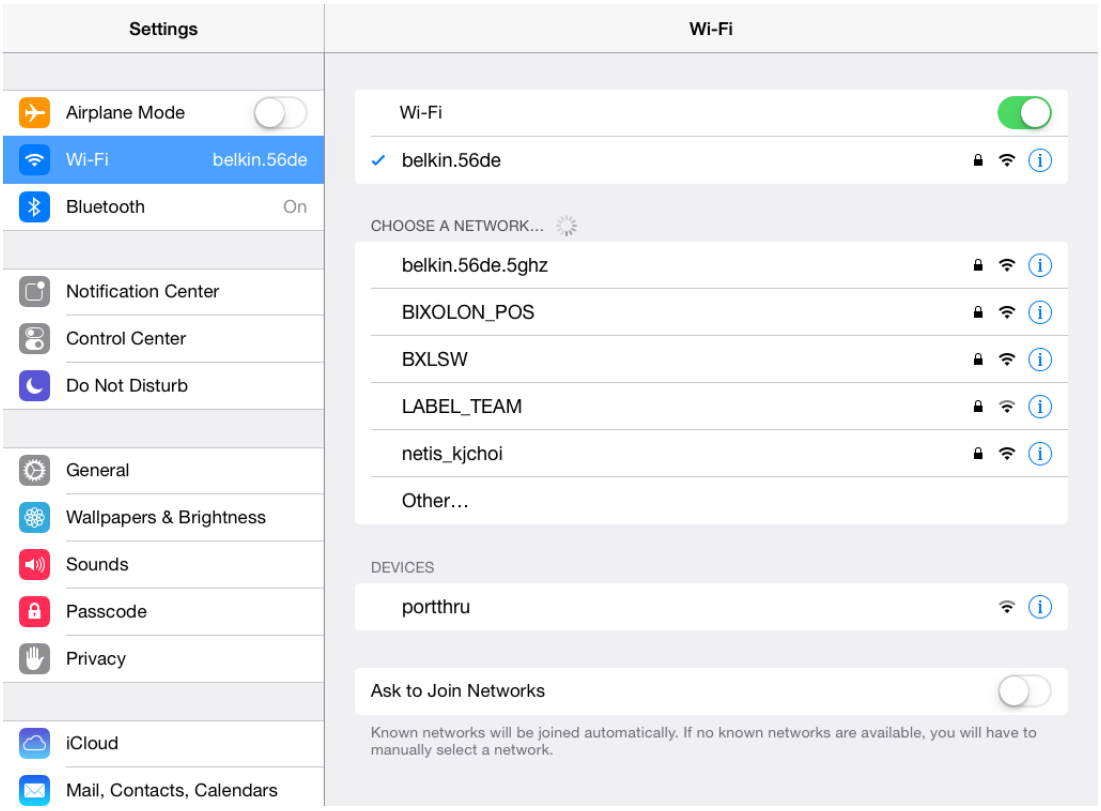
4. Select [Scan] to search the printer to connect and perform pairing.

5. Enter the PIN code. Default PIN code of BIXOLON printers is “0000”.



3-2-2 Network–Infra structure Mode

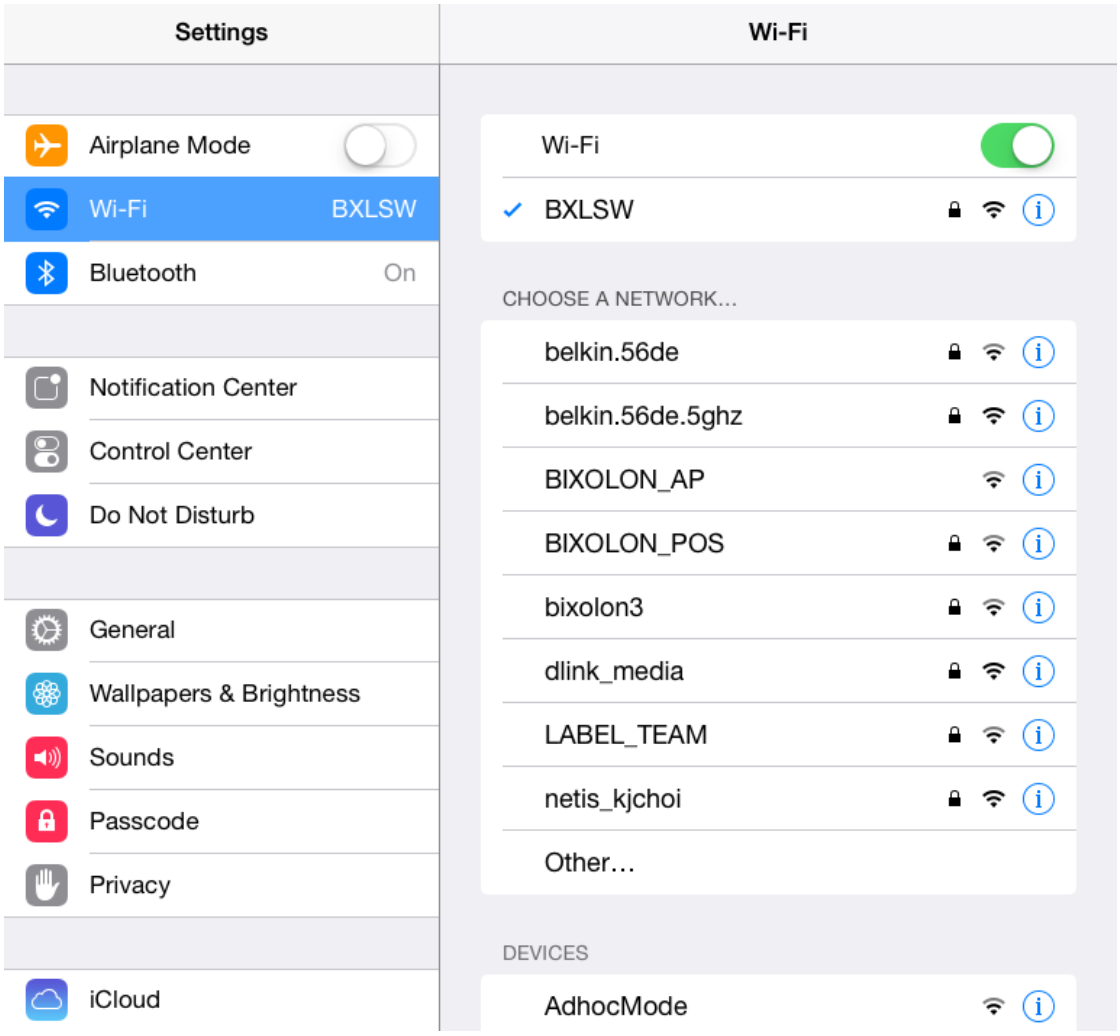
1. Select [Settings].
2. Wi-Fi should be turned on.
3. Connect the device to the same network that the printer is connected to.



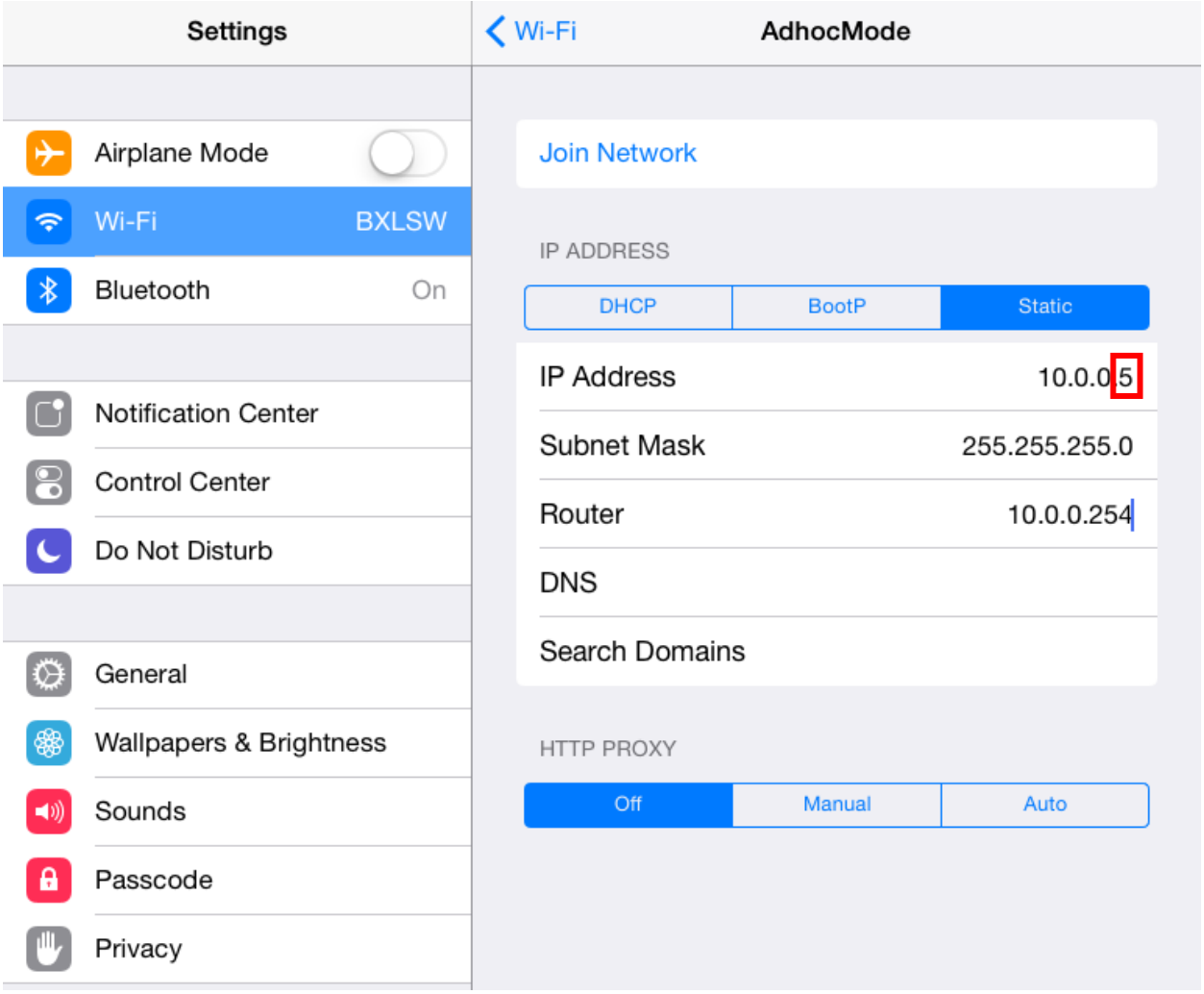
3-2-3 Network–Ad Hoc Mode

1. If the "Network Mode" of the printer is specified as "ADHOC", It is needed to configure IP Address".

2. Click the " ⓘ " button next to SSID.



3. Assign IP address which consists of four decimal numbers as below.
   ※ Modify the fourth part if the ip address of the printer is "10.0.0.1"Select [Settings].



4. Package Contents

- Manual in Korean
  Manual_BXL SDK for iOS_UPOS Compliant API Reference Guide_korean_ Rev_x_xx.pdf
- Manual in English
  Manual_BXL SDK for iOS_UPOS Compliant API Reference Guide_english_ Rev_x_xx.pdf
- Libs/lib/libBixolonUPOS.a: UPOS SDK – Library type.
- Libs/framework/frmBixolonUPOS: UPOS SDK –Framework type.
- samples/nativeSample: UPOS sample application Based on iOS
- samples/phonegapSample: Web App sampling using PhoneGap

5. Constants (Defines)

5-1 Result Code

- These constants are used for the results returned from methods after executing specific functions.

Code DEFINE	Description
UPOS_SUCCESS	Operation is successful.
UPOS_E_CLOSED	Device to access is closed.
UPOS_E_CLAIMED	Claim method should be called first.
UPOS_E_NOTCLAIMED	Device is not in Claim state.
UPOS_E_NOSERVICE	Function is not supported.
UPOS_E_DISABLED	Not enabled.
UPOS_E_ILLEGAL	Illegal access or unsupported function
UPOS_E_NOHARDWARE	Device is not connected.
UPOS_E_OFFLINE	Device is off-line.
UPOS_E_NOEXIST	Target does not exist.
UPOS_E_EXISTS	Target already exists.
UPOS_E_FAILURE	The requested operation failed.
UPOS_E_TIMEOUT	Timeout
UPOS_E_BUSY	Device is busy executing previously requested operation.
UPOS_E_EXTENDED	Device error. Refer to the ResultCode Extended section for more details.
UPOS_E_DEPRECATED	The function is currently not used.

5-2 OpenResult Code

- These constants are the results returned by the Open method.

Code DEFINE	Description
UPOS_SUCCESS	Open operation is successful.
UPOS_OR_ALREADYOPEN	Device is already open.
UPOS_OR_REGBADNAME	The specified device name cannot be found in the stored device list.
UPOS_OR_FAILEDOPEN	The execution of the Open method failed, but specific reason is unknown.

5-3 State Code

- These are the constants that are used for the property of State.

Code DEFINE	Description
UPOS_S_CLOSED	Device is closed.
UPOS_S_IDLE	Device is in standby state without error.
UPOS_S_BUSY	Device is currently busy executing another method.
UPOS_S_ERROR	There is an error.

5-4 Transaction Print

- These are the constants for setting the Transaction mode.

Code DEFINE	Description
PTR_TR_TRANSACTION	Initialize the buffer to Empty state and start the Transaction mode.
PTR_TR_NORMAL	Terminate the Transaction mode and print the data stored in the buffer.

5-5 Alignment

- These are the constants required for specifying alignment.
  (For Barcodes)

Code DEFINE	Description
PTR_BC_LEFT	Align to left
PTR_BC_CENTER	Align to center
PTR_BC_RIGHT	Align to right

(For Images)

Code DEFINE	Description
PTR_BM_LEFT	Align to left
PTR_BM_CENTER	Align to center
PTR_BM_RIGHT	Align to right

5-6 Barcode Type

- These are the constants required for specifying barcode type when printing a barcode

Code DEFINE	Description
PTR_BCS_UPCA	UPCA
PTR_BCS_UPCE	UPCE
PTR_BCS_JAN8	JAN8
PTR_BCS_EAN8	EAN8
PTR_BCS_JAN13	JAN13
PTR_BCS_EAN13	EAN13
PTR_BCS_TF	Standard(ordiscrete) 2 of 5
PTR_BCS_ITF	Interleaved 2 of 5
PTR_BCS_Codabar	Codabar
PTR_BCS_Code39	Code39
PTR_BCS_Code93	Code93
PTR_BCS_Code128	Code 128 ※ Special Character of Code 128 Special Characters: Ascii Representation Code A: {A Code B: {B Code C: {C
PTR_BCS_UPCA_S	UPC-A with supplemental barcode
PTR_BCS_UPCE_S	UPC-E with supplemental barcode
PTR_BCS_UPCD1	UPC-D1
PTR_BCS_UPCD2	UPC-D2
PTR_BCS_UPCD3	UPC-D3
PTR_BCS_UPCD4	UPC-D4
PTR_BCS_UPCD5	UPC-D5
PTR_BCS_EAN8_S	EAN8 with supplemental barcode
PTR_BCS_EAN13_S	EAN13 with supplemental barcode
PTR_BCS_EAN128	EAN128
PTR_BCS_OCRA	OCR “A”
PTR_BCS_OCRB	OCR “B”
PTR_BCS_Code128_Parsed	Code 128 with parsing
PTR_BCS_GS1DATABAR	GS1 DataBar Omnidirectional
PTR_BCS_GS1DATABAR_E	GS1 DataBar Stacked Omnidirectional
PTR_BCS_GS1DATABAR_S	GS1 DataBar Expanded
PTR_BCS_GS1DATABAR_E_S	GS1 DataBar Expanded Stacked
PTR_BCS_PDF417	PDF 417
PTR_BCS_MAXICODE	MAXI Code
PTR_BCS_DATAMATRIX	Data Matrix
PTR_BCS_QRCODE	QR Code
PTR_BCS_UQRCODE	Micro QR Code
PTR_BCS_AZTEC	Aztec
PTR_BCS_UPDF417	Micro PDF 417

5-7 Barcode Text Position

- These are the constants for specifying the text printing option and printing position when the specific barcode supports text printing.

Code DEFINE	Description
PTR_BC_TEXT_NONE	Do not print text. Print barcode only.
PTR_BC_TEXT_ABOVE	Print text above the barcode.
PTR_BC_TEXT_BELOW	Print text below the barcode.

5-8 StatusUpdateEvent

- These are the constants for specifying parameters in the event of StatusUpdateEvent Delegate.

Code DEFINE	Description
PTR_SUE_IDLE	Printer is in the idle state.
UPOS_SUE_POWER_ONLINE	Printer is in the online state.
UPOS_SUE_POWER_OFF	Printer is in the offline state. All of them have the same effects in this SDK unless there is a separate comment.
UPOS_SUE_POWER_OFFLINE	
UPOS_SUE_POWER_OFF_OFFLINE	
PTR_SUE_COVER_OPEN	Printer cover is open.
PTR_SUE_COVER_OK	Printer cover is closed.
PTR_SUE_REC_EMPTY	Printer paper is empty.
PTR_SUE_REC_NEAREMPTY	Printer is almost out of paper.
PTR_SUE_REC_PAPEROK	There is sufficient printing paper.
PTR_SUE_REC_BATTERY_NORMAL	The Printer Battery is enough.
PTR_SUE_REC_BATTERY_LOW	The Printer Battery is low.

6. Summary of Classes

6-1 Classes supported by this SDK

Class Type	Description	Related Classes
(1) Single Device	Information about each device is contained.	- Device Class Reference (Common Device) - [Printer] Printer Class Reference - [CashDrawer] Cash Drawer Class Reference - [MSR] MSR Class Reference
(2) Device List Manager	Multiple “(1) Single Device” can be stored as an array.	- [Common] Device Controller Class Reference - [Printer] Printer List Class Reference - [CashDrawer] Cash Drawer List Class Reference - [MSR] MSR List Class Reference (Stored MSR List)
(3) Device Controller	One of the devices stored in iDevice as “(2) Device List Manager” can be controlled. Example: Text printing, image/barcode printing in case of printer	- Device Controller Class Reference (Common Controller) - [Printer] Printer Controller Class Reference - [CashDrawer] Cash Drawer Controller Class Reference - [MSR] MSR Controller Class Reference
(4) Delegate	Events can be delegated from each device to the application level, and this class should be used in order to use this function.	- [Common] Delegater Class Reference

6-2 Support Table

This table contains the list of methods supported by each device controller.

6-2-1 Printer Method

Method	Value	Pre-task
open	O	-
claim	O	open
releaseDevice	O	open – claim
close	O	open
beginInsertion	X	open – claim – deviceEnabled
beginRemoval	X	open – claim – deviceEnabled
changePrintSide	X	open – claim – deviceEnabled
clearPrintArea	O	open – claim – deviceEnabled
cutPaper	O	open – claim – deviceEnabled
drawRuledLine	X	open – claim – deviceEnabled
endInsertion	X	open – claim – deviceEnabled
endRemoval	X	open – claim – deviceEnabled
markFeed	X	open – claim – deviceEnabled
pageModePrint	O	open – claim – deviceEnabled
printBarCode	O	open – claim – deviceEnabled
printBitmap	O	open – claim – deviceEnabled
printImmediate	O	open – claim – deviceEnabled
printMemoryBitmap	X	open – claim – deviceEnabled
printNormal	O	open – claim – deviceEnabled
printTwoNormal	X	open – claim – deviceEnabled
rotatePrint	X	open – claim – deviceEnabled
setBitmap	X	open – claim – deviceEnabled
setLogo	X	open – claim – deviceEnabled
transactionPrint	O	open – claim – deviceEnabled
validateData	X	open – claim – deviceEnabled
displayString	O	open – claim – deviceEnabled
displayStringAtLine	O	open – claim – deviceEnabled
clearScreen	O	open – claim – deviceEnabled
setDisplayCharacterset	O	open – claim – deviceEnabled
setInternationalCharacterset	O	open – claim – deviceEnabled
storeImage	O	open – claim – deviceEnabled
storeImageFile	O	open – claim – deviceEnabled
displayImage	O	open – claim – deviceEnabled
clearImage	O	open – claim – deviceEnabled

O: Supported X: Not supported

6-2-2 CashDrawer Method

Method	Value	Pre-task
open	O	-
claim	O	open
releaseDevice	O	open – claim
close	O	open
openDrawer	O	open – claim – deviceEnabled

O: Supported X: Not supported

6-2-3 MSR Method

Method	Value	Pre-task
open	O	-
claim	O	open
releaseDevice	O	open – claim
close	O	open

O: Supported X: Not supported

6-2-4 SCR Method

Method	Value	선행작업
open	O	-
claim	O	Open
releaseDevice	O	open – claim
close	O	open
beginInsertion	O	open – claim – deviceEnabled
beginRemoval	O	open – claim – deviceEnabled
endInsertion	O	open – claim – deviceEnabled
endRemoval	O	open – claim – deviceEnabled

O: Supported X: Not supported

7. [Common] Device Class Reference

Inherits from	NSObject
Framework	libBixolonUPOS.a
[Availability]	iOS 7.0 and later
Class name	UPOSDevice
Declared	UPOSDevices.h

7-1 Overview

UPOSDevice Class is an object that contains the common information about all target control devices controlled by the controller of each device.

7-2 Available Properties

7-2-1 modelName

This property defines the model name of each device.

[Declare]

```
@property (retain)    NSString          *modelName;
```

[Availability]
SDK 1.0.0 and later

7-2-2 ldn (Logical Device Name)

This property defines the logical name of each device.
It can be used by the Open method.

[Declare]

```
@property (retain)    NSString          *ldn;
```

[Availability]
SDK 1.0.0 and later

7-2-3 InterfaceType

This property defines the method of connecting each device.

[Declare]

```
@property (retain)    NSString          *interfaceType;
```

[Availability]
SDK 1.0.0 and later

7-2-4 address

This is the property and the address value of each device.
The value will be the IP address instead of the MAC address if the Interface Type is _CONNECT_BLUETOOTH.

[Declare]

```
@property (retain)   NSString          *interfaceType;
```

[Availability]
SDK 1.0.0 and later

7-2-5 port

This property is used for the Network Port that is used for socket (Wi-Fi / Ethernet) Communication.This property may be ignored if the Interface Type is _CONNECT_BLUETOOTH.

[Declare]

```
@property (retain)   NSString          *port;
```

[Availability]
SDK 1.0.0 and later

7-3 Available Method

- Not applicable

8. [Common] Device List Class Reference

Inherits from	NSObject
Framework	libBixolonUPOS.a
[Availability]	iOS 8.0 and later
Class name	UPOSDevices
Declared	UPOSDevices.h

8-1 Overview

UPOSDevices Class is an object that contains the device list in the iDevice.

8-2 Available Properties

- Not applicable

8-3 Available Method

8-3-1 getList

This method reads the stored device list.

[See Also]

- Device Class Reference (Common Device)
- Device Controller Class Reference (Common Controller)
- [Printer] Printer Controller Class Reference
- [CashDrawer] Cash Drawer Controller Class Reference
- [MSR] MSR Controller Class Reference
- [Printer] Printer Class Reference
- [CashDrawer] Cash Drawer Class Reference
- [MSR] MSR Class Reference
- [Printer] Printer List Class Reference
- [CashDrawer] Cash Drawer List Class Reference
- [MSR] MSR List Class Reference (Stored MSR List)

[Declare]

```
-(NSMutableArray*)getList;
```

[Parameters]

None

[Return Value]

- NSMutableArray*:
  The list of devices is stored as an array.
  Each object is stored as a Device Class.

[Example]

Calling the methods of this class directly may not work correctly. In this case, use the methods defined for each device.

[Availability]

SDK 1.0.0 and later

8-3-2 getDeviceIdentity

This method obtains the Identity string of the current device.

[Declare]

```
-(NSString*)getDeviceIdentity;
```

[Parameters]
None

[Return Value]
- NSString*:
  The identity of the current device is returned as string type.

[Example]
Calling the methods of this class directly may not work correctly. In this case, use the methods defined for each device.

[Availability]
SDK 1.0.0 and later

8-3-3 save

This method saves the current list of devices.

[Declare]

```
-(BOOL) save;
```

[Parameters]
None

[Return Value]
- BOOL:
  YES if operation is successful.

[Example]
Calling the methods of this class directly may not work correctly.
In this case, use the methods defined for each device.

[Availability]
SDK 1.0.0 and later

8-3-4 addDevice

This method adds a device to the current device list.
The device's addition may not be reflected to the device list when the list is refreshed if the data is not saved with the save method after using this method.

[See Also]

- Refer to Device List Class Reference (stored device list) :: save

[Declare]

```
-(BOOL) addDevice:(UPOSDevice*)device;
```

[Parameters]

- (UPOSDevice*) device
  Object that contains the information about the device to add

[Return Value]

- BOOL:
  YES if the operation is successful.

[Example]

Calling the methods of this class directly may not work correctly.
In this case, use the methods defined for each device.

[Availability]

SDK 1.0.0 and later

8-3-5 removeDevice

This method deletes a device from the device list.
The deleted device may not be reflected to the device list when the list is refreshed if the data is not saved with save method after using this method.

[See Also]
- Refer to Device List Class Reference (stored device list) :: save

[Declare]

```
-(BOOL) removeDevice:(UPOSDevice*)device;
```

[Parameters]
- (UPOSDevice*) device
  Object that contains the information about the device to add

[Return Value]
- BOOL:
  YES if the operation is successful.

[Example]
Calling the methods of this class directly may not work correctly.
In this case, use the methods defined for each device.

[Availability]
SDK 1.0.0 and later

9. [Common] Device Controller Class Reference

Inherits from	NSObject
Framework	libBixolonUPOS.a
[Availability]	iOS 7.0 and later
Class name	UPOSDeviceController
Declared	UPOSDeviceController.h

9-1 Overview

UPOSDeviceController Class is the main object to control the common functions of the devices supported by this SDK.

9-2 Available Properties

9-2-1 CheckHealthText

This is the content that is printed when using the checkHealth function of each device.
This value may not have any meaning if the device is not an output device.

[Declare]

```
@property (strong, readonly)        NSString*     CheckHealthText;
```

[Availability]
SDK 1.0.0 and later

9-2-2 Claimed

This indicates the state of Claim.
If this property is YES, it means that the Claim: API of each device is called and executed successfully. This property is initialized when the Claim method is called.

[See Also]
- Claim

[Declare]

```
@property (readonly)                BOOL          Claimed;
```

[Availability]
SDK 1.0.0 and later

9-2-3 DeviceEnabled

This property is an option to use the device.
This function may not be available if the property of DeviceEnabled is NO, even when the state if the property of Claimed is YES.

[See Also]
- Claimed

[Declare]

```
@property                    BOOL        DeviceEnabled;
```

[Availability]
SDK 1.0.0 and later

9-2-4 OpenResult

The execution result of the Open method is stored in this property.
This property is initialized when the Open method is called.

[See Also]
- OpenResult Code
- open

[Declare]

```
@property (readonly)         NSInteger   OpenResult;
```

[Availability]
SDK 1.0.0 and later

9-2-5 ResultCode

The result of the last called method is stored in this property.

[See Also]
- Constants_(Defines) :: Result Code

[Declare]

```
@property (readonly)         NSInteger   ResultCode;
```

[Availability]
SDK 1.0.0 and later

9-2-6 ResultCodeExtended

This property contains the detailed error information when a device error occurs.
It is initialized when UPOS_E_EXTENDED is stored as a result of the ResultCode.

[See Also]
- Constants (Defines) :: Result Code
- [Common] Device Controller Class Reference :: Available Properties :: ResultCode

[Declare]

```
@property (readonly)        NSInteger    ResultCode;
```

[Availability]
SDK 1.0.0 and later

9-2-7 OutputID

OutputComplete: Delegate will be called when the printing operation is executed in the Async Mode. In this case, this property will be incremented by one.

[Declare]

```
@property (readonly)        NSInteger    OutputID;
```

[Availability]
SDK 1.0.0 and later

9-2-8 State

State value of the class is stored in this property.

[See Also]
- Constants (Defines) :: State Code

[Declare]

```
@property (readonly)        NSInteger    State;
```

[Availability]
SDK 1.0.0 and later

9-3 Available Method

9-3-1 open

This method initiates the use of the printer class, and it includes the initialization process such as memory allocation. This method should be called first before calling the claim and other subsequent methods.

[See Also]
- Constants (Defines) :: OpenResult Code
- OpenResult
- ldn (Logical Device Name)

[Declare]

```
-(NSInteger) open : (NSString*)logicalDeviceName;
```

[Parameters]
(NSString*) logicalDeviceName
- Model name of the device to open or stored device name.
  Refer to ldn (Logical Device Name)

[Return Value]
NSInteger
- UPOS_SUCCESS when successful.

[Example]
Calling the methods of this class directly may not work correctly.
In this case, use the methods defined for each device.

[Availability]
SDK 1.0.0 and later

9-3-2 claim

This method tries to open the port specified in the device information, and it includes some initialization processes such as memory allocation and initialization.

This method should be called before using the device.

[See Also]
- Constants (Defines) :: Result Code
- Claimed
- open

[Declare]

```
-(NSInteger) claim : (NSInteger)timeout;
```

[Parameters]
(NSInteger) timeout
- This system tries to open the port for the duration specified in this parameter.

[Return Value]
NSInteger
- UPOS_SUCCESS if the operation is successful.

[Example]
Calling the methods of this class directly may not work correctly.
In this case, use the methods defined for each device.

[Availability]
SDK 1.0.0 and later

9-3-3 releaseDevice

This method terminates the use of the port of the claimed device and releases the physical resources. Some of the memory resources may also be released as a result.

[See Also]
- claim
- Claimed

[Declare]

```
-(NSInteger) open : (NSString*)logicalDeviceName;
```

[Parameters]
(NSString*) logicalDeviceName
- Model name of the device to open or stored device name
  Refer to ldn (Logical Device Name)

[Return Value]
NSInteger
- UPOS_SUCCESS if the operation is successful.

[Example]
Calling the methods of this class directly may not work correctly.
In this case, use the methods defined for each device.

[Availability]
SDK 1.0.0 and later

9-3-4 close

This method terminates the use of the open device.
Some of the memory resources may also be released as a result.

[See Also]
- Constants (Defines) :: Result Code
- open

[Declare]

```
-(NSInteger) claim : (NSInteger)timeout;
```

[Parameters]
(NSInteger) timeout
- The system will try to execute the corresponding operation for the duration specified by this parameter.

[Return Value]
NSInteger
- UPOS_SUCCESS if the operation is successful.

[Example]
Calling the methods of this class directly may not work correctly.
In this case, use the methods defined for each device.

[Availability]
SDK 1.0.0 and later

9-3-5 checkHealth

This method checks whether the device is operating correctly.
This method only works after Open / Claim / DeviceEnabled are executed correctly.

[See Also]
- Constants (Defines) :: Result Code
- open
- claim
- DeviceEnabled

[Declare]

```
-(NSInteger) checkHealth : (NSInteger) level;
```

[Parameters]
(NSInteger) level
- Fixed Value: UPOS_CH_INTERNAL

[Return Value]
NSInteger
- UPOS_SUCCESS if the operation is successful.

[Example]
Calling the methods of this class directly may not work correctly.
In this case, use the methods defined for each device.

[Availability]
SDK 1.0.0 and later

10. [Printer] Printer Class Reference

Inherits from	NSObject
Framework	libBixolonUPOS.a
[Availability]	iOS 7.0 and later
Class name	UPOSPrinter
Declared	UPOSDevices.h

10-1 Overview

The UPOSPrinter Class is the object that contains the information of the target device controlled by the UPOSPrinterController.

[See Also]

- [Common] Device Class Reference

10-2 Available Properties

- Refer to [Common] Device Class Reference :: Available Properties

10-3 Available Method

- Refer to [Common] Device Class Reference :: Available Method

11. [Printer] Printer List Class Reference

Inherits from	UPOSDevices
Framework	libBixolonUPOS.a
[Availability]	iOS 7.0 and later
Class name	UPOSPrinters
Declared	UPOSDevices.h

11-1 Overview

The UPOSPrinters Class is the object that contains the printer list stored in iDevice.
It is a set of UPOSPrinter objects.

[See Also]

- Printer Controller Class Reference

11-2 Available Properties

- Not applicable

11-3 Available Method

11-3-1 getList

This method reads the stored device list.

[See Also]
- Device Class Reference (Common Device)
- Device Controller Class Reference (Common Controller)
- Printer Controller Class Reference

[Declare]
- Refer to Device List Class Reference (stored device list) :: getList

[Parameters]
None

[Return Value]
- NSMutableArray*:
  The list of devices is stored as an array.
  Each object is stored as a Device Class.

[Example]

```
UPOSPrinterController* _uposPrinterController = [ UPOSPrinterController new];
UPOSPrinters* printerList    = [_uposPrinterController    getRegisteredDevice];

NSMutableArray* printerArray = [printerList getList];

if(printerArray == nil)
{
        NSLog(@"Failed to read the stored printer list.");
}
else
{
        NSLog(@"Stored printer list is read successfully.");
}
```

[Availability]
SDK 1.0.0 and later

11-3-2 getDeviceIdentity

This method reads the Idenitity string of the current device.

[Declare]

- Refer to Device List Class Reference (stored device list) :: getDeviceIdentity

[Parameters]

None

[Return Value]

- NSString*:
  The Identity of the current device is returned as a string.

[Example]

```
UPOSPrinterController* _uposPrinterController = [ UPOSPrinterController new];
UPOSPrinters* printerList    = [_uposPrinterController    getRegisteredDevice];

NSString* strIdentity = [printerList getDeviceIdentity];

if(printerArray == nil)
{
        NSLog(@"Failed to read the Identity value.");
}
else
{
        NSLog(@" Identity of UPOSPrinterController :%@.", strIdentity);
}
```

[Availability]

SDK 1.0.0 and later

11-3-3 save

This method saves the current list of devices.

[See Also]
- Printer List Class Reference (stored printer list) :: removeDevice
- Printer List Class Reference (stored printer list) :: addDevice

[Declare]
- Refer to Device List Class Reference (stored device list) :: save

[Parameters]
None

[Return Value]
- BOOL:
  YES if the operation is successful.

[Example]

```
UPOSPrinterController* _uposPrinterController = [ UPOSPrinterController new];
UPOSPrinters* printerList    = [_uposPrinterController    getRegisteredDevice];

UPOSPrinter* newDevice = [UPOSPrinter new];
newDevice.modelName    = @"SRP-350plusIII";
newDevice.ldn                = @"type1";
newDevice.interfaceType = @"4"; // Bluetooth
newDevice.address          = @"74:F0:E1:XX:XX:XX";
newDevice.port               = @"";
[printerList addDevice:newDevice];    // Add device
[newDevice release];

if([printerList save])
{
        NSLog(@"This list is saved successfully.");
}
else
{
        NSLog(@"Failed to save the list.");
}
```

[Availability]
SDK 1.0.0 and later

11-3-4 addDevice

This method adds a device to the current device list.
The device's addition may not be reflected to the list when the list is refreshed if the data is not saved by using the save method after using this method.

[See Also]
- Printer List Class Reference (stored printer list) :: save
- Printer List Class Reference (stored printer list) :: removeDevice

[Declare]
- Refer to Device List Class Reference (stored device list) :: addDevice

[Parameters]
- (UPOSDevice*) device
  This is the object that contains the information about the device to add.

[Return Value]
- BOOL:
  YES if the operation is successful.

[Example]

```
UPOSPrinterController* _uposPrinterController = [ UPOSPrinterController new];
UPOSPrinters* printerList    = [_uposPrinterController    getRegisteredDevice];

UPOSPrinter* newDevice = [UPOSPrinter new];
newDevice.modelName          = @"SRP-350plusIII";
newDevice.ldn                = @"type1";
newDevice.interfaceType      = @"4"; // Bluetooth
newDevice.address            = @"74:F0:E1:XX:XX:XX";
newDevice.port               = @"";
if([printerList addDevice:newDevice])     // Add device
{
        NSLog(@"Device is added");
}
else
{
        NSLog(@"Failed to add device.");
}

[newDevice release];
[printerList save];
```

[Availability]
SDK 1.0.0 and later

11-3-5 removeDevice

This method deletes a device from the device list.
The device's deletion may not be reflected to the device list when the list is refreshed if the data is not saved by using the save method after using this method.

[See Also]
- Printer List Class Reference (stored printer list) :: save
- Printer List Class Reference (stored printer list) :: addDevice

[Declare]
- Refer to Device List Class Reference (stored device list) :: removeDevice

[Parameters]
- (UPOSDevice*) device
  This is the object that contains the information about the device to add.

[Return Value]
- BOOL:
  YES if the operation is successful.

[Example]

```
UPOSPrinterController* _uposPrinterController = [ UPOSPrinterController new];
UPOSPrinters* printerList    = [_uposPrinterController    getRegisteredDevice];

UPOSPrinter* willRemoveDevice = [[printerList getList] objectAtIndex:0];
if([printerList removeDevice:willRemoveDevice])     // Remove device
{
        NSLog(@"0th device is removed from the list.");
}
else
{
        NSLog(@"Failed to remove 0th device from the list.");
}

[newDevice release];
[printerList save];
```

[Availability]
SDK 1.0.0 and later

12. [Printer] Printer Controller Class Reference

Inherits from	UPOSDeviceController
Framework	libBixolonUPOS.a
[Availability]	iOS 7.0 and later
Class name	UPOSPrinterController
Declared	UPOSPrinterController.h

12-1 Overview

UPOSPrinterController Class is the main object for printer control.

12-2 Properties

Properties are determined based on the following criteria of printer types.
(Some properties like paper setting may vary, depending on the printer model.)

12-2-1 Capability Properties

Capability Name	1 Color Thermal
CapCompareFirmwareVersion	FALSE
CapPowerReporting	TRUE
CapStatisticsReporting	FALSE
CapUpdateFirmware	FALSE
CapUpdateStatistics	FALSE
CapTransaction	TRUE
CapCoverSensor	TRUE
CapConcurrentRecSlp	FALSE
CapConcurrentJrnSlp	FALSE
CapConcurrentJrnRec	FALSE
CapCharacterSet	TRUE
CapRecUnderline	TRUE
CapRecPageMode	FALSE
CapCuncurrentPageMode	FALSE
CapRecStamp	FALSE
CapRecRotate180	FALSE
CapRecRight90	FALSE
CapRecPapercut	TRUE
CapRecNearEndSensor	TRUE
CapRecMarkFeed	FALSE
CapRecLeft90	FALSE
CapRecItalic	FALSE
CapRecEmptySensor	TRUE
CapRecDwideDhigh	TRUE
CapRecDwide	TRUE
CapRecDhigh	TRUE
CapRecColor	FALSE
CapRecCartridgeSensor	FALSE
CapRecBold	TRUE
CapRecBitmap	TRUE
CapRecBarCode	TRUE
CapRec2Color	FALSE
CapRecPresent	TRUE

* Some of the Capability properties may vary, depending on the printer model.

12-2-2 Default values and range of properties

- List Properties

Model	RecLineCharsList	RecBarCode RotationList	FontTypefaceList	RecBitmapList
SPP-R210	“32,42”	“0”	“”	“0”
SPP-R220	“32,42”	“0”	“”	“0”
SPP-R310	“48,64”	“0”	“”	“0”
SPP-R410	“69,92”	“0”	“”	“0”
SPP-R200III	“32,42”	“0”	“”	“0”
SRP-350plusIII	“42,56”	“0”	“”	“0”
SRP-352plusIII	“48,64”	“0”	“”	“0”
SRP-350III	“42,56”	“0”	“”	“0”
SRP-352III	“48,64”	“0”	“”	“0”
SRP-F310II	“42,56”	“0”	“”	“0”
SRP-F312II	“48,64”	“0”	“”	“0”
SRP-F313II	“50,67”	“0”	“”	“0”
SRP-275III	“33,44”	“0”	“”	“0”
SRP-380	“42,56”	“0”	“”	“0”
SRP-382	“48,64”	“0”	“”	“0”
SRP-330II	“42,56”	“0”	“”	“0”
SRP-332II	“48,64”	“0”	“”	“0”
SRP-340II	“42,56”	“0”	“”	“0”
SRP-342II	“48,64”	“0”	“”	“0”
SRP-S300	“48,64”	“0”	“”	“0”
SRP-Q300	“42,56”	“0”	“”	“0”
SRP-Q302	“48,64”	“0”	“”	“0”
SPP-R418	“69,92”	“0”	“”	“0”
SRP-QE300	“42,56”	“0”	“”	“0”
SRP-QE302	“48,64”	“0”	“”	“0”

- CharacterSetList Properties

Model	Value
SPP-R210 / SPP-R220 / SPP-R310 / SPP-R410 / SPP-R418 / SPP-R200III / SRP-350plusIII / SRP-352plusIII / SRP-350III / SRP-352III / SRP-275III / SRP-F310II / SRP-F312II / SRP-F313II / SRP-380 / SRP-382 / SRP-330II / SRP-332II / SRP-340II / SRP-342II / SRP-S300 / SRP-Q300 / SRP-Q302 / SRP-QE300 / SRP-QE302	"437,737,775,850,852,855,857,858,860,862,863, 864,865,866,928,Farsi,Thai12,Thai14,Thai16, Thai18,Thai421250,1251,1252,1253,1254,1255, 1256,1257,1258"

- Paper Width and Height Properties

Model	RecLineHeight	RecLineWidth	RecLinePaperCut
SPP-R210	24	384	-
SPP-R220	24	384	-
SPP-R310	24	576	-
SPP-R410	24	832	-
SPP-R200III	24	384	-
SRP-350plusIII	24	512	5
SRP-352plusIII	24	576	5
SRP-350III	24	512	5
SRP-352III	24	576	5
SRP-F310II	24	512	5
SRP-F312II	24	576	5
SRP-F313II	24	604	5
SRP-275III	24	400	5
SRP-380	24	512	5
SRP-382	24	576	5
SRP-330II	24	512	5
SRP-332II	24	576	5
SRP-340II	24	512	5
SRP-342II	24	576	5
SRP-S300	24	576	5
SRP-Q300	24	512	5
SRP-Q302	24	576	5
SPP-R418	24	832	-
SRP-QE300	24	512	5
SRP-QE302	24	576	5

- RecLineSpacing Properties

Property	Range		
	Default Value	Minimum Value	Maximum Value
RecLineSpacing	16	127	0

12-2-3 RecLineChars

This property specifies the number of characters that can be displayed per line.
One of the numbers shown in RecLineCharsList can be selected.

[Declare]

```
@property                    NSInteger        RecLineChars;
```

[Availability]
SDK 1.0.0 and later

12-2-4 RecLineCharsList

This property specifies the number of characters that can be displayed per line.
The number of characters per line depends on the width of the font.

[Declare]

```
@property (strong, readonly)        NSString*        RecLineCharsList;
```

[Availability]
SDK 1.0.0 and later

12-2-5 RecLineSpacing

This is the space between lines.

[Declare]

```
@property                           NSInteger        RecLineSpacing;
```

[Availability]
SDK 1.0.0 and later

12-2-6 RecLineWidth

This is the default width of paper that is supported by the printer.

[Declare]

```
@property (readonly)                NSInteger        RecLineWidth;
```

[Availability]
SDK 1.0.0 and later

12-2-7 RecEmpty

This indicates whether there is any paper in the printer.
YES means that the printer is out of paper.

[Declare]

```
@property (readonly)                BOOL             RecEmpty;
```

[Availability]
SDK 1.0.0 and later

12-2-8 RecNearEnd

This indicates that there is paper in the printer, but more needs to be added soon.
YES means that the printer is nearly out of paper.

[Declare]

```
@property (readonly)        BOOL        RecNearEnd;
```

[Availability]
SDK 1.0.0 and later

12-2-9 AsyncMode

Methods related to printing operate in asynchronous mode if it is set to YES.
If it is set to NO, then printing related methods operate in synchronous mode.

Initial setting is NO, and it is initialized whenever the printer is open.
In case of asynchronous mode, the OutputCompleteEvent can be used to check whether printing operation is completed.

[See Also]
- outputCompleteEvent

[Declare]

```
@property (nonatomic)        BOOL        AsyncMode;
```

[Availability]
SDK 1.0.0 and later

12-2-10 CharacterSet

This is the character set to be used for printing.

[Declare]

```
@property (nonatomic)        NSInteger        CharacterSet;
```

[Availability]
SDK 1.0.0 and later

12-2-11 CharacterSetList

This is the list of character sets supported by the printer.

[Declare]

```
@property (strong, readonly)        NSString*        CharacterSetList;
```

[Availability]
SDK 1.0.0 and later

12-2-12 CoverOpen

This indicates the printer cover status.
YES means that the cover is open.

[Declare]

```
@property                    BOOL            CoverOpen;
```

[Availability]
SDK 1.0.0 and later

12-2-13 ErrorLevel

This indicates the error status.

[Declare]

```
@property (readonly)         NSInteger       ErrorLevel;
```

[Availability]
SDK 1.0.0 and later

12-2-14 ErrorString

Error status can be checked through this string.

[Declare]

```
@property (strong, readonly)  NSString*      ErrorString;
```

[Availability]
SDK 1.0.0 and later

12-2-15 FlagWhenIdle

This property specifies the option whether to receive the statusUpdateEvent when an error condition is cleared. If it is set to YES, then the statusUpdateEvent is generated whenever the error condition is cleared.

The initial setting is NO, and it is initialized whenever the device is open.

[Declare]

```
@property (nonatomic)        BOOL            FlagWhenIdle;
```

[Availability]
SDK 1.0.0 and later

12-3 Available Method

12-3-1 open

This method initiates the use of the printer class, and it includes the initialization process such as memory allocation. This method should be called first before calling the claim and other subsequent methods.

[See Also]

- Constants (Defines) :: OpenResult Code
- [Common] Device Controller Class Reference :: Available Properties :: OpenResult
- [Common] Device Controller Class Reference :: Available Properties :: ldn (Logical Device Name)
- [Printer] Printer List Class Reference :: addDevice

[Declare]

- [Common] Device Controller Class Reference :: Available Method :: open

[Parameters]

(NSString*) logicalDeviceName

- Model name of the device to open or stored device name Refer to ldn (Logical Device Name)
- The device to use should be added in advance using the [Printer] Printer List Class Reference :: addDevice.

[Return Value]

NSInteger

- UPOS_SUCCESS if the operation is successful.

[Example]

```
UPOSPrinterController* _uposPrinterController = [ UPOSPrinterController new];
UPOSPrinters* printerList    = [_uposPrinterController    getRegisteredDevice];

if(UPOS_SUCCESS == [_uposPrinterController open:@"type1"])
{
        if(UPOS_SUCCESS == [_uposPrinterController claim:3000])
        {
                _uposDeviceController.deviceEnabled = YES;
                        //    Printer can be used now.

        }
}
```

[Availability]

SDK 1.0.0 and later

12-3-2 claim

This method tries to open the port specified in the device information, and it includes some initialization processes such as memory allocation and initialization.

This method should be called before using the device.

[See Also]
- Constants (Defines) :: Result Code
- [Common] Device Controller Class Reference :: Available Method :: open
- [Printer] Printer Controller Class Reference :: Available Method :: open
- [Common] Device Controller Class Reference :: Available Method :: claim
- [Common] Device Controller Class Reference :: Available Properties :: Claimed

[Declare]
- [Common] Device Controller Class Reference :: Available Method :: claim

[Parameters]
(NSInteger) timeout
- The system will try to execute the corresponding operation for the duration specified by this parameter.

[Return Value]
NSInteger
- UPOS_SUCCESS if the operation is successful.

[Example]

```
UPOSPrinterController* _uposPrinterController = [ UPOSPrinterController new];
UPOSPrinters* printerList    = [_uposPrinterController    getRegisteredDevice];

if(UPOS_SUCCESS == [_uposPrinterController open:@"type1"])
{
        if(UPOS_SUCCESS == [_uposPrinterController claim:3000])
        {
                _uposDeviceController.deviceEnabled = YES;
                        //    Printer can be used now.
        }
}
```

[Availability]
SDK 1.0.0 and later

12-3-3 releaseDevice

This method terminates the use of the port of the claimed device and releases the physical resources. Some of the memory resources may also be released as a result.

[See Also]

- Constants (Defines) :: Result Code
- [Common] Device Controller Class Reference :: Available Method :: open
- [Printer] Printer Controller Class Reference :: Available Method :: open
- [Common] Device Controller Class Reference :: Available Method :: claim
- [Printer] Printer Controller Class Reference :: Available Method :: claim
- [Common] Device Controller Class Reference::Available Properties::DeviceEnabled
- [Common] Device Controller Class Reference :: Available Properties :: Claimed
- [Printer] Printer Controller Class Reference :: Available Method :: releaseDevice

[Declare]

- [Common] Device Controller Class Reference :: Available Method :: releaseDevice

[Parameters]

(NSString*) logicalDeviceName

- Model name of the device to open or stored device name
  Refer to ldn (Logical Device Name)

[Return Value]

NSInteger

- UPOS_SUCCESS if the operation is successful.

[Example]

```
UPOSPrinterController* _uposPrinterController = [ UPOSPrinterController new];
UPOSPrinters* printerList    = [_uposPrinterController    getRegisteredDevice];

///    After using the printer


//    Printer Closing procedure
_uposDeviceController.deviceEnabled = NO;
[_uposPrinterController releseDevice];
[_uposPrinterController close];
```

[Availability]

SDK 1.0.0 and later

12-3-4 close

This method terminates the use of the open device.
Some of the memory resources may also be released as a result.

[See Also]
- Constants (Defines) :: Result Code
- [Common] Device Controller Class Reference :: Available Method :: open
- [Printer] Printer Controller Class Reference :: Available Method :: open
- [Common] Device Controller Class Reference :: Available Method :: claim
- [Printer] Printer Controller Class Reference :: Available Method :: claim
- [Common] Device Controller Class Reference::Available Properties::DeviceEnabled
- [Common] Device Controller Class Reference :: Available Properties :: Claimed
- [Common] Device Controller Class Reference :: Available Method :: close

[Declare]
- [Common] Device Controller Class Reference :: Available Method :: close

[Parameters]
(NSInteger) timeout
- The system will try to execute the corresponding operation for the duration specified by this parameter.

[Return Value]
NSInteger
- UPOS_SUCCESS if the operation is successful

[Example]

```
UPOSPrinterController* _uposPrinterController = [ UPOSPrinterController new];
UPOSPrinters* printerList    = [_uposPrinterController    getRegisteredDevice];

///    After using the printer


//    Printer Closing procedure
_uposDeviceController.deviceEnabled = NO;
[_uposPrinterController releaseDevice];
[_uposPrinterController close];
```

[Availability]
SDK 1.0.0 and later

12-3-5 cutPaper

This methods cut the paper if the corresponding model has the Auto Cutter.
This method is available after executing open-claim-enable methods.
The operation is asynchronous if AsyncMode is set to True.

[See Also]

- Constants (Defines) :: Result Code
- [Common] Device Controller Class Reference :: Available Method :: open
- [Printer] Printer Controller Class Reference :: Available Method :: open
- [Common] Device Controller Class Reference :: Available Method :: claim
- [Printer] Printer Controller Class Reference :: Available Method :: claim
- [Common] Device Controller Class Reference::Available Properties::DeviceEnabled
- [Common] Device Controller Class Reference :: Available Properties :: Claimed

[Declare]

```
-(NSInteger) cutPaper : (NSInteger)percentage;
```

[Parameters]

(NSInteger) percentage

- This parameter specifies the paper cutting level.
- Higher number means more cutting.

[Return Value]

NSInteger

- UPOS_SUCCESS if the operation is successful.
  (Refer to Constants (Defines) :: Result Code)

[Example]

```
UPOSPrinterController* _uposPrinterController = [ UPOSPrinterController new];
UPOSPrinters* printerList    = [_uposPrinterController    getRegisteredDevice];

///    After printer initialization (open-claim-deviceEnable)
[_uposPrinterController cutPaper:100];
//    Printer Closing procedure
_uposDeviceController.deviceEnabled = NO;
[_uposPrinterController releaseDevice];
[_uposPrinterController close];
```

[Availability]

SDK 1.0.0 and later

12-3-6 markFeed

This method feeds the paper to the next printing position.
This method is available after executing open-claim-enable methods.
The operation is asynchronous if AsyncMode is set to True.

[See Also]

- Constants (Defines) :: Result Code
- [Common] Device Controller Class Reference :: Available Method :: open
- [Printer] Printer Controller Class Reference :: Available Method :: open
- [Common] Device Controller Class Reference :: Available Method :: claim
- [Printer] Printer Controller Class Reference :: Available Method :: claim
- [Common] Device Controller Class Reference::Available Properties::DeviceEnabled
- [Common] Device Controller Class Reference :: Available Properties :: Claimed

[Declare]

```
-(NSInteger) markFeed : (NSInteger)type;
```

[Parameters]

(NSInteger) type

- This parameter specifies the Mark type.

Value	Description
PTR_MF_TO_TAKEUP	The printer keeps feeding the paper until the mark on the paper is recognized by the mark sensor of printer.
PTR_MF_TO_CUTTER	The printer feeds the paper until the printed contents on the paper are not cut.
PTR_MF_TO_CURRENT_TOP	Not supported
PTR_MF_TO_NEXT_TOF	The printer keeps feeding the paper until the next mark of the paper is recognized by the printer's mark sensor.

[Return Value]

NSInteger

- UPOS_SUCCESS if the operation is successful.
  (Refer to Constants (Defines) :: Result Code)

[Example]

```
UPOSPrinterController* _uposPrinterController = [ UPOSPrinterController new];
UPOSPrinters* printerList    = [_uposPrinterController    getRegisteredDevice];

///    After printer initialization (open-claim-deviceEnable)
  [_uposPrinterController markeFeed:PTR_MF_TO_TAKEUP];
//    Printer Closing procedure
_uposDeviceController.deviceEnabled = NO;
[_uposPrinterController releaseDevice];
[_uposPrinterController close];
```

[Availability]

SDK 1.0.0 and later

12-3-7 printBarcode

This method prints barcodes.

[See Also]

- Constants (Defines) :: Result Code
- Constants (Defines) :: Barcode Type
- Constants (Defines) :: Alignment
- [Common] Device Controller Class Reference :: Available Method :: open
- [Printer] Printer Controller Class Reference :: Available Method :: open
- [Common] Device Controller Class Reference :: Available Method :: claim
- [Printer] Printer Controller Class Reference :: Available Method :: claim
- [Common] Device Controller Class Reference::Available Properties::DeviceEnabled
- [Common] Device Controller Class Reference :: Available Properties :: Claimed

[Declare]

```
-(NSInteger) printBarcode: (NSInteger) station
                     data : (NSString*) data
               symbology : (NSInteger) symbology
                  height : (NSInteger) height
                   width: (NSInteger) width
               alignment : (NSInteger) alignment
               textPostion: (NSInteger) textPosition;
```

[Parameters]

(NSInteger) station
- Fixed to PTR_S_RECEIPT

(NSString) data
- Data to be included in the barcode, which may vary, depending on the barcode type.

(NSInteger) symbology
- Type of barcode. (Refer to Constants (Defines) :: Barcode Type)

(NSInteger) height
- Height of the barcode.

(NSInteger) width
- Width of the barcode.

(NSInteger) alignment
- Barcode alignment. (Refer to Constants (Defines) :: Alignment)

(NSInteger) textProsition
- Position of the text to be printed with barcode.
  (Refer to Constants (Defines) :: Barcode Text Position)

[Return Value]

NSInteger
- UPOS_SUCCESS if the operation is successful.
  (Refer to Constants (Defines) :: Result Code)

[Example]

```
UPOSPrinterController* _uposPrinterController = [ UPOSPrinterController new];
UPOSPrinters* printerList    = [_uposPrinterController    getRegisteredDevice];

///    After printer initialization (open-claim-deviceEnable)
[_uposPrinterController printBarcode          :PTR_S_RECEIPT
                                       data     :@"1234567890123"
                                 symbology     : PTR_BCS_EAN13
                                       height  :100
                                       width   :200
                                  alignment    : PTR_BC_CENTER
                                   textPostion:PTR_BC_TEXT_BELOW];

//    Printer Closing procedure
_uposDeviceController.deviceEnabled = NO;
[_uposPrinterController releaseDevice];
[_uposPrinterController close];
```

[Availability]

SDK 1.0.0 and later

12-3-8 printBitmap (File Printing)

This method prints images.

[See Also]

- Constants (Defines) :: Result Code
- Constants (Defines) :: Alignment
- [Common] Device Controller Class Reference :: Available Method :: open
- [Printer] Printer Controller Class Reference :: Available Method :: open
- [Common] Device Controller Class Reference :: Available Method :: claim
- [Printer] Printer Controller Class Reference :: Available Method :: claim
- [Common] Device Controller Class Reference::Available Properties::DeviceEnabled
- [Common] Device Controller Class Reference :: Available Properties :: Claimed

[Declare]

```
-(NSInteger)     printBitmap    : (NSInteger) station
                    Filename    : (NSString*) fileName
                       width    : (NSInteger) width
                   alignment    : (NSInteger) alignment;
```

[Parameters]

(NSInteger) station
- Fixed to PTR_S_RECEIPT

(NSString) fileName
- Path of the image file.

(NSInteger) width
- Width of the image .
  (The image will be printed at the specified width within the range supported by the printer, irrespective of the size of the image.)

(NSInteger) alignment
- Alignment of the image. (Refer to Constants (Defines) :: Alignment)

[Return Value]

NSInteger
- UPOS_SUCCESS if the operation is successful.
  (Refer to Constants (Defines) :: Result Code)

[Example]

```
UPOSPrinterController* _uposPrinterController = [ UPOSPrinterController new];
UPOSPrinters* printerList    = [_uposPrinterController    getRegisteredDevice];

///    After printer initialization (open-claim-deviceEnable)
NSString    *path    =    [[NSBundle    mainBundle]    pathForResource:@"Sample"
ofType:@"png"];
[_uposPrinterController printBitmap:PTR_S_RECEIPT
                                fileName        :path
                                      width     :200
                                alignment       : PTR_BM_CENTER];

//    Printer Closing procedure
_uposDeviceController.deviceEnabled = NO;
[_uposPrinterController releaseDevice];
[_uposPrinterController close];
```

[Availability]

SDK 1.0.0 and later

12-3-9 printBitmap (UIImage Printing)

This method prints images.

[See Also]

- Constants (Defines) :: Result Code
- Constants (Defines) :: Alignment
- [Common] Device Controller Class Reference :: Available Method :: open
- [Printer] Printer Controller Class Reference :: Available Method :: open
- [Common] Device Controller Class Reference :: Available Method :: claim
- [Printer] Printer Controller Class Reference :: Available Method :: claim
- [Common] Device Controller Class Reference::Available Properties::DeviceEnabled
- [Common] Device Controller Class Reference :: Available Properties :: Claimed

[Declare]

```
-(NSInteger) printBitmap      : (NSInteger) station
                   image      : (NSString*) image
                   width      : (NSInteger) width
               alignment      : (NSInteger) alignment;
```

[Parameters]

(NSInteger) station
- Fixed to PTR_S_RECEIPT

(UIImage) image
- Image to print

(NSInteger) width
- Width of the printed image
  (The image will be printed at the specified width within the range supported by the printer, irrespective of the size of the image.)

(NSInteger) alignment
- Alignment of the image (Refer to Constants (Defines) :: Alignment)

[Return Value]

NSInteger
- UPOS_SUCCESS if the operation is successful.
  (Refer to Constants (Defines) :: Result Code)

[Example]

```
UPOSPrinterController* _uposPrinterController = [ UPOSPrinterController new];
UPOSPrinters* printerList    = [_uposPrinterController    getRegisteredDevice];

///    After printer initialization (open-claim-deviceEnable)
IBOulet UIImageView * _imageView; // Views that are connected to resources
[_uposPrinterController printBitmap:PTR_S_RECEIPT
                                 image  :_imageView.image
                                 width  :200
                         alignment      : PTR_BM_CENTER];

//    Printer Closing procedure
_uposDeviceController.deviceEnabled = NO;
[_uposPrinterController releaseDevice];
[_uposPrinterController close];
```

[Availability]

SDK 1.0.0 and later

12-3-10 printNormal

This method prints text.

[See Also]

- Constants (Defines) :: Result Code
- [Common] Device Controller Class Reference :: Available Method :: open
- [Printer] Printer Controller Class Reference :: Available Method :: open
- [Common] Device Controller Class Reference :: Available Method :: claim
- [Printer] Printer Controller Class Reference :: Available Method :: claim
- [Common] Device Controller Class Reference::Available Properties::DeviceEnabled
- [Common] Device Controller Class Reference :: Available Properties :: Claimed

[Declare]

```
-(NSInteger) printNormal        : (NSInteger) station
                         data   : (NSString*) data;
```

[Parameters]

(NSInteger) station

- Fixed to PTR_S_RECEIPT

(NSString) data

- This parameter is the data to print.
- It includes printable text, escape sequences, carriage returns, and line feeds data.

[Return Value]

NSInteger

- UPOS_SUCCESS if the operation is successful.
  (Refer to Constants (Defines) :: Result Code)

[Example]

```
UPOSPrinterController* _uposPrinterController = [ UPOSPrinterController new];
UPOSPrinters* printerList    = [_uposPrinterController    getRegisteredDevice];

///    After printer initialization (open-claim-deviceEnable)
IBOulet UIImageView * _imageView; // Views that are connected to resources
[_uposPrinterController printNormal          :PTR_S_RECEIPT
                                   data      :@"|N|cAPrint
CenterAlienedText\r\n"];

//    Printer Closing procedure
_uposDeviceController.deviceEnabled = NO;
[_uposPrinterController releaseDevice];
[_uposPrinterController close];
```

[Availability]

SDK 1.0.0 and later

12-3-11 transactionPrint

This method prints text.

[See Also]

- Constants (Defines) :: Result Code
- Constants (Defines) :: Transaction Print
- [Common] Device Controller Class Reference :: Available Method :: open
- [Printer] Printer Controller Class Reference :: Available Method :: open
- [Common] Device Controller Class Reference :: Available Method :: claim
- [Printer] Printer Controller Class Reference :: Available Method :: claim
- [Common] Device Controller Class Reference::Available Properties::DeviceEnabled
- [Common] Device Controller Class Reference :: Available Properties :: Claimed

[Declare]

```
-(NSInteger)        printNormal: (NSInteger) station
                                 control : (NSInteger) control;
```

[Parameters]

(NSInteger) station

- Fixed to PTR_S_RECEIPT

(NSString) control

- Transaction Mode (Refer to Constants (Defines) :: Transaction Print)

[Return Value]

NSInteger

- UPOS_SUCCESS if the operation is successful.
  (Refer to Constants (Defines) :: Result Code)

[Example]

```
UPOSPrinterController* _uposPrinterController = [ UPOSPrinterController new];
UPOSPrinters* printerList    = [_uposPrinterController    getRegisteredDevice];

///    After printer initialization (open-claim-deviceEnable)
IBOulet UIImageView * _imageView; // Views that are connected to resources
[_uposPrinterController transactionPrint    :PTR_S_RECEIPT
                       Control    : PTR_TR_TRANSACTION];


[_uposPrinterController printNormal    :PTR_S_RECEIPT
                        data    :@"|N|cAPrint CenterAlienedText 1 \r\n"];
[_uposPrinterController printNormal    :PTR_S_RECEIPT
                        data    :@"|N|cAPrint CenterAlienedText 2 \r\n"];
[_uposPrinterController printNormal    :PTR_S_RECEIPT
                        data    :@"|N|cAPrint CenterAlienedText 3 \r\n"];


// Contents in the buffer are printed, and the transaction mode is terminated.
[_uposPrinterController transactionPrint    :PTR_S_RECEIPT
                       control    : PTR_TR_NORMAL];
//    Printer Closing procedure
_uposDeviceController.deviceEnabled = NO;
[_uposPrinterController releaseDevice];
[_uposPrinterController close];
```

[Availability]

SDK 1.0.0 and later

12-3-12 displayString

This method displays a string on the Customer Display.

[See Also]

- Constants (Defines) :: Result Code
- [Common] Device Controller Class Reference :: Available Method :: open
- [Printer] Printer Controller Class Reference :: Available Method :: open
- [Common] Device Controller Class Reference :: Available Method :: claim
- [Printer] Printer Controller Class Reference :: Available Method :: claim
- [Common] Device Controller Class Reference::Available Properties::DeviceEnabled
- [Common] Device Controller Class Reference :: Available Properties :: Claimed

[Declare]

```
-(NSInteger) displayString:(NSString*)string;
```

[Parameters]

(NSString*) string

- Data to display

[Return Value]

NSInteger

- UPOS_SUCCESS if the operation is successful.
  (Refer to Constants (Defines) :: Result Code)

[Example]

```
UPOSPrinterController* _uposPrinterController = [ UPOSPrinterController new];
UPOSPrinters* printerList    = [_uposPrinterController    getRegisteredDevice];

///    After printer initialization (open-claim-deviceEnable)
  [_uposPrinterController displayString: @"testDisplay" ];

//    Printer Closing procedure
_uposDeviceController.deviceEnabled = NO;
[_uposPrinterController releaseDevice];
[_uposPrinterController close];
```

[Availability]

SDK 1.0.0 and later

12-3-13 displayStringAtLine

This method displays a string in a specific line of the Customer Display. all existing data will be erased. In case that input data excesses one row, the results varies depending on settings of Customer Display. Only Bixolon Customer Display is supported.

[See Also]

- Constants (Defines) :: Result Code
- [Common] Device Controller Class Reference :: Available Method :: open
- [Printer] Printer Controller Class Reference :: Available Method :: open
- [Common] Device Controller Class Reference :: Available Method :: claim
- [Printer] Printer Controller Class Reference :: Available Method :: claim
- [Common] Device Controller Class Reference::Available Properties::DeviceEnabled
- [Common] Device Controller Class Reference :: Available Properties :: Claimed

[Declare]

```
-(NSInteger) displayStringAtLine:(NSInteger)line
                                 data:(NSString*)data;
```

[Parameters]

(NSInteger)line

- Line number to display. (Range: 1 ≤ line ≤ 2)

(NSString*)data

- Data to display.

[Return Value]

NSInteger

- UPOS_SUCCESS if the operation is successful.
  (Refer to Constants (Defines) :: Result Code)

[Example]

```
UPOSPrinterController* _uposPrinterController = [ UPOSPrinterController new];
UPOSPrinters* printerList    = [_uposPrinterController    getRegisteredDevice];

///    After printer initialization (open-claim-deviceEnable)
  [_uposPrinterController displayStringAtLine:1 data:@"testDisplay"];

//    Printer Closing procedure
_uposDeviceController.deviceEnabled = NO;
[_uposPrinterController releaseDevice];
[_uposPrinterController close];
```

[Availablility]

SDK 1.0.0 and later

12-3-14 clearScreen

This method clears the Screen.

[See Also]

- Constants (Defines) :: Result Code
- [Common] Device Controller Class Reference :: Available Method :: open
- [Printer] Printer Controller Class Reference :: Available Method :: open
- [Common] Device Controller Class Reference :: Available Method :: claim
- [Printer] Printer Controller Class Reference :: Available Method :: claim
- [Common] Device Controller Class Reference::Available Properties::DeviceEnabled
- [Common] Device Controller Class Reference :: Available Properties :: Claimed

[Declare]

```
-(NSInteger) clearScreen;
```

[Return Value]

NSInteger

- UPOS_SUCCESS if the operation is successful.
  (Refer to Constants (Defines) :: Result Code)

[Example]

```
UPOSPrinterController* _uposPrinterController = [ UPOSPrinterController new];
UPOSPrinters* printerList    = [_uposPrinterController    getRegisteredDevice];

///    After printer initialization (open-claim-deviceEnable)
  [_uposPrinterController clearScreen];

//    Printer Closing procedure
_uposDeviceController.deviceEnabled = NO;
[_uposPrinterController releaseDevice];
[_uposPrinterController close];
```

[Availablility]

SDK 1.0.0 and later

12-3-15 setDisplayCharacterSet

This function sets the codepage of the Customer Display.

[See Also]

- Constants (Defines) :: Result Code
- [Common] Device Controller Class Reference :: Available Method :: open
- [Printer] Printer Controller Class Reference :: Available Method :: open
- [Common] Device Controller Class Reference :: Available Method :: claim
- [Printer] Printer Controller Class Reference :: Available Method :: claim
- [Common] Device Controller Class Reference::Available Properties::DeviceEnabled
- [Common] Device Controller Class Reference :: Available Properties :: Claimed

[Declare]

```
-(NSInteger) setDisplayCharacterSet:(NSInteger)characterSet;
```

[Parameters]

(NSInteger)characterSet

- Codepage value of Customer Display

[Return Value]

NSInteger

- UPOS_SUCCESS if the operation is successful.
  (Refer to Constants (Defines) :: Result Code)

[Example]

```
UPOSPrinterController* _uposPrinterController = [ UPOSPrinterController new];
UPOSPrinters* printerList    = [_uposPrinterController    getRegisteredDevice];

///    After printer initialization (open-claim-deviceEnable)
  [_uposPrinterController setDisplayCharacterSet:866];

//    Printer Closing procedure
_uposDeviceController.deviceEnabled = NO;
[_uposPrinterController releaseDevice];
[_uposPrinterController close];
```

[Availablility]

SDK 1.0.0 and later

12-3-16 setInternationalCharacterset

This function sets the international character set of the Customer Display

[See Also]

- Constants (Defines) :: Result Code
- [Common] Device Controller Class Reference :: Available Method :: open
- [Printer] Printer Controller Class Reference :: Available Method :: open
- [Common] Device Controller Class Reference :: Available Method :: claim
- [Printer] Printer Controller Class Reference :: Available Method :: claim
- [Common] Device Controller Class Reference::Available Properties::DeviceEnabled
- [Common] Device Controller Class Reference :: Available Properties :: Claimed

[Declare]

```
-(NSInteger) setInternationalCharacterset:(NSInteger) internationalCharset;
```

[Parameters]

(NSInteger) internationalCharset

- International character set value of. Customer Display

[Return Value]

NSInteger

- UPOS_SUCCESS if the operation is successful.
  (Refer to Constants (Defines) :: Result Code)

[Example]

```
UPOSPrinterController* _uposPrinterController = [ UPOSPrinterController new];
UPOSPrinters* printerList    = [_uposPrinterController    getRegisteredDevice];

///    After printer initialization (open-claim-deviceEnable)
  [_uposPrinterController setInternationalCharacterset:1];

//    Printer Closing procedure
_uposDeviceController.deviceEnabled = NO;
[_uposPrinterController releaseDevice];
[_uposPrinterController close];
```

[Availablility]

SDK 1.0.0 and later

12-3-17 storeImage

This function stores the image data of the Customer Display.
Only Bixolon Customer Display is supported.

[See Also]

- Constants (Defines) :: Result Code
- [Common] Device Controller Class Reference :: Available Method :: open
- [Printer] Printer Controller Class Reference :: Available Method :: open
- [Common] Device Controller Class Reference :: Available Method :: claim
- [Printer] Printer Controller Class Reference :: Available Method :: claim
- [Common] Device Controller Class Reference::Available Properties::DeviceEnabled
- [Common] Device Controller Class Reference :: Available Properties :: Claimed

[Declare]

```
-(NSInteger) storeImage:(UIImage*)image
                  width:(NSInteger)width
            imageNumber:(NSInteger)imageNumber;
```

[Parameters]

(UIImage*)image
- Image data to save.

:(NSInteger)width
- Image width

(NSInteger)imageNumber
- Number to save Image. Available from 1 to 5.

[Return Value]

NSInteger
- UPOS_SUCCESS if the operation is successful.
  (Refer to Constants (Defines) :: Result Code)

[Example]

```
UPOSPrinterController* _uposPrinterController = [ UPOSPrinterController new];
UPOSPrinters* printerList    = [_uposPrinterController    getRegisteredDevice];

///    After printer initialization (open-claim-deviceEnable)
@property (retain, nonatomic) IBOutlet UIImageView *imageView;
[_uposPrinterController storeImage: imageView .image
                  width: imageView.image.size.width imageNumber:1];

//    Printer Closing procedure
_uposDeviceController.deviceEnabled = NO;
[_uposPrinterController releaseDevice];
[_uposPrinterController close];
```

[Availablility]

SDK 1.0.0 and later

12-3-18 storeImageFile

This function stores the image file of the Customer Display.
Only Bixolon Customer Display is supported.

[See Also]
- Constants (Defines) :: Result Code
- [Common] Device Controller Class Reference :: Available Method :: open
- [Printer] Printer Controller Class Reference :: Available Method :: open
- [Common] Device Controller Class Reference :: Available Method :: claim
- [Printer] Printer Controller Class Reference :: Available Method :: claim
- [Common] Device Controller Class Reference::Available Properties::DeviceEnabled
- [Common] Device Controller Class Reference :: Available Properties :: Claimed

[Declare]

```
-(NSInteger) storeImageFile:(NSString*)filename
                      width:(NSInteger)width
                imageNumber:(NSInteger)imageNumber;
```

[Parameters]
(NSString*)filename
- The full path of the image file

:(NSInteger)width
- Image width

(NSInteger)imageNumber
- Number to save Image. Available from 1 to 5

[Return Value]
NSInteger
- UPOS_SUCCESS if the operation is successful.
  (Refer to Constants (Defines) :: Result Code)

[Example]

```
UPOSPrinterController* _uposPrinterController = [ UPOSPrinterController new];
UPOSPrinters* printerList    = [_uposPrinterController    getRegisteredDevice];

///    After printer initialization (open-claim-deviceEnable)
NSString *path
      = [[NSBundle mainBundle] pathForResource:@"Sample" ofType:@"png"];

[_uposPrinterController storeImageFile:path
    width:[UIImage imageWithContentsOfFile:
    [NSString stringWithFormat:@"%@",path]].size.width imageNumber:1];

//    Printer Closing procedure
_uposDeviceController.deviceEnabled = NO;
[_uposPrinterController releaseDevice];
[_uposPrinterController close];
```

[Availablility]
SDK 1.0.0 and later

12-3-19 displayImage

This function displays the image stored in the Customer Display.
Only Bixolon Customer Display is supported.

[See Also]

- Constants (Defines) :: Result Code
- [Common] Device Controller Class Reference :: Available Method :: open
- [Printer] Printer Controller Class Reference :: Available Method :: open
- [Common] Device Controller Class Reference :: Available Method :: claim
- [Printer] Printer Controller Class Reference :: Available Method :: claim
- [Common] Device Controller Class Reference::Available Properties::DeviceEnabled
- [Common] Device Controller Class Reference :: Available Properties :: Claimed

[Declare]

```
-(NSInteger) displayImage:(NSInteger)imageNumber
                          xPos:(NSInteger)xPos
                          yPos:(NSInteger)yPos;
```

[Parameters]

(NSInteger)imageNumber

- Number to save Image. Available from 1 to 5

(NSInteger)xPos

- x-coordinate

(NSInteger)yPos

- y-coordinate

[Return Value]

NSInteger

- UPOS_SUCCESS if the operation is successful.
  (Refer to Constants (Defines) :: Result Code)

[Example]

```
UPOSPrinterController* _uposPrinterController = [ UPOSPrinterController new];
UPOSPrinters* printerList    = [_uposPrinterController    getRegisteredDevice];

///    After printer initialization (open-claim-deviceEnable)
[_uposPrinterController displayImage:1 xPos:0 yPos:0];

//    Printer Closing procedure
_uposDeviceController.deviceEnabled = NO;
[_uposPrinterController releaseDevice];
[_uposPrinterController close];
```

[Availablility]

SDK 1.0.0 and later

12-3-20 clearImage

This function deletes the image stored in the Customer Display and erases the image displayed on the screen. Only Bixolon Customer Display is supported.

[See Also]

- Constants (Defines) :: Result Code
- [Common] Device Controller Class Reference :: Available Method :: open
- [Printer] Printer Controller Class Reference :: Available Method :: open
- [Common] Device Controller Class Reference :: Available Method :: claim
- [Printer] Printer Controller Class Reference :: Available Method :: claim
- [Common] Device Controller Class Reference::Available Properties::DeviceEnabled
- [Common] Device Controller Class Reference :: Available Properties :: Claimed

[Declare]

```
-(NSInteger) clearImage:(bool)isAll
                imageNumber:(NSInteger)imageNumber;
```

[Parameters]

(bool)isAll

- Whether to delete all stored images.
- if true, remove all stored images.
- If false, a specific number must be specified.

(NSInteger)imageNumber

- Number to save Image. Available from 1 to 5

[Return Value]

NSInteger

- UPOS_SUCCESS if the operation is successful.
  (Refer to Constants (Defines) :: Result Code)

[Example]

```
UPOSPrinterController* _uposPrinterController = [ UPOSPrinterController new];
UPOSPrinters* printerList    = [_uposPrinterController    getRegisteredDevice];

///    After printer initialization (open-claim-deviceEnable)
[_uposPrinterController clearImage:NO imageNumber:1];

//    Printer Closing procedure
_uposDeviceController.deviceEnabled = NO;
[_uposPrinterController releaseDevice];
[_uposPrinterController close];
```

[Availablility]

SDK 1.0.0 and later

12-3-21 checkBattStatus

Request battery status information..

[See Also]

- Constants (Defines) :: Result Code
- [Common] Device Controller Class Reference :: Available Method :: open
- [Printer] Printer Controller Class Reference :: Available Method :: open
- [Common] Device Controller Class Reference :: Available Method :: claim
- [Printer] Printer Controller Class Reference :: Available Method :: claim
- [Common] Device Controller Class Reference::Available Properties::DeviceEnabled
- [Common] Device Controller Class Reference :: Available Properties :: Claimed

[Declare]

```
-(long) checkBattStatus;
```

[Return Value]

long

- UPOS_SUCCESS if the operation is successful.
  (Refer to Constants (Defines) :: Result Code)

[Example]

```
UPOSPrinterController* _uposPrinterController = [ UPOSPrinterController new];
UPOSPrinters* printerList    = [_uposPrinterController    getRegisteredDevice];

///     After printer initialization (open-claim-deviceEnable)

[_uposPrinterController checkBattStatus];

//     Printer Closing procedure
_uposDeviceController.deviceEnabled = NO;
[_uposPrinterController releaseDevice];
[_uposPrinterController close];
```

[Availablility]

SDK 1.0.13 and later

12-4 Available Delegate

12-4-1 StatusUpdateEvent

This is an event that is generated when the status of the printer is changed.

[See Also]
- [Common] Delegater Class Reference :: Available Delegate :: StatusUpdateEvent
- Constants (Defines) :: StatusUpdateEvent

[Declare]
- Refer to [Common] Delegater Class Reference :: Available Delegate :: StatusUpdateEvent

[Parameters]
(NSNumber*) Status
- It includes the changed status value.
  (Refer to Constants (Defines) :: StatusUpdateEvent)

[Return Value]
- void

[Example]

```
-(void) StatusUpdateEvent: (NSNumber*) Status
{
    NSLog(@"!!!!!!!!!!!!    StatusUpdateEvent : %ld !!!!!!!!!!!", (long)Status.integerValue);
    NSString *message;
    switch([ Status integerValue])
        {
        case PTR_SUE_COVER_OPEN:
            message = [NSString stringWithFormat:@"[StatusUpdateEvent] Cover
Open"];
            break;
        case PTR_SUE_COVER_OK:
            message = [NSString stringWithFormat:@"[StatusUpdateEvent] Cover
OK"];
            break;
        case PTR_SUE_REC_EMPTY:
            message = [NSString stringWithFormat:@"[StatusUpdateEvent] Paper
Empty"];
            break;
        case PTR_SUE_REC_PAPEROK:
            message = [NSString stringWithFormat:@"[StatusUpdateEvent] Paper
OK"];
            break;
        case PTR_SUE_REC_NEAREMPTY:
            message = [NSString stringWithFormat:@"[StatusUpdateEvent] Paper
Near End"];
            break;
        case UPOS_SUE_POWER_OFF:
        case UPOS_SUE_POWER_OFF_OFFLINE:
        case UPOS_SUE_POWER_OFFLINE:
            message = [NSString stringWithFormat:@"[StatusUpdateEvent] Device off
or offLine"];
            break;
        case UPOS_SUE_POWER_ONLINE:
            message = [NSString stringWithFormat:@"[StatusUpdateEvent] Device
OnLine"];
            break;
        default:
            message = [NSString stringWithFormat:@"[StatusUpdateEvent]
UNKNOWN"];
        }

        NSLog(@"%@", message)
}
```

[Availability]

SDK 1.0.0 and later

12-4-2 OutputCompleteEvent

This event is generated when each device completes the received output request.

[See Also]
- [Printer] Printer Controller Class Reference :: Available Method
- [Printer] Printer Controller Class Reference :: Available Method :: cutPaper
- [Printer] Printer Controller Class Reference :: Available Method :: markFeed
- [Printer] Printer Controller Class Reference :: Available Method :: printBarcode
- [Printer] Printer Controller Class Reference :: Available Method :: printBitmap (File Printing)
- [Printer] Printer Controller Class Reference :: Available Method :: printBitmap (UIImage Printing)
- [Printer] Printer Controller Class Reference :: Available Method :: printNormal
- [Printer] Printer Controller Class Reference :: Available Method :: transactionPrint

[Declare]
- Refer to [Common] Delegater Class Reference :: Available Delegate :: DataEvent

[Parameters]
(NSNumber*) Status
- It includes the changed status value.
  (Refer to Constants (Defines) :: StatusUpdateEvent)

[Return Value]
- void

[Declare]

```
- (void)OutputCompleteEvent:(NSNumber*)OutputID;
```

[Availability]
SDK 1.0.0 and later

13. [CashDrawer] Cash Drawer Class Reference

13-1 Overview

The UPOSCashDrawer Class is the object that contains the information about target devices controlled by the UPOSCDController.

[See Also]
- [Common] Device Class Reference

13-2 Available Properties

[See Also]
- Refer to [Common] Device Class Reference :: Available Properties

13-2-1 selectedPrinterName

Name of the printer to connect the cash drawer

[Declare]

```
@property (retain)   NSString          *selectedPrinterName;
```

[Availability]
SDK 1.0.0 and later

13-2-2 pinNumber

This is the PIN number that is required to open the cash drawer.

[Declare]

```
@property (retain)   NSNumber          *pinNumber;
```

[Availability]
SDK 1.0.0 and later

13-2-3 pinLevel

This is the level of sensing for monitoring the cash drawer open status.
Change this option if the status of the cash drawer is reversed in the display.

[Declare]

```
@property (retain)   NSNumber          *pinLevel;
```

[Availability]
SDK 1.0.0 and later

13-2-4 pulseOnTime

This determines the time duration to turn on the signal when opening the cash drawer,

[Declare]

```
@property (retain)   NSNumber          *pinLevel;
```

[Availability]
SDK 1.0.0 and later

13-2-5 pulseOffTime

This determines the time when the signal will be turned off after supplying the signal while opening the cash drawer.

[Declare]

```
@property (retain)   NSNumber          *pinLevel;
```

[Availability]
SDK 1.0.0 and later

13-3 Available Method

- Refer to [Common] Device Class Reference :: Available Method

14. [CashDrawer] Cash Drawer List Class Reference

Inherits from	UPOSDevices
Framework	libBixolonUPOS.a
[Availability]	iOS 7.0 and later
Class name	UPOSCashDrawers
Declared	UPOSDevices.h

14-1 Overview

The UPOSCashDrawers Class is the object that contains the list of cash drawers stored in iDevice. (It is a set of UPOSCashDrawer objects.)

[See Also]

- Cash Drawer Class Reference

14-2 Available Properties

- Not applicable

14-3 Available Method

14-3-1 getList

This method reads the stored device list.

[See Also]
- Device Class Reference (Common Device)
- Device Controller Class Reference (Common Controller)
- Cash Drawer Class Reference

[Declare]
- Refer to Cash Drawer List Class Reference (stored list of cash drawers) :: getList

[Parameters]
None

[Return Value]
- NSMutableArray*:
  The list of devices is stored as an array.
  Each object is stored as a Device Class.

[Example]

```
UPOSCDController* _uposCDController = [UPOSCDController new];
UPOSCashDrawers* cdList    = [_ uposCDController getRegisteredDevice];

NSMutableArray* cdArray = [cdrList getList];

if(cdArray == nil)
{
        NSLog(@"Failed to read the list of CashDrawers.");
}
else
{
        NSLog(@"List of CashDrawers is read.");
}
```

[Availability]
SDK 1.0.0 and later

14-3-2 getDeviceIdentity

This method reads the Identity string of the current device.

[Declare]

- Refer to Device List Class Reference (stored device list) :: getDeviceIdentity

[Parameters]

None

[Return Value]

- NSString*:
  The Identity of the current device is returned as a string.

[Example]

```
UPOSCDController* _uposCDController = [UPOSCDController new];
UPOSCashDrawers* cdList    = [_ uposCDController getRegisteredDevice];

NSString* strIdentity = [cdList getDeviceIdentity];

if(cdArray == nil)
{
        NSLog(@"Failed to read the identity value.");
}
else
{
        NSLog(@" Identity of UPOSCDController :%@.", strIdentity);
}
```

[Availability]

SDK 1.0.0 and later

14-3-3 save

This method saves the current device list.

[See Also]

- Cash Drawer List Class Reference (stored list of cash drawers) :: removeDevice
- Cash Drawer List Class Reference (stored list of cash drawers) :: addDevice

[Declare]

- Refer to Device List Class Reference (stored device list) :: save

[Parameters]

None

[Return Value]

- BOOL:
  YES if the operation is successful.

[Example]

```
UPOSCDController* _uposCDController = [UPOSCDController new];
UPOSCashDrawers* cdList    = [_ uposCDController getRegisteredDevice];

UPOSCashDrawer* newDevice = [UPOSCashDrawer new];
newDevice.modelName       = @"SRP-CDW";
newDevice.ldn             =  @"type1";newDevice.selectedPrinterName  =  @"SRP-
350plusIII";
newDevice.pinNumber = 2;
newDevice.pinLevel = 0;
newDevice.pulseOnTime = 100;
newDevice.pulseOffTime = 400;
[cdList addDevice:newDevice];    // Add Device
[newDevice release];
if([printerList save])
{
        NSLog(@"List is saved.");
}
else
{
        NSLog(@"List cannot be saved.");
}
```

[Availability]

SDK 1.0.0 and later

14-3-4 addDevice

This method adds a device to the current device list.
The device's addition may not be reflected to the device list when the list is refreshed if the data is not saved by using the save method after using this method.

[See Also]
- Cash Drawer List Class Reference (stored list of cash drawers) :: save
- Cash Drawer List Class Reference (stored list of cash drawers) :: removeDevice

[Declare]
- Refer to Device List Class Reference (stored device list) :: addDevice

[Parameters]
- (UPOSDevice*) device
  Object that contains the information about the device to add.

[Return Value]
- BOOL:
  YES if the operation is successful.

[Example]

```
UPOSCDController* _uposCDController = [UPOSCDController new];
UPOSCashDrawers* cdList    = [_ uposCDController getRegisteredDevice];

UPOSCashDrawer* newDevice = [UPOSCashDrawer new];
newDevice.modelName          = @"SRP-CDW";
newDevice.ldn                =@"type1";newDevice.selectedPrinterName = @"SRP-
350plusIII";
newDevice.pinNumber = 2;
newDevice.pinLevel = 0;
newDevice.pulseOnTime = 100;
newDevice.pulseOffTime = 400;
if([cdList addDevice:newDevice])     //    Add device
{
            NSLog(@"Device is added successfully.");
}
else
{
            NSLog(@""Failed to add device.");
}

[newDevice release];
[cdList save];
```

[Availability]
SDK 1.0.0 and later

14-3-5 removeDevice

This method deletes a device from the device list.
The device's deletion may not be reflected to the device list when the list is refreshed if the data is not saved by using the save method after using this method.

[See Also]
- Cash Drawer List Class Reference (stored list of cash drawers) :: save
- Cash Drawer List Class Reference (stored list of cash drawers) :: addDevice

[Declare]
- Refer to Device List Class Reference (stored device list) :: removeDevice

[Parameters]
- (UPOSDevice*) device
  Object that contains the information about the device to add

[Return Value]
- BOOL:
  YES if the operation is successful.

[Example]

```
UPOSCDController* _uposCDController = [UPOSCDController new];
UPOSCashDrawers* cdList    = [_ uposCDController getRegisteredDevice];

UPOSCashDrawer* * willRemoveDevice = [[printerList getList] objectAtIndex:0];
if([cdList removeDevice:willRemoveDevice])     //    Remove device
{
        NSLog(@"0th device is removed from the list.");
}
else
{
        NSLog(@"Failed to remove 0th device from the list.");
}

[newDevice release];
[cdList save];
```

[Availability]
SDK 1.0.0 and later

15. [CashDrawer] Cash Drawer Controller Class Reference

Inherits from	UPOSDeviceController
Framework	libBixolonUPOS.a
[Availability]	iOS 7.0 and later
Class name	UPOSCDController
Declared	UPOSCDController.h

15-1 Overview

The UPOSCDController Class is the main object that controls common functions of the devices supported by this SDK.

15-2 Available Properties

15-2-1 DrawerOpened

This indicates the status of CashDrawer.
YES means that the Drawer is open.

[Declare]

```
@property (readonly)        BOOL        DrawerOpened;
```

[Availability]
SDK 1.0.0 and later

15-3 Available Method

15-3-1 open

This method initiates the use of the cashdrawer class, and it includes the initialization process such as memory allocation. This method should be called first before calling the claim and other subsequent methods.

[See Also]
- Constants (Defines) :: OpenResult Code
- [Common] Device Controller Class Reference :: Available Properties :: OpenResult
- [Common] Device Controller Class Reference :: Available Properties :: ldn (Logical Device Name)
- [CashDrawer] Cash Drawer List Class Reference ::Available Method :: addDevice

[Declare]
- [Common] Device Controller Class Reference :: Available Method :: open

[Parameters]
(NSString*) logicalDeviceName
- Model name of the device to open or stored device name
  Refer to ldn (Logical Device Name)
- [CashDrawer] Cash Drawer List Class Reference :: Available Method :: addDevice
  The device should be added by using the [CashDrawer] Cash Drawer List Class Reference :: addDevice method first in order to use it.

[Return Value]
NSInteger
- UPOS_SUCCESS if the operation is successful.

[Example]

```
UPOSCDController* _uposCDController = [UPOSCDController new];
UPOSCashDrawers* cdList    = [_ uposCDController getRegisteredDevice];

if(UPOS_SUCCESS == [_ uposCDController open:@"type1"])
{
        if(UPOS_SUCCESS == [_uposCDController claim:3000])
        {
                _uposCDController.deviceEnabled = YES;
                        //   Cash drawer can be used now.

        }
}
```

[Availability]
SDK 1.0.0 and later

15-3-2 claim

This method tries to open the port specified in the device information, and it includes some initialization processes such as memory allocation and initialization.

This method should be called before using the device.

[See Also]
- Constants (Defines) :: Result Code
- [Common] Device Controller Class Reference :: Available Method :: open
- [CashDrawer] Cash Drawer Controller Class Reference :: Available Method :: open
- [Common] Device Controller Class Reference :: Available Method :: claim
- [Common] Device Controller Class Reference :: Available Properties :: Claimed

[Declare]
- [Common] Device Controller Class Reference :: Available Method :: claim

[Parameters]
(NSInteger) timeout
- The system will try to execute the corresponding operation for the duration specified by this parameter.

[Return Value]
NSInteger
- UPOS_SUCCESS if the operation is successful.

[Example]

```
UPOSCDController* _uposCDController = [UPOSCDController new];
UPOSCashDrawers* cdList    = [_ uposCDController getRegisteredDevice];

if(UPOS_SUCCESS == [_ uposCDController open:@"type1"])
{
        if(UPOS_SUCCESS == [_ uposCDController claim:3000])
        {
                _ uposCDController.deviceEnabled = YES;
                        //    Cash drawer can be used
        }
}
```

[Availability]
SDK 1.0.0 and later

15-3-3 releaseDevice

This method terminates the use of the port of the claimed device and releases the physical resources. Some of the memory resources may also be released as a result.

[See Also]
- Constants (Defines) :: Result Code
- [Common] Device Controller Class Reference :: Available Method :: open
- [CashDrawer] Cash Drawer Controller Class Reference :: Available Method :: open
- [Common] Device Controller Class Reference :: Available Method :: claim
- [CashDrawer] Cash Drawer Controller Class Reference :: Available Method :: claim
- [Common] Device Controller Class Reference::Available Properties::DeviceEnabled
- [Common] Device Controller Class Reference :: Available Properties :: Claimed
- [CashDrawer]CashDrawerControllerClassReference :: AvailableMethod :: releaseDevice

[Declare]
- [Common] Device Controller Class Reference :: Available Method :: releaseDevice

[Parameters]
(NSString*) logicalDeviceName
- Model name of the device to open or stored device name
  Refer to ldn (Logical Device Name)

[Return Value]
NSInteger
- UPOS_SUCCESS if the operation is successful.

[Example]

```
UPOSCDController* _uposCDController = [UPOSCDController new];
UPOSCashDrawers* cdList    = [_ uposCDController getRegisteredDevice];

///    After using the cash drawer


//     CashDrawer Closing procedure
_ uposCDController deviceEnabled = NO;
[_ uposCDController releseDevice];
[_ uposCDController close];
```

[Availability]
SDK 1.0.0 and later

15-3-4 close

This method terminates the use of the open device.
Some of the memory resources may also be released as a result.

[See Also]
- Constants (Defines) :: Result Code
- [Common] Device Controller Class Reference :: Available Method :: open
- [CashDrawer] Cash Drawer Controller Class Reference :: Available Method :: open
- [Common] Device Controller Class Reference :: Available Method :: claim
- [CashDrawer] Cash Drawer Controller Class Reference :: Available Method :: claim
- [Common] Device Controller Class Reference::Available Properties::DeviceEnabled
- [Common] Device Controller Class Reference :: Available Properties :: Claimed
- [Common] Device Controller Class Reference :: Available Method :: close

[Declare]
- [Common] Device Controller Class Reference :: Available Method :: close

[Parameters]
(NSInteger) timeout
- The system will try to execute the corresponding operation for the duration specified by this parameter.

[Return Value]
NSInteger
- UPOS_SUCCESS if the operation is successful.

[Example]

```
UPOSCDController* _uposCDController = [UPOSCDController new];
UPOSCashDrawers* cdList    = [_ uposCDController getRegisteredDevice];

///    After using the cash drawer

//     CashDrawer Closing procedure
_ uposCDController.deviceEnabled = NO;
[_uposCDController releaseDevice];
[_uposCDController close];
```

[Availability]
SDK 1.0.0 and later

15-3-5 OpenDrawer

This method opens the cash drawer.

[See Also]

- Constants (Defines) :: Result Code
- [Common] Device Controller Class Reference :: Available Method :: open
- [CashDrawer] Cash Drawer Controller Class Reference :: Available Method :: open
- [Common] Device Controller Class Reference :: Available Method :: claim
- [CashDrawer] Cash Drawer Controller Class Reference :: Available Method :: claim
- [Common] Device Controller Class Reference::Available Properties::DeviceEnabled
- [Common] Device Controller Class Reference :: Available Properties :: Claimed

[Declare]

```
-(NSInteger) OpenDrawer;
```

[Parameters]

None

[Return Value]

NSInteger

- UPOS_SUCCESS if the operation is successful.

[Example]

```
UPOSCDController* _uposCDController = [UPOSCDController new];
UPOSCashDrawers* cdList    = [_ uposCDController getRegisteredDevice];

///    After initialization of CashDrawer (open-claim-deviceEnable)
  [_uposCDController openDrawer];

//    Cash drawer Closing procedure
_ uposCDController.deviceEnabled = NO;
[_ uposCDController releaseDevice];
[_ uposCDController close];
```

[Availability]

SDK 1.0.0 and later

15-4 Available Delegate

15-4-1 StatusUpdateEvent

This is an event that is generated when the status of CashDrawer is changed.

[See Also]
- [Common] Delegater Class Reference :: Available Delegate :: StatusUpdateEvent
- Constants (Defines) :: StatusUpdateEvent

[Declare]
- Refer to [Common] Delegater Class Reference :: Available Delegate :: StatusUpdateEvent

[Parameters]
(NSNumber*) Status
- It includes the changed status value.
  (Refer to Constants (Defines) :: StatusUpdateEvent)

[Return Value]
- void

[Example]

```
-(void) StatusUpdateEvent: (NSNumber*) Status
{
    NSLog(@"!!!!!!!!!!!!    StatusUpdateEvent : %ld !!!!!!!!!!!", (long)Status.integerValue);
    NSString *message;
    switch([ Status integerValue])
        {

        case CASH_SUE_DRAWEROPEN:
            message = [NSString stringWithFormat:@"[StatusUpdateEvent] Cash
Drawer Opened"];
            break;
        case CASH_SUE_DRAWERCLOSED:
             message = [NSString stringWithFormat:@"[StatusUpdateEvent] Cash
Drawer Closed"];
             break;
        case UPOS_SUE_POWER_OFF:
        case UPOS_SUE_POWER_OFF_OFFLINE:
        case UPOS_SUE_POWER_OFFLINE:
            message = [NSString stringWithFormat:@"[StatusUpdateEvent] Device off
or offLine"];
            break;
        case UPOS_SUE_POWER_ONLINE:
            message = [NSString stringWithFormat:@"[StatusUpdateEvent] Device
OnLine"];
            break;
        default:
            message = [NSString stringWithFormat:@"[StatusUpdateEvent]
UNKNOWN"];
        }

    NSLog(@”%@”, message)
}
```

[Availability]

SDK 1.0.0 and later

16. [MSR] MSR Class Reference

16-1 Overview

The UPOSMSR Class is the object that contains the information about target devices controlled by the UPOSMSRController.

16-2 Available Properties

- Refer to [Common] Device Class Reference :: Available Properties

16-3 Available Method

- Refer to [Common] Device Class Reference :: Available Method

17. [MSR] MSR List Class Reference (list of stored MSR)

Inherits from	UPOSDevices
Framework	libBixolonUPOS.a
[Availability]	iOS 7.0 and later
Class name	UPOSMSRs
Declared	UPOSDevices.h

17-1 Overview

The UPOSMSRs Class is the object that contains the list of MSR stored in iDevice.
(It is a set of UPOSMSR objects.).

[See Also]

- MSR Class Reference

17-2 Available Properties

- Not applicable

17-3 Available Method

17-3-1 getList

This method reads the stored device list.

[See Also]
- Device Class Reference (Common Device)
- Device Controller Class Reference (Common Controller)
- MSR Class Reference

[Declare]
- Refer to MSR List Class Reference (list of stored MSR) :: getList

[Parameters]
None

[Return Value]
- NSMutableArray*:
  The list of devices is stored as an array.
  Each object is stored as a Device Class.

[Example]

```
UPOSMSRController* _uposMSRController = [UPOSMSRController new];
UPOSMSRs* msrList = [_uposMSRController getRegisteredDevice];

NSMutableArray* msrArray = [ msrList getList];

if( msrArray == nil)
{
        NSLog(@"Failed to read the stored list of MSR.");
}
else
{
        NSLog(@"Stored list of MSR is read successfully.");
}
```

[Availability]
SDK 1.0.0 and later

17-3-2 getDeviceIdentity

This method reads the Idenitity String of the current device.

[Declare]

- Refer to Device List Class Reference (stored device list) :: getDeviceIdentity

[Parameters]

None

[Return Value]

- NSString*:
  The Identity of the current device is returned as a string.

[Example]

```
UPOSMSRController* _uposMSRController = [UPOSMSRController new];
UPOSMSRs*    msrList = [_uposMSRController getRegisteredDevice];

NSString* strIdentity = [ msrList getDeviceIdentity];

if( strIdentity == nil)
{
        NSLog(@"Failed to read the Identity value.");
}
else
{
        NSLog(@" Identity of UPOSMSRController :%@.", strIdentity);
}
```

[Availability]

SDK 1.0.0 and later

17-3-3 save

This method saves the current list of devices.

[See Also]
- MSR List Class Reference (list of stored MSR) :: removeDevice
- MSR List Class Reference (list of stored MSR) :: addDevice
- Device List Class Reference (stored device list) :: save

[Declare]
- Refer to Device List Class Reference (stored device list) :: save

[Parameters]
None

[Return Value]
- BOOL:
  YES if the operation is successful.

[Example]

```
UPOSMSRController* _uposMSRController = [UPOSMSRController new];
UPOSMSRs *    msrList = [_uposMSRController getRegisteredDevice];

UPOSMSR* newDevice = [UPOSMSR new];
newDevice.modelName          = @"SRP-350plusIII";
newDevice.ldn                = @"type1";
newDevice.interfaceType      = @"4"; // Bluetooth
newDevice.address            = @"74:F0:E1:XX:XX:XX";
newDevice.port               = @"";

[printerList addDevice:newDevice];    // Add device
[newDevice release];
if([ msrList save])
{
        NSLog(@"List is saved.");
}
else
{
        NSLog(@"List cannot be saved.");
}
```

[Availability]
SDK 1.0.0 and later

17-3-4 addDevice

This method adds a device to the current device list.
The device's addition may not be reflected to the list when the list is refreshed if the data is not saved by using the save method after using this method.

[See Also]
- MSR List Class Reference (list of stored MSR) :: save
- MSR List Class Reference (list of stored MSR) :: removeDevice
- Device List Class Reference (stored device list) :: addDevice

[Declare]
- Refer to Device List Class Reference (stored device list) :: addDevice

[Parameters]
- (UPOSDevice*) device
  Object that contains the information about the device to add

[Return Value]
- BOOL:
  YES if the operation is successful.

[Example]

```
UPOSMSRController* _uposMSRController = [UPOSMSRController new];
UPOSMSRs *    msrList = [_uposMSRController getRegisteredDevice];

UPOSMSR* newDevice = [UPOSMSR new];
newDevice.modelName          = @"SRP-350plusIII";
newDevice.ldn                = @"type1";
newDevice.interfaceType      = @"4"; // Bluetooth
newDevice.address            = @"74:F0:E1:XX:XX:XX";
newDevice.port               = @"";
[printerList addDevice:newDevice];    // Add device
[newDevice release];
if([ msrList addDevice:newDevice])     // Add device
{
        NSLog(@"Device is added successfully.");
}
else
{
        NSLog(@""Failed to add device.");
}

[newDevice release];
[msrList save];
```

[Availability]
SDK 1.0.0 and later

17-3-5 removeDevice

This method deletes a device from the device list.
The device's deletion may not be reflected to the device list when the list is refreshed if the data is not saved by using the save method after using this method.

[See Also]
- MSR List Class Reference (list of stored MSR) :: save
- MSR List Class Reference (list of stored MSR) :: addDevice
- Device List Class Reference (stored device list) :: removeDevice

[Declare]
- Refer to Device List Class Reference (stored device list) :: removeDevice

[Parameters]
- (UPOSDevice*) device
  Object that contains the information about the device to add

[Return Value]
- BOOL:
  YES if the operation is successful.

[Example]

```
UPOSMSRController* _uposMSRController = [UPOSMSRController new];
UPOSMSRs *    msrList = [_uposMSRController getRegisteredDevice];

UPOSMSR * willRemoveDevice = [[printerList getList] objectAtIndex:0];
if([ msrList removeDevice:willRemoveDevice])     // Remove device
{
        NSLog(@"0th device is removed from the list.");
}
else
{
        NSLog(@"Failed to remove 0th device from the list.");
}

[newDevice release];
[ msrList save];
```

[Availability]
SDK 1.0.0 and later

18. [MSR] MSR Controller Class Reference

Inherits from	NSObject
Framework	libBixolonUPOS.a
[Availability]	iOS 7.0 and later
Class name	UPOSMSRController
Declared	UPOSMSRController.h

18-1 Overview

The UPOSDeviceController Class is the main object that controls the common functions of the devices supported by this SDK.

18-2 Available Properties

18-2-1 Track1Data

This contains the most recently obtained Track 1 data of the MSR card.

[Declare]

```
@property (strong, readonly)        NSString*                    Track1Data;
```

[Availability]
SDK 1.0.0 and later

18-2-2 Track2Data

This contains the most recently obtained Track 2 data of the MSR card.

[Declare]

```
@property (strong, readonly)        NSString*                    Track2Data;
```

[Availability]
SDK 1.0.0 and later

18-2-3 Track3Data

This contains the most recently obtained Track 3 data of the MSR card.

[Declare]

```
@property (strong, readonly)        NSString*                    Track3Data;
```

[Availability]
SDK 1.0.0 and later

18-3 Available Method

18-3-1 open

This method initiates the use of the MSR class, and it includes the initialization process such as memory allocation. This method should be called first before calling the claim and other subsequent methods.

[See Also]

- Constants (Defines) :: OpenResult Code
- [Common] Device Controller Class Reference :: Available Properties :: OpenResult
- [Common] Device Controller Class Reference :: Available Properties :: ldn (Logical Device Name)
- [MSR] MSR Controller Class Reference :: Available Method :: addDevice

[Declare]

- [Common] Device Controller Class Reference :: Available Method :: open

[Parameters]

(NSString*) logicalDeviceName

- Model name of the device to open or stored device name
  Refer to ldn (Logical Device Name)
- The device should be added by using the [MSR] MSR Controller Class Reference :: Available Method :: addDevice method in order to use it.

[Return Value]

NSInteger

- UPOS_SUCCESS if the operation is successful.

[Example]

```
UPOMSRController* _uposMSRController    = [UPOSMSRController new];
UPOSMSRs* msrList    = [_uposMSRController getRegisteredDevice];

if(UPOS_SUCCESS == _uposMSRController    open:@"type1"])
{
        if(UPOS_SUCCESS == [_uposMSRController claim:3000])
        {
                _uposMSRController.deviceEnabled = YES;
                        //    MSR can be used now.

        }
}
```

[Availability]

SDK 1.0.0 and later

18-3-2 claim

This method tries to open the port specified in the device information, and it includes some initialization processes such as memory allocation and initialization.

This method should be called before using the device.

[See Also]
- Constants (Defines) :: Result Code
- [Common] Device Controller Class Reference :: Available Method :: open
- [MSR] MSR Controller Class Reference :: Available Method :: open
- [Common] Device Controller Class Reference :: Available Method :: claim
- [Common] Device Controller Class Reference :: Available Properties :: Claimed

[Declare]
- [Common] Device Controller Class Reference :: Available Method :: claim

[Parameters]
(NSInteger) timeout
- The system will try to execute the corresponding operation for the duration specified by this parameter.

[Return Value]
NSInteger
- UPOS_SUCCESS if the operation is successful.

[Example]

```
UPOMSRController* _uposMSRController    = [UPOSMSRController new];
UPOSMSRs* msrList    = [_ uposMSRController getRegisteredDevice];

if(UPOS_SUCCESS == [_uposMSRController    open:@"type1"])
{
        if(UPOS_SUCCESS == [_uposMSRController claim:3000])
        {
                _uposMSRController.deviceEnabled = YES;
                        //    MSR can be used now
        }
}
```

[Availability]
SDK 1.0.0 and later

18-3-3 releaseDevice

This method terminates the use of the port of the claimed device and releases the physical resources. Some of the memory resources may also be released as a result.

[See Also]
- Constants (Defines) :: Result Code
- [Common] Device Controller Class Reference :: Available Method :: open
- [MSR] MSR Controller Class Reference :: Available Method :: open
- [Common] Device Controller Class Reference :: Available Method :: claim
- [MSR] MSR Controller Class Reference :: Available Method :: claim
- [Common] Device Controller Class Reference::Available Properties::DeviceEnabled
- [Common] Device Controller Class Reference :: Available Properties :: Claimed
- [MSR] MSR Controller Class Reference :: AvailableMethod :: releaseDevice

[Declare]
- [Common] Device Controller Class Reference :: Available Method :: releaseDevice

[Parameters]
(NSString*) logicalDeviceName
- Model name of the device to open or stored device name
  Refer to ldn (Logical Device Name)

[Return Value]
NSInteger
- UPOS_SUCCESS if the operation is successful.

[Example]

```
UPOMSRController* _uposMSRController    = [UPOSMSRController new];
UPOSMSRs* msrList    = [_ uposMSRController getRegisteredDevice];

///    After using MSR


//     MSR Closing procedure
_uposMSRController deviceEnabled = NO;
[_uposMSRController releseDevice];
[_uposMSRController close];
```

[Availability]
SDK 1.0.0 and later

18-3-4 close

This method terminates the use of the open device.
Some of the memory resources may also be released as a result.

[See Also]
- Constants (Defines) :: Result Code
- [Common] Device Controller Class Reference :: Available Method :: open
- [MSR] MSR Controller Class Reference :: Available Method :: open
- [Common] Device Controller Class Reference :: Available Method :: claim
- [MSR] MSR Controller Class Reference :: Available Method :: claim
- [Common] Device Controller Class Reference::Available Properties::DeviceEnabled
- [Common] Device Controller Class Reference :: Available Properties :: Claimed
- [Common] Device Controller Class Reference :: Available Method :: close

[Declare]
- [Common] Device Controller Class Reference :: Available Method :: close

[Parameters]
(NSInteger) timeout
- The system will try to execute the corresponding operation for the duration specified by this parameter.

[Return Value]
NSInteger
- UPOS_SUCCESS if the operation is successful.

[Example]

```
UPOMSRController* _uposMSRController    = [UPOSMSRController new];
UPOSMSRs* msrList    = [_ uposMSRController getRegisteredDevice];

///    After using MSR


//     MSR Closing procedure
_uposMSRController deviceEnabled = NO;
[_uposMSRController releaseDevice];
[_uposMSRController close];
```

[Availability]
SDK 1.0.0 and later

18-4 Available Delegate

18-4-1 DataEvent

This is the delegate that is passed when MSR data is generated by scanning a card with the MSR reader.

[See Also]
- [Common] Delegater Class Reference :: Available Delegate :: DataEvent

[Declare]
- Refer to [Common] Delegater Class Reference :: Available Delegate :: DataEvent

[Parameters]
(NSNumber*) Status
- Track information stored in the recognized MSR data is passed.

[Return Value]
- void

[Example]

```
- (void) DataEvent:(NSNumber*) Status
{
        NSLog(@"!!!!!!!!!!!!  (MSR) Data Event : %ld !!!!!!!!!!!", (long) Status integerValue );

    [self updateStringEventHistory:[NSString stringWithFormat:@"============= Data Event =============="]];
    [self updateStringEventHistory:[NSString stringWithFormat:@"track 1: %@", _uposMSRController.Track1Data]];
    [self updateStringEventHistory:[NSString stringWithFormat:@"track 2: %@", _uposMSRController.Track2Data]];
    [self updateStringEventHistory:[NSString stringWithFormat:@"track 3: %@\r\n\r\n\r\n", _uposMSRController.Track3Data]];
}
```

[Availability]
SDK 1.0.0 and later

19. [SCR] SCR Class Reference

Inherits from	NSObject
Framework	libBixolonUPOS.a
[Availability]	iOS 7.0 and later
Class name	UPOSScr
Declared	UPOSDevices.h

19-1 Overview

UPOSSCR Class is an object that contains the common information about all target control devices controlled by the controller of each device.

[See Also]

- [Common] Device Class Reference

19-2 Available Properties

- [Common] Device Class Reference :: Available Properties Reference

19-3 Available Method

- [Common] Device Class Reference :: Available Method Reference

20. [SCR] SCR List Class Reference(stored scr list)

Inherits from	UPOSDevices
Framework	libBixolonUPOS.a
[Availability]	iOS 7.0 and later
Class name	UPOSSCRs
Declared	UPOSDevices.h

20-1 Overview

UPOSSCRs Class is an object that contains the device list in the iDevice.

[See Also]

- SCR Controller Class Reference

20-2 Available Properties

- Not applicable

20-3 Available Method

20-3-1 getList

This method reads the stored device list.

[See Also]
- Device Class Reference (Common Device)
- Device Controller Class Reference (Common Controller)
- SCR Class Reference

[Declare]
- Refer to Cash Drawer List Class Reference (stored list of cash drawers) :: getList

[Parameters]
None..

[Return Value]
- NSMutableArray*:
  This list of devices is stored as an array.
  Each object is stored as a Device Class.

[Example]

```
UPOSSCRController* _uposSCRController = [UPOSSCRController new];
UPOSSCRs* scrList = [_uposSCRController getRegisteredDevice];

NSMutableArray* scrArray = [ msrList getList];

if( scrArray == nil)
{
        NSLog(@“Failed to read the listof SCR.”);
}
else
{
        NSLog(@”list of SCR is read.”);
}
```

[Availablility]
SDK 1.0.4 and later

20-3-2 getDeviceIdentity

This method reads the Identity string of the current device.

[Declare]

- Refer to Device List Class Reference (stored device list) :: getDeviceIdentity

[Parameters]

없음.

[Return Value]

- NSString*:
  The Identity of the current device is returned as a string.

[Example]

```
UPOSSCRController* _uposSCRController = [UPOSSCRController new];
UPOSSCRs*    msrList = [_uposSCRController getRegisteredDevice];

NSString* strIdentity = [ msrList getDeviceIdentity];

if( strIdentity == nil)
{
        NSLog(@"Failed to read the identity value.");
}
else
{
        NSLog(@" identity of UPOSSCRController:%@.", strIdentity);
}
```

[Availablility]

SDK 1.0.0 and later

20-3-3 save

This method saves the current list of devices.

[See Also]
- SCR List Class Reference (stored SCR list) :: Available Method :: addDevice
- SCR List Class Reference (stored SCR list) :: Avilable Method :: removeDevice
- Device List Class Reference (stored device list) :: Available Method :: save

[Declare]
- Refer to Device List Class Reference (stored device list) :: Available Method :: save

[Parameters]
None.

[Return Value]
- BOOL:
  YES if the operation is successful.

[Example]

```
UPOSSCRController* _uposSCRController = [UPOSSCRController new];
UPOSSCRs *    scrList = [_uposSCRController getRegisteredDevice];

UPOSSCR* newDevice = [UPOSSCR new];
newDevice.modelName          = @"SPP-R210S";
newDevice.ldn                        = @"type1";
newDevice.interfaceType      = @"4"; // Bluetooth
newDevice.address            = @"74:F0:E1:XX:XX:XX";
newDevice.port                       = @"";

[scrList addDevice:newDevice];    // Add device
[newDevice release];
if([ scrList save])
{
        NSLog(@"List is saved.");
}
else
{
        NSLog(@"Fail to List.");
}
```

[Availablility]
SDK 1.0.4 and later

20-3-4 addDevice

This method adds a device to the current device list.
The device's addition may not be reflected to the device list when the list is refreshed if the data is not saved by using the save method after using this method.

[[See Also]

- Device List Class Reference (stored device list) :: Available Method :: save
- SCR List Class Reference (stored scr list) :: Available Method :: removeDevice
- Device List Class Reference (stored device list) :: Available Method :: addDevice

[Declare]

- Refer to Device List Class Reference (stored device list) :: Available Method :: addDevice.

[Parameters]

- (UPOSDevice*) device
  Object that contains the information about the device to add

[Return Value]

- BOOL:
  YES if the operation is successful.

[Example]

```
UPOSSCRController* _uposSCRController = [UPOSSCRController new];
UPOSSCRs *    scrList = [_uposMSRController getRegisteredDevice];

UPOSSCR* newDevice = [UPOSSCR new];
newDevice.modelName          = @"SPP-R210";
newDevice.ldn                          = @"type1";
newDevice.interfaceType      = @"4"; // Bluetooth
newDevice.address            = @"74:F0:E1:XX:XX:XX";
newDevice.port                         = @"";
[printerList addDevice:newDevice];    // Add printer
[newDevice release];
if([ scrList addDevice:newDevice])     // add SCR
{
        NSLog(@"Device is added successfully.");
}
else
{
        NSLog(@"Failed to add device.");
}

[newDevice release];
[scrList save];
```

[Availablility]

SDK 1.0.4 and later

20-3-5 removeDevice

This method deletes a device from the device list.
The device's deletion may not be reflected to the device list when the list is refreshed if the data is not saved by using the save method after using this method

[See Also]

- SCR List Class Reference (stored scr list) :: Available Method :: save
- SCR List Class Reference (stored scr list) :: Available Method :: addDevice
- Device List Class Reference (stored device list) :: Available Method :: removeDevice

[Declare]

- Refer to Device List Class Reference (stored device list) :: Available Method :: removeDevice.

[Parameters]

- (UPOSDevice*) device
  Object that contains the information about the device to add

[Return Value]

- BOOL:
  YES if the operation is successful.

[Example]

```
UPOSSCRController* _uposSCRController = [UPOSSCRController new];
UPOSSCRs * scrList = [_uposSCRController getRegisteredDevice];

UPOSSCR * willRemoveDevice = [[scrList getList] objectAtIndex:0];
if([scrList removeDevice:willRemoveDevice])     //remove from scrList
{
        NSLog(@"0th Device is removed from the list.");
}
else
{
        NSLog(@"Failed to remove 0th device from the list.");
}

[newDevice release];
[scrList save];
```

[Availablility]

SDK 1.0.4 and later

21. [SCR] SCR Controller Class Reference

Inherits from	NSObject
Framework	libBixolonUPOS.a
[Availability]	iOS 7.0 and later
Class name	UPOSSCRController
Declared	UPOSSCRController.h

21-1 Overview

The UPOSSCRController Class is the main object that controls the common functions of the devices supported by this SDK.

21-2 Available Properties

- Not applicable

21-3 Available Method

21-3-1 open

This method initiates the use of the msr class, and it includes the initialization process such as memory allocation. This method should be called first before calling the claim and other subsequent methods.

[See Also]
- Constants (Defines) :: OpenResult Code
- [Common] Device Controller Class Reference :: Available Properties :: OpenResult
- [Common] Device Controller Class Reference :: Available Properties :: ldn (Logical Device Name)
- SCR List Class Reference (stored scr list) :: Available Method :: addDevice

[Declare]
- [Common] Device Controller Class Reference :: Available Method :: open

[Parameters]
(NSString*) logicalDeviceName
- Model name of the device to open or stored device name
  Refer to ldn (Logical Device Name)
- The device should be added by using the [SCR] SCR Controller Class Reference :: Available Method :: addDevice method in order to use it.

[Return Value]
NSInteger
- UPOS_SUCCESS if the operation is successful.

[Example]

```
UPOSCRController* _uposSCRController    = [UPOSSCRController new];
UPOSSCRs* msrList    = [_uposSCRController getRegisteredDevice];

if(UPOS_SUCCESS == _uposSCRController    open:@"type1"])
{
        if(UPOS_SUCCESS == [_uposSCRController claim:3000])
        {
                _uposSCRController.deviceEnabled = YES;
                        //    SCR can be used now.

        }
}
```

[Availablility]
SDK 1.0.4 and later

21-3-2 claim

This method tries to open the port specified in the device information, and it includes some initialization processes such as memory allocation and initialization.

This method should be called before using the device.

[See Also]
- Constants (Defines) :: Result Code
- [Common] Device Controller Class Reference :: Available Method :: open
- [SCR] SCR Controller Class Reference :: Available Method :: open
- [Common] Device Controller Class Reference :: Available Method :: claim
- [Common] Device Controller Class Reference :: Available Properties :: Claimed

[Declare]
- [Common] Device Controller Class Reference :: Available Method :: claim

[Parameters]
(NSInteger) timeout
- The system will try to execute the corresponding operation for the duration specified by this parameter.

[Return Value]
NSInteger
- UPOS_SUCCESS if the operation is successful.

[Example]

```
UPOSCRController* _uposSCRController    = [UPOSSCRController new];
UPOSSCRs* scrList    = [_ uposSCRController getRegisteredDevice];

if(UPOS_SUCCESS == [_uposSCRController    open:@"type1"])
{
        if(UPOS_SUCCESS == [_uposSCRController claim:3000])
        {
                _uposSCRController.deviceEnabled = YES;
                        //    SCR can be used now.
        }
}
```

[Availablility]
SDK 1.0.4 and later

21-3-3 releaseDevice

This method terminates the use of the port of the claimed device and releases the physical resources. Some of the memory resources may also be released as a result.

[See Also]
- Constants (Defines) :: Result Code
- [Common] Device Controller Class Reference :: Available Method :: open
- [SCR] SCR Controller Class Reference :: Available Method :: open
- [Common] Device Controller Class Reference :: Available Method :: claim
- [SCR] SCR Controller Class Reference :: Available Method :: claim
- [Common] Device Controller Class Reference::Available Properties::DeviceEnabled
- [Common] Device Controller Class Reference :: Available Properties :: Claimed

[Declare]
- [Common] Device Controller Class Reference :: Available Method :: releaseDevice

[Parameters]
(NSString*) logicalDeviceName
- Model name of the device to open or stored device name
  Refer to ldn (Logical Device Name)

[Return Value]
NSInteger
- UPOS_SUCCESS if the operation is successful.

[Example]

```
UPOSCRController* _uposSCRController    = [UPOSSCRController new];
UPOSSCRs* msrList    = [_ uposSCRController getRegisteredDevice];

///    After using SCR


//     SCR Closing procedure
_uposSCRController deviceEnabled = NO;
[_uposSCRController releseDevice];
[_uposSCRController close];
```

[Availablility]
SDK 1.0.4 and later

21-3-4 close

This method terminates the use of the open device.
Some of the memory resources may also be released as a result.

[See Also]

- Constants (Defines) :: Result Code
- [Common] Device Controller Class Reference :: Available Method :: open
- [SCR] SCR Controller Class Reference :: Available Method :: open
- [Common] Device Controller Class Reference :: Available Method :: claim
- [SCR] SCR Controller Class Reference :: Available Method :: claim
- [Common] Device Controller Class Reference::Available Properties::DeviceEnabled
- [Common] Device Controller Class Reference :: Available Properties :: Claimed
- [Common] Device Controller Class Reference :: Available Method :: close

[Declare]

- [Common] Device Controller Class Reference :: Available Method :: close

[Return Value]

NSInteger

- UPOS_SUCCESS if the operation is successful.

[Example]

```
UPOSSCRController* _uposSCRController    = [UPOSSCRController new];
UPOSSCRs* msrList    = [_ uposSCRController getRegisteredDevice];

///    After using SCR


//     SCR Closing procedure
_uposSCRController deviceEnabled = NO;
[_uposSCRController releaseDevice];
[_uposSCRController close];
```

[Availablility]

SDK 1.0.4 and later

21-3-5 beginInsertion

This method waits to card insertion.

[See Also]
- constants (Defines) :: Result Code

[Declare]

```
-(NSInteger) beginInsertion : (NSInteger) timeout;
```

[Parameters]
(NSInteger) timeout
- The system will wait to insertion of card for the duration specified by this parameter.

[Return Value]
NSInteger
- UPOS_SUCCESS if the operation is successful.

[Example]

```
UPOSSCRController* _uposSCRController    = [UPOSSCRController new];
UPOSSCRs* scrList    = [_ uposSCRController getRegisteredDevice];

///    After SCR initialization (open-claim-deviceEnable)
NSInteger result = [_uposSCRController beginInsertion:3000];    //    waiting for card
insertion during 3000ms(about 3seconds)
NSLog(@“result : %ld”, (long)result);
//      SCR Closing procedure
_uposSCRController deviceEnabled = NO;
[_uposSCRController releseDevice];
[_uposSCRController close];

//    delegate
-(void) StatusUpdateEvent:(NSNumber*) Status{
NSLog(@"!!!!!!!!!!!!    StatusUpdateEvent : %ld !!!!!!!!!!!", (long)Status.integerValue);
      NSString *message;
      switch([Status integerValue])
      {
            case UPOS_SUE_POWER_OFF:
            case UPOS_SUE_POWER_OFF_OFFLINE:
            case UPOS_SUE_POWER_OFFLINE:
                  break;
            case UPOS_SUE_POWER_ONLINE:
                  break;
            case SC_SUE_CARD_PRESENT:
                  NSLog(@ “Card Insterted”);
}
```

[Availablility]
SDK 1.0.4 and later

21-3-6 endInsertion

This method terminates the card insertion.

[See Also]
- Constants (Defines) :: Result Code

[Declare]

```
-(NSInteger) endInsertions;
```

[Return Value]
NSInteger
- UPOS_SUCCESS if the operation is successful.

[Example]

```
UPOSSCRController* _uposSCRController    = [UPOSSCRController new];
UPOSSCRs* scrList    = [_ uposSCRController getRegisteredDevice];

///    After SCR initialization (open-claim-deviceEnable)
NSInteger result = [_uposSCRController beginInsertion:3000];    //    waiting for card
insertion during 3000ms(about 3seconds)
NSLog(@“result : %ld”, (long)result);
//      SCR Closing procedure
_uposSCRController deviceEnabled = NO;
[_uposSCRController releseDevice];
[_uposSCRController close];

//    delegate
-(void) StatusUpdateEvent:(NSNumber*) Status{
NSLog(@"!!!!!!!!!!!!    StatusUpdateEvent : %ld !!!!!!!!!!!", (long)Status.integerValue);
      NSString *message;
      switch([Status integerValue])
      {
            case UPOS_SUE_POWER_OFF:
            case UPOS_SUE_POWER_OFF_OFFLINE:
            case UPOS_SUE_POWER_OFFLINE:
                  break;
            case UPOS_SUE_POWER_ONLINE:
                  break;
            case SC_SUE_CARD_PRESENT:
                  NSLog(@ “Card Insterted”);
                  [_uposSCRController endInsertion];
}
```

[Availablility]
SDK 1.0.4 and later

21-4 Available Delegate

21-4-1 DataEvent

This is the delegate that is passed when MSR data is generated by scanning a card with the SCR reader.

[See Also]
- [Common] Delegater Class Reference :: Available Delegate :: DataEvent

[Declare]
- Refer to [Common] Delegater Class Reference :: Available Delegate :: DataEvent

[Parameters]
(NSNumber*) Status
- Fixed to 'SC_READ_DATA'.

[Return Value]
- void

[Example]

```
- (void) DataEvent:(NSNumber*) Status
{
        NSLog(@"!!!!!!!!!!!!  (SCR) Data Event : %ld !!!!!!!!!!!", (long) Status integerValue );

        NSData* data = nil;
        [_uposSCRController readData:SC_READ_DATA data:&data];
        NSLog(@"read SCR Data : %@", data);

}
```

[Availablility]
SDK 1.0.0 and later

22. [Common] Delegater Class Reference

Inherits from	NSObject
Framework	libBixolonUPOS.a
[Availability]	iOS 7.0 and later
Class name	UPOSDeviceControlDelegate
Declared	UPOSDeviceControlDelegate.h

22-1 Overview

The UPOSDeviceControlDelegate Class is the main object to pass events in this SDK.

22-2 Available Properties

- Not applicable

22-3 Available Delegate

22-3-1 DataEvent

This is the delegate that is passed when data is generated from each device.

[Declare]

```
-(void)DataEvent:(NSNumber*) Status;
```

[Availability]
SDK 1.0.0 and later

22-3-2 StatusUpdateEvent

It is an event that is generated when the status of CashDrawer is changed.

[Declare]

```
-(void)StatusUpdateEvent:(NSNumber*) Status;
```

[Availability]
SDK 1.0.0 and later

22-3-3 OutputCompleteEvent

This event is generated when each device completes the received output request.

[Declare]

```
-(void)OutputCompleteEvent:(NSNumber*)OutputID;
```

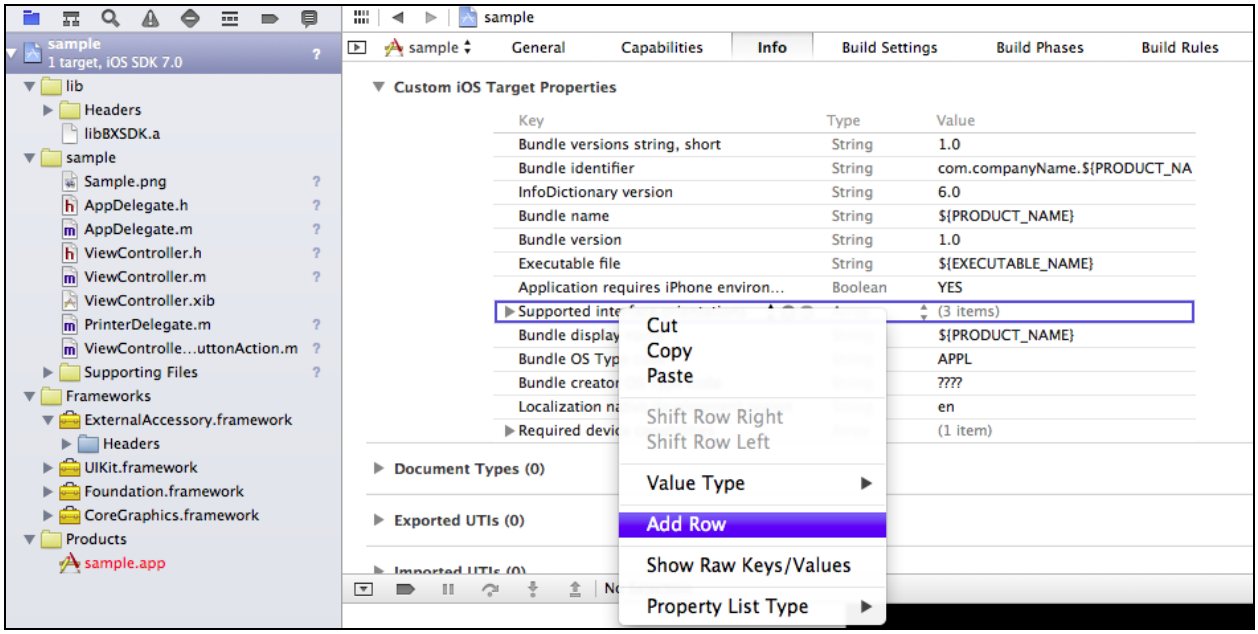
[Availability]
SDK 1.0.0 and later

23. Sample Program

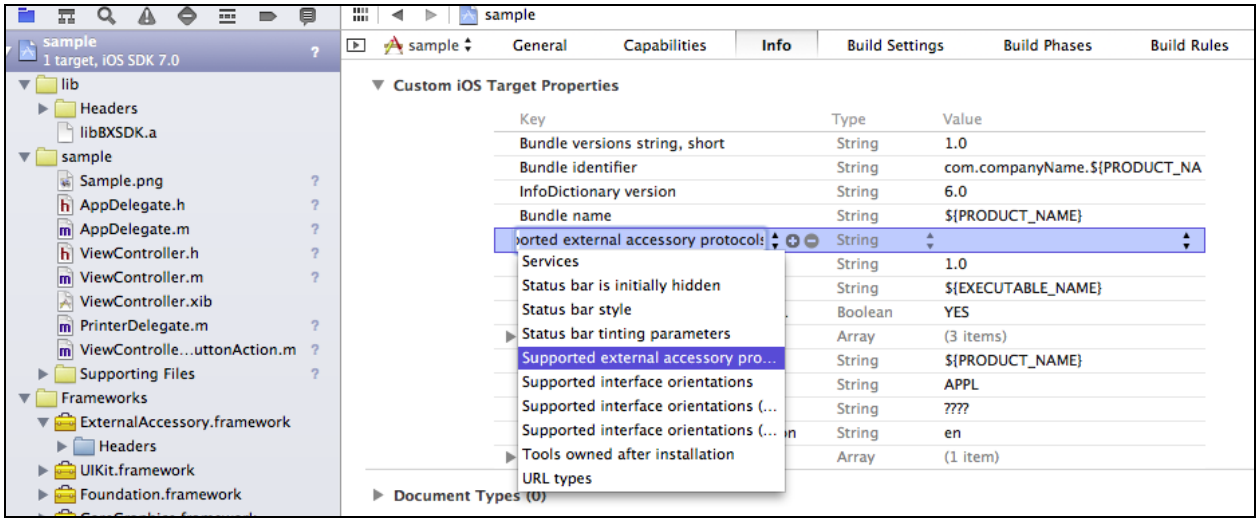
23-1 Setting Project

[Note] Registration with the Apple Developer Program is required to develop iOS applications. Refer to the Apple Developer's Website (http://developer.apple.com/devcenter/ios) for details.

23-1-1 Adding ExternalAccessory.framework

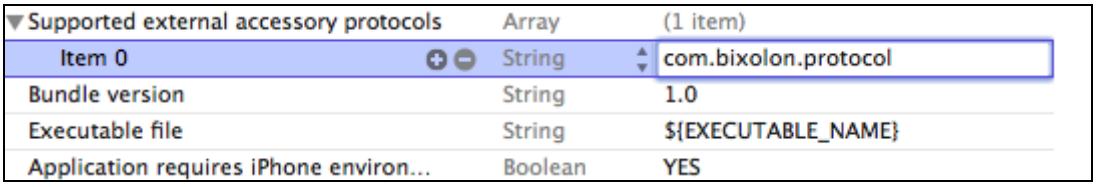


1. Select project file
2. Select [Info] tab
3. Ctrl-click in the Area 3
4. Select [Add Row] from the pop up menu



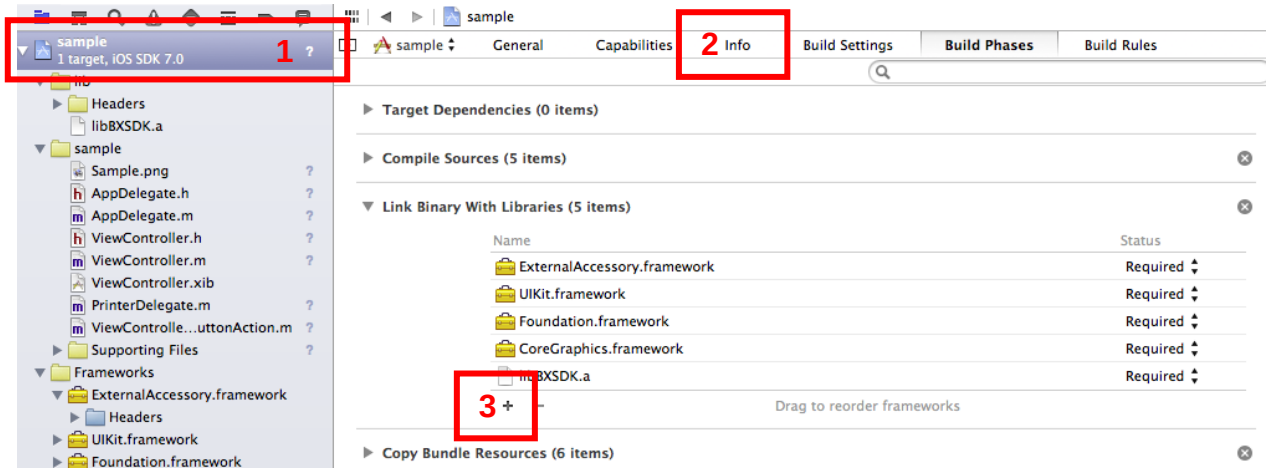
5. Enter "Supported external accessory protocols".

6. Enter “com.bixolon.protocol” in the Items field.

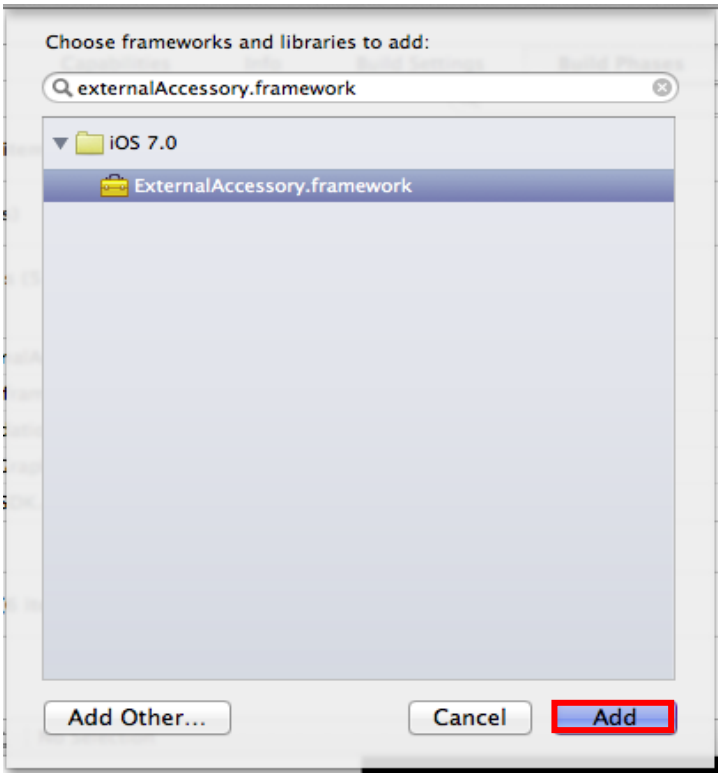


23-1-2 Adding Bluetooth Protocol

1. Select Project, Info, and + in the order shown below.



2. Search “ExternalAccessory.framework” and click the [Add] to add it.



23-2 nativeSample

It is an example of an iOS application program using the BXL SDK for iOS_UPOS Compliant.

23-3 phonegapSample

It is an example of an iOS web application program using the BXL SDK for iOS_UPOS Compliant and it was built with PhoneGap version 3.1.0-0.15.0.

23-3-1 Setting Environment

- Install Node.js (refer to http://nodejs.org)
- Install PhoneGap (refer to http://docs.phonegap.com/en/edge/index.html)
- Add plugins and build the software (refer to http://docs.phonegap.com/en/edge/guide_cli_index.md.html#The%20Command-Line%20Interface)

23-3-2 Settings to use BXL SDK for iOS_UPOS Compliant for applications using PhoneGap

- Add [CordovaLib/Classes/UPOSService.h], [Classes/UPOSService.m].
- Add [plugins/com.bixolon.upos.service/www/upos_service.js]
- Add the following codes in [www/cordova_plugin.js].

```
cordova.define('cordova/plugin_list', function(require, exports, module) {
module.exports = [
        {
          "file": "plugins/com.bixolon.upos.service/www/upos_service.js",
           "id": "com.bixolon.upos.service.upos_service",
           "clobbers": [
                "upos_service"
                ]
        }
]
});
```

- Add the following codes in [config.xml].

```
<feature name="UPOSService">
        <param name="ios-package"    value="UPOSService" />
</feature>
```

24. Error Information

This section is to explain returned error information when use Printer methods.
For more details, please refer to the UPOS specifications.

24-1 Error list

- claim

Method	Errors
claim	UPOS_E_ILLEGAL
	UPOS_E_TIMEOUT

- checkHealth

Method	Errors
checkHealth	UPOS_E_ILLEGAL

- printNormal

Method	Errors
printNormal	UPOS_E_ILLEGAL
	UPOS_E_BUSY

- printImmediate

Method	Errors
printImmediate	UPOS_E_ILLEGAL

- cutPaper

Method	Errors
cutPaper	UPOS_E_BUSY
	UPOS_E_ILLEGAL

- printBitmap

Method	Errors
printBitmap	UPOS_E_BUSY
	UPOS_E_ILLEGAL
	UPOS_E_NOEXIST

- transactionPrint

Method	Errors
transactionPrint	UPOS_E_BUSY
	UPOS_E_ILLEGAL