CITIZEN

iOS Label Print SDK (Swift)

Programming Manual

Version 1.01

CITIZEN SYSTEMS JAPAN CO., LTD.

Revision History

Date	Version	Description
Aug 28, 2018	1.00	First issue.
Oct 10, 2018	1.01	- Modified the printer model name. (Page 6,28,30) - Added the framework for the Swift4.2 to SDK type. (Page 7) - Added a locale for Chinese model and Korean model to drawTextPtrFont. (Page 59) - Added the Chinese model and the Korean model to the definition of the locale. (Page 91)

Permission Notice

1. Unauthorized use of all or any part of this document is prohibited.
2. The information in this document is subject to change without prior notice.
3. This document has been created with full attention. If, however, you find an error or question, please contact us.
4. We shall not be liable for any effect resulting from operation regardless of the above item 3.
5. If you do not agree with the above terms, you are not permitted to use this SDK.

Trademark

Table of Contents

1. Introduction

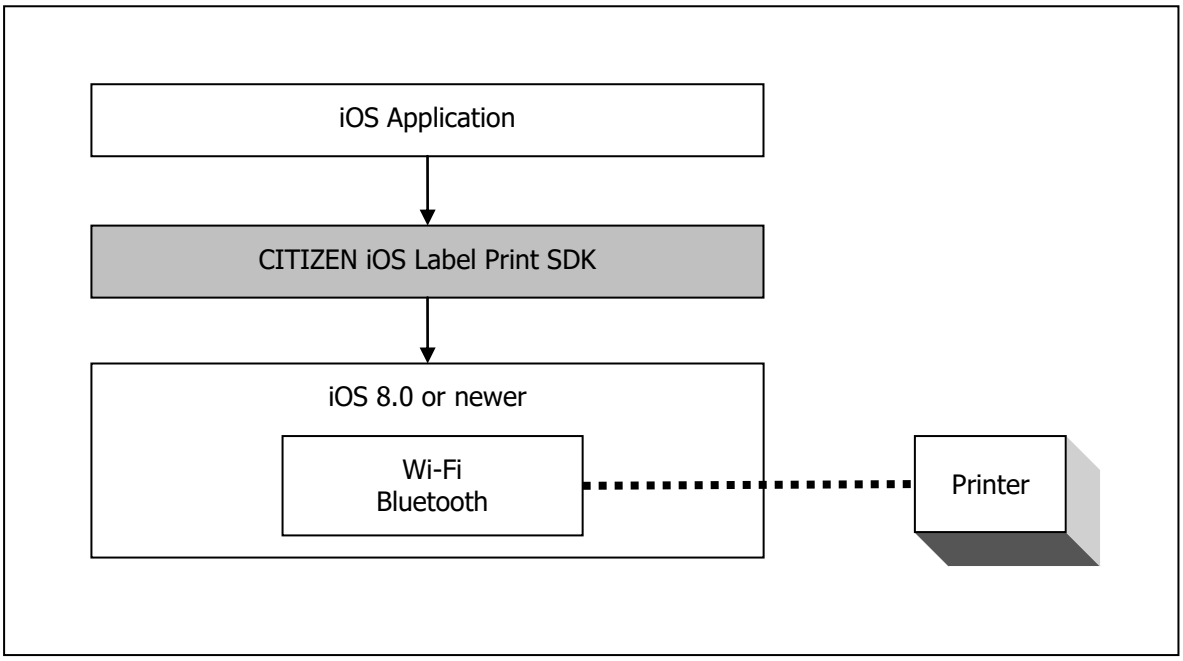
This document is a CITIZEN iOS Label Print SDK Programing Manual.

1.1. Who should read this document

This document is intended to be iOS application (Swift) developers to take advantage of CITIZEN Label printers.

1.2. System summary

This SDK assumes that it is referred from iOS application using CITIZEN Label printers.



System diagram of the SDK

Supported terminals

The specification of terminals supported by this SDK are as follows.
OS: iOS 8.0 or newer
Interface: Wi-Fi, Bluetooth
Development environment: Xcode 9.0 or newer

Supported printer models

Object Model	Interface	Print Method
CL-S700/703	Wired/Wireless LAN,	Direct thermal/Thermal transfer
CL-E720/730	Wired/Wireless LAN, Bluetooth	Direct thermal/Thermal transfer
CL-S621/631	Wired/Wireless LAN	Direct thermal/Thermal tranfer
CL-S521/531	Wired/Wireless LAN	Direct thermal
CL-S400DT	Wired/Wireless LAN, Bluetooth	Direct thermal
CL-E300/303	Wired LAN	Direct thermal
CL-E321/331	Wired LAN	Direct thermal/Thermal tranfer

Refer to each user's manual for more details.

1.3. Definition method

SDK Type

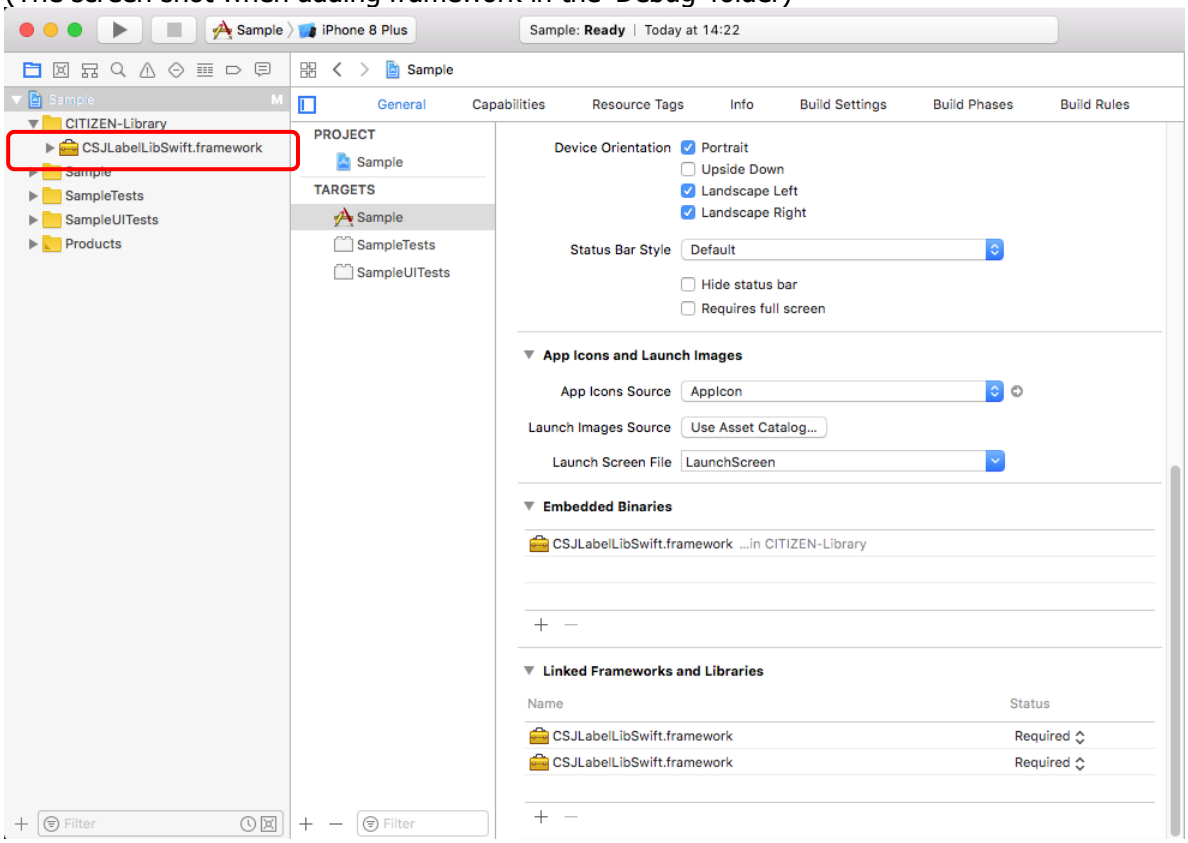
Select according to the development environment of the SDK (Framework).

SDK (Framework)	Development environment	Target OS
Swift 4.0	Xcode 9.0	iOS 11 or earlier
Swift 4.0.2	Xcode 9.1	iOS 11.1 or earlier
Swift 4.1	Xcode 9.3	iOS 11.3 or earlier
Swift 4.2	Xcode 10.0	iOS 12 or earlier

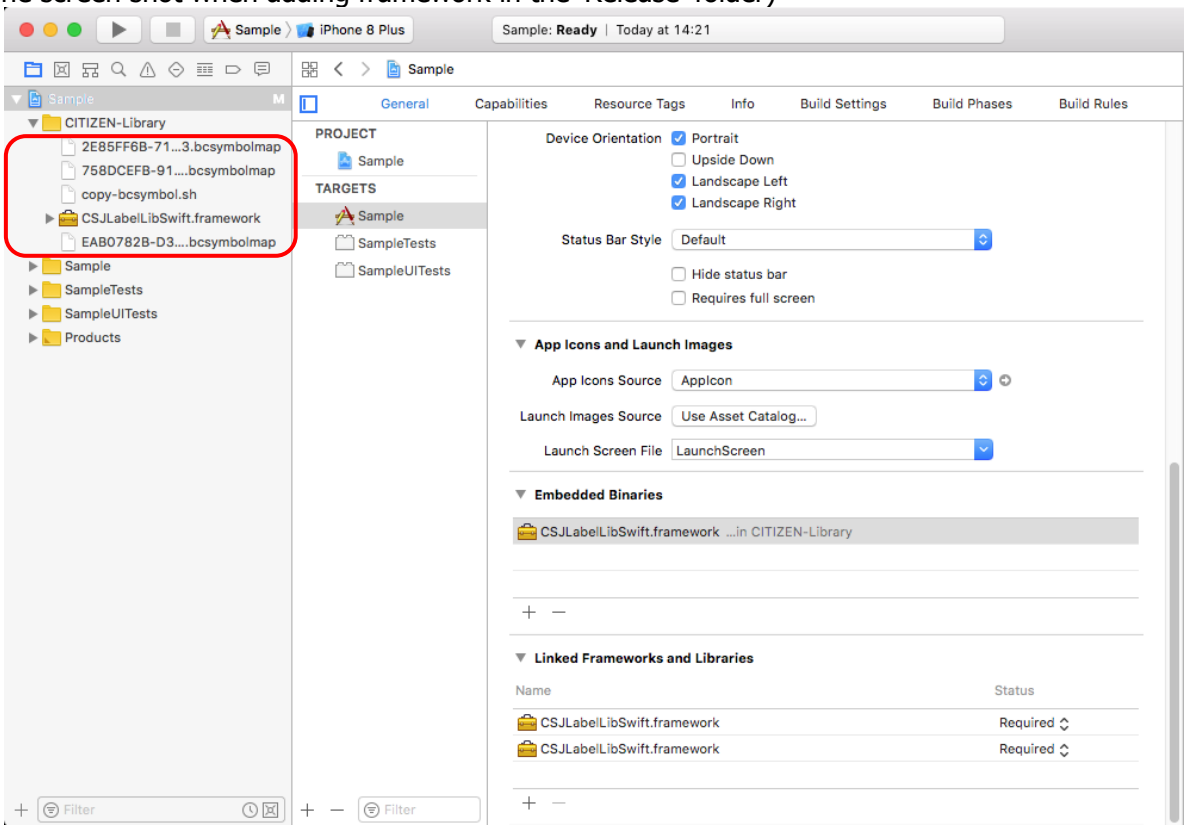
Add the SDK to Xcode

Add the framework file (CSJLabelLibSwift.framework) to the project to develop.
Framework files are stored is divided into 'Debug' folder and 'Release' folder. Since the framework file in the 'Debug' folder contains the binary for the iPhone-simulator, please use this at the time of application development. The framework file in the 'Release' folder does not contain the binary for the simulator. Please use this at the time of archive file making to submit to iTunes Connect.
Drag the framework file from the 'Finder' to any place in the "Project Navigator" of Xcode.(because it is contained in the folder named 'CITIZEN-Library', drag this folder every.) In the 'Release' folder, three bcsymbolmap files and one script files (copy-bcsymbol.sh) other than 'CSJLabelLibSwift.framework' are contained. Please copy these files to the same place as 'CSJLabelLibSwift.framework from the 'Finder'.

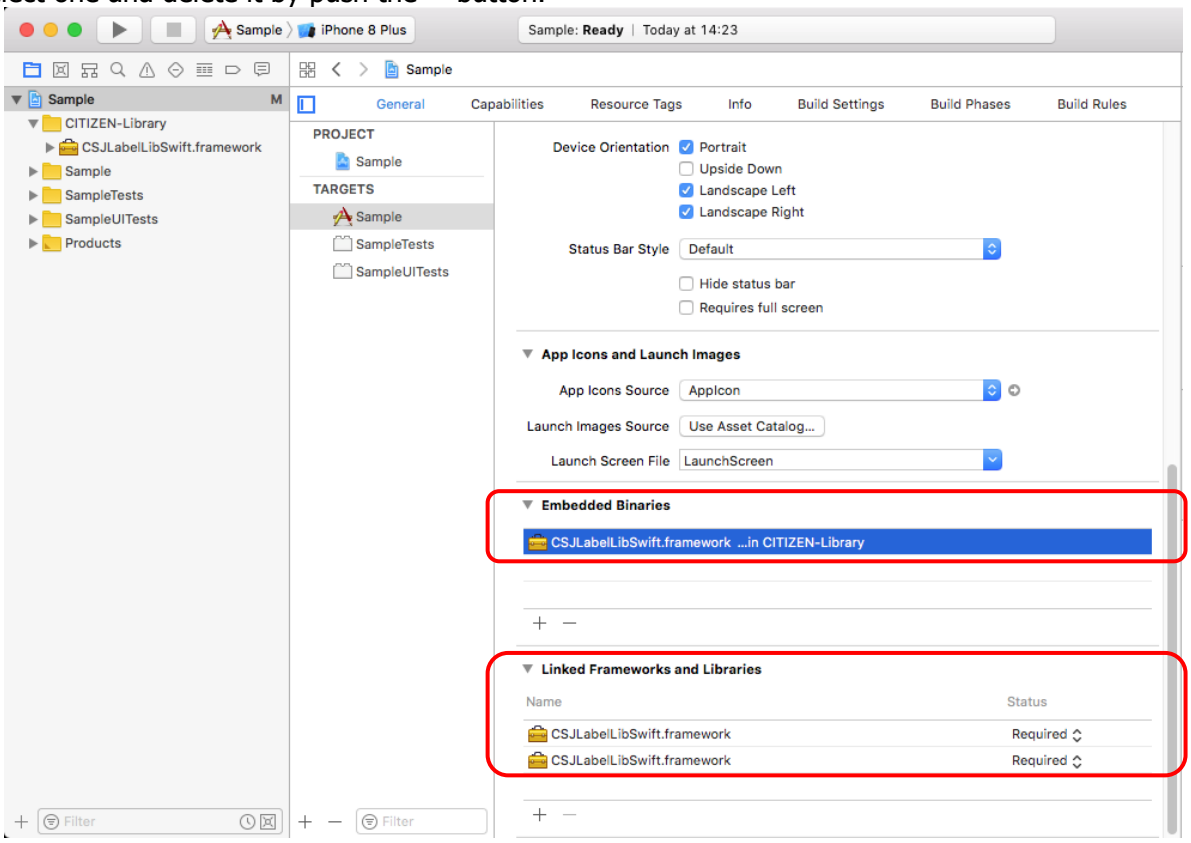
(The screen shot when adding framework in the 'Debug' folder)



(The screen shot when adding framework in the 'Release' folder)

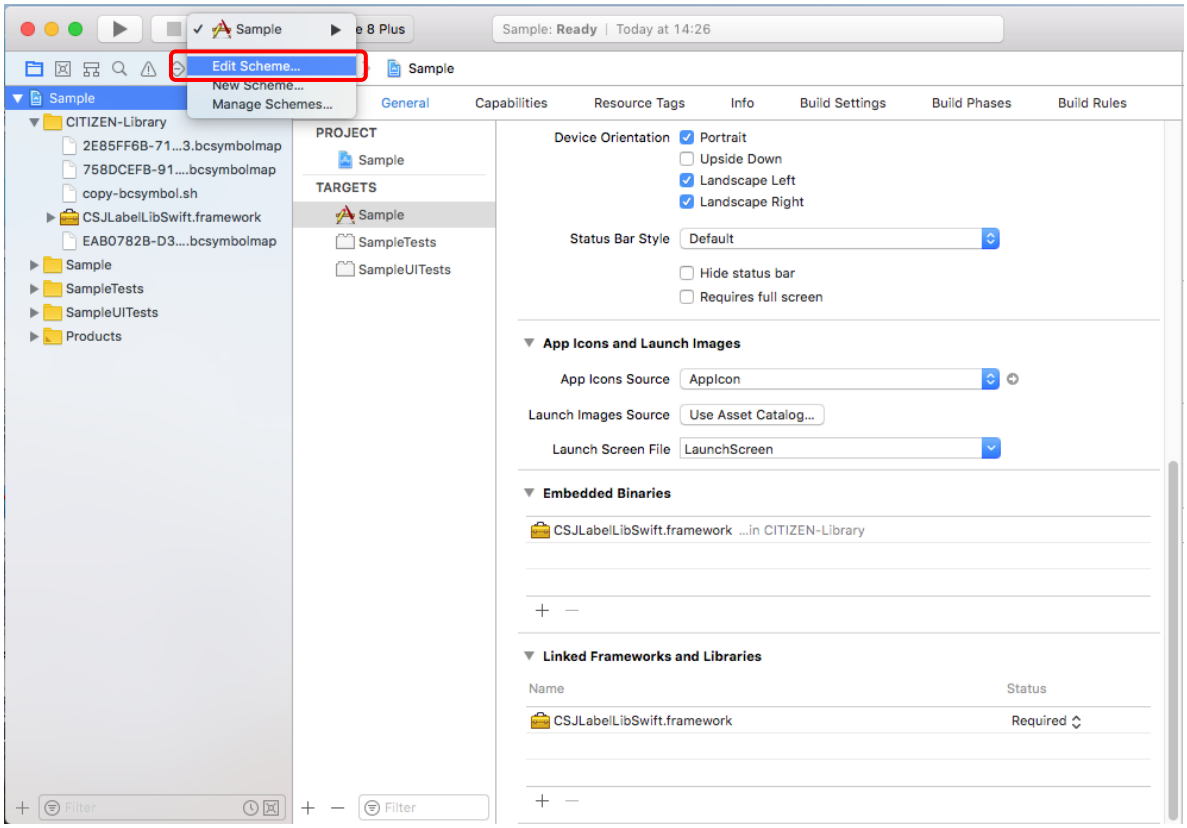


The Project file is selected from "Project Navigator",and it clicks 'Targets' -> 'General'. And expand the items in the "Embedded Binaries", and add the 'CSJLabelLibSwift.framework' by pressing the '+' button. If two 'CSJLabelLibSwift.framework was added to "Linked Frameworks and Libraries" after addition, select one and delete it by push the '-' button.

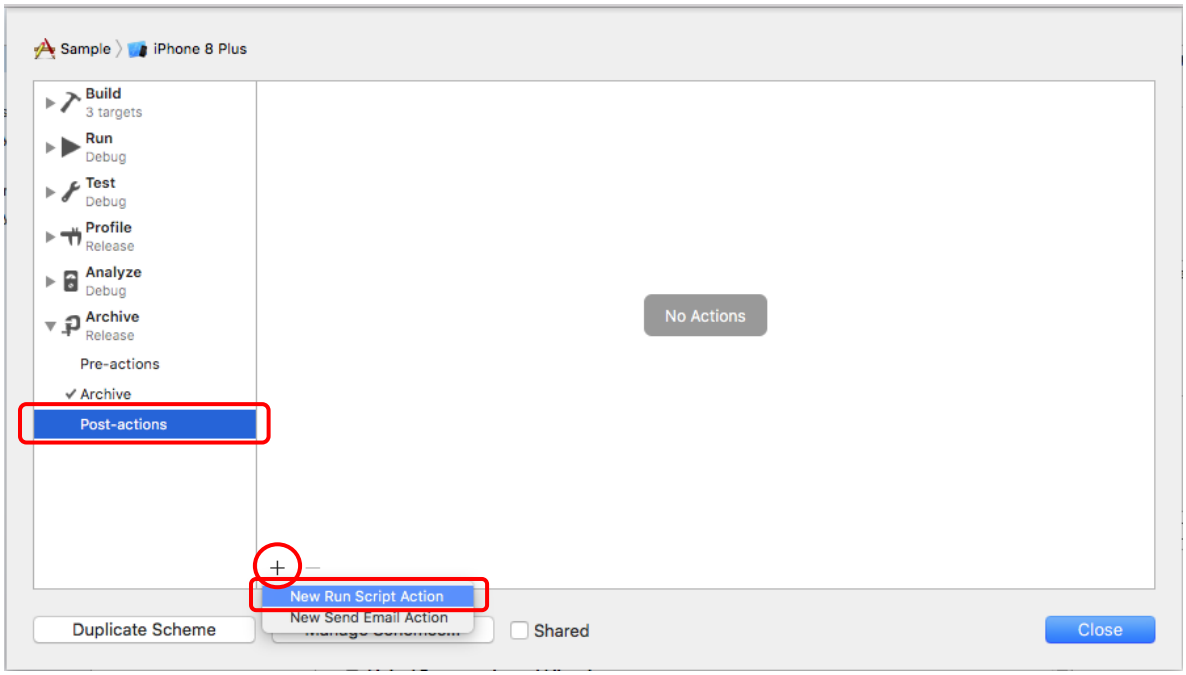


Setting at the time of the archive file making

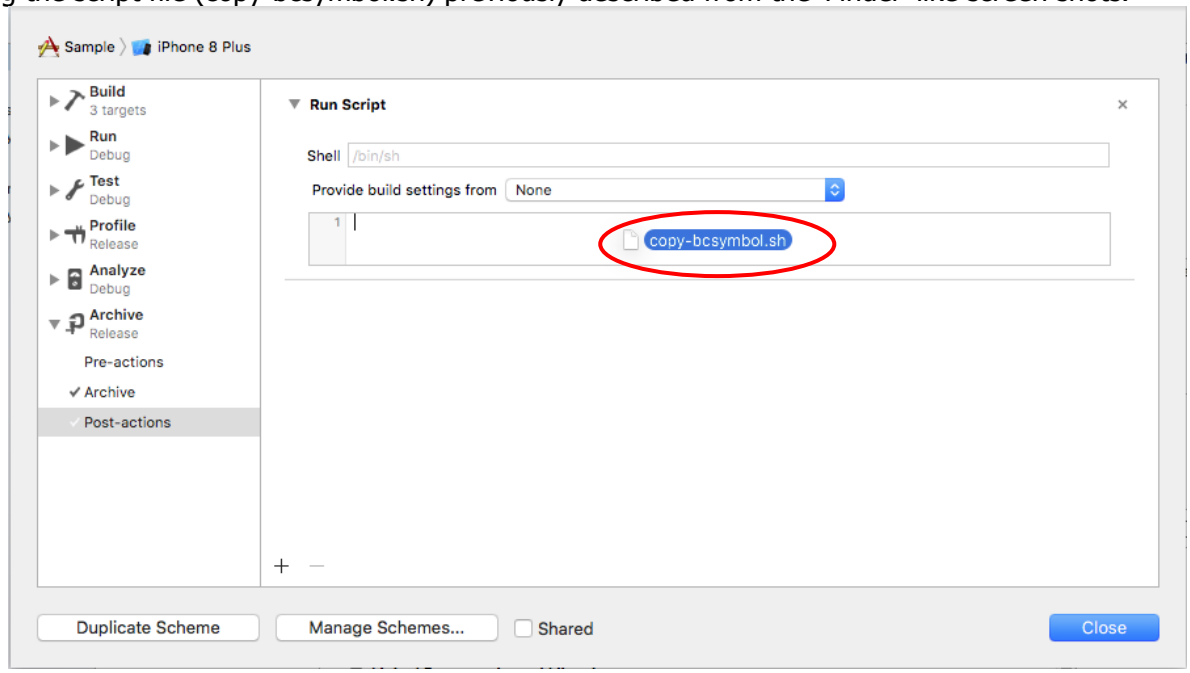
Add a script to run when the archive.
Select the "Edit Scheme".



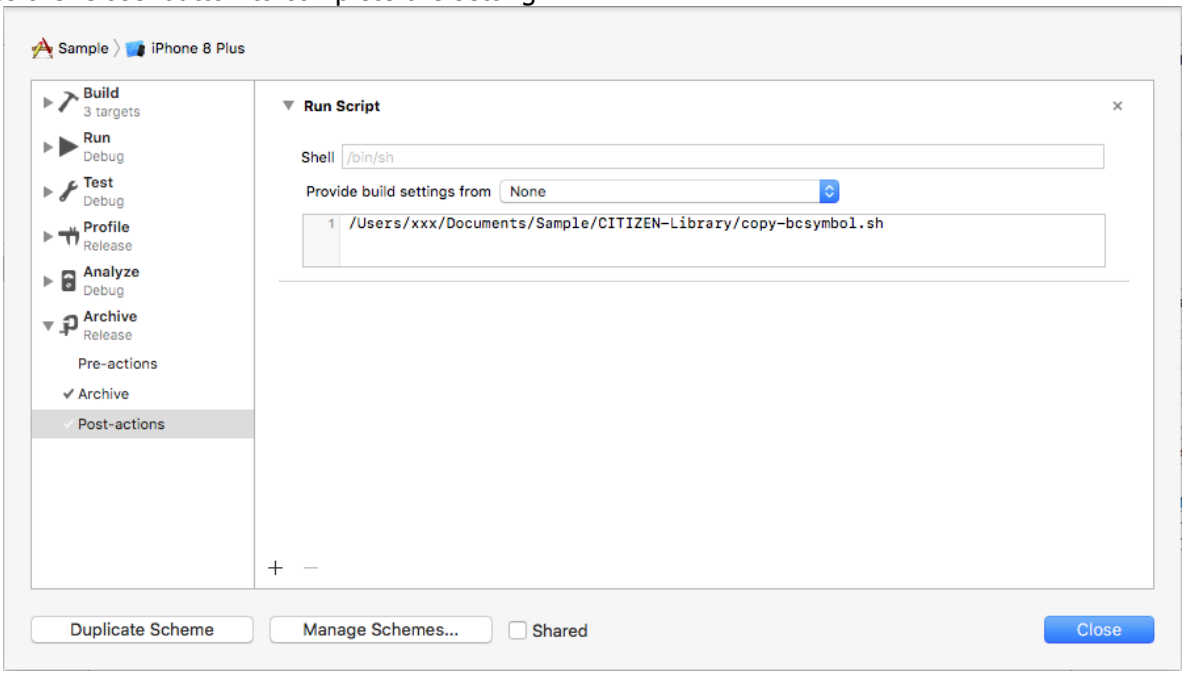
Select 'Archieve' -> 'Post-actions'. Press the bottom left of the '+' and then select the "New Run Script Action".



Drag the script file (copy-bcsymbol.sh) previously described from the 'Finder' like screen shots.

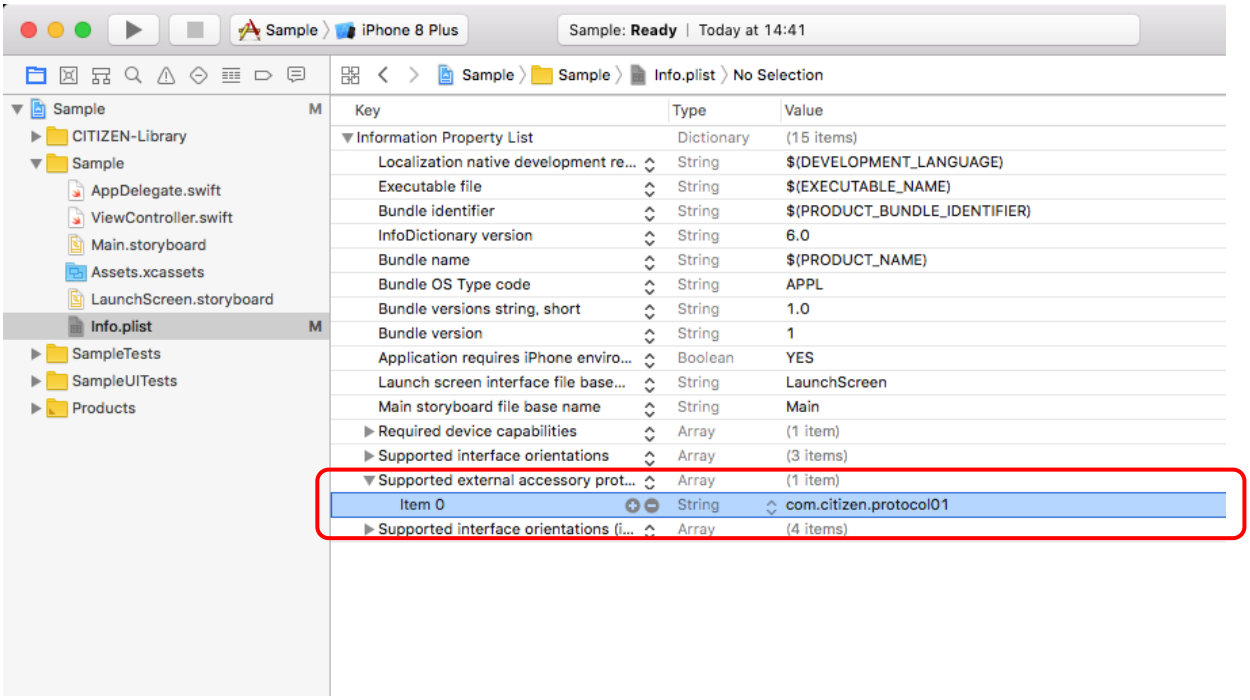


Press the 'Close' button to complete the setting.



Set the development of Bluetooth

Protocol name is added. "Info.plist" file is opened from "Project Navigator". "Add Row" is chosen from the pop-up menu (Open when Control + click), "Supported external accessory protocols" is selected and added. After that, "Supported external accessory protocols" is opened, and "com.citizen.protocol01" is input to the column of "Item 0".



It is completion in the above.

1.4. Use to Bluetooth

Pairing and connection of Bluetooth

To communicate the iOS device with the printer in Bluetooth, it is necessary to discover the printer from the iOS device and to do a pairing. (The following screenshot are examples in iPod Touch/iOS11.2.)

The power supply of the printer is turned on. 'Bluetooth' is chosen from the 'Settings' of the iOS device, and it looks for a peripheral Bluetooth devices. It begins to look for when turning it on when Bluetooth is off.

Peripheral Bluetooth devices are displayed after a while. The device displayed as "CITIZEN SYSTEMS" is a printer. The pairing is necessary when using it for the first time. The connection establishes the printer that the pairing has already been done at once.*

To do the pairing, the tap does the part of "Model name_(lower 2 digits of BD address)". The connection establishes it after a while.

*About Auto Reconnect

The connection from the printer side to the iOS device is recovered for memory switch 13-6 "Auto Reconnect" of the printer. It is necessary to connect it from the “Settings – Bluetooth” of iOS device every time the Bluetooth is disconnected for the specification of iOS. To request the connection automatically to the iOS device connected at the end from the printer side when this function is valid, the connection on the iOS device becomes unnecessary.

Please be invalid “Auto Reconnect” because this function decreases convenience when two or more iOS devices and printers are alternately used.

Please be invalid “Auto Reconnect” when it is connected with the device of other OS.

1.5. Class summary

Methods (LabelPrinter class)

No	Method	Description
1	LabelPrinter	Constructor.
2	connect	Connects to a printer.
3	disconnect	Disconnects with a printer.
4	printerCheck	Gets status of a printer.
5	print	Prints labels.
6	storeNVBitmap	Stores a bitmap image into the flash memory.
7	clearOutput	Clears all data in the printer buffer.
8	sendData	Sends binary data to a printer.
9	searchCitizenPrinter	Searches available printers and gets printer information.
10	searchLabelPrinter	Searches available printers and gets addresses.
11	setLog	Sets the log function.

Properties (LabelPrinter class)

No	Property	Attribute	Description
1	HorizontalMagnification	R/W	Horizontal magnification of the label.
2	VerticalMagnification	R/W	Vertical magnification of the label.
3	FormatAttribute	R/W	Specifies how to print overlapped objects.
4	ContinuousMediaLength	R/W	Label length for continuous paper.
5	MeasurementUnit	R/W	Measurement unit. Inches or millimeters.
6	PrintSpeed	R/W	Print speed.
7	FeedSpeed	R/W	Feed speed.
8	SlewSpeed	R/W	Slew speed.
9	BackupSpeed	R/W	Backup speed.
10	PrintDarkness	R/W	Print darkness.
11	DoubleHeat	R/W	Double heat.
12	VerticalOffset	R/W	Vertical offset of the printing position.
13	HorizontalOffset	R/W	Horizontal offset of the printing position.
14	MediaHandling	R/W	Media handling after print. (Pause, Cut or Peel)
15	StartOffset	R/W	Vertical offset of the paper position when start printing.
16	StopOffset	R/W	Vertical offset of the paper position when stop printing.
17	LabelSensor	R/W	Label sensor type. (See through, Reflect or None)
18	PrintMethod	R/W	Thermal transfer or Direct thermal.
19	SensorLocation	R/W	Front or Rear. (for the models which have multiple sensors)
20	CommandInterpreterInAction	R	Status: CommandInterpreterInAction
21	PaperError	R	Status: PaperError
22	RibbonEnd	R	Status: RibbonEnd
23	BatchProcessing	R	Status: BatchProcessing
24	Printing	R	Status: Printing
25	Pause	R	Status: Pause
26	WaitingForPeeling	R	Status: WaitingForPeeling

Methods (LabelDesign class)

No	Method	Description
1	LabelDesign	Constructor.
2	drawTextPtrFont	Draws text by using a printer device font.
3	drawTextDLFont	Draws text by using a downloaded font.
4	drawTextLocalFont	Draws text by using a font installed in the terminal.
5	drawNVBitmap	Draws a bitmap image stored in the flash memory of the printer.
6	drawBitmap	Draws a bitmap file in the computer.
7	drawBarCode	Draws a barcode.
8	drawMaxiCode	
9	drawPDF417	
10	drawDataMatrix	
11	drawQRCode	
12	drawAztec	
13	drawGS1DataBar	
14	drawLine	Draws a line.
15	drawRect	Draws a rectangular.
16	fillRect	Draws a rectangular filled with a specified pattern.
17	drawCircle	Draws a circle.
18	fillCircle	Draws a circle filled with a specified pattern.
19	drawPolygon	Draws a polygon.
20	fillPolygon	Draws a polygon filled with a specified pattern.
21	embedRawDesignCommand	Embeds raw printer commands.

2. API specification

2.1. Return value

Return value	Description
CLS_SUCCESS (0)	The operation has been done successfully.
CLS_E_CONNECTED (1001)	The printer is already connected.
CLS_E_DISCONNECT (1002)	The printer is not connected.
CLS_E_NOTCONNECT (1003)	Failed to connect to the printer.
CLS_E_CONNECT_NOTFOUND (1004)	Failed to confirm the model name after connecting to the printer.
CLS_E_ILLEGAL (1101)	The operation is not supported by the printer or an invalid parameter was used.
CLS_E_OFFLINE (1102)	The printer is offline.
CLS_E_NOEXIST (1103)	The file name does not exist.
CLS_E_FAILURE (1104)	The service could not perform the requested procedure.
CLS_E_TIMEOUT (1105)	The service has been timed out waiting for a response from the printer.
CLS_E_NO_LIST (1106)	A printer could not be found for the printer search.
CLS_EPTR_BADFORMAT (1203)	The file format is not supported.

2.2. LabelPrinter class

2.2.1 Constructor

Syntax
LabelPrinter

Description
Creates and initialize an instance.

Example
```
var : printer? = CSJLabelLibSwift.LabelPrinter()
```

2.2.2 connect method

Syntax

1) func connect(connectType: Int32, withAddrress addr: String?) -> Int32
2) func connect(connectType: Int32, withAddrress addr: String?, withPort port: Int32) -> Int32
3) func connect(connectType: Int32, withAddrress addr: String?, withPort port: Int32, withTimeout timeout: Int32) -> Int32

Parameters

Parameter	[IN/OUT]	Description	Setting range
connectType	[IN]	Interface type	CLS_PORT_WiFi: Wi-Fi connection CLS_PORT_BLUETOOTH: Bluetooth connection
addr	[IN]	IP address to connect or Bluetooth device address or Bluetooth device name	WiFi: 0.0.0.0～255.255.255.255 Bluetooth: 00:00:00:00:00:00～ FF:FF:FF:FF:FF:FF Device name
port	[IN]	Port number	
timeout	[IN]	Timeout (msec)	

Description

This method is used to connect the printer. Please specify the type and address of the printer connection.
When Bluetooth is used, it is possible to connect with only the printer with which pairing and connecting by the BLUETOOTH Setting of the iOS device (Please refer to "1.4 Use of Bluetooth"). And, it should be iOS6 or more to be connected by using the Bluetooth device address. Please connect by using the Bluetooth device name besides.
Connection port number is valid only if Wi-Fi is specified for the connection type. If it is omitted, it connects with number 9100.
Timeout is gives the maximum number of milliseconds to connect printer. If it is omitted, it connects with 4000 milliseconds in the case of Wi-Fi and 8000 milliseconds in the case of Bluetooth.
The recommended timeout is more than 8000 milliseconds in the case of Bluetooth.
When connecting to the printer, this SDK also checks the status of the printer and the supporting models.
When communication with the printer is not necessary, must execute the dissconnect method to disconnect the printer connection. When not disconnect, the next connection will be an error.

Return value

Returns CLS_SUCCESS (0) on success, an error code otherwise. See below for the error codes.
"2.1 Return value" describes all error codes.

Error code	Description
CLS_E_CONNECTED (1001)	The printer is already connected.
CLS_E_NOTCONNECT (1003)	Failed to connec to the printer. (1) The printer is not connected. (2) The printer is not turned on. (3) The port handle could not be obtained.
CLS_E_CONNECT_NOTFOUND (1004)	Failed to confirm if the printer is a supported model. (1) The printer model is not supported.

Example

```
printer!.connect(CLS_PORT_WiFi, withAddrress: "192.168.182.100", withPort:
        9100)
printer!.connect(CLS_PORT_BLUETOOTH, withAddrress: "00:01:90:F0:81:AB",
        withPort: 9100, withTimeout:8000)
printer!.connect(CLS_PORT_BLUETOOTH, withAddrress: "CITIZEN SYSTEMS",
        withPort: 9100, withTimeout:8000)
```

2.2.3 disconnect method

Syntax

func disconnect() -> Int32

Parameters

none

Description

Disconnects a printer.
This must be called when a connection is no longer needed or when an error occurs.

Return value

Returns CLS_SUCCESS(0) on success, an error code otherwise. See "2.1 Return value" for the error codes.

Example

```
printer!.disconnect()
```

2.2.4 printerCheck method

Syntax
func printerCheck() -> Int32

Parameters
none

Description
Gets the status of the printer and stores information into the properties, CommandInterpreterInAction, PaperError, RibbonEnd, BatchProcessing, Printing, Pause and WaitingForPeeling.
When this method returns an error, a communication failure or a device error might occur. The connection should be reestablished by using disconnect/connect methods in this case.

This method should be used to check the printer status before printing especially after a long idle.

This method can also be used to keep a network connection by calling it periodically.

In case of network connection with the CL - S5xx / 6xx / 70x, set "Parallel Error Output" to OFF beforehand. In case of parallel connection with all models, set "Parallel Error Output" to OFF. This can be done in the "Advanced" tab of the Label Printer Utility.

Return value
Returns CLS_SUCCESS (0) on success. See "2.1 Return value" for the error codes.

Example

```
// PrinterCheck
if CLS_SUCCESS == printer!.printerCheck() {

    // CommandInterpreterInAction Property
    if 1 == printer!.getCommandInterpreterInAction() {
        // Command interpreter in action
    }

    // PaperError Property
    if 1 == printer!.getPaperError() {
        // Paper error
    }

    // RibbonEnd Property
    if 1 == printer!.getRibbonEnd() {
        // Ribbon end
    }

    // BatchProcessing Property
    if 1 == printer!.getBatchProcessing() {
        // Batch processing
    }

    // Printing Property
    if 1 == printer!.getPrinting() {
        // Printing
    }

    // Pause Property
    if 1 == printer!.getPause() {
        // Pause
```

```
    }

    // WaitingForPeeling Property
    if 1 == printer!.getWaitingForPeeling() {
        // Waitiong for peeling
    }
}
else
{
    // Fail
}
```

2.2.5 print method

Syntax

func print (design: LabelDesign!, quantity: Int32) -> Int32

Parameters

Parameter	[IN/OUT]	Description	Setting range
design	[IN]	Instance of LabelDesignclass	
quantity	[IN]	Number of labels to print	1 - 9999

Description

Prints labels by sending a label design created in the LabelDesign class, followed by printer configuration commands if any of properties are set.

Return value

Returns CLS_SUCCESS (0) on success. See "2.1 Return value" for the error codes.

Example

```
design.fillCircle(50, y: 50, radius: 50, pattern: CLS_SHADED_PTN_11)
printer!.print(design, quantity: 1)
```

2.2.6 storeNVBitmap method

Syntax

func storeNVBitmap (filePath: String!, name: String!, rotation: Int32, width: Int32, height: Int32) -> Int32

Parameters

Parameter	[IN/OUT]	Description	Setting range
filePath	[IN]	Path to a bitmap file to store	
name	[IN]	File name to store	ASCII characters except below: -Underscore cannot be the first character. -The following symbols. \ / : * ? " < > |
rotation	[IN]	Direction of rotation	CLS_RT_NORMAL : No rotate CLS_RT_RIGHT90 : Rotate right 90 CLS_RT_ROTATE180 : Rotate 180 CLS_RT_LEFT90 : Rotate left 90
width	[IN]	Horizontal size (pixels)	
height	[IN]	Vertical size (pixels)	

Description

Stores a bitmap image into the flash memory with specified settings such as file name, rotation, width and height. The stored bitmap image can be used by the drawNVBitmap method.
Supported graphic file formats are BMP/GIF/EXIF/JPG/PNG/TIFF.

The image is resized by the specified width and height with keeping the original aspect ratio.
Example : The image size will be 200 x 50 pixels under the following conditions.
The original image size : 400 x 100 pixels
Width : 200
Height : 200

If 0 is set to either width or height, the image size will be calculated by another parameter.
Example : The image size will be 800 x 200" pixels under the following conditions.
The original image size : 400 x 100 pixels
Width : 0
Height : 200

Return value

Returns CLS_SUCCESS (0) on success. See "2.1 Return value" for the error codes.

Example

```
// Retrieve files stored in resources
let FileName = NSBundle.mainBundle().pathForResource("sample_1.jpg",
        ofType: nil)

// Store bitmap
printer!.storeNVBitmap(FileName, name: "Sample", rotation: CLS_RT_NORMAL, ,
        width: 0, height: 0)

// Draw bitmap
design.drawNVBitmap("Sample", hexp: 1, vexp: 1, x: 0, y: 0)
```

```
// Print
printer!.print(design, quantity: 1)
```

2.2.7 clearOutPut method

Syntax
func clearOutput() -> Int32

Parameters
none

Description
Clears data in the print buffer. This triggers the initialization process which is equivalent to the boot sequence.

Return value
Returns CLS_SUCCESS (0) on success. See "2.1 Return value" for the error codes.

Example
```
printer!.clearOutput()
```

2.2.8 sendData method

Syntax

func sendData (data: UnsafeMutablePointer<Int8>, withLength length: UInt) -> Int32

Parameters

Parameter	[IN/OUT]	Description	Setting range
data	[IN]	Send data	
length	[IN]	Length of send data	

Description

Sends binary data to a printer.
This can be used only when sending raw printer commands to the printer.

Return value

Returns CLS_SUCCESS (0) on success. See "2.1 Return value" for the error codes.

Example

```
var data: [Int8] = [0x01, 0x23]  // [01]# (Reset)

printer!.sendData(&data, withLength: 2)
```

2.2.9 searchCitizenPrinter method

Syntax

class func searchCitizenPrinter(ifType: Int32, withSearchTime searchTime: Int32, result: UnsafeMutablePointer<Int32>) -> [AnyObject]?

Parameters

Parameter	[IN/OUT]	Description	Setting range
ifType	[IN]	Connection type	CLS_PORT_WiFi: Wi-Fi connection CLS_PORT_BLUETOOTH: Bluetooth connection
searchTime	[IN]	Time out (sec)	1 - 30
result	[OUT]	Error code	---

The output parameter, result receives CLS_SUCCESS (0) on success, an error code otherwise. See below for the error codes. "2.1 Return value" describes all error codes.

Error code	Description
CLS_E_ILLEGAL (1101)	Invalid parameter. (1) The connect type is unsupported. (2) The time out value is out of range.
CLS_E_NO_LIST (1106)	No printer found.

Description

This method is used to search the printer and to obtain the list of the printer information. Please specify the type of the printer connection and the search time. Only WiFi connection is available at the time of the simulator use. After search time passed, set a result to the result parameter and return the information of the found printers as NSArray type.
In case of CLS_PORT_WiFi, the CL-S400/521/531/621/631/700 series and the CL-E720/730/300/303/321/331 series can be found. It may fail to find when the time out value is less than 3 seconds depanding on the network environment.
If the connection type is CLS_PORT_BLUETOOTH, the search completes immediately, regardless of the search time.

Return value

Returns a list of printer information when successful, returns empty otherwise.
The information will be stored in the CitizenPrinterInfo class type. Available information inside of the class is depanding on the connection type.

Connection Type	Property	Available information
CLS_PORT_WiFi	ipAddress	IP Address
	macAddress	MAC Address
	deviceName	(Empty character)
	bdAddress	(Empty character)
CLS_PORT_Bluetooth	ipAddress	(Empty character)
	macAddress	(Empty character)
	deviceName	Bluetooth device name *Depends on printer model
	bdAddress	Bluetooth device address

Example

```
var result = Int32(0)
let array = printer.searchCitizenPrinter(CLS_PORT_WiFi, withSearchTime: 4,
        result: &result)!
for var i = 0; i < array.count; i++ {
```

```
        let prninfo = array[i] as? CitizenPrinterInfo
        println("IP Address : ¥(prninfo.ipAddress)")
        println("MAC Address : ¥(prninfo.macAddress)")
    }
```

2.2.10 searchLabelPrinter method

Syntax

class func searchLabelPrinter(ifType: Int32, withSearchTime searchTime: Int32, result: UnsafeMutablePointer<Int32>) -> [AnyObject]?

Parameters

Parameter	[IN/OUT]	Description	Setting range
ifType	[IN]	Connection type	CLS_PORT_WiFi: Wi-Fi connection CLS_PORT_BLUETOOTH: Bluetooth connection
searchTime	[IN]	Time out (sec)	1 - 30
result	[OUT]	Error code	

The output parameter, result receives CLS_SUCCESS (0) on success, an error code otherwise. See below for the error codes. "2.1 Return value" describes all error codes.

Error code	Description
CLS_E_ILLEGAL (1101)	Invalid parameter. (1) The connect type is unsupported. (2) The time out value is out of range.
CLS_E_NO_LIST (1106)	No printer found.

Description

This method is used to search the printer and to obtain the list of the addresses. Please specify the type of the printer connection and the search time. Only WiFi connection is available at the time of the simulator use. After search time passed, set a result to the result parameter and return the information of the found printers as NSArray type.
In case of CLS_PORT_WiFi, the CL-S400/521/531/621/631/700 series and the CL-E720/730/300/303/321/331 series can be found. It may fail to find when the time out value is less than 3 seconds depanding on the network environment.
If the connection type is CLS_PORT_BLUETOOTH, the search completes immediately, regardless of the search time.

Return value

In the case of CLS_PORT_WiFi for the connection type, return the list of IP addresses of the printers when a search succeeded. When a search fails, return the empty list.
In the case of CLS_PORT_BLUETOOTH for the connection type and iOS6 or more, return the list of Bluetooth device addresses of the printers when a search succeeded. When a search fails, return the empty list. Because it does not correspond to information on the Bluetooth device address for less than iOS6, the empty list is returned.

Example

```
var result = Int32(0)
let array = printer.searchLabelPrinter(CLS_PORT_WiFi, withSearchTime: 4,
        result: &result)!
```

2.2.11 setLog method

Syntax

func setLog(mode: Int32, withPath path: String?, withMaxSize maxSize: Int32)

Parameters

Parameter	[IN/OUT]	Description	Setting range
mode	[IN]	Logging mode	0: None 1: Access logs 2: Error logs
path	[IN]	File path to store	The folder in the application sharing folder.
maxSize	[IN]	Maximum Log Size	0: Unlimited 1 - : Maximum size (MB)

Description

Sets the logging function. See "3.2 Logging function" for more details.

Return value

none

Example

```
printer.setLog(1, withPath: "/", withMaxSize: 10)
```

2.2.12 HorizontalMagnification property

Syntax
Int32

Attribute
Read/Write

Description
Horizontal dot size. "1" or "2".

Setter method
func setHorizontalMagnification(horizontalMagnification: Int32) -> Int32

"1" or "2" must be set as a parameter.
Returns CLS_SUCCESS (0) on success. See "2.1 Return value" for the error codes.

Getter method
func getHorizontalMagnification() -> Int32

Example

```
var pixelSize: Int32! = 0

printer!.setHorizontalMagnification(2)
pixelSize = printer!.getHorizontalMagnification()
```

2.2.13 VerticalMagnification property

Syntax
Int32

Attribute
Read/Write

Description
Vertical dot size. "1", "2" or "3".

Setter method
func setVerticalMagnification(verticalMagnification: Int32) -> Int32

"1", "2" or "3" must be set as a parameter.
Returns CLS_SUCCESS (0) on success. See "2.1 Return value" for the error codes.

Getter method
func getVerticalMagnification() -> Int32

Example

```
var pixelSize: Int32! = 0

printer!.setVerticalMagnification(3)
pixelSize = printer!.getVerticalMagnification()
```

2.2.14 FormatAttribute property

Syntax
Int32

Attribute
Read/Write

Description
Specifies how to print the area where multiple objects are overlapped.

0: XOR mode. Overlapped area will not be printed.
1: OR mode. Overlapped area will be printed.

Setter method
func setFormatAttribute(formatAttribute: Int32) -> Int32

Returns CLS_SUCCESS (0) on success. See "2.1 Return value" for the error codes.

Getter method
func getFormatAttribute() -> Int32

Returns the set value. If nothing has been set, it returns CLS_PROPERTY_DEFAULT(999999).

Example

```
var attr: Int32! = 0

printer!.setFormatAttribute(1)
attr = printer!.getFormatAttribute()
```

Printing Result

XOR 

OR 

2.2.15 ContinuousMediaLength property

Syntax
Int32

Attribute
Read/Write

Description
Sets a paper length for continuous paper.

Inch system	0001 – 9999 (0.01 - 99.99 inches)
Metric system	0001 – 9999 (0.1 - 999.9 mm)

Setter method
func setContinuousMediaLength(continuousMediaLength: Int32) -> Int32

The default value in the printer is "0000."
Returns CLS_SUCCESS (0) on success. See "2.1 Return value" for the error codes.

Getter method
func getContinuousMediaLength() -> Int32

Returns the set value. If nothing has been set, it returns CLS_PROPERTY_DEFAULT(999999).

Example
```
var mediaLength: Int32! = 0

printer!.setContinuousMediaLength(2000)
mediaLength = printer!.getContinuousMediaLength()
```

2.2.16 MeasurementUnit property

Syntax
Int32

Attribute
Read/Write

Description
Sets the measurement unit.

Value	Description
CLS_UNIT_MILLI (0)	Metric system
CLS_UNIT_INCH (1)	Inch system

Setter method
func setMeasurementUnit(measurementUnit: Int32) -> Int32

The default value in the printer is "1."
Returns CLS_SUCCESS (0) on success. See "2.1 Return value" for the error codes.

Getter method
func getMeasurementUnit() -> Int32

Returns the set value. If nothing has been set, it returns CLS_PROPERTY_DEFAULT(999999).

Example
```
var unit: Int32! = 0

printer!.setMeasurementUnit(CLS_UNIT_MILLI)
unit = printer!.getMeasurementUnit()
```

2.2.17 PrintSpeed property

Syntax
Int32

Attribute
Read/Write

Description
Sets the print speed.
See "3.3.Predefined Constants", No.21 for available value.
*Initial and maximum value vary depending on the printer model.

Setter method
func setPrintSpeed(printSpeed: Int32) -> Int32

Returns CLS_SUCCESS (0) on success. See "2.1 Return value" for the error codes.

Getter method
func getPrintSpeed() -> Int32

Returns the set value. If nothing has been set, it returns CLS_PROPERTY_DEFAULT(999999).

Example
```
var printSpeed: Int32! = 0

printer!.setPrintSpeed(CLS_SPEEDSETTING_X)
printSpeed = printer!.getPrintSpeed()
```

2.2.18 FeedSpeed property

Syntax
Int32

Attribute
Read/Write

Description
Sets the feed speed.
See "3.3.Predefined Constants", No.21 for available value.
*Initial and maximum value vary depending on the printer model.

Setter method
func setFeedSpeed(_ feedSpeed: Int32) -> Int32

Returns CLS_SUCCESS (0) on success. See "2.1 Return value" for the error codes.

Getter method
func getFeedSpeed() -> Int32

Returns the set value. If nothing has been set, it returns CLS_PROPERTY_DEFAULT(999999).

Example
```
var feedSpeed: Int32! = 0

printer!.setFeedSpeed(LabelConst.CLS_SPEEDSETTING_W)
feedSpeed = printer!.getFeedSpeed()
```

2.2.19 SlewSpeed property

Syntax
Int32

Attribute
Read/Write

Description
Sets the slew speed.
See "3.3.Predefined Constants", No.21 for available value.
*Initial and maximum value vary depending on the printer model.

Setter method
func setSlewSpeed(slewSpeed: Int32) -> Int32

Returns CLS_SUCCESS (0) on success. See "2.1 Return value" for the error codes.

Getter method
func getSlewSpeed() -> Int32

Returns the set value. If nothing has been set, it returns CLS_PROPERTY_DEFAULT(999999).

Example
```
var slewSpeed: Int32! = 0

printer!.setSlewSpeed(CLS_SPEEDSETTING_V)
slewSpeed = printer!.getSlewSpeed()
```

2.2.20 BackupSpeed property

Syntax
Int32

Attribute
Read/Write

Description
Sets the backup speed. See below for the acceptable values.

Value	Description
CLS_SPEEDSETTING_A or CLS_SPEEDSETTING_B	1.0 inch(25.4mm) / sec
CLS_SPEEDSETTING_C or CLS_SPEEDSETTING_D	2.0 inch(50.8mm) / sec
CLS_SPEEDSETTING_E or CLS_SPEEDSETTING_F	3.0 inch(76.2mm) / sec
CLS_SPEEDSETTING_G or CLS_SPEEDSETTING_H	4.0 inch(101.6mm) / sec
CLS_SPEEDSETTING_I or CLS_SPEEDSETTING_J	5.0 inch(127.0mm) / sec
CLS_SPEEDSETTING_K or CLS_SPEEDSETTING_L	6.0 inch(152.4mm) / sec
CLS_SPEEDSETTING_M or CLS_SPEEDSETTING_N	7.0 inch(177.8mm) / sec
CLS_SPEEDSETTING_O	8.0 inch(203.2mm) / sec
CLS_SPEEDSETTING_1 - CLS_SPEEDSETTING_8	1.0 inch(25.4mm) / sec – 8.0 inch(203.2mm) / sec

*Initial and maximum value vary depending on the printer model.

Setter method
func setBackupSpeed(backupSpeed: Int32) -> Int32

Returns CLS_SUCCESS (0) on success. See "2.1 Return value" for the error codes.

Getter method
func getBackupSpeed() -> Int32

Returns the set value. If nothing has been set, it returns CLS_PROPERTY_DEFAULT(999999).

Example

```
var backupSpeed: Int32! = 0

printer!.setBackupSpeed(CLS_SPEEDSETTING_O)
backupSpeed = printer!.getBackupSpeed()
```

2.2.21 PrintDarkness property

Syntax
Int32

Attribute
Read/Write

Description
Sets the print density. The acceptable values are below:

Setting range	0 - 30
Default value	10

Setter method
func setPrintDarkness(printDarkness: Int32) -> Int32

Returns CLS_SUCCESS (0) on success. See "2.1 Return value" for the error codes.

Getter method
func getPrintDarkness() -> Int32

Returns the set value. If nothing has been set, it returns CLS_PROPERTY_DEFAULT(999999).

Example
```
var printDarkness: Int32! = 0

printer!.setPrintDarkness(30)
printDarkness = printer!.getPrintDarkness()
```

2.2.22 DoubleHeat property

Syntax
Int32

Attribute
Read/Write

Description
Sets the double heat option. The acceptable values are below:

0: OFF (Default)
1: ON

Setter method
func setDoubleHeat(doubleHeat: Int32) -> Int32

Returns CLS_SUCCESS (0) on success. See "2.1 Return value" for the error codes.

Getter method
func getDoubleHeat() -> Int32

Returns the set value. If nothing has been set, it returns CLS_PROPERTY_DEFAULT(999999).

Example
```
var doubleHeat: Int32! = 0

printer!.setDoubleHeat(1)
doubleHeat = printer!.getDoubleHeat()
```

2.2.23 VerticalOffset property

Syntax
Int32

Attribute
Read/Write

Description
Sets the vertical offset to adjust the vertical printing position on the paper.

Inch system	0000 – 9999 (0.00 inch - 99.99 inches)
Metric system	0000 – 9999 (0.0 mm - 999.9 mm)
Default value	0000

Setter method
func setVerticalOffset(verticalOffset: Int32) -> Int32

Returns CLS_SUCCESS (0) on success. See "2.1 Return value" for the error codes.

Getter method
func getVerticalOffset() -> Int32

Returns the set value. If nothing has been set, it returns CLS_PROPERTY_DEFAULT(999999).

Example
```
var verticalOffset: Int32! = 0

printer!.setVerticalOffset(10)
verticalOffset = printer!.getVerticalOffset()
```

2.2.24 HorizontalOffset property

Syntax
Int32

Attribute
Read/Write

Description
Sets the horizontal offset to adjust the horizontal printing position on the paper.

Inch system	0000 – 9999 (0.00 inch - 99.99 inches)
Metric system	0000 – 9999 (0.0 mm - 999.9 mm)
Default value	0000

Setter method
func setHorizontalOffset(horizontalOffset: Int32) -> Int32

Returns CLS_SUCCESS (0) on success. See "2.1 Return value" for the error codes.

Getter method
func getHorizontalOffset() -> Int32

Returns the set value. If nothing has been set, it returns CLS_PROPERTY_DEFAULT(999999).

Example
```
var horizontalOffset: Int32! = 0

printer!.setHorizontalOffset(20)
horizontalOffset = printer!.getHorizontalOffset()
```

2.2.25 MediaHandling property

Syntax
Int32

Attribute
Read/Write

Description
Sets the way how to handle media after printing.

Value	Description
CLS_MEDIAHANDLING_NONE (0)	To print without any special actions.
CLS_MEDIAHANDLING_TEAROFF (1)	To feed labels to the tear bar.
CLS_MEDIAHANDLING_DISPENSES (2)	To feed labels to the tear bar, and wait for removal.
CLS_MEDIAHANDLING_PAUSE (3)	To pause after printing.
CLS_MEDIAHANDLING_CUT (4)	To cut after printing.
CLS_MEDIAHANDLING_CUTANDPAUSE (5)	To cut and pause after printing.
CLS_MEDIAHANDLING_PEELOFF (6)	To peel labels off the backing and wait for removal.
CLS_MEDIAHANDLING_REWIND (7)	To use a rewinder.

Setter method
func setMediaHandling(mediaHandling: Int32) -> Int32

Returns CLS_SUCCESS (0) on success. See "2.1 Return value" for the error codes.

Getter method
func getMediaHandling() -> Int32

Returns the set value. If nothing has been set, it returns CLS_PROPERTY_DEFAULT(999999).

Example

```
var mediaHandling: Int32! = 0

printer!.setMediaHandling(CLS_MEDIAHANDLING_PAUSE)
mediaHandling = printer!.getMediaHandling()
```

2.2.26 StartOffset property

Syntax
Int32

Attribute
Read/Write

Description
Specifies the distance between paper sensor and print head to change the print starting position.

Inch system			Metric system		
Initial value	Minimum value	Max value	Initial value	Minimum value	Max value
0220	0120	0320	0559	0305	0813

*0.01 inches or 0.1 mm

Setter method
func setStartOffset(startOffset: Int32) -> Int32

Make sure to set a valid value since the consistency with the selected measurement unit will never be checked.
Returns CLS_SUCCESS (0) on success. See "2.1 Return value" for the error codes.

Getter method
func getStartOffset() -> Int32

Returns the set value. If nothing has been set, it returns CLS_PROPERTY_DEFAULT(999999).

Example
```
var startOffset: Int32! = 0

printer!.setStartOffset(220)
startOffset = printer!.getStartOffset()
```

2.2.27 StopOffset property

Syntax
Int32

Attribute
Read/Write

Description
Specifies the distance between paper sensor and cutter or peeler to change the print stop position.

Note that the initial values are calculated to stop paper at an appropriate position. An insufficient value truncates the printed label, an exceeded value truncates the next label.

Media handling	Inch system			Metric system		
	Initial value	Minimum value	Max value	Initial value	Minimum value	Max value
None	0000	0000	9999	0000	0000	9999
Cutter	0100			0254		
Peel Off	0050			0127		
Tear Off	0070			0178		

*0.01 inches or 0.1 mm

Setter method
func setStopOffset(stopOffset: Int32) -> Int32

Returns CLS_SUCCESS (0) on success. See "2.1 Return value" for the error codes.

Getter method
func getStopOffset() -> Int32

Returns the set value. If nothing has been set, it returns CLS_PROPERTY_DEFAULT(999999).

Example
```
var stopOffset: Int32! = 0

printer!.setStopOffset(240)
stopOffset = printer!.getStopOffset()
```

2.2.28 LabelSensor property

Syntax
Int32

Attribute
Read/Write

Description
Sets the label sensor type.

Value	Description
CLS_SELSENSOR_NONE (0)	None. This assumes that a continuous paper roll is used. Therefore a valid length must be set in the ContinuousMediaLength property, returns CLS_E_ILLEGAL (1101) otherwise.
CLS_SELSENSOR_SEETHROUGH (1)	Gap sensor. To detect a gap between labels or a notch on tags.
CLS_SELSENSOR_REFLECT (2)	Reflect sensor. Detect a black mark on the back side of the label.

Setter method
func setLabelSensor(labelSensor: Int32) -> Int32

Returns CLS_SUCCESS (0) on success. See "2.1 Return value" for the error codes.

Getter method
func getLabelSensor() -> Int32

Returns the set value. If nothing has been set, it returns CLS_PROPERTY_DEFAULT(999999).

Example

```
// Other than "CLS_SELSENSOR_NONE"
var labelSensor: Int32! = 0

printer!.setLabelSensor(CLS_SELSENSOR_SEETHROUGH);
labelSensor = printer!.getLabelSensor();


// "CLS_SELSENSOR_NONE"
var labelSensor: Int32! = 0

printer!.setContinuousMediaLength(100)
printer!.setLabelSensor(CLS_SELSENSOR_NONE)
labelSensor = printer!.getLabelSensor()
```

2.2.29 PrintMethod property

Syntax
Int32

Attribute
Read/Write

Description
Sets the print method to either thermal transfer or direct thermal.

Value	Description
CLS_PRTMETHOD_TT (0)	Thermal transfer
CLS_PRTMETHOD_DT (1)	Direct thermal

Setter method
func setPrintMethod(printMethod: Int32) -> Int32

Returns CLS_SUCCESS (0) on success. See "2.1 Return value" for the error codes.

Getter method
func getPrintMethod() -> Int32

Returns the set value. If nothing has been set, it returns CLS_PROPERTY_DEFAULT(999999).

Example

```
var printMethod: Int32! = 0

printer!.setPrintMethod(CLS_PRTMETHOD_DT)
printMethod = printer!.getPrintMethod()
```

2.2.30 SensorLocation property

Syntax
Int32

Attribute
Read/Write

Description
Selects the sensor to use for the models which have multiple paper sensors, front and rear.
The choice will be stored in the non-volatile flash memory and will be kept until overwritten.

Value	Description
CLS_SENS_LOCATION_FRONT (0)	Front sensor
CLS_SENS_LOCATION_ADJUSTABLE (1)	Rear sensor

Setter method
func setSensorLocation(sensorLocation: Int32) -> Int32

Returns CLS_SUCCESS (0) on success. See "2.1 Return value" for the error codes.

Getter method
func getSensorLocation() -> Int32

Returns the set value. If nothing has been set, it returns CLS_PROPERTY_DEFAULT(999999).

Example
```
var location: Int32! = 0

printer!.setSensorLocation(CLS_SENS_LOCATION_ADJUSTABLE)
location = printer!.getSensorLocation()
```

2.2.31 CommandInterpreterInAction property

Syntax
Int32

Attribute
Read

Description
Shows the printer status if the interpreter is processing commands.

Note that the status properties won't be updated automatically.
printerCheck method checks the status and updates the value, must be called before referring to this property.

Setter method
none

Getter method
func getCommandInterpreterInAction() -> Int32

Returns one of the following values.

Value	Description
0	No
1	Yes
CLS_PROPERTY_DEFAULT(999999)	Nothing has been set.

Example
See printerCheck method.

2.2.32 PaperError property

Syntax
Int32

Attribute
Read

Description
Shows the printer status if there is a paper error.

Note that the status properties won't be updated automatically.
printerCheck method checks the status and updates the value, must be called before referring to this property.

Setter method
none

Getter method
func getPaperError() -> Int32

Returns one of the following values.

Value	Description
0	No
1	Yes
CLS_PROPERTY_DEFAULT(999999)	Nothing has been set.

Example
See printerCheck method.

2.2.33 RibbonEnd property

Syntax
Int32

Attribute
Read

Description
Shows the printer status if there is a ribbon end error.

Note that the status properties won't be updated automatically.
printerCheck method checks the status and updates the value, must be called before referring to this property.

Setter method
none

Getter method
func getRibbonEnd() -> Int32

Returns one of the following values.

Value	Description
0	No
1	Yes
CLS_PROPERTY_DEFAULT(999999)	Nothing has been set.

Example
See printerCheck method.

2.2.34 BatchProcessing property

Syntax
Int32

Attribute
Read

Description
Shows the status if the printer is processing batch print jobs.

Note that the status properties won't be updated automatically.
printerCheck method checks the status and updates the value, must be called before referring to this property.

Setter method
none

Getter method
func getBatchProcessing() -> Int32

Returns one of the following values.

Value	Description
0	No
1	Yes
CLS_PROPERTY_DEFAULT(999999)	Nothing has been set.

Example
See printerCheck method.

2.2.35 Printing property

Syntax
Int32

Attribute
Read

Description
Shows the status if the printer is currently printing.

Note that the status properties won't be updated automatically.
printerCheck method checks the status and updates the value, must be called before referring to this property.

Setter method
none

Getter method
func getPrinting() -> Int32

Returns one of the following values.

Value	Description
0	No
1	Yes
CLS_PROPERTY_DEFAULT(999999)	Nothing has been set.

Example
See printerCheck method.

2.2.36 Pause property

Syntax
Int32

Attribute
Read

Description
Shows the status if the printer is in the pause mode.

Note that the status properties won't be updated automatically.
printerCheck method checks the status and updates the value, must be called before referring to this property.

Setter method
none

Getter method
func getPause() -> Int32

Returns one of the following values.

Value	Description
0	No
1	Yes
CLS_PROPERTY_DEFAULT(999999)	Nothing has been set.

Example
See printerCheck method.

2.2.37 WaitingForPeeling property

Syntax
Int32

Attribute
Read

Description
Shows the status if the printer is in the wait for peeling status.

Note that the status properties won't be updated automatically.
printerCheck method checks the status and updates the value, must be called before referring to this property.

Setter method
none

Getter method
func getWaitingForPeeling() -> Int32

Returns one of the following values.

Value	Description
0	No
1	Yes
CLS_PROPERTY_DEFAULT(999999)	Nothing has been set.

Example
See printerCheck method.

2.3. LabelDesign class

2.3.1 Constructor

Syntax
LabelDesign

Parameters
none

Description
Creates and initialize an instance.

Example
```
var : design? = CSJLabelLibSwift.LabelDesign()
```

2.3.2 drawTextPtrFont method

Syntax

func drawTextPtrFont(data: String!, locale: Int32, font: Int32, rotation: Int32, hexp: Int32, vexp: Int32, size: Int32, x: Int32, y: Int32) -> Int32

Parameters

Parameter	[IN/OUT]	Description	Setting range
data	[IN]	String data	* Unsupported characters will be replaced with blank text.
locale	[IN]	Locale	CLS_LOCALE_JP: Japan CLS_LOCALE_OTHER: Other CLS_LOCALE_CN: Chinese CLS_LOCALE_KR: Korean
font	[IN]	Font typeface	CLS_PRT_FNT_0: SystemFont 0 CLS_PRT_FNT_1: SystemFont 1 CLS_PRT_FNT_2: SystemFont 2 CLS_PRT_FNT_3: SystemFont 3 CLS_PRT_FNT_4: SystemFont 4 CLS_PRT_FNT_5: SystemFont 5 CLS_PRT_FNT_6: SystemFont 6 CLS_PRT_FNT_7: SystemFont 7 CLS_PRT_FNT_8: SystemFont 8 CLS_PRT_FNT_TRIUMVIRATE: Smooth font CLS_PRT_FNT_TRIUMVIRATE_B: Smooth font (bold) CLS_PRT_FNT_KANJI: Kanji (left to right) CLS_PRT_FNT_KANJIT: Kanji (top to bottom)
rotation	[IN]	Direction of rotation	CLS_RT_NORMAL: No rotation CLS_RT_RIGHT90: Rotate CW 90 CLS_RT_ROTATE180: Rotate CW 180 CLS_RT_LEFT90: Rotate CCW 90
hexp	[IN]	Horizontal magnification	1 – 24
vexp	[IN]	Vertical magnification	1 – 24
size	[IN]	Font size	CLS_PRT_FNT_SIZE_4: 4pt CLS_PRT_FNT_SIZE_5: 5pt CLS_PRT_FNT_SIZE_6: 6pt CLS_PRT_FNT_SIZE_8: 8pt CLS_PRT_FNT_SIZE_10: 10pt CLS_PRT_FNT_SIZE_12: 12pt CLS_PRT_FNT_SIZE_14: 14pt CLS_PRT_FNT_SIZE_18: 18pt CLS_PRT_FNT_SIZE_24: 24pt CLS_PRT_FNT_SIZE_30: 30pt CLS_PRT_FNT_SIZE_36: 36pt CLS_PRT_FNT_SIZE_48: 48pt CLS_PRT_FNT_KANJI_SIZE_16: Kanji 16dot CLS_PRT_FNT_KANJI_SIZE_24: Kanji 24dot CLS_PRT_FNT_KANJI_SIZE_32: Kanji 32dot * CLS_PRT_FNT_KANJI_SIZE_48: Kanji 48dot * * Not supported in locale for Chinese model.
x	[IN]	X-coordinate	0000 - 9999 * The origin is at bottom-left.(0, 0)
y	[IN]	Y-coordinate	

Description

Draws characters by using a printer device font with specifying options such as rotation, magnification, size and coordinates.

Available font sizes depend on typeface.
Kanji: 16, 24, 32, 48 *dots
Smooth font: 4, 5, 6, 8, 10, 12, 14, 18, 24, 30, 36, 48
Other: Fixed sizes depend on typeface.

Return value

Returns CLS_SUCCESS(0) on success, an error code otherwise. See "2.1 Return value" for the error codes.

Example

```
// Locale:OTHER
design.drawTextPtrFont("drawTextPtrFont",
        locale: CLS_LOCALE_OTHER, font: CLS_PRT_FNT_TRIUMVIRATE,
        rotation: CLS_RT_NORMAL, hexp: 1, vexp: 1,
        size: CLS_PRT_FNT_SIZE_10, x: 100, y: 100)


// Locale:JP
design.drawTextPtrFont("テキスト印刷(プリンタフォント)",
        locale: CLS_LOCALE_JP, font: CLS_PRT_FNT_KANJI,
        rotation: CLS_RT_NORMAL, hexp: 1, vexp: 1,
        size: CLS_PRT_FNT_KANJI_SIZE_16, x: 100, y: 300)


// Locale:CN
design.drawTextPtrFont("测试打印", locale: CLS_LOCALE_CN,
        font: CLS_PRT_FNT_KANJI, rotation: CLS_RT_NORMAL,
        hexp: 1, vexp: 1, size: CLS_PRT_FNT_KANJI_SIZE_16,
        x: 100, y: 300);


// Locale:KR
design.drawTextPtrFont("테스트 인쇄", locale: CLS_LOCALE_KR,
        font: CLS_PRT_FNT_KANJI, rotation: CLS_RT_NORMAL,
        hexp: 1, vexp: 1, size: CLS_PRT_FNT_KANJI_SIZE_16,
        x: 100, y: 300);
```

2.3.3 drawTextDLFont method

Syntax

func drawTextDLFont(data: String!, encoding: Int32, fontID: String!, rotation: Int32, hexp: Int32, vexp: Int32, point: Int32, x: Int32, y: Int32) -> Int32

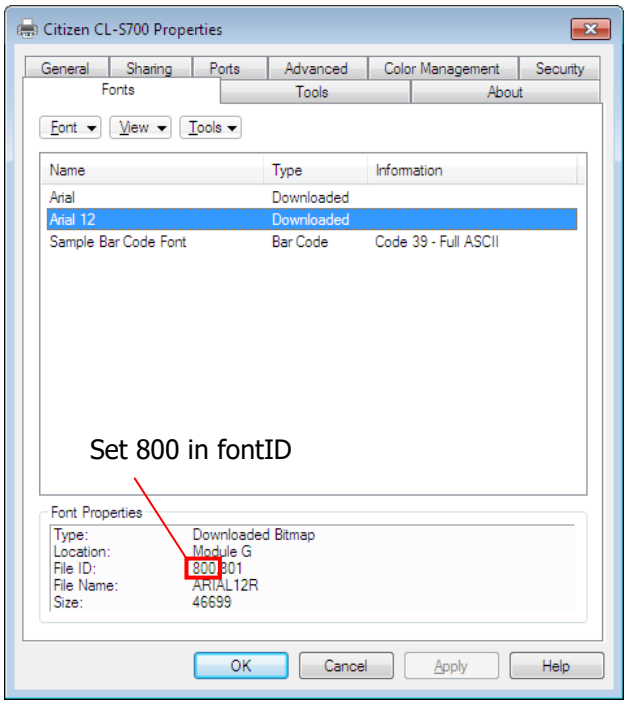
Parameters

Parameter	[IN/OUT]	Description	Setting range
data	[IN]	String data	* Unsupported characters will be replaced with blank text.
encoding	[IN]	Character encoding	See "3.3.Predefined Constants."
fontID	[IN]	Font ID stored in the printer	Numeric characters - Bitmap font Alphanumeric characters - TrueType font
rotation	[IN]	Direction of rotation	CLS_RT_NORMAL: No rotation CLS_RT_RIGHT90: Rotate CW 90 CLS_RT_ROTATE180: Rotate CW 180 CLS_RT_LEFT90: Rotate CCW 90
hexp	[IN]	Horizontal magnification	1 - 24
vexp	[IN]	Vertical magnification	1 - 24
point	[IN]	Font size	001 - 999
x	[IN]	X-coordinate	0000 - 9999 * The origin is at bottom-left.(0, 0)
y	[IN]	Y-coordinate	

Description

Draws characters by using a downloaded font with specifying options such as rotation, magnification, size and coordinates.

In case of bitmap font, the driver shows two IDs. Use the smaller number of ID.



Return value

Returns CLS_SUCCESS(0) on success, an error code otherwise. See "2.1 Return value" for the error codes.

Example

```
// TrueType font
design.drawTextDLFont("TrueType",
        encoding: CLS_ENC_CDPG_IBM850, fontID: "S50",
        rotation: CLS_RT_NORMAL, hexp: 1, vexp: 1, point: 12,
        x: 100, y: 100)


// Bitmap font
design.drawTextDLFont("Bitmap",
        encoding: CLS_ENC_CDPG_IBM850, fontID: "800",
        rotation: CLS_RT_NORMAL, hexp: 1, vexp: 1, point: 12,
        x: 100, y: 200)
```

2.3.4 drawTextLocalFont method

Syntax

1) func drawTextLocalFont(data: String!, fontName: String!, rotation: Int32, hRatio: Int32, vRatio: Int32, point: Int32, style: Int32, x: Int32, y: Int32) -> Int32
2) func drawTextLocalFont(data: String!, fontName: String!, rotation: Int32, hRatio: Int32, vRatio: Int32, point: Int32, style: Int32, x: Int32, y: Int32 , resolution: Int32) -> Int32
3) func drawTextLocalFont(data: String!, fontName: String!, rotation: Int32, hRatio: Int32, vRatio: Int32, point: Int32, style: Int32, x: Int32, y: Int32 , resolution: Int32 , measurementUnit: Int32) -> Int32

Parameters

Parameter	[IN/OUT]	Description	Setting range
data	[IN]	String data	
fontName	[IN]	Font name	Font name
rotation	[IN]	Direction of rotation	CLS_RT_NORMAL: No rotation CLS_RT_RIGHT90: Rotate CW 90 CLS_RT_ROTATE180: Rotate CW 180 CLS_RT_LEFT90: Rotate CCW 90
hRatio	[IN]	Horizontal ratio [%]	1 - 1000
vRatio	[IN]	Vertical ratio [%]	1 - 1000
point	[IN]	Font size	
style	[IN]	Font style	CLS_FNT_DEFAULT : None CLS_FNT_BOLD : Bold CLS_FNT_REVERSE : Reverse CLS_FNT_UNDERLINE : Underline CLS_FNT_ITALIC : Italic CLS_FNT_STRIKEOUT : Strikethrough * Use "|" to specify multiple options.
x	[IN]	X-coordinate	0000 – 9999 * The origin is at bottom-left.(0, 0)
y	[IN]	Y-coordinate	
resolution	[IN]	Resolution [dpi]	CLS_PRT_RES_203(203dpi) CLS_PRT_RES_300(300dpi) * 203 dpi by default.
measurementUnit	[IN]	Measurement unit	CLS_UNIT_MILLI CLS_UNIT_INCH * CLS_UNIT_INCH by default.

Description

Draws characters by using a font installed in the computer, with specifying options such as rotation, magnification, size and coordinates. What this method does internally is to generate a graphic image based on the given parameters, to store the graphic image into the printer and to print the stored graphic image. This doesn't check the file availability or validity. To get a precise printing result, make sure to select the resolution and the mesurement unit as same as the actual printer.

Return value

Returns CLS_SUCCESS(0) on success, an error code otherwise. See "2.1 Return value" for the error codes.

Example

```
// 203 dpi(default), inch(default)
design.drawTextLocalFont("drawTextLocalFont",
        fontName: "Arial Hebrew", rotation: CLS_RT_NORMAL,
        hRatio: 100, vRatio: 100, point: 10,
        style: CLS_FNT_BOLD | CLS_FNT_ITALIC, x: 10, y: 0)


// 300 dpi, inch(default)
design.drawTextLocalFont("drawTextLocalFont",
        fontName: "Arial Hebrew", rotation: CLS_RT_NORMAL,
        hRatio: 100, vRatio: 100, point: 10,
        style: CLS_FNT_DEFAULT, x: 10, y: 50,
        resolution: CLS_PRT_RES_300)


// 300 dpi, mm
design.drawTextLocalFont("drawTextLocalFont",
        fontName: "Arial Hebrew", rotation: CLS_RT_NORMAL,
        hRatio: 100, vRatio: 100, point: 10,
        style: CLS_FNT_DEFAULT, x: 10, y: 100,
        resolution: CLS_PRT_RES_300, measurementUnit: CLS_UNIT_MILLI)
```

2.3.5 drawNVBitmap method

Syntax

func drawNVBitmap(name: String!, hexp: Int32, vexp: Int32, x: Int32, y: Int32) -> Int32

Parameters

Parameter	[IN/OUT]	Description	Setting range
name	[IN]	Graphic file name	ASCII characters except below: -Underscore cannot be the first character. -The following symbols. \ / : * ? " < > |
hexp	[IN]	Horizontal magnification	1 - 24
vexp	[IN]	Vertical magnification	1 - 24
x	[IN]	X-coordinate	0000 - 9999 * The origin is at bottom-left.(0, 0)
y	[IN]	Y-coordinate	

Description

Draws a graphic by using a graphic image stored in the printer, with specifying options such as magnification and coordinates. This doesn't check the file availability or validity.

Return value

Returns CLS_SUCCESS(0) on success, an error code otherwise. See "2.1 Return value" for the error codes.

Example

See storeNVBitmap method.

2.3.6 drawBitmap method

Syntax

1) func drawBitmap(filePath: String!, rotation: Int32, width: Int32, height: Int32, x: Int32, y: Int32) -> Int32
2) func drawBitmap(filePath: String!, rotation: Int32, width: Int32, height: Int32, x: Int32, y: Int32 , resolution: Int32, measurementUnit: Int32) -> Int32

Parameters

Parameter	[IN/OUT]	Description	Setting range
filePath	[IN]	Path to a local graphic file	
rotation	[IN]	Direction of rotation	CLS_RT_NORMAL: No rotation CLS_RT_RIGHT90: Rotate CW 90 CLS_RT_ROTATE180: Rotate CW 180 CLS_RT_LEFT90: Rotate CCW 90
width	[IN]	Drawing width [Pixel]	
height	[IN]	Drawing height [Pixel]	
x	[IN]	X-coordinate	0000-9999 * The origin is at bottom-left.(0, 0)
y	[IN]	Y-coordinate	
resolution	[IN]	Resolution [dpi]	CLS_PRT_RES_203(203dpi) CLS_PRT_RES_300(300dpi) * 203 dpi by default.
measurementUnit	[IN]	Measurement Unit	CLS_UNIT_MILLI CLS_UNIT_INCH * CLS_UNIT_INCH by default.

Description

Draws a graphic image stored in the local computer, with specifying options such as rotation, size and coordinates.
What this method does internally is to store a graphic image into the printer and to print the stored graphic image. This doesn't check the availability or validity of the stored file. To get a precise printing result, make sure to select the resolution and the mesurement unit as same as the actual printer.
Supported graphic file formats are BMP/GIF/EXIF/JPG/PNG/TIFF.

The graphic will be resized based on the width/height parameters with keeping the aspect ratio and taking the maximum available size to fit.

Example: The sizes will be "200x50" in the case below.
Original sizes: 400x100 pixels
Width parameter: 200
Height parameter: 200

When 0 is set to either width or height, the sizes will be calculated based on another (non-zero) parameter. When both parameters are zero, the original sizes will be used.

Example : The sizes will be "800x200" in the case below.
Original sizes: 400x100 pixels
Width parameter: 0
Height parameter: 200

Return value

Returns CLS_SUCCESS(0) on success, an error code otherwise. See "2.1 Return value" for the error codes.

Example

```
// Retrieve files stored in resources
let FileName = NSBundle.mainBundle().pathForResource("sample_1.jpg",
        ofType: nil)

// 203 dpi(default), inch(default)
design.drawBitmap(FileName, rotation: CLS_RT_NORMAL,
        width: 0, height: 0, x: 10, y: 10)


// 300 dpi, mm
design.drawBitmap("Sample.bmp", rotation: CLS_RT_NORMAL,
        width: 0, height: 0, x: 10, y: 10,
        resolution: CLS_PRT_RES_300, measurementUnit: CLS_UNIT_MILLI)
```

2.3.7 drawBarCode method

Syntax

func drawBarCode(data: String!, symbology: Int32, rotation: Int32, thick: Int32, narrow: Int32, height: Int32, x: Int32, y: Int32, showText: Int32) -> Int32

Parameters

Parameter	[IN/OUT]	Description	Setting range
data	[IN]	String data	ASCII code character string.
symbology	[IN]	Barcode type	CLS_BCS_CODE39： Code 3 of 9 CLS_BCS_UPCA： UPC-A CLS_BCS_UPCE： UPC-E CLS_BCS_INTERLEAVED25： Interleaved 2 of 5 CLS_BCS_CODE128： Code 128 CLS_BCS_EAN13： EAN-13(JAN-13) CLS_BCS_EAN8： EAN-8(JAN-8) CLS_BCS_HIBC： HIBC CLS_BCS_CODABAR： CODABAR(NW-7) CLS_BCS_INT25： Int 2 of 5 CLS_BCS_PLESSEY： Plessey CLS_BCS_CASECODE： CASE CODE CLS_BCS_UPC2DIG： UPC 2DIG ADD CLS_BCS_UPC5DIG： UPC 5DIG ADD CLS_BCS_CODE93： Code93 CLS_BCS_ITF14： ITF-14 CLS_BCS_ZIP： ZIP CLS_BCS_ITF16： IFT-16 CLS_BCS_UCCEAN128： UCC/EAN-128 CLS_BCS_INDUSTRIAL25： Industrial 2 of 5 CLS_BCS_UCCEAN128KMART： UCC/EAN-128(for K-MART) CLS_BCS_COOP25： COOP 2 of 5 CLS_BCS_UCCEAN128RANDOMWEIGHT： UCC/EAN-128 Random Weight CLS_BCS_TELEPEN： Telepen
rotation	[IN]	Direction of rotation	CLS_RT_NORMAL: No rotation CLS_RT_RIGHT90: Rotate CW 90 CLS_RT_ROTATE180: Rotate CW 180 CLS_RT_LEFT90: Rotate CCW 90
thick	[IN]	Thick bar width	1-24
narrow	[IN]	Narrow bar width	1-24
height	[IN]	Height of barcode	001-999
x	[IN]	X-coordinate	0000-9999 * The origin is at bottom-left.(0, 0)
y	[IN]	Y-coordinate	
showText	[IN]	Human readable text	CLS_BCS_TEXT_HIDE： Hide CLS_BCS_TEXT_SHOW： Show

Description

Draws a 1D (one-dimensional) barcode with specifying options such as barcode type, rotation, thick/narrow bar width, coordinates and human readable text.

* Supported thick/narrow bar ratio or data type depends on symbology. Refer to the command reference for details.

Return value

Returns CLS_SUCCESS(0) on success, an error code otherwise. See "2.1 Return value" for the error codes.

Example

```
// Code3of9
var code39 = "ABC123456789"
design.drawBarCode(code39,
        symbology: CLS_BCS_CODE39, rotation: CLS_RT_NORMAL,
        thick: 6, narrow: 2, height: 50, x: 10, y: 10,
        showText: CLS_BCS_TEXT_SHOW)


// UPC-A
var upca = "01234567890"
design.drawBarCode(upca,
        symbology: CLS_BCS_UPCA, rotation: CLS_RT_NORMAL,
        thick: 2, narrow: 2, height: 50, x: 10, y: 100,
        showText: CLS_BCS_TEXT_SHOW)


// UPC-E
var upce = "123456"
design.drawBarCode(upce,
        symbology: CLS_BCS_UPCE, rotation: CLS_RT_NORMAL,
        thick: 2, narrow: 2, height: 50, x: 10, y: 100,
        showText: CLS_BCS_TEXT_SHOW)


// Interleaved2of5
var interleaved25 = "1234567890"
design.drawBarCode(interleaved25,
        symbology: CLS_BCS_INTERLEAVED25, rotation: CLS_RT_NORMAL,
        thick: 5, narrow: 2, height: 50, x: 10, y: 100,
        showText: CLS_BCS_TEXT_SHOW)


// Code128
var code128 = "1234567890"
design.drawBarCode(code128,
        symbology: CLS_BCS_CODE128, rotation: CLS_RT_NORMAL,
        thick: 2, narrow: 4, height: 50, x: 10, y: 10,
        showText: CLS_BCS_TEXT_SHOW)


// EAN-13
var ean13 = "123456789012"
design.drawBarCode(ean13,
        symbology: CLS_BCS_EAN13, rotation: CLS_RT_NORMAL,
        thick: 3, narrow: 3, height: 50, x: 10, y: 100,
        showText: CLS_BCS_TEXT_SHOW)
```

```
// EAN-8
var ean8 = "1234567"
design.drawBarCode(ean8,
        symbology: CLS_BCS_EAN8, rotation: CLS_RT_NORMAL,
        thick: 4, narrow: 4, height: 50, x: 10, y: 10,
        showText: CLS_BCS_TEXT_SHOW)


// HIBC
var hibc = "1234567890"
design.drawBarCode(hibc,
        symbology: CLS_BCS_HIBC, rotation: CLS_RT_NORMAL,
        thick: 6, narrow: 2, height: 50, x: 10, y: 100,
        showText: CLS_BCS_TEXT_SHOW)


// CODABAR
var codabar = "a1234567890b"
design.drawBarCode(codabar,
        symbology: CLS_BCS_CODABAR, rotation: CLS_RT_NORMAL,
        thick: 6, narrow: 2, height: 40, x: 10, y: 10,
        showText: CLS_BCS_TEXT_SHOW)


// Interleaved2of5 W/BARS
var int25 = "1234567890"
design.drawBarCode(int25,
        symbology: CLS_BCS_INT25, rotation: CLS_RT_NORMAL,
        thick: 5, narrow: 2, height: 50, x: 10, y: 100,
        showText: CLS_BCS_TEXT_SHOW)


// PLESSEY
var plessey = "1234567890"
design.drawBarCode(plessey,
        symbology: CLS_BCS_PLESSEY, rotation: CLS_RT_NORMAL,
        thick: 4, narrow: 2, height: 50, x: 10, y: 10,
        showText: CLS_BCS_TEXT_SHOW)


// CASE CODE
var casecode = "1234567890123"
design.drawBarCode(casecode,
        symbology: CLS_BCS_CASECODE, rotation: CLS_RT_NORMAL,
        thick: 5, narrow: 2, height: 50, x: 10, y: 100,
        showText: CLS_BCS_TEXT_SHOW)

// UPC 2DIG ADD
var upca = "01234567890"
var upc2dig = "12"

design.drawBarCode(upca,
        symbology: CLS_BCS_UPCA, rotation: CLS_RT_NORMAL,
        thick: 2, narrow: 2, height: 50, x: 10, y: 100,
        showText: CLS_BCS_TEXT_SHOW)

design.drawBarCode(upc2dig,
        symbology: CLS_BCS_UPC2DIG, rotation: CLS_RT_NORMAL,
        thick: 4, narrow: 2, height: 50, x: 130, y: 100,
        showText: CLS_BCS_TEXT_SHOW)
```

```
// UPC 5DIG ADD
var upca = "01234567890"
var upc5dig = "12345"

design.drawBarCode(upca,
        symbology: CLS_BCS_UPCA, rotation: CLS_RT_NORMAL,
        thick: 2, narrow: 2, height: 50, x: 10, y: 100,
        showText: CLS_BCS_TEXT_SHOW)

design.drawBarCode(upc5dig,
        symbology: CLS_BCS_UPC5DIG, rotation: CLS_RT_NORMAL,
        thick: 4, narrow: 2, height: 50, x: 130, y: 100,
        showText: CLS_BCS_TEXT_SHOW)


// Code93
var code93 = "1234ABCD"
design.drawBarCode(code93,
        symbology: CLS_BCS_CODE93, rotation: CLS_RT_NORMAL,
        thick: 6, narrow: 6, height: 50, x: 10, y: 10,
        showText: CLS_BCS_TEXT_SHOW)


// ITF-14
var itf14 = "1234567890123"
design.drawBarCode(itf14,
        symbology: CLS_BCS_ITF14, rotation: CLS_RT_NORMAL,
        thick: 5, narrow: 2, height: 50, x: 10, y: 100,
        showText: CLS_BCS_TEXT_SHOW)


// ZIP
var zip = "123456789"
design.drawBarCode(zip,
        symbology: CLS_BCS_ZIP, rotation: CLS_RT_NORMAL,
        thick: 1, narrow: 1, height: 10, x: 10, y: 10,
        showText: CLS_BCS_TEXT_SHOW)


// ITF-16
var itf16 = "123456789012345"
design.drawBarCode(itf16,
        symbology: CLS_BCS_ITF16, rotation: CLS_RT_NORMAL,
        thick: 5, narrow: 2, height: 50, x: 10, y: 100,
        showText: CLS_BCS_TEXT_SHOW)


// UCC/EAN-128
var uccean128 = "12345678901234567890"
design.drawBarCode(uccean128,
        symbology: CLS_BCS_UCCEAN128, rotation: CLS_RT_NORMAL,
        thick: 4, narrow: 4, height: 50, x: 10, y: 10,
        showText: CLS_BCS_TEXT_SHOW)
```

```
// Industrial2of5
var industrial25 = "1234567890"
design.drawBarCode(industrial25,
        symbology: CLS_BCS_INDUSTRIAL25, rotation: CLS_RT_NORMAL,
        thick: 5, narrow: 2, height: 50, x: 10, y: 100,
        showText: CLS_BCS_TEXT_SHOW)


// UCC/EAN-128(for K-MART)
var uccean128kmart = "12345678901234567 8"
design.drawBarCode(uccean128kmart,
        symbology: CLS_BCS_UCCEAN128KMART, rotation: CLS_RT_NORMAL,
        thick: 3, narrow: 3, height: 50, x: 10, y: 10,
        showText: CLS_BCS_TEXT_SHOW)


// COOP 2 of 5
String coop25 = "0123456789"
design.drawBarCode(coop25,
        symbology: CLS_BCS_COOP25, rotation: CLS_RT_NORMAL,
        thick: 10, narrow: 4, height: 50, x: 10, y: 100,
        showText: CLS_BCS_TEXT_SHOW)


// UCC/EAN-128 Random Weight
var uccean128randomweight = "12345678901234567890123456789099 99"
design.drawBarCode(uccean128randomweight,
        symbology: CLS_BCS_UCCEAN128RANDOMWEIGHT, rotation: CLS_RT_NORMAL,
        thick: 2, narrow: 2, height: 50, x: 10, y: 10,
        showText: CLS_BCS_TEXT_SHOW)


// Telepen
var telepen = "1234567890"
design.drawBarCode(telepen,
        symbology: CLS_BCS_TELEPEN, rotation: CLS_RT_NORMAL,
        thick: 2, narrow: 2, height: 50, x: 10, y: 100,
        showText: CLS_BCS_TEXT_SHOW)
```

2.3.8 drawMaxiCode method

Syntax

func drawMaxiCode(data: [Any]!, rotation: Int32, x: Int32, y: Int32) -> Int32

Parameters

Parameter	[IN/OUT]	Description	Setting range
data	[IN]	String data	ASCII code character string array
rotation	[IN]	Direction of rotation	CLS_RT_NORMAL: No rotation CLS_RT_RIGHT90: Rotate CW 90 CLS_RT_ROTATE180: Rotate CW 180 CLS_RT_LEFT90: Rotate CCW 90
x	[IN]	X-coordinate	0000–9999 * The origin is at bottom-left.(0, 0)
y	[IN]	Y-coordinate	

Description

Draws the UPS MaxiCode barcode with specifying options such as rotation and coordinates.
The following information are required in the data parameter in order as below:

1) 5-digit - Zip code
2) 4-digit -+4 Zip code
3) 3-digit – Country Code
4) 3-digit - Class of Service
5) Up to 84 digits other information

Return value

Returns CLS_SUCCESS(0) on success, an error code otherwise. See "2.1 Return value" for the error codes.

Example

```
let data: [String] = ["90501", "6282", "840", "001", "1Z00004951"]
design.drawMaxiCode(data, rotation: CLS_RT_NORMAL, x: 100, y: 0)
```

2.3.9 drawPDF417 method

Syntax

func drawPDF417(data: String!, encoding: Int32, rotation: Int32, exp: Int32, ecLevel: Int32, x: Int32, y: Int32) -> Int32

Parameters

Parameter	[IN/OUT]	Description	Setting range
data	[IN]	String data	* The characters available with the specified encoding.
encoding	[IN]	Character encoding	See "3.3.Predefined Constants."
rotation	[IN]	Direction of rotation	CLS_RT_NORMAL: No rotation CLS_RT_RIGHT90: Rotate CW 90 CLS_RT_ROTATE180: Rotate CW 180 CLS_RT_LEFT90: Rotate CCW 90
exp	[IN]	Magnification	1-5
ecLevel	[IN]	Error correction level	CLS_PDF417_EC_LEVEL_0 : Level 0 CLS_PDF417_EC_LEVEL_1 : Level 1 CLS_PDF417_EC_LEVEL_2 : Level 2 CLS_PDF417_EC_LEVEL_3 : Level 3 CLS_PDF417_EC_LEVEL_4 : Level 4 CLS_PDF417_EC_LEVEL_5 : Level 5 CLS_PDF417_EC_LEVEL_6 : Level 6 CLS_PDF417_EC_LEVEL_7 : Level 7 CLS_PDF417_EC_LEVEL_8 : Level 8
x	[IN]	X-coordinate	0000–9999 * The origin is at bottom-left.(0, 0)
y	[IN]	Y-coordinate	

Description

Draws the PDF417 barcode with specifying options such as character encoding, rotation, magnification, EC level and coordinates.

Return value

Returns CLS_SUCCESS(0) on success, an error code otherwise. See "2.1 Return value" for the error codes.

Example

```
design.drawPDF417("0123456789",
        encoding: CLS_ENC_CDPG_US_ASCII, rotation: CLS_RT_NORMAL,
        exp:3, ecLevel: CLS_PDF417_EC_LEVEL_0, x: 0, y: 0)
```

2.3.10 drawDataMatrix method

Syntax

func drawDataMatrix(data: String!, encoding: Int32, rotation: Int32, exp: Int32, ecLevel: Int32, x: Int32, y: Int32) -> Int32

Parameters

Parameter	[IN/OUT]	Description	Setting range
data	[IN]	String data	-Numeric: Up to 3,116 characters -Alphanumeric: Up to 2,335 characters * The characters available with the specified encoding.
encoding	[IN]	Character encoding	See #9 in "3.3.Predefined Constants."
rotation	[IN]	Direction of rotation	CLS_RT_NORMAL: No rotation CLS_RT_RIGHT90: Rotate CW 90 CLS_RT_ROTATE180: Rotate CW 180 CLS_RT_LEFT90: Rotate CCW 90
exp	[IN]	Magnification	1-15
ecLevel	[IN]	Error correction level	CLS_DATAMATRIX_EC_LEVEL_200 : 200
x	[IN]	X-coordinate	0000-9999 * The origin is at bottom-left.(0, 0)
y	[IN]	Y-coordinate	

Description

Draws the DataMatrix barcode with specifying options such as character encoding, rotation, magnification, EC level and coordinates.

Return value

Returns CLS_SUCCESS(0) on success, an error code otherwise. See "2.1 Return value" for the error codes.

Example

```
design.drawDataMatrix("0123456789",
        encoding: CLS_ENC_CDPG_US_ASCII, rotation: CLS_RT_NORMAL,
        exp:15, ecLevel: CLS_DATAMATRIX_EC_LEVEL_200, x: 0, y: 0)
```

2.3.11 drawQRCode method

Syntax

func drawQRCode(data: String!, encoding: Int32, rotation: Int32, exp: Int32, ecLevel: Int32, x: Int32, y: Int32) -> Int32

Parameters

Parameter	[IN/OUT]	Description	Setting range
data	[IN]	String data	* The characters available with the specified encoding.
encoding	[IN]	Character encoding	See #9 in "3.3.Predefined Constants."
rotation	[IN]	Direction of rotation	CLS_RT_NORMAL: No rotation CLS_RT_RIGHT90: Rotate CW 90 CLS_RT_ROTATE180: Rotate CW 180 CLS_RT_LEFT90: Rotate CCW 90
exp	[IN]	Magnification	1-15
ecLevel	[IN]	Error correction level	CLS_QRCODE_EC_LEVEL_L: Level L (7%) CLS_QRCODE_EC_LEVEL_M: Level M (15%) CLS_QRCODE_EC_LEVEL_Q: Level Q (25%) CLS_QRCODE_EC_LEVEL_H: Level H (30%)
x	[IN]	X-coordinate	0000-9999 * The origin is at bottom-left.(0, 0)
y	[IN]	Y-coordinate	

Description

Draws the QR code with specifying options such as character encoding, rotation, magnification, EC level and coordinates.

Return value

Returns CLS_SUCCESS(0) on success, an error code otherwise. See "2.1 Return value" for the error codes.

Example

```
design.drawQRCode("ｱｲｳｴｵ12345",
        encoding: CLS_ENC_CDPG_SHIFT_JIS, rotation: CLS_RT_NORMAL,
        exp:10, ecLevel: CLS_QRCODE_EC_LEVEL_H, x: 100, y: 100)
```

2.3.12 drawAztec method

Syntax

func drawAztec(data: String!, encoding: Int32, rotation: Int32, exp: Int32, ecLevel: Int32, x: Int32, y: Int32) -> Int32

Parameters

Parameter	[IN/OUT]	Description	Setting range
data	[IN]	String data	* The characters available with the specified encoding.
encoding	[IN]	Character encoding	See #9 in "3.3.Predefined Constants."
rotation	[IN]	Direction of rotation	CLS_RT_NORMAL: No rotation CLS_RT_RIGHT90: Rotate CW 90 CLS_RT_ROTATE180: Rotate CW 180 CLS_RT_LEFT90: Rotate CCW 90
exp	[IN]	Magnification	1-15
ecLevel	[IN]	Error correction level	CLS_AXTEC_EC_LEVEL_000: Level 0 (Error correction ratio 23%)
x	[IN]	X-coordinate	0000-9999 * The origin is at bottom-left.(0, 0)
y	[IN]	Y-coordinate	

Description

Draws the Aztec barcode with specifying options such as character encoding, rotation, magnification, EC level and coordinates.

Return value

Returns CLS_SUCCESS(0) on success, an error code otherwise. See "2.1 Return value" for the error codes.

Example

```
design.drawAztec("0123456789",
        encoding: CLS_ENC_CDPG_US_ASCII, rotation: CLS_RT_NORMAL,
        exp:10, ecLevel: CLS_AXTEC_EC_LEVEL_000, x: 0, y: 0)
```

2.3.13 drawGS1DataBar method

Syntax

func drawGS1DataBar(data: [Any]!, type _type: Int32, rotation: Int32, exp: Int32,
x: Int32, y: Int32) -> Int32

Parameters

Parameter	[IN/OUT]	Description	Setting range
data	[IN]	String data	ASCII code character string.
type	[IN]	Barcode type	CLS_GS1_DATABAR_OMNI_DIRECTIONAL: Omni-dirctional CLS_GS1_DATABAR_COMPOSITE: Conposite CLS_GS1_DATABAR_TRUNCATION: Truncation CLS_GS1_DATABAR_STACKED: Stacked CLS_GS1_DATABAR_STACKED_OMNI_DIRECTIONAL: Stacked Omni-dirctional CLS_GS1_DATABAR_LIMITED: Limited CLS_GS1_DATABAR_EXPANDED: Expanded
rotation	[IN]	Direction of rotation	CLS_RT_NORMAL: No rotation CLS_RT_RIGHT90: Rotate CW 90 CLS_RT_ROTATE180: Rotate CW 180 CLS_RT_LEFT90: Rotate CCW 90
exp	[IN]	Magnification	1-15
x	[IN]	X-coordinate	0000-9999 * The origin is at bottom-left.(0, 0)
y	[IN]	Y-coordinate	

Description

Draws the GS1DataBar barcode with specifying options such as rotation, magnification and coordinates.

* The available text types vary depending on the symbology. Refer to the "GS1 DataBar (RSS) in the command reference.

Return value

Returns CLS_SUCCESS(0) on success, an error code otherwise. See "2.1 Return value" for the error codes.

Example

```
// GS1 DataBar Omni-Directional
let omnidirectional: [String] = ["1234567890123"]
design.drawGS1DataBar(omnidirectional,
        type: CLS_GS1_DATABAR_OMNI_DIRECTIONAL,
        rotation: CLS_RT_NORMAL, exp: 3, x: 10, y: 10)


// GS1 DataBar Composite
let composit: [String] = ["1234567890123", "1234567890-07/07/07"]
design.drawGS1DataBar(composit,
        type: CLS_GS1_DATABAR_COMPOSITE,
        rotation: CLS_RT_NORMAL, exp: 3, x: 10, y: 110)
```

```
// GS1 DataBar Truncation
let truncation: [String] = ["1234567890123"]
design.drawGS1DataBar(truncation,
        type: CLS_GS1_DATABAR_TRUNCATION,
        rotation: CLS_RT_NORMAL, exp: 3, x: 10, y: 210)


// GS1 DataBar Stacked
let stacked: [String] = ["1234567890123"]
design.drawGS1DataBar(stacked,
        type: CLS_GS1_DATABAR_STACKED,
        rotation: CLS_RT_NORMAL, exp: 3, x: 10, y: 310)


// GS1 DataBar Stacked-Omni-Directional
let stackedOD: [String] = ["1234567890123"]
design.drawGS1DataBar(stackedOD,
        type: CLS_GS1_DATABAR_STACKED_OMNI_DIRECTIONAL,
        rotation: CLS_RT_NORMAL, exp: 3, x: 10, y: 410)


// GS1 DataBar Limited
let limited: [String] = ["1234567890123"]
design.drawGS1DataBar(limited,
        type: CLS_GS1_DATABAR_LIMITED,
        rotation: CLS_RT_NORMAL, exp: 3, x: 10, y: 610)


// GS1 DataBar Expanded
let expanded: [String] = ["041234567890123"]
design.drawGS1DataBar(expanded,
        type: CLS_GS1_DATABAR_EXPANDED,
        rotation: CLS_RT_NORMAL, exp: 3, x: 10, y: 710)
```

2.3.14 drawLine method

Syntax

func drawLine(x1: Int32, y1: Int32, x2: Int32, y2: Int32, thickness: Int32) -> Int32

Parameters

Parameter	[IN/OUT]	Description	Setting range
x1	[IN]	Start position (X-coordinate)	0000-9999 * The origin is at bottom-left.(0, 0)
y1	[IN]	Start position (Y-coordinate)	
x2	[IN]	End position (X-coordinate)	
y2	[IN]	End position (Y-coordinate)	
thickness	[IN]	Line width (Reference point is center)	0000-9999

Description

Draws a line of the specified width.
Start and stop positions will be located at the center of the line. Each coordinate must be within the range of 0000-9999. Moreover the whole line image (rectangule) including the line thickness must be within the range of 0000-9999 as well. Returns CLS_E_ILLEGAL(1101) otherwise.

Return value

Returns CLS_SUCCESS(0) on success, an error code otherwise. See "2.1 Return value" for the error codes.

Example

```
// FormatAttribute:OR
printer!.setFormatAttribute(1)

// Bar
design.drawLine (20, y1: 30, x2: 20, y2: 300, thickness: 10)

// Horizontal line
design.drawLine (16, y1: 34, x2: 200, y2: 34, thickness: 10)
```

2.3.15 drawRect method

Syntax

func drawRect(x: Int32, y: Int32, width: Int32, height: Int32, thickness: Int32) -> Int32

Parameters

Parameter	[IN/OUT]	Description	Setting range
x	[IN]	Start position (X-coordinate)	0000 - 9999 * The origin is at bottom-left.(0, 0)
y	[IN]	Start position (Y-coordinate)	
width	[IN]	Box width	0000 - 9999
height	[IN]	Box height	
thickness	[IN]	Line width	0000 - 9999

Description

Draws a box of the specified width, height and line width. Thicker line width expands the line inward and doesn't affect the outline size.

Return value

Returns CLS_SUCCESS(0) on success, an error code otherwise. See "2.1 Return value" for the error codes.

Example

```
design.drawRect(20, y: 30, width: 180, height: 280, thickness: 10)
```

2.3.16 fillRect method

Syntax

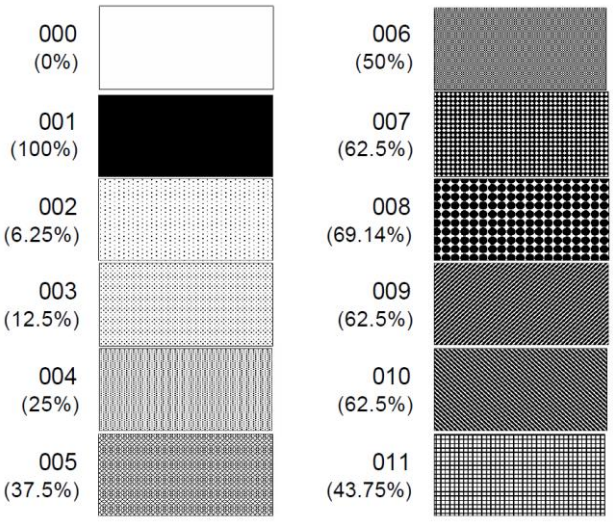
func fillRect(x: Int32, y: Int32, width: Int32, height: Int32, pattern: Int32) -> Int32

Parameters

Parameter	[IN/OUT]	Description	Setting range
x	[IN]	Start position (X-coordinate)	0000 - 9999 * The origin is at bottom-left.(0, 0)
y	[IN]	Start position (Y-coordinate)	
width	[IN]	Box width	0000 - 9999
height	[IN]	Box height	
pattern	[IN]	Shaded pattern	CLS_SHADED_PTN_0 : Shaded pattern 000 CLS_SHADED_PTN_1 : Shaded pattern 001 CLS_SHADED_PTN_2 : Shaded pattern 002 CLS_SHADED_PTN_3 : Shaded pattern 003 CLS_SHADED_PTN_4 : Shaded pattern 004 CLS_SHADED_PTN_5 : Shaded pattern 005 CLS_SHADED_PTN_6 : Shaded pattern 006 CLS_SHADED_PTN_7 : Shaded pattern 007 CLS_SHADED_PTN_8 : Shaded pattern 008 CLS_SHADED_PTN_9 : Shaded pattern 009 CLS_SHADED_PTN_10 : Shaded pattern 010 CLS_SHADED_PTN_11 : Shaded pattern 011

Description

Draws a box of the specified width, height and shaded pattern.



Return value

Returns CLS_SUCCESS(0) on success, an error code otherwise. See "2.1 Return value" for the error codes.

Example

```
design.fillRect(20, y: 30, width: 180, height: 280,
        pattern: CLS_SHADED_PTN_10)
```

2.3.17 drawCircle method

Syntax

func drawCircle(x: Int32, y: Int32, radius: Int32) -> Int32

Parameters

Parameter	[IN/OUT]	Description	Setting range
x	[IN]	Start position (X-coordinate, center)	0000 - 9999 * The origin is at bottom-left.(0, 0)
y	[IN]	Start position (Y-coordinate, center)	
radius	[IN]	Radius	0000 - 0398

Description

Draws a circle of the specified radius.

Return value

Returns CLS_SUCCESS(0) on success, an error code otherwise. See "2.1 Return value" for the error codes.

Example

```
design.drawCircle(50, y: 50, radius: 15)
```

2.3.18 fillCircle method

Syntax

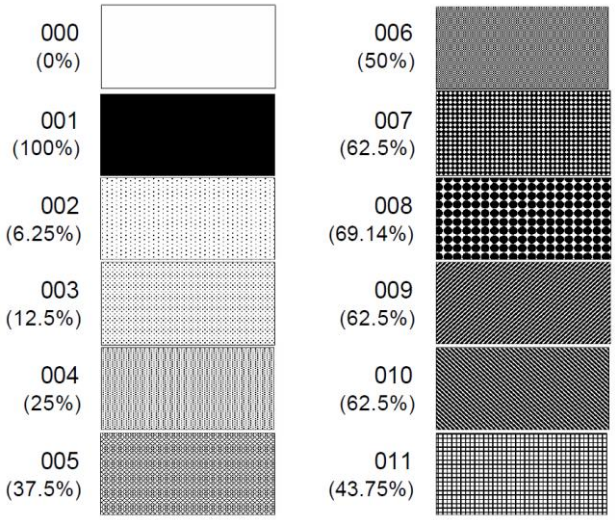
func fillCircle(x: Int32, y: Int32, radius: Int32, pattern: Int32) -> Int32

Parameters

Parameter	[IN/OUT]	Description	Setting range
x	[IN]	Start position (X-coordinate, center)	0000 - 9999 * The origin is at bottom-left.(0, 0)
y	[IN]	Start position (Y-coordinate, center)	
radius	[IN]	Radius	0000 - 0398
pattern	[IN]	Shaded pattern	CLS_SHADED_PTN_0 : Shaded pattern 000 CLS_SHADED_PTN_1 : Shaded pattern 001 CLS_SHADED_PTN_2 : Shaded pattern 002 CLS_SHADED_PTN_3 : Shaded pattern 003 CLS_SHADED_PTN_4 : Shaded pattern 004 CLS_SHADED_PTN_5 : Shaded pattern 005 CLS_SHADED_PTN_6 : Shaded pattern 006 CLS_SHADED_PTN_7 : Shaded pattern 007 CLS_SHADED_PTN_8 : Shaded pattern 008 CLS_SHADED_PTN_9 : Shaded pattern 009 CLS_SHADED_PTN_10 : Shaded pattern 010 CLS_SHADED_PTN_11 : Shaded pattern 011

Description

Draws a circle of the specified radius and shaded pattern.



Return value

Returns CLS_SUCCESS(0) on success, an error code otherwise. See "2.1 Return value" for the error codes.

Example

```
design.fillCircle(100, y: 100, radius: 40, pattern: CLS_SHADED_PTN_2)
```

2.3.19 drawPolygon method

Syntax

func drawPolygon(x: Array<Any>!, y: Array<Any>!) -> Int32

Parameters

Parameter	[IN/OUT]	Description	Setting range
x	[IN]	X points	0000 - 9999 * The origin is at bottom-left.(0, 0)
y	[IN]	Y points	

Description

Draws a closed polygon defined by arrays of x and y coordinates. Each pair of (x, y) coordinates defines a point. The number of elements must match. Returns CLS_E_ILLEGAL(1101) otherwise.

Return value

Returns CLS_SUCCESS(0) on success, an error code otherwise. See "2.1 Return value" for the error codes.

Example

```
let x: [Any] = [NSNumber(value: 100), NSNumber(value: 200),
        NSNumber(value: 250), NSNumber(value: 150)]
let y: [Any] = [NSNumber(value: 100), NSNumber(value: 100),
        NSNumber(value: 200), NSNumber(value: 200)]
design.drawPolygon(x, y: y)      // Draw a parallelogram
```

2.3.20 fillPolygon method

Syntax

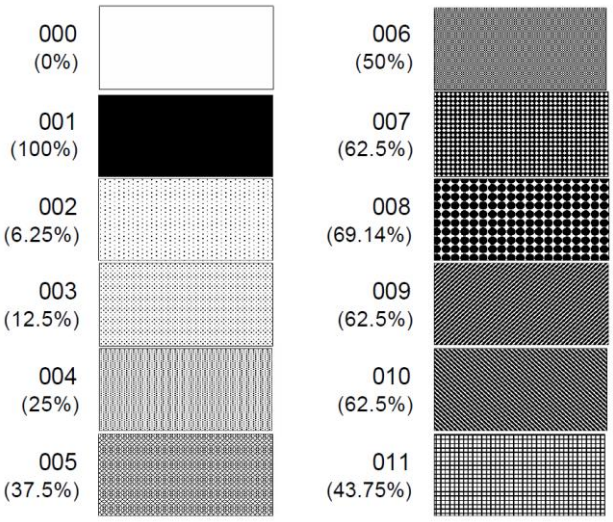
func fillPolygon(x: Array<Any>!, y: Array<Any>!, pattern: Int32) -> Int32

Parameters

Parameter	[IN/OUT]	Description	Setting range
x	[IN]	X points	0000 - 9999 * The origin is at bottom-left.(0, 0)
y	[IN]	Y points	
pattern	[IN]	Shaded pattern	CLS_SHADED_PTN_0 : Shaded pattern 000 CLS_SHADED_PTN_1 : Shaded pattern 001 CLS_SHADED_PTN_2 : Shaded pattern 002 CLS_SHADED_PTN_3 : Shaded pattern 003 CLS_SHADED_PTN_4 : Shaded pattern 004 CLS_SHADED_PTN_5 : Shaded pattern 005 CLS_SHADED_PTN_6 : Shaded pattern 006 CLS_SHADED_PTN_7 : Shaded pattern 007 CLS_SHADED_PTN_8 : Shaded pattern 008 CLS_SHADED_PTN_9 : Shaded pattern 009 CLS_SHADED_PTN_10 : Shaded pattern 010 CLS_SHADED_PTN_11 : Shaded pattern 011

Description

Draws a closed polygon defined by arrays of x and y coordinates, with the specified shaded pattern. Each pair of (x, y) coordinates defines a point. The number of elements must match. Returns CLS_E_ILLEGAL(1101) otherwise.



Return value

Returns CLS_SUCCESS(0) on success, an error code otherwise. See "2.1 Return value" for the error codes.

Example

```
let x: [Any] = [NSNumber(value: 100), NSNumber(value: 200),
        NSNumber(value: 250), NSNumber(value: 150)]
let y: [Any] = [NSNumber(value: 100), NSNumber(value: 100),
        NSNumber(value: 200), NSNumber(value: 200)]
design.fillPolygon(x, y: y, pattern: CLS_SHADED_PTN_3)
// Draw a parallelogram
```

2.3.21 embedRawDesignCommand method

Syntax

func embedRawDesignCommand(data: UnsafeMutablePointer<Int8>,
withLength length: UInt) -> Int32

Parameters

Parameter	[IN/OUT]	Description	Setting range
data	[IN]	Design command	Data of binary.
length	[IN]	Length of send data	

Description

Embeds raw printer commands to the label design. This allows adding raw label format commands individually.

The SendData method in the LabelPrinter class requires complete label commands (both system level commands and label format commands).

Return value

Returns CLS_SUCCESS(0) on success, an error code otherwise. See "2.1 Return value" for the error codes.

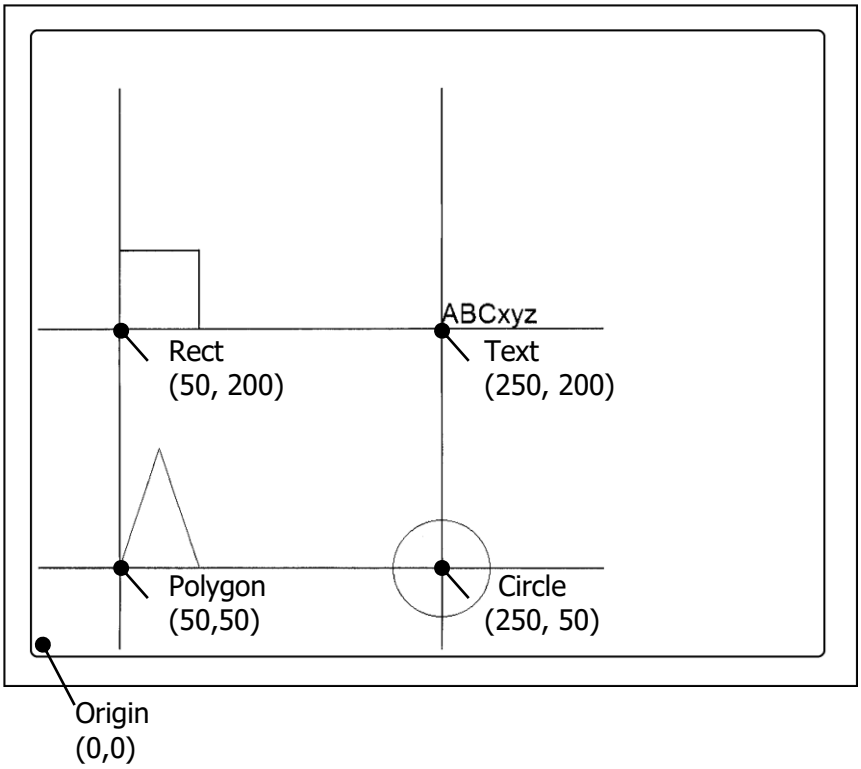
Example

```
To add a box command,
1X1100002000100b0300025001000100:

var hexdata: [Int8] =  [
        0x31, 0x58, 0x31, 0x31, 0x30, 0x30, 0x30, 0x30, 0x32, 0x30,
        0x30, 0x30, 0x31, 0x30, 0x30, 0x62, 0x30, 0x33, 0x30, 0x30,
        0x30, 0x32, 0x35, 0x30, 0x30, 0x31, 0x30, 0x30, 0x30, 0x31,
        0x30, 0x30, 0x0D, 0x0A ]

design.embedRawDesignCommand(&hexdata, withLength: 34)
```

3. Appendix

3.1. Specifying object position

The coordinate origin of the label design is at bottom-left. The measurement unit, either inches or millimeters, is set in the printer. The MeasurementUnit property in the LabelPrinter class allows setting the measurement unit.



Example

```
// Polygon
let x: [Any] = [NSNumber(value: 50), NSNumber(value: 75),
        NSNumber(value: 100)]
let y: [Any] = [NSNumber(value: 50), NSNumber(value: 125),
        NSNumber(value: 50)]
design.drawPolygon(x, y: y)

// Rect
design.drawRect(50, y: 200, width: 50, height: 50, thickness: 1)

// Circle
design.drawCircle(250, y: 50, radius: 30)

// Text
design.drawTextLocalFont("ABCxyz",
        fontName: "Arial Hebrew", rotation: CLS_RT_NORMAL,
        hRatio: 100, vRatio: 100, point: 12,
        style: CLS_FNT_DEFAULT, x: 250, y: 200)

// Line
design.drawLine(0,   y1: 200, x2: 350, y2: 200, thickness: 1)    // Line-1
design.drawLine(50,  y1: 0,   x2: 50,  y2: 350, thickness: 1)    // Line-2
design.drawLine(0,   y1: 50,  x2: 350, y2: 50,  thickness: 1)    // Line-3
design.drawLine(250, y1: 0,   x2: 250, y2: 350, thickness: 1)    // Line-4
```

3.2. Logging function

This SDK has a logging function which keeps history of executing methods or reading/writing properties. This can be enabled by using the setLog method, or placing a file "CSJLabelLib.cfg" in the same folder as the SDK.

< Example of CSJLabelLib.cfg >

```
[LogSetting]        ・・・Section name (Fixed)
LogMode=1           ・・・Specifies the log mode.
LogPath=/           ・・・Specifies the folder in the application sharing folder to store the log files.
LogMaxSize=10       ・・・Specifies the maximum size of log file in MB.
```

Setting items

- LogMode
  Specifies a log mode:
  0: None
  1: Access log
  2: Error log

- LogPath
  Specifies a folder to store the log files. "/" of the application sharing folder by default.

- LogMaxSize
  Specifies maximum size to log file in MB. "0" sets the size to "unlimited."

Log file name

Log files will be stored with a file name "CSJLabelLib_" and a number which indicates the day of week(0 to 6. 0: Sunday, 1: Monday...), and a file extension ".log."

Example: CSJLabelLib_1.log

When the same file name exists, the file will be overwritten if the file is one week older, new logs will be added to the existing file if the file is created on the same day.

Log format

The log file keeps the information of executed methods, accessed properties, timestamps and results.

```
--- Example 1, method (Connect) ---

2018/07/24 11:08:07.850 001 METHOD call   connect(1, 00:01:90:DF:CB:6D, 9100, 0)
2018/07/24 11:08:08.567 001 METHOD result connect() -> Success(0)


--- Example 2, method (PrintText) ---

-----------------Parameter Detail----------------------
drawTextPtrFont("Sample Print", 0, 10, 1, 1, 1, 8, 20, 300) -> 0
drawQRCode("DrawQRCode", 850, 1, 4, 3, 20, 220) -> 0
fillRect(20, 150, 350, 40, 11) -> 0
drawBarCode("0123456789", 104, 1, 3, 3, 30, 20, 70, 1) -> 0
-------------------------------------------------------
2018/07/24 11:08:25.519 001 METHOD result print() -> Success(0)


--- Example, set to a property ---

2018/07/24 11:08:29.716 001 PROPERTY set  PrintDarkness <- 10 : Success(0)
```

```
--- Example, get from a property ---

2018/07/24 11:08:29.688 001 PROPERTY get  PrintDarkness -> 999999
```

* The logging function could be a "bottleneck" of processing since it tries to keep the information of every single execution of a method or access to a property.

* Logging could fail without a notification for the reason below.
  - A write-protected device is selected as a storage location.
  - No enough space in the selected storage device.
  - The storage location contains a file with write-protection.
  - No access privilege to a file or folder.
  - Another program is using the log file.

3.3. Predefined Constants

No	Type	Name	Data Type	Value	Description
1	Result/Error	CLS_SUCCESS	int	0	Successfully completed
		CLS_E_CONNECTED	int	1001	Already connected
		CLS_E_DISCONNECT	int	1002	Not connected
		CLS_E_NOTCONNECT	int	1003	Failed to connect
		CLS_E_CONNECT_NOTFOUND	int	1004	Non supported model
		CLS_E_ILLEGAL	int	1101	Unsupported or invalid parameter
		CLS_E_OFFLINE	int	1102	Off-line
		CLS_E_NOEXIST	int	1103	File does not exist
		CLS_E_FAILURE	int	1104	Process failure
		CLS_E_TIMEOUT	int	1105	Timeout
		CLS_E_NO_LIST	int	1106	Printer cannot be found
		CLS_EPTR_BADFORMAT	int	1203	Unsupported file format
2	Property default value	CLS_PROPERTY_DEFAULT	int	999999	Default value of property
3	Status of printer	CLS_STS_NO	int	0	Status: NO
		CLS_STS_YES	int	1	Status: YES
4	Connection interface	CLS_PORT_WiFi	int	0	Network
		CLS_PORT_Bluetooth	int	1	Bluetooth
		CLS_PORT_USB	int	3	USB
5	Serial communication condition	CLS_COM_BAUDRATE_1200	int	1200	Baud rate : 1200
		CLS_COM_BAUDRATE_2400	int	2400	Baud rate : 2400
		CLS_COM_BAUDRATE_4800	int	4800	Baud rate : 4800
		CLS_COM_BAUDRATE_9600	int	9600	Baud rate : 9600
		CLS_COM_BAUDRATE_19200	int	19200	Baud rate : 19200
		CLS_COM_BAUDRATE_38400	int	38400	Baud rate : 38400
		CLS_COM_BAUDRATE_57600	int	57600	Baud rate : 57600
		CLS_COM_BAUDRATE_115200	int	115200	Baud rate : 115200
		CLS_COM_PARITY_NONE	int	0	Parity : NONE
		CLS_COM_PARITY_ODD	int	1	Parity : ODD
		CLS_COM_PARITY_EVEN	int	2	Parity : EVEN
		CLS_COM_HANDSHAKE_DTRDSR	int	0	Flow control : DTR/DSR
		CLS_COM_HANDSHAKE_XONXOFF	int	1	Flow control : XON/XOFF
6	Locale	CLS_LOCALE_JP	int	0	Japanese model
		CLS_LOCALE_OTHER	int	1	Non-Japanese model
		CLS_LOCALE_CN	int	2	Chinese model
		CLS_LOCALE_KR	int	3	Korean model
7	Font typeface	CLS_PRT_FNT_0	int	0	System font : 0
		CLS_PRT_FNT_1	int	1	System font : 1
		CLS_PRT_FNT_2	int	2	System font : 2
		CLS_PRT_FNT_3	int	3	System font : 3
		CLS_PRT_FNT_4	int	4	System font : 4
		CLS_PRT_FNT_5	int	5	System font : 5
		CLS_PRT_FNT_6	int	6	System font : 6
		CLS_PRT_FNT_7	int	7	System font : 7
		CLS_PRT_FNT_8	int	8	System font : 8
		CLS_PRT_FNT_TRIUMVIRATE	int	9	Smooth font (Triumvirate)

		CLS_PRT_FNT_TRIUMVIRATE_B	int	10	Smooth font (Triumvirate Bold)
		CLS_PRT_FNT_KANJI	int	11	Kanji (Writing from left to right)
		CLS_PRT_FNT_KANJIT	int	12	Kanji (Writing from top to bottom)
8	Font size	CLS_PRT_FNT_SIZE_4	int	0	4pt
		CLS_PRT_FNT_SIZE_5	int	1	5pt
		CLS_PRT_FNT_SIZE_6	int	2	6pt
		CLS_PRT_FNT_SIZE_8	int	3	8pt
		CLS_PRT_FNT_SIZE_10	int	4	10pt
		CLS_PRT_FNT_SIZE_12	int	5	12pt
		CLS_PRT_FNT_SIZE_14	int	6	14pt
		CLS_PRT_FNT_SIZE_18	int	7	18pt
		CLS_PRT_FNT_SIZE_24	int	8	24pt
		CLS_PRT_FNT_SIZE_30	int	9	30pt
		CLS_PRT_FNT_SIZE_36	int	10	36pt
		CLS_PRT_FNT_SIZE_48	int	11	48pt
		CLS_PRT_FNT_KANJI_SIZE_16	int	100	Kanji 16dot
		CLS_PRT_FNT_KANJI_SIZE_24	int	101	Kanji 24dot
		CLS_PRT_FNT_KANJI_SIZE_32	int	102	Kanji 32dot
		CLS_PRT_FNT_KANJI_SIZE_48	int	103	Kanji 48dot
9	Encoding	CLS_ENC_CDPG_DEFAULT	int	0	Default value
		CLS_ENC_CDPG_IBM037	int	37	IBM037 IBM EBCDIC (USA-Canada)
		CLS_ENC_CDPG_IBM437	int	437	IBM437 OEM USA
		CLS_ENC_CDPG_IBM500	int	500	IBM500 IBM EBCDIC (International)
		CLS_ENC_CDPG_IBM737	int	737	ibm737 Greek (DOS)
		CLS_ENC_CDPG_IBM775	int	775	ibm775 Baltic (DOS)
		CLS_ENC_CDPG_IBM850	int	850	ibm850 Western European(DOS)
		CLS_ENC_CDPG_IBM852	int	852	ibm852 Central Europe (DOS)
		CLS_ENC_CDPG_IBM855	int	855	IBM855 OEM Cyrillic
		CLS_ENC_CDPG_IBM857	int	857	ibm857 Turkish (DOS)
		CLS_ENC_CDPG_IBM860	int	860	IBM860 Portuguese (DOS)
		CLS_ENC_CDPG_IBM861	int	861	ibm861 Icelandic (DOS)
		CLS_ENC_CDPG_IBM863	int	863	IBM863 French (Canada)(DOS)
		CLS_ENC_CDPG_IBM865	int	865	IBM865 Norwegian (DOS)
		CLS_ENC_CDPG_CP866	int	866	cp866 Cyrillic (DOS)
		CLS_ENC_CDPG_IBM869	int	869	ibm869 Greek modern (DOS)
		CLS_ENC_CDPG_WINDOWS_874	int	874	windows-874 Thai (Windows)
		CLS_ENC_CDPG_CP875	int	875	cp875 IBM EBCDIC (Greek modern)
		CLS_ENC_CDPG_SHIFT_JIS	int	932	shift_jis Japanese (Shift JIS)
		CLS_ENC_CDPG_GB2312	int	936	gb2312 simplified Chinese (GB2312)
		CLS_ENC_CDPG_KS_C_5601_1987	int	949	ks_c_5601-1987 Korean
		CLS_ENC_CDPG_BIG5	int	950	big5 traditional Chinese (Big5)
		CLS_ENC_CDPG_IBM1026	int	1026	IBM1026 IBM EBCDIC (Turkish Latin-5)
		CLS_ENC_CDPG_UTF_16	int	1200	utf-16 Unicode
		CLS_ENC_CDPG_UNICODEFFFE	int	1201	unicodeFFFE Unicode (Big-Endian)
		CLS_ENC_CDPG_WINDOWS_1250	int	1250	windows-1250 Central European (Windows)
		CLS_ENC_CDPG_WINDOWS_1251	int	1251	windows-1251 Cyrillic (Windows)
		CLS_ENC_CDPG_WINDOWS_1252	int	1252	Windows-1252Western European (Windows)
		CLS_ENC_CDPG_WINDOWS_1253	int	1253	windows-1253 Greek (Windows)

CLS_ENC_CDPG_WINDOWS_1254	int	1254	windows-1254 Turkish (Windows)
CLS_ENC_CDPG_WINDOWS_1255	int	1255	windows-1255 Hebrew (Windows)
CLS_ENC_CDPG_WINDOWS_1256	int	1256	windows-1256 Arabic (Windows)
CLS_ENC_CDPG_WINDOWS_1257	int	1257	windows-1257 Baltic (Windows)
CLS_ENC_CDPG_WINDOWS_1258	int	1258	windows-1258 Vietnamese(Windows)
CLS_ENC_CDPG_JOHAB	int	1361	Johab Korean (Johab)
CLS_ENC_CDPG_MACINTOSH	int	10000	macintosh Western European (Mac)
CLS_ENC_CDPG_X_MAC_JAPANESE	int	10001	x-mac-japanese Japanese (Mac)
CLS_ENC_CDPG_X_MAC_CHINESETRAD	int	10002	x-mac-chinesetrad traditional Chinese (Mac)
CLS_ENC_CDPG_X_MAC_KOREAN	int	10003	x-mac-korean Korean (Mac)
CLS_ENC_CDPG_X_MAC_GREEK	int	10006	x-mac-greek Greek (Mac)
CLS_ENC_CDPG_X_MAC_CYRILLIC	int	10007	x-mac-cyrillic Cyrillic (Mac)
CLS_ENC_CDPG_X_MAC_CHINESESIMP	int	10008	x-mac-chinesesimp simplified Chinese (Mac)
CLS_ENC_CDPG_X_MAC_ROMANIAN	int	10010	x-mac-romanian Romanian (Mac)
CLS_ENC_CDPG_X_MAC_UKRAINIAN	int	10017	x-mac-ukrainian Ukrainian (Mac)
CLS_ENC_CDPG_X_MAC_CE	int	10029	x-mac-ce Central Europe (Mac)
CLS_ENC_CDPG_X_MAC_ICELANDIC	int	10079	x-mac-icelandic Icelandic (Mac)
CLS_ENC_CDPG_X_MAC_TURKISH	int	10081	x-mac-turkish Turkish (Mac)
CLS_ENC_CDPG_X_MAC_CROATIAN	int	10082	x-mac-croatian Croat (Mac)
CLS_ENC_CDPG_X_CHINESE_CNS	int	20000	x-Chinese-CNS traditional Chinese (CNS)
CLS_ENC_CDPG_US_ASCII	int	20127	us-ascii US-ASCII
CLS_ENC_CDPG_X_CP20261	int	20261	x-cp20261 T.61
CLS_ENC_CDPG_IBM290	int	20290	IBM290 IBM EBCDIC (Japanese Katakana)
CLS_ENC_CDPG_KOI8_R	int	20866	koi8-r Cyrillic (KOI8-R)
CLS_ENC_CDPG_EUC_JP_JIS	int	20932	EUC-JP Japanese (JIS 0208-1990 and 0212-1990)
CLS_ENC_CDPG_X_CP20936	int	20936	x-cp20936 simplified Chinese (GB2312-80)
CLS_ENC_CDPG_X_CP20949	int	20949	x-cp20949 Korean Wansung
CLS_ENC_CDPG_X_CP21027	int	21027	x-cp21027 Ext Alpha Lowercase
CLS_ENC_CDPG_KOI8_U	int	21866	koi8-u Cyrillic (KOI8-R)
CLS_ENC_CDPG_ISO_8859_1	int	28591	iso-8859-1 Western European (ISO)
CLS_ENC_CDPG_ISO_8859_2	int	28592	iso-8859-2 Central Europe (ISO)
CLS_ENC_CDPG_ISO_8859_4	int	28594	iso-8859-4 Baltic (ISO)
CLS_ENC_CDPG_ISO_8859_5	int	28595	iso-8859-5 Cyrillic (ISO)
CLS_ENC_CDPG_ISO_8859_7	int	28597	iso-8859-7 Greek (ISO)
CLS_ENC_CDPG_ISO_8859_9	int	28599	iso-8859-9 Turkish (ISO)
CLS_ENC_CDPG_ISO_8859_13	int	28603	iso-8859-13 Estonian (ISO)
CLS_ENC_CDPG_ISO_8859_15	int	28605	iso-8859-15 Latin 9 (ISO)
CLS_ENC_CDPG_ISO_2022_JP	int	50220	iso-2022-jp Japanese (JIS)
CLS_ENC_CDPG_CSISO2022JP	int	50221	csISO2022JP Japanese (JIS-Allow 1 byte Kana)
CLS_ENC_CDPG_ISO_2022_JP_S	int	50222	iso-2022-jp Japanese (JIS-Allow 1 byte Kana - SO/SI)
CLS_ENC_CDPG_ISO_2022_KR	int	50225	iso-2022-kr Korean (ISO)
CLS_ENC_CDPG_X_CP50227	int	50227	x-cp50227 simplified Chinese (ISO-2022)

		CLS_ENC_CDPG_EUC_JP	int	51932	euc-jp Japanese (EUC)
		CLS_ENC_CDPG_EUC_CN	int	51936	EUC-CN simplified Chinese (EUC)
		CLS_ENC_CDPG_EUC_KR	int	51949	euc-kr Korean (EUC)
		CLS_ENC_CDPG_HZ_GB_2312	int	52936	hz-gb-2312 simplified Chinese (HZ)
		CLS_ENC_CDPG_GB18030	int	54936	GB18030 simplified Chinese (GB18030)
		CLS_ENC_CDPG_UTF_7	int	65000	utf-7 Unicode (UTF-7)
		CLS_ENC_CDPG_UTF_8	int	65001	utf-8 Unicode (UTF-8)
10	Font style	CLS_FNT_DEFAULT	int	0	None
		CLS_FNT_BOLD	int	8	Bold
		CLS_FNT_REVERSE	int	16	Reverse
		CLS_FNT_UNDERLINE	int	128	Underline
		CLS_FNT_ITALIC	int	256	Italic
		CLS_FNT_STRIKEOUT	int	512	Strikethrough
11	Direction of rotation	CLS_RT_NORMAL	int	1	No rotation
		CLS_RT_RIGHT90	int	2	Rotate 90 CW
		CLS_RT_ROTATE180	int	3	Rotate 180
		CLS_RT_LEFT90	int	4	Rotate 90 CCW
12	Barcode symbology	CLS_BCS_CODE39	int	100	Code 3 of 9
		CLS_BCS_UPCA	int	101	UPC-A
		CLS_BCS_UPCE	int	102	UPC-E
		CLS_BCS_INTERLEAVED25	int	103	Interleaved 2 of 5
		CLS_BCS_CODE128	int	104	Code 128
		CLS_BCS_EAN13	int	105	EAN-13(JAN-13)
		CLS_BCS_EAN8	int	106	EAN-8(JAN-8)
		CLS_BCS_HIBC	int	107	HIBC
		CLS_BCS_CODABAR	int	108	CODABAR(NW-7)
		CLS_BCS_INT25	int	109	Int 2 of 5
		CLS_BCS_PLESSEY	int	110	Plessey
		CLS_BCS_CASECODE	int	111	CASE CODE
		CLS_BCS_UPC2DIG	int	112	UPC 2DIG ADD (supplementary code of 2 digits for UPC)
		CLS_BCS_UPC5DIG	int	113	UPC 5DIG ADD (supplementary code of 5 digits for UPC)
		CLS_BCS_CODE93	int	114	Code93
		CLS_BCS_ITF14	int	115	(Japanese model)ITF-14
		CLS_BCS_ZIP	int	116	(Non-Japanese model)ZIP
		CLS_BCS_ITF16	int	117	(Japanese model)ITF-16
		CLS_BCS_UCCEAN128	int	118	(Non-Japanese model)UCC/EAN-128
		CLS_BCS_INDUSTRIAL25	int	119	(Japanese model)Industrial 2 of 5
		CLS_BCS_UCCEAN128KMART	int	120	(Non-Japanese model) UCC/EAN-128(for K-MART)
		CLS_BCS_COOP25	int	121	(Japanese model)COOP 2 of 5
		CLS_BCS_UCCEAN128RANDOMWEIGHT	int	122	(Non-Japanese model) UCC/EAN-128 Random Weight
		CLS_BCS_TELEPEN	int	123	Telepen
13	Human readable text of barcode	CLS_BCS_TEXT_HIDE	int	0	Hide
		CLS_BCS_TEXT_SHOW	int	1	Show
14	Error correction level (PDF417)	CLS_PDF417_EC_LEVEL_0	int	0	Level 0
		CLS_PDF417_EC_LEVEL_1	int	1	Level 1

		CLS_PDF417_EC_LEVEL_2	int	2	Level 2
		CLS_PDF417_EC_LEVEL_3	int	3	Level 3
		CLS_PDF417_EC_LEVEL_4	int	4	Level 4
		CLS_PDF417_EC_LEVEL_5	int	5	Level 5
		CLS_PDF417_EC_LEVEL_6	int	6	Level 6
		CLS_PDF417_EC_LEVEL_7	int	7	Level 7
		CLS_PDF417_EC_LEVEL_8	int	8	Level 8
15	Error correction level (Data Matrix)	CLS_DATAMATRIX_EC_LEVEL_200	int	200	
16	Error correction level (QR Code)	CLS_QRCODE_EC_LEVEL_L	int	0	Error correction level L (7%)
		CLS_QRCODE_EC_LEVEL_M	int	1	Error correction level M (15%)
		CLS_QRCODE_EC_LEVEL_Q	int	2	Error correction level Q (25%)
		CLS_QRCODE_EC_LEVEL_H	int	3	Error correction level H (30%)
17	Error correction level (Aztec)	CLS_AXTEC_EC_LEVEL_000	int	0	Error correction ratio 23%
18	Barcode symbology (GS1 DataBar)	CLS_GS1_DATABAR_OMNI_DIRECTIONAL	int	0	GS1DataBar Omni-dirctional
		CLS_GS1_DATABAR_COMPOSITE	int	1	GS1DataBar Conposite
		CLS_GS1_DATABAR_TRUNCATION	int	2	GS1DataBar Truncation
		CLS_GS1_DATABAR_STACKED	int	3	GS1DataBar Stacked
		CLS_GS1_DATABAR_STACKED_OMNI_DIRECTIONAL	int	4	GS1DataBar Stacked Omni-dirctional
		CLS_GS1_DATABAR_LIMITED	int	5	GS1DataBar Limited
		CLS_GS1_DATABAR_EXPANDED	int	6	GS1DataBar Expanded
19	Shaded pattern	CLS_SHADED_PTN_0	int	0	Shaded pattern : 000
		CLS_SHADED_PTN_1	int	1	Shaded pattern : 001
		CLS_SHADED_PTN_2	int	2	Shaded pattern : 002
		CLS_SHADED_PTN_3	int	3	Shaded pattern : 003
		CLS_SHADED_PTN_4	int	4	Shaded pattern : 004
		CLS_SHADED_PTN_5	int	5	Shaded pattern : 005
		CLS_SHADED_PTN_6	int	6	Shaded pattern : 006
		CLS_SHADED_PTN_7	int	7	Shaded pattern : 007
		CLS_SHADED_PTN_8	int	8	Shaded pattern : 008
		CLS_SHADED_PTN_9	int	9	Shaded pattern : 009
		CLS_SHADED_PTN_10	int	10	Shaded pattern : 010
		CLS_SHADED_PTN_11	int	11	Shaded pattern : 011
20	Measurement unit	CLS_UNIT_MILLI	int	0	Metric
		CLS_UNIT_INCH	int	1	Inch
21	Speed setting	CLS_SPEEDSETTING_1	int	1	1：1.0inch(25.4mm) / sec
		CLS_SPEEDSETTING_2	int	2	2 : 2.0inch(50.8mm) / sec
		CLS_SPEEDSETTING_3	int	3	3 : 3.0inch(76.2mm) / sec
		CLS_SPEEDSETTING_4	int	4	4 : 4.0inch(101.6mm) / sec
		CLS_SPEEDSETTING_5	int	5	5 : 5.0inch(127.0mm) / sec
		CLS_SPEEDSETTING_6	int	6	6 : 6.0inch(152.4mm) / sec
		CLS_SPEEDSETTING_7	int	7	7 : 7.0inch(177.8mm) / sec
		CLS_SPEEDSETTING_8	int	8	8 : 8.0inch(203.2mm) / sec
		CLS_SPEEDSETTING_9	int	9	9 : 9.0inch(228.6mm) / sec
		CLS_SPEEDSETTING_A	int	10	A : 1.0inch(25.4mm) / sec

		CLS_SPEEDSETTING_B	int	11	B : 1.0inch(25.4mm) / sec
		CLS_SPEEDSETTING_C	int	12	C : 2.0inch(50.8mm) / sec
		CLS_SPEEDSETTING_D	int	13	D : 2.0inch(50.8mm) / sec
		CLS_SPEEDSETTING_E	int	14	E : 3.0inch(76.2mm) / sec
		CLS_SPEEDSETTING_F	int	15	F : 3.0inch(76.2mm) / sec
		CLS_SPEEDSETTING_G	int	16	G : 4.0inch(101.6mm) / sec
		CLS_SPEEDSETTING_H	int	17	H : 4.0inch(101.6mm) / sec
		CLS_SPEEDSETTING_I	int	18	I : 5.0inch(127.0mm) / sec
		CLS_SPEEDSETTING_J	int	19	J : 5.0inch(127.0mm) / sec
		CLS_SPEEDSETTING_K	int	20	K : 6.0inch(152.4mm) / sec
		CLS_SPEEDSETTING_L	int	21	L : 6.0inch(152.4mm) / sec
		CLS_SPEEDSETTING_M	int	22	M : 7.0inch(177.8mm) / sec
		CLS_SPEEDSETTING_N	int	23	N : 7.0inch(177.8mm) / sec
		CLS_SPEEDSETTING_O	int	24	O : 8.0inch(203.2mm) / sec
		CLS_SPEEDSETTING_P	int	25	P : 8.0inch(203.2mm) / sec
		CLS_SPEEDSETTING_Q	int	26	Q : 9.0inch(228.6mm) / sec
		CLS_SPEEDSETTING_R	int	27	R : 9.0inch(228.6mm) / sec
		CLS_SPEEDSETTING_S	int	28	S : 10.0inch(254.0mm) / sec
		CLS_SPEEDSETTING_T	int	29	T : 10.0inch(254.0mm) / sec
		CLS_SPEEDSETTING_U	int	30	U : 11.0inch(279.4mm) / sec
		CLS_SPEEDSETTING_V	int	31	V : 11.0inch(279.4mm) / sec
		CLS_SPEEDSETTING_W	int	32	W : 12.0inch(304.8mm) / sec
		CLS_SPEEDSETTING_X	int	33	X : 12.0inch(304.8mm) / sec
22	Media handling	CLS_MEDIAHANDLING_NONE	int	0	None
		CLS_MEDIAHANDLING_TEAROFF	int	1	Tear-off
		CLS_MEDIAHANDLING_DISPENSES	int	2	Dispense
		CLS_MEDIAHANDLING_PAUSE	int	3	Pause
		CLS_MEDIAHANDLING_CUT	int	4	Cut
		CLS_MEDIAHANDLING_CUTANDPAUSE	int	5	Cut and Pause
		CLS_MEDIAHANDLING_PEELOFF	int	6	Peel-off
		CLS_MEDIAHANDLING_REWIND	int	7	Rewind
23	Label sensor	CLS_SELSENSOR_NONE	int	0	None
		CLS_SELSENSOR_SEETHROUGH	int	1	See Through
		CLS_SELSENSOR_REFLECT	int	2	Reflect
24	Print method	CLS_PRTMETHOD_TT	int	0	Thermal transfer
		CLS_PRTMETHOD_DT	int	1	Direct thermal
25	Sensor location	CLS_SENS_LOCATION_FRONT	int	0	Front sensor
		CLS_SENS_LOCATION_ADJUSTABLE	int	1	Rear sensor

CITIZEN iOS Label Print SDK (Swift) - Programming Manual
October 10, 2018 Version 1.01
CITIZEN SYSTEMS JAPAN CO., LTD.
http://www.citizen-systems.co.jp/