

# CITIZEN

## **iOS Layout SDK**

Programming Manual  
for Version 1.3.0

**CITIZEN SYSTEMS JAPAN CO., LTD.**

## Revision History

Date	Version	Description
2017.06.12	1.3.0.0	- First issue. (Supported iOS)
2017.12.07	1.3.0.1	- Updated supported OS : iOS 11.2 - Updated required software : iOS POS Print SDK Version 1.17 or newer
2018.10.25		- Updated supported OS : iOS 12.0
2019.04.03		- Updated supported OS : iOS 12.2
2019.10.07		- Updated supported OS : iOS 13.1 / iPad OS 13.1
2019.11.12		- Updated supported OS : iOS 13.2 / iPad OS 13.2
2020.03.13		- Updated supported OS : iOS 13.3.1 / iPad OS 13.3.1
2021.10.28		- Updated supported OS : iOS 15 / iPad OS 15 - Modified the definition method description.
2024.02.02		- Updated supported OS : iOS 17 / iPad OS 17 - Supported privacy manifest .

## 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 library.

## Copyright / Trademarks

- The copyright for this Programming Manual belongs to Citizen Systems Japan Co., Ltd.
- CITIZEN is a registered trademark of Citizen Watch Co., Ltd.
- Windows and Windows Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
- Pentium is a trademark or registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.
- QR Code and iQR Code are registered trademarks of DENSO WAVE INCORPORATED in Japan and in other countries.
- Android is a trademark of Google Inc.
- IOS is a trademark or registered trademark of Cisco in the U.S. and other countries and is used under license.
- Oracle and JavaScript are registered trademarks of Oracle and/or its affiliates.

All other company and/or brand/product names are trademarks and/or registered trademarks of their respective owners.

---

## Table of Contents

Revision History.....	2
Permission Notice.....	3
Copyright / Trademarks.....	3
Table of Contents.....	4
1. Introduction.....	5
1.1. Who should read this document.....	5
1.2. System summary .....	5
1.3. Supported terminals.....	6
1.4. Supported printer .....	6
1.5. Definition method .....	6
1.6. Functions list.....	8
2. Library interface .....	9
2.1. Constructor .....	9
2.2. open .....	10
2.3. close.....	11
2.4. beginPrint.....	12
2.5. endPrint.....	13
2.6. doPrint.....	14
2.7. setFrame .....	15
2.8. setPartsData .....	16
2.9. addFrame.....	17
3. Example of using the method ( swift ).....	18
4. Notes .....	19
4.1. About the detection of print completion .....	19

## 1. Introduction

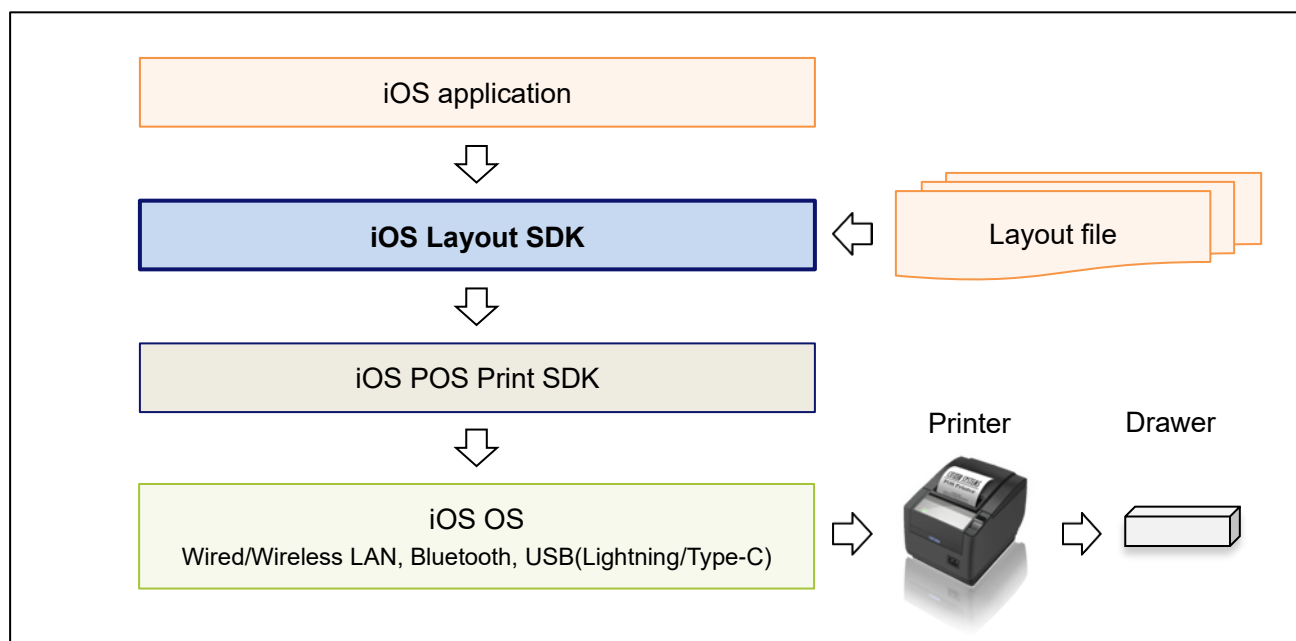
This document is a programming manual for the iOS Layout SDK.

### 1.1. Who should read this document

This document is intended for reference of iOS application developers that use the CITIZEN layout file.

### 1.2. System summary

This library is referred by iOS application program to use CITIZEN layout file.



System diagram of the library

### 1.3. Supported terminals

OS	iOS 8 – 17 / iPad OS 13 - 17
Required	Swift 3.0.1 or newer
	iOS POS Print SDK Version 1.17 or newer

For more information about the iOS POS Print SDK, please refer to the "iOS POS Print SDK Programming Manual".

### 1.4. Supported printer

Target printer of this library is supported by the iOS POS Print SDK.

Refer to user's manual of each printer for more details.

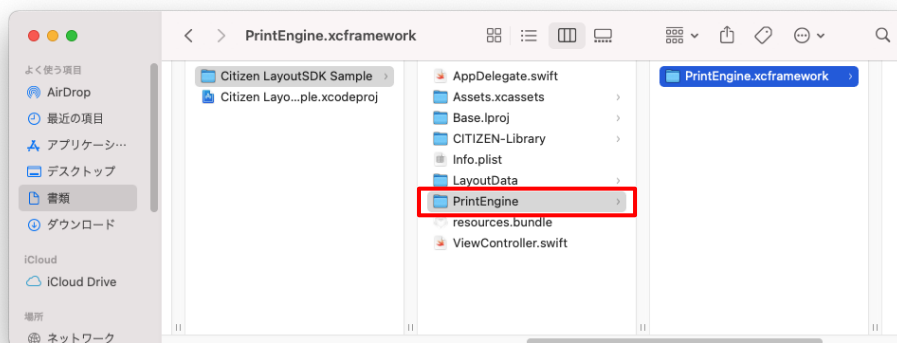
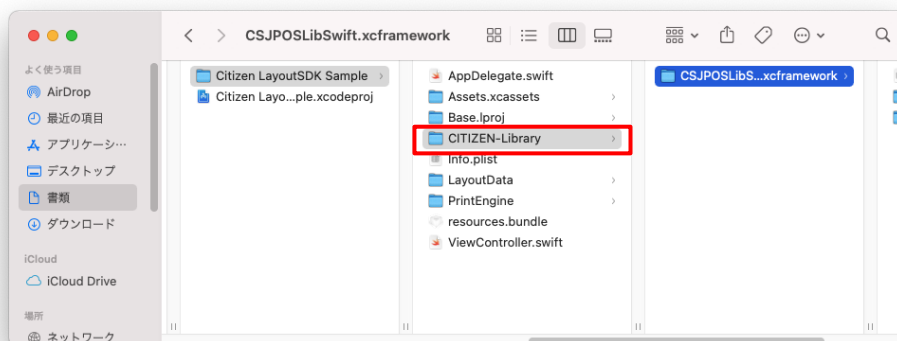
### 1.5. Definition method

Add the library file that is provided, if you use this library, please import the following classes.

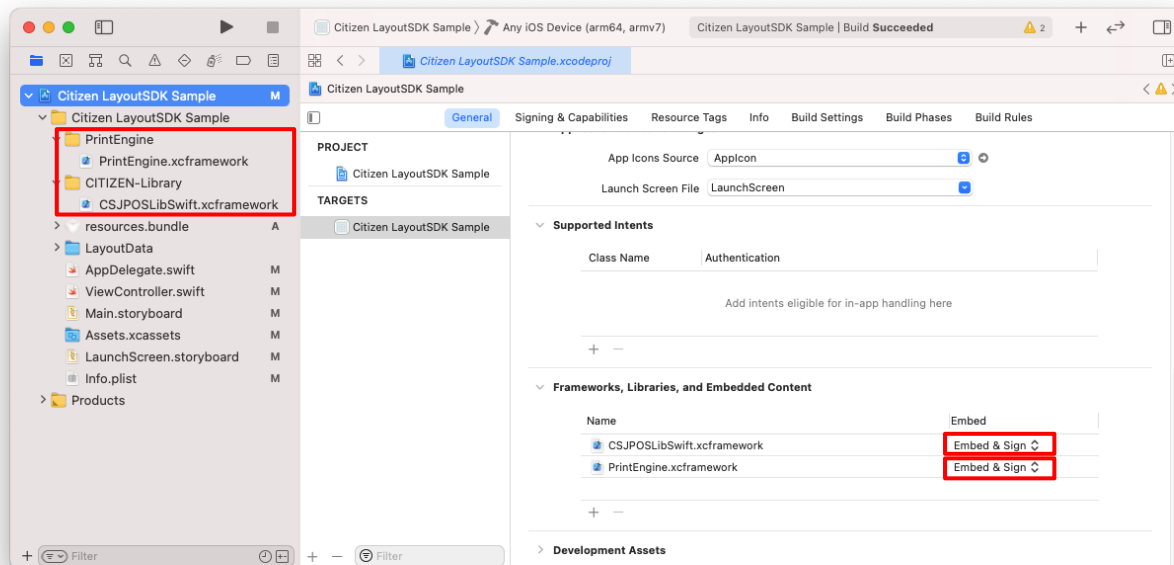
```
import CSJPOSLibSwift;           // iOS POS Print SDK
import PrintEngine;             // iOS Layout SDK
```

< Add provided library files to Xcode >

Copy the iOS POS Print SDK and iOS Layout SDK at the 'Finder' to the project under development.



Drag the framework file from the 'Finder' to any place in the "Project Navigator" of Xcode. Select the project file from "Project Navigator". Then, change the "Embed" setting of the framework you just added displayed in "Frameworks, Libraries, and Embedded Content" of "TARGET"- "General" to "Embed & Sign".



## 1.6. Functions list

This library provides the following functions.

### Methods list

No.	Name	Function
1	PrintEngine	This is constructor method.
2	open	Opens the XML layout file with the specified path.
3	close	Closes the XML layout file that is currently open.
4	beginPrint	Starts preparing for printing.
5	endPrint	Discards the printing data.
6	doPrint	Executes printing from the connected printer.
7	initFrame	Searches for the frame to set up the printing data. It will search by the specified frame name and returns the internally controlled frame number.
8	setPartsData	Sets up the printing data (text, barcode, image) for the part. It will search for the part subjected to setting by the frame number obtained by <code>initFrame()</code> and the part name and sets the specified character string.
9	addFrame	Registers a frame set up with printing data as a print subject.



## 2. Library interface

### 2.1. Constructor

#### <Definition>

```
PrintEngine ()
```

#### <Function>

It is the constructor for this library. Create an instance.

#### <Argument>

None.

#### <Returned value>

None.

#### <Example>

```
let engine: PrintEngine? = PrintEngine()
```

## 2.2. open

### <Definition>

```
func open( stream: InputStream ) -> Int32
```

### <Function>

Loads the XML layout file passed as an InputStream object.

Loaded information of the layout file will be kept in PrintEngine class until you call the `close()` method.

#### - Note -

InputStream object does not close at this class, you must close it at the caller.

### <Argument>

#### **stream**

Specifies the InputStream object of the XML layout file to be used as template.

### <Returned value>

- |             |   |  |
|-------------|---|--|
| 0           | : | Indicates that the XML layout file was opened properly.  |
| Less than 0 | : | Indicates that some kind of error has occurred, including incorrect format of XML layout file. |

### <Example>

```
var result: Int32 = engine?.open( stream: stream! )
```

## 2.3. close

### <Definition>

```
func close()
```

### <Function>

Discards the information of the layout file that was loaded by the open method.

After calling this method, `initFrame()` / `setPartsData()` / `addFrame()` / `doPrint()` method will result in an error. You must call the `open()` method again.

### <Argument/returned value>

None.

### <Example>

```
engine.close ();
```

## 2.4. beginPrint

### <Definition>

```
func beginPrint() -> Int32
```

### <Function>

Prepares creation for the printing layout.

The printing layout is created by `initFrame()` / `setPartsData()` / `addFrame()` method after executing this method that will be kept in `PrintEngine` class until you call the `endPrint()` method.

### <Argument/returned value>

- |             |   |  |
|-------------|---|--|
| 0           | : | Indicates that the process was completed successfully. |
| Less than 0 | : | Indicates that some kind of error has occurred.        |

### <Example>

```
engine.beginPrint ();
```

## 2.5. endPrint

### <Definition>

```
func endPrint()
```

### <Function>

Discards the printing layout.

After calling this method, you cannot print by `doPrint()` method.

You must call the `beginPrint()` method to create the printing layout, again.

### <Argument/returned value>

None.

### <Example>

```
engine.endPrint ();
```

## 2.6. doPrint

### <Definition>

```
func doPrint( printer : ESCPOSPrinter, isReverse : Bool ) -> Int32
```

### <Function>

Prints the printing layout that you have created at the specified printer.

#### - Note -

You must specify the ESCPOSPrinter object that the printer and connection has been established by ESCPOSPrinter.connect() method.

### <Argument>

#### **printer**

Specifies the ESCPOSPrinter object.

#### **isReverse**

Specifies the upside down printing.

true : Print in reverse order and upside down.

false : Print in forward order.

### <Returned value>

- 0 : Indicates that the process was completed successfully.
- Greater than 0 : Indicates that the problem occurs during printing and it is an error code of ESCPOSPrinter class.
- Less than 0 : Indicates that some kind of error has occurred.

### <Example>

```
let printer: ESCPOSPrinter? = CSJPOSlibSwift.ESCPOSPrinter()
printer!.setEncoding( String.Encoding.isoLatin1 )
result = printer!.connect( CMP_PORT_WiFi, withAddress: "192.168.182.100" )
if CMP_SUCCESS == result {
    engine!.doPrint( printer: printer!, isReverse: false )
    printer!.disconnect();
}
```

## 2.7. initFrame

### <Definition>

```
func initFrame( frameName : String ) -> Int32
```

### <Function>

Searches the frame to set up the printing data.

It will search by the specified frame name and return the internally controlled frame number.

### <Argument>

#### **frameName**

Specifies the name of frame to be subjected.

### <Returned value>

1 or larger : Indicates the internally controlled frame number.

Less than 0 : Indicates that some kind of error has occurred, including failure to find the specified frame.

### <Example>

```
let frameIndex = engine!.initFrame( frameName: "Frame1" )
```

## 2.8. setPartsData

### <Definition>

```
func setPartsData( frameIndex : Int32, partsName : String, setText : String ) -> Int32  
func setPartsData( frameIndex : Int32, partsName : String, image : UIImage ) -> Int32
```

### <Function>

Sets up the printing data (text, barcode, image) for the part.

It will search for the part subjected to setting by the frame number obtained by `initFrame()` and the part name and sets the specified character string or data.

### <Argument>

#### **frameIndex**

Specifies the frame number to be subjected (obtained by `initFrame()`).

#### **partsName**

Specifies the name of the part to be subjected.

#### **setText**

Specifies the printing data (character string) to be set up. Apply to the text / barcode parts.

#### **setData**

Specifies the printing data (byte array) to be set up. Apply to the image part.

### <Returned value>

- 0 : Indicates that the process was completed successfully.
- Less than 0 : Indicates that some kind of error has occurred, including failure to find the specified part.

### <Example>

```
engine!.setPartsData( frameIndex: frameIndex, partsName: "Text1", setText: "New Text" )
```



## 2.9. addFrame

### <Definition>

```
func addFrame( frameIndex : Int32 ) -> Int32
```

### <Function>

Registers a frame set up with printing data as a print subject.

### <Argument>

#### **frameIndex**

Specifies the frame number to be subjected (obtained by `initFrame()`).

### <Returned value>

- |             |   |  |
|-------------|---|--|
| 0           | : | Indicates the internally controlled frame number from print registration.                      |
| Less than 0 | : | Indicates that some kind of error has occurred, including failure to find the specified frame. |

### <Example>

```
engine!.addFrame( frameIndex: frameIndex )
```

### 3. Example of using the method ( swift )

```

import CSJPOSLibSwift;                                // iOS POS Print SDK
import PrintEngine;                                   // iOS Layout SDK
:
if let path: String = Bundle.main.path( forResource: "R.raw. my_layout_file", ofType: "XML" ) {
    let stream: InputStream? = InputStream( fileAtPath: path )
    let engine: PrintEngine? = PrintEngine()           // iOS Layout SDK
    var result: Int32 = engine?.open( stream: stream! )
    if 0 == result {
        _ = engine!.beginPrint()
        let frameIndex: Int32 = engine!.initFrame( frameName: " Frame1" )
        _ = engine!.setPartsData( frameIndex: frameIndex, partsName: " Text1", setText: " New Text " )
        _ = engine!.addFrame( frameIndex: frameIndex )

        let printer: ESCPOSPrinter? = CSJPOSLibSwift.ESCPOSPrinter()
        _ = printer!.setEncoding( String.Encoding.isoLatin1 )
        result = printer!.connect( CMP_PORT_WIFI, withAddress: "192.168.128.100" )
        if CMP_SUCCESS == result {
            _ = engine!.doPrint( printer: printer!, isReverse: false )
            _ = printer!.disconnect()
        }

        engine!.endPrint()
        engine!.close()
    }
    stream!.close()
}

```

- Note -

Please refer to the Layout SDK sample program for more information.

<http://www.citizen-systems.co.jp/english/support/download/prINTER/sdk/index.html>

## 4. Notes

Notes of this library are as follows.

### 4.1. About the detection of print completion

The library will make the print using the print completion confirmation that provided by iOS POS Print SDK.

For more information, please refer to the "iOS POS Print SDK Programming Manual".

## **iOS Layout SDK**

Programming Manual  
for Version 1.3.0