

# CITIZEN

Windows Layout SDK via POS Print SDK

Programming Manual  
for Version 1.4.0

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

## Revision History

Date	Version	Description
2017.09.29	1.4.0.0	- First issue.

## 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.
- 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 PC .....	6
1.4. Supported printer .....	6
1.5. Definition method .....	7
1.6. Library files.....	8
1.7. Functions list .....	9
2. Library interface .....	10
2.1. Constructor .....	10
2.2. Open .....	11
2.3. Close.....	12
2.4. BeginPrint .....	13
2.5. EndPrint .....	14
2.6. DoPrint .....	15
2.7. InitFrame.....	17
2.8. SetPartsData.....	18
2.9. AddFrame .....	19
2.10. PrintOption Class .....	20
3. Example of using the method ( C# ) .....	22
4. Notes .....	23
4.1. About the detection of print completion .....	23

## 1. Introduction

This document is a programming manual for the Windows Layout SDK via POS Print SDK.

### 1.1. Who should read this document

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

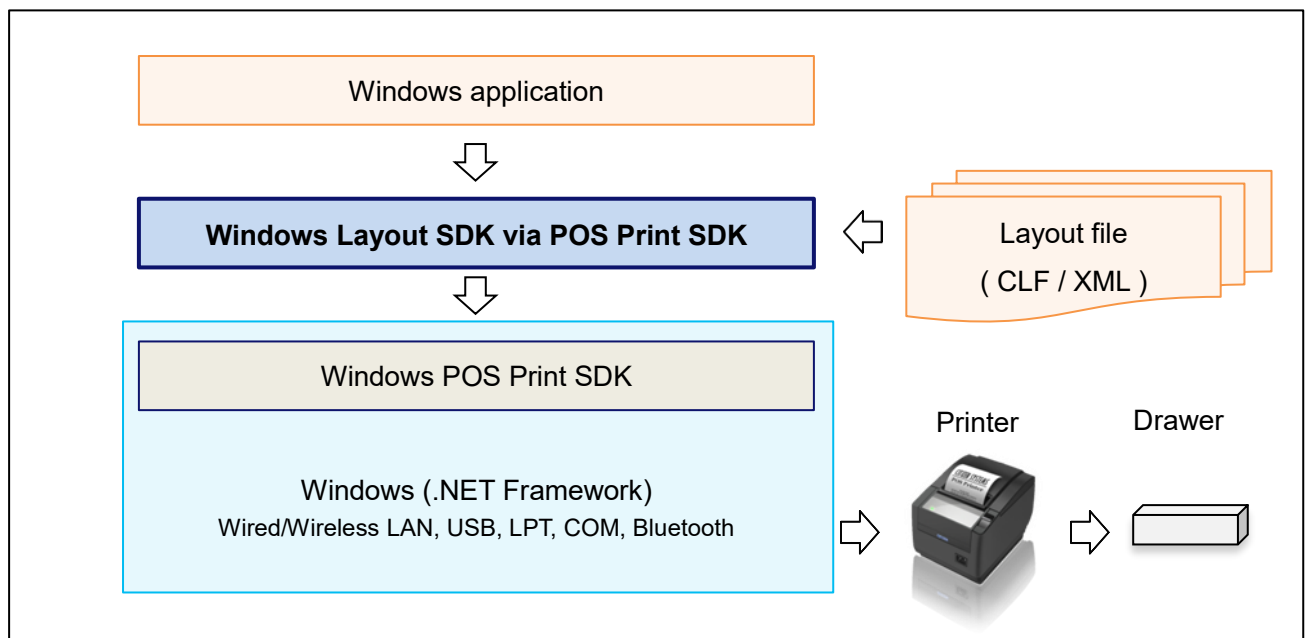
### 1.2. System summary

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

This library supports CLF format and XML format of CITIZEN layout file.

In CLF format, print using PC font.

In XML format, print using printer font.



System diagram of the library

The CITIZEN layout file is created with the Layout Editor.

For details on the Layout Editor, please refer to "Layout SDK User's Guide".

### 1.3. Supported PC

OS	<ul style="list-style-type: none"><li>• Windows XP</li><li>• Windows 7 (32bit, 64bit)</li><li>• Windows 8 (32bit, 64bit), Windows 8.1 (32bit, 64bit)</li><li>• Windows 10 (32bit, 64bit)</li><li>• Windows Server 2008, Windows Server 2008R2</li><li>• Windows Server 2012</li></ul>
Required	<ul style="list-style-type: none"><li>• .NET Framework 4.0</li><li>• Windows POS Print SDK Version 1.06.4 or later<sup>*1</sup></li></ul>

<sup>\*1</sup>: "**Windows POS Print SDK**" is included in the Layout SDK.

"**Windows printer driver**" is not required.

If you install the Windows printer driver, you need to uncheck the "**Enable bidirectional support**" option under the **Ports setting** of the Windows printer driver.

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

### 1.4. Supported printer

Target printer of this library is supported by the CITIZEN printer driver.

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

## 1.5. Definition method

### Adding Library

1. Install the **Layout SDK ([LayoutSDK\_Setup\_en])**.

For more information, please refer to the “Layout SDK Users Guide”.

2. This SDK is structured by the following three files. All of them must be stored in the same folder as the application program. These are Windows POS Print SDK library files.

- CSJPOSLib.dll
- CSJPOSLibW32.dll
- CSJPOSLibW64.dll

3. Select [**Copy always**] in the [**Copy to Output Directory**] property of the [**Properties**] window of the three added files.

### Adding Reference

To add a reference in Visual C#:

1. In **Solution Explorer**, right-click the project node and click **Add Reference**.
2. In the **Add Reference** dialog box, click **Browse** tab.
3. Select the following two files, and then click **OK**.

[Project folder]¥CSJPOSLib.dll

C:¥Program Files¥CITIZEN¥Layout Utilities¥CSJSDKLayoutPrintEngine.dll

To add a reference in Visual Basic:

4. In **Solution Explorer**, double-click the **My Project** node for the project.
5. In the **Project Designer**, click the **References** tab.
6. Click the **Add** button to open the **Add Reference** dialog box.
7. In the **Add Reference** dialog box, click **Browse** tab.
8. Select the following file, and then click **OK**.

[Project folder]¥CSJPOSLib.dll

C:¥Program Files¥CITIZEN¥Layout Utilities¥CSJSDKLayoutPrintEngine.dll

### Adding Namespace

In the case of Visual C#:

```
using com.citizen.sdk;  
using com.citizen.sdk.poslayout;
```

In the case of Visual Basic:

```
Imports com.citizen.sdk;  
Imports com.citizen.sdk.poslayout;
```

## 1.6. Library files

The library consists of the following files:

- AxInterop.QRMAKERADLib.dll
- Citizen.LayoutUtilities.Common.dll
- **CSJPOSLib.dll**<sup>†</sup>
- CSJPOSLibW32.dll<sup>\*</sup>
- CSJPOSLibW64.dll<sup>\*</sup>
- **CSJSDKLayoutPrintEngine.dll**<sup>†</sup>
- GrapeSystems.Core.Common.dll
- GrapeSystems.Core.Drawing.Fx20.dll
- GrapeSystems.Core.Parts.dll
- GrapeSystems.Core.Parts.Frames.dll
- GrapeSystems.Library.BarcodeAd.dll
- GrapeSystems.Library.Controls.dll
- GrapeSystems.Library.Image.dll
- Interop.QRMAKERADLib.dll

---

<sup>\*</sup>: This file requires Copy to the target project.

<sup>†</sup>: This file requires **Add Reference** to the target project.



## 1.7. Functions list

This library provides the following functions.

### Methods list

No.	Name	Function
1	LayoutPrintEngine	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 specified 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.

### Class

No.	Name	Function
10	PrintOption	This is printing option. It is an argument of <code>DoPrint ()</code> .

## 2. Library interface

### 2.1. Constructor

#### < Definition >

```
com.citizen.sdk.poslayout.LayoutPrintEngine ()
```

#### < Function >

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

#### < Argument >

None.

#### < Returned value >

None.

#### < Example >

```
com.citizen.sdk.poslayout.LayoutPrintEngine lpe =  
new com.citizen.sdk.poslayout.LayoutPrintEngine();
```

## 2.2. Open

### < Definition >

```
int Open ( string pathName )
```

### < Function >

Opens the CLF / XML layout file with the specified path.

### < Argument >

#### **pathName**

Specifies the full path of the CLF / XML layout file to be used as template.

### < Returned value >

- 0 : Indicates that the CLF / XML layout file was opened properly.
- Other than 0 : Indicates that some kind of error has occurred, including failure to find the CLF / XML layout file.

### < Example >

```
lpe.Open( "my_layout_File.CLF" );
```

## 2.3. Close

### < Definition >

```
void Close ()
```

### < Function >

Closes the CLF / XML layout file that is currently open.

### < Argument / returned value >

None.

### < Example >

```
lpe.Close();
```

## 2.4. BeginPrint

### < Definition >

```
void BeginPrint ()
```

### < Function >

Starts preparing for printing.

### < Argument / returned value >

None.

### < Example >

```
lpe.BeginPrint();
```

## 2.5. EndPrint

### < Definition >

```
void EndPrint ()
```

### < Function >

Discards the printing data.

### < Argument / returned value >

None.

### < Example >

```
lpe.EndPrint();
```

## 2.6. DoPrint

### < Definition >

```
int DoPrint ( ESCPOSPrinter printer, boolean isReverse )  
int DoPrint ( ESCPOSPrinter printer, boolean isReverse, PrintOption option )  
int DoPrint ( ESCPOSPrinter printer, boolean isReverse, boolean isAlternativeFont )  
int DoPrint ( ESCPOSPrinter printer, boolean isReverse, boolean isAlternativeFont, PrintOption  
             option )
```

### < Function >

Executes printing from 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.

The ESCPOSPrinter class is provided by Windows POS Print SDK.

#### **isReverse**

Specifies the upside down printing.

true: Print in reverse order and upside down.

false: Print in forward order.

#### **isAlternativeFont**

Specifies the alternative font.

If isAlternativeFont is omitted, print it with false.

Only the text part in the CLF layout file is valid.

true: Prints with the setting of the [**Printer Font**] property of the text part.

false: Prints with the setting of the [**Font**] property of the text part.

#### **option**

Specifies the printing options. For details, see the PrintOption class.

If **option** is omitted, the behavior depends on the format of the layout file.

CLF format: Prints with default value of PrintOption class.

XML format: Prints with the [Driver Settings] at export.

## &lt; Returned value &gt;

- 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.

## &lt; Example &gt;

```
com.citizen.sdk.ESCPOSPrinter printer = new com.citizen.sdk.ESCPOSPrinter ();
printer.setEncoding ( "ISO-8859-1" );
result = printer.Connect ( ESCPOSConst.CMP_PORT_WiFi, "192.168.10.100" );
if ( ESCPOSConst.CMP_SUCCESS == result ) {
    lpe.DoPrint ( printer, false, false, option );
    printer.disconnect ();
}
```



## 2.7. InitFrame

### < Definition >

```
int InitFrame ( string frameName )
```

### < Function >

Searches for 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 >

0 or larger : Indicates the internally controlled frame number.

-1 : Indicates that some kind of error has occurred, including failure to find the specified frame.

### < Example >

```
int frameIndex = lpe.InitFrame( "Frame1" );
```

## 2.8. SetPartsData

### < Definition >

```
int SetPartsData ( int frameIndex, int partsIndex, string setText )
```

### < 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.

### < 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 (text, barcode, image) to be set up.

### < Returned value >

- 0 : Indicates that the process was completed successfully.
- Other than 0 : Indicates that some kind of error has occurred, including setup failure.

### < Example >

```
lpe.SetPartsData( frameIndex, partsIndex, "New Text" );
```

```
lpe.SetPartsData( frameIndex, partsIndex, "New Image File Path" );
```

## 2.9. AddFrame

### < Definition >

```
int AddFrame ( int frameIndex )
```

### < 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 or larger | : | Indicates the internally controlled frame number from print registration.       |
| -1          | : | Indicates that some kind of error has occurred, including registration failure. |

### < Example >

```
lpe.AddFrame( frameIndex );
```

## 2.10. PrintOption Class

### < Definition >

**com.citizen.sdk.poslayout.PrintOption ()**

### < Function >

This is printing option. It is an argument of DoPrint ().

### < Argument / returned value >

None.

### < Class member >

No.	Name	Type	Meaning	Setting range
1	MapMode	int	Mapping mode	CPE_MM_METRIC*: 0.01 millimeter. CPE_MM_ENGLISH: 0.001 inch.
2	Halftone	int	Halftone	CPE_HT_THRESHOLD: Threshold CPE_HT_DITHER*: Dither
3	ColorMode	int	Color printing mode	CPE_CMD_MONO*: Monochrome CPE_CMD_GRAY*1: Grayscale
4	PaperMedia	int	Paper type	CPE_PAPER_NORMAL*: Normal CPE_PAPER_LABEL_BM*2: Label/BM
5	CutterMode	int	Cutter mode	CPE_CUT_NONE: No Cut CPE_CUT_PARTIAL*: Partial Cut CPE_CUT_FULL: Full Cut
6	PaperFeed	int	Number of paper feed	Expressed in the unit of measure given by MapMode (default: millimeter). *3 1270*
7	DrawerOpen1	int	Timing of drawer 1	CPE_DRAWER_NEVER*: Never CPE_DRAWER_START: Print Start CPE_DRAWER_END: Print End
8	DrawerOpen2	int	Timing of drawer 2	CPE_DRAWER_NEVER*: Never CPE_DRAWER_START: Print Start CPE_DRAWER_END: Print End
9	PulseWidth1	int	Pulse width of drawer 1	1* ~ 8 ( x 100 ms) *4
10	PulseWidth2	int	Pulse width of drawer 2	1* ~ 8 ( x 100 ms) *5
11	BuzzerStart	int	Number of buzzers at the start of printing	0* ~ 9

12	BuzzerEnd	int	Number of buzzers at the end of printing	0* ~ 9
13	LogoStart	int	Logo setting at the start of printing	CPE_LOGO_NONE*: None CPE_LOGO_PRINT01~20*6: 1 ~ 20
14	LogoEnd	int	Logo setting at the end of printing	CPE_LOGO_NONE*: None CPE_LOGO_PRINT01~20*6: 1 ~ 20

\*: Default value.

\*1: Required the gradation compatible printer.

\*2: Required the label or black-mark compatible printer.

\*3: You need to define CPE\_CUT\_NONE to **CutterMode**.

\*4: **PulseWidth1** is invalid, when **DrawerOpen1** is CPE\_DRAWER\_NEVER.

\*5: **PulseWidth2** is invalid, when **DrawerOpen2** is CPE\_DRAWER\_NEVER.

\*6: You need to register the logo in the printer beforehand.

< Example >

```
com.citizen.sdk.poslayout.PrintOption option = new com.citizen.sdk.poslayout.PrintOption();
option.CutterMode = LPEConst.CPE_CUT_PARTIAL;
option.DrawerOpen1 = LPEConst.CPE_DRAWER_START;
option.PulseWidth1 = 1;
option.LogoStart = LPEConst.CPE_LOGO_PRINT01;
```

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

```

com.citizen.sdk.poslayout.LayoutPrintEngine lpe =
    new com.citizen.sdk.poslayout.LayoutPrintEngine();           // Windows Layout SDK

com.citizen.sdk.poslayout.PrintOption option = new com.citizen.sdk.poslayout.PrintOption ();
option.CutterMode = LPEConst.CPE_CUT_PARTIAL;
option.DrawerOpen1 = LPEConst.CPE_DRAWER_START;
option.PulseWidth1 = 1;
option.LogoStart = LPEConst.CPE_LOGO_PRINT01;

int result = lpe.Open( "CLF / XML Layout File Name" );
if ( LPEConst.CPE_SUCCESS == result ) {
    lpe.BeginPrint();

    int frameIndex = lpe.InitFrame( "Frame1" );
    lpe.SetPartsData( frameIndex, "Text1", "New Text" );
    lpe.AddFrame( frameIndex );

    com.citizen.sdk.ESCPOSPrinter printer = new com.citizen.sdk.ESCPOSPrinter();
    printer.SetEncoding( "ISO-8859-1" );
    result = printer.Connect( ESCPOSConst.CMP_PORT_WiFi, "192.168.182.100" );
    if ( ESCPOSConst.CMP_SUCCESS == result ) {
        lpe.DoPrint( printer, false, false, option );
        printer.Disconnect();
    }

    lpe.EndPrint();
    lpe.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 Windows POS Print SDK.

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

Windows Layout SDK via POS Print SDK

Programming Manual  
for Version 1.4.0