CITIZEN

Windows Layout SDK via Printer Driver

Programming Manual

for Version 1.4.0

CITIZEN SYSTEMS JAPAN CO., LTD.

Revision History

Date	Version	Description
2009.06.01	1.0.0.0	- First issue. (Layout Utilities Users Guide)
2009.11.19	1.0.1.0	- Changed to hide the printing dialog in Layout Print Engine.
2013.11.22	1.1.0.0	- Added support OS: Windows 7, Windows 8, Windows 8.1, Windows Server 2008, Windows Server 2008R2, Windows Server 2012 - Supports the creation of 64-bit user applications. - Supports multi-threaded user applications for printing. - Added support barcode: MaxiCode, Data Matrix, GS1 Composite, GS1 Databar (RSS), iQR - Support for dot-by-dot print of the image. And, Changed the property name from [FixedAspect] to [SizeMode]. - Added speed up method for the first printing: PreparePrint(), HidePreview()
2014.09.08	1.2.0.0	- No update of Windows Layout SDK
2016.08.16	1.3.0.0	- Added support OS: Windows 10 - Changed to .Net Framework 4.0 based. - Change the name "Layout Print Engine" to "Layout SDK". - Separated from users guide for Layout SDK programmer. (This document)
2017.09.29	1.4.0.0	- Change the name "Windows Layout SDK" to "Windows Layout SDK via Printer Driver ".

Permission Notice

Copyright / Trademarks

Table of Contents

1. Introduction

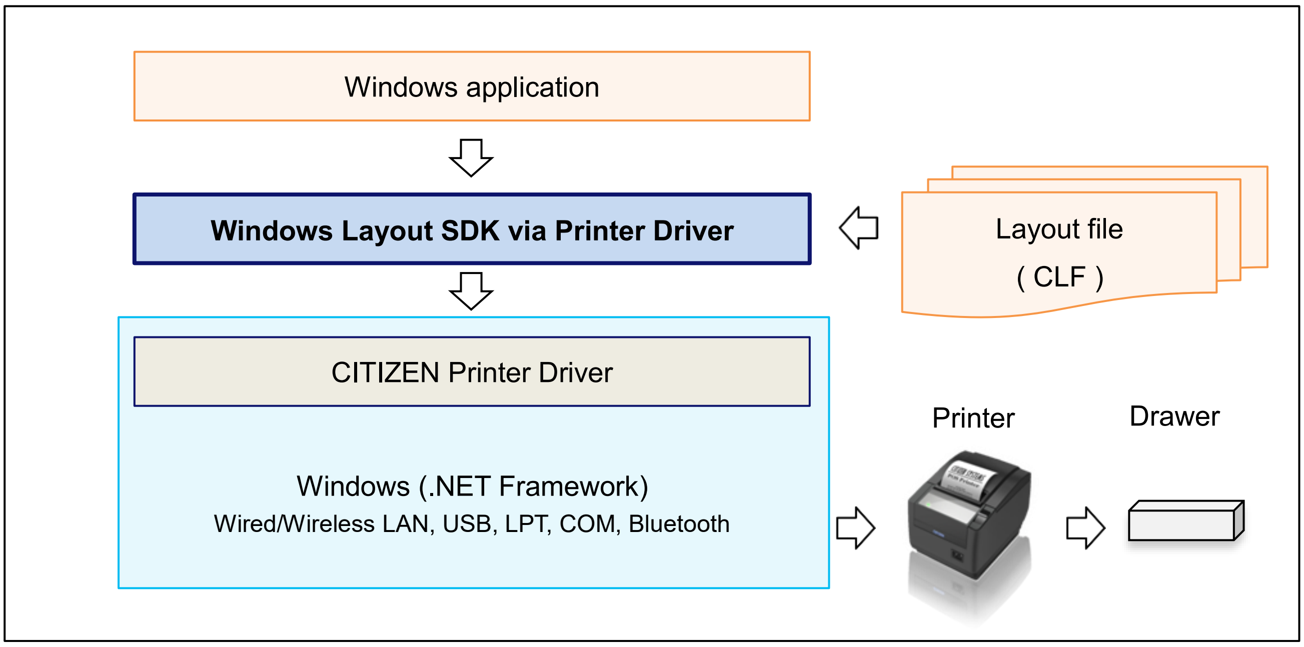
This document is a programming manual for the Windows Layout SDK via Printer Driver.

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 of CITIZEN layout file.
In CLF format, print using PC 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	・ Windows XP
	・ Windows 7 (32bit, 64bit)
	・ Windows 8 (32bit, 64bit), Windows 8.1 (32bit, 64bit)
	・ Windows 10 (32bit, 64bit)
	・ Windows Server 2008, Windows Server 2008R2
	・ Windows Server 2012
Required	・ .NET Framework 4.0
	・ CITIZEN Printer Driver[*1]

[*1]: "**CITIZEN Text Only Printer**" driver is not supported.

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”.

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 file, and then click **OK**.

```
C:¥Program Files¥CITIZEN¥Layout Utilities¥Citizen.LayoutUtilities.Printing.dll
```

To add a reference in Visual Basic:

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

```
C:¥Program Files¥CITIZEN¥Layout Utilities¥Citizen.LayoutUtilities.Printing.dll
```

Adding Namespace

In the case of Visual C#:

```
using Citizen.LayoutUtilities.Printing;
```

In the case of Visual Basic:

```
Imports Citizen.LayoutUtilities.Printing;
```

1.6. Library files

The library consists of the following files:

- AxInterop.QRMAKERADLib.dll
- Citizen.LayoutUtilities.Common.dll
- **Citizen.LayoutUtilities.Printing.dll**[*]
- 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

[*]: 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	Controller	This is constructor method.
2	Open	Opens the CLF layout file with the specified path.
3	Close	Closes the CLF 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	DoPreview	Displays the print preview for the specified printer.
8	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.
9	GetParts	Searches for the parts to set up the printing data. It will search for the parts subjected to setting by the frame number obtained by InitFrame() and the part name and returns the internally controlled part number.
10	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 InitFrame() and the part number obtained by GetParts() and sets the specified character string.
11	AddFrame	Registers a frame set up with printing data as a print subject.
12	PreparePrint	Prepares to speed up the first printing.
13	HidePreview	

2. Library interface

2.1. Constructor

< Definition >

```
Citizen.LayoutUtilities.Printing.Controller ()
```

< Function >

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

< Argument >

None.

< Returned value >

None.

< Example >

```
Citizen.LayoutUtilities.Printing.Controller clpe =
        new Citizen.LayoutUtilities.Printing.Controller();
```

2.2. Open

< Definition >

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

< Function >

Opens the CLF layout file with the specified path.

< Argument >

pathName

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

< Returned value >

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

< Example >

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

2.3. Close

< Definition >

```
void Close ()
```

< Function >

Closes the CLF layout file that is currently open.

< Argument / returned value >

None.

< Example >

```
clpe.Close();
```

2.4. BeginPrint

< Definition >

```
void BeginPrint ()
```

< Function >

Starts preparing for printing.

< Argument / returned value >

None.

< Example >

```
clpe.BeginPrint();
```

2.5. EndPrint

< Definition >

```
void EndPrint ()
```

< Function >

Discards the printing data.

< Argument / returned value >

None.

< Example >

```
clpe.EndPrint();
```

2.6. DoPrint

< Definition >

```
int DoPrint ( string printerName )
```

< Function >

Executes printing from the specified printer.

< Argument >

printerName

Specifies the name of printer to print.

The default printer is used to print when blank is specified.

< Returned value >

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

< Example >

```
clpe.DoPrint( "Printer Name" );
```

2.7. DoPreview

< Definition >

```
int DoPreview ( string printerName )
```

< Function >

Displays the print preview for the specified printer.

< Argument >

printerName

Specifies the name of the printer to execute print preview.

The print preview by the default printer will be displayed if blank is specified.

< Returned value >

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

< Example >

```
clpe.DoPreview( "Printer Name" );
```

2.8. 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 = clpe.InitFrame( "Frame1" );
```

2.9. GetParts

< Definition >

```
int GetParts ( int frameIndex, string partsName )
```

< Function >

Searches for the parts to set up the printing data.

It will search for the parts subjected to setting by the frame number obtained by InitFrame() and the part name and return the internally controlled part number.

< Argument >

frameIndex

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

partsName

Specifies the name of the part to be subjected.

< Returned value >

0 or larger	:	Indicates the internally controlled part number.
-1	:	Indicates that some kind of error has occurred, including failure to find the specified part.

< Example >

```
int partsIndex = clpe.GetParts( frameIndex, "Text1" );
```

2.10. 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 number obtained by GetParts and sets the specified character string.

< Argument >

frameIndex

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

partsIndex

Specifies the part number to be subjected (obtained by GetParts).

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 >

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

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

2.11. 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 >

```
clpe.AddFrame( frameIndex );
```

2.12. PreparePrint

< Definition >

```
int PreparePrint ( string pathName )
```

< Function >

Prepares to speed up the first printing.

< Argument >

pathName

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

< Returned value >

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

< Example >

```
clpe.PreparePrint( "my_layout_File.CLF" );
```

-Note-

This software is based on .NET Framework technology.

.NET Framework libraries are loaded automatically into memory at the start of printing without users being aware.

The first printing becomes slow because it requires several seconds for the loading process. Then the printing after that will be faster.

PreparePrint(), HidePreview() are the methods that improve the speed of the first printing. Please run this method once before the first printing in your program.

Depending on the operating environment, improvement may not lasts long.

If the slower printing is observed after the first printing, please consider to run this method on a regular basis.

2.13. HidePreview

< Definition >

```
int HidePreview ( string pathName )
```

< Function >

Prepares to speed up the first printing.

In HidePreview(), the first print will be speeded up from PreparePrint(). However, please take into account that the processing time of HidePreview() is longer and the preview generation dialog is displayed.

< Argument >

pathName

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

< Returned value >

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

< Example >

```
clpe.HidePreview( "my_layout_File.CLF" );
```

-Note-

This software is based on .NET Framework technology.

.NET Framework libraries are loaded automatically into memory at the start of printing without users being aware.

The first printing becomes slow because it requires several seconds for the loading process. Then the printing after that will be faster.

PreparePrint(), HidePreview() are the methods that improve the speed of the first printing. Please run this method once before the first printing in your program.

Depending on the operating environment, improvement may not lasts long.

If the slower printing is observed after the first printing, please consider to run this method on a regular basis.

3. Example of using the method (C#)

```
Citizen.LayoutUtilities.Printing.Controller clpe =
        new Citizen.LayoutUtilities.Printing.Controller();    // Windows Layout SDK

int result = clpe.Open( " CLF Layout File Name" );
if (0 == result) {
        clpe.BeginPrint();

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

        clpe.DoPrint( "Printer Name" );

        clpe.EndPrint();
        clpe.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 Spooler service that provided by Windows. Therefore you cannot detect the print completion. Please note.

Windows Layout SDK via Printer Driver

Programming Manual
for Version 1.4.0