

# CITIZEN

JavaScript Layout SDK

Programming Manual  
for Version 1.3.0

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

## Revision History

Date	Description
2016.08.16	1.3.0.0 release (First issue)
2023.06.21	Documentation update - Added <a href="#">About print width setting</a> - Added SetRecLineWidth() to <a href="#">Example of using the method ( JavaScript )</a>

## 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 JavaScript terminals.....	6
1.4. Supported printer .....	6
1.5. Definition method .....	6
1.6. Functions list .....	7
2. Library interface .....	8
2.1. Constructor .....	8
2.2. openFilePath.....	9
2.3. openDomObj.....	10
2.4. close.....	11
2.5. beginPrint.....	12
2.6. endPrint.....	13
2.7. preSend.....	14
2.8. initFrame .....	15
2.9. setPartsData .....	16
2.10. setImageData .....	17
2.11. addFrame .....	18
3. Example of using the method ( JavaScript ).....	19
4. Notes .....	21
4.1. About the detection of print completion .....	21
4.2. About print width setting .....	21

## 1. Introduction

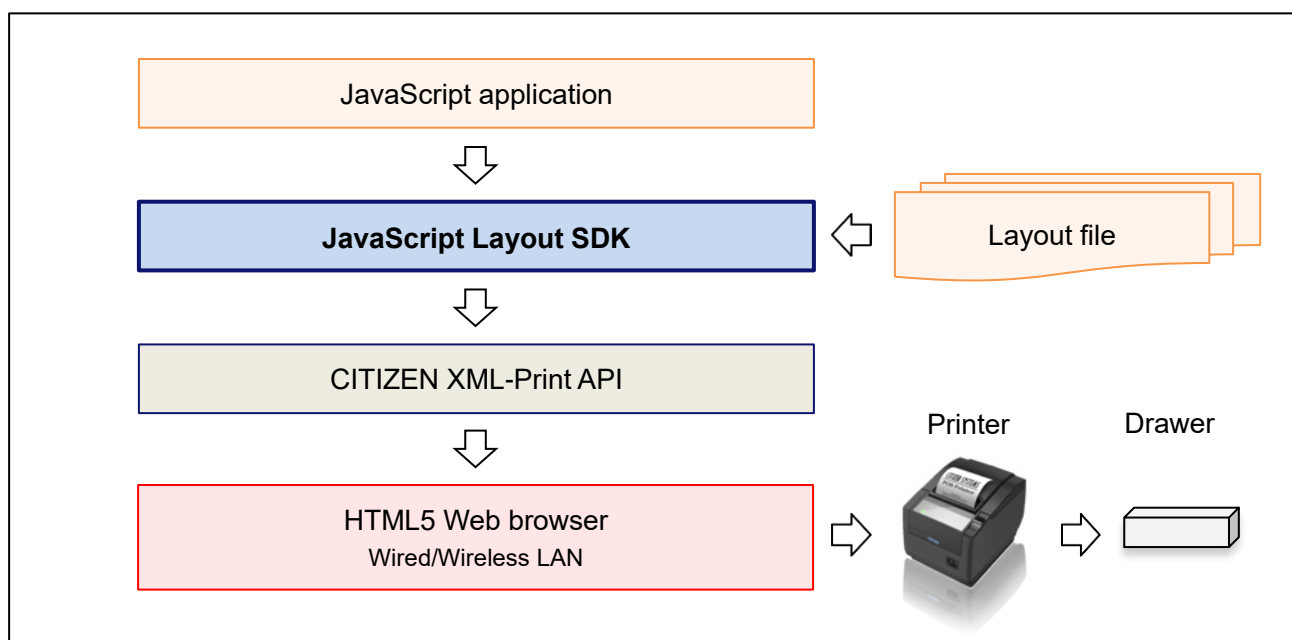
This document is a programming manual for the JavaScript Layout SDK.

### 1.1. Who should read this document

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

### 1.2. System summary

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



System diagram of the library

### 1.3. Supported JavaScript terminals

Web browser	HTML5 supported
Required	CITIZEN XML-Print API Version 1.02 or newer jQuery Version 1.1.4 or newer

For more information about the CITIZEN XML-Print API, please refer to the "CITIZEN XML-Print Service Programming Manual".

JavaScript Layout SDK will use the jQuery.

### 1.4. Supported printer

Target printer of this library is supported by the CITIZEN XML-Print Service.

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.

```
<script type="text/javascript" src="PrintEngine.js"></script>           // JavaScript Layout SDK
<script type="text/javascript" src="jquery-1.7.1.min.js"></script>     // jQuery
<script type="text/javascript" src="cxmmlp-api.js"></script>           // CITIZEN XML-Print API
```

## 1.6. Functions list

This library provides the following functions.

### Methods list

No.	Name	Function
1	PrintEngine	This is constructor method.
2	openFilePath	Opens the XML layout file on the Web server.
2	openDomObj	Open the DOM object that is parsed from the XML layout file.
3	close	Closes the XML layout file that is currently open.
4	beginPrint	Starts preparing for printing.
5	endPrint	Discards the printing data.
6	preSend	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 initFrame() and the part name and sets the specified character string.
8	setPartsData	Registers a frame set up with printing data as a print subject.
9	addFrame	Opens the XML layout file on the Web server.

No.	Property	Function
1	OnLoad	Sets the callback function that will be called when the reading of the XML layout file is complete. Please refer <a href="#">Example of using the method ( JavaScript )</a> .

## 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>

```
var clpe = new citizen.PrintEngine ();
```



## 2.2. openFilePath

### <Definition>

```
int openFilePath ( filePath )
```

### <Function>

Load the layout file from the Web server, and call the callback function.

Please set the callback function to the OnLoad property in advance. You can see the success or failure of the loading by the information of the argument of the callback function.

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

### <Argument>

**filePath** (String)

Specifies the full path of the XML layout file on the Web server.

### <Returned value>

0 : Indicates that the layout file was opened properly.\*<sup>1</sup>

Less than 0 : Indicates that some kind of error has occurred, including incorrect format of the layout file.

\*<sup>1</sup> : The returned value may not reflect the results of loading of the image parts. Please judgment by the argument of the callback function.

### <Example>

```
clpe.openFilePath ( "my_layout_File.XML" );
```

## 2.3. openDomObj

### <Definition>

```
int openDomObj ( domObj )
```

### <Function>

Load the DOM object (Document type) that parsed from the XML layout file, and call the callback function.

Please set the callback function to the OnLoad property in advance. You can see the success or failure of the loading by the information of the argument of the callback function.

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

### <Argument>

**domObj** (Document)

Specifies the DOM object that parsed from the XML layout file.

### <Returned value>

0 : Indicates that the layout file was opened properly. <sup>\*1</sup>

Less than 0 : Indicates that some kind of error has occurred, including incorrect format of the layout file.

<sup>\*1</sup> : The returned value may not reflect the results of loading of the image parts. Please judgment by the argument of the callback function.

### <Example>

## 2.4. close

### <Definition>

```
void close ()
```

### <Function>

Discards the information of the layout file that was loaded by the `openFilePath()` / `openDomObj()` method.

After calling this method, `initFrame()` / `setPartsData()` / `addFrame()` / `preSend()` method will result in an error. You must call the `openFilePath()` or `openDomObj()` method again.

### <Argument/returned value>

None.

### <Example>

```
clpe.close ();
```

## 2.5. beginPrint

### <Definition>

```
int beginPrint ()
```

### <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>

```
clpe.beginPrint ();
```

## 2.6. endPrint

### <Definition>

```
void endPrint ()
```

### <Function>

Discards the printing layout.

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

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

### <Argument/returned value>

None.

### <Example>

```
clpe.endPrint ();
```

## 2.7. preSend

### <Definition>

```
int preSend ( cxp, isReverse )
```

### <Function>

Prepares to send to the printer based on the print buffer.

### <Argument>

**cxp** (CXMLPrint object)

Specifies the CXMLPrint object which has been subjected to necessary initial setting according to the output destination printer.

**isReverse** (Boolean)

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.

Less than 0 : Indicates that some kind of error has occurred.

### <Example>

```
var cxp = new citizen.CXMLPrint ();  
clpe.preSend ( cxp, false );  
cxp.Send ( "http://192.168.129.82:8080/" ); // Send print data
```

## 2.8. initFrame

### <Definition>

```
int initFrame ( frameName )
```

### <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** (String)

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>

```
var frameIndex = clpe.initFrame ( "Frame1" );
```

## 2.9. setPartsData

### <Definition>

```
int setPartsData ( frameIndex, partsName, setText )
```

### <Function>

Sets up the printing data (text, barcode) 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** (int)

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

**partsName** (String)

Specifies the name of the part to be subjected.

**setText** (String)

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

### <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>

```
clpe.setPartsData ( frameIndex , "Text1" , "NewText" );
```



## 2.10. setImageData

### <Definition>

```
int setImageData ( frameIndex, partsName, imageObj )
```

### <Function>

Sets up the printing data (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** (int)

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

**partsName** (String)

Specifies the name of the part to be subjected.

**imageObj** (Image object)

Specifies the printing data (Image object) to be set up. Apply to the image parts.

### <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>

```
clpe.setImageData ( frameIndex , "Image1", imageObj );
```

## 2.11. addFrame

### <Definition>

```
int addFrame ( frameIndex )
```

### <Function>

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

### <Argument>

**frameIndex** (int)

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>

```
clpe.addFrame ( frameIndex );
```

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

```

<script type="text/javascript" src="jquery-1.7.1.min.js"></script>
<script type="text/javascript" src="cxmip-api.js"></script>
<script type="text/javascript" src="PrintEngine.js"></script>

<script language="javascript" type="text/javascript">
    function doPrint() {
        var clpe = new citizen.PrintEngine();// JavaScript Layout SDK
        clpe.OnLoad = doPrintProcess;
        clpe.openFilePath( "my_layout_File.XML" );
    }
    function doPrintProcess(clpe, result ) {
        if ( 0 > result ) { return; }                // Layout file open error

        var cxp = new citizen.CXMLPrint();
        cxp.MessageID( "12345678" );
        cxp.OnError = function(res) { window.alert( res.ResponseCode ); };
        cxp.OnReceive = function(res) { window.alert( res.status ); };
        cxp.SetEncoding( "USA" );                    // Codepage PC437
        // cxp.SetEncoding( "Multilingual" );          // Codepage PC850
        // cxp.SetRecLineWidth( 384 );                  // MSW8-1: 384 dots (58 mm paper)
        // cxp.SetRecLineWidth( 576 );                  // MSW8-1: 576 dots (80 mm paper)
        // cxp.SetRecLineWidth( 832 );                  // MSW8-1: 832 dots (112 mm paper)

        clpe.beginPrint();
        var frameIndex = clpe.initFrame( "Frame1" );
        clpe.setPartsData( frameIndex , "Text1", "NewText" );
        clpe.addFrame( frameIndex );

        clpe.preSend( cxp, false );
        cxp.Send( "http://192.168.129.82:8080/" );    // Send print data

        clpe.endPrint();
        clpe.close();
    }
</script>

```

- 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 CITIZEN XML-Print API.

For more information, please refer to the "CITIZEN XML-Print Service Programming Manual".

### 4.2. About print width setting

Set the print width with the `SetRecLineWidth()` method of the CITIZEN XML-Print API. For the setting value, specify the value of the printer's print width MSW8-1. If the MSW8-1 is 576 dots (80 mm paper), you can omit the print width setting. Refer to user's manual of each printer for MSW8-1.

Example when MSW8-1 is 384 dots (58 mm paper)

```
cxp.SetRecLineWidth( 384 );
```

Example when MSW8-1 is 576 dots (80 mm paper)

```
cxp.SetRecLineWidth( 576 );
```

Example when MSW8-1 is 832 dots (112 mm paper)

```
cxp.SetRecLineWidth( 832 );
```

JavaScript Layout SDK

Programming Manual  
for Version 1.3.0