CITIZEN

Android Layout SDK

Programming Manual
for Version 1.3.0

CITIZEN SYSTEMS JAPAN CO., LTD.

Revision History

Date	Version	Description
2014.09.08	1.2.0.0	- First issue. (Layout Utilities Users Guide)
2016.08.16	1.3.0.0	- Supported "Straight Line" and "Rectangle" on the layout for the mobile terminal. - Changed the layout file from CLFX to XML for the mobile terminal. - Change the designation from "Layout Print Engine" to "Layout SDK". - Separated from users guide for Layout SDK programmer. (This document)

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

Table of Contents

1. Introduction

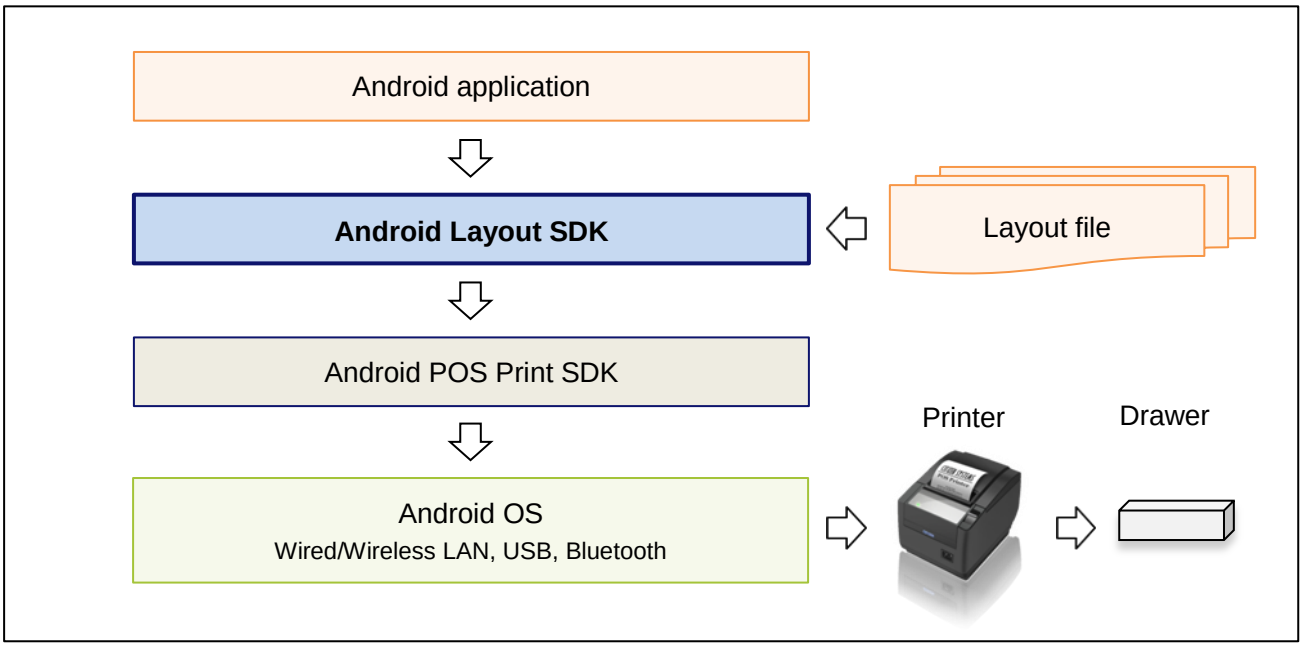
This document is a programming manual for the Android Layout SDK.

1.1. Who should read this document

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

1.2. System summary

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



System diagram of the library

1.3. Supported terminals

OS	Android 2.3.3(API 10) or newer for Wi-Fi, Bluetooth interface Android 3.1(API 12) or newer for Wi-Fi, Bluetooth, USB Host interface
Required	Android POS Print SDK Version 1.14 or newer

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

1.4. Supported printer

Target printer of this library is supported by the Android 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 com.citizen.sdk.ESCPOSConst;          // Android POS Print SDK
import com.citizen.sdk.ESCPOSPrinter;        // Android POS Print SDK
import com.citizen.sdk.LayoutPrintEngine;    // Android Layout SDK
```

1.6. 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 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.
9	addFrame	Registers a frame set up with printing data as a print subject.

2. Library interface

2.1. Constructor

<Definition>

```
LayoutPrintEngine ()
```

<Function>

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

<Argument>

None.

<Returned value>

None.

<Example>

```
LayoutPrintEngine lpe = new LayoutPrintEngine ();
```

2.2. open

<Definition>

```
int open ( InputStream stream )
```

<Function>

Loads the XML layout file passed as an InputStream object.

Loaded information of the layout file will be kept in LayoutPrintEngine 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 InputSteam 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>

```
InputStream stream = getResources ().openRawResource ( R.raw.my_layout_file );
lpe.open ( stream );
```

2.3. close

<Definition>

```
void 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>

```
lpe.close ();
```

2.4. 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 LayoutPrintEngine 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>

```
lpe.beginPrint ();
```

2.5. endPrint

<Definition>

```
void 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>

```
lpe.endPrint ();
```

2.6. doPrint

<Definition>

```
int doPrint ( ESCPOSPrinter printer, boolean isReverse )
```

<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>

```
ESCPOSPrinter printer = new ESCPOSPrinter ();
printer.setEncoding ( "Shift_JIS" );
result = printer.connect ( ESCPOSConst.CMP_PORT_WiFi, "192.168.10.100" );
if ( ESCPOSConst.CMP_SUCCESS == result ) {
        lpe.doPrint ( printer, false );
        printer.disconnect ();
}
```

2.7. initFrame

<Definition>

```
int initFrame ( string 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

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>

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

2.8. setPartsData

<Definition>

```
int setPartsData ( int frameIndex, string partsName, string setText )
int setPartsData ( int frameIndex, string partsName, byte[] setData )
```

<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>

```
lpe.setPartsData ( frameIndex, "Text1", "New Text" );
```

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	:	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>

```
lpe.addFrame ( frameIndex );
```

3. Example of using the method (java)

```
import com.citizen.sdk.ESCPOSConst;              // Android POS Print SDK
import com.citizen.sdk.ESCPOSPrinter;            // Android POS Print SDK
import com.citizen.sdk.LayoutPrintEngine;        // Android Layout SDK
:
LayoutPrintEngine lpe = new LayoutPrintEngine();        // Android Layout SDK
InputStream stream = getResources().openRawResource( R.raw.my_layout_file );
int result = lpe.open( stream );
if ( 0 == result ) {
        lpe.beginPrint();
        int frameIndex = lpe.initFrame( "Frame1" );
        lpe.setPartsData( frameIndex, "Text1", "New Text" );
        lpe.addFrame( frameIndex );

        ESCPOSPrinter printer = new 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 );
                printer.disconnect();
        }

        lpe.endPrint();
        lpe.close();
}
try {
      stream.close();
} catch (IOException e) {
      e.printStackTrace();
}
```

- 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 Android POS Print SDK.

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

Android Layout SDK

Programming Manual
for Version 1.3.0