

# CITIZEN

## Android Layout SDK

プログラミング マニュアル  
Version 1.3.0 用

シチズン・システムズ株式会社

## 更新履歴

年月日	バージョン	履歴
2014.09.08	1.2.0.0	・新規 (Layout Utilities ユーザーズ・ガイド)
2016.08.16	1.3.0.0	・モバイル端末用レイアウトで直線部品と矩形部品に対応。 ・モバイル端末用レイアウトファイルを CLFX から XML へ変更。 ・「Layout Print Engine」から「Layout SDK」へ呼称を変更。 ・Layout SDK プログラマー向けにユーザーズ ガイドから分離。(本書)

## ご注意

1. 本書の内容の一部、または全部を無断で転載することは、固くお断りいたします。
2. 本書の内容については、事前の予告なしに変更することがあります。
3. 本書の内容については万全を期して作成いたしました。が、万一誤り・お気付きの点がございましたら、ご連絡くださいますようお願いいたします。
4. 運用した結果の影響につきましては、3項にかかわらず責任を負いかねますのでご了承ください。
5. 上記に同意いただけない場合は、本ライブラリをご使用いただけません。

## 著作権・商標について

- このプログラミング マニュアルの著作権は、シチズン・システムズ株式会社にあります。
  - CITIZEN は、シチズン時計株式会社の登録商標です。
  - Windows 及び Windows Server は、米国 Microsoft Corporation の米国およびその他の国における登録商標です。
  - Pentium は、米国およびその他の国における、Intel Corporation またはその子会社の商標または登録商標です。
  - QR コード、iQR コードは、(株)デンソーウェーブの登録商標です。
  - Android は、米国およびその他の国における Google Inc.の商標または登録商標です。
  - JavaScript は、米国およびその他の国における Oracle の商標または登録商標です。
- その他、記載されている会社名、製品名は、各社の商標または登録商標です。

## 目次

更新履歴 .....	2
ご注意 .....	3
著作権・商標について .....	3
目次.....	4
1. はじめに.....	5
1.1. ドキュメント対象範囲.....	5
1.2. システム概要.....	5
1.3. 対応 Android 端末 .....	6
1.4. 対応プリンター.....	6
1.5. 定義方法.....	6
1.6. 機能一覧.....	7
2. ライブラリ インターフェース .....	8
2.1. コンストラクタ.....	8
2.2. open メソッド.....	9
2.3. close メソッド .....	10
2.4. beginPrint メソッド .....	11
2.5. endPrint メソッド .....	12
2.6. doPrint メソッド .....	13
2.7. setFrame メソッド .....	14
2.8. setPartsData メソッド.....	15
2.9. addFrame メソッド .....	16
3. コード サンプル .....	17
4. 注意事項 .....	18

## 1. はじめに

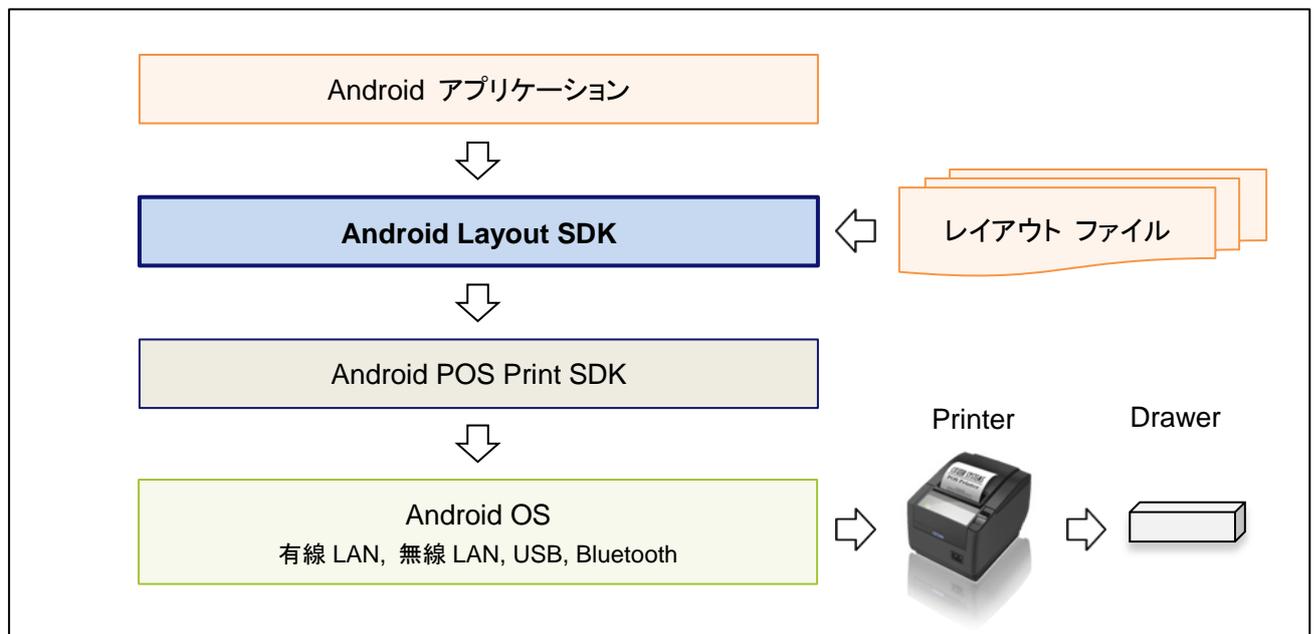
本ドキュメントは、**Android Layout SDK** のプログラミング マニュアルです。

### 1.1. ドキュメント対象範囲

本ドキュメントは、CITIZEN レイアウト ファイルを利用する Android アプリケーションの開発者が参照することを目的としています。

### 1.2. システム概要

本ライブラリは CITIZEN レイアウト ファイルを利用する Android アプリケーションから参照されることを想定しています。



ライブラリ システム 構成図

### 1.3. 対応 Android 端末

OS	Android 2.3.3(API 10) 以降 for Wi-Fi, Bluetooth interface
	Android 3.1(API 12) 以降 for Wi-Fi, Bluetooth, USB Host interface
動作に必要なソフトウェア	Android POS Print SDK Version 1.14 以降

Android POS Print SDK の詳細については、「Android POS Print SDK プログラミング マニュアル」をご参照ください。

### 1.4. 対応プリンター

本ライブラリの対象プリンターは、Android POS Print SDK が対応するプリンターとなります。

各プリンターの機能詳細については、プリンターの取り扱い説明書をご参照ください。

### 1.5. 定義方法

本ライブラリを使用する場合は、提供ファイルをライブラリ追加し、以下のクラスをインポートしてください。

```
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. 機能一覧

本ライブラリは以下の機能を提供します。

### メソッド一覧

No.	名称	機能
1	LayoutPrintEngine	コンストラクタです。インスタンスを生成します。
2	open	指定されたパス名の XML レイアウト ファイルを開きます。
3	close	現在開 いている XML レイアウト ファイルを閉じます。
4	beginPrint	印刷の準備を開始します。
5	endPrint	印刷データを破棄します。
6	doPrint	接続済みのプリンターで印刷を実行します。
7	initFrame	印刷用データを設定するフレームを検索します。 指定されたフレーム名で検索し、内部で管理しているフレーム番号を返します。
8	setPartsData	部品に印刷用データ(テキスト、バーコード、イメージ)を設定します。 initFrame()で取得したフレーム番号および部品名で、設定対象とする部品を検索し、指定された文字列を設定します。
9	addFrame	印刷用データが設定されたフレームを印刷対象として登録します。

---

## 2. ライブラリ インターフェース

### 2.1. コンストラクタ

#### <定義>

```
LayoutPrintEngine ()
```

#### <機能>

コンストラクタです。インスタンスを生成します。

#### <引数>

なし

#### <戻り値>

なし

#### <使用例>

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

## 2.2. open メソッド

### <定義>

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

### <機能>

InputStream オブジェクトとして渡された XML レイアウト ファイルを読み込みます。

読み込んだレイアウト ファイルの情報は、close() メソッドが呼び出されるまで LayoutPrintEngine クラス内で保持されます。

— POINT —

InputStream オブジェクトは本クラス内では閉じないため、呼び出し側で閉じる必要があります。

### <引数>

**stream**

テンプレートとなる XML レイアウト ファイルの InputStream オブジェクトを指定します。

### <戻り値>

- 0 : 正常に XML レイアウト ファイルを開くことができたことを意味します。
- 0 未満 : XML レイアウト ファイルの不正なフォーマットなど、何らかのエラーが発生したことを意味します。

### <使用例>

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

## 2.3. close メソッド

### <定義>

```
void close ()
```

### <機能>

`open()` メソッドで読み込んだレイアウト ファイルの情報を破棄します。

本メソッド呼出し後、`initFrame()` / `setPartsData()` / `addFrame()` / `doPrint()` メソッドはエラーとなります。

再度、`open()`メソッドを呼び出す必要があります。

### <引数・戻り値>

なし

### <使用例>

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

## 2.4. beginPrint メソッド

### <定義>

```
int beginPrint ()
```

### <機能>

印刷レイアウトの作成準備を行います。

本メソッド呼出し後に行う、`initFrame()` / `setPartsData()` / `addFrame()` メソッドにて作成した印刷レイアウトは、`endPrint()` メソッドが呼び出されるまで `LayoutPrintEngine` クラス内で保持されます。

### <引数・戻り値>

- 0 : 正常終了したことを意味します。
- 0未満 : 何らかのエラーが発生したことを意味します。

### <使用例>

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

## 2.5. endPrint メソッド

### <定義>

```
void endPrint ()
```

### <機能>

印刷レイアウトを破棄します。

本メソッド呼出し後は、doPrint() メソッドによる印刷はできません。

再度、beginPrint() メソッドを呼び出し、印刷レイアウトを作成する必要があります。

### <引数・戻り値>

なし

### <使用例>

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

## 2.6. doPrint メソッド

### <定義>

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

### <機能>

作成済みの印刷レイアウトを指定プリンターで印刷します。

#### — POINT —

ESCPOSPrinter.connect () メソッドにて接続が確立した ESCPOSPrinter オブジェクトを指定する必要があります。

### <引数>

#### **printer**

ESCPOSPrinter オブジェクトを指定します。

ESCPOSPrinter クラスは Android POS Print SDK が提供するクラスです。

#### **isReverse**

上下反転印刷を指定します。

true : 上下反転 (180 度回転) で逆順印刷 (最終フレームから出力) します。

false : 正順印刷 (先頭フレームから出力) します。

### <戻り値>

- 0 : 正常終了したことを意味します。
- 0 超 : ESCPOSPrinter クラスのエラーコードです。印刷中に問題が発生したことを意味します。
- 0 未満 : 何らかのエラーが発生したことを意味します。

### <使用例>

```
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. setFrame メソッド

### <定義>

```
int setFrame ( string frameName )
```

### <機能>

印刷用データを設定するフレームを検索します。

指定されたフレーム名で検索し、内部で管理しているフレーム番号を返します。

### <引数>

#### **frameName**

対象とするフレーム名を指定します。

### <戻り値>

1 以上 : 内部で管理しているフレーム番号を意味します。

0 未満 : 指定されたフレームが見つからないなど、何らかのエラーが発生したことを意味します。

### <使用例>

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

## 2.8. setPartsData メソッド

### <定義>

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

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

### <機能>

部品に印刷用データ(テキスト、バーコード、イメージ)を設定します。

`initFrame()`で取得したフレーム番号と部品名で、

設定対象とする部品を検索し、指定された文字列またはデータを設定します。

### <引数>

#### **frameIndex**

対象とするフレーム番号 (`initFrame()`で取得したものを)指定します。

#### **partsName**

対象とする部品名を指定します。

#### **setText**

設定する印刷用データ(文字列)を指定します。テキスト／バーコード部品に適用します。

#### **setData**

設定する印刷用データ(byte 配列)を指定します。イメージ部品に適用します。

### <戻り値>

0 : 正常終了したことを意味します。

0 未満 : 指定された部品が見つからないなど、何らかのエラーが発生したことを意味します。

### <使用例>

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

## 2.9. addFrame メソッド

### <定義>

```
int addFrame ( int frameIndex )
```

### <機能>

印刷用データが設定されたフレームを、印刷対象として登録します。

### <引数>

#### **frameIndex**

対象とするフレーム番号 (`initFrame()` で取得したものを) 指定します。

### <戻り値>

- 0 : 内部で管理している印刷登録時のフレーム番号を意味します。
- 0未満 : 指定されたフレームが見つからないなど、何らかのエラーが発生したことを意味します。

### <使用例>

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

### 3. コード サンプル ( 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();
}
```

—POINT—

詳しくはレイアウト SDK サンプル プログラムをご参照ください。

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

## 4. 注意事項

本ライブラリの注意事項を以下に示します。

### 3.1. 印刷完了確認について

本ライブラリは、Android POS Print SDK が提供する印刷完了確認機能を利用しています。印刷完了確認機能の詳細については、「Android POS Print SDK プログラミング マニュアル」をご参照ください。

## **Android Layout SDK**

プログラミング マニュアル  
Version 1.3.0 用