

# CITIZEN

モバイル プリンター

Android モジュール プログラム ガイド

ESC/POS, CPCL

Ver. 1.00

**シチズン・システムズ株式会社**

## 更新履歴

日付	バージョン	対象 SDK	履歴
2012/11/29	0.08		新規
2014/03/18	1.00	1.064	USB インターフェース対応

# 1. 目次

Android モジュール プログラム ガイド .....	0
更新履歴 .....	1
1. 目次 .....	2
2. はじめに .....	3
3. 全体の流れ .....	4
4. 詳細 .....	5
4.1. プリンターと接続 .....	5
4.1.1. コード例 Bluetooth インターフェース利用の場合 .....	6
4.1.2. コード例 Wi-Fi インターフェース利用の場合 .....	7
4.1.3. コード例 USB インターフェース利用の場合 .....	8
4.2. プリンター ステータスを取得確認 .....	10
4.2.1. コード例 ESC/POS コマンド利用の場合 .....	11
4.2.2. コード例 CPCL コマンド利用の場合 (CMP-20 非対応) .....	12
4.3. 印刷データを送信 .....	13
4.3.1. コード例 ESC/POS コマンドを利用の場合 .....	14
4.3.2. コード例 CPLC コマンドを利用の場合 .....	15
4.4. 印刷データの送信完了を待つ .....	16
4.4.1. コード例 .....	16
4.5. プリンターとの接続を切断する .....	17
4.5.1. コード例 Bluetooth インターフェース利用の場合 .....	17
4.5.2. コード例 Wi-Fi インターフェース利用の場合 .....	18
4.5.3. コード例 USB インターフェース利用の場合 .....	19

## 2. はじめに

この「Android モジュール プログラム ガイド」では、Android モバイル アプリケーションの開発に必要な Jar パッケージファイルから得られるメソッドの利用する上で全体の流れや注意点について説明します。

プリンターの仕様付き不明の点は、CMP-20/30 の技術マニュアル、ESC/POS のコマンドマニュアル、CPCL のコマンドマニュアルを参考にして頂きますよう、お願いいたします。

- Note -

- ・ESC/POS コマンドと CPCL コマンドの選択の目安、  
レシートのように印刷する長さが変化する場合は ESC/POS コマンドを、  
ラベル用紙やプリプリント用紙のように印刷する長さが固定の場合は CPCL コマンドを、  
選択してください。  
なお、CMP-20 は CPCL コマンド非対応となります。ご注意ください。

- Note -

- ・CMP-30 で利用できる ESC/POS コマンドと CPCL コマンドの選択は、Windows 上で動作するユーティリティ上で設定します。

- Note -

- ・本ガイドに掲載するコードは、一例であり、動作を保証するためのコードではありません。また、バージョンアップにより仕様が変わる場合があります。お客様の環境で十分に評価の上、ご利用ください。

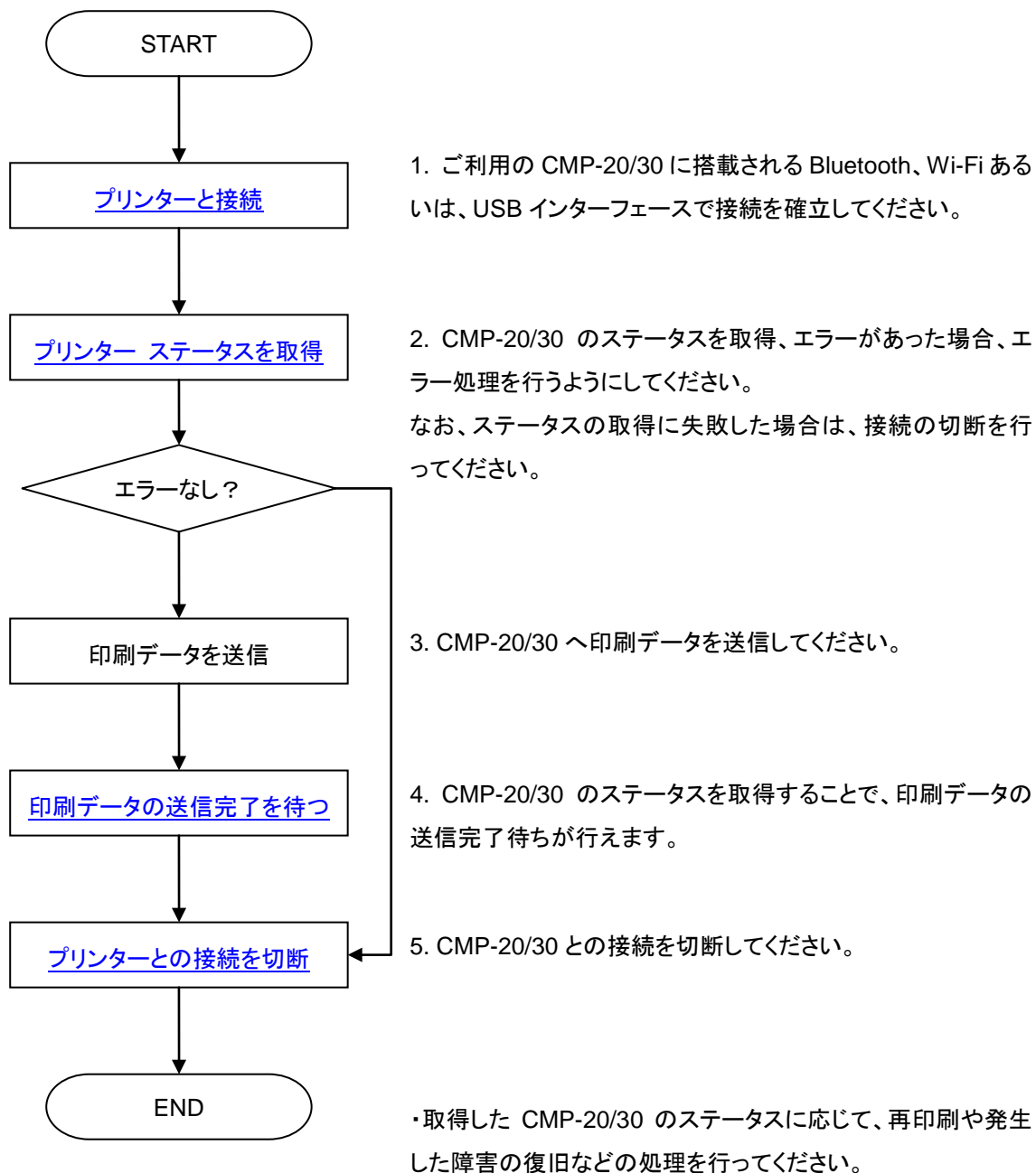
- Note -

- ・一部の Android4.2.X 端末において、Bluetooth 接続ができない現象を確認しています。  
お客様の環境で十分に評価の上、ご利用ください。

New

### 3. 全体の流れ

以下に CMP-20/30 で印刷を行う際の全体の流れを示します。



- Note -

・「プリンターとの接続を切断」した直後に、「プリンターと接続」を行った場合、接続に失敗する場合があります。切断と接続が連続する場合は、切断後に 500ms 以上の待ち時間を設定してください。

## 4. 詳細

### 4.1. プリンターと接続

アプリケーションは、getInstance() メソッドを利用してインスタンスを取得し、connect() メソッドでプリンターと接続してください。プリンターと接続できなかった場合、connect() メソッドは例外エラーを返しますので、try/catch で補足してください。

New

USB インターフェース利用の場合、Bluetooth、Wi-Fi インターフェースとは異なった処理が必要となります。USBPort クラスの connect\_device() メソッドでプリンターと接続してください。プリンターと接続できなかった場合、connect\_device() メソッドの戻り値 USBPortConnection に null を返します。

#### 【利用するクラス】

BluetoothPort クラス、または、WiFiPort クラス、または、USBPort クラスと USBPortConnection クラス

#### - Note -

- ・1 台の Android 端末と、複数台の CMP-20/30 と、を同時に接続することはできません。
- また、複数台の Android 端末と、1 台の CMP-20/30 と、を同時に接続することはできません。
- 上記のような複数台が共有して運用したい場合は、1 台の Android 端末と、1 台の CMP-20/30 と、だけが接続されるようにして運用してください。

#### - Note -

- ・CMP-20/30 を無線でご利用いただく場合、ご利用環境によって接続が不安定な場合があります。
- もし、そのような環境で接続が途切れた場合は、接続を切断し、接続からやり直す必要があります。

#### 4.1.1. コード例 Bluetooth インターフェース利用の場合

```
private com.citizen.port.android.BluetoothPort port;

private java.lang.Thread hThread = null;

// インスタンスを取得
port = com.citizen.port.android.BluetoothPort.getInstance();

boolean bConnected = false;

long lConnectTimeout = 10000; // 接続タイムアウト (10sec)
long lStartTime = System.currentTimeMillis();
do {
    try {
        port.connect("12:34:56:78:90:AB"); // 接続 (Bluetooth アドレス)
        bConnected = true; // 接続成功
        break;
    } catch (IOException e) {
        try {
            try {
                port.disconnect();
            } catch (IOException e1) {}
            java.lang.Thread.sleep(100); // 再接続のためのウェイト(0.1sec)
        } catch (InterruptedException e2) {}
    }
} while (lConnectTimeout > System.currentTimeMillis() - lStartTime);
if (!bConnected) {
    return; // 接続失敗
}

// RequestHandler を開始
hThread = new java.lang.Thread( new
    com.citizen.request.android.RequestHandler());
hThread.start();
```

##### - Note -

- ・電波状況により接続エラーとなる場合があります。上記コード例のようにリトライを行ってください。
- ・Bluetooth アドレスは、12 文字、かつ、アルファベットは大文字、で指定してください。

#### 4.1.2. コード例 Wi-Fi インターフェース利用の場合

```
private com.citizen.port.android.WiFiPort port;

private java.lang.Thread hThread = null;

// インスタンスを取得
port = com.citizen.port.android.WiFiPort.getInstance();

boolean bConnected = false;

long lConnectTimeout = 10000; // 接続タイムアウト(10sec)
long lStartTime = System.currentTimeMillis();
do {
    try {
        port.connect("192.168.0.10"); // 接続 (IP アドレス)
        bConnected = true; // 接続成功
        break;
    } catch (IOException e) {
        try {
            try {
                port.disconnect();
            } catch (IOException e1) {}
            java.lang.Thread.sleep(100); // 再接続のためのウェイト(0.1sec)
        } catch (InterruptedException e2) {}
    }
} while (lConnectTimeout > System.currentTimeMillis() - lStartTime);
if (!bConnected) {
    return; // 接続失敗
}

// RequestHandler を開始
hThread = new java.lang.Thread( new
    com.citizen.request.android.RequestHandler());
hThread.start();
```



#### 4.1.3. コード例 USB インターフェース利用の場合

All New

```
private android.content.Context mContext;

private android.hardware.usb.UsbManager mUsbManager;

private com.citizen.port.android.USBPort mUsbPort;

private com.citizen.port.android.USBPortConnection mUsbPortConnection;

private static final String ACTION_USB_PERMISSION = "com.your.app.USB_PERMISSION";

mContext = this.getActivity().getApplicationContext();
mUsbManager = (UsbManager)mContext.getSystemService(Context.USB_SERVICE);
mUsbPort = new com.citizen.port.android.USBPort(mUsbManager, mContext);
if (mUsbPortConnection == null) {
    // BroadcastReceiver 登録
    context.registerReceiver(mUsbReceiver, new IntentFilter(ACTION_USB_PERMISSION));
    // 接続
    mUsbPortConnection = mUsbPort.connect_device(USBPort.CMP_PORT_USB);
    if (mUsbPortConnection == null) {
        return; // 接続失敗
    }
}

private final BroadcastReceiver mUsbReceiver = new BroadcastReceiver() {
    public void onReceive (Context context, Intent intent) {
        String action = intent.getAction();
        if (ACTION_USB_PERMISSION.equals(action)) {
            synchronized (this) {
                UsbDevice usbDevice =
                    (UsbDevice)intent.getParcelableExtra(UsbManager.EXTRA_DEVICE);
                if (intent.getBooleanExtra(UsbManager.EXTRA_PERMISSION_GRANTED, false)) {
                    if (usbDevice != null) {
                        // USB デバイス接続許可
                    }
                } else {
                    // USB デバイス接続許可を拒否
                }
            }
        }
        mContext.unregisterReceiver(mUsbReceiver); // BroadcastReceiver 登録を削除
    }
}
```

- Note -

・CMP-20/30 のパワーセーブモードを無効に設定してご利用ください。

CMP-20/30 の USB インターフェースはパワーセーブモード非対応です。

・電源オン後や USB ケーブル接続後の初回の connect\_device 接続は失敗します。

その際に表示される USB デバイス接続許可ダイアログの OK ボタンをタップしてください。この手動操作により USB デバイスとの接続が許可され、はじめて connect\_device 接続が成功します。

また、この許可した状態は Android 端末に保存されないため、運用時に注意が必要です。

以下は表示される USB デバイス接続許可ダイアログの一例です。



#### 4.2. プリンター ステータスを取得確認

アプリケーションは、印刷の前後において、printerCheck() メソッド、status() メソッドを順に実行し、プリンターの状態を現す status() メソッドの戻り値を取得し、エラーなし状態であることを確認してください。

##### 【利用するクラス】

ESCPOSPrinter クラス、または、CPCLPrinter クラス

##### - Note -

- ・CMP-20/30 がパワーセーブモード中は、印刷等のメソッドが正常に機能しません。必ず事前にパワーセーブモードを解除してください。パワーセーブモードの解除は、printerCheck() メソッドの実行により解除することができます。(Ver. 1055 以降)

##### - Note -

- ・CMP-20/30 のパワーセーブモードの設定は、Windows 上で動作するユーティリティで設定します。

New

##### - Note -

- ・USB インターフェース利用の場合、ESCPOSPrinter クラス、または、CPCLPrinter クラスをインスタンス化する際、第二引数に USBPortConnection を必ず指定してください。

以下に一例を示します。

```
com.citizen.jpos.printer.ESCPOSPrinter printer = new  
com.citizen.jpos.printer.ESCPOSPrinter("ISO-8859-1", mUsbPortConnection);
```

#### 4.2.1. コード例 ESC/POS コマンド利用の場合

```
// プリンター ステータスの取得と確認
com.citizen.jpos.printer.ESCPOSPrinter printer = new
com.citizen.jpos.printer.ESCPOSPrinter("ISO-8859-1");
if (ESCPOSConst.CMP_SUCCESS == printer.printerCheck()) {
    int status = printer.status();
    if (ESCPOSConst.CMP_STS_NORMAL == status) {
        // エラーなし状態
    } else {
        if ((ESCPOSConst.CMP_STS_MSR_READ & status) > 0) {
            // MSR 読み取りモード状態
        }
        if ((ESCPOSConst.CMP_STS_PAPER_EMPTY & status) > 0) {
            // 用紙なし状態
        }
        if ((ESCPOSConst.CMP_STS_COVER_OPEN & status) > 0) {
            // カバー オープン状態
        }
        if ((ESCPOSConst.CMP_STS_BATTERY_LOW & status) > 0) {
            // バッテリー容量低下状態
        }
    }
} else {
    // ステータス取得に失敗
}
```

- Note -

・エラー処理はそれぞれ実装してください。

#### 4.2.2. コード例 CPCL コマンド利用の場合 (CMP-20 非対応)

```
// プリンター ステータスの取得と確認
com.citizen.jpos.printer.CPCLPrinter printer = new
com.citizen.jpos.printer.CPCLPrinter("ISO-8859-1");
if (CPCLConst.CMP_SUCCESS == printer.printerCheck()) {
    int status = printer.status();
    if (CPCLConst.CMP_STS_CPCL_NORMAL == status) {
        // エラーなし状態
    } else {
        if ((CPCLConst.CMP_STS_CPCL_BUSY & status) > 0) {
            // ビジー状態
        }
        if ((CPCLConst.CMP_STS_CPCL_PAPER_EMPTY & status) > 0) {
            // 用紙なし状態
        }
        if ((CPCLConst.CMP_STS_CPCL_COVER_OPEN & status) > 0) {
            // カバー オープン状態
        }
        if ((CPCLConst.CMP_STS_CPCL_BATTERY_LOW & status) > 0) {
            // バッテリー容量低下状態
        }
    }
} else {
    // ステータス取得に失敗
}
```

- Note -

・エラー処理はそれぞれ実装してください。

#### 4.3. 印刷データを送信

アプリケーションは、『[プリンター ステータスを取得確認](#)』した後に各種メソッドを利用して印刷データを送信してください。

##### 【利用するクラス】

ESCPOSPrinter クラス、または、CPCLPrinter クラス

##### - Note -

- ・日本語 を印刷したい場合は、ESCPOSPrinter、または、CPCLPrinter の引数 キャラクターセットに "Shift\_JIS" を指定してください。

##### - Note -

- ・USB インターフェース利用の場合、ESCPOSPrinter クラス、または、CPCLPrinter クラスをインスタンス化する際、第二引数に USBPortConnection を必ず指定してください。

以下に一例を示します。

```
com.citizen.jpos.printer.ESCPOSPrinter printer = new  
com.citizen.jpos.printer.ESCPOSPrinter("ISO-8859-1", mUsbPortConnection);
```

New

#### 4.3.1. コード例 ESC/POS コマンドを利用の場合

```
// 印刷

com.citizen.jpos.printer.ESCPOSPrinter printer = new
com.citizen.jpos.printer.ESCPOSPrinter("ISO-8859-1");

try {
    // テキストを印刷
    printer.printNormal("Barcode CODE39 :\n");

    // CODE39 を印刷
    printer.printBarCode("0123456789", CMPPrint.CMP_BCS_Code39, 8*10, 2,
        CMPPrint.CMP_ALIGNMENT_CENTER, CMPPrint.CMP_HRI_TEXT_BELOW);

    // 用紙を 5mm フィード
    printer.printNormal((char)ESCPOS.ESC + " |40uF\n");
}

catch (UnsupportedEncodingException e) {
    // エラー
}
```

- Note -

- ESC/POS コマンドは、行単位で印刷処理を行います。このため順次印刷処理が行われます。

#### 4.3.2. コード例 CPLC コマンドを利用の場合

```
// 印刷
com.citizen.jpos.printer.CPCLPrinter printer = new
com.citizen.jpos.printer.CPCLPrinter("ISO-8859-1");

// 用紙のフォームを設定 ( オフセット=0, 縦横解像度=200dpi, ラベル長=25mm, 印刷回数=1 )
printer.setForm(0, 200, 200, 8*25, 1);

// 用紙の種類を設定 ( ラベル用紙 )
printer.setMedia(CPCLConst.CMP_CPCL_LABEL);

// フォームへテキストを配置
printer.printCPCLText(CPCLConst.CMP_CPCL_NO_ROTATION, 5, 0, 30, 0, "CITIZEN
SYSTEMS URL", 0);

// フォームへ QRcode バーコードを配置
printer.printCPCL2DBarcode(CPCLConst.CMP_CPCL_NO_ROTATION,
CPCLConst.CMP_CPCL_BCS_QRCODE, 30, 40, 4, 0, 1, 8,
"http://citizen-systems.co.jp");

try {
    // 印刷開始
    printer.printForm();
}

catch (UnsupportedEncodingException e) {
    // エラー
}
```

##### - Note -

・CPCL コマンドは、ページ単位で印刷処理を行います。このため 1 ページ分の印刷データの送信を完了するまで印刷処理が開始されません。



#### 4.4. 印刷データの送信完了を待つ

アプリケーションは、印刷データを送信する各メソッド実行後、印刷データの送信の完了を確認するために `printerCheck()` メソッドを実行してください。

##### 【利用するクラス】

ESCPOSPrinter クラス、または、CPCLPrinter クラス

##### 4.4.1. コード例

コード例については、『[プリンター ステータスを取得確認](#)』を参照してください。

#### 4.5. プリンターとの接続を切断する

アプリケーションは、印刷の終了、あるいは、何らかのエラーが発生した後に disconnect() メソッドを実行して接続を切断してください。

New

USB インターフェース利用の場合、Bluetooth、Wi-Fi インターフェースとは異なった処理が必要となります。USBPortConnection クラスの close() メソッドを実行して接続を切断してください。

##### 【利用するクラス】

BluetoothPort クラス、または、WiFiPort クラス、または、USBPortConnection クラス

##### 4.5.1. コード例 Bluetooth インターフェース利用の場合

```
private com.citizen.port.android.BluetoothPort port;

// インスタンスを取得
port = com.citizen.port.android.BluetoothPort.getInstance();
try {
    // 切断
    port.disconnect();
}
catch (IOException e) {
    // エラー
}
catch (InterruptedException e) {
    // エラー
}
if ((hThread != null) && (hThread.isAlive())) {
    hThread.interrupt();
    hThread = null;
}
```

#### 4.5.2. コード例 Wi-Fi インターフェース利用の場合

```
private com.citizen.port.android.WiFiPort port;

// インスタンスを取得
port = com.citizen.port.android.WiFiPort.getInstance();
try {
    // 切断
    port.disconnect();
}
catch (IOException e) {
    // エラー
}
catch (InterruptedException e) {
    // エラー
}
if ((hThread != null) && (hThread.isAlive())) {
    hThread.interrupt();
    hThread = null;
}
```

#### 4.5.3. コード例 USB インターフェース利用の場合

```
if (mUSBPortConnection != null) {  
    try {  
        // 切断  
        mUSBPortConnection.close();  
        mUSBPortConnection = null;  
    }  
    catch (InterruptedException e) {  
        // エラー  
    }  
}
```

All New