Android Software Development Kit

DK-AirCash 向け StarIO 使用方法

本 SDK には Android デバイスのための Java Eclipse プロジェクトを含んでいます。

Android バージョン	対応インターフェイス	適用 StarIO JAR ファイル
Android OS 3.1 以上	Bluetooth, 有線 LAN/無線 LAN	StarIOPort3.1.jar (Default)
Android OS 2.2, 2.3, 3.0	Bluetooth, 有線 LAN/無線 LAN	StarIOPort3.1.jar (Default)
Android OS 2.1	Bluetooth, 有線 LAN/無線 LAN	StarIOPort.jar

必要なツール:

- JDK 6
- Eclipse
- Android SDK Manager
- ADT (Android Development Tool) Plugin for Eclipse
- USB Windows Driver by Android Device Manufacturer
- StarIO Android SDK

この SDK は、有線 LAN/無線 LAN インターフェースの場合、DK-AirCash ファームウェアバージョン 2.0 以降に対応しています。
Bluetooth インターフェースの場合、 ファームウェアバージョン 1.0 から対応しています。

目次

- 本書に関して
- Star DK-AirCash を Android デバイスに接続するには
- StarIO Android SDK を実行するためには
- Star DK-AirCash と SDK を使用する
- Android SDK 概要
- StarIO - (StarIOPort3.1.jar)
- StarIOAndroid SDK 機能
  - Help
  - Get Printer Status / Get Cash Drawer Status
  - Get Printer Firmware Information / Get Cash Drawer Firmware
  - Get Cash Drawer Dip Switch Information
  - Sample Receipt + Open Cash Drawer
  - Open Cash Drawer
- StarIO を使用するアプリケーション開発のために
- 追加リソース
- ASCII コード表
- SDK マニュアル 改訂履歴

本書に関して

このマニュアルは、StarIO とスター DK-AirCash が通信を行う Android アプリケーションの作成方法を解説しています。

また、このマニュアルは、Java 言語を使用するアプリケーション・システム開発者を対象に作成しており、利用者は Java 言語の基礎を理解していることを前提としています。

この SDK は Android 用に作成されています。
スター精密グローバルサポートサイトの Developers セクションには、その他のオペレーティングシステムとプログラミング言語に利用可能な SDK が用意されています。最新の SDK、テクニカルドキュメント、FAQ、及び、その他の追加情報については、Developers セクションをご確認ください。

表示マークの説明:

警告		潜在的な問題について説明します。
禁止		禁止事項について説明します。
メモ		重要な情報とヒントを提供します。

注意事項:

- Android は、Google Inc.の商標または登録商標です。
- 本マニュアルの内容は、予告無く変更する場合があります。
- スター精密株式会社は、正確な情報を提供するためにあらゆる措置を取っていますが、誤りや不作為について責任を負うものではありません。
- スター精密株式会社は、このマニュアルに記載されている情報の使用に起因するいかなる損害に対しても責任を負うものではありません。
- 本マニュアルの一部、あるいは全部を無断で複写・複製・転載することは、固くお断りします。

© 2012-2015 Star Micronics Co., Ltd.

Star デバイスを Android デバイスに接続するには

以下は Android 4.0 を搭載したタブレットでの設定方法です。
デバイスの種類や OS のバージョンによって、表示が異なる場合があります。

有線 LAN/無線 LAN インターフェイス

Star DK-AirCash は、工場出荷時の初期設定は DHCP が有効になっています。使用するネットワークが DHCP をサポートする場合、本製品が自動的に IP アドレスを取得できるように、必要なネットワーク構成を構築してください。

また、本製品の#9100 Multi Session を無効に設定して使用してください。

＃9100 Multi Session の確認・変更、固定 IP アドレスの設定方法については、こちらのリンクの「対応 OS・環境一覧 ＞ ユーティリティ一覧」より「イーサネットプリンター利用手引き」をご参照ください。固定 IP アドレスの設定に必要となる製品固有の情報は、製品に付属のアドレス印字ラベルに記載されています。

TCP/IP 接続が可能な DK-AirCash の設定を行う場合、スター精密の提供する Star Setting Utility*を使用して設定することもできます。Star Setting Utility は Google play よりダウンロードしてください。

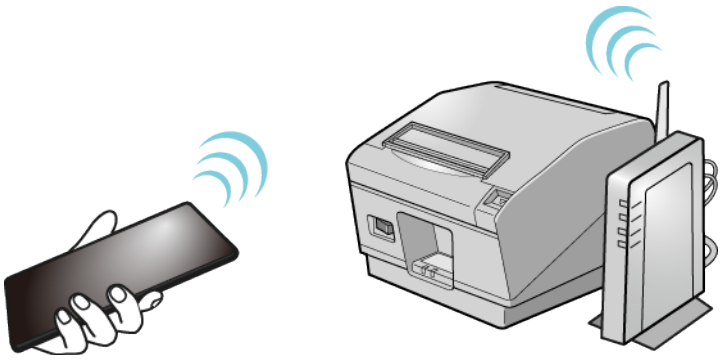
*Star Setting Utility は、IP アドレスの割り当てられていない（IP アドレス：0.0.0.0）状態では使用できません。

◆Android デバイスの接続設定

1. Star POS デバイスに IP アドレスを割り当て、ネットワークに接続します。
2. 「設定」をタップします。
3. 「Wi-Fi」を ON に設定します。



4. Star POS デバイスと同じネットワークに接続します。



Bluetooth

Star POS デバイスは、工場出荷時の初期設定では各機種ごとに"DK-AirCash"等、共通の Bluetooth デバイス名が設定されています。同じ Bluetooth デバイス名の機種を複数台配置して運用される場合、Bluetooth デバイス名の変更を行うと Star POS デバイスの判別が付けやすく便利です。

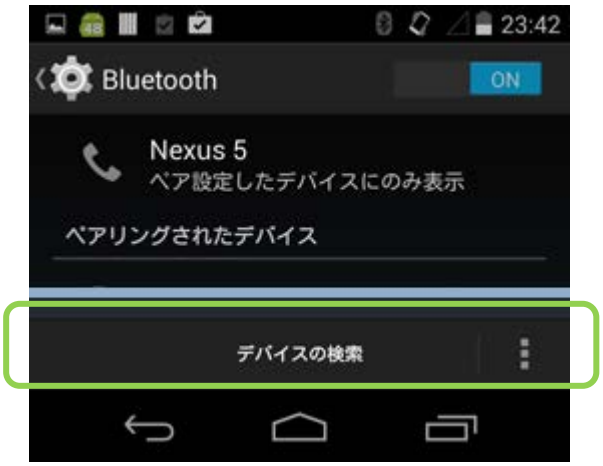
Bluetooth デバイス名の変更等、Star POS デバイスの有線 LAN/無線 LAN/Bluetooth 設定値は、スター精密の提供する Star Setting Utility を使用して変更することができます。Star Setting Utility は Google play よりダウンロードしてください。

◆ペアリング

1. 設定を行う Star POS デバイスの Bluetooth が有効になっていることを確認して、デバイスの電源を投入します。
   Star POS デバイスのセキュリティ方式が SSP の場合は、PAIR ボタンを 5 秒以上押してペアリング可能にします。
2. Android デバイスにて「設定」をタップします。
3. 「Bluetooth」を ON に設定します。



4. 「デバイスの検索」をタップします。

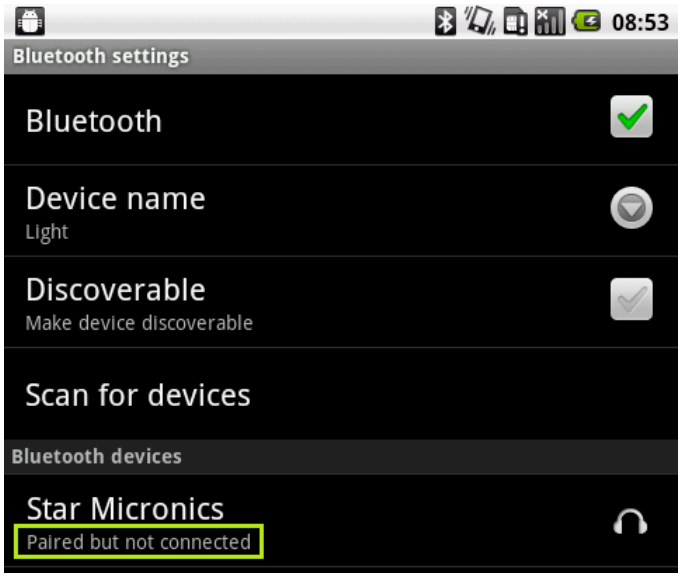


Bluetooth デバイスのスキャンが行われ、見つかったデバイスが表示されます。
設定を行う Star POS デバイスのデバイス名をタップします。

5. PIN Code を使用する場合、PIN 番号（パスキー）を入力します。工場出荷時の初期値は“1234”です。(SSP の場合はこの操作は必要ありません。)



6. Android V2.2 などの古い Android OS をご使用の場合、“Paired but not connected”と表示されますが、問題ありません。



StarIO Android SDK を実行するためには

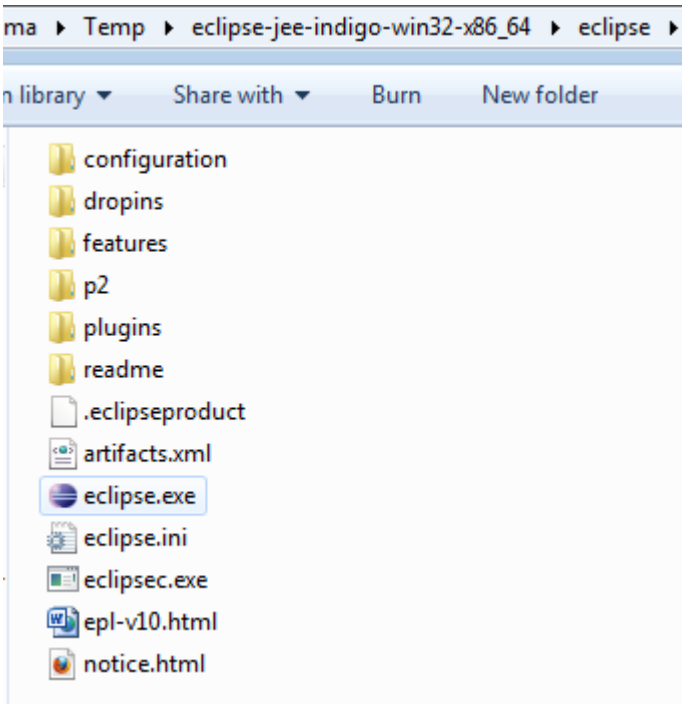
Android アプリケーションを開発するためには、Eclipse がインストールされている必要があります。Android アプリケーション開発環境の設定については、本 SDK の“documentation”フォルダー内の“AndroidDevelopmentSetup_Jp.pdf”をご参照ください。

このセクションでは、以下の項目について説明します。

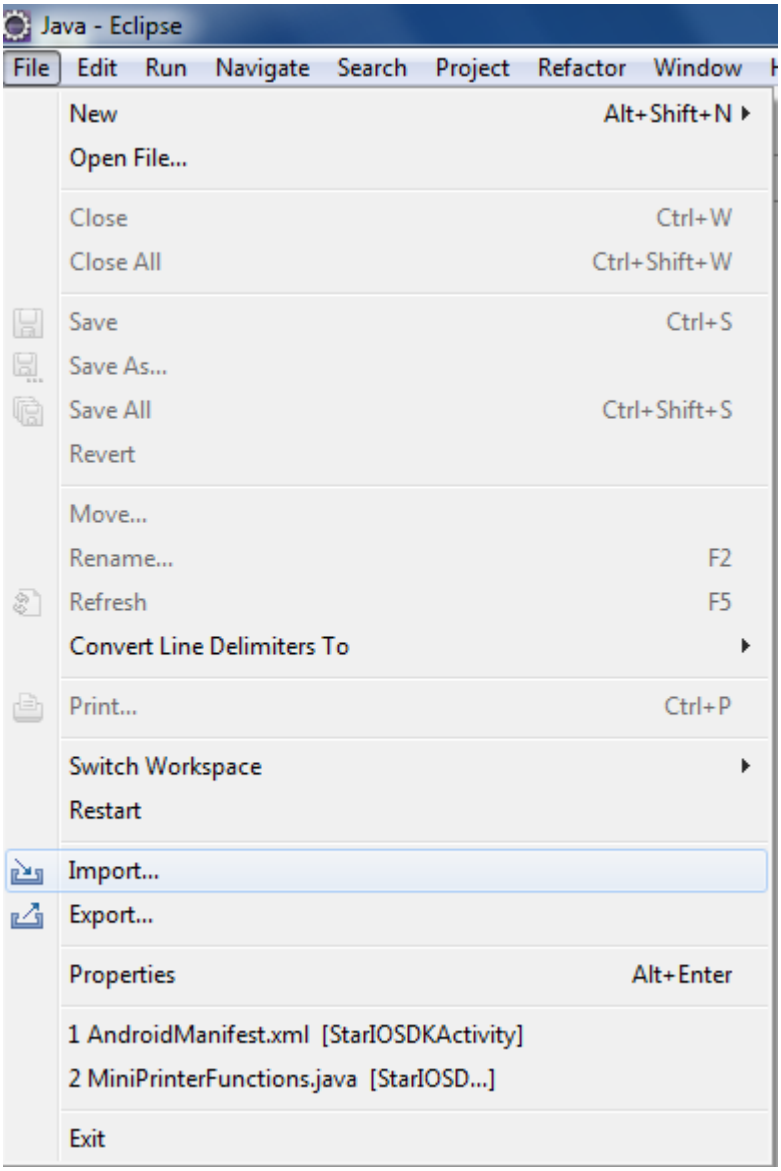
1. Android SDK プロジェクトの Eclipse への追加
2. Eclipse のデバッグモード設定
3. プロジェクトの実行

Android SDK プロジェクトの Eclipse への追加

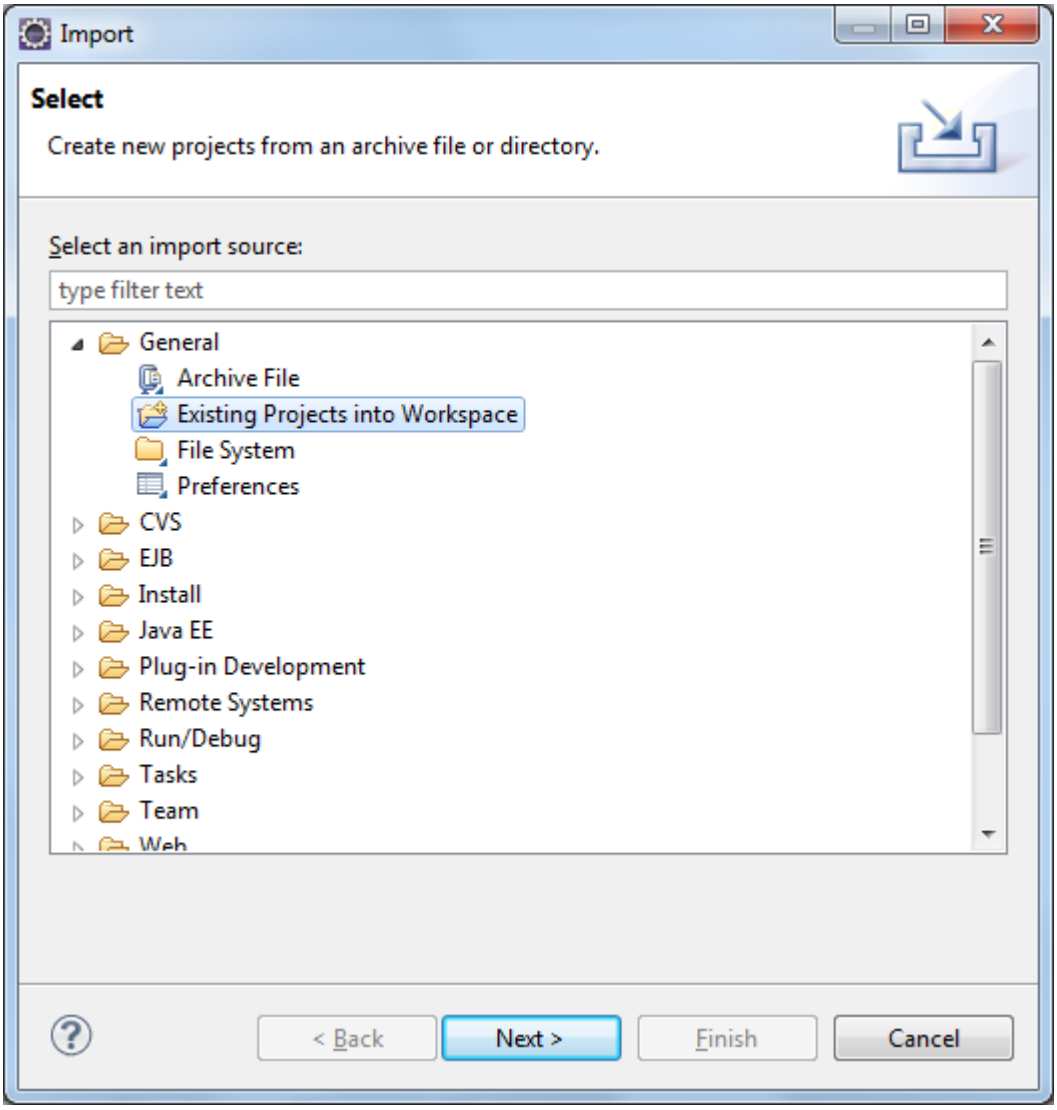
1. Eclipse を起動します。



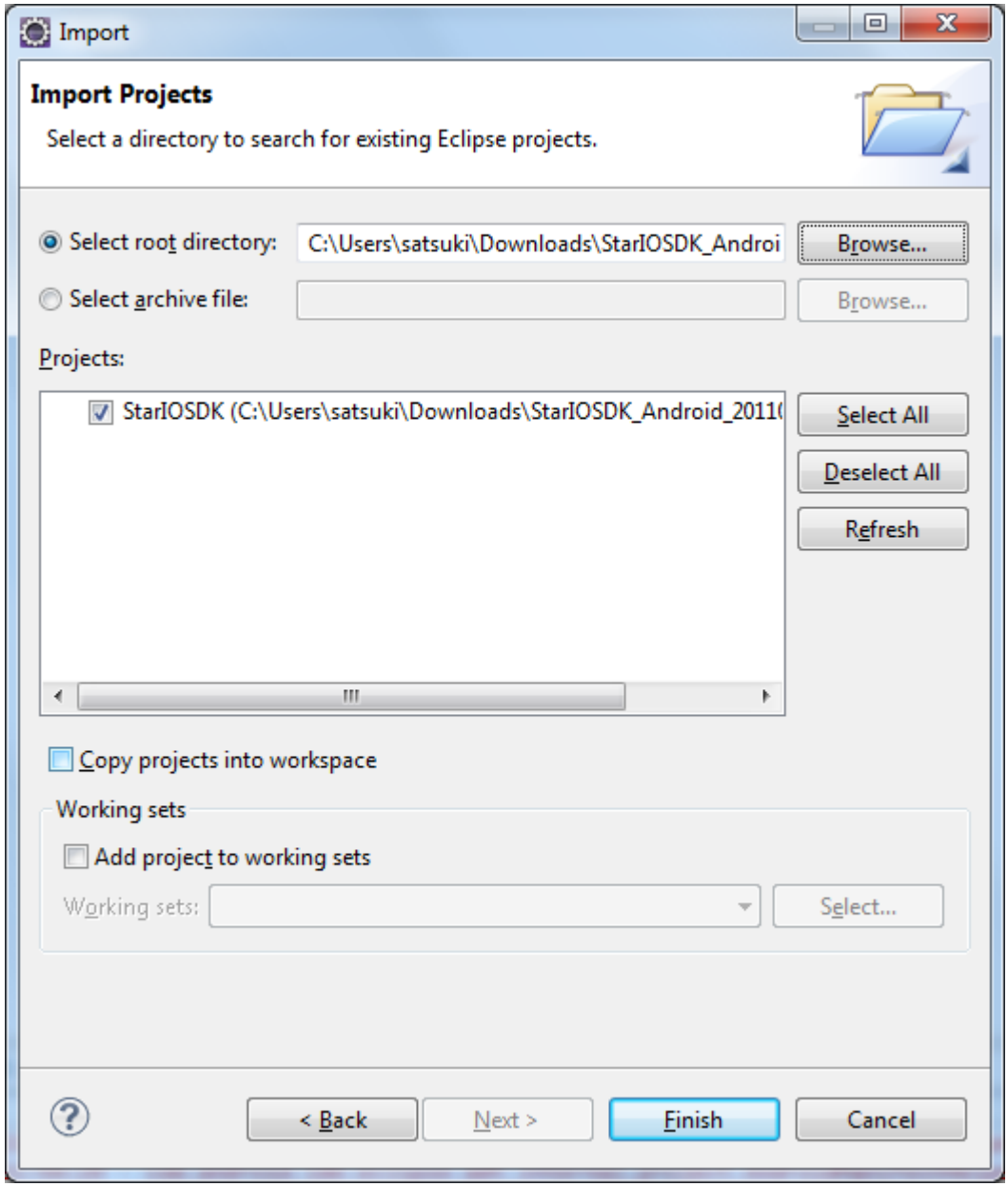
2. File → Import をクリックします。



3. General → Existing Projects into Workspace を選択し、「Next」をクリックします。

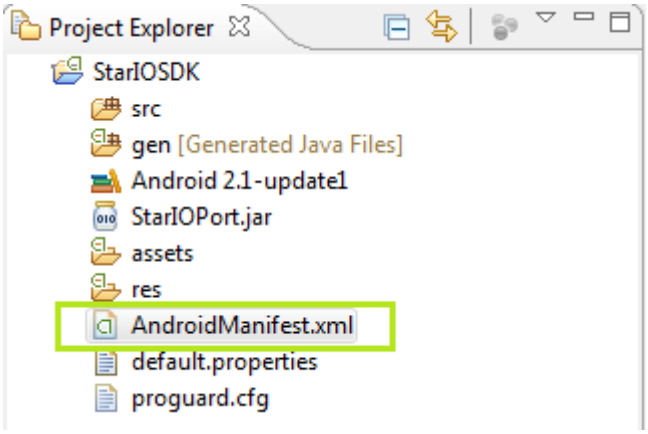


4. Star Android SDK を解凍したフォルダーを参照し、「Finish」をクリックします。

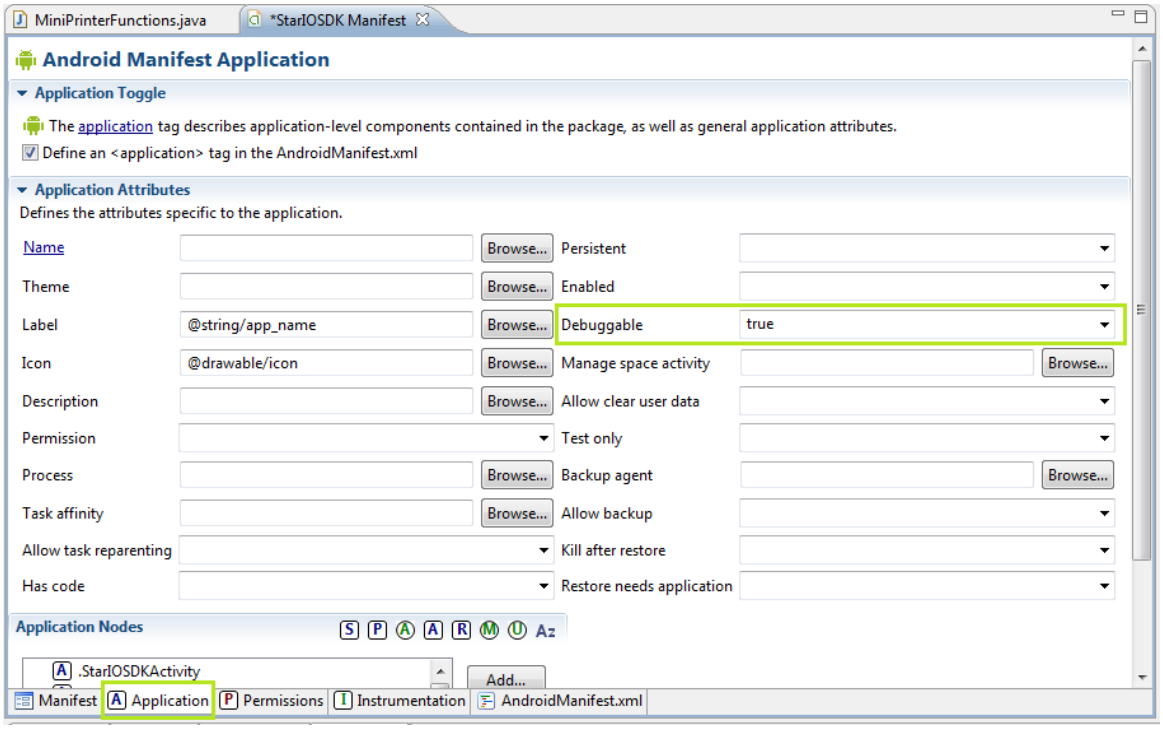


Eclipse のデバッグモード設定

1. “AndroidManifest.xml”をクリックします。

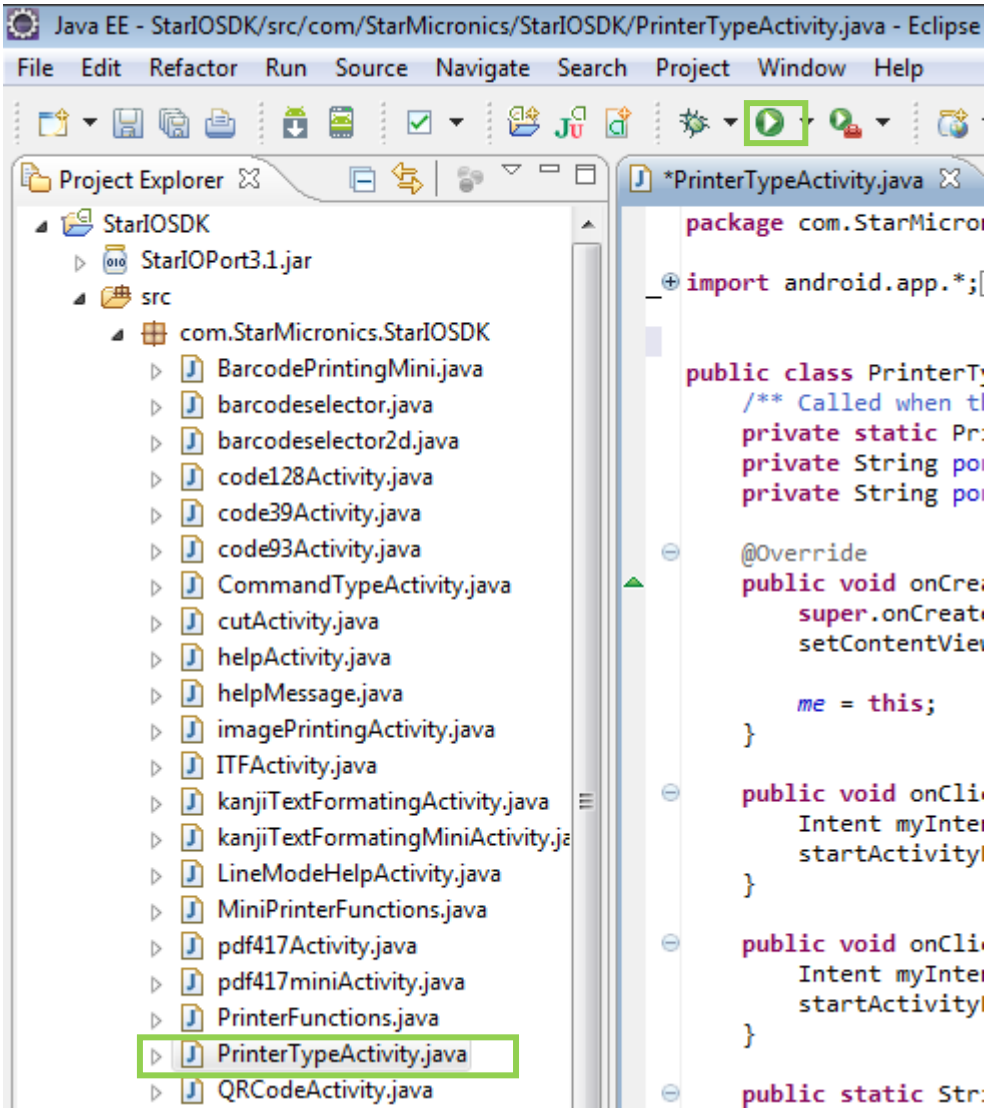


2. “Click the Application”タブをクリックします。
   “Debuggable”に true を設定します。

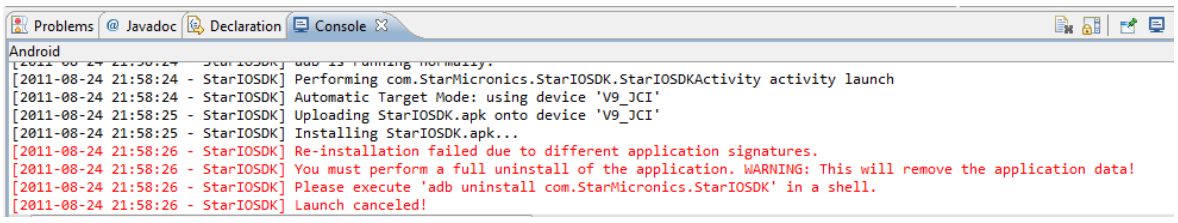


プロジェクトの実行

1. PrinterTypeActivity.java を選択して、  をクリックします。



※Android デバイス上に別 PC の“apk”を再インストールすると、以下のエラーが発生します。
このエラーを回避するには、Android デバイス上のアプリケーションを一旦削除してから、再度実行してください。



Star デバイスと SDK を使用する

ポート名とインターフェイスの関係

StarIO は、DK-AirCash と通信するために特定のポート名を使用します。
ポート名は、以下の通り正しく設定しないと DK-AirCash との通信を行えません。また、用途に合わせて PortSettings 等の設定も必要です。詳しくはこちらをご参照ください。

Interface	Port Name
有線 LAN/無線 LAN	TCP:IP アドレス
Bluetooth	BT: BT:デバイス名 BT:MAC アドレス

◆既知の問題

PIN コード設定の Bluetooth インターフェイスを使用する場合に、端末によってはペアリング後、getPort()メソッドを実行すると PinCode 入力が要求され、PinCode を入力しても getPort メソッドが失敗して印刷またはプリンターステータスの取得が行えない現象が確認されています。

上記の現象が発生した場合は、以下の【回避策】をお試しください。

※上記現象が確認されているデバイス

- Arrows Tab Wi-Fi FAR75A(Android OS バージョン 4.0.3)

【回避策】

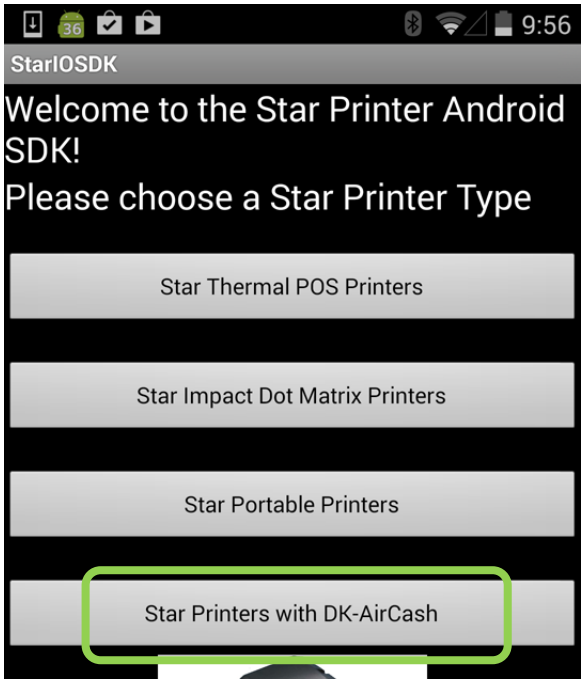
StarIOPort クラスの portSetting プロパティに空文字を指定してください。

現行： portSettings = “;P”;

変更後： portSettings = “”;

DK-AirCash を使用する

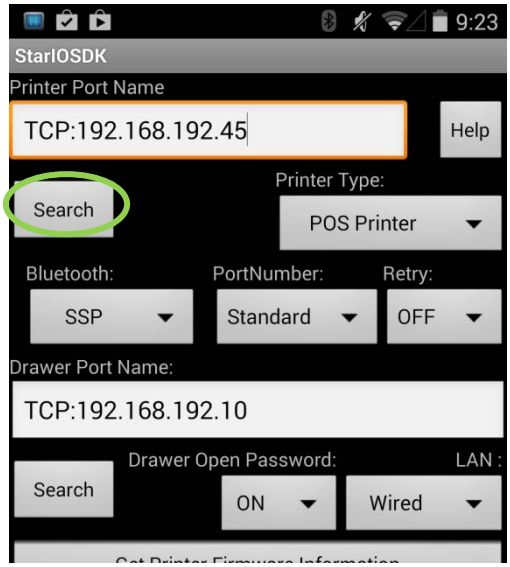
1. “Star Printers with DK-AirCash”をタップします。



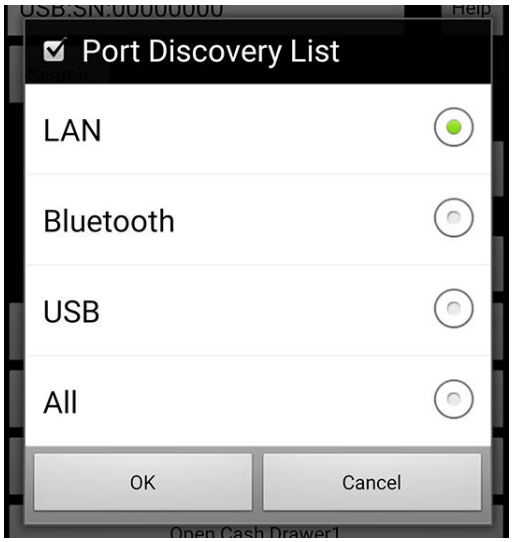
有線 LAN/無線 LAN デバイスの設定

・接続するプリンターを設定する

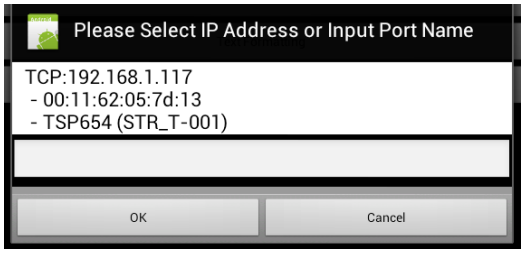
1. ネットワークに接続されている全ての Star の有線 LAN/無線 LAN プリンターを検索するために”Search”をタップします。



2. LAN もしくは All をタップします。

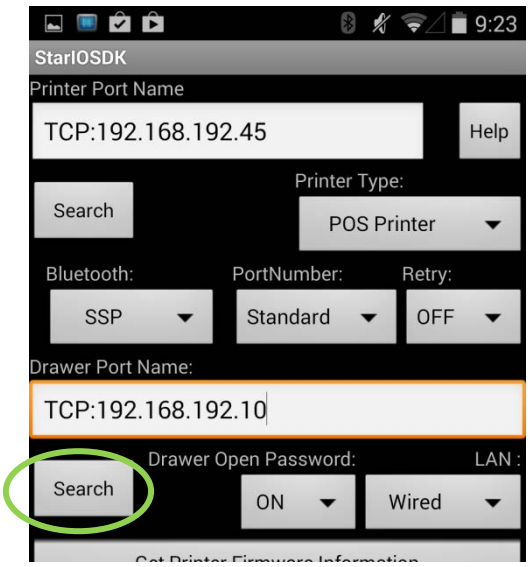


3. 検索されたプリンター一覧から、接続するプリンターの名前をタップしてください。
デバイスの MAC アドレスとモデル名は IP アドレスの下に表示されます。

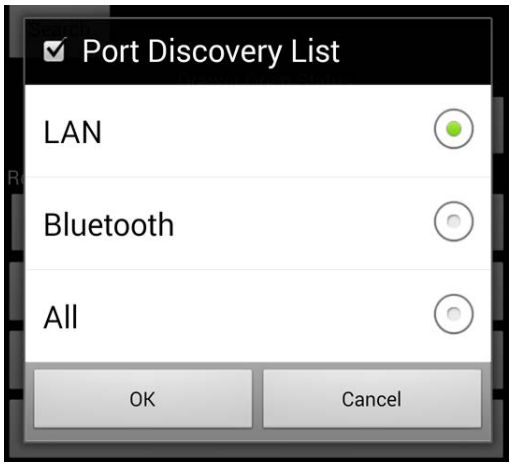


・接続する DK-AirCash を検索して設定する

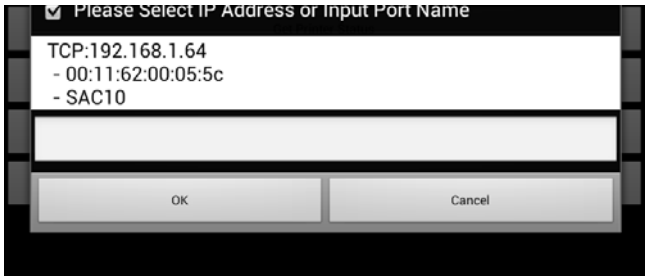
1. ネットワークに接続されている全ての DK-AirCash を検索するために”Search”をタップします。



2. LAN もしくは All をタップします。



3. 検索された DK-AirCash 一覧から、接続する DK-AirCash の名前をタップしてください。
デバイスの MAC アドレスとモデル名は IP アドレスの下に表示されます。

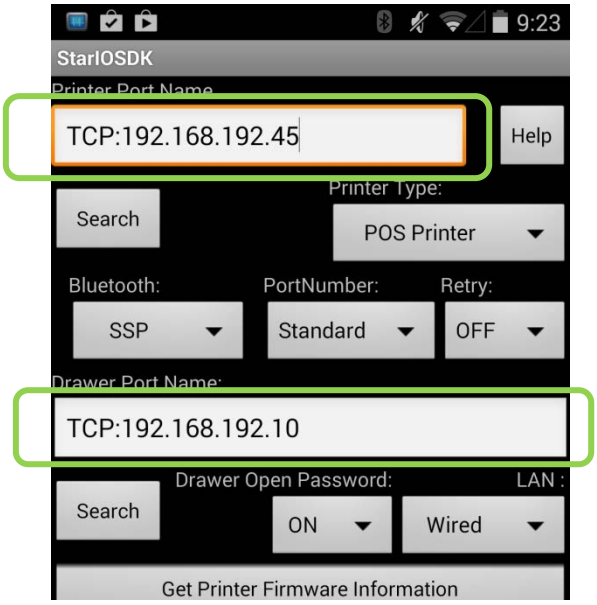


・接続するプリンターまたは DK-AirCash を手動で設定する

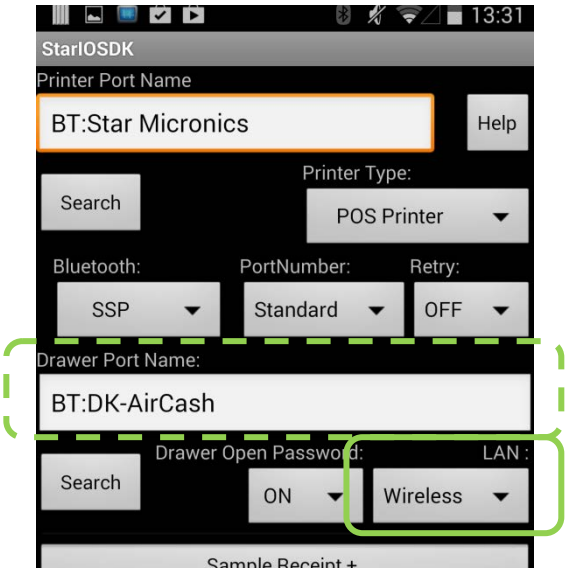
「Printer Port Name:」または「Drawer Port Name:」に以下のように手動で入力します。

TCP:<IP アドレス>

“TCP:”の後にデバイスの IP アドレスを追加します。(括弧は不要です)



・有線 LAN/無線 LAN デバイス – 接続方法選択



Drawer Port Name に設定した DK-AirCash が有線 LAN/無線 LAN デバイスの場合、接続方法に合わせて、LAN の設定を行います。

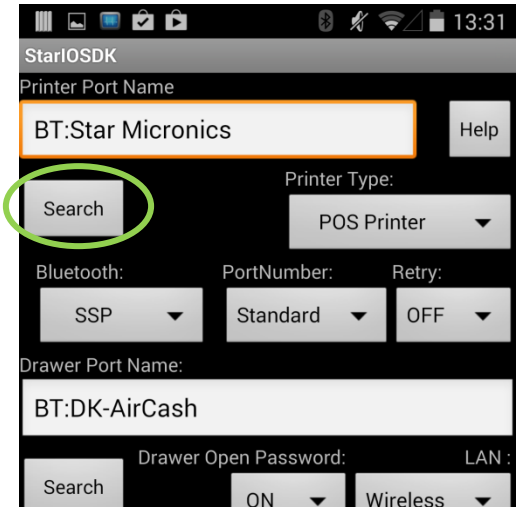
有線 LAN で接続する場合は“Wired”を選択します。getPort メソッドの portSetting パラメータには””（空文字列）が設定されます。

無線 LAN で接続する場合は“Wireless”を選択します。getPort メソッドの portSetting パラメータには”;wl”が設定されます。

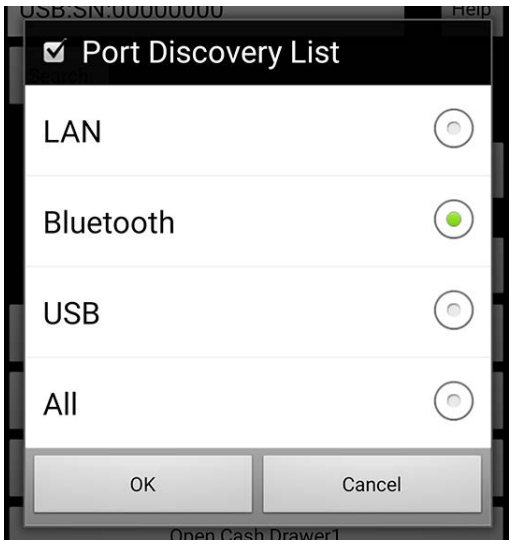
Bluetooth デバイスの設定

・接続するプリンターを設定する

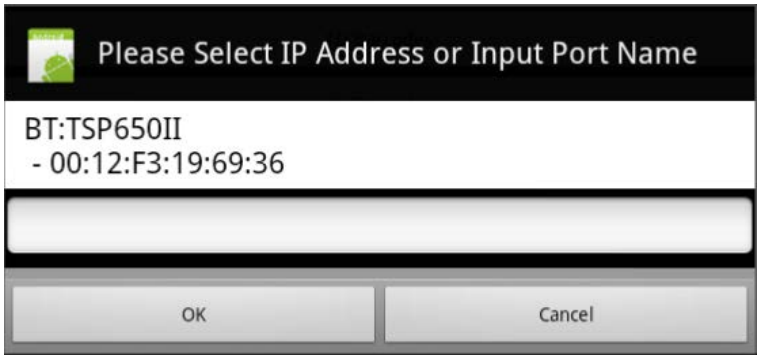
1. ペアリングされている全ての Star Bluetooth デバイスを検索するために、“Search”をタップしてください。



2. Bluetooth もしくは All をタップしてください。

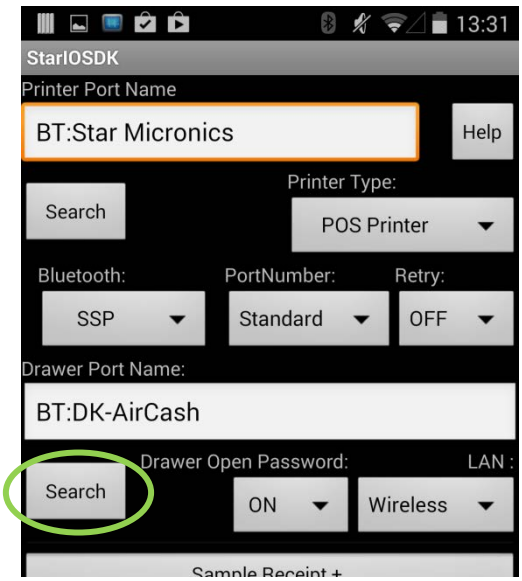


3. 接続したいデバイスの名前を選択してください。

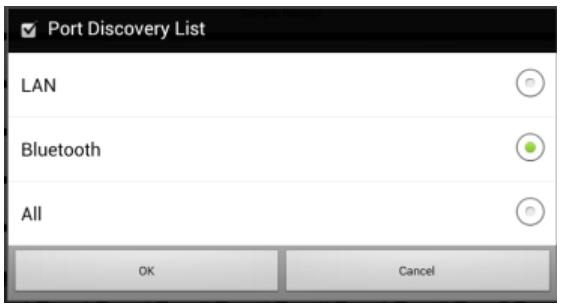


・接続する DK-AirCash を設定する

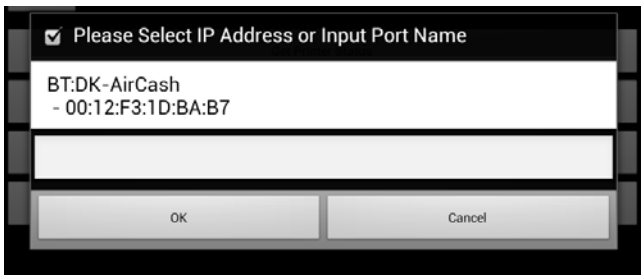
1. ペアリングされている全ての DK-AirCash を検索するために、“Search”をタップしてください。



2. Bluetooth もしくは All をタップしてください。



3. 検索されたデバイス一覧から、接続するデバイスの名前を選択してください。



・接続するプリンターまたは DK-AirCash を手動で設定する



Star Bluetooth デバイスを一台のみで使用する場合、パラメータの入力は必要ありません。



複数の Star Bluetooth デバイスを使用する場合、パラメータを入力します。

Star の Bluetooth デバイスを使用するためのポート設定は、以下 3 通りです。

1). BT:

Star Bluetooth デバイスが一台のみ接続されている場合、パラメータ無しの“BT:”指定で、自動的に接続されているデバイスと通信します。

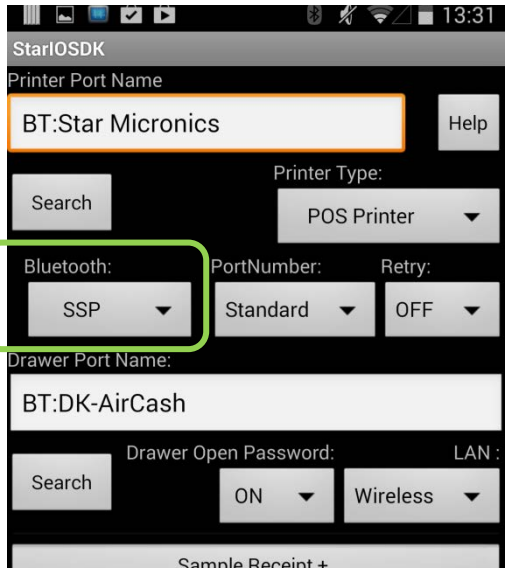
2). BT:<デバイス名>

“BT:”の後にデバイスのデバイス名を追加します。(括弧は不要です)

3). BT:<MAC アドレス>

“BT:”の後にデバイスの MAC アドレスを追加します。(括弧は不要です)
MAC アドレスは、製品裏側に貼り付けられたラベルに記載されています。

Bluetooth デバイス – セキュリティ方式選択

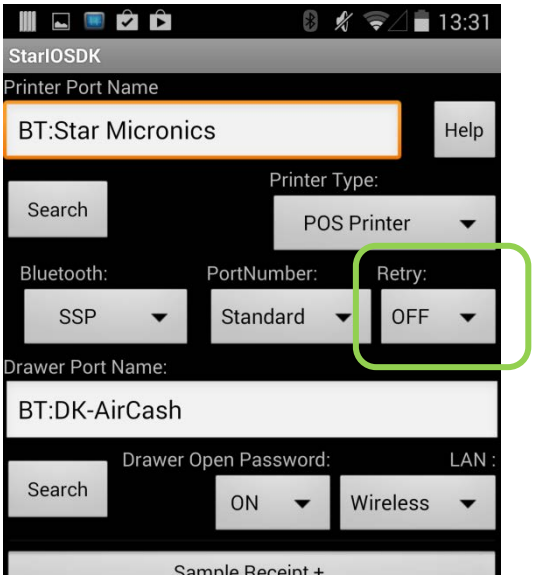


この設定によって、StarIO が Connect に使用する BluetoothAPI を切り替えます。Star Bluetooth デバイスとペアリングする際に使用したセキュリティ方式を選択してください。

SSP を選択した場合、getPort メソッドの portSetting のパラメータには、リトライが OFF の場合""(空文字列)が設定されます。また、リトライが ON の場合”;l”が設定されます。

PINCode を選択した場合、getPort メソッドの portSetting のパラメータには、リトライが OFF の場合 ";p"が設定されます。また、リトライが ON の場合";p;l”が設定されます。

Bluetooth デバイス - リトライ設定



この設定によって StarIO が Connect のリトライ設定を切り替えます。
リトライ時間は getPort メソッドで指定する timeout 時間が設定されます。

ON の場合リトライを行います。
　getPort メソッドの portSetting のパラメータには、セキュリティ方式が SSP の場合　”;l”が設定されます。また、セキュリティ方式が PINCode の場合　”;p;l”が設定されます。

OFF の場合リトライは行いません。
　getPort メソッドの portSetting のパラメータには、セキュリティ方式が SSP の場合　””(空文字)が、また、セキュリティ方式が PINCode の場合　”;p”が設定されます。

Android エミュレーター(AVD)を使用する

AVD(Android Virtual Device)を使用して、SDK の動作を確認できます。

ただし、Star Bluetooth デバイスとの通信は行えません。

Android SDK 概要

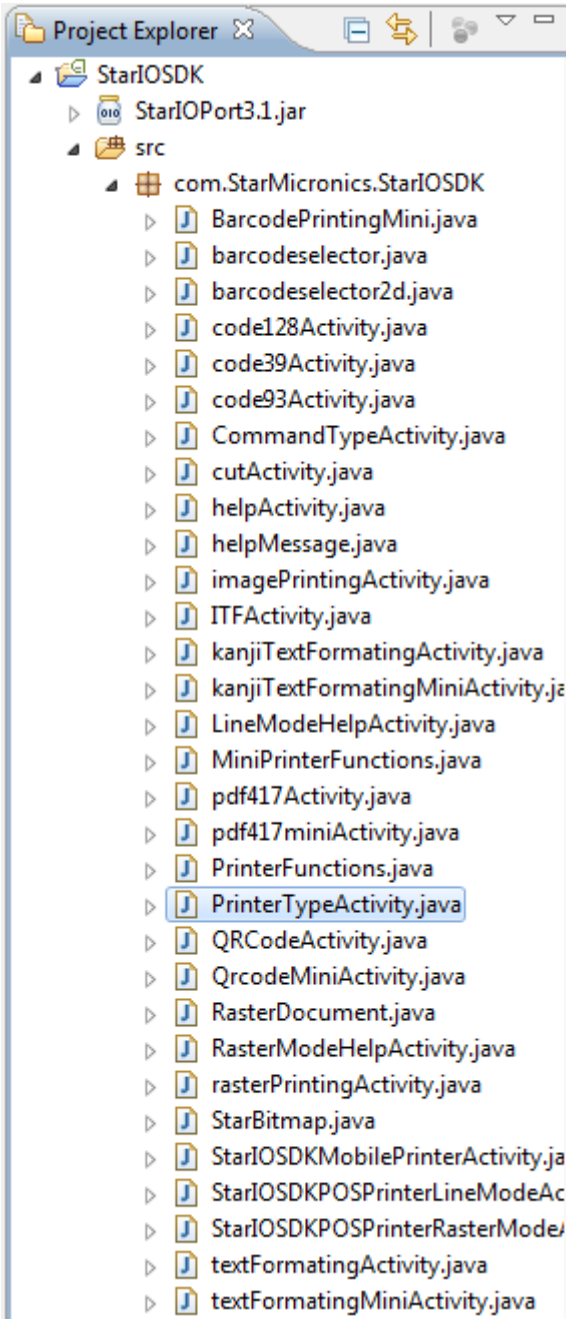
この概要では、SDK の主要コンポーネントについて簡単に説明します。

全ての機能が src の com.StarMicronics.StarIOSDK パッケージに位置しています。

PrinterTypeActivity.java を選択しプログラムを実行してください。このソースコードが POS デバイスおよびモバイルデバイスの両方の起点となります。

他のソース・ファイルをクリックすることにより、特定の機能がどのように動作するか確かめてください。例えば、“code128Activity.java”は GUI の中の 1D barcode Code128 に相当します。

全ての機能が DK-AirCash に利用可能ではありませんので、ご注意ください。各 SDK マニュアルの最初のページには、どの機能がサポートされているか記載されています。



StarIO – (StarIOPort.jar)

StarIOPort.jar もしくは StarIOPort3.1.jar ファイルをプロジェクトに追加することで、StarIO メソッドを利用することができます。

StarIO Library の選択

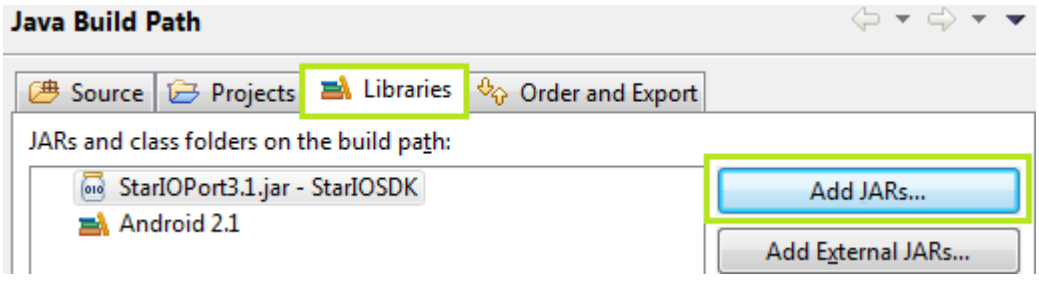
StarIO SDK には、２つのライブラリファイルが含まれています。アプリケーションが開発されている Android OS のバージョンに合わせて、いずれかを使用します。

Android OS Version	Star Interfaces	StarIO JAR File
Android OS 3.1 and Higher	Bluetooth, 有線 LAN/無線 LAN	StarIOPort3.1.jar (Default)
Android OS 2.2, 2.3, 3.0	Bluetooth, 有線 LAN/無線 LAN	StarIOPort3.1.jar (Default)
Android OS 2.1	Bluetooth, 有線 LAN/無線 LAN	StarIOPort.jar

プロジェクトへの StarIO ライブラリの追加

プロジェクトへの StarIO ライブラリの追加は以下の手順で行います。

1. SDK のパッケージから、Project Explorer に StarIO ライブラリをドラッグ＆ドロップします。
2. プロジェクト・フォルダーを右クリックして、プロパティを開きます。
3. Java Build Path パスをクリックします。
4. 「Libraries」タブをクリックし、“Add JARs”ボタンを押下します。
   StarIO Library を選択し、“OK”ボタンを押下します。



5. main code の先頭に以下を追加してください。

```
import com.starmicronics.stario.StarIOPort;
import com.starmicronics.stario.StarIOPortException;
import com.starmicronics.stario.StarPrinterStatus;
```

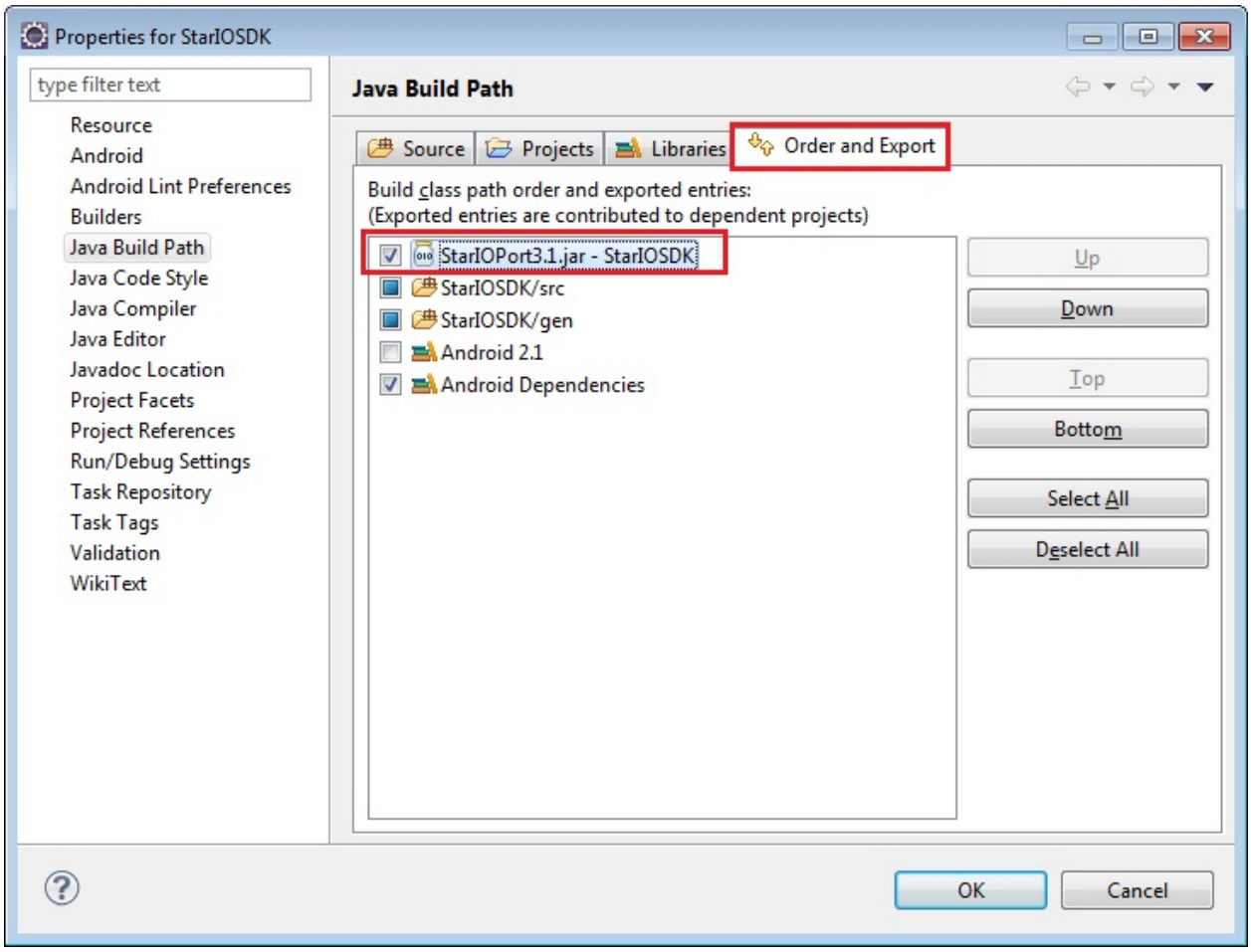
以上で、全ての StarIO メソッドを利用可能となります。

◆Android SDK Tools r17 以上でライブラリの追加をする際の注意

Android SDK Tools r17 から、ライブラリの管理方法が変更となりました。

必ず「StarIOSDK/src」の上に「StarIOPort3.1.jar」を配置する必要があります。
このメソッドは、r16 以前では問題がありませんでしたが、r17 以上を使用する場合にはエラーが発生する原因となります。

ご使用の環境により「StarIOPort.jar」を利用する場合にも、同様の注意が必要です。



1. プロジェクト名を右クリックして「Properties」を開きます。
2. 「Java Build Path」をクリックして「Order and Export」タブを開きます。
3. 「StarIOPort JAR ファイル」を選択してアプリケーションコードの上（最上位）に移動します。

StarIOPort ライブラリの設定

Android OS V2.2 以降 :

Include the default library: StarIOPort3.1.jar

上記の StarIO ライブラリを選択することで、StarIO SDK アプリケーションが動作します。
ソースコードの変更は不要です。

<対応インターフェイス>

Android OS 3.1 以上 :	Bluetooth, 有線 LAN/無線 LAN
Android OS 2.2, 2.3, 3.0 :	Bluetooth, 有線 LAN/無線 LAN

Android OS V2.1 以前 :

Include the legacy library: StarIOPort.jar

上記の StarIO ライブラリを選択することで、StarIO SDK アプリケーションが動作します。
ソースコードの変更が必要です。

PrinterFunctions.java

```
StarIOPort port = null;
try
{
    /*
        using StarIOPort3.1.jar (support USB Port)
        Android OS Version: upper 2.2
    */
    port = StarIOPort.getPort(portName, portSettings, 10000, context);
    /*
        using StarIOPort.jar
        Android OS Version: under 2.1
        port = StarIOPort.getPort(portName, portSettings, 10000);
    */
```

1. ソースコード→コメント :
   port = StarIOPort.getPort(portName, portSettings, 10000, context);
2. コメント→ソースコード :
   port = StarIOPort.getPort(portName, portSettings, 10000);

この変更により、従来の StarIOPort.jar のライブラリが使用できるようになります。

StarIO メソッド概要

StarIO Port クラス

StarIO Port クラスは、portName (String)、portSettings (String)、timeout (int)の 3 つのプロパティを持ちます。



各変数は、getPort(portName,portSettings,timeout);で使用します。getPort メソッドを使用するためには、値を持つ 3 つのクラス変数を割り当てます。

portName は、デバイスへの通信のポートを指定するために使用されるものです。

例): “TCP:192.168.1.2”

portSettings は、ポートの設定を行います。
有線 LAN/無線 LAN インターフェイスを使用する場合、接続方法の設定を行います。
また、Bluetooth インターフェイスを使用する場合、セキュリティ方式の設定とリトライの設定を行います。

Interface	Port Name	Port Settings	
有線 LAN **無線 LAN**	TCP:IP アドレス	空文字	有線 LAN 使用
		;wl	無線 LAN 使用
Bluetooth	BT: BT:デバイス名 BT:MAC アドレス	空文字	SSP 使用、リトライ無効
		;p	PIN コード使用
		;l	リトライ有効

Port Setting プロパティ 指定例）

portSettings = “;p;l”;　　…PIN コード使用、リトライ有効の場合

timeout は、ミリ秒のタイムアウト値で、内部制御と API での通信に使用されるものです。(このパラメータは、API が制限された時間内で完了することを保証しますが、正確なタイムアウトの長さを保証するものではありません。)

●メソッド

getPort

```
public static StarIOPort getPort(String portName, String portSettings, int TimeoutMillis)
                                                                    throws StarIOPortException
```

getPort は、DK-AirCash ポートのオープンに使用されます。portName と portSettings に有効なポート名とポート設定を指定することで、DK-AirCash に接続することができます。

```
//The following would be an actual usage of getPort:
StarIOPort port = null;
try
{
   port = StarIOPort.getPort(portName, portSettings, 10000);
}
catch (StarIOPortException e)
{
   //There was an error opening the port
}
```

getPort メソッドを呼び出すことで、StarIOPort クラスのインスタンスを生成することができます。上記の例では、“port”を作成し、null を割り当てています。getPort を含む下の行で、実際のポートハンドルを取得しています。



getPort を使用する場合は、常に try catch を使用してください。上記の例のような try catch を使用しなければ、通信エラーでポートオープンができなかった場合、プログラムはクラッシュする可能性があります。

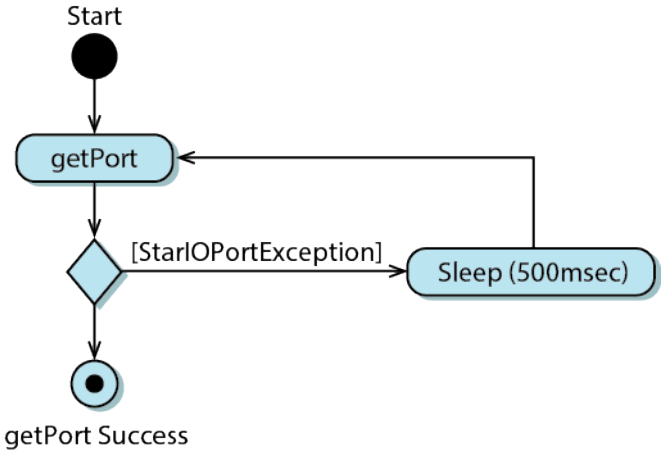
Bluetooth プリンタを使用する場合、端末、使用環境によって、getPort が失敗し、Bluetooth デバイスとの接続ができず印刷、ステータス取得ができない場合があります。

getPort メソッドの PortSettings パラメータに";l"、または";p;l"を指定することで、Bluetooth デバイスとの接続が失敗した場合にリトライを実行します。timeout パラメータに指定した時間の間繰り返し、timeout 時間経過しても接続が成功しない場合、StarIOPortException が発生します。

Bluetooth プリンタを使用する場合、端末、使用環境によって、getPort が失敗し、Bluetooth デバイスとの接続ができず印刷、ステータス取得ができない場合があります。

getPort が失敗した場合(StarIOPortException が発生します)、下記 2.3 を繰り返す処理を行ってください。

◆getPort リトライのフロー



1.getPort が失敗する
(StarIOPortException が発生する)

2.一定時間(例：500msec)おく

3.getPort を実行する

4.getPort が成功した場合、印刷またステータス取得などの処理に進んでください。

(*) 一定時間の値や getPort の繰り返し回数は、端末や使用環境によって異なるため、十分な検証を行い決定してください。

searchPrinter

```
public static ArrayList<PortInfo> searchPrinter(String target)
                                                    throws StarIOPortException
```

searchPrinter は、DK-AirCash を検索し、検索結果を ArrayList<PortInfo>にて返します。

戻り値の PortInfo クラスは、ポート名、DK-AirCash の MAC アドレス、デバイスモデル名を持ち、それぞれ String getPortName(), String getMacAddress(), String getModelName()メソッドで取得することができます。ポート名は、getPort()の portName 引数として使用することができます。

Parameters:

Target	– “TCP:” を指定した場合：	DK-AirCash(有線 LAN/無線 LAN 接続)を検索
	“BT:”を指定した場合：	DK-AirCash とプリンター(Bluetooth 接続)を検索

```
//The following would be an actual usage of searchPrinter:
try
{
   List<PortInfo> portList = StarIOPort.searchPrinter(“TCP:”);
   for (PortInfo port : portList) {
      Log.i (“LOG”, “Port Name: “ + port.getPortName());
      Log.i (“LOG”, “MAC Address: ” + port.getMacAddress());
      Log.i (“LOG”, “Model Name: “ + port.getModelName());
   }
 }
catch (StarIOPortException e)
{
   //There was an error opening the port
}
```

上記の例では、ネットワーク上の DK-AirCash を検索して一覧を取得し、その内容をログに出力します。



本 API はデバイスを確実に検出する事を保証するものではありません。

readPort

```
public int readPort(Byte[] readBuffer, int offset, int size)
                                                    throws StarIOPortException
```

このメソッドは、デバイスからデータを読み込みます。DK-AirCash から Raw byte を読み取る必要のある場合のみ使用してください。



ステータスの取得にこのメソッドを使用しないでください。
ステータスの取得は、RetreiveStatus メソッドを使用してください。

Parameters:

readbuffer	– データが読み込まれる Byte 配列のバッファ
offset	- ReadBuffer にデータを書き込み始める場所を指定
size	– 読み込むバイト数の合計

Returns:

実際に読み込まれたバイト数。データが全て読み取れなかった時でも、この関数は成功します。アプリケーションは、期待されるデータが全て読み取れるまで、この関数を複数回、呼び出す必要があります。または、しきい値に達するまで再試行をするようにします。

Throws:

StarIOPortException – 通信エラーが発生したとき

releasePort

```
public static void releasePort(StarIOPort port)
```

このメソッドは、指定されたポートへの接続をクローズします。

Parameters:

port –以前に初期化されたポートを表すポートタイプ



データ通信を行わない時は、ポートをリリース(クローズ)してください。
ポートをオープンしたままにしておくと、今後のポートオープンの呼び出し要求が失敗する可能性があります。



[有線 LAN/無線 LAN インターフェースを使用する場合]
releasePort メソッド実行後すぐに、getPort メソッドを実行すると失敗する場合があります。
releasePort メソッド実行後、500 ミリ秒の時間をおいてから getPort メソッドを実行してください。

writePort

```
public void writePort(Byte[] writeBuffer, int offset, int size)
                                                    throws StarIOPortException
```

このメソッドは、デバイスにデータを書き込みます。DK-AirCash にコマンドの送信を行う場合に使用します。

下記は、デバイスにデータを送信する最も簡単な方法です。

SDK の“printToPrinter”には、デバイスにデータが送信されたことを確認する方法を示すコードを記述してあります。しかし、下記コードは単にデータを送信しているだけです。

```
//Set a byte array to send to the drawer
//command = { Kick cash drawer }
Byte[] command = new Byte[]{ 0x07 };

try
{
   port.writePort(command, 0, command.length);
}
catch (StarIOPortException e)
{
   //There was an error writing to the port
}
```

※安全なプログラミングをするために try catch を使用してください。

Parameters:

writeBuffer – 出力データを格納する Byte 配列のバッファ
offset – writeBuffer からデータを読み込み始める場所を指定
size – 書き込むバイト数の合計

Throws:

StarIOPortException – 通信エラーが発生したとき

retreiveStatus

```
public StarPrinterStatus retreiveStatus ()
                                        throws StarIOPortException
```

このメソッドは、StarIO で詳細なステータスをデバイスから取得します。

Returns:

StarPrinterStatus 構造体は、現在の DK-AirCash のステータスを保持します。

Throws:

StarIOPortException – 通信エラーが発生したとき

このメソッドは StarPrinterStatus と呼ばれる StarIO の構造体を使用します。

この構造体は、ブーリアン型とバイナリーの両方の形式でデバイスステータスを保持します。
下記を行うことにより、プロジェクトで StarPrinterStatus オブジェクトを作成してください。

```
StarPrinterStatus status = port.retreiveStatus();

if (status.offline == false)
{
    if (status.compulsionSwitch == true) {
        //Cash drawer is open
    }
    else {
        //Cash drawer is closed
    }
}
else {
    //If true, then the drawer  is offline.
}
```

beginCheckedBlock

public StarPrinterStatus **beginCheckedBlock** ()
throws StarIOPortException

このメソッドは、endCheckedBlock メソッドとセットで使用してデータ送信終了の確認を行います。データ送信の直前に beginCheckedBlock メソッドを実行します。

Returns:
StarPrinterStatus 構造体は、現在のデバイスのステータスを保持します。

Throws:
StarIOPortException – 通信エラーが発生したとき
デバイスがオフラインのとき

サンプルコードはこちらをご参照ください。

endCheckedBlock

public StarPrinterStatus **endCheckedBlock** ()
throws StarIOPortException

このメソッドは、beginCheckedBlock メソッドとセットで使用します。

DK-AirCash の状態を監視し、送信したコマンドが DK-AirCash に処理されると制御を返します。

タイムアウト時間(*1) 内に印刷が完了しなかった場合や、印刷中にプリンタエラーが発生した場合は、例外 StarIOPortException をスローします。

(*1) タイムアウト時間は、setEndCheckedBlockTimeoutMillis メソッドで設定した値が使用されます。初期値は getPort で指定したタイムアウト時間となります。また、10 秒未満の値が指定された場合にはタイムアウトは 10 秒になります。

Returns:
StarPrinterStatus 構造体は、現在のデバイスのステータスを保持します。

Throws:
StarIOPortException
- 通信エラーが発生した
- タイムアウト時間内にデバイスからの応答がない

サンプルコードはこちらをご参照ください。

```
try
{
    port = StarIOPort.getPort(portName, portSettings, 10000, context);

    //Start checking the completion of data transmission
    StarPrinterStatus status = port.beginCheckedBlock();

    //Printing
    byte[] command = PrinterFunctions.createPrintData(paperWidthInch);
    port.writePort(command, 0, command.length);

    //End checking the completion of data transmission
    status = port.endCheckedBlock();

    if (status.offline == true) {
       //If true, then the drawer is offline.
    }
}
catch (StarIOPortException e)
{
    Log.d("StarIOSample", "An timeout error has occurred during printing.");
}
finally
{
    if (port != null)
    {
        try
        {
            StarIOPort.releasePort(port);
         }
         catch (StarIOPortException e) {}
    }
}
```

setEndCheckedBlockTimeoutMillis

```
public void setEndCheckedBlock ( int endCheckedBlockTimeoutMillis)
```

endCheckedBlock メソッドのタイムアウト時間を設定します。[単位: ミリ秒]

印刷に時間がかかる場合、この値を大きくする事で endCheckedBlock メソッドの印刷完了待ち時間を伸ばす事ができます。

初期値は、getPort メソッドで指定したタイムアウト時間となります。



指定した時間が 10 秒未満の場合、タイムアウトは 10 秒になります。

getFirmwareInformation

```
public Map<String, String> getFirmwareInformation()
                                        throws StarIOPortException
```

このメソッドは、プリンターのモデル名とファームウェアバージョンを取得します。
取得結果を Map<String, String>にて返します。

Returns:
Key に”ModelName”を設定することで戻り値からモデル名を取得します。
Key に”FirmwareVersion”を設定することで戻り値からファームウェアバージョンを取得します。

Throws:

StarIOPortException – 通信エラーが発生したとき

Note:

・取得に失敗した場合、空文字を返します。

getStarIOVersion

public static synchronized String **getStarIOVersion**()

このメソッドは、StarIO のバージョンを取得します。

Returns:
StarIO バージョン

getDipSwitchInformation

public Map<String, Boolean> **getDipSwitchInformation**()
throws StarIOPortException

このメソッドは、DK-AirCash の DIP Switch の設定状態を取得します。
取得結果を Map<String, Boolean>にて返します。

Returns:
Key に”DIPSW11”や“DIPSW12”を設定することで戻り値から DIP Switch の状態を取得します。

Key 設定可能文字列：

“DIPSW11”, “DIPSW12”, “DIPSW13”, “DIPSW14”,
“DIPSW15”, “DIPSW16”, “DIPSW17”, “DIPSW18”,
“DIPSW21”, “DIPSW22”, “DIPSW23”, “DIPSW24”,
“DIPSW25”, “DIPSW26”, “DIPSW27”, “DIPSW28”,

true	- DIP Switch の状態が ON の場合
false	- DIP Switch の状態が OFF の場合

Throws:

StarIOPortException	– 通信エラーが発生したとき – DK-AirCash 以外のモデルに使用したとき

StarBluetoothManager クラス

StarBluetoothManager クラスは、Bluetooth インターフェイスの各種設定を行うためのクラスです。StarIO Port クラスと同時に使用しないでください。

●コンストラクタ

StarBluetoothManager

```
public StarBluetoothManager(String portName, String portSetting, int timeoutMillis,
StarBluetoothManager.StarDeviceType starDeviceType)
                                                    throws StarIOPortException
```

Parameters:

portName – 接続するデバイスのポート名
例）BT:StarMicronics

portSetting - デバイスに接続するためのポート設定
例）Bluetooth インターフェイスで PIN コードを使用している場合、
";p"を指定してください。
SSP を指定している場合、空文字を指定してください。

timeoutMillis
– ミリ秒のタイムアウト値で、内部制御と API での通信に使用される
（このパラメータは、API が制御された時間内で完了することを保証
しますが、正確なタイムアウトの長さを保証するものではありません。）

starDeviceType
– 検索するデバイスの種類
例）StarDeviceTypeDesktopPrinter

Throws:

StarIOPortException – starDeviceType に StarDeviceTypePortablePrinter が設定されているとき
– portName に BT:が設定されていないとき

●メソッド

open

```
public void open()
```

StarBluetooth デバイスとの接続を開きます。

Throws:

StarIOPortException – 通信エラーが発生したとき

close

```
public void close()
```

StarBluetooth デバイスとの接続を閉じます。

Throws:

StarIOPortException – 通信エラーが発生したとき

apply

```
public void apply()
```

StarBluetooth デバイスには以下のメソッドで指定した値を設定します。

- setBluetoothDevice メソッド
- setiOSPortName メソッド
- setAutoConnect メソッド
- setPicCode メソッド
- setSecurityType メソッド

Throws:

StarIOPortException – 通信エラーが発生したとき

– SecurityType が PIN コードに設定されていて、AutoConnect が ON に設定されているとき

loadSetting

public void **loadSetting**()

接続した StarBluetooth デバイスから設定を取得します。

Throws:

StarIOPortException – 通信エラーが発生したとき

getBluetothDeviceName

public String **getBluetoothDeviceName**()

loadSetting メソッドで取得した、または SetBluetoothDeviceName メソッドで設定した Bluetooth デバイス名を取得します。loadSetting メソッド、または setBluetoothDeviceName メソッドを呼び出した後に使用してください。

Returns:

BluetoothDeviceName

setBluetoothDeviceName

public void **setBluetoothDeviceName**(String BluetoothDeviceName)

Bluetooth デバイス名を 1 文字以上、16 文字以下で設定します。
Bluetooth デバイス名を変更するには、setBluetoothDeviceName メソッド後に apply メソッドを実行してください。

Parameters:

BluetoothDeviceName – Bluetooth デバイス名

Throws:

StarIOPortException – 使用可能文字列以外を設定したとき
指定した文字列が 1 文字以上、16 文字以下でないとき

使用可能文字列： 0-9 a-z A-Z ; : ! ? # $ % & , . @ _ - = Space
/ * + ~ ^ [{ ( ] }) | \

getiOSPortName

public String **getiOSPortName**()

loadSetting メソッドで取得した、または setiOSPortName メソッドで設定した iOSPort 名を取得します。loadSetting メソッド、または setiOSPortName メソッドを呼び出した後に使用してください。

Returns:

iOSPortName

setiOSPortName

public void **setiOSPortName**(String iOSPortName)

iOSPort 名を 1 文字以上 16 文字以下で設定します。
iOSPort 名を変更するには、setiOSPortName メソッド後に apply メソッドを実行してください。

Parameters:

iOSPortName – iOS ポート名

Throws:

StarIOPortException – 使用可能文字列以外を設定したとき
指定した文字列が 1 文字以上、16 文字以下でないとき

使用可能文字列： 0-9 a-z A-Z ; : ! ? # $ % & , . @ _ - = Space
/ * + ~ ^ [{ ( ] }) | \

getPinCode

public String **getPinCode**()

setPinCode メソッドで設定した PIN コードの設定を取得します。setPinCode メソッドを呼び出した後に使用してください。

Returns:

PinCode

setPinCode

public void **setPinCode**(String pinCode)

StarBluetooth デバイスの PIN コードを 4 文字以上、16 文字以下で設定します。
PIN コードを変更するには、setPinCode メソッド後に apply メソッドを実行してください。

Parameters:

PinCode – PIN コード
例）1234

Throws:

StarIOPortException – 使用可能文字列以外を設定したとき
指定した文字列が 4 文字以上、16 文字以下でないとき

使用可能文字列： 0-9 a-z A-Z

getAutoConnect

public boolean **getAutoConnect**()

loadSetting メソッドで取得した、または setAutoConnect メソッドで設定した AutoConnection の設定を取得します。loadSetting メソッド、または setAutoConnect メソッドを呼び出した後に使用してください。

Returns:

true - AutoConnection 設定が ON の場合
false - AutoConnection 設定が OFF の場合



Android デバイスで使用する場合は AutoConnection を
OFF にしてください。

setAutoConnect

public void **setAutoConnect**(boolean autoConnect)

StarBluetooth デバイスの AutoConnection を設定します。
AutoConnection の設定を変更するには、setAutoConnect メソッド後に apply メソッドを実行してください。

Parameters:

autoConnect　　例）false



Android デバイスで使用する場合は AutoConnection を OFF にしてください。

getSecurityType

public StarBluetoothManager.StarBluetoothSecurity **getSecurityType**()

loadSetting メソッドで取得した、または setSecurityType で設定した Bluetooth セキュリティタイプ（SSP or PIN Code）を取得します。loadSetting メソッドまたは setSecurityType メソッドを呼び出した後に使用してください。

Returns:

StarBluetoothManager.StarBluetoothSecurity

setSecurityType

public void **setSecurityType**(StarBluetoothManager.StarBluetoothSecurity securityType)

StarBluetooth デバイスのセキュリティタイプ（SSP or PIN Code）を設定します。セキュリティタイプを変更するには、setSecurityType メソッド後に apply メソッドを実行してください。

Parameters:

securityType　　例）StarBluetoothManager.StarBluetoothSecurity.SSP

getPortName

```
public String getPortName()
```

StarBluetoothManager コンストラクタで指定した portName を取得します。

Returns:
portName

getPortSetting

```
public String getPortSetting()
```

StarBluetoothManager コンストラクタで指定した portSetting を取得します。

Returns:
portSetting

getTimeoutMillis

```
public int getTimeoutMillis()
```

StarBluetoothManager コンストラクタで指定した timeoutMillis を取得します。

Returns:
timeoutMillis

getDeviceType

```
public String getDeviceType()
```

StarBluetoothManager コンストラクタで指定した starDeviceType を取得します。

Returns:
StarBluetoothManager.StarDeviceType

isOpened

```
public boolean isOpened()
```

ポートの状態を取得します。

Returns:

true - ポートがオープンされている

StarPrinterStatus 構造体　ステータスリスト

メンバ名	説明	型	詳細	DK-AirCash
blackMarkError	ブラックマークエラー	boolean	DK-AirCash 非対応	
compulsionSwitch	コンパルジョン SW	boolean	ドロワのコンパルジョン SW が押されているとtrue となる。 通常は false。	✓
coverOpen	カバーの状態	boolean	DK-AirCash 非対応	
cutterError	オートカッターエラー	boolean	DK-AirCash 非対応	
etbAvailable	ETB 使用可否	boolean	ETB が使用可能な場合に true となる。 使用できない場合は false。	✓
etbCounter	ETB カウンタ	int	現在の ETB カウンタの値。	✓
headThermistorError	ヘッドサーミスタエラー	boolean	DK-AirCash 非対応	
offline	ON-LINE/OFF-LINE 状態	boolean	オフラインの場合に true となる。 オンライン時は false。	✓
overTemp	印字ヘッド高温による停止中	boolean	DK-AirCash 非対応	
presenterPaperJamError	プレゼンター紙ジャムエラー	boolean	DK-AirCash 非対応	
presenterState	プレゼンタ用紙位置	int	DK-AirCash 非対応	
raw	ステータスのバイト列	byte[63]	ステータスのバイト列 (例: HEX 23 86 00 00 00 00 00 00 00)	✓
rawLength	raw の長さ	int	raw の長さ	✓
receiptPaperEmpty	用紙エンド	boolean	DK-AirCash 非対応	
receiptPaperNearEmptyInner	用紙ニアエンド(内側)	boolean	DK-AirCash 非対応	
receiveBufferOverflow	受信バッファオーバーフロー	boolean	DK-AirCash 非対応	
unrecoverableError	復帰不可能エラー	boolean	復帰不可能エラー(ヘッドサーミスタエラー、オートカッターエラー、電源電圧エラー等)が発生した場合に true となる。 通常は false。	✓
voltageError	電源電圧エラー	boolean	DK-AirCash 非対応	

StarIO Android SDK 機能

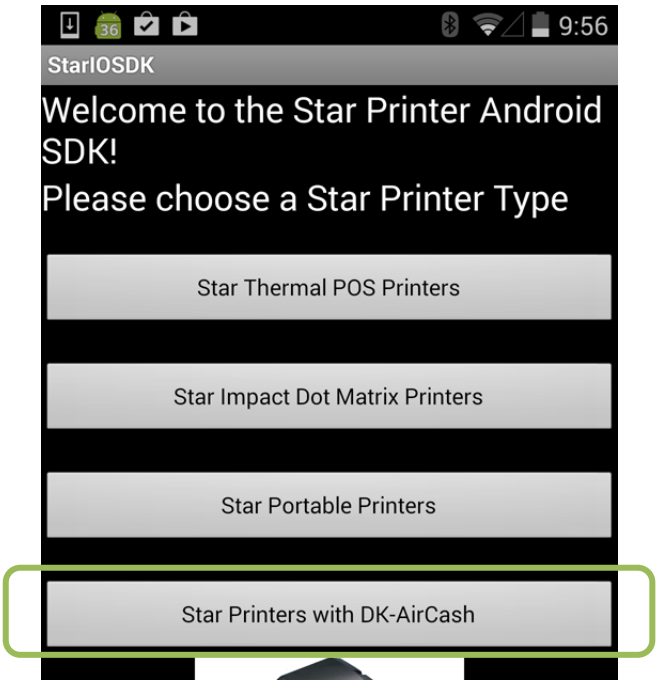
SDK 機能と StarIO のデバイスコマンドについての概要

これらのコマンドは、全て Star Line Mode に記載されています。

また、この SDK では詳細情報の参照先が、Star Line Mode コマンドマニュアルのページやセクションとなっています。そのため、特定のコマンドに関する詳細な情報が必要な場合は、StarLine Mode コマンドマニュアルをダウンロードし参照してください。

デバイスとコマンド種類の選択

1. “Star Printers with DK-AirCash”をタップしてください。

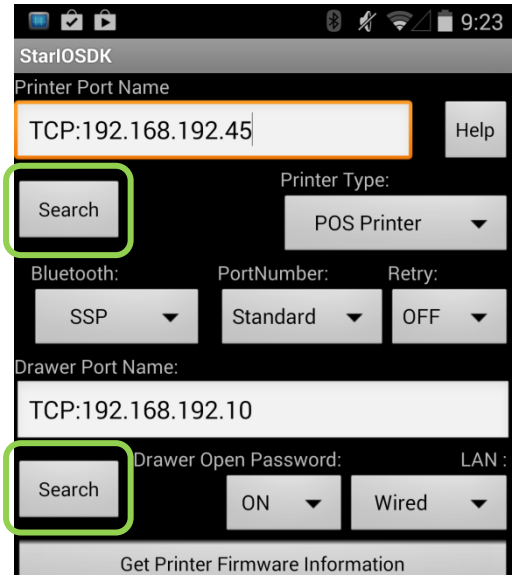


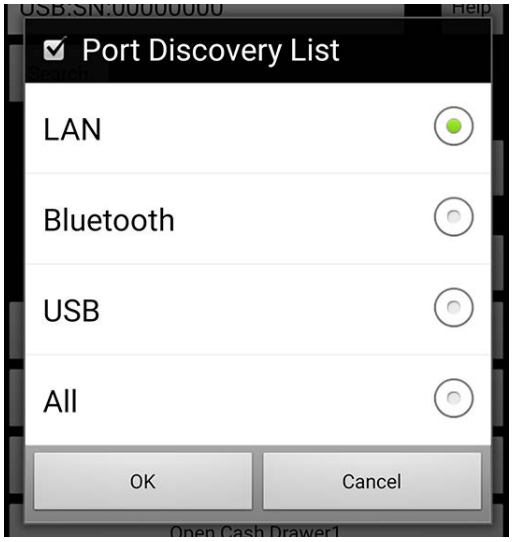
コマンド別サンプル機能対応リスト

Command Samples Include:

- Get Printer Firmware Information
- Get DK-AirCash Firmware Informtion
- Get DK-AirCash Dip Switch Information
- Get Printer Status
- Get DK-AirCash Status
- Sample Receipt + Open Cash Draweer1 via DK-AirCash
- Open Cash Drawer1 via DK-AirCash / Open Cash Drawer2 via DK-AirCash
- DK-AirCash Bluetooth Setting

Port Discovery





ネットワークに接続されている Star POS デバイスを自動的に検索します。検索結果より、接続したいデバイスを選択してください。詳細はここを参照してください。

---

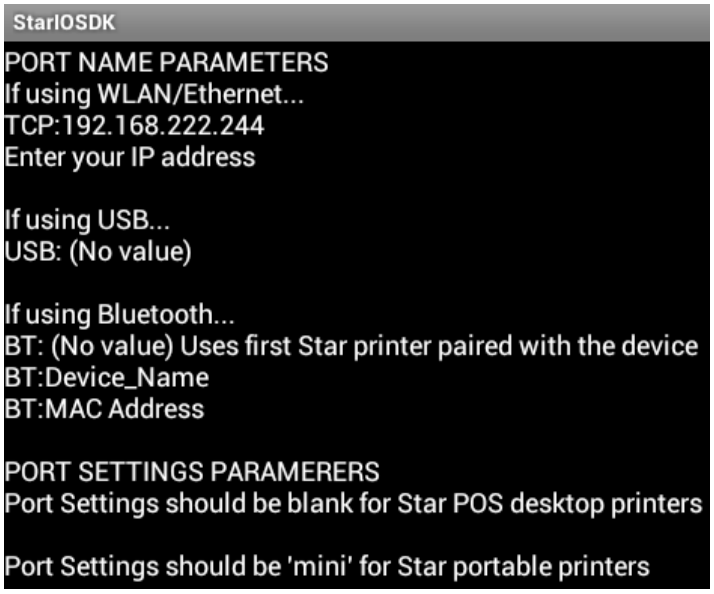
get StarIOVersion





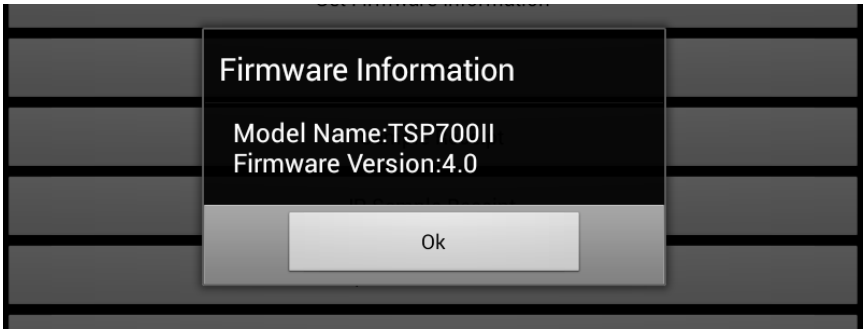
StarIO のバージョンを取得します。

Help



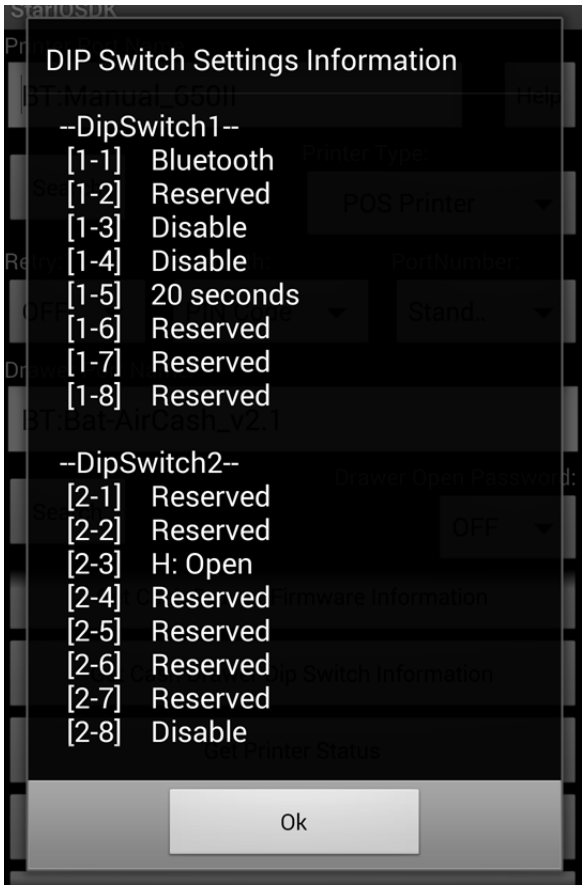
Help は、Bluetooth/有線 LAN/無線 LAN のパラメータと StarIO ポートの設定に関する情報を表示します。

Get Printer Firmware Information / Get DK-AirCash Firmware Information

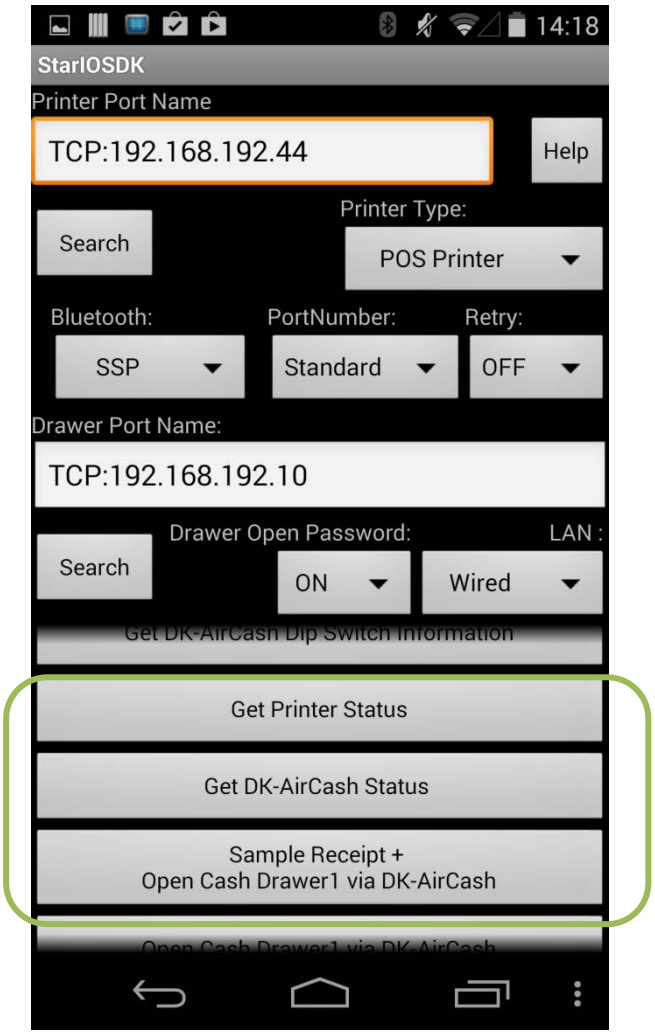


Printer Port Name / Drawer Port Name に設定されたデバイスのファームウェア情報を表示します。

Get DK-AirCash Dip Switch Information



Drawer Port Name に設定されたデバイスの DIP Switch 設定情報を表示します。



Get Printer Status　/Get DK-AirCash Status

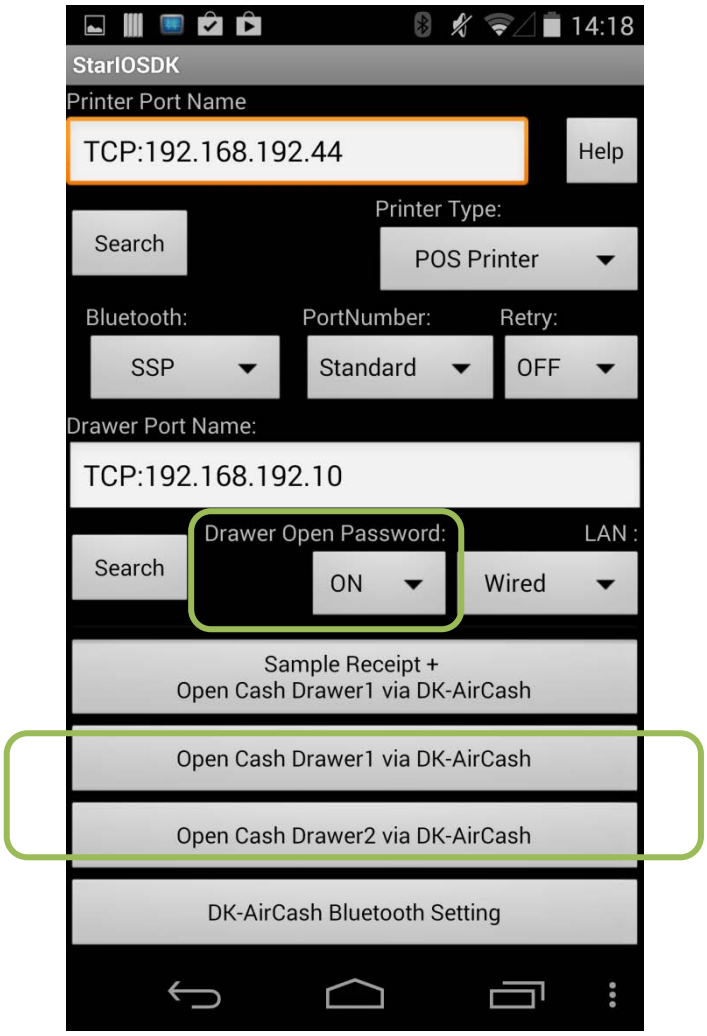
接続されたプリンターまたはキャッシュドロアのステータスを表示します。

Sample Receipt + Open Cash Drawer1 via DK-AirCash

プリンターが接続されていればサンプルレシートの印刷をします。また、キャッシュドロアが接続されていればキャッシュドロアをオープンします。
Drawer Open Password に ON が設定されていれば、キャッシュドロワーオープンの前にパスワードの入力を要求します。設定されているパスワードは「1234」です。

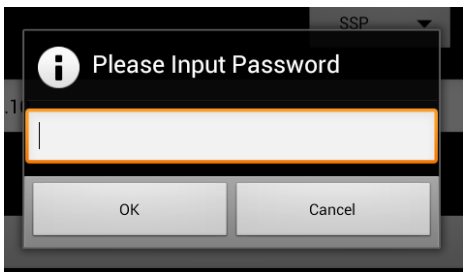
プリンターが接続されていない場合、キャッシュドロアは作動しません。

Open Cash Drawer1 via DK-AirCash /
Open Cash Drawer2 via DK-AirCash

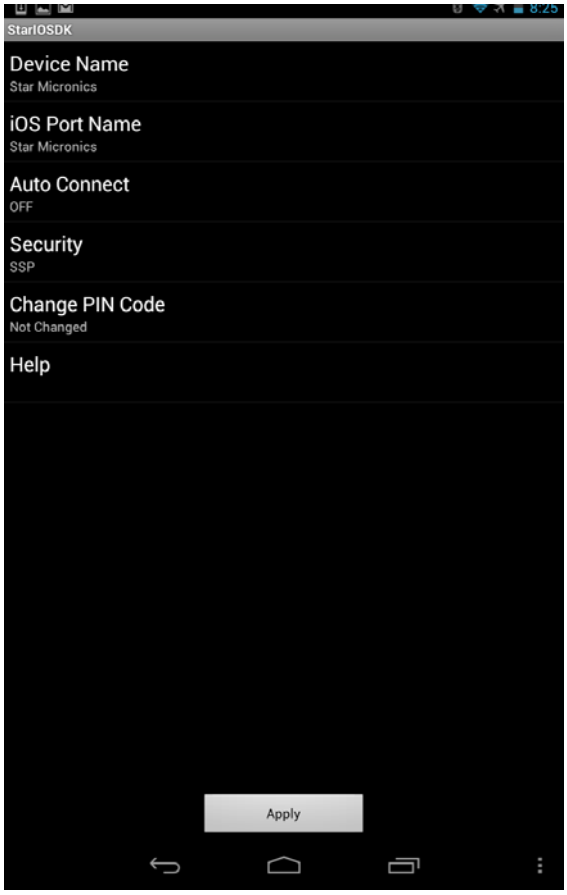


キャッシュドロアが接続されていればキャッシュドロアをオープンします。

Drawer Open Password に ON が設定されていれば、キャッシュドロアオープンの前にパスワードの入力を要求します。設定されているパスワードは「1234」です。



DK-AirCash Bluetooth Setting



PortName に指定された Bluetooth デバイスに接続して、Bluetooth インターフェイスの各種設定を変更します。

変更された値は、デバイスの電源再投入・再ペアリングを行った後に有効になります。

StarIO を使用するアプリケーション開発のために

#1: 大規模なプロジェクトをコーディングしている場合、全ての印刷メソッドを抽象化してクラスを作成してください。これはソースコードの再利用に役立ちます。また、特定のコードを見つけるのが容易になり、時間の節約にもなります。StarIO を唯一のクラスに存在させることによって、オブジェクト指向プログラミングを実現できます。

#2: ASCII と Unicode、16 進と 10 進、及び、Byte と Char の違いと定義が何であるかを確認してください。1 バイトは、通常、8 桁の 2 進数(1 と 0)で 8 ビットです。これらのバイトは、ちょうど 8 ビットのバイナリーデータです。しかし、byte は int または char でもありえます。3 つの異なる変数の型は、基本的に同じ方法でデータを保持しますが、わずかな違いがあります。印刷ジョブのデータを格納する変数を選択する際、byte の代わりに char、int、または string で試してみてください。
ASCII から Unicode(また逆も同様に)への変換は、時として安全ではありません。そのため、Encoding クラスがどのように機能するのかを確認してください。Unicode でよく見られる間違いの例として、“Culture-sensitive searching and casing”、“Surrogate pairs”、“Combining characters”、“Normalization”などがあります。こちらのサイトをご確認ください。

#3: 16 進ダンプモードについて 作成したアプリケーションのデバッグの際、デバイスの 16 進ダンプモードを使用してください。16 進ダンプモードを使用することで、アプリケーションから送信したデータをデバイスが正しく受信できたか確認する事ができます。16 進ダンプモードへの設定は、各デバイスの製品仕様書を参照ください。

#4: StarIO コマンド・コードをリバース・エンジニアリングしないでください。全ての StarIO コマンドは、コマンドマニュアルより参照可能です。また、本 SDK を活用することで、アプリケーション作成時の工数を大幅に削減可能です。

#5: 本 SDK に記載されていないコマンドについては、SDK のコードサンプルを参照してください。また、**スター精密グローバルサポートサイト**の Developers セクションにアクセスすることで、より詳細な情報を入手可能です。

#6: Android 以外の OS(例: iOS) に対応した SDK をお探しの場合には、**スター精密グローバルサポートサイト**の Developers セクションをご覧ください。

追加リソース

以下のリンクよりプログラマーマニュアルを入手してください。

スター精密 サポート情報

スター精密グローバルサポートサイト

FAQ を参照してください。
グローバルサポートサイトより、以下の情報を入手できます。

- 最新バージョンの SDK マニュアル／ソースコード
- アドイス／業界情報
- スター精密デバイスドライバ
- 技術的な質問／サポート

その他の印刷に関する詳細情報

Android Developers Resource
Android の公式開発リソース

System Requirements for Android Development
Windows、Linux、Mac のシステム要件に関して

Unicode.org
ユニコードコンソーシアム – Unicode の詳細について

1D Barcodes
Barcode Island - バーコードの詳細について

2D Barcodes
QR Codes、PDF417 に関する情報について

Code Pages
コード・ページに関する情報について

ASCII コード表

10 進	16 進	文字	10 進	16 進	文字	10 進	16 進	文字	10 進	16 進	文字
0	0	NUL	16	10	DLE	32	20	(space)	48	30	0
1	1	SOH	17	11	DC1	33	21	!	49	31	1
2	2	STX	18	12	DC2	34	22	"	50	32	2
3	3	ETX	19	13	DC3	35	23	#	51	33	3
4	4	EOT	20	14	DC4	36	24	$	52	34	4
5	5	ENQ	21	15	NAK	37	25	%	53	35	5
6	6	ACK	22	16	SYN	38	26	&	54	36	6
7	7	BEL	23	17	ETB	39	27	'	55	37	7
8	8	BS	24	18	CAN	40	28	(	56	38	8
9	9	TAB	25	19	EM	41	29	)	57	39	9
10	A	LF	26	1A	SUB	42	2A	*	58	3A	:
11	B	VT	27	1B	ESC	43	2B	+	59	3B	;
12	C	FF	28	1C	FS	44	2C	,	60	3C	<
13	D	CR	29	1D	GS	45	2D	-	61	3D	=
14	E	SO	30	1E	RS	46	2E	.	62	3E	>
15	F	SI	31	1F	US	47	2F	/	63	3F	?

10進	16進	文字	10進	16進	文字	10進	16進	文字	10進	16進	文字
64	40	@	80	50	P	96	60	`	112	70	p
65	41	A	81	51	Q	97	61	a	113	71	q
66	42	B	82	52	R	98	62	b	114	72	r
67	43	C	83	53	S	99	63	c	115	73	s
68	44	D	84	54	T	100	64	d	116	74	t
69	45	E	85	55	U	101	65	e	117	75	u
70	46	F	86	56	V	102	66	f	118	76	v
71	47	G	87	57	W	103	67	g	119	77	w
72	48	H	88	58	X	104	68	h	120	78	x
73	49	I	89	59	Y	105	69	i	121	79	y
74	4A	J	90	5A	Z	106	6A	j	122	7A	z
75	4B	K	91	5B	[	107	6B	k	123	7B	{
76	4C	L	92	5C	¥	108	6C	l	124	7C	|
77	4D	M	93	5D	]	109	6D	m	125	7D	}
78	4E	N	94	5E	^	110	6E	n	126	7E	~
79	4F	O	95	5F	_	111	6F	o	127	7F	□

SDK マニュアル 改訂履歴

Release Date	SDK Package Version	Update
Apr. 20 2015	3.14.0	- 誤記訂正
Mar. 18 2015	3.13.1	- PortDiscovery USB 対応追加
Oct. 10 2014	3.13.0	- Wireless LAN 対応追加
Jul. 14 2014	3.12.0	- getStarIO Version メソッド追加 - getDipSwitchInformation メソッド追加 - portOpen リトライ機能追加
Mar. 10 2014	3.10.0	- StarBluetoothManager クラス追加 - getFirmwareInformation メソッド追加 - setEndCheckedBlockTimeoutMillis メソッド追加 - Open Cash Drawer2 機能追加
Dec. 24 2013	3.8.0	- 注記追加
Jul. 3 2013	3.7.0	- 新規リリース

Star Micronics is a global leader in the manufacturing of small printers. We apply over 50 years of knowhow and innovation to provide elite printing solutions that are rich in stellar reliability and industry-respected features. Offering a diverse line of Thermal, Hybrid, Mobile, Kiosk and Impact Dot Matrix printers, we are obsessed with exceeding the demands of our valued customers every day.

We have a long history of implementations into Retail, Point of Sale, Hospitality, Restaurants and Kitchens, Kiosks and Digital Signage, Gaming and Lottery, ATMs, Ticketing, Labeling, Salons and Spas, Banking and Credit Unions, Medical, Law Enforcement, Payment Processing, and more!

High Quality POS Receipts, Interactive Coupons with Triggers, Logo Printing for Branding, Advanced Drivers for Windows, Mac and Linux, Complete SDK Packages, Android, iOS, Blackberry Printing Support, OPOS, JavaPOS, POS for .NET, Eco-Friendly Paper and Power Savings with Reporting Utility, ENERGY STAR, MSR Reading, *future*PRNT, StarPRNT… How can Star help you fulfill the needs of your application?

Don't just settle on hardware that won't work as hard as you do. Demand everything from your printer. Demand a Star!

Star Micronics Worldwide

Star Micronics Co., Ltd.
536 Nanatsushinya
Shimizu-ku, Shizuoka 424-0066
Japan
+81-54-347-2163
http://www.star-m.jp/eng/index.htm

Star Micronics America, Inc.
65 Clyde Road. Suite G
Somerset, NJ 08873
USA
1-848-216-3300
http://www.starmicronics.com

Star Micronics EMEA
Star House
Peregrine Business Park, Gomm Road
High Wycombe, Buckinghamshire HP13 7DL
UK
+44-(0)-1494-471111
http://www.star-emea.com

Star Micronics Southeast Asia Co., Ltd.
Room 2902C. 29th Fl. United Center Bldg.
323 Silom Road, Silom Bangrak,
Bangkok 10500
Thailand
+66-2-631-1161 x 2
http://www.starmicronics.co.th/