Android Software Development Kit

How to Use StarIO for DK-AirCash

This SDK contains a Java Eclipse project for Android devices.

Supported Versions	Supported Star Interfaces	Compatible StarIO JAR File
Android OS 3.1 and Higher	Bluetooth, Wired LAN, Wireless LAN	StarIOPort3.1.jar (Default)
Android OS 2.2, 2.3, 3.0	Bluetooth, Wired LAN, Wireless LAN	StarIOPort3.1.jar (Default)
Android OS 2.1	Bluetooth, Wired LAN, Wireless LAN	StarIOPort.jar

Tools Needed:

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

In case of LAN I/F, DK-AirCash firmware Version 2.0 or later is required for this SDK.
In case of Bluetooth I/F, DK-AirCash firmware Version 1.0 or later is required for this SDK.

Table of Contents

- About this Manual
- Star Printer Compatibility Chart
- Connecting a Star Printer to an Android Device
- Configuring an Android Device for Development
- How to Import and Run the Android SDK
- Using the SDK with Star Micronics POS Printers
- Overview of how this Android SDK is designed
- StarIO - (StarIOPort3.1.jar)
- Functionality
    - Port Discovery
    - Help
    - Get Cash Drawer Firmware Information
    - Get Printer Status / Get Cash Drawer Status
    - Sample Receipt + Open Cash Drawer
    - Open Cash Drawer1 / Open Cash Drawer2
    - Cash Drawer Bluetooth Setting
- Tips for software application development when using StarIO
- Additional Resources
- ASCII Table
- SDK Version History

About this Manual

This manual is designed to help you understand StarIO and how to build an Android application to interact with Star DK-AirCash. It is important to understand the basics of the Java language. Although this SDK is for Android, there are SDKs available for many different operating systems and programming languages at our Global Support Site. Check the Developers Section of our site for the newest SDKs, technical documentation, FAQs, and many more additional resources.

Key Legend:

Warning		Explains potential issues
Avoid Doing This		Explains things not to do
Note		Provides important information and tips

CAUTION:

- Android is a trademark of Google Inc.
- The information in this manual is subject to change without notice.
- STAR MICRONICS CO., LTD. has taken every measure to provide accurate information, but assumes no liability for errors or omissions.
- STAR MICRONICS CO., LTD. is not liable for any damages resulting from the use of information contained in this manual.
- Reproduction in whole or in part is prohibited.

©2012-2015 Star Micronics Co., Ltd.

Connecting a Star Device to an Android Device

These screenshots were captured using an Android 4.0 tablet. Screenshots and wording can vary between Android operating systems.

Wired LAN/Wireless LAN Interface

Assign an IP Address to the Star Printer and connect it to the Wired LAN/Wireless LAN. Starndard Star Printers do not ship with an IP Address pre-assinged; this can set by a DHCP network.

Use Star POS Printers with the #9100 Multi Session disabled. The setting can be confirmed by Test Print which can be executed by holding down the printer's feed button while turning the printer on.

Please refer to "guidelines-ethernet_win_en.pdf" for how to confirm and change the #9100 Multi Session setting and how to set the Static IP Address.

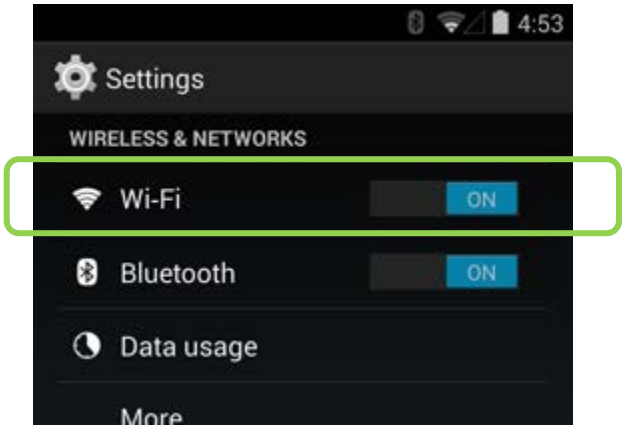
You can set POS Printers which can connect to TCP/IP by using Star Setting Utility(*). Please download it from Google play.

(*) Star Setting Utility do not support the TSP100LAN and printers which are not assinged the IP Address. (IP Address: 0.0.0.0)

Wired LAN/Wireless LAN Access Permission

1. Assign an IP Address to the Star DK-AirCash and connect it to the Wired LAN/Wireless LAN.

2. Tap Settings.

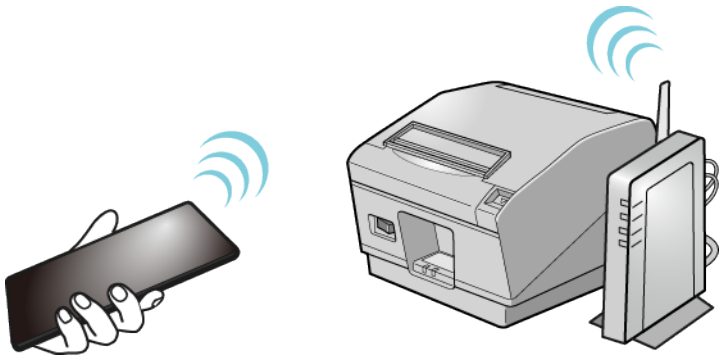


3. Ensure Wi-Fi is enabled.

4. Connect to the same network the Star DK-AirCash is on.

5. Use Search or enter the port name and IP Address in the PortName field.

6. No further settings are necessary to connect the printer as the Android device will output data directly to the device's TCP/IP port.



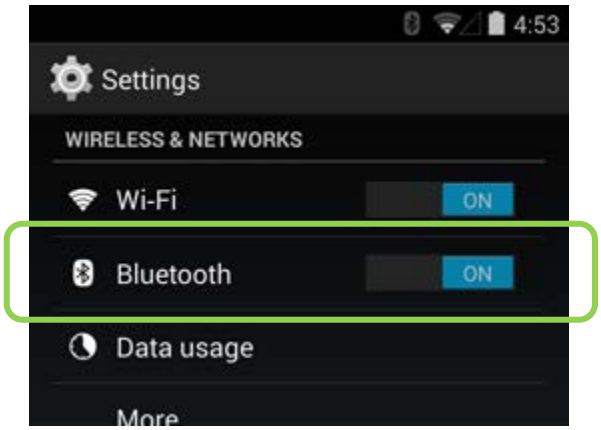
Bluetooth

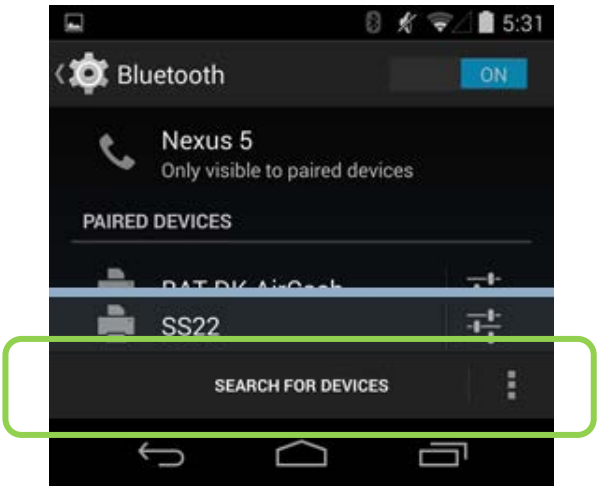
All Star Bluetooth devices have each initial device name as a factory default setting, such as "Star Micronics" and "DK-AirCash". When using multiple devices which have same device name, it is useful to change the device name for identifying each device easily.

You can change the LAN/Bluetooth settings of Star POS devices such as a device name setting by using the Star Setting Utility provided by Star Micronics. Please download it from Google play.

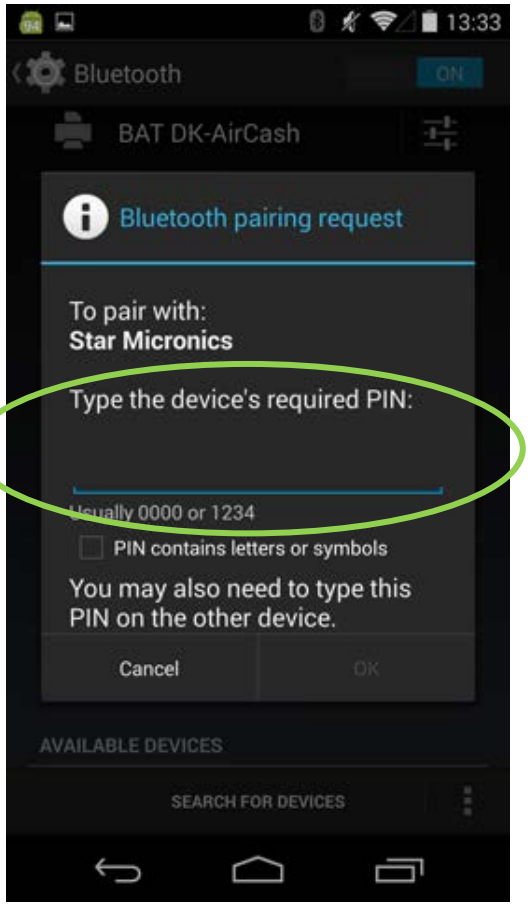
- Pairing of a Star POS device with your Android Device

1. Ensure Bluetooth is enabled and the Star POS Device is powered on.
   When the security setting of the Star POS device is set to SSP, press the PAIR button for more than 5 seconds to make it available for pairing.
2. Tap Settings.

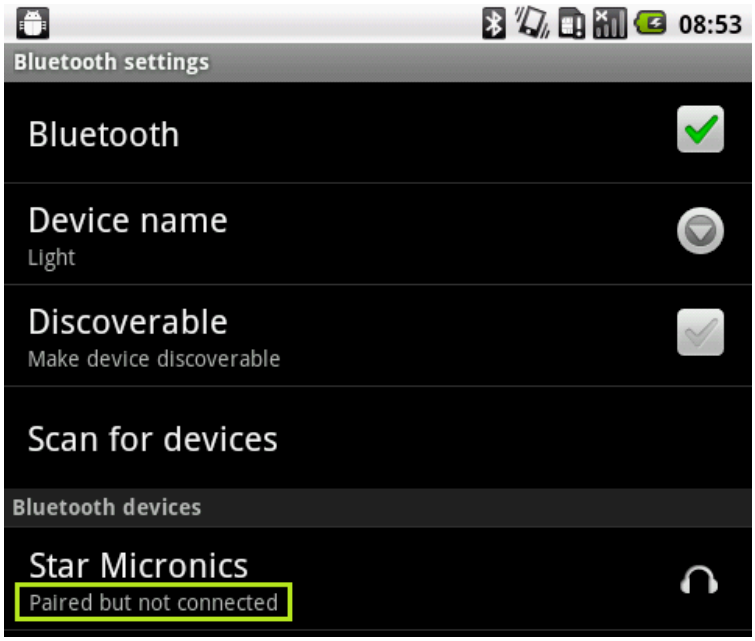




4. Tap Search for Devices.
   Find the Star POS Device you wish to connect to and tap it to pair.



5. If use the “PIN Code”, enter the PIN number. The factory default for standard Star POS Device is “1234”. (If use the “SPP”, do not need this operation.)



6. Older Android Operating Systems, such as V2.2 in the above screenshot, display the message “Paired but not connected”. This is no problem.

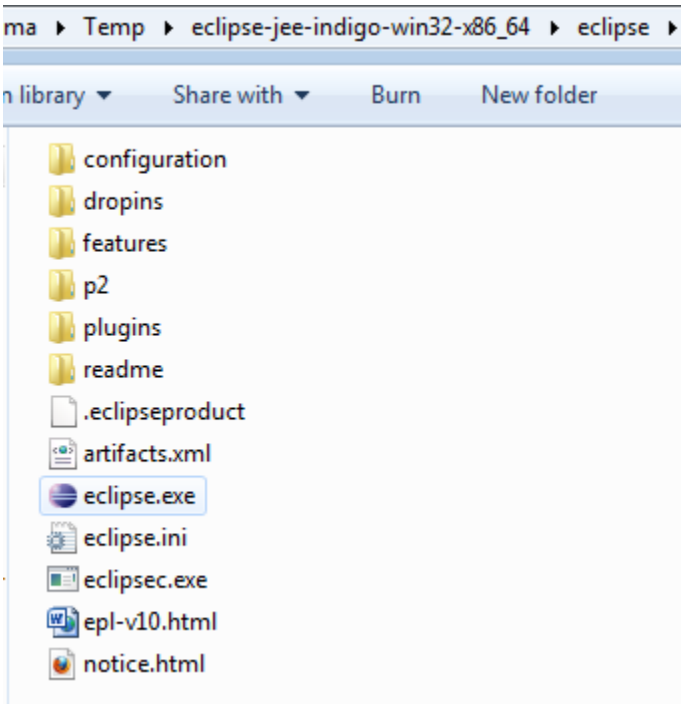
Importing and Running the StarIO Android SDK

It is assumed Eclipse has already been configured to support Android development. Should you need assistance with this, refer to Star's Android Setup Guide in the documentation folder of this SDK package.

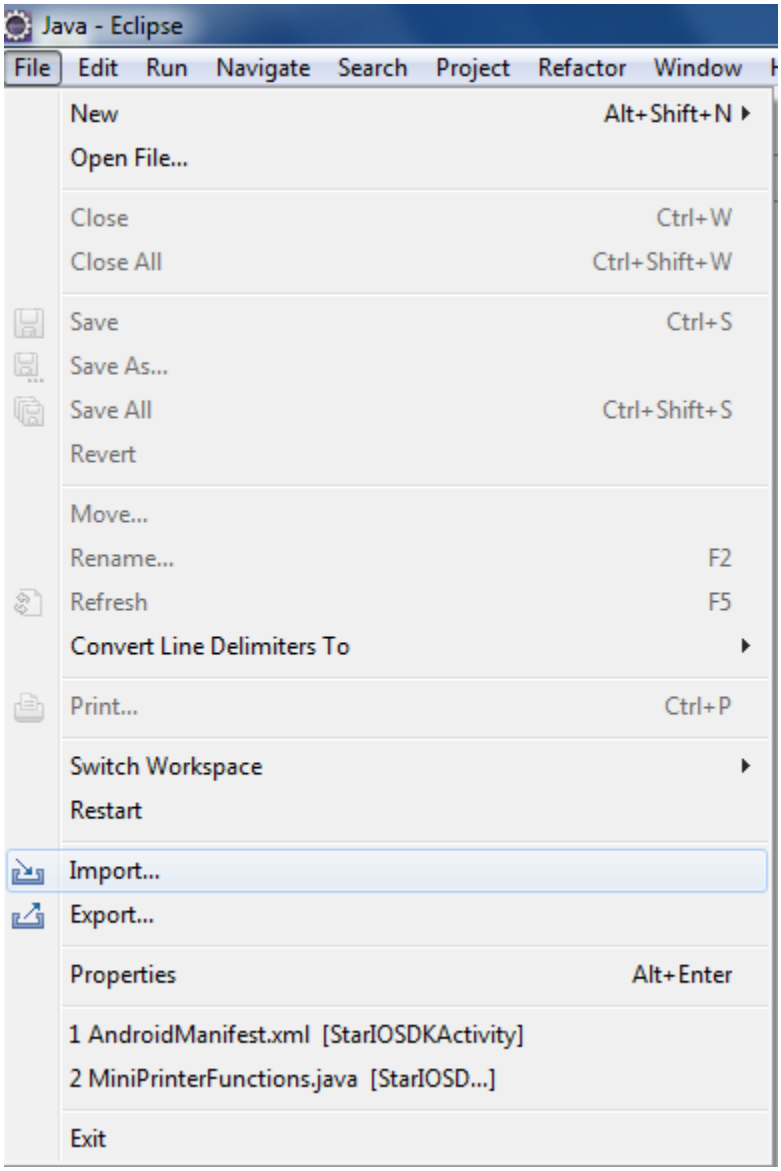
This section will explain:

1. How to import the Android SDK project into Eclipse.
2. Enabling debug mode in Eclipse.
3. Running the project.

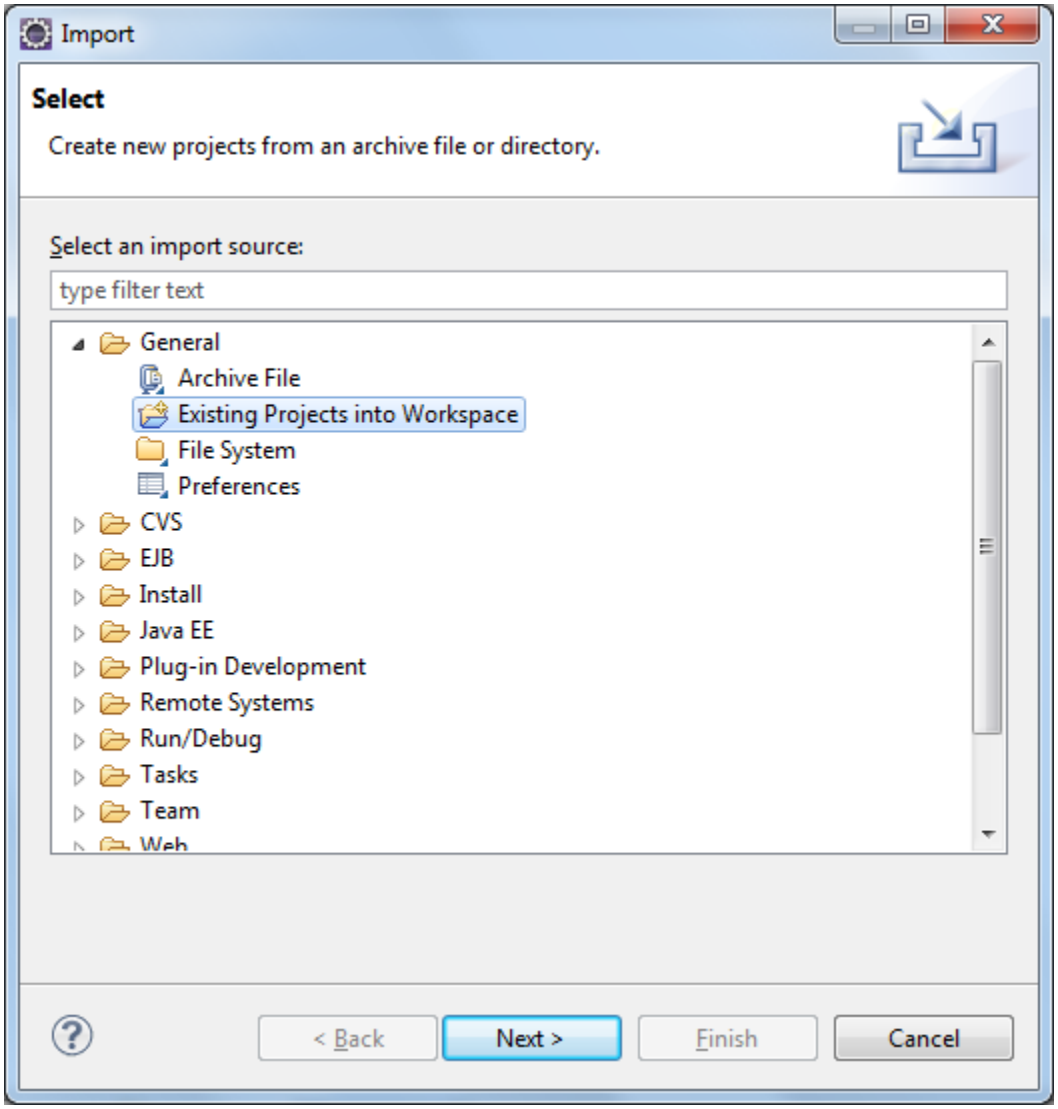
How to import the Android SDK project into Eclipse:



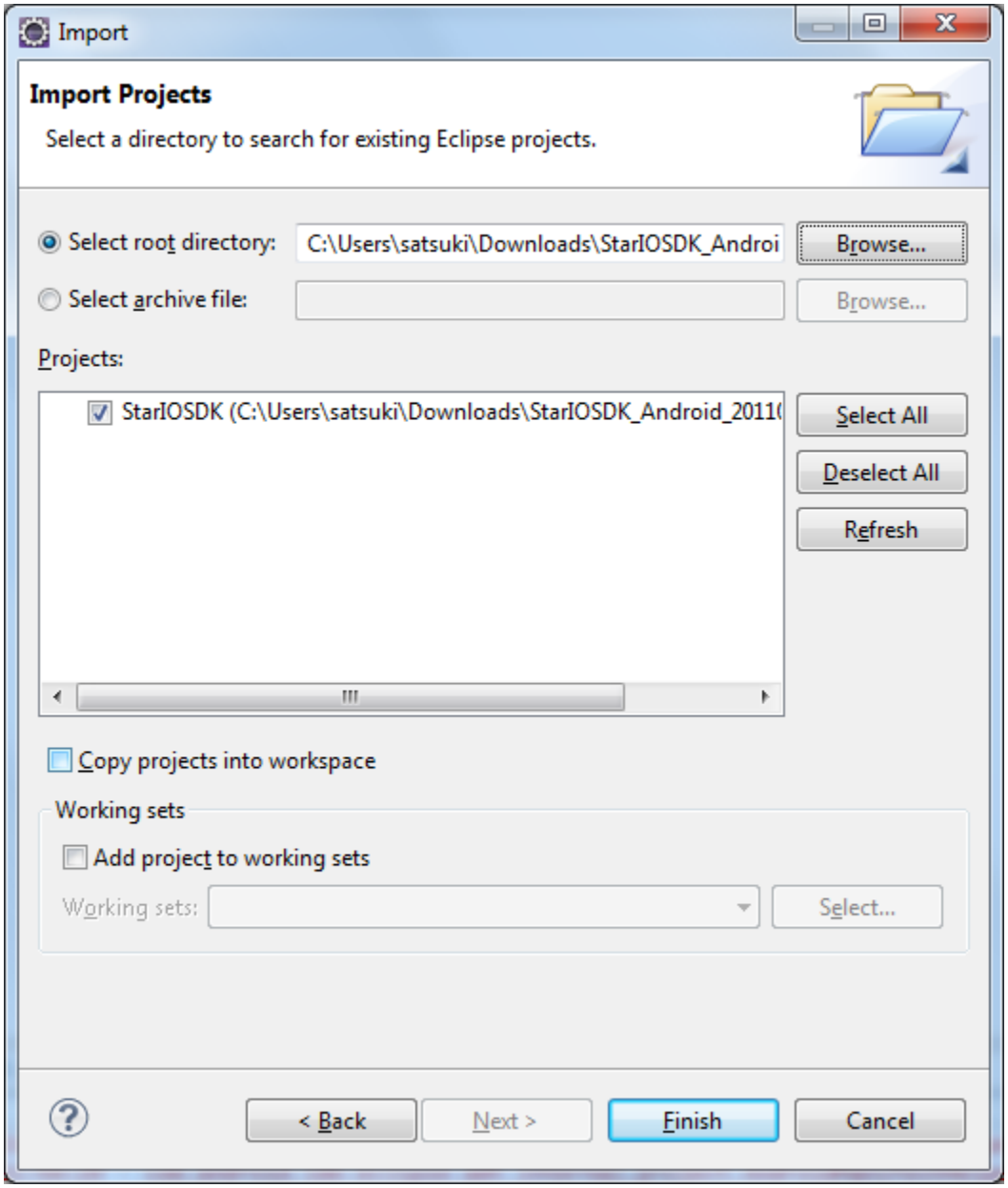
1. Execute Eclipse.



2. File > Import

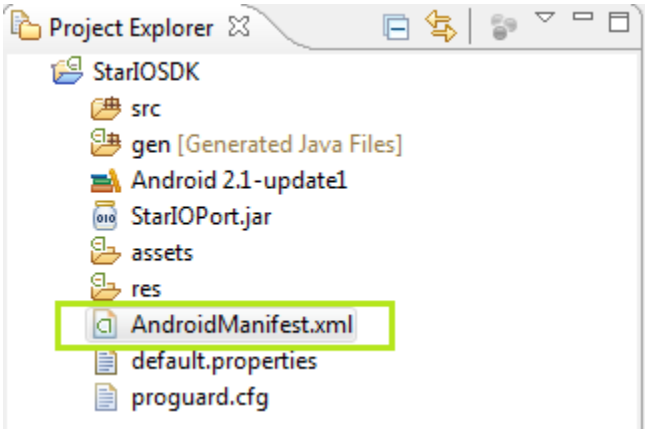


3. General > Existing Projects into Workspace

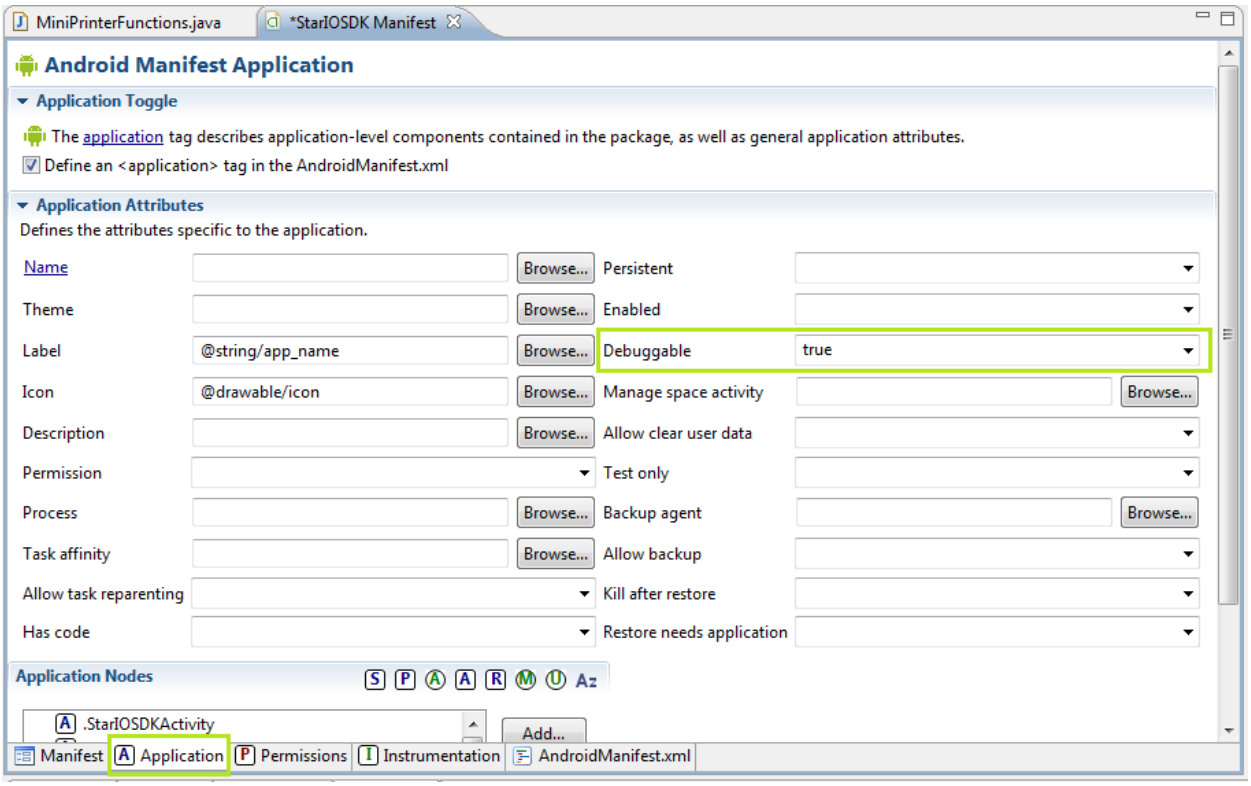


4. Browse to the location where the Star SDK is saved. Click Finish.

Enabling debug mode in Eclipse:



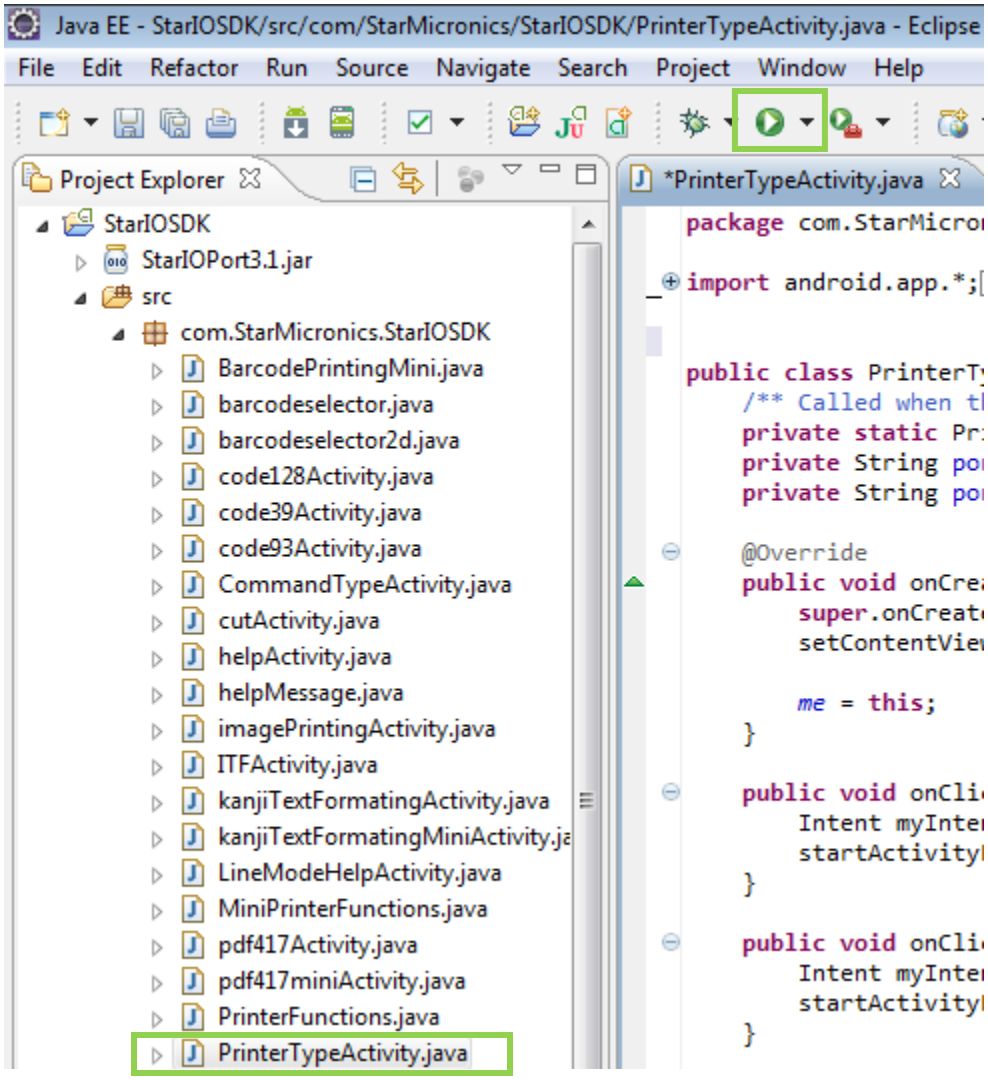
1. Click AndroidManifest.xml.



2. Click the Application tab.

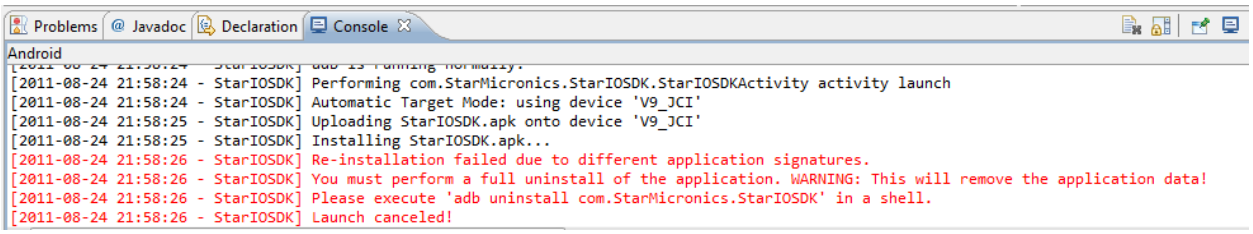
3. Set Debuggable to true.

Running the project:



1. Choose PrinterTypeActivity.java.

2. Click the green arrow.



Tip: The above error occurs if you attempt to reinstall the .apk from a different host. To remedy it, delete the application on your Android device and rerun the app.

Using the SDK with Star Devices

Port Name and Interface Relation:

StarIO uses specific port names to identify what port will be used. If these port names are not entered correctly, the application will fail to communicate with the DK-AirCash. For detailed information, please see here.

Interface	Port Name
Wired LAN **Wireless LAN**	TCP:IP Address
Bluetooth	BT: BT:Device Name BT:Printer MAC Address

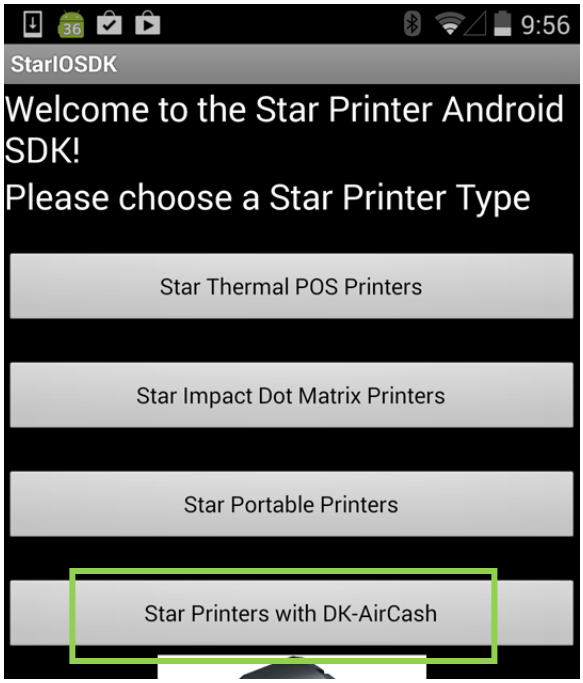
◆Notification

In case of using Bluetooth interface with PIN code setting, when you execute getPort method, some terminals(*) require you to input PIN code again.
Even though you input it, the application fails getPort method and cannot print or cannot get status.
In this case, please try the following countermeasure.

Change port settings from [portSettings = ";p";] to [portSettings = "";]

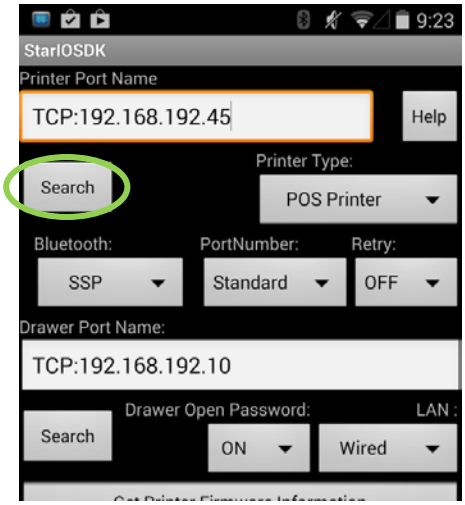
(*) We found this issue with Arrows Tab Wi-Fi FAR75A
(Android OS V.4.0.3, Japan model)

Using a DK-AirCash

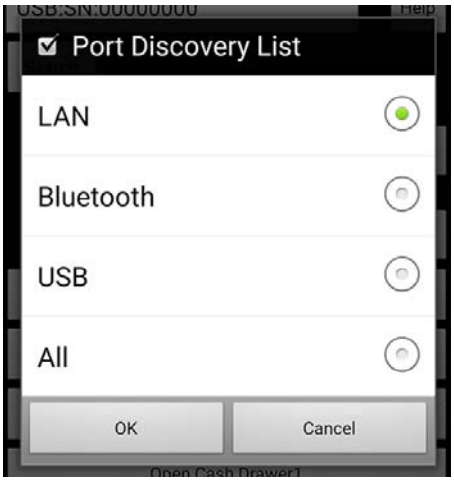


1. Tap “Star Printers with DK-AirCash”.

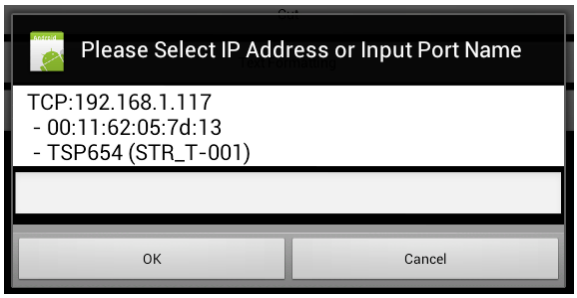
Wired LAN and Wireless LAN Printers - Port Discovery



1. Tap “Search” to find all connected Star Printers. Alternatively.

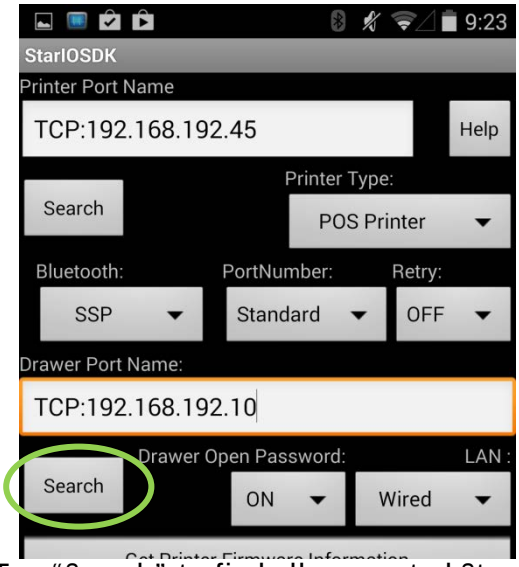


2. Tap the interface type of the printer you want to connect to.
When tapping "LAN", the model names of Wired LAN/Wireless LAN printers you can connect to be displayed.

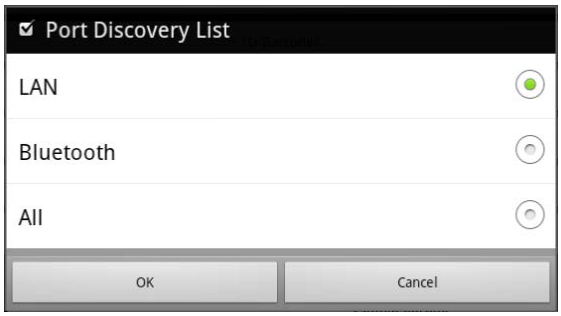


3. Tap the name of printer you want to connect to. The printer’s MAC Address and model name are displayed beneath the IP Address.

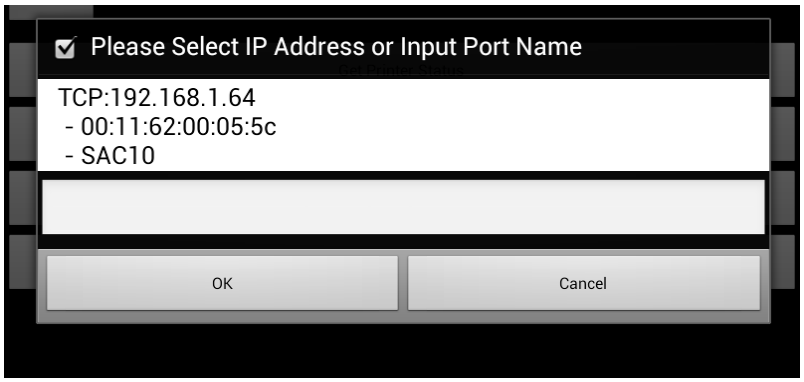
DK-AirCash - Port Discovery



1. Tap “Search” to find all connected Star DK-AirCashs.



2. Tap the interface type of the printer you want to connect to.
When tapping "LAN", the model names of Wired LAN/Wireless LAN printers you can connect to be displayed.

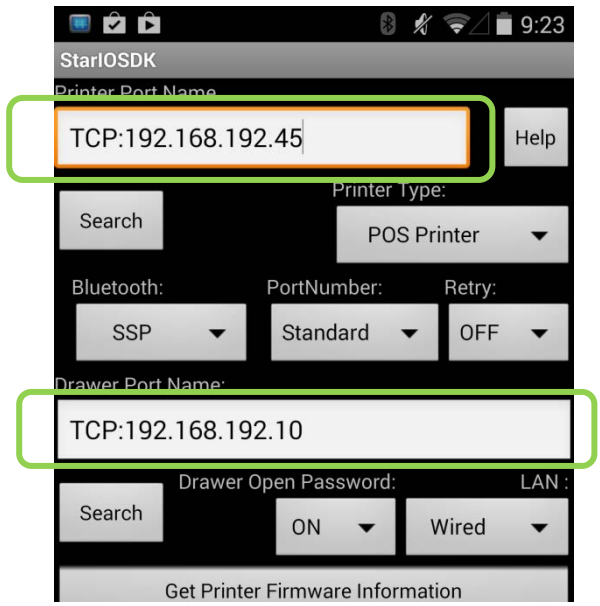


3. Tap the name of printer you want to connect to. The device’s MAC Address and model name are displayed beneath the IP Address.

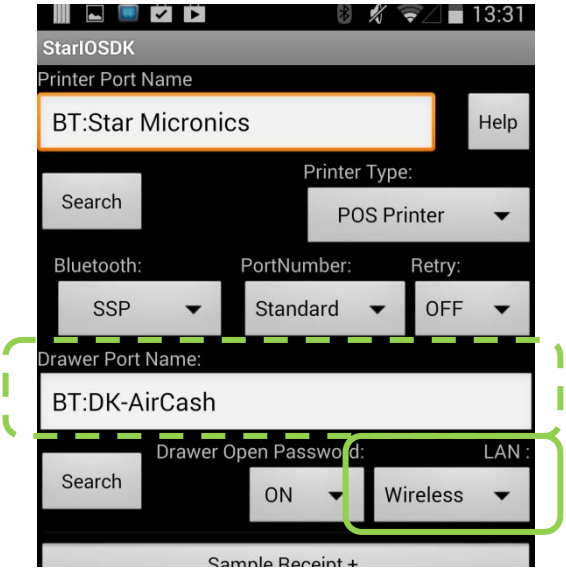
Wired LAN/Wireless LAN Printers and DK-AirCash - Manual Port Entry

The IP Address can be manually typed into the “Printer Port Name” or “Drawer Port Name” field.

If manually entered, type: TCP:<IP Address> without brackets



Wired LAN and Wireless LAN devices - How to select LAN type

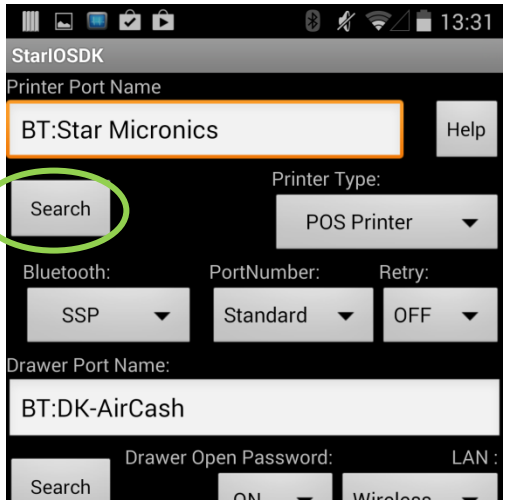


When the DK-AirCash which is specified by Drawer Port Name is Wired or Wireless LAN device, select the LAN type for your connection.

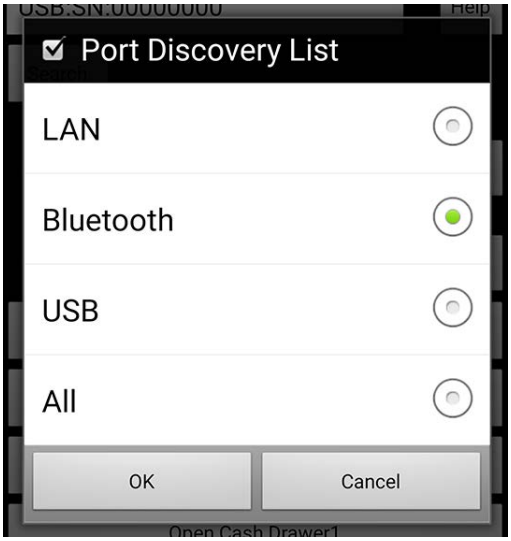
In case of wired LAN connection, select "Wired". " " (a empty string) is specified for the portSetting parameter of getPort method.

In case of wireless LAN connection, select "Wireless". ";wl" is specified for the portSetting parameter of getPort method.

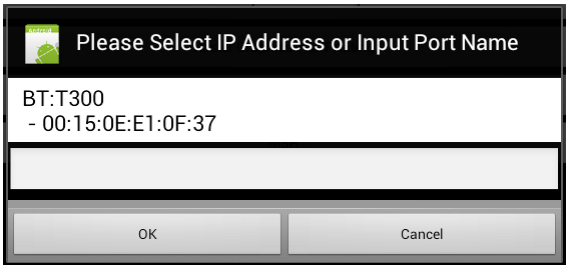
Bluetooth Printers - Port Discovery



1. Tap “Search” to find all connected Star Printers. Alternatively, the device name can be manually typed into the “PortName” field.

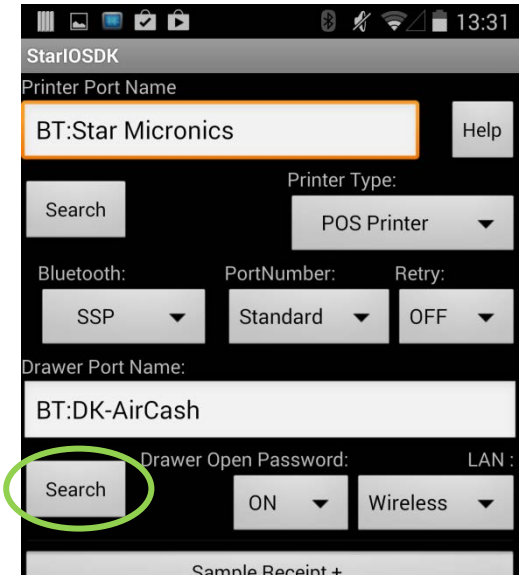


2. Tap the interface type of the printer you want to connect to.
When tapping "Bluetooth", the port names of paired printers you can connect to are displayed.

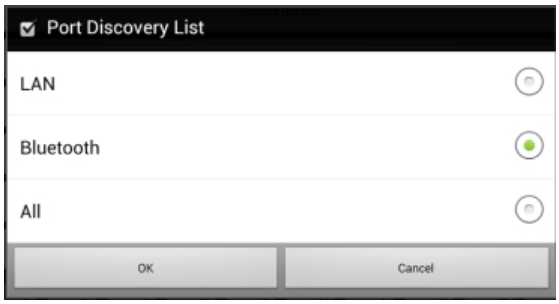


3. Tap the name of printer you want to connect to.

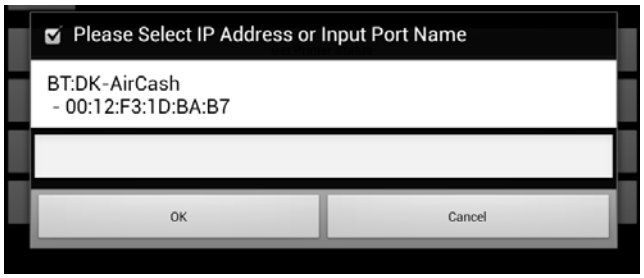
Bluetooth DK-AirCash - Port Discovery



1. Tap “Search” to find all connected Star DK-AirCashs.



2. Tap the interface type of the printer you want to connect to.
When tapping "Bluetooth", the port names of paired printers you can connect to are displayed.

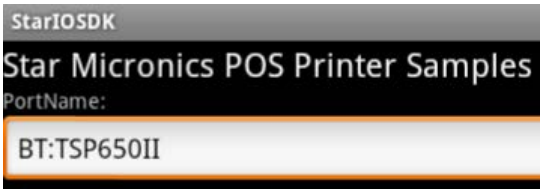


3. Tap the name of printer you want to connect to.

Bluetooth Printers and DK-AirCash – Manual Port Entry



Above: No parameters needed if using only one Star Bluetooth Device



Above: Using the Device Name as the parameter

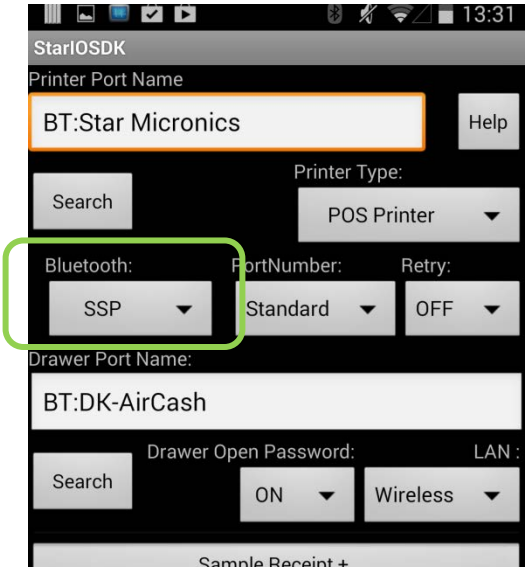
There are three types of port details that can be used for Star Bluetooth Device:

1. BT: If only one Star Bluetooth Printer is connected, using "BT:" with no parameters will automatically communicate to the only printer connected.

2. BT:<DeviceName> Add the full device name after "BT:" without brackets. This is case sensitive so ensure it matches exactly.

3. BT:<MAC Address> Add the device's MAC Address after "BT:" without brackets.

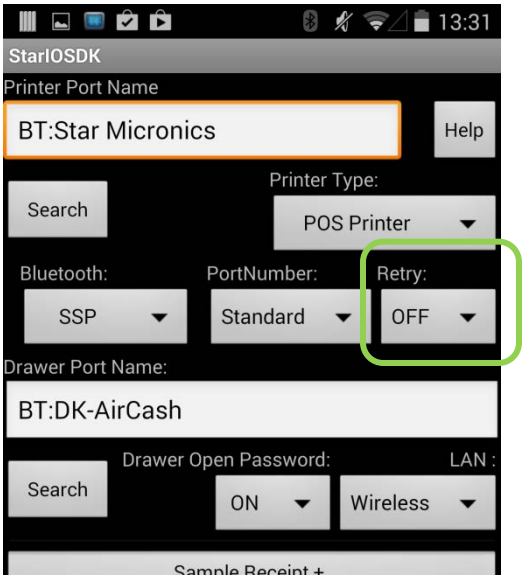
Bluetooth Device - Security Method



This setting shifts the Bluetooth API used for the Connect function of StarIO.
Select the same security method which was used when your Star Bluetooth devices were paired.

When the SSP is selected, "" (the empty string) will specified for the portSetting parameter of the getPort method if the retry is OFF and ";l" if it is ON.
When the PINCode is selected, ";p" will be specified for the portSetting parameter of the getPort method if the retry is OFF and ";p;l" if it is ON.

Bluetooth Device - Retry Setting



This setting shifts the Connection retry setting for StarIO between ON and OFF.
The retry time value is the timeout value specified in getPort method parameter.

When this setting is ON, a retry will be performed.
";l" will be specified for the the portSetting parameter of the getPort method if the security method is SSP and ";p;l" if it is PIN Code.

When this setting is OFF, no retry will be performed.
"" (the empty string) will specified for the portSetting parameter of the getPort method if the security method is SSP and ";p" if it is PIN Code.

Using an Android emulator (AVD)

The operation of SDK can be checked by using the AVD (Android Virtual Device).
It is impossible to communicate with Star Bluetooth/USB printers.

Overview of How This Android SDK is Designed

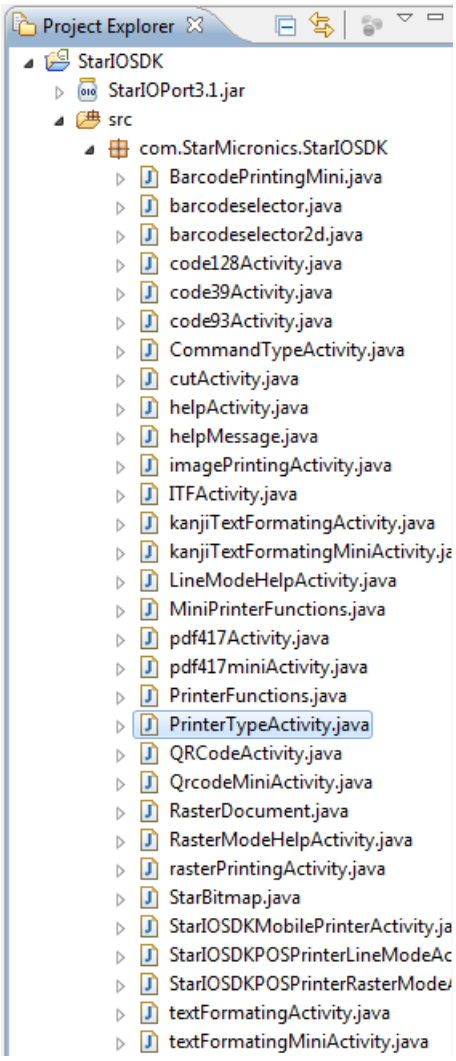
This overview will touch briefly on key components of the SDK.

All functionality is located in the src folder in the com.StarMicronics.StarIOSDK package.

Run the program by selecting PrinterTypeActivity.java; this source code is the starting point for a DK-AirCash.

See how specific functions work by clicking on the other source files. For example, “code128Activity.java” corresponds to the 1D barcode Code128 in the GUI.

It is important to note that not every function is available for a DK-AirCash. The first page of each SDK manual shows which functions are supported.



StarIO – (StarIOPort3.1.jar)

The file StarIOPort3.1.jar is a library that you can include into your Java projects to expose StarIO methods.

Selecting a StarIO Library:

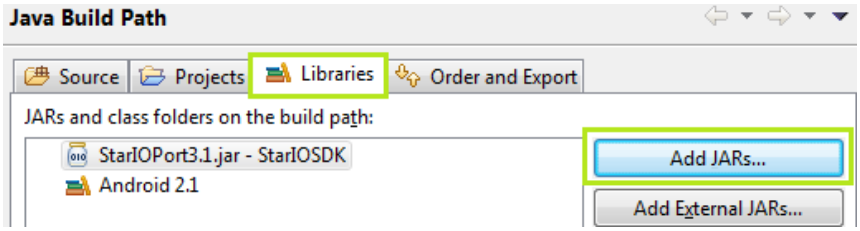
The StarIO SDK contains two library files. Use the one which corresponds to the Android OS Version your application is being developed for.

Supported Versions	Supported Star Interfaces	Compatible StarIO JAR File
Android OS 3.1 and Higher	Bluetooth, Wired LAN, Wireless LAN	StarIOPort3.1.jar (Default)
Android OS 2.2, 2.3, 3.0	Bluetooth, Wired LAN, Wireless LAN	StarIOPort3.1.jar (Default)
Android OS 2.1	Bluetooth, Wired LAN, Wireless LAN	StarIOPort.jar

How to include StarIO into your project:

To include this library into your project:

1. Drag the appropriate StarIO library file into the Project Explorer from the SDK package
2. Right click the project folder and choose Properties
3. Click Java Build Path



4. Click Libraries and the Add JARs button
5. At the top of your main code add:

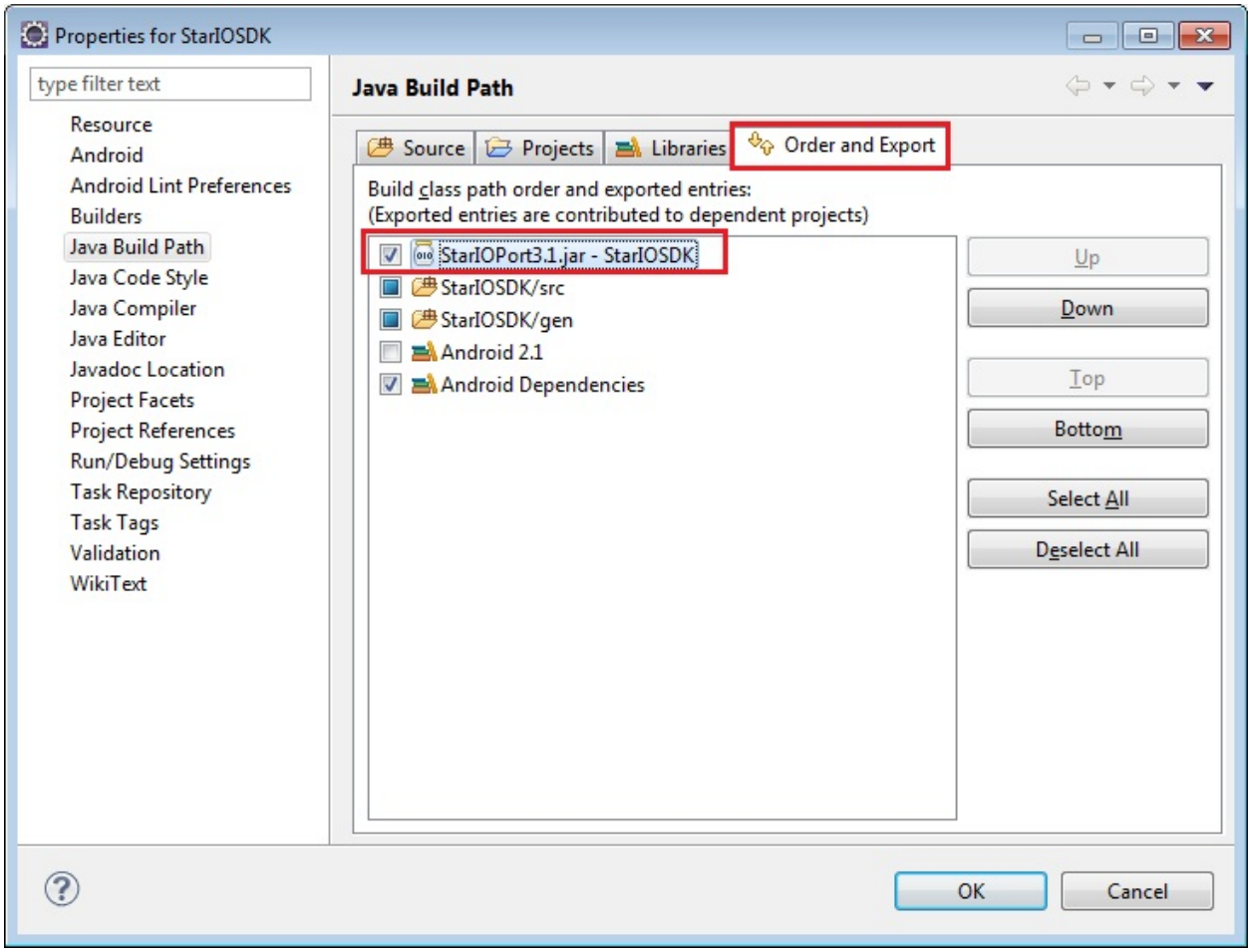
```
import com.starmicronics.stario.StarIOPort;
import com.starmicronics.stario.StarIOPortException;
import com.starmicronics.stario.StarPrinterStatus;
```

6. Now you can access all of StarIO's methods!

Android SDK Tools r17 and Higher:

Library management has been changed starting from Android SDK Tools r17. It is essential to order the StarIOPort JAR before StarIOSDK/src. This method presented no problem from Android SDK Tools r16 and lower, but caused the application to crash beginning with r17.

Star Micronics rectified this issue in the StarIO Android SDK V2.4. Please ensure the same caution is taken when using the StarIOPort JAR in your own application.



1. Right click the project name and select properties

2. Click Java Build Path and then Order and Export

3. Highlight the StarIOPort JAR and relocate it to above the application code

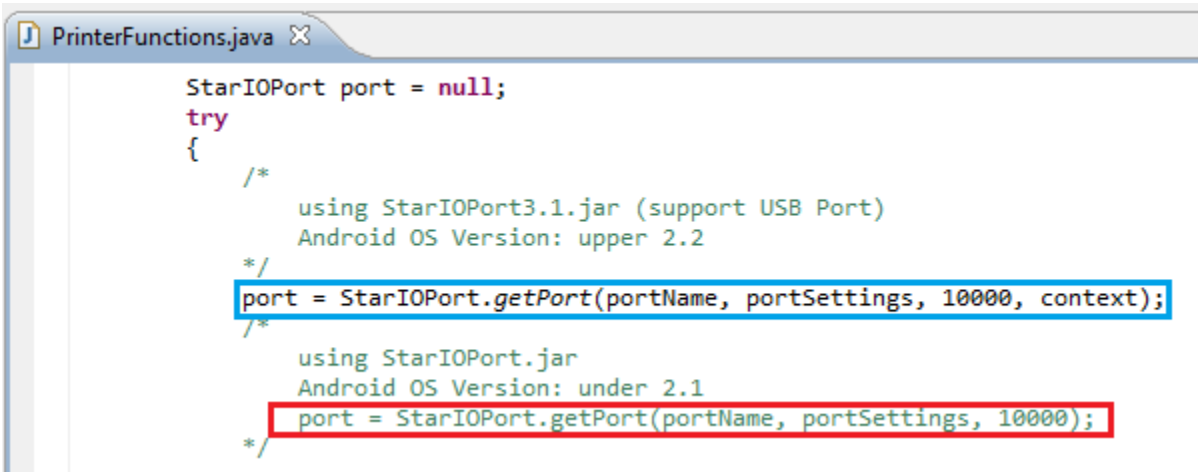
Android OS V2.2 and Higher:

Include the default library: StarIOPort3.1.jar
Upon including this library, the StarIO SDK application will successfully run. No modifications to the source code are necessary.

Android OS 3.1 and higher: Supports Bluetooth and Wired LAN/Wireless LAN connections.
Android OS 2.2, 2.3, 3.0: Supports Bluetooth and Wired LAN/Wireless LAN connections.

Android OS V2.1:

Include the legacy library: StarIOPort.jar
Upon including this library, the source code of the StarIO SDK application must be modified.



1. Comment this line: port = StarIOPort.getPort(portName, portSettings, 10000, context);
2. Uncomment this line: port = StarIOPort.getPort(port Name, portSettings, 10000);

This allows for the use of the legacy StarIOPort.jar library.

Android OS 2.1 supports Wired LAN/Wireless LAN and Bluetooth connections.

StarIO Methods Overview

StarIO Port class

StarIO Port class includes 3 properties; portName (String), portSettings (String), and Timeout (int).



These 3 variables will be "read only" if accessed directly. To assign them use getPort(portName,portSettings,timeout); which will allow you to pass in variables to this methods which then assigns the 3 class variables with values.

portName is what you will be using to specify the port of communication to the DK-AirCash.

Ex. TCP:192.168.1.2

portSettings specify the following settings depending on your application.

- LAN type setting
- Security method setting
- Retry setting

A blank string should be specified for interfaces other than Bluetooth.

Interface	Port Name	Port Settings	
Wired LAN **Wireless LAN**	TCP:IP Address	the empty string	Wired LAN
		;wl	Wireless LAN
Bluetooth	BT:	the empty string	Use SSP, Retry OFF
	BT:Device Name	;p	Use PIN Code
	BT:Printer MAC Address	;l	Retry ON

Example of Port Setting Property :

portSettings = ";p;l" *⋯ In case of using the PIN code and retry ON*

timeout is a millisecond timeout controlled internally and is used for communication in the APIs (this parameter guarantees that all of the below APIs will complete in a bounded amount of time, but does NOT guarantee the exact timeout length)

● Method

getPort

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

getPort is what you will be using to "open" the port to the DK-AirCash. Using one of the valid inputs for portName and portSettings, you can connect to the DK-AirCash.

```
//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
}
```

`IPort` is a part of StarIO and this will allow you to create a "port" handle.

The above example shows the port being created and set to null then being assigned the actual port hook on the following line that contains getPort.



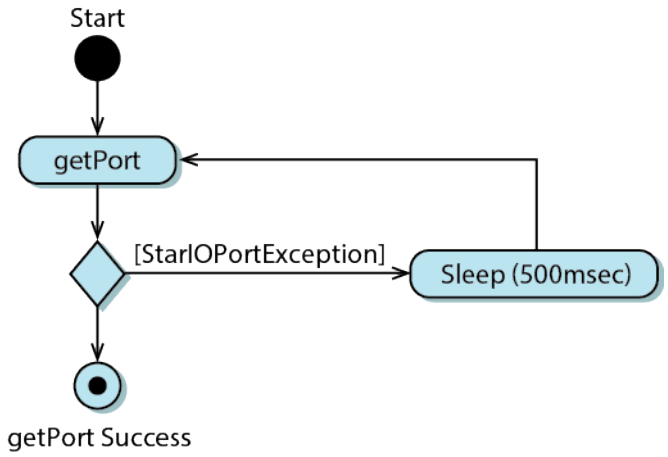
Always use a try, catch when using **getPort**. If the port cannot be opened because of connection problems, your program will crash unless you use a try, catch like the above example.

In case your application sends data to bluetooth DK-Aircash, the getPort method may sometimes fail to get status or print as the result of failing to connect with the bluetooth device. The rate of failure depends on the host device or possibly other conditions.

When ";l" or ";p;l" is specified for the portSetting parameter of the getPort method, a retry will be performed in case of a failure of connection with the bluetooth device.
The retry will be repeated during a time specified in the timeout parameter and the StarIOPortEX\xception will occur if a connection does succeed within a timeout time.

When getPort fails a StarIOPortException is thrown. Please repeat the operations mentioned in article 2 and 3 below (see getPort Retry Flow):

◆**getPort Retry Flow**



1. getPort fails
   (StarIOPortException is thrown).
2. Wait for certain time (e.g. 500ms)
3. Execute getPort again
4. When getPort is successful, go to procedures such as printing or getting status.

(Remark)
The number of attempts to execute getPort to obtain a port handle depends on the Host device(Android Device). Some devices have a higher failure rate than others, and may require additional attempts to getPort. Please decide based on tests conducted with your Android device.

searchPrinter

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

searchPrinter detects DK-AirCashes in LAN and return search result as ArrayList<PortInfo>.

Set “TCP:” as Argument value target.

PortInfo class of return value includes, PortName, MAC address, and you can get them by String getPortName(), String getMacAddress(), and String getModelName() method.)

And you can use Port Name as Argument value of getPort.

```
//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
}
```

This sample shows “detect DK-AirCash in LAN as a list, and output to log.



This API do not guarantee the discovery of devices.

readPort

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

This method reads data from the device. Only use this if you really need to read raw bytes from the printer.



Do not use this method to read raw status.
Use RetreiveStatus for getting status.

Parameters:

`readBuffer` - A Byte Array buffer into which data is read.

`offset` - specifies where to begin writing data into the readBuffer[]

`size` - Total number of bytes to read.

Returns:

The number of bytes that were actually read. Under some interface types, this function will succeed even when no data was read in. Your application should call this function a limited number of times until the expected data has been read in or until an application determined retry threshold has been reached.

Throws:

`StarIOPortException` - when a communication failure occurs

releasePort

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

This function closes a connection to the port specified.

Parameters:

`port` – StarIOPort type representing a previously initialized port.



When do not send the data, release (close) ports.
Leaving a port open will cause future calls to open the port to fail.



For Wired LAN/Wireless LAN Interface:

The getPort method may fail if you run it just after running releasePort.
You must run the getPort method after 500msec from releasePort.

writePort

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

This method writes data to the device. Use this to send commands to the DK-AirCash.
To check the completion of command transfer.

```
//Set a byte array to send to the printer
//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
}
```

Remember to use a try, catch for safe programming practices.

Parameters:

`writeBuffer` - Contains the output data in a byte array.

`offset` - Specifies where to begin pulling data from writeBuffer .

`size` - Number of bytes to write.

Throws:

`StarIOPortException` - when a communication failure occurs

retreiveStatus

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

This method retrieves detailed status from the printer with StarIO.

Returns:

`StarPrinterStatus` structure giving the current device status

Throws:

`StarIOPortException` - when a communication failure occurs

This method uses a class structure that is included with StarIO called `StarPrinterStatus`

This structure gives the printer's status in both boolean and binary form.

Create the `StarPrinterStatus` object in your project by doing the following:

```
StarPrinterStatus status = port.retreiveStatus();
if (status.offline == false)
{
    if (status.blackMarkError == true) {
        //There was a black mark error
    }

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

beginCheckedBlock

public StarPrinterStatus **beginCheckdBlock ()** throws StarIOPortException

This method is used in combination with endCheckedBlock and checks the completion of printing. To check if the whole data is completely printed, you need to run this method just before sending print data and endCheckedBlock just after sending print data.

Returns:

`StarPrinterStatus` structure giving the current device status

Throws:

`StarIOPortException` - when a communication failure occurs
- when the printer is off line

See the sample code here.

Caution in using the button security function (Only for the North America market)

beginCheckedBlock reads the timeout setting of the DK-AirCash and sets setEndCheckedBlockTimeoutMillis to the proper value automatically.

endCheckedBlock

public StarPrinterStatus **endCheckdBlock ()** throws StarIOPortException

This method monitors DK-AirCash status and when the transferred data is processed completely by DK-AirCash, returns control to the application.

This method is used together with the beginCheckedBlock method as a pair.

In case the transfer of data is not completed before the timeout (*1) or an error occurs during command execution, StarIO throws a StarIOPortException.

(*1) To set the timeout value for endCheckedBlock execution time, please call setEndCheckedBlockTimeoutMillis method of StarIOPort object.

The default timeout value is the value specified in getPort method parameter.

Please adjust the endCheckedBlockTimeoutMillis value to be longer than printing time. If printing is not complete before endCheckedBlockTimeoutMillis value is reached, a StarIOPortException will be thrown.

The minimum timeout value that can be specified in the getPort or setEndCheckedBlockTimeoutMillis methods is 10 seconds. If less than 10 seconds is specified, StarIO will default to 10 seconds.

Returns:

`StarPrinterStatus` structure giving the current device status

Throws:

`StarIOPortException`

- An error sending the command (such as Off-Line)
- No response for the completion of printing from a printer within the timeout

See the sample code here.

Caution in using the button security function (Only for the North America market)

If this method is used after having sent drawer open command, it returns control either when button is pressed or timeout is over before the button is pressed.

You can confirm the drawer status by getParsedStatus method 150ms after control is returned.

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

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

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

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

      if (status.offline == true) {
         //If true, then the printer 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 vold **setEndCheckedBlockTimeoutMillis (** int endCheckedBlockTimeoutMillis**)**

This method sets the endCheckedBlock method's timeout value [unit: ms]

If it takes a long time to print, please increase this value to allow for enough time to complete the print job.

Default value is the timeout value designated by getPort method.



Timeout length is 10 seconds if specified less than 10 seconds.

Caution in using the button security function (Only for the North America market)

If the button security is utilized, please call beginCheckedBlock prior to and endCheckedBlock posterior to the drawer open command transference.

Please do not use this method, since endCheckedBlock method timeout value is automatically adjusted optimally by beginCheckedBlock method.

getFirmwareInformation

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

This method gets a model name and firmware version of the printer and returns Map<String, String> as an acquisition result.

Returns:
Gets a model name from the return value by setting the Key to “ModelName”.
Gets a firmware version from the return value by setting the Key to “FirmwareVersion”.

Throws:

StarIOPortException - when a communication failure occurs

Note:

・If it failed to get information, it returns an empty string.

getStarIOVersion

```
public Static synchronized String  getStarIOVersion()
```

This method gets a StarIO Version of the printer.

Returns:
StarIO Version

getDipSwitchInformation

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

This method gets the DIP switch setting information of the DK-AirCash.
It returns Map<String, Boolean> as an acquisition result.

Returns:
Gets the Dip switch setting information from the return value by setting the Key to "DIPSW11" or "DIPSW12".

Valid character strings for KEY
"DIPSW11", "DIPSW12", "DIPSW13", "DIPSW14",
"DIPSW15", "DIPSW16", "DIPSW17", "DIPSW18",
"DIPSW21", "DIPSW22", "DIPSW23", "DIPSW24",
"DIPSW25", "DIPSW26", "DIPSW27", "DIPSW28",

`true` - in case the DIP switch is ON
`false` - in case the DIP switch is OFF

Throws:
`StarIOPortException` - when a communication failure occurs
- when used with a model other than the DK-AirCash

SMBluetoothManager Class:

SMBluetoothManager Class specifies various settings of the Bluetooth interface.
It can not be used with SMPort Class.

● Constructor

StarBluetoothManager

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

Parameters:

portName – the port name for connecting to the device.
Ex. BT:StarMicronics

portSetting - the port setting for connecting to the device.
Ex. Specify ";p" when using the PIN code with Bluetooth interface.
Specify the empty string when SSP is selected.

timeoutMillis
- timeout is a millisecond timeout controlled internally and is used for communication in the APIs (this parameter guarantees that all of the below APIs will complete in a bounded amount of time, but does NOT guarantee the exact timeout length)

starDeviceType
– the type of the device to be searched
Ex. StarDeviceTypeDesktopPrinter

Throws:

StarIOPortException – when starDeviceType is set to StarDeviceTypePortablePrinter
– when portName is not set to BT:

● Method

open

```
public void open()
```

This method is used to open connection to the star Bluetooth device.

Throws:

`StarIOPortException` - when a communication failure occurs

close

```
public void close()
```

This method is used to close communication with the star Bluetooth device.

Throws:

`StarIOPortException` - when a communication failure occurs

apply

```
public void apply()
```

StarBluetooth device is set to the value specified by the following methods.

- setBluetoothDevice
- setiOSPortName
- setAutoConnect
- setPinCode
- setSecurityType

Throws:

`StarIOPortException` - when a communication failure occurs
- when SecurityType is set to PIN code and AutoConnect is set to ON

loadSetting

```
public void loadSetting()
```

This method gets the value specified from the star Bluetooth device.

Throws:

`StarIOPortException` - when a communication failure occurs

getBluetoothDeviceName

```
public String getBluetoothDeviceName()
```

This method gets the Bluetooth device name which was acquired by the loadSetting method or specified by the setBluetoothDeviceName method. Use this method after calling the loadSetting method or the setBluetoothDeviceName method.

Returns:

BluetoothDeviceName

setBluetoothDeviceName

```
public void setBluetoothDeviceName( string BluetoothDeviceName)
```

This method sets the Bluetooth device name of at least 1 character, with a max length of 16 characters.

To change the Bluetooth device name, execute the apply method after the setBluetoothDeviceName method.

Parameters:

BluetoothDeviceName

Throws:

StarIOPortException
- when any invalid characters are used
- when the length of the set character string is not between 1 to 16 characters.

Valid characters:

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

getiOSPortName

```
public String  getiOSPortName()
```

This method gets the iOS port name which was acquired by the loadSetting method or specified by the setiOSPortName method. Use this method after calling the loadSetting method or the setiOSPortName method.

Returns:

iOSPortName

setiOSPortName

```
public void  setBluetoothDeviceName( string iOSPortName)
```

This method sets the iOS port name of at least 1 character, with a max length of 16 characters.

To change the iOS port name, execute the apply method after the setiOSPortName method.

Parameters:

iOSPortName

Throws:

StarIOPortException	– when any invalid characters are used – when the length of the set character string is not between 1 to 16 characters.

Valid characters:

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

getPinCode

```
public String getPinCode()
```

This method gets the PIN code which was specified by the setPinCode method. Use this method after calling the setPinCode method.

Returns:

PinCode

setPinCode

```
public void setPinCode( string pinCode)
```

This method sets the PIN code of Star Bluetooth Device of at least 4 character, with a max length of 16 characters.

To change the PIN code, execute the apply method after the setPinCode method.

Parameters:

PinCode *Ex.* 1234

Throws:

StarIOPortException
- when any invalid characters are used
- when the length of the set character string is not between 4 to 16 characters.

Valid characters: 0-9 a-z A-Z

getAutoConnect

```
public boolean getAutoConnect()
```

This method gets the value specified of Auto Connection which was acquired by the loadSetting method or specified by the setAutoConnect method. Use this method after calling the loadSetting method or the setAutoConnect method.

Returns:

true - when AutoConnection setting is ON
false - when AutoConnection setting is OFF



When using Android devices, set the AutoConnection to OFF.

setAutoConnect

```
public void setAutoConnect( boolean autoConnect)
```

This method sets the Auto Connection of Star Bluetooth Device.

To change the Auto Connection, execute the apply method after the setAutoConnect method.

Parameters:

autoConnect *Ex.* false



When using Android devices, set the AutoConnection to OFF.

getSecurityType

public StarBluetoothManager.StarBluetoothSecurity **getSecurityType()**

This method gets the Bluetooth security type(SSP or PIN Code) which was acquired by the loadSetting method or specified by the setSecurityType method. Use this method after calling the loadSetting method or the setSeurityType method.

Returns:

StarBluetoothManager.StarBluetoothSecurity

setSecurityType

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

This method sets the Bluetooth security type(SSP or PIN Code) of star Bluetooth device.

To change the Bluetooth security type, execute the apply method after the setSecurityType method.

Parameters:

securityType *Ex.* StarBluetoothManager.StarBluetoothSecurity.SSP

getPortName

public String **getPortName()**

This method gets the port name specified by the StarBluetoothManager constructor.

Returns:

portName

getPortSetting

public String **getPortSetting()**

This method gets the port setting specified by the StarBluetoothManager constructor.

Returns:

portSetting

getTimeoutMillis

```
public String getTimeoutMillis()
```

This method gets the TimeoutMillis specified by the StarBluetoothManager constructor.

Returns:
timeoutMillis

getDeviceType

```
public String getDeviceType()
```

This method gets the star Device type specified by the StarBluetoothManager constructor.

Returns:
StarBluetoothManager.StarDeviceType

isOpened

```
public boolean isOpened()
```

This method acquires the port status.

Returns:
true - The port is opened.

Status List of the class structure StarPrinterStatus

Member name	Contents	Type	Detail	DK-AirCash
blackMarkError	Black Mark Error	boolean	DK-AirCash is not supported.	
compulsionSwitch	Compulsion SW	boolean	You can check status of CashDrawer (Open or Close) " true " : Compulsion SW is pressed. " false " : Compulsion SW is not pressed.	✓
coverOpen	Cover Status	boolean	DK-AirCash is not supported.	
cutterError	Auto-cutter Error	boolean	DK-AirCash is not supported.	
etbAvailable	ETB available or not	boolean	" true " : available to use " false " : not available to use	✓
etbCounter	ETB Counter	int	You can get current value of ETB	✓
headThermistorError	Head Thermistor Error	boolean	DK-AirCash is not supported.	
offline	ONLINE/OFFLINE Status	boolean	You can check status of Online or offline. " true " : Printer is Offline. " false " : Printer is Online	✓
overTemp	Stopped by high head temperature	boolean	DK-AirCash is not supported.	
presenterPaperJamError	Presenter Paper Jam Error	boolean	DK-AirCash is not supported.	
presenterState	Presenter Paper Position	int	DK-AirCash is not supported.	
raw	Byte column of status	byte[63]	Byte column of status (example : HEX 23 86 00 00 00 00 00 00 00)	✓
rawLength	raw length	int	raw length	✓
receiptPaperEmpty	Paper end	boolean	DK-AirCash is not supported.	
receiptPaperNearEmptyInner	Paper Near-end (Inner Side)	boolean	DK-AirCash is not supported.	
receiveBufferOverflow	Receive Buffer Overflow	boolean	DK-AirCash is not supported.	
unrecoverableError	Non-recoverable Error	boolean	" true " : Unrecoverable error occurs. " false " : Unrecoverable error does not occur. Unrecoverable error : Head Thermistor Error, Auto-cutter Error, Electric Voltage Error and etc.)	✓
voltageError	Electric Voltage Error	boolean	DK-AirCash is not supported.	

StarIO Android SDK Functionality

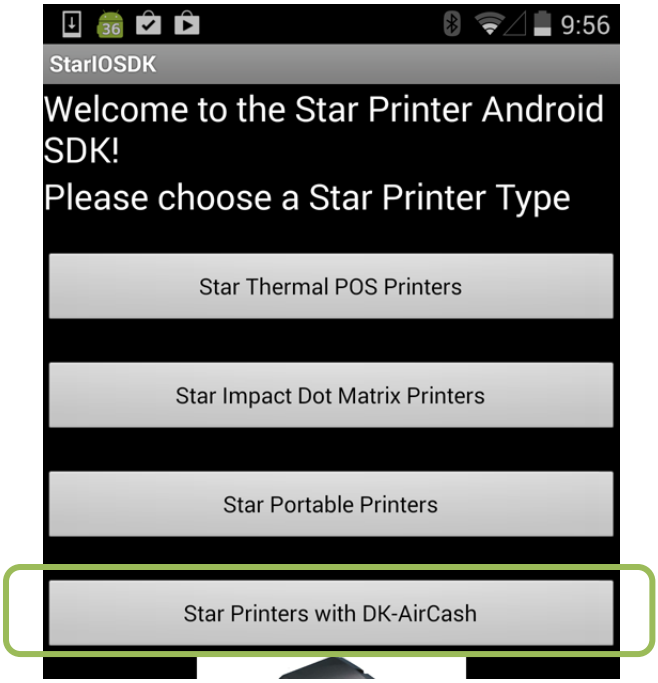
Overview of this SDK functionality and StarIO Printer Commands

All of these commands can be found in the Star Line Mode Command Manual.

This SDK also has page and section references to the Line Mode Manual for more information so please download and study it if you need more detail on a specific command.

Choosing a Printer and Communication Type

1. Tap “Star Printers with DK-AirCash”.



Supported Samples by Command Type

Get Printer Firmware Information

Get DK-AirCash Firmware Information

Get DK-AirCash Dip Switch Information

Get Printer Status

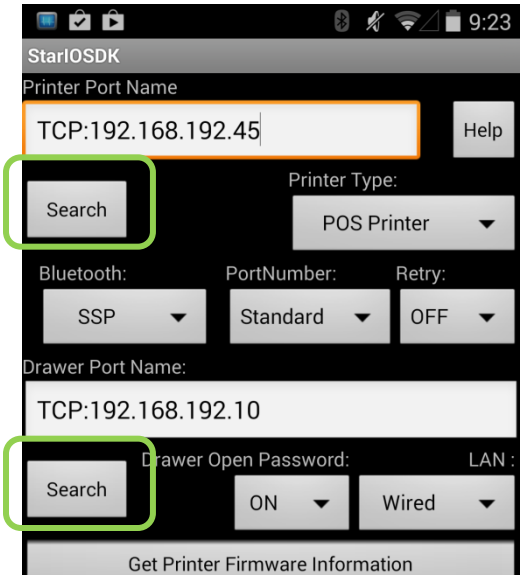
Get DK-AirCash Status

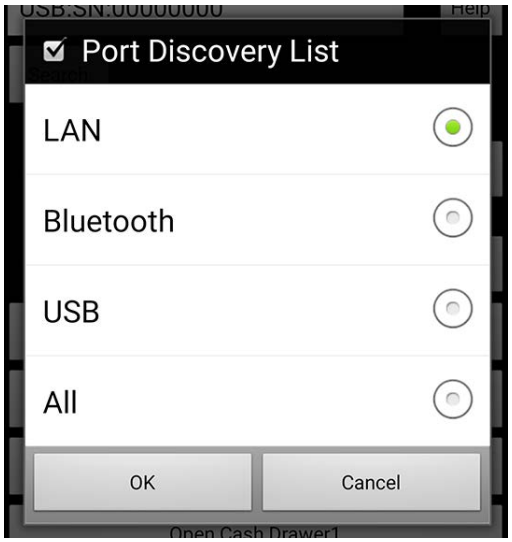
Sample Receipt + Open Cash Drawer1 via DK-AirCash

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

DK-AirCash Bluetooth Setting

Port Discovery

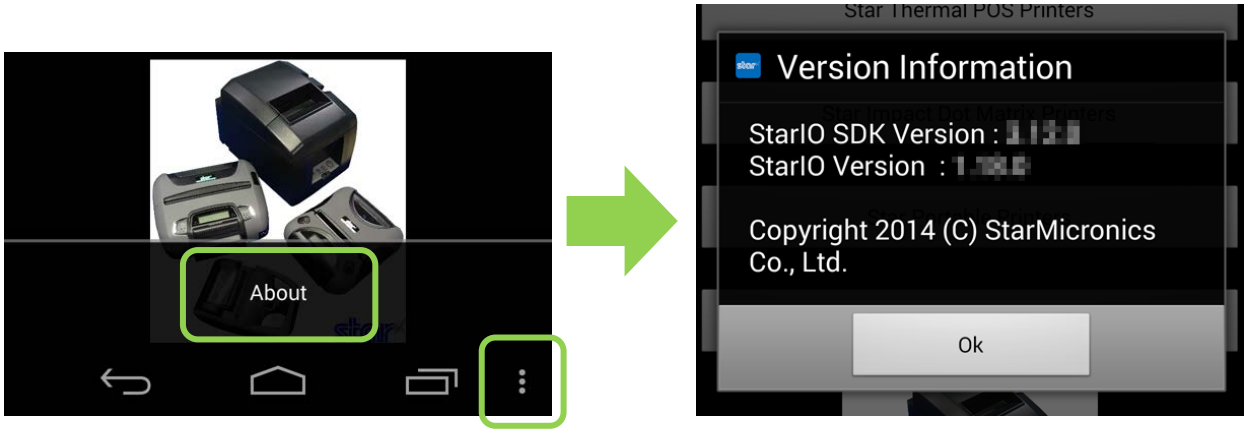




Automatically detects which Star Micronics POS devices are connected to the Wired LAN/Wireless LAN. Tap the printer to connect to it. [This feature is documented in greater detail here](#).

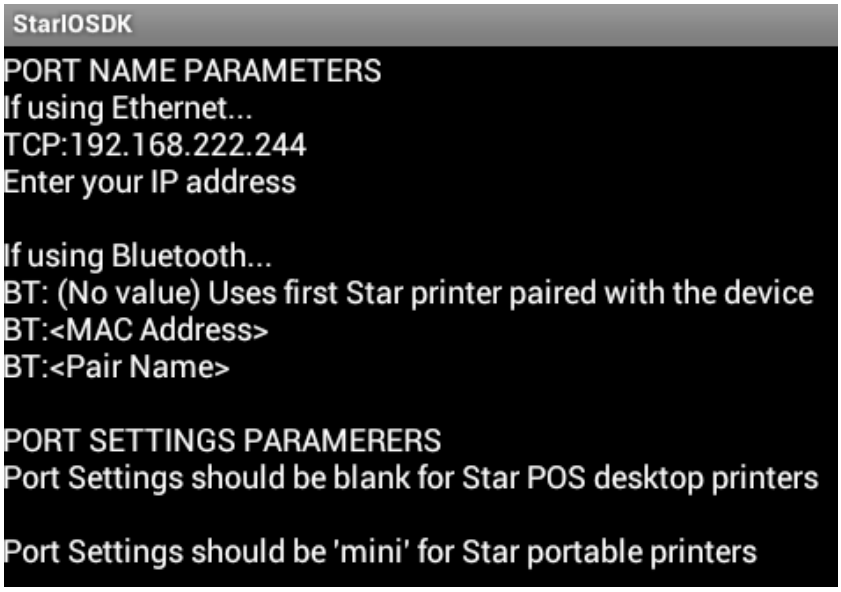
---

get StarIO Version



This method gets a StarIO Version of the printer.

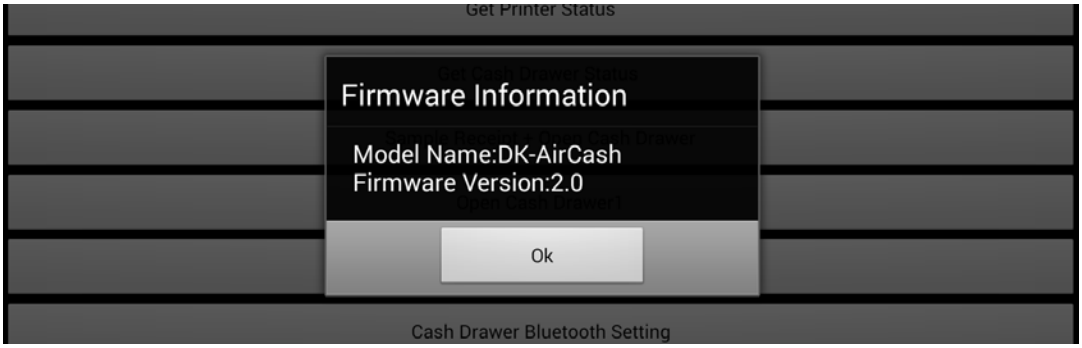
Help



Displays rules for manually entering the printer's port data into the "PortName" field. Manually entering port data is unnecessary if Port Discovery is used.

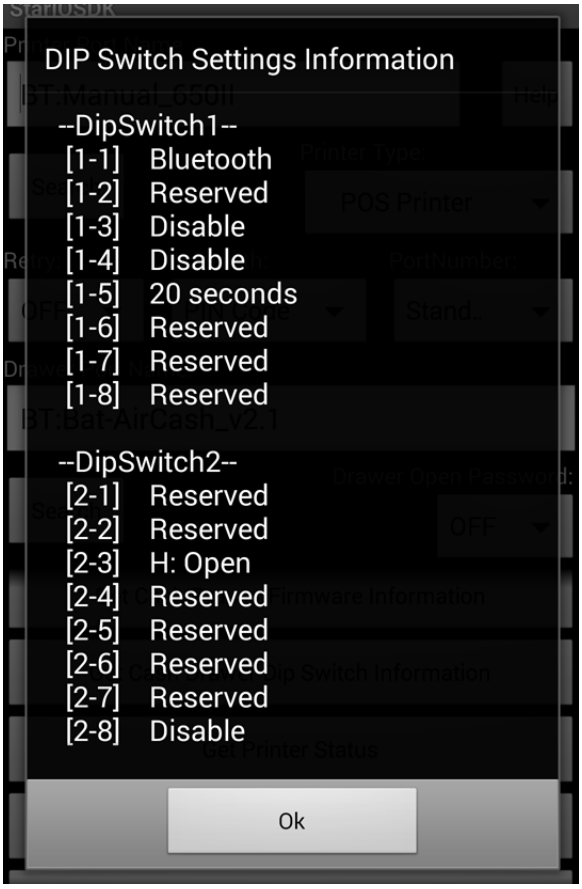
Get Printer Firmware Information /

Get DK-AirCash Firmware Information

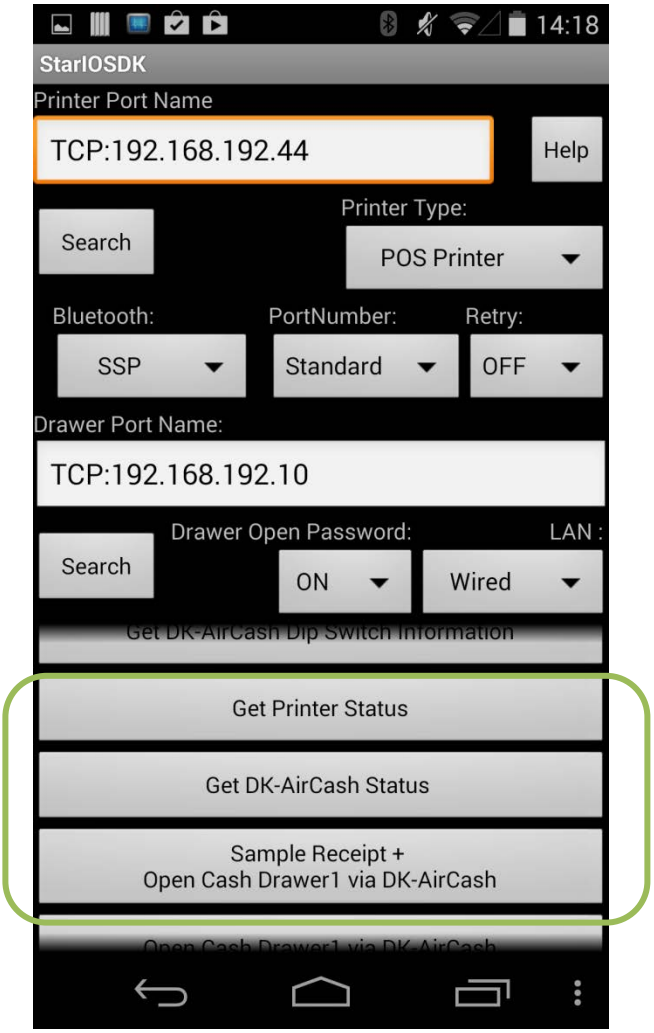


Displays firmware information of the Printer or DK-AirCash specified by Port Name.

Get DK-AirCash Dip Switch Information



Displays the DIP switch setting information of the device specified in the Drawer Port Name.



Get Printer Status /Get DK-AirCash Status

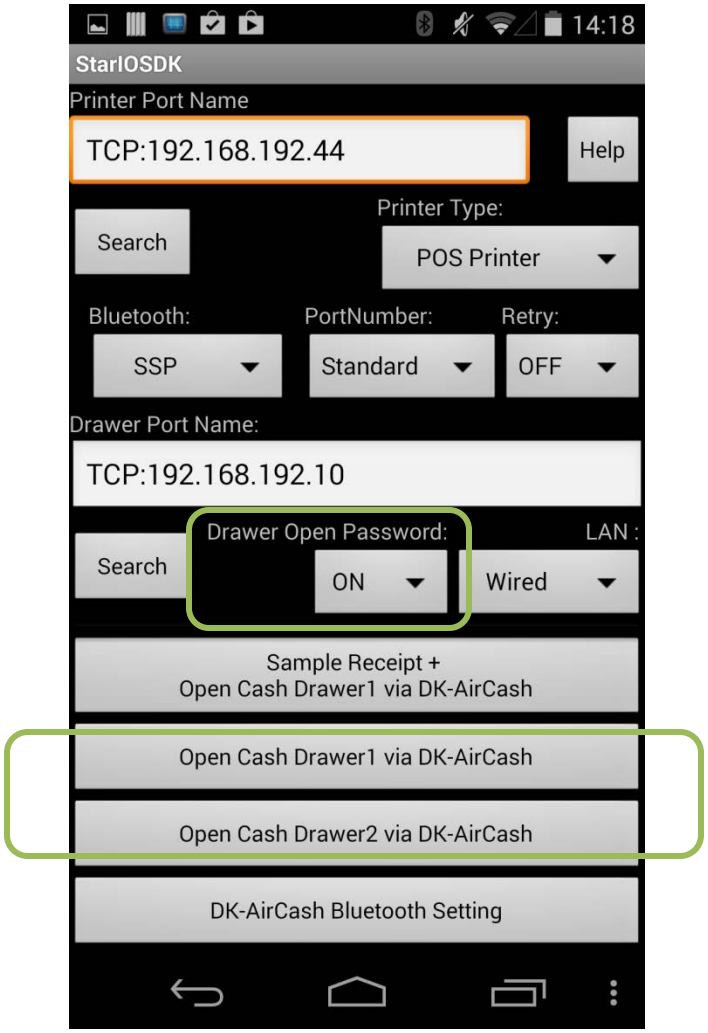
The status of the connected printer or cash drawer is dsplayed.

Sample Receipt + Open Cash Drawer1 via DK-AirCash

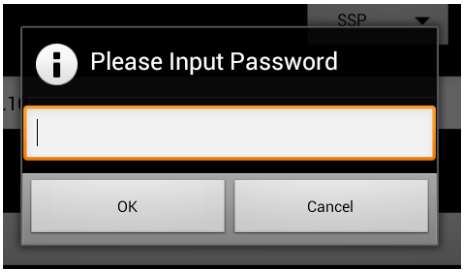
When a printer is connected, it prints a sample receipt and a cash drawer is opened if connected. When Drawer Open Password is set to ON, a password is required to open the cash drawer. The default password is "1234"

When a printer is not connected, a cash drawer does not work.

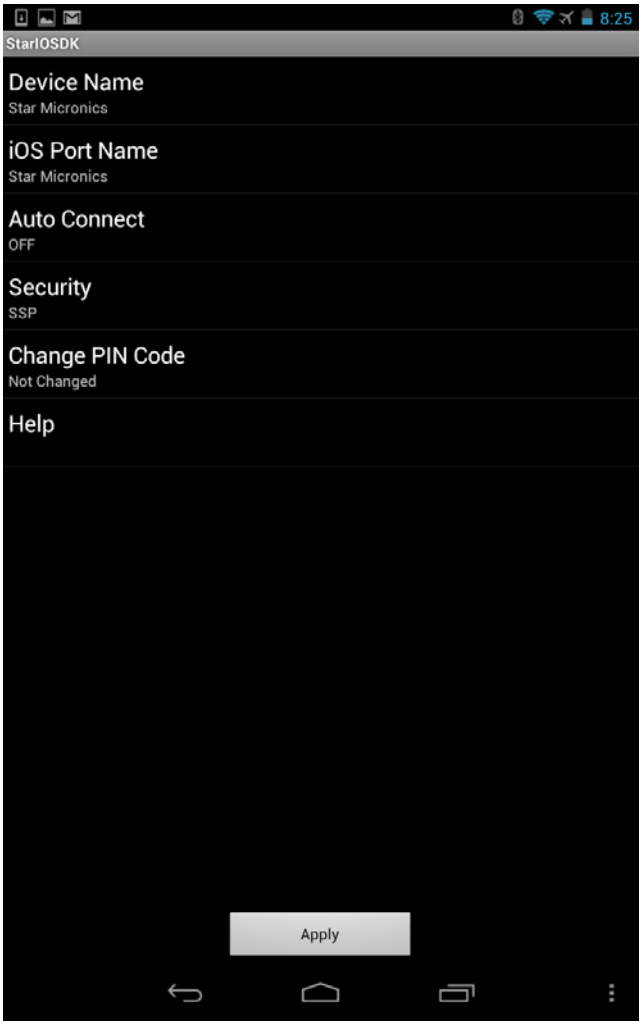
Open Cash Drawer1 via DK-AirCash / Open Cash Drawer2 via DK-AirCash



A cash drawer is opened if connected. When Drawer Open Password is set to ON, a password is required to open the cash drawer. The default password is "1234".



DK-AirCash Bluetooth Setting



Connects to the Bluetooth device which is specified for PortName and changes various settings of the Bluetooth interface.

The values applied with this method are effective after turning the device off and on and paring again.

Tips for App Development when using StarIO

Star Micronics prides itself as the industry leader in great POS products and with great power comes great responsibility. Below is a tips section just to help you get on the fast track to software development with StarIO.

TIP #1: If you are going to be coding a large project, create a class to abstract all the printing methods into class(s) instead of having the code reside in the main code block. This will help with code reusability and will also save you time in the long run from having to find one line of code in the main code. By having StarIO only reside in the class(s), you will be fully taking advantage of object oriented programming.

TIP #2: Know what the differences and definitions of (ASCII & Unicode), (Hex & Decimal), and (Byte & Char) are. A byte is normally 8-bits long which would be 8 digits of binary (1s and 0s). These bytes are just 8 bits of binary data but bytes can also be int or char. The three different variable types basically hold the data in the same way but there are slight differences. Try to code with Bytes instead of Chars, ints, or strings when choosing a variable to contain your print job data. ASCII to Unicode and vice versa conversions are sometimes unsecure so make sure you know what and how the encoding class works with these. Big mistakes made in Unicode are culture-sensitive search and casing, surrogate pairs, combining characters, and normalization.

TIP #3: HEX DUMP MODE! If you are debugging and your application seems to have a bug in it use hex dump mode on the printer. This is the best way to verify what is being sent out of the computer is being received correctly. To put the printer in hex dump mode, turn the printer off, open the cover to the paper, hold the feed button down, turn the printer back on, close the cover, let go of the feed button. Hex dump mode is a sure fire way to verify hex data is sent correctly. When in hex dump mode, printer functions will not work.

TIP #4: Do not waste time trying to reverse engineer StarIO command codes. All the available StarIO commands are available in the Thermal Line Mode Spec Manual and that is the best resource to use when researching a specific StarIO command. This SDK & Manual was built to help you (The Developer) have a very easy job ahead of you to program for Star Printers.

TIP #5: If there is a command that is not covered in this SDK but you wish to see a code snippet of that command in use then visit our Developers section for a possible code block that matches your needs.

TIP #6: Looking for an iOS printing SDK? Visit our Developers section to get access to Star developer tools for these environments.

Additional Resources

This section will share resources that will help you develop good software with StarIO.

Please get the programmers manual for Star Portable Printers from the link below.

Star Micronics Developers Network

Browse Star Micronics' FAQs, ask a question, look up information, etc.

The Developers Network gets you access to:

- Updated Versions of this Manual and Source Code
- Getting Started Advice and Industry Information
- Star Micronics Printer Drivers
- Technical Questions/Support

Android Developers Resource

The official Android development resource.

System Requirements for Android Development

See requirements for Windows, Linux, and Mac.

Unicode.org

The Unicode Consortium – Good place to learn more about Unicode.

1D Barcodes

Barcode Island is a great resource for specs on 1D barcodes.

2D Barcodes

Great place for information on 2D Barcodes, QR Codes, and PDF417

Code Pages

Learn about Code Pages here.

ASCII Table Resource

ASCII	Hex	Symbol
0	0	NUL
1	1	SOH
2	2	STX
3	3	ETX
4	4	EOT
5	5	ENQ
6	6	ACK
7	7	BEL
8	8	BS
9	9	TAB
10	A	LF
11	B	VT
12	C	FF
13	D	CR
14	E	SO
15	F	SI

ASCII	Hex	Symbol
16	10	DLE
17	11	DC1
18	12	DC2
19	13	DC3
20	14	DC4
21	15	NAK
22	16	SYN
23	17	ETB
24	18	CAN
25	19	EM
26	1A	SUB
27	1B	ESC
28	1C	FS
29	1D	GS
30	1E	RS
31	1F	US

ASCII	Hex	Symbol
32	20	(space)
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/

ASCII	Hex	Symbol
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

ASCII	Hex	Symbol
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O

ASCII	Hex	Symbol
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_

ASCII	Hex	Symbol
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o

ASCII	Hex	Symbol
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	□

Use this to compare hex values to symbol (ASCII) values.

SDK Package Version History

Release Date	SDK Package Version	Update
Apr. 20 2015	3.14.0	- Error correction
Mar. 18 2015	3.13.1	- Update PortDiscovery
Oct. 20 2014	3.13.0	- Added Wireless LAN support
Jul. 14 2014	3.12.0	- Added getStarIO Version method - Added getDipSwitchInformation method
Mar. 17 2014	3.10.0	-Added StarBluetoothManager class -Added getFirmwareInformation method -Added setEndCheckedBlockTimeoutMillis method -Added Open Cash Drawer2 function
Dec. 24 2013	3.8.0	- Added notifications
Jul. 3 2013	3.7.0	- Initial Release

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/