Android Software Development Kit

How to Use StarIO for ProxiPRNT

This SDK contains a Java Eclipse project for Android devices.

Supported Versions	Supported Star Interfaces	Compatible StarIO JAR File
Android OS 4.3 and Higher	Bluetooth, Wired LAN, Wireless LAN	StarIOPort3.1.jar (Default)

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

Supported Device:

- BLED10-U(dedicated hardware: Bluetooth Low Energy USB Dongle)
- Tablet devices with Android 4.3 or later and Bluetooth 4.0[*1]

[*1] Some of smart devices do not work with ProxiPRNT. Please check that your device works with ProxiPRNT before use.

Nexus7 (2013 Asus) does not work properly with ProxiPRNT. Please do NOT use it with ProxiPRNT.

Table of Contents

- About this Manual
- Star POS Deivce Compatibility Chart
- Connecting a Star POS Device to an Android Device
- Configuring an Android Device for Development
- How to Import and Run the Android SDK
- Using the SDK with Star POS Printers
- Overview of how this Android SDK is designed
- StarIO - (StarIOPort3.1.jar)
- 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 POS Devices. 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.

Star POS Device Compatibility Chart

This table shows the compatibility of the ProxiPRNT with Star POS devices.

◆Compatibility List of Android OS Version and print modes

Star POS Device		Android OS Version 4.3 later	Line	Raster
Model	Interface			
TSP100ECO	USB*	✓		✓
TSP100U	USB*	✓		✓
TSP100GT	USB*	✓		✓
TSP100LAN	Wired LAN	✓		✓
TSP650	USB*	✓	✓	✓
	Wired LAN	✓	✓	✓
TSP650II	USB*	✓	✓	✓
	Wired LAN	✓	✓	✓
	Bluetooth	✓	✓	✓
FVP10	USB*	✓	✓	✓
	Wired LAN	✓	✓	✓
TSP700II	USB*	✓	✓	✓
	Wired LAN	✓	✓	✓
TSP800II	USB*	✓	✓	✓
	Wired LAN	✓	✓	✓
TUP500	USB*	✓	✓	✓
	Wired LAN	✓	✓	✓
TUP900	USB*	✓	✓	✓
DK-AirCash	Wired LAN	✓		
	Wireless LAN			
	Bluetooth	✓		

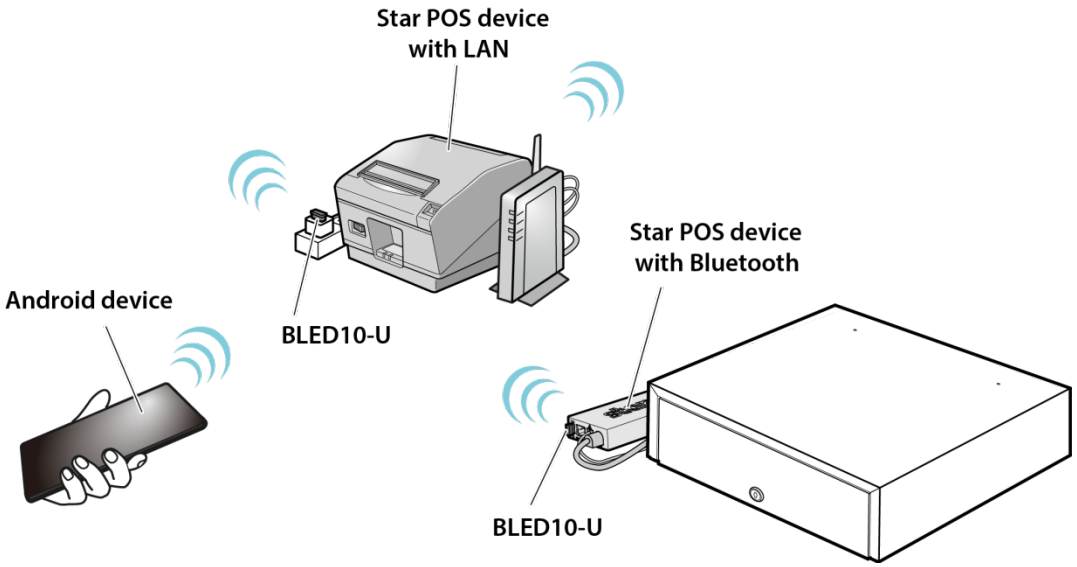
*To use a USB interface device, Apple AirMac series are required.

Note 1: The TSP100 Series does not support the 1D Barcodes, 2D Barcodes, Cut Patterns, and Text Formatting samples in this application. These samples were written using Star Line Mode, while the TSP100 natively understands Raster Mode. Kindly reference Star Micronics' Raster Mode commands in the programming manual for the Cut and Text Formatting commands. It is recommended to print barcodes as graphics for this model.

Note 2: This SDK offers the most popular features, but not all printer functionality has been included (such as Presenter Options for the TUP500). The commands not included in this SDK application are available in the Line Mode Programming Manual.

Connecting a Star Device to an Android Device

First, connect the Star POS device (a Star POS printer or the DK-AirCash) to the Android device.



◆Network Interface

Assign an IP Address to the Star Printer and connect it to the network. Standard Star POS Devices do not ship with an IP Address pre-assigned; this can set by a DHCP network.

Use Star POS Devices 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.

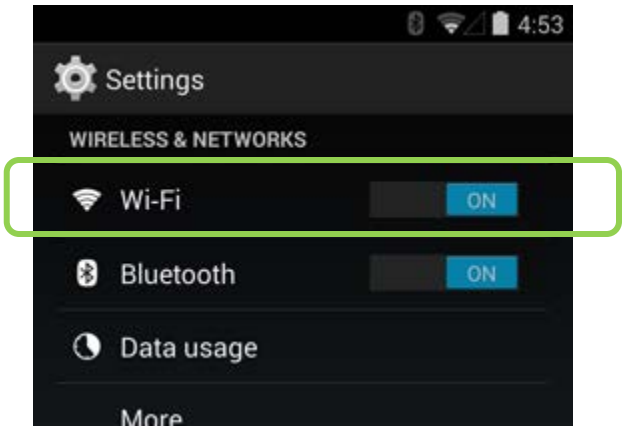
Refer to "guidelines-ethernet_en.pdf" for how to confirm and change the #9100 Multi Session setting and how to set the Static IP Address.

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 assigned the IP Address. (IP Address: 0.0.0.0)

1. Assign an IP Address to the Star POS Devices and connect it to the network.

2. Tap Settings.

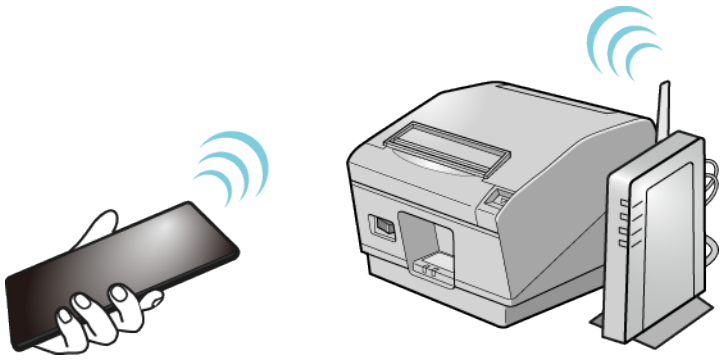


3. Ensure Wi-Fi is ON.

4. Connect to the same network the Star POS Device 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.



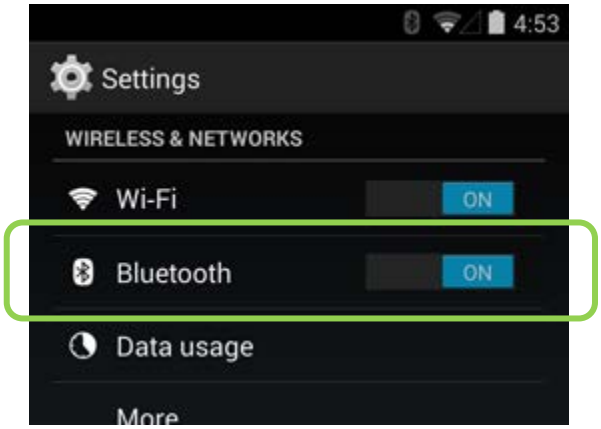
◆Bluetooth Interface

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.

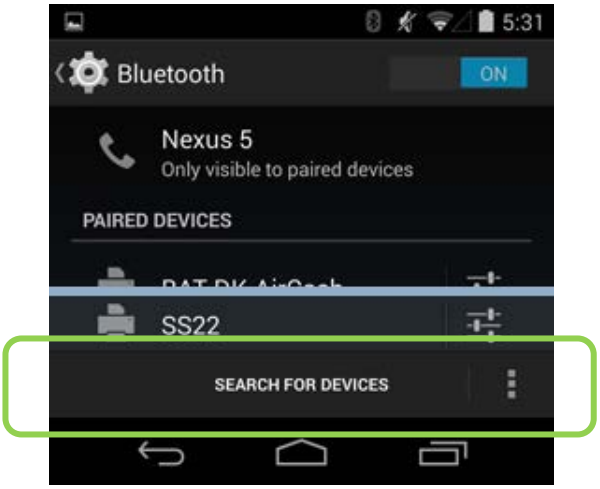
You can change the LAN/Bluetooth settings of Star POS devices such as a device name setting by using the Star Setting Utility provided by Star Micronics. Please download it from Google play.

- Pairing of a Star POS device with your Android Device

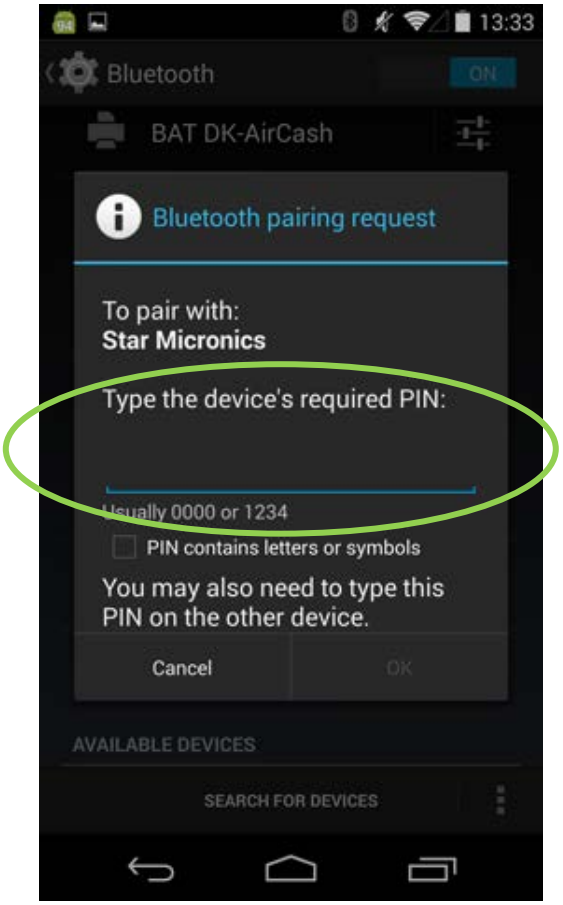
1. Ensure Bluetooth is enabled and the Star POS Device is powered on.
   When the security setting of the Star POS device is set to SSP, press the PAIR button for more than 5 seconds to make it available for pairing.
2. Tap Settings.



3. Tap Bluetooth to ON.



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.)

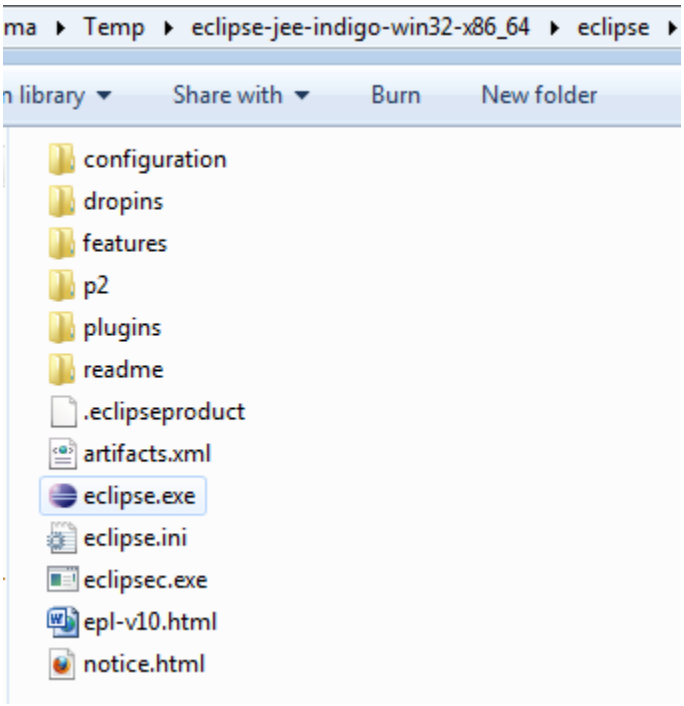
Importing and Running the StarIO Android SDK

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.

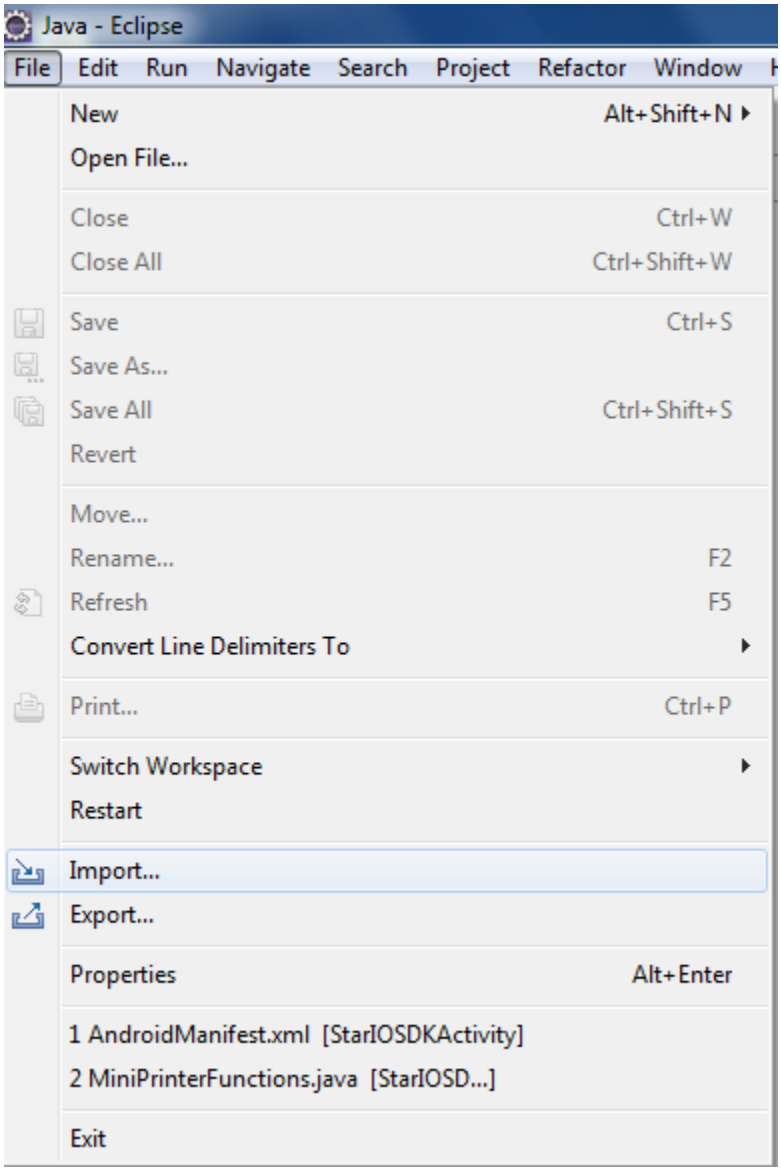
This section will explain:

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

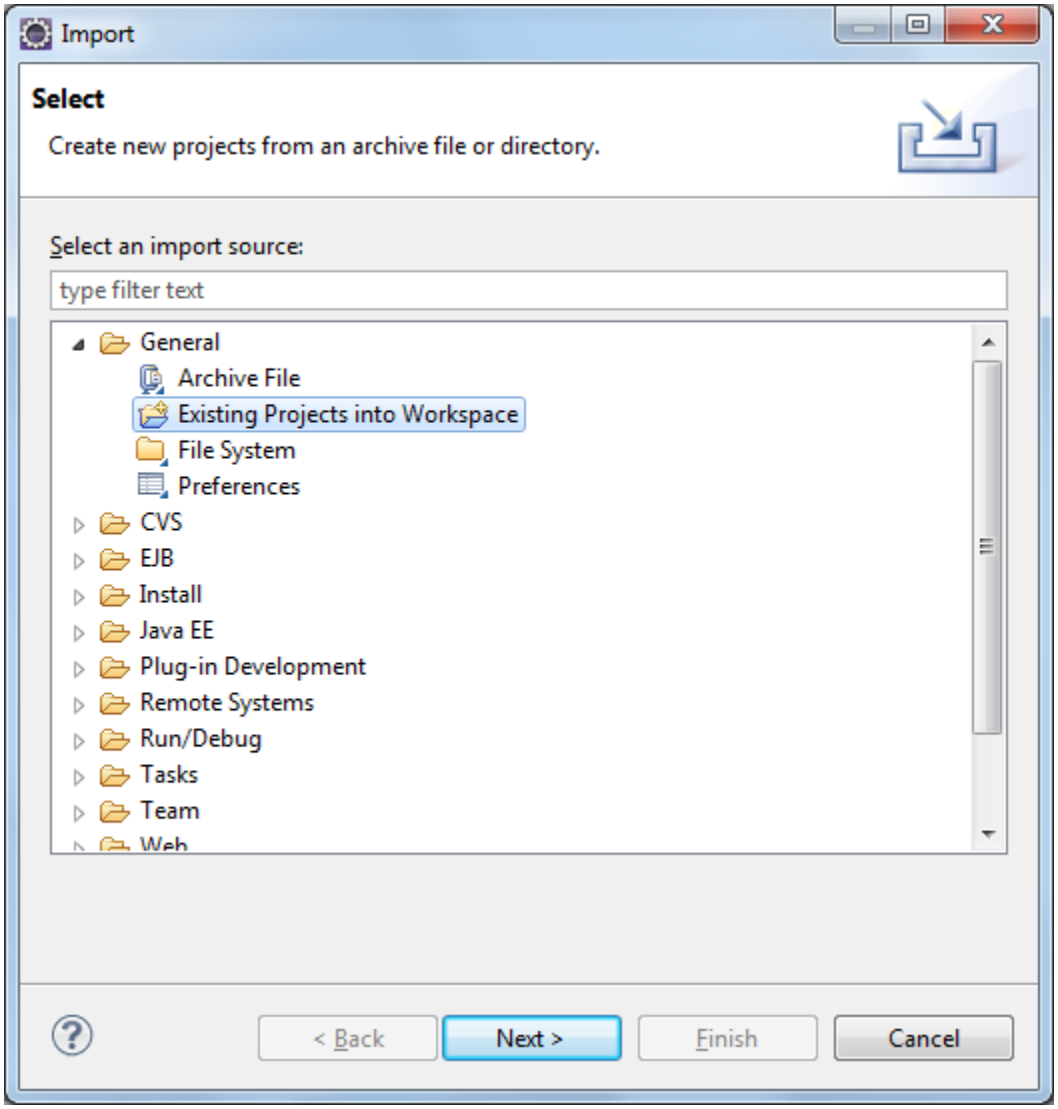
How to import the Android SDK project into Eclipse:



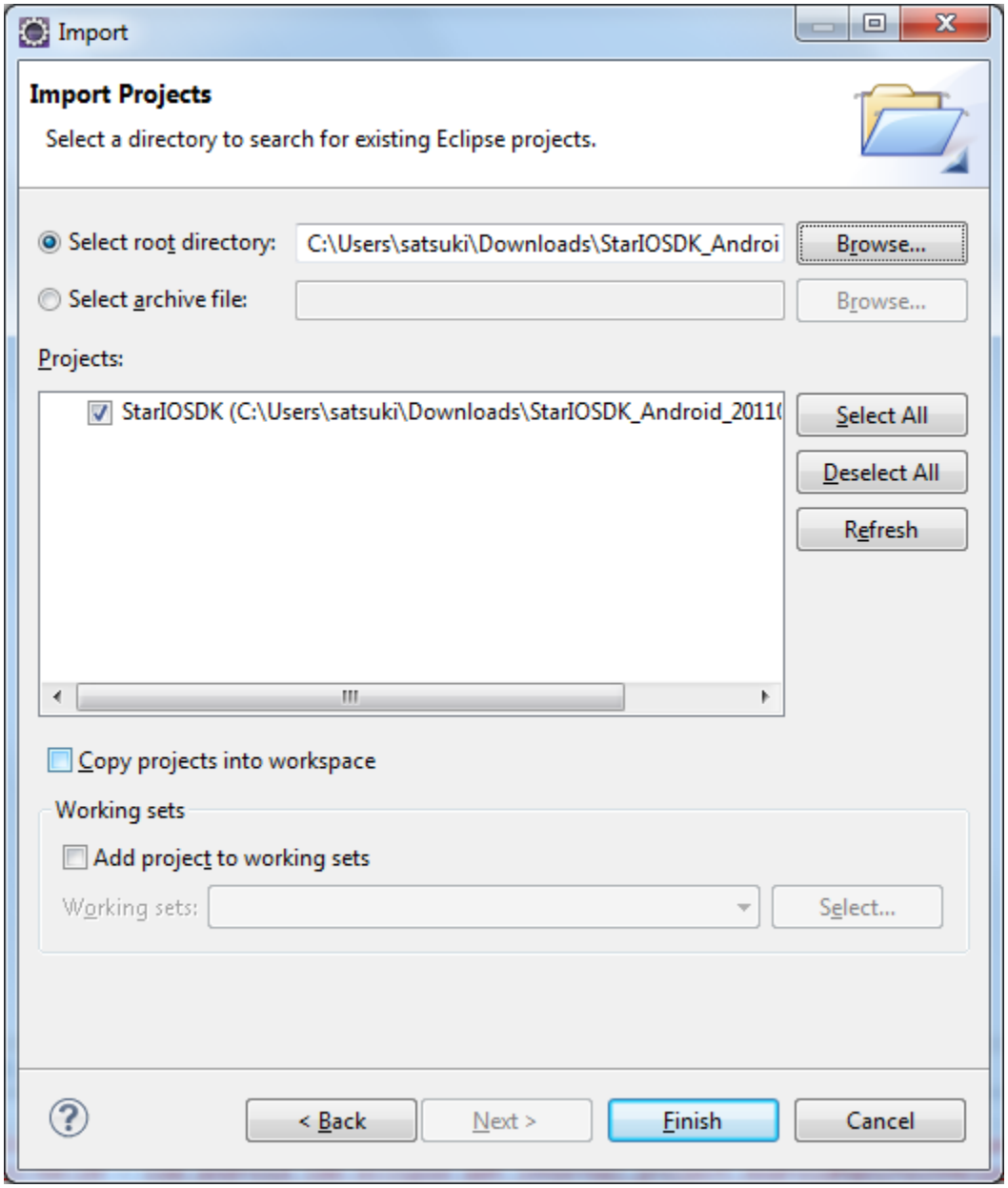
1. Execute Eclipse.



2. File > Import

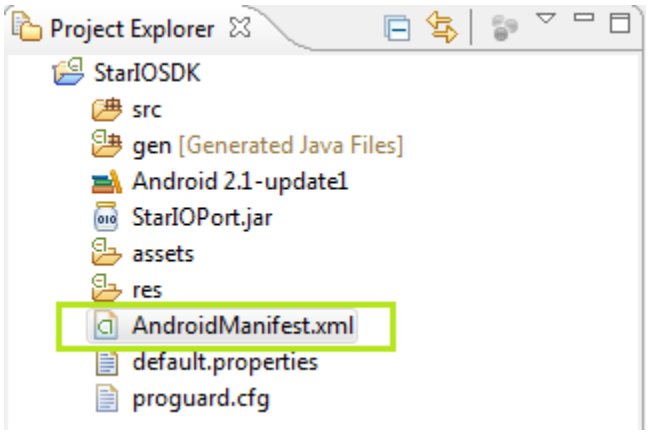


3. General > Existing Projects into Workspace

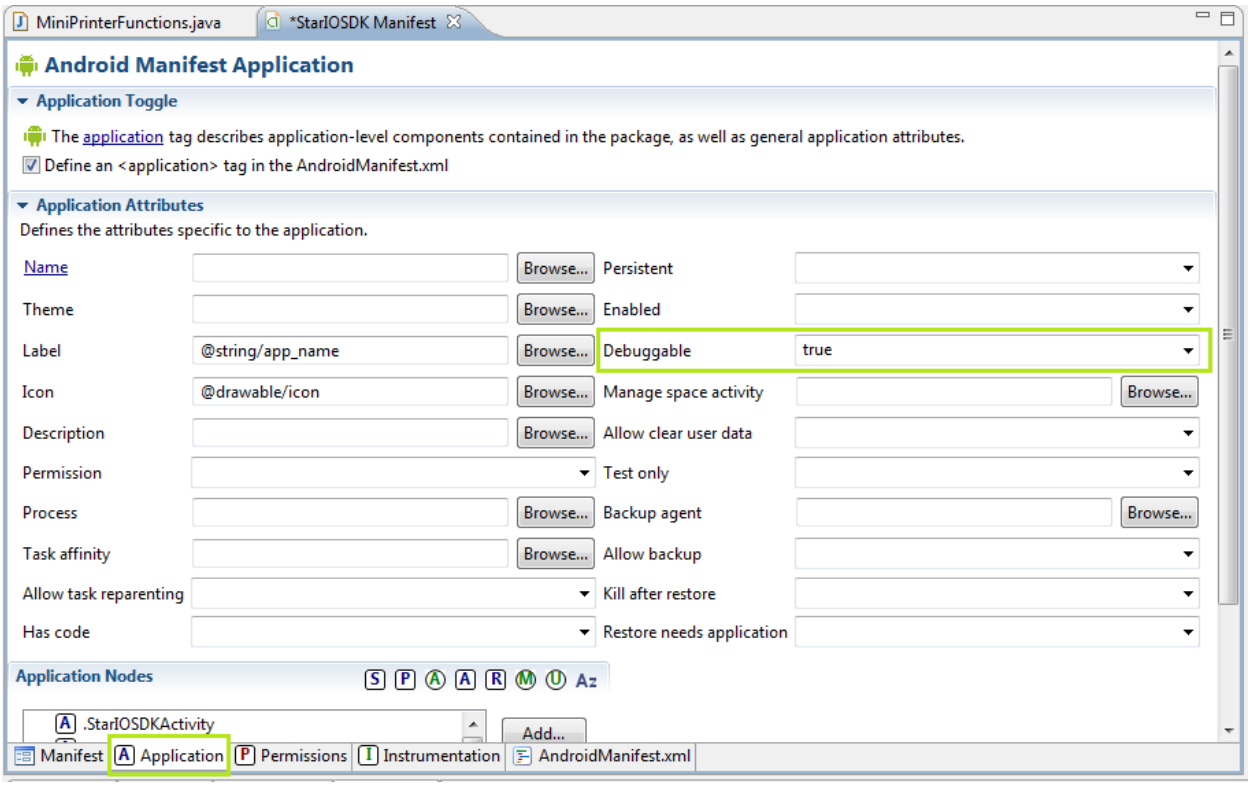


4. Browse to the location where the Star SDK is saved. Click Finish.

Enabling debug mode in Eclipse:



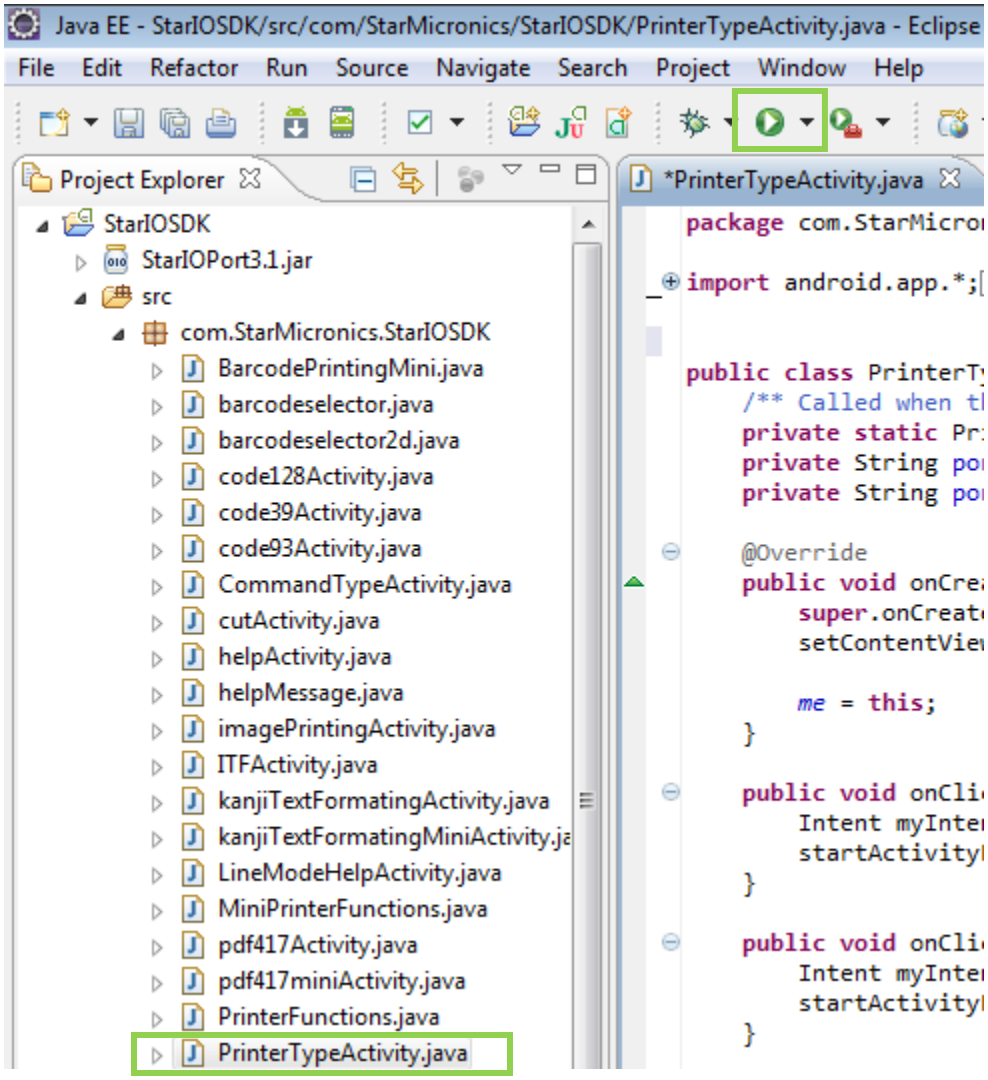
1. Click AndroidManifest.xml.



2. Click the Application tab.

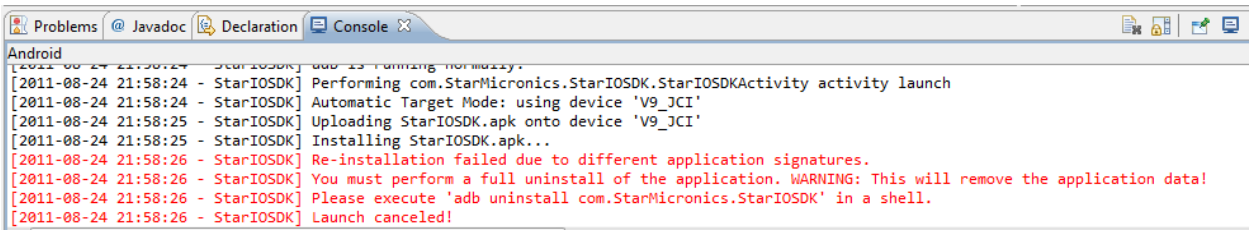
3. Set Debuggable to true.

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

About the BLED10-U

The BLED10-U uses radio waves and the radio intensity varies depending on:

- the actual radio wave environment
- an installation place of the BLED10-U
- a direction of the Android Device
- an angle of the Android Device
- a user's handling way of the Android device, and so on.

When creating your App, design the threshold value setting to be conducted by holding your Android device in the same direction as the actual use, and do not allow the auto rotation of the screen.

When you specify the threshold value for the BLED10-U, use the actual device in the actual operation environment in conformity with the use conditions of the Android device. If you can not obtain the stable operative condition at the actual operation, the operating conditions for setting the threshold value such like a direction and an angle of the Android device may not be reproduced.

Before starting the actual field operation, please check sufficiently that the device does not work in the uninteded environment and adjust the threshold.



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 Star POS Devices. 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

(*If use the “PIN Code” by Bluetooth Interface, need the Port Settings. No Port Settings are required when using other Interface and “SPP” by Bluetooth Interface.)

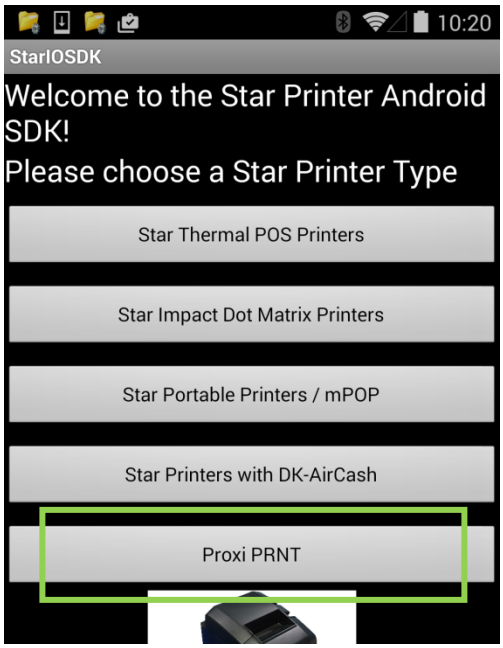
Notification:
In case of using Bluetooth interface with PIN code setting, when you execute getPort method, some terminals(*) require you to input PIN code again.
Even though you input it, the application fails getPort method and cannot print or cannot get status.
In this case, please try the following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

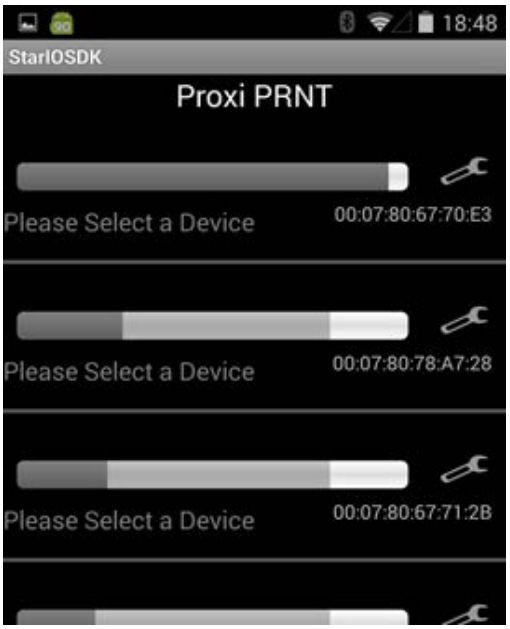
1. Tap “ProxiPRNT”.



·Connecting the BLED10-U with the Star POS device

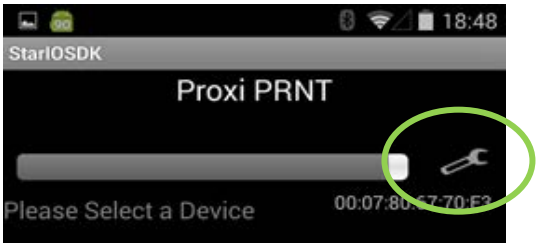
1. Connect the BLED10-U to the USB connector (TYPE-A) to supply power.

Note:When using multiple BLED10-Us, you can identify the device by moving your Android device close to the BLED10-U and checking that the RSSI level has increased.

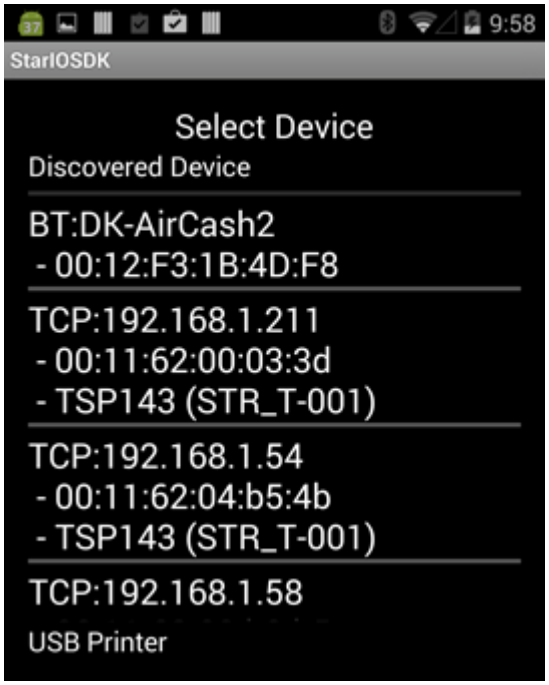




* The RSSI bar displays the signal strength in dark grey before the connection.

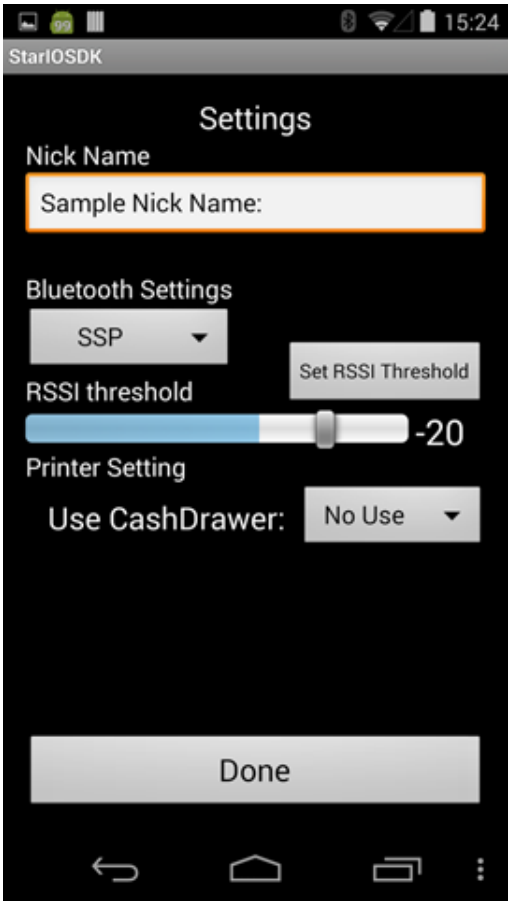


2. The RSSI bar of the BLED10-U is displayed. Tap  to go to the setting window.



3. All the Star Ethernet devices available for your Android device and all the Star Bluetooth devices paired with your Android device are displayed. Select the Star POS device to connect with the BLED10-U to be configured and tap it.

4. Configure the setting for the selected Star POS device. The setting contents are different for every Star POS device. After completing the configuration of the necessary settings, tap “Done”.



Nick Name

Enter the Nick name to identify the connection of the BLED10-U and the Star POS device.

Bluetooth Settings（Bluetooth device only）

Select the security setting used to be paired with the Star POS device to be connected.

RSSI threshold

Set the RSSI threshold.
Set the device in the actual operation environment and tap [Set RSSI Threshold] button.
Check the RSSI in more than one place and adjust the threshold value. See here for details.

Use CashDrawer

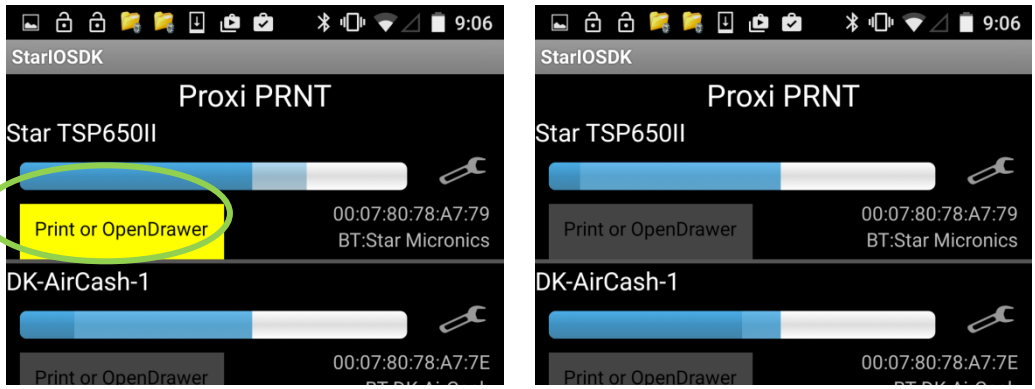
Select this item when using a cash drawer.
When using the DK-AirCash (Bluetooth model), select “Use (DK-AirCash)”.

* If “Done” button is hidden and the soft keyboard is displayed, hide the soft keyboad to and display “Done” button.

5. The threshold set to the RSSI bar is shown in blue and the actual signal strength is shown in light blue. If the signal strength is less than the threshold, it is displayed in deep blue.

   If the RSSI exceeds the threshold, the “Print or OpenDrawer” button will be displayed and it can be operated.

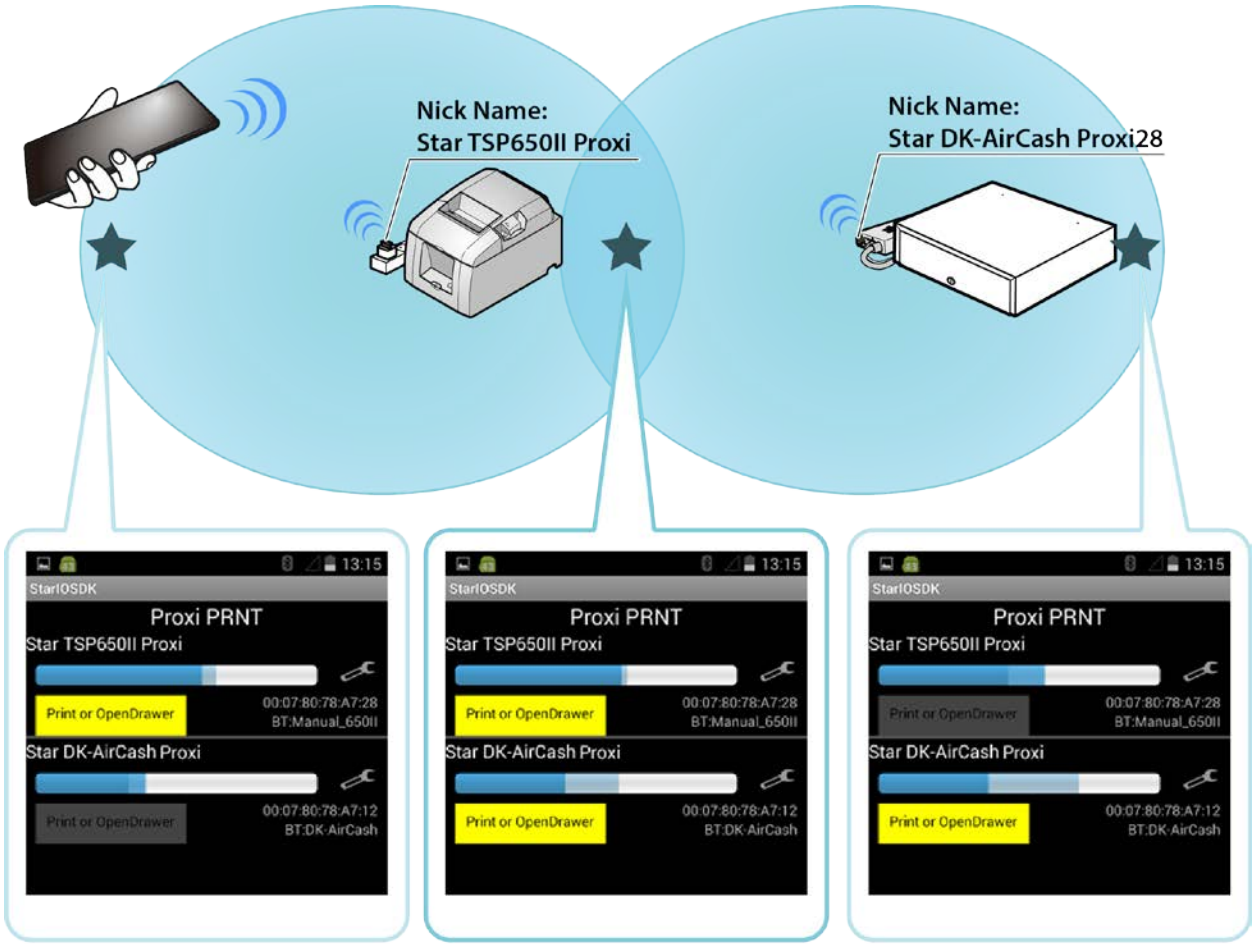
   Check that the device can not work in the uninteded environment (as shown in the right screen).



   Tap the “Print or OpenDrawer” button to perform the test print and open the drawer (if selected).

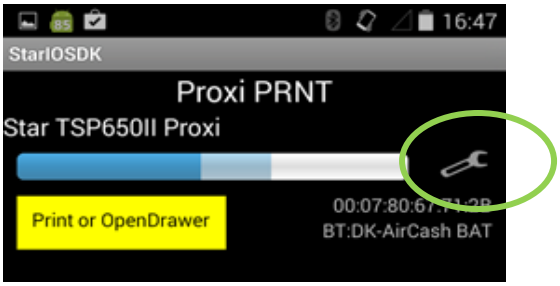
   * The raster mode is used for the test print in this SDK.

Example) The RSSI level at each use position (a star mark) when using multiple BLED10-Us



Delete/edit the connection of the BLED10-U and the Star POS device

1. Tap  shown on the right of the RSSI bar to edit the settings of the connection.



When the Bluetooth device name or the IP address of the connected device is changed after connection has been configured, be sure to change the Port Name of the connection information.

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 devices.

Overview of How This Android SDK is Designed

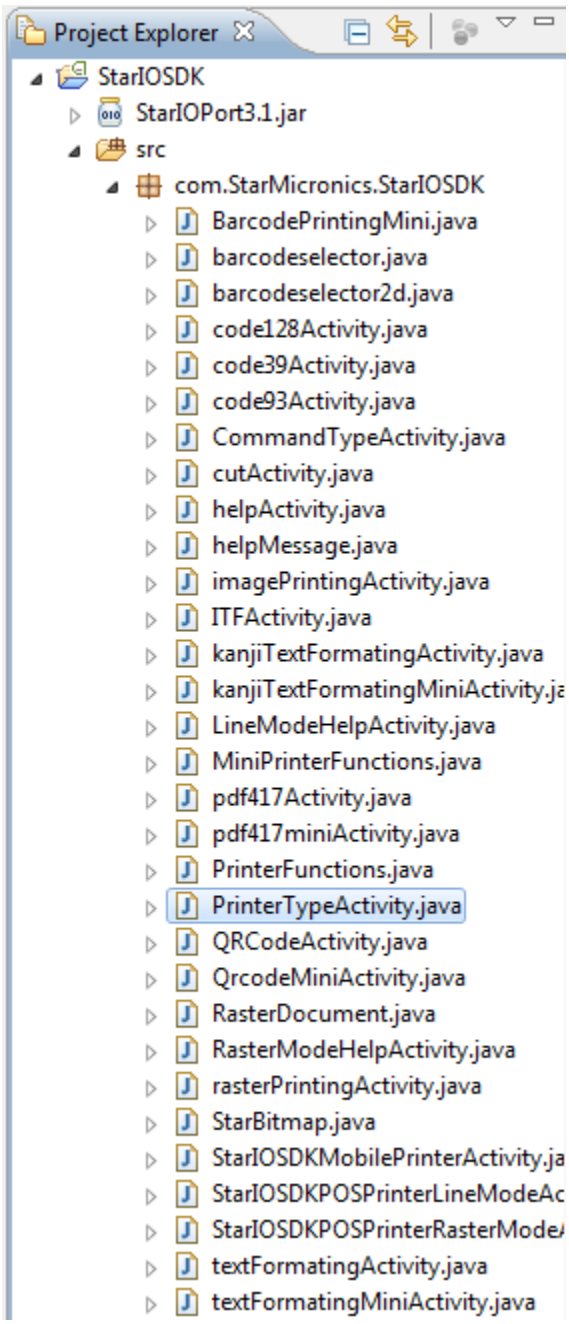
This overview will touch briefly on key components of the SDK.

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 Star POS Devices.

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 Star POS Devices. 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:

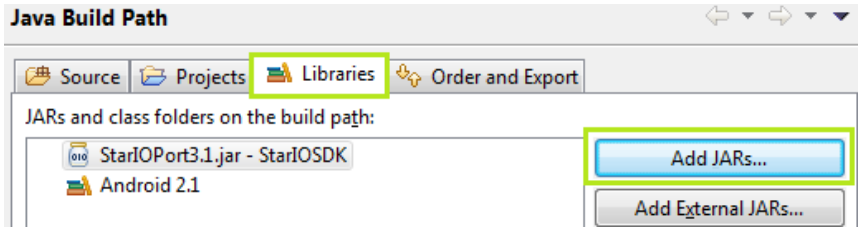
The StarIO SDK contains two library files. Use the following libraries when developing an application using ProxiPRNT.

Supported Versions	Supported Star Interfaces	Compatible StarIO JAR File
Android OS 4.3 and Higher	Bluetooth, Wired LAN / Wireless LAN	StarIOPort3.1.jar (Default)

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:

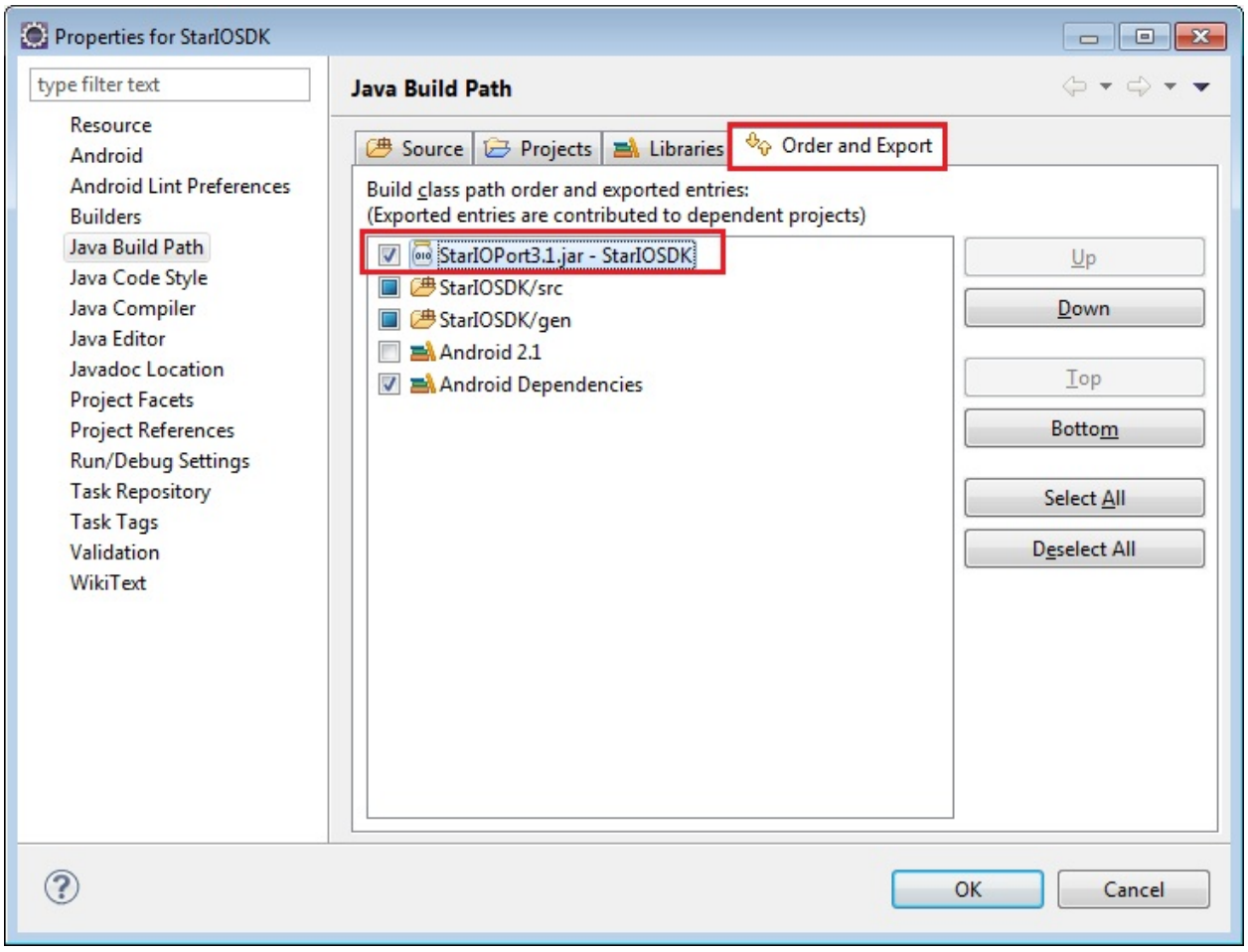
```
import com.starmicronics.stario.StarIOPort;
import com.starmicronics.stario.StarIOPortException;
import com.starmicronics.stario.StarPrinterStatus;
import com.starmicronics.stario.StarProxiPRNTManager;
import com.starmicronics.stario.StarProxiPRNTManagerCallback;
```

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 V4.3 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 4.3 and higher: Supports Bluetooth and Ethernet 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 Star POS Devices.

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)

[Use share printer function with Apple AirPort Express]
Set AirPort Express IP Address for **portName**.

Ex. TCP:192.168.1.2

Set port number for **portSettings**.

Increase the port number in sequential order from 9100 to 9109 until communication is successful.

Ex. 9100

[Use the "PIN Code", when pairing Bluetooth Device]

Set ";p" for **portSettings**.

* For details, refer to here.

● 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 Star POS Devices. Using one of the valid inputs for portName and portSettings, you can connect to the Star POS Devices.

```
//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.

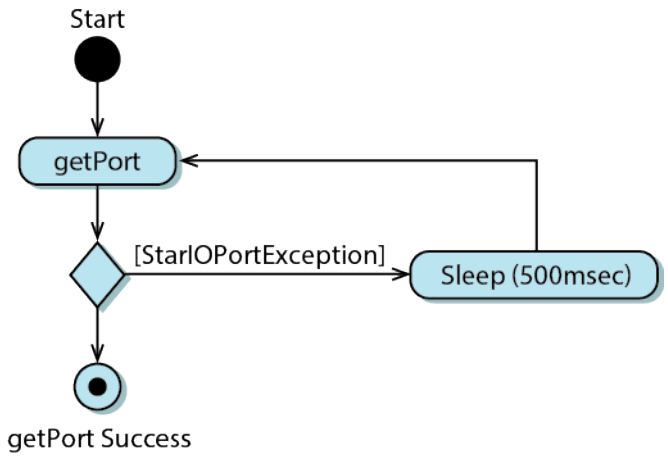


Always use a try, catch when using **getPort**. If the port cannot be opened because of connection problems, your program will crash unless you use a try, catch like the above example.

In case your application sends data to bluetooth Star POS Devices, 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 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 Star POS Devices 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 Star POS Devices 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 Ethernet 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 Star POS Devices. 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 Star POS Devices and sets setEndCheckedBlockTimeoutMillis to the proper value automatically.

endCheckedBlock

public StarPrinterStatus **endCheckdBlock ()** throws StarIOPortException

This method monitors Star POS Devices status and when the transferred data is processed completely by Star POS Devices, 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.

StarProxiPRNTManager Class:

StarProxiPRNTManager Class utilizes the RSSI broadcast by the dongle (BLED10-U) which supports Bluetooth Low Energy to use a POS printer or a DK-AirCash placed nearby.

● Method

getSharedManager

```
public static StarProxiPRNTManager getSharedManager()
```

Return:

Gets an instance of StarProxiPRNTManager

Throws:

StarIOPortException	- when the Android OS version is lower than Version 4.3 (-API level 18)

addSettingForDKAirCashPortName

```
public void  addSettingForDKAirCashPortName(String portName, String portSettings,
                                            String macAddress, int threshold, String nickName)
```

Adds the settings of the DK-AirCash to be connected with the BLED10-U.

Parameters:

portName	– Identifies the printer to use
portSettings	– The settings for the port
macAddress	– Mac address of the BLED10-U
threshold	– Threshold
nickName	– Nick name to connect the BLED-10U and the DK-AirCash

Throws:

StarIOPortException

- when portName or macAddress is null
- when the same portName has been specified
- when the same macAddress has been specified

addSettingForPrinterPortName

```
public void addSettingForPrinterPortName(String portName, String portSettings,
                    boolean withDrawer, String macAddress, int threshold, String nickName)
```

Adds the settings of the POS Printer to be connected with the BLED10-U.

Parameters:

portName– Identifies the printer to use
portSettings – The settings for the port
withDrawer – Presence/absence of a cash drawer
macAddress – Mac address of the BLED10-U
threshold – Threshold
nickName – Nick name to connect the BLED-10U and the POS Printer

Throws:

StarIOPortException

- when portName or macAddress is null
- when the same portName has been specified
- when the same macAddress has been specified

getClosestCashDrawer

```
public String getClosestCashDrawer()
```

When the RSSI value exceeds the threshold, gets the port name of the DK-AirCash.
When RSSI value is lower than the threshold, the port name is an empty string.

Return:

portName

getClosestPrinter

public String **getClosestPrinter**()

When the RSSI value exceeds the threshold, gets the port name of the POS Printer.
When RSSI value is lower than the threshold, the port name is an empty string.

Return:
portName

getRSSI

public int **getRSSI**(String macAddress)

Gets the RSSI value of the connected printer.

Parameters:

macAddress	- Mac address of the BLED10-U

Return:
RSSI value

Throws:

StarIOPortException	– when the Android OS version is lower than Version 4.3 (API level 18)

getSettings

```
public HashMap<String, Object>  getSettings(String portName)
```

This method gets the setting information connected to the BLED10-U by the port name. The acquired result is returned by HashMap<String, Object>.

Return:

- Gets the device type (DK-AirCash or Printer) in the string style by setting the key to StarProxiPRNTManager.KEY_STAR_DEVICETYPE.
- Gets the presence/absence information (true/false) in the boolean style by setting the key to StarProxiPRNTManager.KEY_STAR_WITHDRAWER.
- Gets the NicName to connect the BLED10-U and the DK-AirCash in the string style by setting the key to StarProxiPRNTManager.KEY_STAR_NICKNAME.
- Gets the portSettings in the string style by setting the key to StarProxiPRNTManager.KEY_STAR_PORTSETTINGS.
- Gets the threshold in the int style by setting the key to StarProxiPRNTManager.KEY_STAR_THRESHOLDRSSI.
- Gets the Mac address in the String style by setting the key to StarProxiPRNTManager.KEY_STAR_MACADDRESS.

getAllSettings

```
public HashMap<String, HashMap<String, Object>>  getAllSettings()
```

This method gets all setting information connected to the BLED10-U by the port name. The acquired result is returned by HashMap<String, Object>.

Return:

- Gets all setting information connected with each BLED10-U by the port name.

removeSettingsWithPortName

public void **removeSettingsWithPortName**(String portName)

This method removes the setting information connected to the BLED10-U by the port name.

deserializeSetting

public boolean **deserializeSetting**(Byte[] settingsByteArray)

Restores the setting information connected with the BLED10-U and sets them.

Returns:

true	- when the restoration is succeeded
false	- when the restoration is failed

serializedSettings

public byte[] **serializedSettings**()

This method acquires the byte array of the setting connected with the BLED10-U.

setCallback

public void **setCallback**(StarProxiPRNTManagerCallback callback)

This method sets a callback. See StarProxiPRNTManagerCallback interface for the callback function.

Parameters:

callback	– specifies the object (StarProxiPRNTManagerCallback interface).

getCallback

public StarProxiPRNTManagerCallback **getCallback**()

This method acquires the StarProxiPRNTManagerCallback object.
Use it after calling the setCallback method.

Return:

-StarProxiPRNTManagerCallback object

startScan

```
public boolean startScan()
```

This method starts a scan of the Bluetooth Low Energy device.

Returns:

true	- when a scan is operated correctly
false	- when a scan is not operated correctly

Throws:

StarIOPortException	– when the Android OS version is lower than Version 4.3 (API level 18)



Do not execute this method when printing data or opening the drawer. Execute stopScan method and stop scanning BLE device before printing data or opening the drawer.

startScan

```
public boolean startScan(int samplingNumber)
```

This method starts a scan of the Bluetooth Low Energy device.

Parameters:

samplingNumber	- set 3, 7 or 10

Returns:

true	- when a scan is operated correctly
false	- when a scan is not operated correctly

Throws:

StarIOPortException	- when the samplingNumber is negative value or 0 – when the Android OS version is lower than Version 4.3 (API level 18)



Do not execute this method when printing data or opening the drawer. Execute stopScan method and stop scanning BLE device before printing data or opening the drawer.

stopScan

```
public void stopScan()
```

This method stops a scan of the Bluetooth Low Energy device.

Throws:

StarIOPortException – when the Android OS version is lower than Version 4.3 (API level 18)



Execute stopScan method and stop scanning BLE device before printing data or opening the drawer.

calibrateActionArea

```
public int calibrateActionArea(String bleMacAddress)
```

This method returens the maximum RSSI value approximately five seconds later.

Parameters:

bleMacAddress - Mac address of the BLED10-U

Throws:

StarIOPortException – when the Android OS version is lower than Version 4.3 (API level 18)

– When the return value is not the following format
Ex.) 00:07:80:A2:70:ab

– When StartScan has been executed already

– When BLED10-U is not found



Execute stopScan method and stop scanning BLE device before executing this method.

StarProxiPRNTManagerCallback Interface

●Method

onClosestCashDrawerFound

```
public abstract void onClosestCashDrawerFound(String portName)
```

Notifies the port name of the drawer connected with the BLED10-U when the current RSSI value exceeds the threshold during a scan of the Bluetooth Low Energy device started by the startScan method.

Parameters:

portName – Port name of the drawer connected with the BLED10-U

onClosestPrinterFound

```
public abstract void onClosestPrinterFound(String portName)
```

Notifies the port name of the printer connected with the BLED10-U when the current RSSI value exceeds the threshold during a scan of the Bluetooth Low Energy device started by the startScan method.

Parameters:

portName – Port name of the Printer connected with the BLED10-U

onStateUpdated

```
public abstract void onStateUpdated(String portName, String macAddress)
```

Notifies the Mac address of the BLED10-U and the port name of the printer or the drawer connected with the BLED10-U when the current RSSI value cannot be acquired for 5 seconds or more during a scan of the Bluetooth Low Energy device started by the startScan method.

Parameters:

portName	– Port name of the Printer or the drawer connected with the BLED10-U
macAddress	– Mac address of the BLED10-U

onPortDiscovered

```
public abstract void onPortDiscovered(String portName,
 StarProxiPRNTManagerCallback.StarDeviceType starDeviceType, String macAddress, int rssi)
```

Notifies the Bluetooth Low Energy device discovered during a device scan started by the startScan method.

Parameters:

portName	– port name of the printer or the drawer connected with the BLED10-U The port name is an empty string if no device is connected.
starDeviceType	– Type of the device which the current RSSI value is acquired for.
macAddress	– Mac address of the BLED10-U
rssi	– RSSI value

Status List of the class structure StarPrinterStatus

Member name	Contents	Type	Detail
blackMarkError	Black Mark Error	boolean	" true " : Black mark error occurs. " false " : Black mark error does not occur. When you set printer to Black mark, and print to not Black mark paper, this error occurs.
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	You can check status of Cover " true " : Cover is opened. " false " : Cover is closed.
cutterError	Auto-cutter Error	boolean	You can check status of Cutter " true " : Cutter error occurs. " false " : Cutter error does not occur.
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	You can check status of Head Thermistor. " true " : Head thermistor detects an abnormal value. " false " : Head thermistor does not detect an abnormal value.
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	" true " : Printer is stopped by head temperature. " false " : Printer is not stopped by head temperature.
presenterPaperJamError	Presenter Paper Jam Error	boolean	You can check status of Paper Jam in Presenter. " true " : Paper jam occurs in presenter . " false " : Paper jam does not occur in presenter .
presenterState	Presenter Paper Position	int	You can check status of Presenter. 0 : State where there is no paper in presenter 1 : State where paper is supplied (loop state) 3 : State where paper is discharged (Can be pulled out) 6 : State where paper is recovered 7 : State where paper is pulled out.
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	" true " : Paper end. " false " : Paper does not end.
receiptPaperNearEmptyInner	Paper Near-end (Inner Side)	boolean	" true " : Paper near-end. " false " : Paper does not near-end.
receiveBufferOverflow	Receive Buffer Overflow	boolean	You can check status of recieved Buffer. " true " : Received buffer is full. " false " : Received buffer is not full.
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	" true " : Printers detects an abnormal power supply voltage. " false " : Printers does not detect an abnormal power supply voltage.

Class structure StarPrinterStatus Supported

Member name	TSP 100 LAN	TSP 100 U	TSP 100 GT	TSP 100 IIU	FVP 10	TSP 650	TSP 650 II	TSP 700 II	TSP 800 II	TUP 500	TUP 900	DK-Air Cash
blackMarkError					✔			✔	✔	✔	✔	
compulsionSwitch	✔	✔	✔	✔	✔	✔	✔	✔	✔			✔
coverOpen	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	
cutterError	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	
etbAvailable	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
etbCounter	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
headThermistorError										✔	✔	
offline	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
overTemp	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	
presenterPaperJamError										✔	✔	
presenterState										✔	✔ After FW ver1.2	
raw	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
rawLength	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
receiptPaperEmpty	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	
receiptPaperNearEmptyInner					✔	✔	✔	✔	✔	✔	✔	
receiveBufferOverflow					✔	✔	✔	✔	✔	✔	✔	
unrecoverableError	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
voltageError										✔	✔	

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	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol
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	?

ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol
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	□

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

SDK Package Version History

Release Date	SDK Package Version	Update
Apr. 21 2014	3.11.0	- Initial Release
May 13 2014	3.11.1	- Changed SDK UI
Aug. 31 2015	3.15.0	- The sampling frequency of startScan method can be changed. - DK-AirCash Wireless LAN support.

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.
1150 King Georges Post Road
Edison, NJ 08837
USA
1-800-782-7636
+1-732-623-5500
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/