# cavo\_sdk Access introduction

## Sdk Introduction

cavo\_sdk is a toolkit developed by cavo, which is only available to partners. Using this toolkit can improve the efficiency of partner development and access.

## Necessary conditions for access

development tools：Android studio

Android version requirements：≥19

Android hardware requirements：support ble

## Use introduction

1. Import sdk

Put RealtekSDK-1.0.1-release.aar UkProtocolLibary.aar in the core directory into the app's libs folder, and add the following configuration to the project's build.gradle

implementation(name: 'UkProtocolLibary', ext: 'aar')

implementation(name: 'RealtekSDK-1.0.1-release', ext: 'aar')

1. Configure Androidmanifest.xml

<uses-permission android:name="android.permission.BLUETOOTH" />

<uses-permission android:name="android.permission.BLUETOOTH\_ADMIN" />

<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />

<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" />

<uses-permission android:name="android.permission.INTERNET" />

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />

1. Initialization

Add the following code on the Application to initialize the SDK

//initialize cavo sdk

WristbandManager.getInstance(this);

1. How to use
2. Sdk only uses the WristbandManager class as the only management class. All device-related operations are implemented through the methods in this class. The WristbandManagerCallback class is an operation callback class. The operations of the device callback app are implemented in this class.。

Scan

WristbandManager.getInstance().startScan();

Connect

WristbandManager.getInstance().connect();

RegisterCallback

WristbandManager.getInstance().registerCallback();

Disconnect

WristbandManager.getInstance().close();

1. Sdk has stored the health data locally. If you need to read the data, please initialize GlobalGreenDAO and call its method.
2. Process example

1) Get WristbandManager instance

2) Scanning device

3) Bind the callback after scanning the device, and then start the connection

4) After the connection is successful, please login first

WrisbandManager.getInstance().startLoginProcess(),

5) and then request the device information

WrisbandManager.getInstance().requestDeviceInfo(),

6) request device support function

WrisbandManager.getInstance().sendFunctionReq(),

7) Set personal information.

WristbandManager.getInstance().setUserProfile();

1. Other operations such as sending synchronous data requests.

WristbandManager.getInstance().sendDataRequest();

## API document

The document is in the apidoc folder of the SDK package. Please open index.html in your browser. Currently the document only supports English.

Operation related classes are in the com.wosmart.ukprotocollibary directory.

Entity related classes are in the com.wosmart.ukprotocollibary.applicationlayer directory

Database-related classes are in the com.wosmart.ukprotocollibary.model directory

## Demo

The SdkDemo project is in the demo directory of the sdk package. Partners can run t he demo to experience the basic functions. Note: The callback of the operation effect is only printed a log.

## Proguard

-keep class com.wosmart.ukprotocollibary.\*\*{\*;}

-keep class org.greenrobot.greendao.\*\*{\*;}

-keepclassmembers class \* extends org.greenrobot.greendao.AbstractDao { public static java.lang.String TABLENAME; }

-keep class \*\*$Properties

## Other considerations

1.Sending data to the device does not support concurrent operations. Do not send data to the device at the same time.

2.The SDK does not store the sport data locally, and the partners need to implement the storage themselves.