# Android SDK Documentation

# 

|  |  |
| --- | --- |
| Version | V1.0.1 |
| Time | 2022.06.17 |
| PIC | Mr.YIN XIAOJUN |
| Revision | Documentation update -------- 20210808 applied to all model \_by MR zhou  Binding methods/activity start methods supplement ----------20211022  Add gain tracker status method in section 8 ------ 20211104  Add set wallpaper setting method in section 6.1 ----20211207  Add walk reminder & period reminder on section 5.72 and 5.13 -20211215  Add Realtek upgrade method + AGPS file update --20220315 |

## One Scan, connect

1. Make sure to apply for the following permissions before scanning

Manifest.permission.ACCESS\_FINE\_LOCATION

Manifest.permission.ACCESS\_COARSE\_LOCATION

Manifest.permission.WRITE\_EXTERNAL\_STORAGE

Manifest.permission.READ\_EXTERNAL\_STORAGE

Manifest.permission.BLUETOOTH\_ADMIN

Manifest.permission.BLUETOOTH

2、Modify the code of binding for the first time according to the following process:：

A.Initiate the disconnection method before starting the scan：BLEManager.disConnect()

B. Start scanning after 5 seconds:：

**Code examples:：**

//Register the callback listener first//

BLEManager.registerScanCallBack(new ScanCallBack.ICallBack(){

@Override

Public void onFindDevice(BLEDevice device){

// Every time a watch is detected, BLEDevice will be called once,BLEDevice is the info of watch

})

// Call interface

BLEManager.startScanDevices()

1. After scanning the the watch that meets the conditions, call the stop scanning immediately BLEManager.stopScanDevices()

（4）Connect the scanned watch

**Code examples:：**

//Register the callback listener first//

BLEManager.registerConnectCallBack(new ConnectCallBack.ICallBack(){

@Override

public void onConnectStart(String macAddress) {}

@Override

public void onConnecting(String macAddress) {}

@Override

public void onRetry(int count, String macAddress) {}

@Override

public void onConnectSuccess(String macAddress) {

//Successful connection callback，macAddress is the MAC address of the watch that is successfully connected

}

@Override

public void onConnectFailed(ConnectFailedReason failedReason, String macAddress) {}

@Override

public void onConnectBreak(String macAddress) {

// when the Bluetooth disconnect, this method will be callback. (such as the watch is too far away from phone and out of Bluetooth range }

@Override

public void onInDfuMode(BLEDevice bleDevice) {}

@Override

public void onDeviceInNotBindStatus(String macAddress) {}

@Override

public void onInitCompleted(String macAddress) {}

})

// Call interface

BLEManager.connect(bleDevice)

1. How to judge if the device is connect or not

BLEManager.isConnected()

If return true, it’s connected. While false means have not being connected

4. Auto Connection

Once the watch is connected to the app successfully, if disconnection happen, SDK will reconnect automatically regularly. Yet if you want to reconnect immediately, call below 2 methods

（1）if (！BLEManager.isConnected()) {

BLEManager.autoConnect();// Reconnect to the watch connect last time automatically

}

（2）if (！BLEManager.isConnected()) {

BLEManager.autoConnect(String macAddress);// Reconnect to the watch with specific Mac address automatically

}

1. Gain the device info of current connected device

BLEDevice device = LocalDataManager.getCurrentDeviceInfo();

1. **BT connection ( Applicable for device support BT calling function )**

**Call method：BTManager.connectBT();**

Remark :

1. This method is used to build BT connection (SDK had realized calling it automatically. In case of failure to build connection, app could call it where in need )
2. SupportFunctionInfo.v2\_get\_bt\_addr if value shows true, means BT function is supported

# Two Binding and unbinding

**1. Binding**

Bind the device after connecting successfully , before binding, check whether it’s bind or not, if not , call the binding method.

**Code examples：**

//Register callback listeners first

BLEManager.registerBindCallBack(new BindCallBack.ICallBack(){

@Override

Public void onSuccess( ){

// Bind a successful callback//

//store the device binding record or device info to the app ( so next time no need to bind again )

}

})

// Call interface

if(!BLEManager.isBind()){

BLEManager.bind()

}

1. **Unbind**

Call unbind if app and watch are connected before unbinding, otherwise call forceUnbind. If connect fail, no need to retry, call BLEManager.bindWithNoRetry();

**Code examples：**

//Register listener callback

you need to register first, then call BLEManager.unbind

BLEManager.registerUnbindCallBack(new UnbindCallBack.ICallBack(){

@Override

Public void onSuccess( ){

// Successfully unbind callback

// App need to delete the device binding info from the device list

BLEManager.disConnect()

}

})

// Call interface

if(BLEManager.isConnected()){

BLEManager.unbind();

} else {

BLEManager.forceUnbind();

BLEManager.disConnect()

// App need to delete the device binding info from the device list

}

If unbinding successfully, normally the watch will vibrate.

Note: After unbinding, you need to call the disconnecting method BLEManager.disconnect () volunteerly

## Three sync data

Sync all data.

1.This function only use two classes: SyncPara and BLEManager. See below explanation:

（1）SyncPara

**Parameters instructions：**

|  |  |
| --- | --- |
| **SyncPara** | |
| timeoutMillisecond | Timeout period for the entire synchronization setting, in milliseconds. Default is 3 minutes |
| ISyncDataListener | Listen for all data callbacks |
| ISyncProgressListener | A callback to listen synchronization progress |
| isNeedSyncConfigData | This member property is not used for the time being , please ignore |
|  |  |
|  |  |
|  |  |

Since there are many callback method for ISyncDataListener , it will be explained in detail in below:：

The following four callback methods are frequently used, other method is similar.

①onGetSportData 是对当天步数数据的回调。HealthSport是当天所有的步数数据的汇总，这个类里面的成员属性如下表所示：（v2协议）

①OnGetSportData is a callback of the number of steps taken that day. HealthSport is a summary of all the steps taken that day. The members attributes of this class are shown in the following table (V2 protocol)：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| sportDataId | dId | year | month | day | totalStepCount | totalCalory | totalDistance | totalActiveTime |
|  |  | Y | M | D | Total steps | Total Calorie | Total Distance | Total activities time,unit :Second |

|  |  |  |  |
| --- | --- | --- | --- |
| keyword | startTime | timeSpace | date |
| N/A | Start time | Time interval | Date |

**HealthSportItem** It’s details of a single piece of data for the steps of the day.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| sportDataId | dId | year | month | day | stepCount | activeTime | calory | distance | date |
|  |  | Y | M | D | Step | Exercise time | Calorie | Distance | Date |

**isSectionItemData** Whether it is the data loaded by page

**②onGetSleepData** It is a callback to the sleep data of the day. HealthSleep is a summary of all sleep data for the day: (V2 protocol)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| sleepDataId | dId | year | month | day | sleepEndedTimeH | sleepEndedTimeM | totalSleepMinutes |
|  |  | Y | M | D | End time/ hour | End time/minutes | Total sleep time  Units : mins |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| lightSleepCount | deepSleepCount | awakeCount | lightSleepMinutes | deepSleepMinutes |
| Times of light sleep | Times of deep sleep | Times of awake | Total sleep time of light sleep (mins) | Total sleep time of deep sleep(mins) |

|  |  |
| --- | --- |
| sleepScore | date |
| Sleep score | Date |

**HealthSleepItem** Details of a single piece of data for the sleep of the day

**Parameters instruction：**

|  |  |
| --- | --- |
| **HealthSleepItem** | |
| year | Y |
| month | M |
| day | D |
| offsetMinute | Offset (unit: minute), used to calculate the time (hour: minute) when the data occurred, When calculating the current point in time, you need to add up all the previous offsets (the cumulative value does not exceed 24\*60=1440); The maximum offset is 255 and the minimum is 0 |
| sleepStatus | Sleep state in the current period: 1: awake 2: light sleep 3: deep sleep |
| date | Date |

**③onGetHeartRateData** It is a callback to the heart rate data of the day. HealthHeartRate is a summary of all heart rate data for the day:

(V2 protocol)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| rateDataId | dId | year | month | day | startTime | silentHeart | warmUpThreshold |
|  | Device ID | Y | M | D | Start time | Resting heart rate | Warm-up exercise threshold |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| burn\_fat\_threshold | aerobic\_threshold | anaerobicThreshold | limit\_threshold | warmUpMins |
| Burn fat exercise threshold | Aerobic exercise threshold | Anaerobic exercise threshold | Limit sport threshold | Warm up mins |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| burn\_fat\_mins | aerobic\_mins | anaerobicMins | limit\_mins | UserMaxHr |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Range1 | Range2 | Range3 | Range4 | Range5 | date |
| Range1 | Range2 | Range3 | Range4 | Range5 | date |

5 heart rate zone

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Range1 | Range2 | Range3 | Range4 | Range5 | date |
| Zone 1 | Zone 2 | Zone 3 | Zone 4 | Zone 5 | Date |

HealthHeartRateItem Is a single piece of data detailing the heart rate for the day：

**Parameters instruction：**

|  |  |
| --- | --- |
| **HealthHeartRateItem** | |
| year | Y |
| month | M |
| day | Day |
| offsetMinute | offset (unit : minutes), used to calculate the time of data occurrence (hours: minutes ) ,  When calculating the current time point, we need to add up all the previous offsets  ( the accumulated value is no more than 24\*60=1440) ;  The offset has a maximum value of 255 and a minimum value of 0 |
| HeartRaveValue | In the automatic measurement mode, the measured heart rate value will generate one value in 15 minutes and up to 96 values a day |
| date | Date |

④**onGetActivityData**  It is a callback of all activity data of the day. (V2 protocol sport data)

HealthActivity：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| activityId | dId | year | month | day | hour | minute | second | hr\_data\_interval\_minute |
|  | Device ID | Y | M | D | H | Min | Sec | 心率数据间隔分钟 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| type | step | durations | calories | distance | avg\_hr\_value | max\_hr\_value |
| Type | Step | Duration time (unit :mins ) | Calorie(Kcal  ) | Distance (meter) | Avg HR | Max heart rate |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| warmUpMins | burn\_fat\_mins | aerobic\_mins | anaerobicMins | limit\_mins |
| warmUp exercise duration (Mins) | burn\_fat exercise duration (mins) | Aerobi exercise duration(mins) | Anaerobic exercise duration (Mins) | Limit exercise duration (mins) |

5 hear rate zone

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| range1 | range2 | range3 | range4 | range5 | date |
| Zone 1 | Zone 2 | Zone 3 | Zone 4 | Zone 5 | date |

|  |  |
| --- | --- |
| hr\_data\_vlaue | The maximum storage time is 2 hours, one data/5s, maximum is 1440 datas (2\*60\*60÷5 = 1440); Greendao cannot store this type, so it is stored in json string format (hr\_data\_vlaue\_json) |
| hr\_data\_vlaue\_json | Json string storing hr\_data\_vlaue data |
| items | Save a single piece of data of steps, calories, and distance |
| items\_json | The json string that saves the items data |

**Parameters instruction**

|  |  |
| --- | --- |
| **HealthActivity.Item** | |
| steps | Steps |
| calories | calories |
| distance | Distance |

⑤onGetHealthSpO2Data It’s callback of blood Oxygen (V3 protocol)

HealthSpO2：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Id | dId | year | month | day | startTime | date |
| Home key | Device iD | Year | Month | Day | Start time of the data since 0:00 | Date |

HealthSpO2Item：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Id | dId | year | month | day | offset | value | date |
| Home key | Device ID | Year | Month | Day | Offset | Value 0-255 | Date |

offset  default unit:S, if function table ex\_table\_main8\_v3\_spo2\_off\_change  = true, units should be min

isSectionItemData ：If it’s page loading data or not

1. .onGetHealthPressureData is call back of stress (V3 protocol)

HealthPressure：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Id | dId | year | month | day | startTime | date |
| Home key | Device ID | Y | M | D | Start time of the data since 0:00 | Time |

HealthPressureItem：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Id | dId | year | month | day | offset | value | date |
| Home key | Device ID | Y | M | D | offset | Value 0-255 | Time |

Offset units min

isSectionItemData :If it’s page loading data or not

6.onGetHealthGpsV3Data It’s GPS health data callback (V3 protocol )

HealthGpsV3：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| year | month | day | hour | minute | second | data\_interval | items | stringItems |
| Year | Month | Day | Hour | Min | Sec | Data interval | HealthGpsItemList set of V3 | source data return by so lib |

HealthGpsItemV3：

|  |  |
| --- | --- |
| latitude | longitude |
| latitude | longitude |

Remark :GPS data are related to the sport via date ( covert the date to the long type timestamp )

6.onGetHealthSwimmingData is callback of swimming data (V3 protocol )

HealthSwimming：

/\*\*

\* year

\*/

public int year;

public int month;// month

public int day;// day

public int hour;// H

public int minute;// Min

public int second;// second

/\*\*

\* Type 0 null, 1 indoor swimming 2. open water swimming ( no data details )

\*/

public int type;

public int calories; // calories

public int distance; // distance

/\*\*

\* Total swimming time ( unit : mins )

\*/

public int duration;

/\*\*

\* swimming laps

\*/

public int trips;

/\*\*

\* average swolf

\*/

public int averageSWOLF;

/\*\*

\* total strokes

\*/

public int totalStrokesNumber;

/\*\*

\* main swimming style

\*0x00 : Medley; 0x01 : Freestyle; 0x02 : Breaststroke; 0x03 : Backstroke; 0x04 : Butterfly stroke \*/

public int swimmingPosture;

/\*\*

\* swimming pool length

\*/

public int poolDistance;

/\*\*

\* distance confirmed by user

\*/

public int confirmDistance;

public List<HealthSwimmingDetail> items; // details

private long dId;// device iD

private Date date;// date

public int avg\_speed;// average pace

public int avg\_frequency;// average stoke frequency

HealthSwimmingDetail：

/\*\*

\* duration, unit:sec

\*/

public int duration;//

/\*\*

\* number of stroke

\*/

public int strokesNumber;

/\*\*

\* stroke efficiency

\*/

public int swolf;

/\*\*

\* swimming Posture;

\*/

public int swimmingPosture;

/\*\*

\* distance

\*/

public int distance;

public int frequency; // frequency (newly added )

public int speed; // pace (newly added )

public int stop\_time;// stop time

public int difference\_time;// Rest time in between of 2 laps units : ms

9.onGetHealthHeartRateSecondData v3 protocol heart rate data

HealthHeartRateSecond：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Id | dId | year | month | day | startTime | date | items （Lise<HealthHeartRateSecondItem >） |
| Home key | Device ID | Y | M | D | Start time | Date | Collection of heart rate data |

HealthHeartRateSecondItem：

|  |  |  |
| --- | --- | --- |
| offset | heartRateVal |  |
| Offset units: S | Heart rate value |  |

offset unit :S

10.onGetHealthSleepV3Data sleep data V3 protocol

HealthSleepV3 field description

public int data\_type; //（return from firmware）algorithm defined return value 1. scientific sleep(with REM and breath score ) 2. nap 3. common sleep ( not scientific sleep )

// fall-asleep time

public int fall\_asleep\_year; // date year

public int fall\_asleep\_month; // month

public int fall\_asleep\_day; // day

public int fall\_asleep\_hour; // // fall-asleep time\_hour

public int fall\_asleep\_minte; // fall-asleep time\_min

// get up time

public int get\_up\_year;

public int get\_up\_month;

public int get\_up\_day;

public int get\_up\_hour; // get\_up\_hour

public int get\_up\_minte; // get\_up\_minute

public int total\_sleep\_time\_mins; //total\_sleep duration unit: mins

public int wake\_mins; // total awake duration unit: mins

public int light\_mins; // total light sleep duration

public int rem\_mins; // total REM duration

public int deep\_mins; // total deep sleep duration

public int wake\_count; // total awake times

public int light\_count; / total light sleep counts

public int rem\_count; // Total REM counts

public int deep\_count; // Total deep sleep counts

public int awrr\_status; // breath status

public int sleep\_score; // sleep score

public int breath\_quality; // breath quality

public List<HealthSleepV3Item> items;

HealthSleepV3Item filed description : Details of sleep

int stage; // stage

int duration; // duration, unit : second

1. onGetHealthSportV3Data（HealthSportV3 healthSportV3）— V3 health data （v3 protocol）

HealthSportV3 field description

public int year;

public int month;

public int day;

public int hour;

public int minute;

public int second;

public int start\_time;

public int per\_minute; //// mins interval of each piece of data (default 15 mis) could be 1 mins, please subject to real value

public int total\_step; // total\_step

public int total\_rest\_activity\_calories; // total activity duration + resting calories

public int total\_distances; // total distance

public int total\_active\_time; // total activities of current day unit min

public int total\_activity\_calories; // total calories

public int item\_count;

public List<Integer> wear\_flag\_array; /

Wearing time of each hour, unit : mins, array lenght 24

public List<Integer> type; // 24H status value 0 means unknow 1means accomplished 2. haven’t accomplished

public List<HealthSportV3Item> items;

public int walk\_goal\_time; //Total walking duration, need to be configured, unit :hour

HealthSportV3Item filed description

public int mode; // Mode 00: quiet 01: mild 10 : Moderately active 11 : intensive

public int step\_count; // step count

public int active\_time; //Active duration unit : mins determined by parameter of maximum per\_minute ,now it is 15minutes.ID206 newly added :time interval 1min.

public int activity\_calories; // calorie of activity

public int distance; // distance unit: meter

public int rest\_activity\_calories; // activity +rest calorie

1. onGetHealthActivityV3Data(HealthActivityV3 healthActivityV3）— v3 sport data （v3 protocol）

HealthActivityV3 field description

/\*\*

\* year

\*/

public int year;

/\*\*

\* month

\*/

public int month;// month

/\*\*

\* day

\*/

public int day;//day

/\*\*

\* hour

\*/

public int hour;//hour

/\*\*

\* Minute

\*/

public int minute;//Minute

/\*\*

\* second

\*/

public int second;//second

public int hr\_data\_interval\_minute;

/\*\*

\* sport type

\*/

public int type; //sport type

/\*\*

\* plan type

\*/

public int act\_type; //Plan type 0 means use type, other type is invalid, use act\_type type // plan type: 0x01: running plan 3km, 0x02: running plan 5km, 0x03: running plan 10km, 0x04: half marathon training (phase 2), 0x05: Marathon Training (Phase 2)

//0x40: 6 minutes relax running, 0x41: 10 minutes relax running, 0x42: 15 minutes relax running, 0x43: Basic combined walking and running 0x44: Advanced combined walking and running , 0x45: Strengthen combined walking and running

//0x80:Stretching after running

//0x80:Stretching after running

/\*\*

\* step

\*/

public int step; //step

/\*\*

\* Duration (unit: seconds)

\*/

public int durations; //Duration

/\*\*

\* Calories (kcal)

\*/

public int calories; //Calories

/\*\*

\* Distance (unit: meters)

\*/

public int distance; //distance

/\*\*

\* average heart rate

\*/

public int avg\_hr\_value;//average heart rate

/\*\*

\* maximum heart rate

\*/

public int max\_hr\_value;//maximum heart rate

/\*\*

\* warm-up minutes

\*/

public int warmUpMins; //warm-up minutes

public int warm\_up\_time;//Warm-up duration (seconds)

/\*\*

\* Fat burning duration (unit: minutes)

\*/

public int burn\_fat\_mins; //Fat burning duration

public int fat\_burning\_time; //Fat burning duration (seconds)

/\*\*

\* Cardio workout duration (unit: minutes)

\*/

public int aerobic\_mins; //Cardio workout duration

public int aerobic\_exercise\_time;//Cardio workout duration (unit: seconds)

/\*\*

\* anaerobic combustion minute data

\*/

public int anaerobicMins; //anaerobic combustion minute data

public int anaerobic\_exercise\_time;//anaerobic combustion duration (seconds)

/\*\*

\* Limit exercise duration (unit: minutes)

\*/

public int limit\_mins; //limit exercise duration (minutes)

public int extreme\_exercise\_time;//Limit exercise duration (seconds)

public int training\_offset;//Course date offset for training starts at zero

public List<ActionTemp> action\_temp;//Action details

public static class ActionTemp{

public int type;//Action type (decimal) 1 fast walk; 2 jogging; 3 moderate running; 4 fast running;

//71 Left leg front stretch; 72 Right leg front stretch; 73 Left leg back stretch; 74 Right leg back stretch; 75 Left leg lunge stretch; 76 Right leg lunge stretch; 77 Stretching the inner left leg; 78 Stretching the inner right leg; 79 Stretching the left calf; 80 Stretching the right calf

public int actual\_time; //Actual exercise time

public int goal\_time; //target time unit is seconds

public int heart\_value; //Heart rate control value

}

/\*\*

\* 5s a heart rate data

\*/

public int[] hr\_data\_vlaue; //The maximum storage time is 2 hours, one data for 5s, and the maximum is 1440; greendao can’t save this type, so it is stored as a json string(hr\_data\_vlaue\_json)

public List<Item> items;

public int grade;////Blood oxygen level 0x00: No level, 0x01: Low level 0x02: Amateur 0x03: Ordinary 0x04: Average 0x05: Good 0x06: Excellent 0x07: Professional

public static class Item{

public int steps;

public int calories;

public int distance;

}

/\*\*

\* Fastest Kilometer pace

\*/

public int fast\_km\_speed;

public List<ItemKMSpeed> items\_km\_speed;

public static class ItemKMSpeed{

public int second;

}

public List<Integer> frequency\_items;

/\*\*

\* mile pace

\*/

public List<Integer> items\_mi\_speed;

/\*\*

\* Kilometer pace

\*/

public int km\_speed;

/\*\*

\* average pace

\*/

public int avg\_speed;

/\*\*

\* maximum pace

\*/

public int max\_speed;//

/\*\*

\* average step pace

\*/

public int avg\_step\_frequency;//

/\*\*

\* maximum step pace

\*/

public int max\_step\_frequency;//

/\*\*

\* average stride

\*/

public int avg\_step\_stride;//

/\*\*

\* maximum stride

\*/

public int max\_step\_stride;//

/\*\*

\* 0: invalid, 1: Activity initiated from app, 2:Activity initiated from watch

\*/

public int sport\_start\_type; //

/\*\*

\* Whether the watch is connected to the app, what kind of report is generated by the app, 1 is connected, 0 is not connected

\*/

public int connect\_app;

public int avg\_pace\_speed;//Average pace Seconds are sent, such as 361，361/60=6 minutes, and the decimals is 1s，6''1'. Kilometers and miles are calculated as miles = kilometers\*1609/1000f

public int fast\_pace\_speed;//fastest pace

public int training\_effect; //Training effect; Unit: None: Range: 1.0 - 5.0 [while transfer the value, need to multiply by 10 times, such as 1.0-5.0 ]

public int vO2max; //maximum vO2; unit: ml/kg/min; range 0-80

public int recovery\_time\_year; // recovery time year

public int recovery\_time\_mon; // recovery time month

public int recovery\_time\_day; //recovery time day

public int recovery\_time\_hour; // recovery time hour

public int recovery\_time\_min; // recovery time mintues

public int recovery\_time\_s; // recovery time seconds

public int min\_hr\_value; //minimum heart rate

public List<Integer> pace\_speed\_items;//real-time pace

public List<Integer> paddle\_number\_items;//Paddle reps save one time per minute

public List<Integer> paddle\_frequency\_items;//Array of paddle frequencies, save average value, frequency: save one time per minute

public List<Integer> tread\_frequency\_items;//Tread frequency array, save average value, frequency: save one time per minute

public int paddle\_number\_count; //Number of paddles

public int paddle\_frequency\_count; //Number of paddle frequencies

public int tread\_frequency\_count; //Number of Tread frequency

public int end\_day; //Number of Tread frequency

public int end\_hour; //Number of Tread frequency

public int end\_minute; //Number of Tread frequency

public int act\_count;// action\_temp, the number of actions

public int completion\_rate;// Action completion rate 0—100

public int hr\_completion\_rate; //heart rate control ratio 0—100

public int in\_class\_calories; //Exercise calories burnt during the course, unit is kcal

（

(2) BLEManager: Management category for understanding how to use SDK methods

(3) After understanding the above two categories, you can start to implement the function of synchronizing all data:

SyncPara syncPara = new SyncPara();  
syncPara.iSyncDataListener = new ISyncDataListener(){  
 @Override  
 public void onGetSportData(HealthSport healthSport, List<HealthSportItem> items, boolean isSectionItemData) {

//todo After getting the data, you can process related business logic, data display and data storage here (other callback methods are similar）  
 }

@Override  
public void onGetSleepData(HealthSleep healthSleep, List<HealthSleepItem> items) {  
}  
  
@Override  
public void onGetHeartRateData(HealthHeartRate healthHeartRate, List<HealthHeartRateItem> items, boolean isSectionItemData) {  
 }  
  
@Override  
public void onGetBloodPressureData(HealthBloodPressed healthBloodPressed, List<HealthBloodPressedItem> items, boolean isSectionItemData) {  
 }  
  
@Override  
public void onGetActivityData(HealthActivity healthActivity) {  
 }  
  
@Override  
public void onGetGpsData(HealthGps healthGps, List<HealthGpsItem> healthGpsItems, boolean isSectionItemData) {  
 }  
  
@Override  
public void onGetHealthSpO2Data(HealthSpO2 healthSpO2, List<HealthSpO2Item> itemList, boolean isSectionItemData) {  
 }  
  
@Override  
public void onGetHealthPressureData(HealthPressure healthPressure, List<HealthPressureItem> itemList, boolean isSectionItemData) {  
 }

@Override //v3 Protocol heart rate callback  
public void onGetHealthHeartRateSecondData(HealthHeartRateSecond healthHeartRateSecond, boolean isSectionItemData) {  
 }  
  
@Override  
public void onGetHealthSwimmingData(HealthSwimming healthSwimming) {  
 }  
  
@Override  
public void onGetHealthActivityV3Data(HealthActivityV3 healthActivityV3) {  
 }  
  
@Override  
public void onGetHealthSportV3Data(HealthSportV3 healthSportV3) {  
 }  
  
@Override  
public void onGetHealthSleepV3Data(HealthSleepV3 healthSleepV3) {  
 }  
  
@Override  
public void onGetHealthGpsV3Data(HealthGpsV3 healthGpsV3) {  
 }  
  
@Override  
public void onGetHealthNoiseData(HealthNoise healthNoise) {  
 }  
  
@Override  
public void onGetHealthTemperature(HealthTemperature healthTemperature) {  
 tvSyncResult.setText("onGetHealthTemperature is ok");  
}  
  
@Override  
public void onGetHealthBloodPressure(HealthBloodPressureV3 healthBloodPressureV3) {  
 tvSyncResult.setText("onGetHealthBloodPressure is ok");  
}  
  
@Override  
public void onGetHealthRespiratoryRate(HealthRespiratoryRate healthRespiratoryRate) {  
  
}

}

syncPara.iSyncProgressListener = new ISyncProgressListener(){  
 @Override  
 public void onStart() { }  
  
 @Override  
 public void onProgress(int progress) { }  
  
 @Override  
 public void onSuccess() { }  
  
 @Override  
 public void onFailed() { }  
};

BLEManager.syncAllData(syncPara);

## Four Getting info

device info

**：**

Interface name:

BLEManager.getBasicInfo

**：**

**Parameter Description:**

|  |  |
| --- | --- |
| **Basic Info** | |
| deviceId | device ID |
| firmwareVersion | Firmware version |
| battStatus | Battery status |
| energe | Battery left |
|  |  |
|  |  |
|  |  |

**Code example:**

Method 1: //Register the callback listener first

BLEManager.registerGetInfoCallBack(new GetDeviceInfoCallBack.ICallBack(){

@Override

Public void onGetBasicInfo(BasicInfo basicInfo){

//Return device information here

}

})

//call interface

BLEManager.getBasicInfo()

Method 2: The sdk stores basicinfo information

BasicInfo basicInfo = LocalDataManager.getBasicInfo();

### Getting Funcition Table

Interface name:

BLEManager.getFunctionTables()

**Code example:**

Method 1: //Register the callback listener first

BLEManager.registerGetDeviceInfoCallBack(new GetDeviceInfoCallBack.ICallBack(){

@Override

Public void onGetFunctionTable(SupportFunctionInfo supportFunctionInfo){

//Return to the list of features supported by the device here

}

})

//call interface

BLEManager.getFunctionTables()

Method 2:

The sdk will also save the function table, and the sdk will automatically get it when it is first bound, and the user calls this

SupportFunctionInfo info = LocalDataManager.getSupportFunctionInfo(); you can get the function table (recommended to use this)

### 3. Get all sports types (24 types)

public static List<Integer> getSupportMotionTypeList() {

SupportFunctionInfo functionInfo = LocalDataManager.getSupportFunctionInfo();

ArrayList<Integer> list = new ArrayList<>();

if (functionInfo == null) {

return list;

}

CommonLogUtil.printAndSave(LogPathImpl.getInstance().getDeviceLogPath(), "Get a list of supported sport types===" + functionInfo.toString());

//outdoor running

if (functionInfo.outdoor\_run) {

list.add(SportType.SOPRT\_TYPE\_OUTDOOR\_RUN);

}

//Running or indoor running

if (functionInfo.indoor\_run) {

list.add(SportType.SOPRT\_TYPE\_INDOOR\_RUN);

}

//Running

if (functionInfo.sport\_type0\_run) {

list.add(SportType.SPORT\_TYPE\_RUN);

}

//outdoor walking

if (functionInfo.outdoor\_walk) {

list.add(SportType.SOPRT\_TYPE\_OUTDOOR\_WALK);

}

//walking

if (functionInfo.sport\_type0\_walk) {

list.add(SportType.SPORT\_TYPE\_WALK);

}

//walking ot indoor walking

if (functionInfo.indoor\_walk) {

list.add(SportType.SOPRT\_TYPE\_INDOOR\_WALK);

}

//on foot

if (functionInfo.sport\_type0\_on\_foot) {

list.add(SportType.SPORT\_TYPE\_ONFOOT);

}

//outdoor cycle

if (functionInfo.outdoor\_cycle) {

list.add(SportType.SOPRT\_TYPE\_OUTDOOR\_CYCLE);

}

//riding

if (functionInfo.sport\_type0\_by\_bike) {

list.add(SportType.SPORT\_TYPE\_CYCLING);

}

//cricket

if (functionInfo.cricket) {

list.add(SportType.SPORT\_TYPE\_CRICKET);

}

//Yoga

if (functionInfo.sport\_type2\_yoga) {

list.add(SportType.SPORT\_TYPE\_YOGA);

}

//indoor cycle

if (functionInfo.indoor\_cycle) {

list.add(SportType.SOPRT\_TYPE\_INDOOR\_CYCLE);

}

//fitness

if (functionInfo.sport\_type1\_fitness) {

list.add(SportType.SPORT\_TYPE\_FITNESS);

}

//swimming

if (functionInfo.sport\_type0\_swim) {

list.add(SportType.SPORT\_TYPE\_SWIM);

}

//pool swimming

if (functionInfo.pool\_swim) {

list.add(SportType.SOPRT\_TYPE\_POOL\_SWIM);

}

//open water swimming

if (functionInfo.open\_water\_swim) {

list.add(SportType.SOPRT\_TYPE\_WATER\_SWIM);

}

//rower

if (functionInfo.rower) {

list.add(SportType.SOPRT\_TYPE\_ROWER);

}

//elliptical

if (functionInfo.elliptical) {

list.add(SportType.SOPRT\_TYPE\_ELLIPTICAL);

}

//mountain climbing

if (functionInfo.sport\_type0\_mountain\_climbing) {

list.add(SportType.SPORT\_TYPE\_CLIMB);

}

//badminton

if (functionInfo.sport\_type0\_badminton) {

list.add(SportType.SPORT\_TYPE\_BADMINTON);

}

//spinning bike

if (functionInfo.sport\_type1\_spinning) {

list.add(SportType.SPORT\_TYPE\_DYNAMIC);

}

//Treadmill

if (functionInfo.sport\_type1\_treadmill) {

list.add(SportType.SPORT\_TYPE\_TREADMILL);

}

//basketball

if (functionInfo.sport\_type2\_basketball) {

list.add(SportType.SPORT\_TYPE\_BASKETBALL);

}

//football

if (functionInfo.sport\_type2\_footballl) {

list.add(SportType.SPORT\_TYPE\_SOCKER);

}

//Tennis

if (functionInfo.sport\_type2\_tennis) {

list.add(SportType.SPORT\_TYPE\_TENNISBALL);

}

//Dance

if (functionInfo.sport\_type3\_dance) {

list.add(SportType.SPORT\_TYPE\_DANCING);

}

//ellipsoid

if (functionInfo.sport\_type1\_ellipsoid) {

list.add(SportType.SPORT\_TYPE\_ELLIPOSID);

}

//Sit-ups

if (functionInfo.sport\_type1\_sit\_up) {

list.add(SportType.SPORT\_TYPE\_SIT\_UP);

}

//push ups

if (functionInfo.sport\_type1\_push\_up) {

list.add(SportType.SPORT\_TYPE\_PUSHUP);

}

//dumbbel

if (functionInfo.sport\_type1\_dumbbell) {

list.add(SportType.SPORT\_TYPE\_DUMBBELLS);

}

//weightlifting

if (functionInfo.sport\_type1\_weightlifting) {

list.add(SportType.SPORT\_TYPE\_LIFTING);

}

//aerobics

if (functionInfo.sport\_type2\_bodybuilding\_exercise) {

list.add(SportType.SPORT\_TYPE\_AEROBICS);

}

//jump rope

if (functionInfo.sport\_type2\_rope\_skipping) {

list.add(SportType.SPORT\_TYPE\_ROPE);

}

//volleyball

if (functionInfo.sport\_type2\_volleyball) {

list.add(SportType.SPORT\_TYPE\_VOLLEYBALL);

}

//pingpong

if (functionInfo.sport\_type2\_table\_tennis) {

list.add(SportType.SPORT\_TYPE\_PINGPONG);

}

//golf

if (functionInfo.sport\_type3\_golf) {

list.add(SportType.SPORT\_TYPE\_GOLF);

}

//baseball

if (functionInfo.sport\_type3\_baseball) {

list.add(SportType.SPORT\_TYPE\_BASEBALL);

}

//ski

if (functionInfo.sport\_type3\_skiing) {

list.add(SportType.SPORT\_TYPE\_SKI);

}

//roller skating

if (functionInfo.sport\_type3\_roller\_skating) {

list.add(SportType.SPORT\_TYPE\_ROLLER);

}

//Pilates

if (functionInfo.pilates) {

list.add(SportType.SPORT\_TYPE\_PILATES);

//Zumba

if (functionInfo.zumba) {

list.add(SportType.SPORT\_TYPE\_ZUMBA);

}

//core training

if (functionInfo.sport\_type3\_core\_training) {

list.add(Sport100Type.SPORT\_TYPE\_CORE\_TRAINING);

}

//traditional strength training

if (functionInfo.sport\_type\_traditional\_strength\_training) {

list.add(Sport100Type.SPORT\_TYPE\_TRADITIONAL\_STRENGTH\_TRAINING);

}

//functional strength training

if (functionInfo.sport\_type3\_strength\_training) {

list.add(Sport100Type.SPORT\_TYPE\_FUNCTIONAL\_STRENGTH\_TRAINING);

}

//high-intensity interval training

if (functionInfo.HIIT) {

list.add(SportType.SOPRT\_TYPE\_HIIT);

}

//Organize and relax

if (functionInfo.sport\_type3\_tidy\_up\_relax) {

list.add(Sport100Type.SPORT\_TYPE\_ORGANIZE\_AND\_RELAX);

}

//Other, free exercise

if (functionInfo.sport\_type0\_other) {

list.add(SportType.SPORT\_TYPE\_OTHER);

}

return list;

}

**4.Get the displayed motion type (v3 protocol)**

**interface name:**

BLEManager.getSupportSportInfoV3()

**Code example:**

Method 1: Register the callback listener first

BLEManager.registerGetDeviceParaCallBack(new GetDeviceParaCallBack.ICallBack(){

@Override

Public void onGetSupportSportInfoV3(SupportSportInfoV3 supportSportInfoV3) {

//This returns the sports types supported by the device

}

})

//call interface

BLEManager.getSupportSportInfoV3()

SupportSportInfoV3 Class Description：

public int defaultShowNum;//The number displayed by default

public int isSupportsSort;//Whether to support default sorting, 0x0 does not support, 0x01 supports

public int maxShowNum;//Maximum number of supported

public int minShowNum;//Minimum number of supported

public List<Integer> sportTypes;//running type,is\_supports\_sort=1 valid,Numbers: default\_show\_num,0 should be filled in the blank

**5. Get shortcut app (v3 protocol)**

**Interface name:**

BLEManager.getMenuList()

**Code example:**

Method 1: Register the callback listener first

BLEManager.registerGetDeviceParaCallBack(new GetDeviceParaCallBack.ICallBack(){

@Override

Public void onGetMenuList(MenuList.DeviceReturnInfo deviceReturnInfo) {

//Here return the device's shortcut application

}

})

//call interfacel

BLEManager.getMenuList()

DeviceReturnInfo Category description

public int minShowNum;//Minimum number of supported

public int maxShowNum;//Maximum number of supported

public int maxNum; //Maximum numbers

public int currentShowNum; //The current number of shows, which means that the first currentShowNum in items is the currently showed shortcut application

public List<MenuList.DeviceReturnInfo.Item> items; //Quick application collection

### 6.Get GPS status

Interface name:

BLEManager.getGpsStatus()

**Code example:**

Method 1: Register the callback listener first

BLEManager.registerGetGpsInfoCallBack(new GpsCallBack.IGetGpsInfoCallBack(){

@Override

public void onGetGpsStatus(GpsStatus gpsStatus) {

//Returns the gps status of the device here

// GPS non-idle state, do not process

if (GpsStatus.STATUS\_IDLE != gpsStatus.gps\_run\_status) {

saveAgpsLog("checkDeviceGpsStatus , gps is not in idle status");

return;

}

//Download GPS file, specific reference ( Point ten Apgs update)

}

})

//call interface

BLEManager.getGpsStatus()

DeviceReturnInfo type description：

public int minShowNum; //Minimum number of shows

public int maxShowNum; // maximum number of shows

public int maxNum; //maximum quantity

public int currentShowNum; //The current number of shows, which means that the first currentShowNum in items is the currently showed shortcut application

public List<MenuList.DeviceReturnInfo.Item> items; //Quick application collection

### 7. Get the version number of ble bt（get version）

First judge whether it supports or not according to the function watchface：

SupportFunctionInfo functionInfo = LocalDataManager.getSupportFunctionInfo();

functionInfo.V3\_support\_set\_v3\_notify\_add\_app\_name 返回ture 代表支持

SupportFunctionInfo functionInfo = LocalDataManager.getSupportFunctionInfo();

functionInfo.V3\_support\_set\_v3\_notify\_add\_app\_name Return true to indicate support

**Interface name：**

BLEManager.getFirmwareAndBt3Version();

**：**

**Code example：**

//register callback listener first

BLEManager.registerGetDeviceParaCallBack(new GetDeviceParaCallBack.ICallBack(){

@Override

public void onGetFirmwareAndBt3Version(FirmwareAndBt3Version firmwareAndBt3Version) {

//Returns the version number of the device here

}

})

//call interface

BLEManager.getFirmwareAndBt3Version()

：

FirmwareAndBt3Version type description：

public int firmware\_version1;//firmware version version1

public int firmware\_version2;//firmware version version2

public int firmware\_version3;//firmware version version3

public int BT\_flag;//BT version valid flag; 0: invalid, 1: indicates that the firmware has the corresponding BT firmware

public int BT\_version1;//BT version version1

public int BT\_version2;//BT version version2

public int BT\_version3;//BT version version3

public int BT\_match\_version1;//The version of BT that needs to be matched version1

public int BT\_match\_version2;//The version of BT that needs to be matched version2

public int BT\_match\_version3;//The version of BT that needs to be matched version3

NOTE：firwareVersion = firmware\_version1 +firmware\_version2 +firmware\_version3; //firmware\_version2,firmware\_version3 less than 10, the front needs to be filled with 0 to display

BT\_version , BT\_match\_version are handled in the same way as firwareVersion

When BT\_version and BT\_mach\_version are inconsistent, you need to upgrade, see the upgrade BT document (12 BT upgrade) for details.

### 8. watch notification app

DeviceParaChangedCallBack.ICallBack changeC = new DeviceParaChangedCallBack.ICallBack() {

@Override

public void onChanged(DeviceChangedPara deviceParaChange) {

}

};

BLEManager.registerDeviceParaChangedCallBack(changeC);

BLEManager.unregisterDeviceParaChangedCallBack(changeC);(Called when ondestory)

This monitor is mainly used to monitor some state changes of the device. The following is the description of the type. The scene that needs to be used can be judged by the corresponding type.

DeviceChangedPara description:

/\*\*

\* data\_type

\* | value | description | remark |

\* | ---- | ------------------------------------- | ---- |

\* | 0x0 | invaild | |

\* | 1 | The watch has been unbinded | |

\* | 2 | Heart rate pattern changes | |

\* | 3 | Blood oxygen production data, change happened | |

\* | 4 | Stress production data, change happened | |

\* | 5 | Quit during Alexa recognition | |

\* | 6 | The firmware initiates a factory reset and notifies the app pop-up reminder | |

\* | 7 | The app needs to enter the camera interface (TIT01 customization) | |

\* | 8 | sos event notification (205 Turkey project custom) | |

\* | 9 | The alarm clock set by alexa, firmware modification, need to send the corresponding notification bit to the app, after the app receives it, send the command to get the alarm clock of V3|

\* | 10 | The firmware has a reminder to delete the schedule. The app side ( cmd = 0x33 , cmd\_id = 0x36 ) queries the list and needs to update the corresponding list data|

\* | 11 | The firmware side has modified the corresponding watchface sub-style, and notified the app to get it（command\\_id=0x33， key= 0x5000）|

\* | 12 |Firmware notification ios update notification icon and name|

\* | 13 |The firmware notifies the app that the icon has been updated, and notifies the app to obtain the status of the updated icon|

\* | 14 |The firmware requests to reset the weather, the app receives it, then sends the weather data|

\* |15 |The number of firmware steps increases by 2000 steps each time, the device will requests the app to synchronize data, and the app calls the synchronization interface|

\* |16 |The firmware detects the end of sleep, requests the app to synchronize sleep data, and the app calls the synchronization interface to synchronize|

\* |17 |Firmware fitness data modification, notify the app to update the fitness data|

\* |18 |The firmware is fully charged and a reminder is sent. After the app receives it, the notification bar shows that the device is fully charged.|

\* |19 |After exercising, after manually measuring heart rate, manually measuring blood oxygen, and manually measuring pressure, the device will automatically request synchronization, first check the link status, this synchronization will not be performed if it is not connected, and the synchronization request will be initiated again after the next automatic synchronization condition is met.|

\* |20 |Firmware modification heart rate notification status type, stress notification status type, blood oxygen notification status type, physiological cycle notification status type, health guidance notification status type, reminder notification status type Notification app update heart rate, pressure, blood oxygen, physiological cycle, health guidance , reminder notification status type|

\* |21 |The firmware pressure value calculation is completed, and the app is notified to obtain the pressure value|

\* |22 |The firmware notifies the app, the firmware pressure calibration failed (the firmware exits the measurement interface/detection failed/detection timed out/not worn)|

\* |23 |When the firmware generates a heart rate alert that is too high or too low, notify the app to obtain heart rate data|

\* |24 |Firmware notifies app bt bluetooth is connected|

\* |25 |Firmware notify app bt bluetooth disconnect|

\* |26 |Firmware bluetooth call starts|

\* |27 |Firmware bluetooth call ended|

\* |28 |The new version of the firmware sends a notification command every 4 minutes and 30 seconds to fix the problem that ios will display offline|

\* |29 |Notify app that exercise starts (Acts on intercepting watchface transmission same as 26)|

\* |30 |Notify the app of the end of the exercise（Acts on intercepting watchface transmission same as 27）|

\* |31 |After the firmware restarts, reconnect and send a notification to the app (the app needs to obtain the firmware version information when it receives the notification)|

\* |32 |When the device is idle (without using aleax), a notification needs to be reported to the app (the time interval is 1 hour)|

\* |33 |Notify the app when the firmware space is completed and continue to transfer the watch face file|

\*/

public int dataType;

### 9.watch control app

DeviceControlAppCallBack.ICallBack callBack= new DeviceControlAppCallBack.ICallBack() {

@Override

public void onControlEvent(DeviceControlAppCallBack.DeviceControlEventType deviceControlEventType, int var2) {

//control event

}

@Override

public void onFindPhone(boolean b, long l) {

//Received a request to find a phone

b true:start looking for a phone false : stop looking for your phone

}

l overtime time

}

@Override

public void onOneKeySOS(boolean b, long l) {

//sos event

b true:start sos event false : stop sos event

}

@Override

public void onAntiLostNotice(boolean b, long l) {

}

};

BLEManager.registerDeviceControlAppCallBack(callBack);

BLEManager.unregisterDeviceControlAppCallBack(callBack);(called when ondestory)

DeviceControlEventType ：description

/\*\*

\* control event

\* <br/>

\* Control type

\*/

public enum DeviceControlEventType {

/\*\*

\* music starts

\* <br/>

\* Start

\*/

START,

/\*\*

\* music pause

\* <br/>

\* Pause

\*/

PAUSE,

/\*\*

\* music stop

\* <br/>

\* Stop

\*/

STOP,

/\*\*

\* previous song

\* <br/>

\* Previous

\*/

PREVIOUS,

/\*\*

\* next song

\* <br/>

\* Next

\*/

NEXT,

/\*\*

\* Take pictures, single shot

\* <br/>

\* Take one photo

\*/

TAKE\_ONE\_PHOTO,

/\*\*

\* Continuous shooting

\* <br/>

\* Take multi photo

\*/

TAKE\_MULTI\_PHOTO,

/\*\*

\* volume increase

\* <br/>

\* Volume up

\*/

VOLUME\_UP,

/\*\*

\* volume decrease

\* <br/>

\* Volume down

\*/

VOLUME\_DOWN,

/\*\*

\* Turn on the camera

\* <br/>

\* Open camera

\*/

OPEN\_CAMERA,

/\*\*

\* Turn off the camera

\* <br/>

\* Close camera

\*/

CLOSE\_CAMERA,

/\*\*

\* answer the phone

\* <br/>

\* Answer phone

\*/

ANSWER\_PHONE,

/\*\*

\* reject the call

\* <br/>

\* Reject phone

\*/

REJECT\_PHONE,

/\*\*

\* phone mute

\* <br/>

\* Reject phone

\*/

MUTE\_PHONE,

/\*\*

\* Request system pairing

\*/

REQUEST\_PAIRED,

/\*\*

\* control music percentage

\*/

VOLUME\_PERCENTAGE

}

### 10. Get screen brightness information

**Interface name:**

BLEManager.getScreenBrightness()

**Code example:**

Method 1: Register the callback listener first

BLEManager.registerGetDeviceParaCallBack(new GetDeviceParaCallBack.ICallBack(){

@Override

public void onGetScreenBrightness(ScreenBrightness screenBrightness) {

}

})

//call interface

BLEManager.getScreenBrightness()

## Five parameter setting

### set time

**Interface name:**

（1）BLEManager.setTime() //This method will directly get the time setting of the phone to the device

（2）BLEManager.setTime(SystemTime time)

**Code example：**

//Register the callback listener first

BLEManager.registerSettingCallBack(new SettingCallBack.ICallBack(){

@Override

Public void onSuccess(SettingType type, Object data){

if (settingType == SettingCallBack.SettingType.TIME) {

//The callback after successful setting, SettingType means the setting type, data meas data returned

}

}

@Override

public void onFailed(SettingCallBack.SettingType settingType) {

}

})

//call interface

SystemTime systemTime = new SystemTime();

systemTime.year = 2021;

systemTime.monuth = 08;

systemTime.day = 26;

systemTime.hour = 17;

systemTime.minute = 30;

systemTime.second = 50;

BLEManager.setTime(systemTime);

### setfindPhone

**interface name：**

（1）BLEManager.setFindPhoneSwitch(boolean isOpen)to the device

**code example：**

//registerSettingCallBack

BLEManager.registerSettingCallBack(new SettingCallBack.ICallBack(){

@Override

Public void onSuccess(SettingType type, Object data){

if (settingType == SettingCallBack.SettingType.FIND\_PHONE\_SWITCH) {

//Callback after successful setting，SettingType type of setting，data returned

}

}

@Override

public void onFailed(SettingCallBack.SettingType settingType) {

}

})

//call interface

BLEManager.setFindPhoneSwitch(true);

### set alarm

Whether support V3 alarm clock： true support，false not support

LocalDataManager.getSupportFunctionInfo().ex\_table\_main8\_v3\_sync\_alarm

**v2 protocol：（**ex\_table\_main8\_v3\_sync\_alarm = false **）**

1. **Get a list of alarms：**

public List<Alarm> getAlarmList() {

List<Alarm> alarmList = LocalDataManager.*getAlarm*();

List<Alarm> list = new ArrayList<>();

if (alarmList != null) {

for (Alarm alarm : alarmList) {

if (alarm.getAlarmStatus() == Alarm.*STATUS\_DISPLAY*) {

list.add(alarm);

}

}

}

return list;

}

### set alarm

**interface name：**

BLEManager.setAlarm(List<Alarm> alarmList)

**code example：**

//registerSettingCallBack

BLEManager.registerSettingCallBack(new SettingCallBack.ICallBack(){

@Override

Public void onSuccess(SettingType type, Object data){

if (settingType == SettingCallBack.SettingType.ALARM) {

//Callback after successful setting，SettingType type of setting，data return }

}

})

//call interface

Alarm mAlarm = new Alarm();

mAlarm.setAlarmHour(hour);

mAlarm.setAlarmMinute(min);

mAlarm.setAlarmType(alarmType);//Alarm type

mAlarm.setOn\_off(onOff);//alarm switch

mAlarm.setWeekRepeat(weeks);//week repeating week

List<Alarm> alarms = new ArrayList<>();// This collection uses the collection obtained above（getAlarmList），Otherwise, the original alarm clock would not exist alarms.add(mAlarm);

The The method of setting the alarm clock needs to define a unique id for the alarm clock, and then send it to the device

public void sendAlarmClock2Device(@NonNull List<Alarm> alarmList) {

for (int i = 0; i < alarmList.size(); i++) {

Alarm alarm = alarmList.get(i);

if (alarm == null) {

continue;

}

alarm.setAlarmId(i + 1);

}

BLEManager.*setAlarm*(alarmList);

}

**v3 protocol：（**ex\_table\_main8\_v3\_sync\_alarm = true**）**

1. Get the alarm clock

private GetDeviceParaCallBack.ICallBack getParaCallBack = new GetDeviceParaCallBack.ICallBack() {

@Override

public void onGetAlarmV3(List<AlarmV3> alarmV3List) {

//闹钟列表

}

}

BLEManager.*registerGetDeviceParaCallBack*(mICallBack);

BLEManager.*getAlarmV3*();

Set alarm

**interface name：**

BLEManager.setAlarmV3(List<Alarm> alarmList)

**code example：**

//registerSettingCallBack

BLEManager.registerSettingCallBack(new SettingCallBack.ICallBack(){

@Override

Public void onSuccess(SettingType type, Object data){

if (settingType == SettingCallBack.SettingType.ALARM\_V3) {

//Callback after successful setting，SettingType type of setting，data return }

}

})

//call interface

Alarm mAlarm = new Alarm();

mAlarm.setAlarmHour(hour);

mAlarm.setAlarmMinute(min);

mAlarm.setAlarmType(alarmType);//Alarm type

mAlarm.setOn\_off(onOff);//alarm switch

mAlarm.setWeekRepeat(weeks);//week repeating week

List<Alarm> alarms = new ArrayList<>();// This collection uses the collection obtained above (alarmV3List), otherwise the original alarm clock will not existalarms.add(mAlarm);

The method of setting the alarm clock needs to define a unique id for the alarm clock, and then send it to the device

public void sendAlarmClock2Device(@NonNull List<Alarm> alarmList) {

for (int i = 0; i < alarmList.size(); i++) {

Alarm alarm = alarmList.get(i);

if (alarm == null) {

continue;

}

alarm.setAlarmId(i + 1);

}

BLEManager.*setAlarmv3*(alarmList);

}

Note: The number of alarm clocks is limited， functionInfo.alarmCount

This is the number of alarm clocks, if it exceeds the limit, the setting will be unsuccessful.

### set step goal

**interface name：**

BLEManager.setGoal(Goal goal)

**code example：**

//registerSettingCallBack

BLEManager.registerSettingCallBack(new SettingCallBack.ICallBack(){

@Override

Public void onSuccess(SettingType type, Object data){

if (settingType == SettingCallBack.SettingType.GOAL) {

//Callback after successful setting，SettingType type of setting，data return

}

}

})

//call interface

Goal goal = new Goal();

goal.sport\_step = 10000;

BLEManager.setGoal(goal);

### reboot

**interface name：**

BLEManager.reBoot()

**code example：**

//registerRebootCallBack

BLEManager.registerRebootCallBack(new registerRebootCallBack.ICallBack(){

@Override

public void onSuccess() {

}

public void onFailed() {

}

})

//call interface

BLEManager.reBoot()

### RestoreFactory

**interface name：**

BLEManager.setRestoreFactory();

**code example：**

//registerSettingCallBack

BLEManager.registerSettingCallBack(new SettingCallBack.ICallBack(){

@Override

Public void onSuccess(SettingType type, Object data){

if (settingType == SettingCallBack.SettingType.RESTORE\_FACTORY) {

//

The callback after the setting is successful，SettingType type of setting，data return

}

}

})

//call interface

BLEManager.setRestoreFactory();

### setUpHandGesture

### **interface name：**

BLEManager.setUpHandGesture(UpHandGesture upHandGesture）

**code example：**

//registerSettingCallBack

BLEManager.registerSettingCallBack(new SettingCallBack.ICallBack(){

@Override

public void onSuccess(SettingType type, Object data){

if (settingType == SettingCallBack.SettingType.UP\_HAND\_GESTURE) {

//Callback after successful setting，SettingType type of setting，data return

}

}

})

//call interface

UpHandGesture upHandGesture = LocalDataManager.getUpHandGesture();

if (upHandGesture == null) {

upHandGesture = new UpHandGesture();

}

upHandGesture.onOff=UpHandGesture.STATE\_ON ;

BLEManager.setUpHandGesture(upHandGesture);

### setUpHandGesture

**interface name：**

BLEManager.setMusicSwitch(boolean isSwitchOn）

**code example：**

//registerSettingCallBack

BLEManager.registerSettingCallBack(new SettingCallBack.ICallBack(){

@Override

public void onSuccess(SettingType type, Object data){

if (settingType == SettingCallBack.SettingType.MUSIC\_SWITCH) {

//Callback after successful setting，Setting Type type of setting，data return

}

}

})

//call interface

BLEManager.setMusicSwitch(true);

### setMusicControlInfo

1.First judge whether to support setting the music name according to the menu：

SupportFunctionInfo functionInfo = LocalDataManager.getSupportFunctionInfo();

functionInfo.V3\_support\_set\_v3\_music\_name return ture means support

**interface name：**

BLEManager.setMusicControlInfo(MusicControlInfo musicControlInfo)

**code example：**

//registerSettingCallBack

BLEManager.registerSettingCallBack(new SettingCallBack.ICallBack(){

@Override

public void onSuccess(SettingType type, Object data){

if (settingType == SettingCallBack.SettingType.MUSIC\_CONTROL\_INFO) {

//Callback after successful setting，SettingType type of setting，data return

}

}

})

//call interface

MusicControlInfo musicInfo = new MusicControlInfo();

musicInfo.status = MusicControlInfo.STATUS\_PLAY;

musicInfo.musicName = "boy";

//musicInfo.singerName = "";//V3\_music\_control\_02\_add\_singer\_name = true

BLEManager.setMusicControlInfo(musicInfo);

MusicControlInfo Field Description：

public static final int STATUS\_INVALID = 0x00;

public static final int STATUS\_PLAY = 0x01;

public static final int STATUS\_PAUSE = 0x02;

public static final int STATUS\_STOP = 0x03;

public int status;// state of music

public int curTimeSecond;// current music time, second

public int totalTimeSecond;// total music time

public String musicName;// Music name String [maximum 64 bytes]

public String singerName;//Singer name information

### setPhoneVoice

**interface name：**

BLEManager.setPhoneVoice(PhoneVoice phoneVoice）

**Code example:**

//registerSettingCallBack

BLEManager.registerSettingCallBack(new SettingCallBack.ICallBack(){

@Override

public void onSuccess(SettingType type, Object data){

if (settingType == SettingCallBack.SettingType.PHONE\_VOICE) {

//Callback after successful setting，SettingType type of setting，data return

}

}

})

//call interface

PhoneVoice phoneVoice = new PhoneVoice();

phoneVoice.total\_voice = 100;//maximum volume

phoneVoice.now\_voice = 50;//current volume

BLEManager.setPhoneVoice(phoneVoice);

### not disturb mode setting

1. According to the function table, determine whether the device supports the setting of Do Not Disturb mode：

public SupportFunctionInfo getSupportFunctionInfo() {

SupportFunctionInfo functionInfo = LocalDataManager.getSupportFunctionInfo();

return functionInfo == null ? new SupportFunctionInfo() : functionInfo;

}

public boolean isSupportGetDndMode() {

return getSupportFunctionInfo().ex\_main3\_get\_do\_not\_disturb; //true mean support settings }

2、Get Do Not Disturb：

**interface name：**

BLEManager.getDoNotDisturbPara()

**code example：**

**NotDisturbPara notDisturbPara;**

//registerSettingCallBack

BLEManager.registerGetDeviceParaCallBack(new BaseDeviceParaCallBack(){

@Override

public void onGetDoNotDisturbPara(NotDisturbPara notDisturbPara) {

if (notDisturbPara == null) { //initialize default values

notDisturbPara = new NotDisturbPara();

notDisturbPara.onOFf = NotDisturbPara.STATE\_OFF;

notDisturbPara.startHour = 9;

notDisturbPara.startMinute = 0;

notDisturbPara.endHour = 18;

notDisturbPara.endMinute = 0;

notDisturbPara.noontimeRestOnOff = NotDisturbPara.STATE\_OFF;

notDisturbPara.noontimeRestStartHour = 22;

notDisturbPara.noontimeRestStartMinute = 0;

notDisturbPara.noontimeRestEndHour = 7;

notDisturbPara.noontimeRestEndMinute = 0;

notDisturbPara.setWeeks(new boolean[]{true, true, true, true, true, true, true});

}

}

}

})

//call interface

BLEManager.getDoNotDisturbPara();

3.Set up do not disturb mode：

**interface name：**

BLEManager.getDoNotDisturbPara()

**code example：**

**NotDisturbPara notDisturbPara;**

BLEManager.registerSettingCallBack(new SettingCallBack.ICallBack(){

@Override

public void onSuccess(SettingType type, Object data){

if (settingType == SettingCallBack.SettingType.NOT\_DISTURB) {

//Callback after successful setting，Setting Type type of setting，data return

}

}

})

//call interface

BLEManager.setNotDisturbPara(**notDisturbPara**);//Set the value obtained above to the device, you can modify.

### set userinfo

**interface name：**

BLEManager.setUserInfo(UserInfo userInfo)

**code example：**

//.registerSettingCallBack

BLEManager.registerSettingCallBack(new SettingCallBack.ICallBack(){

@Override

Public void onSuccess(SettingType type, Object data){

if (settingType == SettingCallBack.SettingType.USER\_INFO) {

//Callback after successful setting，SettingType type of setting，data return

}

}

})

//call interface

UserInfo userInfo = new UserInfo();

userInfo.year = 2001;

userInfo.month = 12;

userInfo.day = 7;

userInfo.weight = 60;

userInfo.height = 180;

userInfo.gender = 1;

BLEManager.setUserInfo(userInfo);

### set unit ，language

**interface name：**

BLEManager.setUnit(Units unit)

**code example：**

//registerSettingCallBack

BLEManager.registerSettingCallBack(new SettingCallBack.ICallBack(){

@Override

Public void onSuccess(SettingType type, Object data){

if (settingType == SettingCallBack.SettingType.UNIT) {

//Callback after successful setting，SettingType type of setting，data return

}

}

})

//call interface

Units mUnits = new Units();

mUnits.weight = Units.WEIGHT\_UNIT\_KG;

mUnits.language = Units.UNIT\_DEFAULT;

mUnits.timeMode = Units.UNIT\_DEFAULT;

BLEManager.setUnit(mUnits);

//constant unit

WEIGHT\_UNIT\_KG//

WEIGHT\_UNIT\_LB//

WEIGHT\_UNIT\_ST//

LANG\_ZH//

LANG\_DE//

LANG\_JA//（There are many languages, so I won't list them all）

TIME\_MODE\_12//12 Hour system

TIME\_MODE\_24//24 Hour system

### set screenBrightness)

**Interface name：**

BLEManager.setScreenBrightness(int level)

**code example：**

//registerSettingCallBack

BLEManager.registerSettingCallBack(new SettingCallBack.ICallBack(){

@Override

Public void onSuccess(SettingType type, Object data){

if (settingType == SettingCallBack.SettingType.SCREEN\_BRIGHTNESS) {

//Callback after successful setting，SettingType type of setting，data return

}

}

})

//call interface

BLEManager.setScreenBrightness(3);

### set period

**Interface name：**

BLEManager.setMenstrual(Menstrual menstrual)

**Parameter Description：**

|  |  |
| --- | --- |
| **Menstrual** | |
| on\_off | Identifies switch status |
| menstrual\_length | menstrual length |
| menstrual\_cycle | menstrual cycle |
| last\_menstrual\_year | Last menstrual period start time - year |
| last\_menstrual\_month | Last menstrual period start time - month |
| last\_menstrual\_day | Last menstrual period start time - day |
| STATUS\_ON | Constant: Indicates that the switch is on |
| STATUS\_OFF | Constant: Indicates that the switch is off |
| ovulation\_interval\_day | The interval between the start of the next period and the day of ovulation, usually 14 days |
| ovulation\_before\_day | The number of fertile days before ovulation, usually 5 |
| ovulation\_after\_day | The number of fertile days after ovulation, usually 5 |
| notify\_flag | 0:invalid ； 1：Allow notifications； 2：silent notification； 3：Turn off notifications |

code example：

//registerOtherProtocolCallBack

BLEManager.registerOtherProtocolCallBack(new OtherProtocolCallBack.ICallBack(){

@Override

Public void onSuccess(OtherProtocolCallBack.SettingType type){

if (settingType == SettingCallBack.SettingType.ALARM) {

//code example，SettingType type of setting，data return }

}

})

Menstrual menstrual = new Menstrual();

menstrual .menstrual\_length = 7;

menstrual .menstrual\_cycle = 30;

menstrual .last\_menstrual\_year = 2021;

menstrual .last\_menstrual\_month = 9;

menstrual .last\_menstrual\_day = 3;

//call interface

BLEManager.setMenstrual(menstrual);

### set long sit remind

**Interface name：**

BLEManager.setLongSit(LongSit longSit)

**Parameter Description：**

|  |  |
| --- | --- |
| **LongSit** | |
| startHour | Start time：hour |
| startMinute | Start time：minute |
| endHour | Start time：hour |
| endMinute | End up time：minute |
| interval | interval |
| repetitions | Integer, bit[1] to bit[7] represent the repetition status from Monday to Sunday respectively |
| onOff | Identifies switch status |
| weeks | Array of booleans: Week repeats, seven booleans represent Monday to Sunday |

code example：

//registerSettingCallBack

BLEManager.registerSettingCallBack(new SettingCallBack.ICallBack(){

@Override

Public void onSuccess(SettingType type, Object data){

if (settingType == SettingCallBack.SettingType.ALARM) {

//Callback after successful setting，SettingType type of setting，data return

}

}

})

LongSit longSit = new LongSit();

longSit.setStartHour(startHour);

longSit.setStartMinute(startMinute);

longSit.setEndHour(endHour);

longSit.setEndMinute(endMinute);

longSit.setInterval(lenth);

longSit.setOnOff(onOff);

longSit.setWeeks(weeks);

//call interface

BLEManager.setLongSit(longSit);

### set drink water remind

**Interface name：**

BLEManager.setDrinkWaterReminder(DrinkWaterReminder drinkWaterReminder)

**Parameter Description：**

|  |  |
| --- | --- |
| **DrinkWaterReminder** | |
| startHour | Start time：hour |
| startMinute | Start time：minute |
| endHour | Start time：hour |
| endMinute | End up time：minute |
| interval | interval |
| repeat | Integer, bit[1] to bit[7] represent the repetition status from Monday to Sunday respectively |
| onOff | Identifies switch status |
| weeks | Array of booleans: Week repeats, seven booleans represent Monday to Sunday |
| STATUS\_ON | Constant: Indicates that the switch is on |
| STATUS\_OFF | Constant: Indicates that the switch is off |
|  |  |

code example：

//registerSettingCallBack

BLEManager.registerSettingCallBack(new SettingCallBack.ICallBack(){

@Override

Public void onSuccess(SettingType type, Object data){

if (settingType == SettingCallBack.SettingType.ALARM) {

//Callback after successful setting，SettingType type of setting，data return

}

}

})

DrinkWaterReminder waterReminder = new DrinkWaterReminder();

waterReminder.setStartHour(9);

waterReminder.setStartMinute(0);

waterReminder.setEndHour(18);

waterReminder.setEndMinute(0);

waterReminder.setInterval(30);

waterReminder.setWeeks(new boolean[]{true, true, true, true, true, false, false});

//call interface

BLEManager.setDrinkWaterReminder(waterReminder);

### set weather

Switch settings

//registerSettingCallBack

BLEManager.registerSettingCallBack(new SettingCallBack.ICallBack(){

@Override

Public void onSuccess(SettingType type, Object data){

if (settingType == SettingCallBack.SettingType.WEATHER\_SWITCH) {

// send weather to device

}

}

})

//true turn on false turn

BLEManager.setWeatherSwitch(isSwitchOn);

**call interface:**

BLEManager.setWeatherData(WeatherInfo weatherInfo)

weather issued

//Support v3 protocol

if (supportFunctionInfo.V3\_support\_set\_v3\_weather) {

BLEManager.*setWeatherDataV3*(weatherInfoV3)

}else {//v2 协议

BLEManager.*setWeatherData*(*mWeatherInfo*);

}

**v2 protocol**

**Parameter Description：**

|  |  |
| --- | --- |
| **WeatherInfo** | |
| type | weather type |
| temp | Current Temperature |
| max\_temp | Higher temperature of the day |
| min\_temp | Lowest temperature of the day |
| humidity | current humidity |
| uv\_intensity | Current UV Intensity |
| aqi | Air index of the day |
| future | Weather conditions for the next 3 days |
|  |  |

|  |  |
| --- | --- |
| **WeatherFutureInfo** | |
| type | weather type |
| min\_temp | Lowest temperature of the day |
| max\_temp | The highest temperature of the day |

type is the constant for weather already defined in WeatherInfo class. E.g

public static int WEATHER\_TYPE\_UNKNOWN = 0x00;//unknown

public static int WEATHER\_TYPE\_CLEAR = 0x01;//clear

public static int WEATHER\_TYPE\_CLOUDY = 0x02;//partly cloudy

public static int WEATHER\_TYPE\_OVERCASTSKY = 0x03;//Negative

public static int WEATHER\_TYPE\_RAIN = 0x04;//rain

public static int WEATHER\_TYPE\_RAINSTORM = 0x05;//rainstorm

/\*\*

\*shower

\*/

public static int WEATHER\_TYPE\_SHOWER = 0x06;

/\*\*

\*Snow

\*/

public static int WEATHER\_TYPE\_SNOW = 0x07;

/\*\*

\*sleet

\*/

public static int WEATHER\_TYPE\_SLEET = 0x08;

/\*\*

\*typhoon

\*/

public static int WEATHER\_TYPE\_TYPHOON = 0x09;

/\*\*

\*sandstorm

\*/

public static int WEATHER\_TYPE\_SANDSTORMS = 0x0A;

/\*\*

\*clear night

\*/

public static int WEATHER\_TYPE\_CLEAR\_NIGHT = 0x0B;

/\*\*

\*cloudy at night

\*/

public static int WEATHER\_TYPE\_CLOUDY\_NIGHT = 0x0C;

/\*\*

\*hot

\*/

public static int WEATHER\_TYPE\_HOT = 0x0D;

/\*\*

\* cold

\*/

public static int WEATHER\_TYPE\_COLD = 0x0E;

/\*\*

\*breeze

\*/

public static int WEATHER\_TYPE\_BREEZE = 0x0F;

/\*\*

\*windy

\*/

public static int WEATHER\_TYPE\_GALE = 0x10;

/\*\*

\*smog

\*/

public static int WEATHER\_TYPE\_HAZE = 0x11;

/\*\*

\*shower

\*/

public static int WEATHER\_TYPE\_ZHENYU = 0x12;

/\*\*

\*Cloudy to clear

\*/

public static int WEATHER\_TYPE\_YUN = 0x13;

/\*\*

\*crescent

\*/

public static int WEATHER\_TYPE\_NEW\_MOON = 0x14;

/\*\*

\*crescent moon

\*/

public static int WEATHER\_TYPE\_MOTH\_EYEBROW\_MOON = 0x15;

/\*\*

\*first quarter moon

\*/

public static int WEATHER\_TYPE\_FIRST\_QUARTER\_MOON = 0x16;

code example：

//registerAppSendDataCallBack

BLEManager.registerAppSendDataCallBack(new AppSendDataCallBack.ICallBack(){

@Override

Public void onSuccess(DataType type){

if (settingType == AppSendDataCallBack.DataType.WEATHER) {

//Callback after successful setting，SettingType type of setting，data return }

}

})

WeatherInfo mWeatherInfo = new WeatherInfo ();

。。。。

//call interface

BLEManager.setWeatherData(mWeatherInfo );

V3 protocol

**Parameter Description：（Refer to SDK demo for details）**

|  |  |
| --- | --- |
| **WeatherInfoV3 （v3weather）** | |
| weather\_type | weather type |
| today\_emp | Today’s Temperature |
| today\_max\_temp | Today’s highest temperature |
| today\_min\_temp | Today’s lowest temperature |
| city\_name | City Name |
| hours\_weather\_items | 24 hour weather List<Hour24> |
| humidity | Humidity 0-100 |
| today\_uv\_intensity | Today’s UV intensity |
| air\_quality | Air quality |
| precipitation\_probability | Precipitation rate |
| wind\_speed | Wind speed |
| future\_items | 7 days weather data List<Future> |

|  |  |
| --- | --- |
| Hour24 (24 hour weather) | |
| weather\_type | Weather type |
| temperature | Temperature |
| probability | Probability |

|  |  |
| --- | --- |
| Future (7 days weather data) | |
| weather\_type | Weather type |
| max\_temp | Maximum temperature |
| min\_temp | Minimum temperature |

type The constants for weather are already defined in the WeatherInfoV3 E.g:

public static int WEATHER\_TYPE\_UNKNOWN = 0x00;//UNKNOWN

public static int WEATHER\_TYPE\_CLEAR = 0x01;//CLEAR

public static int WEATHER\_TYPE\_CLOUDY = 0x02;//CLOUDY

public static int WEATHER\_TYPE\_OVERCASTSKY = 0x03;//OVERCAST SKY

public static int WEATHER\_TYPE\_RAIN = 0x04;//RAIN

public static int WEATHER\_TYPE\_RAINSTORM = 0x05;//RAINSTORM

/\*\*

\*SHOWER

\*/

public static int WEATHER\_TYPE\_SHOWER = 0x06;

/\*\*

\*SNOW

\*/

public static int WEATHER\_TYPE\_SNOW = 0x07;

/\*\*

\*SLEET

\*/

public static int WEATHER\_TYPE\_SLEET = 0x08;

/\*\*

\*TYPHOON

\*/

public static int WEATHER\_TYPE\_TYPHOON = 0x09;

/\*\*

\*SANDSTORMS

\*/

public static int WEATHER\_TYPE\_SANDSTORMS = 0x0A;

/\*\*

\*CLEAR NIGHT

\*/

public static int WEATHER\_TYPE\_CLEAR\_NIGHT = 0x0B;

/\*\*

\*CLOUDY NIGHT

\*/

public static int WEATHER\_TYPE\_CLOUDY\_NIGHT = 0x0C;

/\*\*

\*HOT

\*/

public static int WEATHER\_TYPE\_HOT = 0x0D;

/\*\*

\*COLD

\*/

public static int WEATHER\_TYPE\_COLD = 0x0E;

/\*\*

\*BREEZE

\*/

public static int WEATHER\_TYPE\_BREEZE = 0x0F;

/\*\*

\*GALE

\*/

public static int WEATHER\_TYPE\_GALE = 0x10;

/\*\*

\*HAZE

\*/

public static int WEATHER\_TYPE\_HAZE = 0x11;

/\*\*

\*RAIN SHOWER

\*/

public static int WEATHER\_TYPE\_ZHENYU = 0x12;

/\*\*

\*CLOUDS EARLY / CLEARING LATE

\*/

public static int WEATHER\_TYPE\_YUN = 0x13;

/\*\*

\*NEW MOON

\*/

public static int WEATHER\_TYPE\_NEW\_MOON = 0x14;

/\*\*

\*WAXING CRESCENT MOON

\*/

public static int WEATHER\_TYPE\_MOTH\_EYEBROW\_MOON = 0x15;

/\*\*

\*FIRST QUARTER MOON

\*/

public static int WEATHER\_TYPE\_FIRST\_QUARTER\_MOON = 0x16;

Code Example

//Register the callback listener first

BLEManager.registerAppSendDataCallBack(new AppSendDataCallBack.ICallBack(){

@Override

Public void onSuccess(DataType type){

if (settingType == AppSendDataCallBack.DataType.WEATHER) {

//Callback after successful setting, SettingType, data returned

}

}

})

WeatherInfoV3 weatherInfoV3 = new WeatherInfoV3();

。。。。

//Call Interface

BLEManager.setWeatherData(weatherInfoV3 );

### Set SOS

**Interface name:**

BLEManager.setOneKeySOSSwitch(boolean isOpen)

**Parameter Description:**

|  |  |
| --- | --- |
| isOpen | true is on, false is off |

Code Example:

//Register the callback listener first

BLEManager.registerSettingCallBack(new SettingCallBack.ICallBack(){

@Override

Public void onSuccess(SettingType type, Object data){

if (settingType == SettingCallBack.SettingType.ONE\_KEY\_SOS) {

//Callback after successful setting, SettingType, data returned

}

}

})

BLEManager.setOneKeySOSSwitch(true);

### Set Walk Reminder

1. Get walk reminder state

*/\*\*  
 \* Get walk reminder state  
 \*  
 \** ***@return*** *\*/*public WalkReminder getWalkReminderState() {  
 WalkReminder walkReminder = LocalDataManager.*getWalkReminder*();  
 if (walkReminder == null) {  
 walkReminder = new WalkReminder();  
 walkReminder.setOnOff(WalkReminder.*OFF*);  
 walkReminder.setWeeks(new boolean[]{true, true, true, true, true, true, true});  
 walkReminder.setStartHour(9);  
 walkReminder.setStartMinute(0);  
 walkReminder.setEndHour(21);  
 walkReminder.setEndMinute(0);  
 walkReminder.setGoalStep(Constants.*WALK\_REMINDER\_GOAL\_DEFAULT*);//Default hourly goal, user can define by themselves  
 }  
 return walkReminder;  
}

1. Set walk reminder

**Interface name:**

BLEManager.setWalkReminder(WalkReminder walkReminder)

**Parameter Description:**

|  |  |
| --- | --- |
| WalkReminder | |
| onOff | On-off |
| goalStep | Step goal |
| startHour | Start time: Hour |
| startMinute | Start time: Minute |
| endHour | End time: Hour |
| endMinute | End time: Minute |
| goalTime | Goal time for walk reminder (Unit: hour) |
| weeks | [An array](https://cn.bing.com/dict/search?q=An array&FORM=BDVSP6&cc=cn) of [Boolean](https://cn.bing.com/dict/search?q=Boolean&FORM=BDVSP6&cc=cn) [values](https://cn.bing.com/dict/search?q=values&FORM=BDVSP6&cc=cn): Week repeats, seven booleans representing Monday to Sunday |
| notifyFlag | //Notification type: 0 invalid; 1: Allow notifications; 2: silent notification; 3:Turn off notifications |

Code Example:

//Register the callback listener first

BLEManager.registerSettingCallBack(new SettingCallBack.ICallBack(){

@Override

Public void onSuccess(SettingType type, Object data){

if (settingType == SettingCallBack.SettingType.WALK\_REMINDER) {

//Callback after successful setting, SettingType, data returned

}

}

})

WalkReminder walkReminder = getWalkReminderState()//The state obtained above, the following is the user modified parameter settings

walkReminder .setStartMinute(startMinute);

walkReminder .setEndHour(endHour);

walkReminder .setEndMinute(endMinute);

walkReminder .setGoalStep(goalStep);

walkReminder .setOnOff(onOff);

walkReminder .setWeeks(weeks);

//Call Interface

BLEManager.setWalkReminder(walkReminder );

### Set Menstrual Remind

**Interface name:**

BLEManager.setMenstrualRemind(MenstrualRemind menstrualRemind)

**Parameter Description:**

|  |  |
| --- | --- |
| MenstrualRemind | |
| start\_day | Start day reminder Days in advance |
| ovulation\_day | Ovulation day reminder Days in advance |
| hour | Reminder time: Hour |
| minute | Reminder time: Minute |
| pregnancy\_day\_before\_remind | Pregnancy day Start time How many days in advance to remind |
| pregnancy\_day\_end\_remind | Pregnancy day End time How many days in advance to remind |
| menstrual\_day\_end\_remind | Menstrual day end How many days in advance to remind |

Code Example:

//Register the callback listener first

BLEManager.registerSettingCallBack(new OtherProtocolCallBack.ICallBack(){

@Override

Public void onSuccess(OtherProtocolCallBack.SettingType type){

if (settingType == SettingCallBack.SettingType.ALARM) {

//Callback after successful setting, SettingType, data returned

}

}

})

MenstrualRemind menstrualRemind = new MenstrualRemind();

menstrualRemind.start\_day = 3;

menstrualRemind.ovulation\_day = 3;

menstrualRemind.hour = 9;

menstrualRemind.minute = 30;

menstrualRemind.pregnancy\_day\_before\_remind = 5;

menstrualRemind.pregnancy\_day\_end\_remind = 2;

menstrualRemind.menstrual\_day\_end\_remind = 1;

//Call Interface

BLEManager.setMenstrualRemind(menstrualRemind);

### Set Motion Intelligent Recognition

**Interface name:**

BLEManager.setActivitySwitch(activitySwitch);

**Parameter Description:**

|  |  |
| --- | --- |
| ActivitySwitch | |
| autoIdentifySportWalk | Automatic sport recognition walking |
| autoIdentifySportRun | Automatic sport recognition running |
| SWITCH\_ON | On |
| SWITCH\_OFF | Off |

1. getState

public ActivitySwitch getMotionRecognitionState() {  
 ActivitySwitch activitySwitch = LocalDataManager.*getActivitySwitch*();  
 return activitySwitch == null ? new ActivitySwitch() : activitySwitch;  
}

1. send to device

example:

//Register the callback listener first

BLEManager.registerSettingCallBack(new OtherProtocolCallBack.ICallBack(){

@Override

Public void onSuccess(OtherProtocolCallBack.SettingType type){

if (settingType == SettingCallBack.SettingType.ACTIVITY\_SWITCH) {

//Callback after successful setting

}

}

})

ActivitySwitch mMotionRecognitionState = getMotionRecognitionState();

mMotionRecognitionState .autoIdentifySportRun = ActivitySwitch .SWITCH\_ON

BLEManager.setActivitySwitch(autoIdentifySportRun );

### Set Heart Rate v3 Protocol

**Interface name:**

BLEManager.setHeartRateMeasureModeV3(heartRateMode);

HeartRateMeasureModeV3 heartRateMode;

BLEManager.registerSettingCallBack(new SettingCallBack.ICallBack(){

@Override

public void onSuccess(SettingCallBack.SettingType type, Object returnData) {

heartRateMode= (HeartRateMeasureModeV3) o;

}

@Override

public void onFailed(SettingCallBack.SettingType type) {

}})

1.HeartRateMeasureModeV3 modeV3 = new HeartRateMeasureModeV3();

modeV3.updateTime = 0;

BLEManager.setHeartRateMeasureModeV3(modeV3);

Call it like this first, you can get the heartRateMode returned by the firmware

2.

heartRateMode.measurementInterval = 5;

heartRateMode.startHour = 9;

heartRateMode.startMinute = 30;

heartRateMode.endHour = 18;

heartRateMode.endMinute = 0;

heartRateMode.updateTime = (int) System.currentTimeMillis();

heartRateMode.mode = HeartRateMeasureModeV3.MODE\_CUSTOM;

BLEManager.setHeartRateMeasureModeV3(heartRateMode);

Here you can modify specific parameters

### Set PRESSURE on\_off

**Interface name:**

BLEManager.setPressureParam(pressureParam);

PressureParam Object Description:

|  |  |
| --- | --- |
| PressureParam | |
| onOff | Main switch 0xAA on, 0x55 off |
| startHour | //The switch time and the reminder time are the same |
| startMinute | Start time: Minute |
| endHour | End time: Hour |
| endMinute | End time: Minute |
| remind\_on\_off | //Pressure reminder switch 0xAA on, 0x55 off, on\_off is off, the reminder does not work |
| interval | //Reminder interval, in minutes, the default is 60 minutes |
| notifyFlag | //Notification type: 0 is invalid; 1: allow notifications; 2: silent notifications; 3: turn off notifications |
| repeat | /Reserved bit0 is invalid, bit1-bit7 are week 1 to week 7 respectively, the same as the alarm clock |

example:

BLEManager.registerOtherProtocolCallBack(mOtherSettingCallback);

private final OtherProtocolCallBack.ICallBack mOtherSettingCallback = new OtherProtocolCallBack.ICallBack() {

@Override

public void onSuccess(OtherProtocolCallBack.SettingType settingType) {

if (settingType == OtherProtocolCallBack.SettingType.PRESSURE) {

//set successfully

}

}

@Override

public void onFailed(OtherProtocolCallBack.SettingType settingType) {

}

};

BLEManager.setPressureParam(pressureParam);

### Set Menu List

**Interface name:**

BLEManager.setMenuList(MenuList menuList)

Code Example:

//Register the callback listener first

BLEManager.registerSettingCallBack(new OtherProtocolCallBack.ICallBack(){

@Override

Public void onSuccess(OtherProtocolCallBack.SettingType type){

if (settingType == SettingCallBack.SettingType.MENU\_LIST\_SET) {

//Callback after successful setting, SettingType, data returned

}

}

})

MenuList menuList = new MenuList();

menuList.items.add(MenuList.MENU\_ALARM);

menuList.items.add(MenuList.MENU\_PRESSURE);//call interface

BLEManager.setMenuList(menuList);

### Frequent Contacts

**Interface name:**

1. Get Frequent Contacts:

BLEManager.queryFrequentContactsV3()

1. Set Frequent Contacts:

BLEManager.deleteFrequentContactsV3(List<FrequentContactsV3> frequentContactsV3List)

//Register the callback listener first

OperateCallBack.ICallBack callBack = new OperateCallBack.ICallBack() {

@Override

public void onSetResult(OperateCallBack.OperateType type, boolean isSuccess) {

//set result

}

@Override

public void onAddResult(OperateCallBack.OperateType type, boolean isSuccess) {

}

@Override

public void onDeleteResult(OperateCallBack.OperateType type, boolean isSuccess) {

}

@Override

public void onModifyResult(OperateCallBack.OperateType type, boolean isSuccess) {

}

@Override

public void onQueryResult(OperateCallBack.OperateType type, Object returnData) {

}

}

BLEManager.registerOperateCallBack(mOperateCallback);

BLEManager.unregisterOperateCallBack(mOperateCallback);（ondestroy call）

FrequentContactsV3 ：

/\*\*

\*contact name

\*/

public String name;

/\*\*

\*contact's phone

\*/

public String phone;

### Set Call Reminder Switch

1. First, check whether the device supports Bluetooth calling according to the function table

SupportFunctionInfo functionInfo = LocalDataManager.getSupportFunctionInfo();

functionInfo.V3\_support\_sync\_contact Return true represent support

**Interface name:**

BLEManager.setNoticeReminderSwitchStatus(status);

Code Example:

//Register the callback listener first

BLEManager.registerSettingCallBack(new SettingCallBack.ICallBack(){

@Override

Public void onSuccess(SettingCallBack.SettingType type){

if (settingType == SettingCallBack.SettingType.NOTICE\_REMINDER\_SWITCH\_STATUS) {

//Callback after successful setting, SettingType, data returned

}

}

})

boolean switch;

NoticeReminderSwitchStatus status = new NoticeReminderSwitchStatus();

status.notify\_switch = NoticeReminderSwitchStatus.NOTIFY\_88;

if (switch) {

status.call\_switch = NoticeReminderSwitchStatus.SWITCH\_ON;

} else {

status.call\_switch = NoticeReminderSwitchStatus.SWITCH\_OFF;

}

BLEManager.setNoticeReminderSwitchStatus(status);

### Set Schedule Reminder

1. First, check whether the device supports schedule reminder according to the function table

SupportFunctionInfo functionInfo = LocalDataManager.getSupportFunctionInfo();

functionInfo.V3\_schedule\_reminder Return true represent support

**Interface name:**

1. Query Schedule Reminder

BLEManager.queryScheduleReminderV3()

1. Delete Schedule Reminder

BLEManager.deleteScheduleReminderV3(List<ScheduleReminderV3> scheduleReminderV3List)

1. Add Schedule Reminder

BLEManager.addScheduleReminderV3(List<ScheduleReminderV3> scheduleReminderV3List)

Code Example:

//Register the callback listener first

//Register the callback listener first

OperateCallBack.ICallBack callBack = new OperateCallBack.ICallBack() {

@Override

public void onSetResult(OperateCallBack.OperateType type, boolean isSuccess) {

//set result

}

@Override

public void onAddResult(OperateCallBack.OperateType type, boolean isSuccess) {

}

@Override

public void onDeleteResult(OperateCallBack.OperateType type, boolean isSuccess) {

}

@Override

public void onModifyResult(OperateCallBack.OperateType type, boolean isSuccess) {

}

@Override

public void onQueryResult(OperateCallBack.OperateType type, Object returnData) {

}

}

BLEManager.registerOperateCallBack(mOperateCallback);

BLEManager.unregisterOperateCallBack(mOperateCallback);（ondestroy call）

### Set World Time

**Interface name:**

BLEManager.setWorldTime(List<WorldTime.Item> itemList)

//Register the callback listener first

BLEManager.registerSettingCallBack(new SettingCallBack.ICallBack(){

@Override

Public void onSuccess(SettingType type, Object data){

if (settingType == SettingCallBack.SettingType.WORLD\_TIME) {

}

}

})

example:

List<WorldTime.Item> list = new ArrayList<>();

WorldTime.Item item = new WorldTime.Item();

item.city\_name = "Beijing";

item.id = 31;

item.min\_offset = 480;

item.sunset\_hour = 4;

item.sunrise\_min = 45;

item.sunset\_hour = 19;

item.sunrise\_min = 44;

item.latitude\_flag = 1;

item.latitude = 3390;// 33.90 \*100

item.longitude\_flag = 1;

item.longitude = 11641; // 116.41\*100

list.add(item);

BLECmdUtils.setWorldTime(itemList);

Item Description：

public int id;

public int min\_offset;

public String city\_name; //city name

public int sunrise\_hour; //sunrise

public int sunrise\_min;

public int sunset\_hour; //sunset

public int sunset\_min;

public int longitude\_flag; // 1: East longitude; 2: West longitude

public int longitude; //Longitude Keep 2 decimal places and expand it by 100 times. The app needs to convert the corresponding points into degrees

public int latitude\_flag; // 1: north latitude； 2：south latitude

public int latitude; //[Latitude](https://cn.bing.com/dict/search?q=latitude&FORM=BDVSP6&cc=cn) Keep 2 decimal places and expand it by 100 times. The app needs to convert the corresponding points into degrees

### Set Night Brightness

**Interface name:**

BLEManager.setSCreenBrightnessConfig(screenBrightness);

//Register the callback listener first

BLEManager.registerSettingCallBack(new SettingCallBack.ICallBack(){

@Override

Public void onSuccess(SettingType type, Object data){

if (settingType == SettingCallBack.SettingType.SCREEN\_BRIGHTNESS) {

}

}

})

ScreenBrightness brights = new ScreenBrightness();

brights.level = 60;

brights.mode = 2;

brights.autoAdjustNight = 3;

brights.startHour = 19;

brights.startMinute = 0;

brights.endHour = 6;

brights.endMinute = 0;

brights.nightLevel = 0;

brights.showInterval = 5;

BLEManager.setSCreenBrightnessConfig(brights);

ScreenBrightness level description:

public int opera = OPERA\_TYPE\_USER;//0x0 automatic, 0x01 manual (if it is automatic synchronization configuration, please send 00, if it is user adjustment, please send 01)

public int level = 100;

public int mode = MODE\_ADJUST\_OFF; //0x00 Turn off automatic adjustment and use the value of level; 0x01 uses the ambient light sensor; 0x02 is used internally by the sdk when synchronizing configuration

public int autoAdjustNight = AUTO\_ADJUST\_NIGHT\_TYPE\_DEVICE\_DEFAULT\_TIME\_RANGE; //0x00 Invalid Defined by firmware 0x01 Disable 0x02 Use the preset time of the wristband to reduce the brightness at night 0x03 Use the set time to reduce the brightness at night

public int startHour;

public int startMinute;

public int endHour;

public int endMinute;

public int showInterval;//show interval

### Night temperature set

1. First judge whether the skin temperature function is supported according to the watch function table：

SupportFunctionInfo functionInfo = LocalDataManager.getSupportFunctionInfo();

functionInfo.V3\_support\_set\_temperature\_switch return ture means support

**Interface name：**

BLEManager.setNightTemperatureMonitoringPara(param);

// Please register call back and monitor

BLEManager.registerSettingCallBack(new SettingCallBack.ICallBack(){

@Override

Public void onSuccess(SettingType type, Object data){

if (settingType == SettingCallBack.SettingType.NIGHT\_TEMPERATURE\_MONITORING) {

}

}

})

example:

NightTemperatureMonitoringPara para = new NightTemperatureMonitoringPara(); param.mode = NightTemperatureMonitoringPara.SWITCH\_OFF;

param.unit=NightTemperatureMonitoringPara.UNIT\_F; BLEManager.setNightTemperatureMonitoringPara (para);

### Blood Oxygen set

1. First judge whether the automatic blood oxygen detection function is supported according to the function table:

SupportFunctionInfo functionInfo = LocalDataManager.getSupportFunctionInfo();

functionInfo.V3\_support\_set\_spo2\_all\_day\_on\_off returns true to represent support

**Interface name:**

BLEManager.setSPO2Param(param);

//register callback listener first

BLEManager.registerOtherProtocolCallBack(new OtherProtocolCallBack.ICallBack() {

@Override

public void onSuccess(OtherProtocolCallBack.SettingType settingType) {

if (settingType == SettingCallBack.SettingType.SPO2) {

}

}

@Override

public void onFailed(OtherProtocolCallBack.SettingType settingType) {

}

});

example:

SPO2Param spO2Param = spO2Param = new SPO2Param();

spO2Param.startHour = 0;

spO2Param.startMinute = 0;

spO2Param.endHour = 23;

spO2Param.endMinute = 59;

spO2Param.lowSpo2OnOff = SPO2Param.STATE\_OFF;

spO2Param.onOff = SPO2Param.STATE\_OFF;

spO2Param.lowSpo2OnValue = 85;

BLEManager.setSPO2Param(spO2Param )

### Set distance calorie goal

**Interface name:**

BLEManager.setCalorieAndDistanceGoal(CalorieAndDistanceGoal calorieAndDistanceGoal)

**Code example:**

//register callback listener first

//register callback listener first

BLEManager.registerOtherProtocolCallBack(new OtherProtocolCallBack.ICallBack() {

@Override

public void onSuccess(OtherProtocolCallBack.SettingType settingType) {

if(settingType==SettingCallBack.SettingType.CALORIE\_DISTANCE\_GOAL) {

}

}

@Override

public void onFailed(OtherProtocolCallBack.SettingType settingType) {

}

});

//call the interface

CalorieAndDistanceGoal calorieAndDistanceGoal = new CalorieAndDistanceGoal();

calorieAndDistanceGoal .calorie = 100;

calorieAndDistanceGoal .distance = 100;

BLEManager.setCalorieAndDistanceGoal(calorieAndDistanceGoal);

CalorieAndDistanceGoal object description:

public int calorie; //calories

public int distance; //distance

public long mid\_high\_time\_goal; //Mid-high exercise duration goal setting

public int calorie\_min; //calorie minimum

public int calorie\_max; //calorie max

public int walk\_goal\_time; //Walk target time

### Set Hand Wear Mode

**Interface name:**

BLEManager.setHandWearMode(HandWearMode handWearMode)；

//register callback listener first

BLEManager.registerSettingCallBack(new SettingCallBack.ICallBack(){

@Override

Public void onSuccess(SettingType type, Object data){

if (settingType == SettingCallBack.SettingType.HAND\_MODE) {

}

}

})

example:

HandWearMode handWearMode = new HandWearMode();

handWearMode.hand = HandWearMode.HAND\_MODE\_LEFT;

BLEManager.setHandWearMode(handWearMode);

### Set Heart Rate Interval

**Interface name:**

BLEManager.setHeartRateInterval(HeartRateInterval interval);

//register callback listener first

BLEManager.registerSettingCallBack(new SettingCallBack.ICallBack(){

@Override

Public void onSuccess(SettingType type, Object data){

}

})

example:

HeartRateInterval heartRateInterval = heartRateInterval = new HeartRateInterval();

if (heartRateInterval == null) {

heartRateInterval = new HeartRateInterval();

}

heartRateInterval.setRemindStartHour(0);

heartRateInterval.setRemindStartMinute(0);

heartRateInterval.setRemindStopHour(23);

heartRateInterval.setRemindStopMinute(59);

heartRateInterval.setMinHr(20);

int maxValue = 220;

heartRateInterval.setUserMaxHR(maxValue);

//warm up

heartRateInterval.setWarmUpThreshold((int) (maxValue \* 0.5));

//fat burning

heartRateInterval.setBurnFatThreshold((int) (maxValue \* 0.6));

//Aerobic exercise

heartRateInterval.setAerobicThreshold((int) (maxValue \* 0.7));

//Anaerobic exercise

heartRateInterval.setAnaerobicThreshold((int) (maxValue \* 0.8));

//Extreme sport

heartRateInterval.setLimintThreshold((int) (maxValue \* 0.9));

BLEManager.setHeartRateInterval(heartRateInterval);

## Six. Watch Dial

### Get watch dial list

WatchPlateCallBack.IOperateCallBack iOperateCallBack = new WatchPlateCallBack.IOperateCallBack() {

@Override

public void onGetPlateFileInfo(WatchPlateFileInfo watchPlateFileInfo) {

if (watchPlateFileInfo == null || watchPlateFileInfo.fileNameList == null){

return;

}

tvOperateTv.setText(watchPlateFileInfo.toString()); //list of dials returned

}

@Override

public void onGetScreenInfo(WatchPlateScreenInfo screenInfo) {

tvOperateTv.setText(GsonUtil.toJson(screenInfo)); //Return screen information

}

@Override

public void onGetCurrentPlate(String uniqueID) {

tvOperateTv.setText("Currently used watch face：" + uniqueID);

}

@Override

public void onSetPlate(boolean isSuccess) {

tvOperateTv.setText(isSuccess ? "Setting succeeded" : "Setting failed");

}

@Override

public void onDeletePlate(boolean isSuccess) {

tvOperateTv.setText(isSuccess ? "Delete successful" : "Delete failed");

}

@Override

public void onGetDialPlateParam(DialPlateParam dialPlateParam) {

}

};

BLEManager.registerWatchOperateCallBack(iOperateCallBack);

if (getSupportFunctionInfo().V3\_get\_watch\_list\_new) { //According to the function dial, call different methods

saveDialLog("getInstalledDialInfo start, V3\_get\_watch\_list\_new");

BLEManager.getDialPlateParam();

} else {

saveDialLog("getInstalledDialInfo start");

BLEManager.getWatchPlateList();

}

### Wallpaper dial

**Setup process:**

**1.getWatchPlateScreenInfo(); Get the screen size**

**2. Select the picture, the size of the screenshot picture（according to the size inwatchPlateScreenInfo）**

**3. Make a picture**

**Wallpaper wallpaper = new Wallpaper();**

**wallpaper.setFileName(path);// where the cropped image is stored**

**wallpaper.setSaveFileName(outputpath);**

**wallpaper.setFormat(5);**

**String json = GsonUtil.toJson(wallpaper);**

**BLEManager.setParaToDeviceByTypeAndJson(5500, json);//Request to make a photo**

**4.Installation**

**BLEManager.startTranCommonFile(FileTransferConfig.getDefaultTransPictureConfig(outputpath.concat(".lz"),, new IFileTransferListener() {**

**@Override**

**public void onStart() { }**

**@Override**

**public void onProgress(int progress) { }**

**@Override**

**public void onSuccess() {**

**//Transfer successfully, set color parameters**

**5.BLEManager.photoWallpaperOperate(params)**

**}**

**@Override**

**public void onFailed(String errorMsg) { }**

**});**

**Interface name:**

BLEManager.createPlateWallpaperFile(WallpaperFileCreateConfig config);

BLEManager.startTranCommonFile(FileTransferConfig config);

**Code example:**

WallpaperFileCreateConfig config = new WallpaperFileCreateConfig();

//App crops the image selected by the user, and generates the original PNG image address

config.setSourceFilePath("/asdcard/demo/test.png");

//After sdk processing, a new wallpaper file will be output

config.setOutFilePath("/adcard1/demo/outFile.png");

//Start making wallpaper file

BLEManager.createPlateWallpaperFile(config);

//The production is complete, transfer the wallpaper file to the device

FileTransferConfig fileConfig = FileTransferConfig.getDefaultTransPictureConfig(config.getOutFilePath(), new IFileTransferListener() {

@Override

public void onStart() { }

@Override

public void onProgress(int progress) { }

@Override

public void onSuccess() { }

@Override

public void onFailed(String errorMsg) { }

});

BLEManager.stopTranCommonFile();

BLEManager.startTranCommonFile(fileConfig);

### Cloud watch dial

**Interface name:**

BLEManager.startSetPlateFileToWatch(WatchPlateSetConfig config)

**Parameter Description:**

|  |  |
| --- | --- |
| **WatchPlateSetConfig** | |
| filePath | dial zip package path |
| uniqueID | Unique ID of watch face , unified App - server - watch |
| isOnlyTranslateWatchFile | Is it just only transfer watch face |
| stateListener | Status monitoring |
| maxRetryTimes | maximum number of retries |
|  |  |
|  |  |

**Code example:**

WatchPlateSetConfig config = new WatchPlateSetConfig();

File file = new File(filePath);

config.uniqueID = file.getName().replace(".zip", "");

config.filePath = filePath;

config.stateListener = new WatchPlateCallBack.IAutoSetPlateCallBack() {

@Override

public void onStart() {

tvState.setText("start");

}

@Override

public void onProgress(int progress) {

tvState.setText("progress = " + progress);

}

@Override

public void onSuccess() {

tvState.setText("success");

}

@Override

public void onFailed() {

tvState.setText("failed");

}

};

BLEManager.startSetPlateFileToWatch(config);

Note: When installing the dial, make sure that there is no dial with the same name in the watch. If there is, you need to delete it before reinstalling

### Set wallpaper watch face color

Register listener:

BLEManager.registerPhotoWallpaperOperateCallBack(new PhotoWallpaperOperateCallBack.ICallBack() {

@Override

public void onOperateResult(PhotoWallpaperOperation.ResponseInfo responseInfo) {

}

});

/\*\*

\* Initialize wallpaper dial color collection

\*/

public List<String> getWallpaperColorList() {

if (mColorList == null) {

mColorList = new ArrayList<>();

}

if (mColorList.isEmpty()) {

mColorList.add("#F2F2F2");

mColorList.add("#000000");

mColorList.add("#FC1E58");

mColorList.add("#FF9501");

mColorList.add("#0091FF");

mColorList.add("#44D7B6");

}

return mColorList;

}

Start setting colors:

PhotoWallpaperOperation.SetParams params = new PhotoWallpaperOperation.SetParams();

params.operate = PhotoWallpaperOperation.SetParams.OPERATE\_SET;

Integer color =Integer.parseInt(getWallpaperColorList().get(colorIndex).replace("#",""),16);

params.time\_color = color;

params.widget\_icon\_color = color;

params.widget\_num\_color = color;

BLEManager.photoWallpaperOperate(params);

### Delete watch dial

BLEManager.deleteWatchPlate（String name）;

name: the name of the watch face, the watch face name returned by the firmware, for example: watcht.iwf must bring .iwf

### 6. Other watch dial operation

BLEManager.getCurrentWatchPlate();

BLEManager.getWatchPlateScreenInfo();

The callback is the same as before，BLEManager.registerWatchOperateCallBack(iOperateCallBack);

## Seven. Upgrade OTA

### Nordic platform upgrade

（1）The service needs to be registered in the AndroidManifest.xml file:

<service

android:name="com.realsil.sdk.dfu.DfuService"

android:enabled="true"

android:exported="false" />

<service

android:name="com.ido.ble.dfu.DFUService"

android:enabled="true"

android:exported="false"

tools:ignore="Instantiatable" />

1. libs needs to depend on dfu.aar

**Interface name:**

|  |  |
| --- | --- |
| BLEManager.startDFU(BleDFUConfig dfuConfig) | start the upgrade |
| BLEManager.cancelDFU() | Cancel upgrade |
| BLEManager.addDFUStateListener(BleDFUState.IListener iListener) | Monitor upgrade status |

**Parameter Description:**

|  |  |  |  |
| --- | --- | --- | --- |
| **BleDFUConfig** | | | |
| Field Name | Is it necessary | illustrate |
| filePath | Yes | ota file path |
| macAddresss | Yes | The mac address of the device |
| deviceID | Yes | Device ID |
| PRN | No |  |
| isNeedReOpenBluetoothSwitchIfFailed | No | Do you need to restart bluetooth when retrying on failure |
|  |  |  |
|  |  |  |

|  |  |
| --- | --- |
| **BleDFUState.IListener** | |
| onPrepare() | start the upgrade |
| onProgress(int progress) | Upgrade progress |
| onSuccess() | update successed |
| onFailed(FailReason failReason) | upgrade unsuccessful |
| onCanceled() | Upgrade canceled |
|  |  |

**example：**

BleDFUState.IListener mDFUStateListener = new BleDFUState.IListener() {

@Override

public void onPrepare() {

}

@Override

public void onDeviceInDFUMode() {

}

@Override

public void onProgress(int i) {

}

@Override

public void onSuccess() {

}

@Override

public void onSuccessAndNeedToPromptUser() {

}

@Override

public void onFailed(BleDFUState.FailReason failReason) {

}

@Override

public void onCanceled() {

}

@Override

public void onRetry(int i) {

}

};

BleDFUConfig config = new BleDFUConfig();

config.setDeviceId(String.valueOf(mDeviceId));//Device id，the basic information of a device include

config.setMacAddress(mDeviceAddress);// device mac

config.setFilePath(mFirmwareFilePath);//file path

config.setPlatform(BleDFUConfig.PLATFORM\_NORDIC);

config.setMaxRetryTime(3);//Retry times

BLEManager.addDFUStateListener(mDFUStateListener);

BLEManager.startDFU(config);

### Apollo platform upgrade, file transfer

To transfer the file way could transfer the file

example:

IFileTransferListener mIFileTransferListener = new IFileTransferListener() {

@Override

public void onStart(){};

@Override

public void onProgress(int progress){};

@Override

public void onSuccess(){};

@Override

public void onFailed(String errorMsg){};

}

FileTransferConfig config = FileTransferConfig.getDefaultApolloOTAConfig(updatePath, mIFileTransferListener);//updatePath The path where the upgrade file is saved

BLEManager.stopTranCommonFile();

BLEManager.startTranCommonFile(config);

### 3. Realtek platform upgrade

**Interface name：**

|  |  |
| --- | --- |
| BLEManager.startDFU(BleDFUConfig dfuConfig) | Start to upgrade |
| BLEManager.cancelDFU() | Cancel upgrade |
| BLEManager.addDFUStateListener(BleDFUState.IListener iListener) | Monitor upgrade status |

**Parameter instructions：**

|  |  |  |  |
| --- | --- | --- | --- |
| **BleDFUConfig** | | | |
| Fileld name | If it is necessary | illustrate |
| filePath | Yes | Ota file path |
| macAddresss | Yes | Device mac address |
| deviceID | Yes | Device ID |
| PRN | No |  |
| isNeedReOpenBluetoothSwitchIfFailed | No | Do you need to restart bluetooth when retrying failed |
| setPlatform | Yes | Value：BleDFUConfig.PLATFORM\_REALTEK （platform setting） |
|  |  |  |

|  |  |
| --- | --- |
| **BleDFUState.IListener** | |
| onPrepare() | Start upgrade |
| onProgress(int progress) | Upgrade process |
| onSuccess() | Upgrade success |
| onFailed(FailReason failReason) | Upgrade fail |
| onCanceled() | Cancel upgrade |
|  |  |

example:

**example：**

BleDFUState.IListener mDFUStateListener = new BleDFUState.IListener() {

@Override

public void onPrepare() {

}

@Override

public void onDeviceInDFUMode() {

}

@Override

public void onProgress(int i) {

}

@Override

public void onSuccess() {

}

@Override

public void onSuccessAndNeedToPromptUser() {

}

@Override

public void onFailed(BleDFUState.FailReason failReason) {

}

@Override

public void onCanceled() {

}

@Override

public void onRetry(int i) {

}

};

BleDFUConfig config = new BleDFUConfig();

config.setDeviceId(String.valueOf(mDeviceId));//Device ID ,It's in the basic information of the device

config.setMacAddress(mDeviceAddress);// MAC address of the device

config.setFilePath(mFirmwareFilePath);// File path

config.setMaxRetryTime(3);//Number of retries

BLEManager.addDFUStateListener(mDFUStateListener);

BLEManager.startDFU(config);

## Eight. Multi-Sport data exchange

**Ⅰ. V2 agreement：(All watches need to be plugged in)**

1.Register to monitor

AppExchangeDataCallBack.ICallBack iCallBack = new AppExchangeDataCallBack.ICallBack() {

@Override //Bracelet reply: The "Start" command sent by App

public void onReplyExchangeDataStart(AppExchangeDataStartDeviceReplyData data) {

}

@Override //Bracelet reply: The "send data" command sent by App

public void onReplyExchangeDateIng(AppExchangeDataIngDeviceReplyData data) {

}

@Override //Bracelet reply: The "stop" command sent by App

public void onReplyExchangeDateStop(AppExchangeDataStopDeviceReplyData data) {

}

@Override //Bracelet reply: The "pause" command sent by App

public void onReplyExchangeDatePause(AppExchangeDataPauseDeviceReplyData data) {

}

@Override // Bracelet reply: The "restore" command sent by App

public void onReplyExchangeDateResume(AppExchangeDataResumeDeviceReplyData data) {

}

@Override // App receives the "stop" command initiated by the bracelet

public void onDeviceNoticeAppStop(DeviceNoticeAppExchangeDataStopPara para) {

}

@Override // App receives the "pause" command initiated by the bracelet

public void onDeviceNoticeAppPause(DeviceNoticeAppExchangeDataPausePara para) {

}

@Override // App receives the "restore" command initiated by the bracelet

public void onDeviceNoticeAppResume(DeviceNoticeAppExchangeDataResumePara para) {

}

};

**BLEManager.registerAppExchangeDataCallBack(iCallBack);**

2、Instruction type of data exchange initiated by App:

(1)start campaign：

Interface：

BLEManager.appExchangeDataStart(para);

example:

AppExchangeDataStartPara para = new AppExchangeDataStartPara();

Para This parameter requires setting time、movement type and other parameters，It depends on the needs，for example：

para.day = mDay =Calendar.getInstance().get(Calendar.DAY\_OF\_MONTH);

para.hour = mHour = Calendar.getInstance().get(Calendar.HOUR\_OF\_DAY);

para.minute = mMin = Calendar.getInstance().get(Calendar.MINUTE);

para.second = mSecond = Calendar.getInstance().get(Calendar.SECOND);

para.sportType = AppExchangeDataStartPara.SPORT\_TYPE\_DANCING;

para.target\_type = AppExchangeDataStartPara.TARGET\_TYPE\_DURATIONS;

para.target\_value = 1;

para.force\_start = AppExchangeDataStartPara.FORCE\_START\_VALID;（Just example）

Set if you want to force start：

para.force\_start = AppExchangeDataStartPara.FORCE\_START\_VALID;

Set if you don’t want to force start：

para.force\_start = AppExchangeDataStartPara.FORCE\_START\_INVALID;

After setting the PARA parameters，use the following code is called to begin exchanging data：

BLEManager.appExchangeDataStart(para);

AppExchangeDataStartDeviceReplyData Five states have been defined：

public static final int CODE\_SUCCESS = 0x00;//Succeed

public static final int CODE\_FAILED\_DEVICE\_ALREADY\_IN\_SPORT\_MODE = 0x01;//fail , Because the device is in motion mode

public static final int CODE\_FAILED\_DEVICE\_LOW\_BATTARY = 0x02;//fail , because the battery is low

public static final int CODE\_FAILED\_DEVICE\_CHARGING = 0x03;//In the charging

public static final int CODE\_FAILED\_DEVICE\_IN\_VOICE\_STATE = 0x04;//In alexa speech recognition

public int ret\_code;//Return status code

1. Transmit Data, swap data：Used to send distance, time to the watch (IDO APP send every 2 seconds，can refer to )

Interface：

BLEManager.appExchangeDataIng(para);

example:

AppExchangeDataIngPara para = new AppExchangeDataIngPara();

para.duration= 20; //movement time

para.distance= 100; // movement distance ，App is calculated according to latitude and longitude，Unit: meter

BLEManager.appExchangeDataIng(para);

1. Suspend movement：

Interface：

BLEManager.appExchangeDataPause(para);

example:

AppExchangeDataIngPara para = new AppExchangeDataIngPara();

para.day = mDay ;

para.hour = mHour ;

para.minute = mMin ;

para.second = mSecond ;

BLEManager.appExchangeDataPause(para);

1. **Resuming sport：**

Interface**：**

**BLEManager.appExchangeDataResume(para);**

example:

AppExchangeDataIngPara para = new AppExchangeDataIngPara();

para.day = mDay ;

para.hour = mHour ;

para.minute = mMin ;

para.second = mSecond ;

BLEManager.appExchangeDataResume(para);

1. **End a movement：**

Interface**：**

**BLEManager.appExchangeDataStop(para);**

example:

AppExchangeDataIngPara para = new AppExchangeDataIngPara();

para.day = mDay ;

para.hour = mHour ;

para.minute = mMin ;

para.second = mSecond ;

para.duration= 20; //movement time

para.distance= 100; // movement distance ，App is calculated according to latitude and longitude，Unit: meter

BLEManager.appExchangeDataStop(para);

Notice:

①The PARE parameters for sending data, pausing, resuming, and ending should be set to the same time as when the start instruction was initiated.

**②After the start, send, pause, resume and end commands initiated by the app to the watch are successful, the APP will receive the callback method returned by the watch ，The developer can then handle the UI display and business logic in the corresponding callback method. Here's how the callback methods correspond：**

**Start ---> onReplyExchangeDataStart(AppExchangeDataStartDeviceReplyData data)**

**Transmit Data ---> onReplyExchangeDateIng(AppExchangeDataIngDeviceReplyData data)**

**Pause ---> onReplyExchangeDatePause(AppExchangeDataPauseDeviceReplyData data)**

**Restore ---> onReplyExchangeDateResume(AppExchangeDataResumeDeviceReplyData data)**

**Finish ---> onReplyExchangeDateStop(AppExchangeDataStopDeviceReplyData data)**

3、Destroying listening files：

Destroy the listening files in the Activity's on Destroy() method：

BLEManager.unregisterAppExchangeDataCallBack(iCallBack);

**V3 protocol：**

SupportFunctionInfo functionInfo = LocalDataManager.getSupportFunctionInfo();

functionInfo.ex\_table\_main9\_v3\_activity\_exchange\_data

**If true is returned, V3 is supported，the following V3 protocols are required for data exchange，the v2 protocol above is also required（Motion start, pause and so on instructions are v2 protocol）**

1. Registering Listener：

BLEManager.registerV3AppExchangeDataCallBack(new V3AppExchangeDataCallBack.ICallBack() {

@Override

public void onReplyExchangeDateIng(V3AppExchangeDataIngDeviceReplyData v3AppExchangeDataIngDeviceReplyData) {

//Number of steps, distance, duration and other data

}

@Override

public void onReplyExchangeDataEndData(V3AppExchangeDataDeviceReplayEndData v3AppExchangeDataDeviceReplayEndData) {

// BLEManager.v3getEndActivityData(); Callback of data when invoked

}

@Override

public void onReplyExchangeHeartRateData(V3AppExchangeDataHeartRate v3AppExchangeDataHeartRate) {

//Heart rate data BLEManager.v3AppExchangeDataGetHeartRate(); The callback when invoked

}

});

2. V3 Data Interaction： The APP can send data (distance,etc.) to the device（ido app Send every 2 seconds）

BLEManager.v3AppExchangeDataIng(data)

example:

V3AppExchangeDataIngPara mV3SwitchDataAppIng = new V3AppExchangeDataIngPara();

mV3SwitchDataAppIng.distance =30; //App calculates distance based on latitude and longitude

BLEManager.v3AppExchangeDataIng(mV3SwitchDataAppIng );

3.

V3 protocol to obtain motion data details：（ido app use every 40 seconds）

BLEManager.v3getEndActivityData();

4 . V3 protocol to obtain heart rate data：（ido app use every 30 seconds）

BLEManager.v3AppExchangeDataGetHeartRate();

V3AppExchangeDataIngDeviceReplyData object strings：

public int type;

public int hour;

public int minute;

public int second;

public int heart\_rate;

public int distance;//distance, according to the set unit data display

public int real\_time\_speed;//The real-time speed Unit: km/h Expand 100 preach

public int real\_time\_speed\_pace;//Real time pace, s

public int km\_speed;//Real time km pace unit s/ km

public int real\_time\_calories;//Dynamic calories

public int steps;

public int swim\_posture;//The main swimming strokes,Use of swimming data

public int status;// status: Start 1, Manual to suspend 2, Finish 3, Automatically suspended 4 , 0 is an invalid state (4 is new add)

public int duration;//Time of duration

//New add，version = 3，Add the corresponding structure data

public int te; //Aerobic training effect grade Unit No scope 0-50 Expand transmission by 10 times.

public int tean; // Anaerobic exercise training effect grade Unit No scope 0-50 Expand transmission by 10 times.

//Motion countdown（Note: decreasing), Or timing at the end of the course (note: incremental) action\_type =1-5 decreasing, action\_type = 6 increasing

public int count\_hour;

public int count\_minute;

public int count\_second;

## Nine. Notifications (The app sends an alert to the watch)

Determine the protocol by using the following method:

SupportFunctionInfo functionInfo = LocalDataManager.getSupportFunctionInfo();

if (functionInfo.ex\_table\_main10\_v3\_notify\_msg ){

BLEManager.setV3MessageNotice(V3MessageNotice v3MessageNotice)

} else {

BLEManager.setNewMessageDetailInfo()

}

**1、The name of the interface：**

BLEManager.setNewMessageDetailInfo() v2 protocol

**Parameter description：**

|  |  |
| --- | --- |
| **NewMessageInfo** | |
| type | information type |
| name | name |
| content | content |
|  |  |
|  |  |
|  |  |

The value of type is already defined as a constant in the NewMessageInfo class, for example :

public static final int TYPE\_FACEBOOK = 0x06;

public static final int TYPE\_TWITTER = 0x07;

public static final int TYPE\_WHATSAPP = 0x08;

public static final int TYPE\_MESSENGER = 0x09;

public static final int TYPE\_INSTAGRAM = 0x0A;

public static final int TYPE\_LINKEDIN = 0x0B;

**Code examples：**

//Send notifications without registering callback listeners

NewMessageInfo newMessageInfo = new NewMessageInfo();

newMessageInfo.type = NewMessageInfo.TYPE\_FACEBOOK;

newMessageInfo.name = “name”;

newMessageInfo.content = “getNotifications”;

//invocation interface

BLEManager.setNewMessageDetailInfo(newMessageInfo);

V3 protocol：

**The name of the interface：**

BLEManager.setV3MessageNotice(V3MessageNotice v3MessageNotice) v3 protocol

**Parameter description：**

|  |  |
| --- | --- |
| **V3MessageNotice** | |
| evtType | information type |
| name | name |
| phoneNumber | phone number |
| dataText | content |
|  |  |

The value of type is already defined as a constant in the V3MessageNotice class, for example:

/\*\*

\* Telephone calls

\*/

public static final int TYPE\_CALL = 0x1000;

private static final int TYPE\_MSG\_BASE = 0x2000;

/\*\*

\* message

\*/

public static final int TYPE\_SMS = TYPE\_MSG\_BASE + 0x01;

/\*\*

\*mail

\*/

public static final int TYPE\_EMAIL = TYPE\_MSG\_BASE + 0x02;

/\*\*

\* wechat

\*/

public static final int TYPE\_WX = TYPE\_MSG\_BASE + 0x03;

public static final int TYPE\_QQ = TYPE\_MSG\_BASE + 0x04;

public static final int TYPE\_WEIBO = TYPE\_MSG\_BASE + 0x05;

public static final int TYPE\_FACEBOOK = TYPE\_MSG\_BASE + 0x06;

public static final int TYPE\_TWITTER = TYPE\_MSG\_BASE + 0x07;

public static final int TYPE\_WHATSAPP = TYPE\_MSG\_BASE + 0x08;

public static final int TYPE\_MESSENGER = TYPE\_MSG\_BASE + 0x09;

public static final int TYPE\_INSTAGRAM = TYPE\_MSG\_BASE + 0x0A;

public static final int TYPE\_LINKEDIN = TYPE\_MSG\_BASE + 0x0B;

public static final int TYPE\_CALENDAR = TYPE\_MSG\_BASE + 0x0C;

public static final int TYPE\_SKYPE = TYPE\_MSG\_BASE + 0x0D;

public static final int TYPE\_ALARM = TYPE\_MSG\_BASE + 0x0E;

public static final int TYPE\_VKONTAKTE = TYPE\_MSG\_BASE + 0x10;

public static final int TYPE\_LINE = TYPE\_MSG\_BASE + 0x11;

public static final int TYPE\_VIBER = TYPE\_MSG\_BASE + 0x12;

public static final int TYPE\_KAKAO\_TALK = TYPE\_MSG\_BASE + 0x13;

public static final int TYPE\_GMAIL = TYPE\_MSG\_BASE + 0x14;

public static final int TYPE\_OUTLOOK = TYPE\_MSG\_BASE + 0x15;

public static final int TYPE\_SNAPCHAT = TYPE\_MSG\_BASE + 0x16;

public static final int TYPE\_TELEGRAM = TYPE\_MSG\_BASE + 0x17;

public static final int TYPE\_OTHER = TYPE\_MSG\_BASE + 0X18;

public static final int TYPE\_CHATWORK = TYPE\_MSG\_BASE + 0x20;

public static final int TYPE\_SLACK = TYPE\_MSG\_BASE + 0x21;

public static final int TYPE\_MAIL\_YAHOO = TYPE\_MSG\_BASE + 0x22;

public static final int TYPE\_TUMBLR = TYPE\_MSG\_BASE + 0x23;

public static final int TYPE\_YOUTUBE = TYPE\_MSG\_BASE + 0x24;

public static final int TYPE\_PINTEREST\_YAHOO = TYPE\_MSG\_BASE + 0x25;

public static final int TYPE\_TIKTOK = TYPE\_MSG\_BASE + 0x26;

public static final int TYPE\_REDBUS = TYPE\_MSG\_BASE + 0X27;

public static final int TYPE\_DAILYHUNT= TYPE\_MSG\_BASE + 0X28;

public static final int TYPE\_HOTSTAR = TYPE\_MSG\_BASE + 0X29;

public static final int TYPE\_INSHORTS = TYPE\_MSG\_BASE + 0X2A;

public static final int TYPE\_PAYTM = TYPE\_MSG\_BASE + 0X2B;

public static final int TYPE\_AMAZON = TYPE\_MSG\_BASE + 0X2C;

public static final int TYPE\_FLIPKART = TYPE\_MSG\_BASE + 0X2D;

public static final int TYPE\_PRIME = TYPE\_MSG\_BASE + 0X2E;

public static final int TYPE\_NETFLIX = TYPE\_MSG\_BASE + 0X2F;

public static final int TYPE\_GPAY = TYPE\_MSG\_BASE + 0X30;

public static final int TYPE\_PHONPE = TYPE\_MSG\_BASE + 0X31;

public static final int TYPE\_SWIGGY = TYPE\_MSG\_BASE + 0X32;

public static final int TYPE\_ZOMATO = TYPE\_MSG\_BASE + 0X33;

public static final int TYPE\_MAKEMYTRIP = TYPE\_MSG\_BASE + 0X34;

public static final int TYPE\_JIOTV = TYPE\_MSG\_BASE + 0X35;

public static final int TYPE\_KEEP = TYPE\_MSG\_BASE + 0X36;

public static final int TYPE\_MICROSOFT = TYPE\_MSG\_BASE + 0X37;

public static final int TYPE\_WHATSAPP\_BUSINESS = TYPE\_MSG\_BASE + 0x38;

public static final int TYPE\_MISSED\_CALL = TYPE\_MSG\_BASE + 0X3A;

public static final int TYPE\_GPAP = TYPE\_MSG\_BASE + 0X3B;

public static final int TYPE\_YT\_MUSIC = TYPE\_MSG\_BASE + 0X3C;

public static final int TYPE\_UBER = TYPE\_MSG\_BASE + 0X3D;

public static final int TYPE\_OLA = TYPE\_MSG\_BASE + 0X3E;

public static final int TYPE\_MATTER = TYPE\_MSG\_BASE + 0X3F;

**Code Examples：**

//Send notifications without registering callback listeners

V3MessageNotice v3MessageNotice = new V3MessageNotice();

v3MessageNotice.evtType = V3MessageNotice.TYPE\_CALL;

v3MessageNotice.contact = contactName;

v3MessageNotice.phoneNumber = incomingNumber;

//invocation interface

BLEManager.setV3MessageNotice(v3MessageNotice);

## Ten. AGPS update

**1. Retrieving the gps status first**

**BLEManager.getGpsStatus()**

**Example of codes**

//Register GPS info callback

BLEManager.registerGetGpsInfoCallBack(new GetDeviceInfoCallBack.ICallBack(){

@Override

Public void onGetGpsStatus(GpsStatus gpsStatus){

//Return to GPS status here

//GPS is occupied status, no process triggered

if (GpsStatus.STATUS\_IDLE != gpsStatus.gps\_run\_status) {

return;

}

//Download Agps files

downloadAgpsFile();//Applied as the protocol

}

})

//Port Adjustment

BLEManager.getGpsStatus()

/\*\*

\* Download Agps files for processing

\*/

private void downloadAgpsFile() {

long timeMillis = System.currentTimeMillis();

//Available offline,over 24hrs after the latest offline upgrade，download offline files

if (isSupportOfflineUpgrade() && Math.abs(SPHelper.getLastAgpsOfflineUpgradeTime() - timeMillis) >= TIME\_MILLIONS\_OF\_DAY) {

String AGPS\_OFFLINE\_FILE\_URL = "http://offline-live1.services.u-blox.com/GetOfflineData.ashx?token=vB6zs0P4F0ayAYBMCzx4rw&gnss=gps,glo&period=1&resolution=1";//offline file url

//Download offline files

//**Attention: save the file as agps.ubx(as a MUST)**

}

//Available online,over 4hrs after the latest online upgrade，download online files

else if (isSupportOnlineUpgrade() && Math.abs(SPHelper.getLastAgpsOnlineUpgradeTime() - timeMillis) >= TIME\_MILLIONS\_OF\_4\_HOUR) {

String AGPS\_ONLINE\_FILE\_URL = "http://online-live1.services.u-blox.com/GetOnlineData.ashx?token=vB6zs0P4F0ayAYBMCzx4rw&gnss=gps,qzss,glo,bds,gal&datatype=eph&format=mga";//online file url

//download online files

//**Attention: save the file as online.ubx(as a MUST)**

}

}

/\*\*

\* whether or not support Agpe update

\*

\* @return

\*/

private boolean isSupportOfflineUpgrade() {

SupportFunctionInfo functionInfo = LocalDataManager.getSupportFunctionInfo();

return functionInfo != null && functionInfo.ex\_gps && functionInfo.agps\_offline;

}

/\*\*

\* whether or not support Agpe online update

\*

\* @return

\*/

private boolean isSupportOnlineUpgrade() {

SupportFunctionInfo functionInfo = LocalDataManager.getSupportFunctionInfo();

return functionInfo != null && functionInfo.ex\_gps && functionInfo.agps\_online; }

2.File Transferring(proceed the file transferring after downloading the gps files)

IFileTransferListener mIFileTransferListener = new IFileTransferListener() {

@Override

public void onStart(){};

@Override

public void onProgress(int progress){};

@Override

public void onSuccess(){

//Transferring complete

};

@Override

public void onFailed(String errorMsg){};

}

FileTransferConfig config = FileTransferConfig.getDefaultUbloxAGpsFileConfig(updatePath, mIFileTransferListener); //updatePath saving path of update files

config.firmwareSpecName = fileName; //fileName files name

config.maxRetryTimes = 0;

BLEManager.stopTranCommonFile();

BLEManager.startTranCommonFile(config);

## Eleven: Music Transferring

1.Registering music callback

OperateCallBack.IMusicCallBack callBack = new OperateCallBack.IMusicCallBack() {

@Override

public void onInvalid(OperateCallBack.OperateType type, boolean isSuccess) {

}

@Override

public void onDeleteMusic(OperateCallBack.OperateType type, boolean isSuccess) {

}

@Override

public void onAddMusic(OperateCallBack.OperateType type, boolean isSuccess, int musicId) {

// BLEManager.addMusicFile(musicFile) result

//isSuccess whether success or not musicId : the music ID returned to

if(isSuccess && musicId>0){

//music transferring

**transMusicFile();**

}

);

@Override

public void onDeleteFolder(OperateCallBack.OperateType type, boolean isSuccess) {

}

@Override

public void onAddFolder(OperateCallBack.OperateType type, boolean isSuccess) {

}

@Override

public void onModifyFolder(OperateCallBack.OperateType type, boolean isSuccess) {

}

@Override

public void onImportFolder(OperateCallBack.OperateType type, boolean isSuccess) {

}

@Override

public void onDeleteFolderMusic(OperateCallBack.OperateType type, boolean isSuccess) {

}

}

register operate music callback：

BLEManager.registerOperateMusicCallBack(callback)

unregister operate music callback：

BLEManager.unregisterOperateMusicCallBack(callback) (onDestory in the callback)

**transMusicFile()**{

SPPFileTransferConfig fileTransferConfig =

SPPFileTransferConfig.getDefaultMusicFileConfig(

music.filePath,

this

);

fileTransferConfig.firmwareSpecName = filename;

BLEManager.startSppTranFile(fileTransferConfig, new ISPPConnectStateListener() {

@Override

public void onStart() {

}

@Override

public void onSuccess() {

//Transferring complete

}

@Override

public void onFailed() {

}

@Override

public void onBreak() {

}

)

}

1. Related Process：

1. create music folders

BLEManager.addMusicFolder(MusicOperate.MusicFolder musicFolder)

（2） delete music folders

BLEManager.deleteMusicFolder(MusicOperate.MusicFolder musicFolder)

(3)update music folders

BLEManager.updateMusicFolder(MusicOperate.MusicFolder musicFolder)

（4）Browse music folders

BLEManager.queryMusicAndFolderInfo()

（5） Add music to folders

BLEManager.moveMusicIntoFolderMusicOperate.MusicFolder musicFolder()

1. Delete music from folders

BLEManager.removeMusicFromFolder()

(7) Add music files

BLEManager.addMusicFile(MusicOperate.MusicFile musicFile)

for example:

MusicOperate.MusicFile musicFile = new MusicOperate.MusicFile();

musicFile .music\_memory = 111;// music size

musicFile.music\_name = " boby"; //music name

musicFile.singer\_name = "zhangsan"; // singer name

BLEManager.addMusicFile(musicFile);

(8) delete music files

BLEManager.deleteMusicFile(MusicOperate.MusicFile musicFile)

OperateCallBack.IMusicCallBack;

The setting results will be shown at OperateCallBack.IMusicCallBack;

MusicOperate data class information：

public class MusicOperate {

public int music\_operate;//1: delete music files; 2 add music files

public int folder\_operate;//1. delete folders 2 create new folders 3 update folders 4 add music to folders(playlist) 5 delete music files from folders

When public MusicFolder folder\_items;//folder\_operate value is 0 it is not functional hence it will not occupy any space

When public MusicFile music\_items;//music\_operate value is 0 it is not functional.

public static class MusicFolder implements Serializable {

private static final long serialVersionUID = 1L;

public int folder\_id;

public String folder\_name;

public int music\_num;

public List<Integer> music\_index;

}

public static class MusicFile implements Serializable {

private static final long serialVersionUID = 1L;

public int music\_id;

public long music\_memory;

public String music\_name;

public String singer\_name;

}

public static class OperateResponse{

public int err\_code;//value 0 is success, otherwise failed

public int operate\_type;//0: failed action; 1 delete music files; 2: add music files 3:delete folders 4: create new folders; 5: playlist modification; 6: playlist transferring; 7:delete from playlist

public int music\_id;//music ID when the action is 2: return to music ID when music adding succeed}

public static class MusicAndFolderInfo {

public long all\_memory;// total available space

public long useful\_memory;// available space-possible space fragments, the available space may not exactly be the total available space-used space

public long used\_memory;// used space

public int folder\_num;

public int music\_num;

public List<MusicFolder> folder\_items;

public List<MusicFile> music\_items;

}

}

## Twelve. Graphic Gallery update

1. Retrieve flash information

Port adjustment:

BLEManager.getFlashBinInfo();

**Example of codes:**

//Register device info callback

BLEManager.registerGetDeviceInfoCallBack(new BaseDeviceInfoCallback(){

@Override

public void onGetFlashBinInfo(FlashBinInfo flashBinInfo) {

if(flashBinInfo.version != flashBinInfo.mactchBVersion){

//need update

}

}

}

})

BLEManager.getFlashBinInfo();

FlashBinInfo：Object info:

public int version; // Current graphic gallery version

public int matchVersion;// Matched graphic gallery version

Attn: the version and the matchVersion need to be updated both if different

1. Font stock update
2. Retrieve the match Version graphic gallery information from server (need to be proceed at

the servers in the background

1. Download graphic gallery pack
2. Send to device

Port adjustment: same as files transferring

BLEManager.startTranCommonFile(FileTransferConfig config);

example:

IFileTransferListener mIFileTransferListener = new IFileTransferListener() {

@Override

public void onStart(){};

@Override

public void onProgress(int progress){};

@Override

public void onSuccess(){};

@Override

public void onFailed(String errorMsg){};

}

FileTransferConfig config = FileTransferConfig.getDefaultApolloOTAConfig(updatePath, mIFileTransferListener);//updatePath saving path of updated files

config.macRetryTimes = 0;

config.zipType = FileTransferConfig.ZIP\_TYPE\_FAST\_LZ;`

BLEManager.startTranCommonFile(config);

## Thirteen: BT update

1. Retrieve the bt version#, when the BT\_Version differs from the BT\_mach\_version, the update is required, please refer to Reference list (IV--6 Retrieving the bt version#)

2. Updating BT

(1) Retrieving the BT\_mach\_version information from server(need to be proceed at the servers in the background)

(2) Download BT firmware

(3) Send to device

Port adjustment: same as files transferring

BLEManager.startTranCommonFile(FileTransferConfig config);

example:

IFileTransferListener mIFileTransferListener = new IFileTransferListener() {

@Override

public void onStart(){};

@Override

public void onProgress(int progress){};

@Override

public void onSuccess(){};

@Override

public void onFailed(String errorMsg){};

}

FileTransferConfig config = FileTransferConfig.getDefaultApolloOTAConfig(updatePath, mIFileTransferListener);//updatePath saving path of updated files

config.macRetryTimes = 0;

config.zipType = FileTransferConfig.ZIP\_TYPE\_FAST\_LZ;

BLEManager.startTranCommonFile(config);