ZG protocol Android Bluetooth SDK documentation

Access instructions

1. Please copy the **xxx.aar** file in the SDK to the lib in your peoject project.

2. In the build.gradle dependencies in your project app, add `implementation(name: 'blesdk-debug', ext: 'aar')` and add the following code to build.gradle android:

```
repositories {
        flatDir {
            dirs 'libs'
        }
    }
```

3. Start **BleService.class** in the Application and add `SuperBleSDK.addBleListener(this, new IDataReceiveHandler())`. For specific instructions, refer to the sample code **BleApplicaiton**

4. Register a broadcast receiver `MyReceiver.class` to inherit `BluetoothCallbackReceiver.class` so that the receiver can receive the data globally, can be uniformly distributed to the page that needs data, copy the `BaseActionUtils.class` and `BluetoothCallbackReceiver. Class` in the Demo to your own project, you can write `oncrete()` in your own application, the specific code is as follows:

```
IntentFilter intentFilter = BaseActionUtils.getIntentFilter();
MyReceiver receiver = new MyReceiver();
LocalBroadcastManager.getInstance(this).registerReceiver(receiver, intentFilter);
```

5. Copy the code in the `BleApplication.class` in the Demo `SuperBleSDK.addBleListener(this, new IDataReceiveHandler())` to include the implementation and send the broadcast to the code in your Application.

6. Initialize the SDKType: Call the code 码 `SuperBleSDK.switchSDKTYpe(this,Constants.Bluetooth.Zeroner_Zg_Sdk)`. Note the second parameter selection **Bluetooth.Zeroner_Zg_Sdk**, then call `MyApplication.getInstance().getmService().setSDKType(context .getApplicationContext(), Constants.Bluetooth.Zeroner_Zg_Sdk)`.** Note that this method must be started in `BleService.class` to be executed. If the project is started, it will start to scan and add a delay.:

```
Handler handler = new Handler(Looper.getMainLooper());
        handler.postDelayed(new Runnable() {
            @Override
            public void run() {

MyApplication.getInstance().getmService().setSDKType(context.getApplicationContext(),
Constants.Bluetooth.Zeroner_Zg_Sdk);
            }
        },3000);
```

Scan and connection

1. It is recommended to use `BluetoothUtil.class` to operate. Scan the `BluetoothUtil.startScan()` method. Go back and forth through `LocalBroadcastManager.getInstance(context).registerReceiver(your receiver object, BaseActionUtils.getIntentFilter()). The callback method is explained as follows:

- `onPreConnect()` Initiate a connection callback
- `onScanResult(WristBand band)` Scan result callback, returning the scanned bracelet in this callback method
- `connectStatue(boolean isConnect)` Connection state callback, disconnect and connection when this method is successful
- `onBluetoothInit()` If the connection is successful, data can be exchanged with Bluetooth, and the method is called by the sending and receiving instructions. **Generally, the method successfully processes the identifier of the Bluetooth connection in order to send and receive instructions.**

2. Call `BluetoothUtil.stopScan()` to stop scanning and call `BluetoothUtil.connect(band)` to connect. **Note that the band object is the band in `onScanResult(WristBand band)`**

---

2. Call `BluetoothUtil.stopScan()` to stop scanning and call `BluetoothUtil.connect(band)` to connect. **Note that the band object is the band in `onScanResult(WristBand band)`**

Send and receive instructions

1. Sending a command requires sending data after `onBluetoothInit()`
2. Commands with return values need to add `BackgroundThreadManager.getInstance().addWriteData(context,cmd)` at the end:

```
/*set time*/
byte[] bytes = ZGSendBluetoothCmdImpl.getInstance().setTimeAndWeather();
BackgroundThreadManager.getInstance().addWriteData(context,bytes);
```

APP send command

> Generally, the command ring is returned to the model to resolve the same, the notification is successful, and the transmission succeeds or fails.

- **set time**

```

//Set time The real time of the current system
public byte[] setTimeAndWeather()
//Set time zone parameters, units (year, month, day, hour, minute, second, week, weather,
temperature)
byte[] setTimeAndWeather(int year, int month, int day, int hour, int minute, int second, int
week,int weather, int temperature)

public byte[] setTimeAndWeather(int weather, int temperature)

//天气范围
//晴天 fine day weather = 0
//多云;cloudy  weather = 1
//阴天;over cast  weather = 2
//小雨; light rain weather = 3
//中雨；moderate rain  weather = 4
//大雨; heavy rain  weather = 5
//阵雨： shower weather = 6
//下雪;snow weather = 7
//雾霾；haze； weather = 8
//沙尘暴；sand storm  weather =9
//多云转晴是； = 10；
//雷雨 = 11；
//未知天气 = 12 unknow

//温度: -50 - 50

```

- **Set alarm clock and schedule**

```
    /**
     * Set the alarm clock & schedule No more than 5 alarm clocks 4 schedules
     *
     * @param context
     * @param zgAlarmClockBeanList alarm list
     * @param scheduleList         schedule list
     */
    @Override
    public void setAlarmClockAndSchedule(Context context,
List<ZGAlarmClockScheduleHandler.ZGAlarmClockBean>
zgAlarmClockBeanList,List<ZGAlarmClockScheduleHandler.ZGSchedule> scheduleList)

    //Alarm clock explanation
     public static class ZGAlarmClockBean {
        //16543210 <(1,0 enable bit )Saturday-Friday >4,3,2,1,》Sunday(0)>
        //max 255 11111111 The highest bit is the enable bit set 1 alarm is valid, followed by
Saturday - ... - Sunday
        //eg:Set Monday should be 10000010 - 0x82 set week 2nd Saturday should be 11000110 -0xC6
        public int alarmSet;//Delete write 0
        public int alarmHour; //SET HOUR
        public int alarmMinute;//MINUTES
        public int alarmRingSetting = 1;//Ringtone setting; the upper three digits are 0~7,
corresponding to different ringtones, low
//5 digits is the number of iterations, default, 0x00, if the ringtone setting is 0
        public int alarm_len; //Up to 15, 5 Chinese characters; 15 characters
    }

    //日程类解释
    public static class ZGSchedule {
        public int scheduler_action;////Whether the schedule is valid, delete the schedule this
bye write 0;
        public int scheduler_year;//year
        public int scheduler_month;//month
        public int scheduler_day;//day
        public int scheduler_hour;//hour
        public int scheduler_minute;//minutes
        public int scheringSetting = 1;//ringtone settings; high three is 0 ~ 7, corresponding to
different ringtones, low
//5 digits is the number of iterations, default, 0x00, if the ringtone setting is 0
    }
```

- **Set lanuage**

```
/**
     * Set the language of the bracelet, currently supports Chinese and English
     *
     * @param context
     * @param type    Language type, 0: English 1: Chinese
     * @return
     */
    /**
     * set lanuage
     *
     * @param context
     * @param type    Language type, 0: English 1: Chinese 2: Japanese 3. German 4. Italian 5.
Korean ...
     * @return
     */
    @Override
    public void setLanguage(Context context, int type)
```

- **Calculating calorie switch**

```
/**
     * Calculating calorie switch
     *
     * @param context
     * @param type    0：off 1：on
     * @return
     */
    public void calcKcal(Context context, int type)
```

- **Metric inch conversion Temperature conversion Time format conversion**

```
/**
     * /**
     * Set metric system
     *
     * @param context
     * @param type    0：Metric 1：Imperial
     * @return
     */
    public void setUnitSwitch(Context context, int type)

    //温度转换
    //0 :Celsius 1:Fahrenheit
    public void setTemperatureUnitSwitch(Context context, int type)

    //0:24 hour 1:12 hour
    public void setTimeDisplay(Context context, int type)
```

- **Set weight Walking Run Walking information**

```
    /**
     * Set weight
     *
     * @param context
     * @param weight  user weight (30~255)kg; default 65kg
     * @return
     */
    public void setUserWeight(Context context, int weight)


    /**
     * set Walking
     *
     * @param context
     * @param stride   (30~160)cm default 55cm
     * @return
     */
    public void setWalkStride(Context context, int stride)

    /**
     * set Run
     *
     * @param context
     * @param stride  (40~250)cm default 90cm
     * @return
     */
    public void setRunStride(Context context, int stride)

    /**
     * set walk run
     *
     * @param context
     * @param stride  wStride walk  rStride run
     * @return
     */
    public void setStride(Context context, int wStride, int rStride)
```

- **set target**

```java
/**
 * set walk target
 *
 * @param context
 * @param target default 8000
 * @return
 */
public void setStepsTarget(Context context, int target)
/**
 * calc
 *
 * @param
 * @param target unit is Kcal default 500 kcal
 * @return
 */

public void setKcalTarget(Context context, int target)

/**
 * distance
 *
 * @param context
 * @param target unit is 0.1KM, default 40(4.0)km
 * @return
 */
@Override
public void setDistanceTarget(Context context, int target
```

- **Call message reminder**

```
/**
Call reminder switch
 0 off 1 on
*/
public void setCallNotificationSwitch(Context context, int type)


/**
     * Call reminder effective time
     *
     * @param context
     * @param startHour
     * @param endHour
     * @return
     */
    @Override
    public void setComingCallHours(Context context, int startHour, int endHour)

    /**
     * Call vibration mode and number of times
     *
     * @param context
     * @param type  Represents the type of incoming call vibration reminder, 0 does not vibrate
     * @param count Number of repetitions
     * @return
     */
    public void comingCallShake(Context context, int type, int count)


    /**
     * Message push switch
     *
     * @param context
     * @param type    0：off 1：on
     * @return
     */
    @Override
    public void setMsgNotificationSwitch(Context context, int type)


     /**
     * Message reminder time setting
     *
     * @param context
     * @param startHour
     * @param endHour
     * @return
     */
    @Override
    public void setComingMessageHours(Context context, int startHour, int endHour)


    /**
     * Message vibration
     *
     * @param context
     * @param type  same as calling
     * @param count
     * @return
     */
    @Override
    public void comingMessageShake(Context context, int type, int count)


    /**
     * notification
     *
     * @param header  put ""
     * @param message
     * @return
     */
    public byte[][] messageNotification(String header, String message)


    /**
     * Telephone notification
     *
     * @param header
     * @param message  Telephone information
     * @return
     */
    public byte[][] callNotification(String header, String callInfo)


    //sample
    byte[][] bytes = SuperBleSDK.getSDKSendBluetoothCmdImpl(mContext).callNotification("",
message);
    for (int i = 0; i < bytes.length; i++) {
        BackgroundThreadManager.getInstance().addWriteDataAsMsg(mContext, bytes[i]);
    }
```

- **Heart rate setting**

```
/**
     * Static heart rate on and start end time
     *
     * @param context
     * @param heartOn
     * @param startHour
     * @param endHour
     * @return
     */
   @Override
   public void heartDetection(Context context, int heartOn, int startHour, int endHour)

   /**
     * Exercise heart rate warning vibration
     *
     * @param context
     * @param type Same call
     * @param count
     * @return
     */
   @Override
   public void heartWarmingShake(Context context, int type, int count)


   /**
     * Heart rate warning switch and warning value settings are only valid in sport mode
     *
     * @param context
     * @param warmingOn
     Training mode heart rate alarm switch; default=1
     0, close;
     1, the heart rate is too high, the alarm is on;
     2, heart rate low alarm and excessive alarm are turned on
     * @param heartHighAlarm max
     * @param heartLowAlarm min
     * @return
     */
   public void setHeartAlarm(Context context, int warmingOn, int heartHighAlarm, int
heartLowAlarm)
```

- **Set all vibrations**

```
  /**
   *
   * Set the vibration mode all together
   *
   * @param context
   * @param phoneType Phone with caller settings
   * @param phoneCount times
   * @param msgType Message type Same as caller setting
   * @param msgCount
   * @param setLongType Sedentary with caller settings
   * @param setLongCount
   * @param heartType Heart rate with caller settings
   * @param heartCount
   */
  public void setShake(Context context, int phoneType, int phoneCount, int msgType, int msgCount,
int setLongType, int setLongCount, int heartType, int heartCount)
```

- **Vibration test**

```
 **
  * Test vibration mode (test motor)
  *
  *
  * @param mode Vibration mode
  * @param times Vibration times
  * @return
  */
 @Ove
 public byte[] testShake(@IntRange(from = 1, to = 7) int mode, @IntRange(from = 0, to = 31) int
times)
```

- **Sedentary setting**

```
/**
  * @param context
  * @param alarm
  =0, sedentary reminder is closed;
  =1, sedentary reminder is turned on, noon disturbance is invalid;
  =2, sedentary reminder is on, noon disturbance is effective
  * @param startHour Sedentary start time default 8
  * @param endHour   Sedentary start time  default 18
  * @return
  */
 @Override
 public void setLongSitAlarm(Context context, int alarm, int startHour, int endHour)

 /**
  * Sedentary reminder
  *
  * @param context
  * @param type  Same call
  * @param count
  * @return
  */
 @Override
 public void comingLongSitShake(Context context, int type, int count)
```

- **Turn the wrist screen**

```
/**
  * Turn over the bowl bright switch
  *
  * @param context
  * @param gestureOn 0：off 1:on
  * @param startHour
  * @param endHour
  * @return
  */
 @Override
 public void setGesture(Context context, int gestureOn, int startHour, int endHour)
```

Bracelet report type command

The return value is returned as a json string, which can be parsed according to Google Gson or Alibaba fastJson.

- **Get firmware information**

```
/**
*获取固件信息
/
public byte[] getHardwareFeatures()


返回的model

public class ZGHardwareInfo {
    //version
    private int dev_version;
    //version string
    private String dev_version_s;
    //dev has screen
    private int dev_screen;
    //has key
    private int dev_key_type;
    //The font type of the device IC, 0 is no font, 1...N is specified later
    private int dev_fontic;
    //The Gsensor type of the device, 0 is no G sensor, 1...N is the model specified later;
    private int dev_gsensor;
    //The motor type of the device, 0 is no motor, 1...N is the model specified later;
    private int dev_moto;
    //The heart rate module of the device, 0 is not, 1...N is the model specified later;
    private int dev_heart;
    //The CFCA module of the device, 0 is not, 1...N is the model specified later;
    private int dev_cfca;
    //The NFC module of the device, 0 is not, 1...N is the model specified later;
    private int dev_nfc;
    //The device retains information; the default is 0;
    private int dev_reserve = 0;
    //model name
    private String model = "";
}
```

- Get bracelet information

```
/**
  * Get bracelet settings information
  *
  * @return
  */
 @Override
 public byte[] getFirmwareInformation()

 //retrurn model
 public class DeviceSetting {
     /**
 BIT[7:4]: 0000 = English; 0001 = Chinese;
 0010 = Japanese; 0011 = German;
 0100 = Italian; 0101 = Korean;
 BIT[3]: 0/1, which means no calculation/calculation of static calories; default 0;
 BIT[2]: 0/1, representing Celsius/F default 0;
 BIT[1]: 0/1 represents metric/imperial, respectively. Default 0
 BIT[0]: 0/1 represents 24 hours / 12 hours, respectively, default
     */
     private int unitSet;
     /**
     User weight, unit kg, default is 65kg, the range setting is
     (30~255)kg;
     */
     private int weight;
     /*
     Walking stride is the length of the stride when the user sets the walking, generally the
height
1/3~1/2, default 55cm, the range setting is (30~160)cm;
     */
     private int walkStride;
     /*
     The walking stride is the length of the stride when the user sets the walking, which is
generally the walking step.
1.5 times longer; default 90cm; range setting is (40~250)cm*/
     private int runStride;
     /*
     The user sets the daily movement step target, the default is 8000; the range setting is
(0~60000); 0 means not detecting exercise goals
     */
     private int stepsOnceday;
     /*
     Set daily calories burned (note, no static calories burned)
Target; unit is big card, the default is 500 big card, the range is (0~20000); 0 generation
Table does not detect calorie targets
     */
     private int calorieOnceday;
     /*
     The user sets the daily distance travel target; the unit is 0.1KM, the default is
40 (4.0) km, range is (0~250)
     */
     private int distanceOnceday;
     /*

     */
     private int stepsReachRing;
     private int caloriesReachRing;
     private int distanceReachRing;
     /*
     0/1, the call reminder is closed/open; the default is open=1;
```

```
    */
    private int comingCallEnable;
    // call prompt set time to start the effective time (> = 0)
    private int comingCallStartHour;
    // call prompt set the end of the effective time (<=23);
    private int comingCallEndHour;
    //High three represent the type of incoming call vibration reminder, 0 does not vibrate, 1~7
corresponds to 7
    //Vibration type, the default is 1, the low 5 bits represent the number of repetitions, the
default is
    //0x001 00001 = 0x21
    private int comingCallRing;
    //0/1, the message reminder is closed/open; the default is open=1;
    private int messageEnable;
    //Message reminder set time to start the effective time (>=0) Default 9
    private int messageStartHour;
    //The end of the effective time of the message reminder setting (<=23); default 20
    private int messageEndHour;
    // Same as ComingCallRing;
    private int messageRing;
    // Automatic background detection of heart rate is turned on, the default is open = 1;
    //=0, close;
    //= 1, open;
    private int quietHeartEnable;
    // Background automatically test the start time of the heart rate, the default all-day test,
=
    private int quietHeartStartHour;
    / / Background automatically test the end of the heart rate, the default all-day test, = 23
    private int quietHeartEndHour;
    // Training mode heart rate alarm switch; default = 1
    //=0, close;
    //=1, the heart rate is too high and the alarm is on;
    //=2, heart rate low alarm and high alarm are turned on;
    //Other invalid
    private int heartAlarmEnable;
    // Training mode heart rate is too high alarm, the default 160
    private int highHeartAlarm;
    // Training mode heart rate low alarm, the default 95
    private int lowHeartAlarm;
    //With ComingCallRing
    private int heartRing;
    // sedentary reminder switch, default =0;
    //=0, sedentary reminder is closed;
    //=1, Sedentary reminder is turned on, no interruption is invalid at noon;
    //=2, Sedentary reminder is turned on, no interruption is effective at noon;
    //Other values are invalid
    private int sitLongAlarmEnable;
    / / Sedentary reminder to start detection time, the default 8
    private int sitlongStartHour;
    // sedentary reminder end detection time, default 18
    private int sitlongEndHour;
    // sedentary reminder reminder: same as ComingCallRing;
    private int sitlongRing;
    // Turn the wrist bright screen is open, the default is =1;
    //0: Closed;
    //1: Open;
    private int rollEnable;
    // Turn the wrist bright screen open time, the default
    private int rollStartHour;
    // Turn the wrist bright screen end time, the default 22 (7 <= effective time <= 22)
    private int rollEndHour;
    //0 = default dial, 1 is another dial;
    private int watchSelect;
    //charge
    private int batteryVolume;
 }
```

- **Get the number of days of exercise**

```
public byte[] getDataDate()

return model

public class EveryDayInfo {
 public long year;
 public int month;
 public int day;
}
```

- **Get the total data for a day**

```
public byte[] getTotalData(TDay tday)

 // Today, Synchronize today's total data
 // T_1, T-1
 // T_2, ...
 // T_3,
 // T_4,
 // T_5,
 // T_6,
 // T_7

 return model
 public class bh_totalinfo {
     private int year;
     private int month;
     private int day;

     private int calorie;//Total calories
     //unit: Meter
     private int distance; //Total distance
     //Time unit: minute
     private int exerciseMinutes;//Training time
     private int sleepMinutes; //Total sleep

     private int latestHeart;//Last heart rate
     private int step;//Total number of steps
 }
```

- **Get a day's step data**

```
//Day with getTotalData().0 means today 1 means yesterday...
public byte[] getDetailWalk(int day)

return model
public class ZgDetailWalkData {
     private int year;
     private int month;
     private int day;
 //    private int count;
     //1440个点，每分钟的步数
     private List<Integer> data;
 }
```

- **Get heart rate data for a certain day**

```
public byte[] readHeartData(int day)

返回model
public class ZGHeartData {

     public int year;
     public int month;
     public int day;
     public int highestHeart;
     public int lowHeart;
     public int averageHeart;
     public int[] staticHeart= new int[144];
 }
```

- **Get training data**

```
public byte[] getDetailSport(int day)

//return model
public class ZgDetailSportData {
    private int year;
    private int month;
    private int day;
    //Number of sports on the day
    private int count;
    private List<Sport> sports;

    public static class Sport{
        //Length of exercise (minutes)
        private int totalMin;
        //Distance from 0:00 on the day (minutes) 120->02:00
        private int startMin;
        //0 o'clock from the day
        private int endMin;
        private int steps;
        //Unit: m
        private float distance;
        //Unit:kcal
        private float calories;
        //Maximum step frequency
        private int spmMax;
        //Average step frequency
        private int spmAvg;
        private int heartMax;
        private int heartAvg;
        //sport type
        private int sportType;
        private List<Integer> heart =new ArrayList<Integer>();
```