QC Wireless SDK Instructions for Use

> 1. Author：James

update note:

> 1. (2021/07/06) scan, connect, measure commands
> 2. (2021/07/20) Add setting command
> 3. (2021/07/21) Increase step count, heart rate, sleep data sync
> 4. (2022/02/28) Add new sleep algorithm
> 5. (2023/03/10) Add message switch synchronization, body temperature, user information, watch manual measurement of heart rate, blood pressure results
> 6. (2023/03/16) Add bt connection and contact person
> 7. (2024/02/23) Add manual pressure, APP movement, pressure synchronization, pressure setting
> 8. (2025/05/11) Add One-click measurement
> 9. (2025/05/11) Temperature switch setting

1.introduce

1.1The role of the SDK

Provide partner companies with the Android Bluetooth SDK for use with Green Orange wireless devices that provide basic and advanced functionality for a major watch or other device. This document is intended to explain the usage context, functionality, etc. of the API. Intended Audience and Reading Recommendations The intended audience and reader recommendations in this article are shown in Annex 1.

Reader	Role
Software Architecture Engineer	Architecture Analysis and Technical Guidance
Android development engineer	Have a certain android development ability, understand Ble related development technology

2. API description

2.0 Access Conditions

> Android 5.0 or above, Bluetooth 4.0 or above.

2.1 Permissions required by the SDK

```
    //network permissions
<uses-permission android:name="android.permission.INTERNET" />
//Bluetooth related permissions
    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
//Storage related permissions
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
//Location permission
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

```
If it is android 12 or higher system
        <uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
    <uses-permission android:name="android.permission.BLUETOOTH_SCAN" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADVERTISE" />
```

2.2 Access conditions

- Green Orange Wireless Wearables
- Green Orange Wireless SDK and documentation

2.3.1API

Scanning Devices

```
    //start scan
    BleScannerHelper.getInstance().scanDevice(final Context context, UUID mUuid,
final ScanWrapperCallback scanCallBack);
//stop scan
    BleScannerHelper.getInstance().stopScan(Context context)
//Specified device scan
    BleScannerHelper.getInstance().scanTheDevice(final Context context, final
String macAddress, final OnTheScanResult scanResult)
```

device connection:BleOperateManager.getInstance()

```
    //direct connection
    BleOperateManager.getInstance().connectDirectly(smartWatch.deviceAddress)
//scan connections
    BleOperateManager.getInstance().connectWithScan(smartWatch.deviceAddress)
//disconnect
    BleOperateManager.getInstance().unBindDevice()
//reconnect
    BleOperateManager.getInstance().setNeedConnect(boolean needConnect)
//Called when bluetooth is turned off
    BleOperateManager.getInstance().setBluetoothTurnOff(false)
    BleOperateManager.getInstance().disconnect()
//Turn on the system bluetooth monitor
    BleOperateManager.getInstance().setBluetoothTurnOff(true)

//bind seccess
      CommandHandle.getInstance()
                .executeReqCmdNoCallback(
        SimpleKeyReq(Constants.CMD_BIND_SUCCESS)
                )
```

2.3.2 Feature list:

Synchronize time, get the list of functions supported by the device

```
    //set time
    CommandHandle.getInstance().executeReqCmd(SetTimeReq(0),
ICommandResponse<SetTimeRsp>() {})
//Callback Description
public class SetTimeRsp extends BaseRspCmd {
    //support body temperature
    public boolean mSupportTemperature;
    //watch face
    public boolean mSupportPlate;
    //Support the menstrual cycle
    public boolean mSupportMenstruation;
    //Support custom watch faces
    public boolean mSupportCustomWallpaper;
    //Support blood oxygen
    public boolean mSupportBloodOxygen;
    //blood pressure support
    public boolean mSupportBloodPressure;
    //Support fatigue
    public boolean mSupportFeature;
    //Support one-key detection
    public boolean mSupportOneKeyCheck;
    //weather support
    public boolean mSupportWeather;
    //Support for new sleep protocol
    public boolean mNewSleepProtocol;
    //Supports up to 6 or 3 dials
    public int mMaxWatchFace;
    // hrv support
    public boolean mSupportHrv;
}

Functional support
 CommandHandle.getInstance()
            .executeReqCmd(
        DeviceSupportReq.getReadInstance(),
                ICommandResponse<DeviceSupportFunctionRsp> {

    //support touch
    public boolean supportTouch;
    //support muslim
    public boolean supportMoslin;
    //support ble pair
    public boolean supportAPPRevision;
    //support Heart rate calibration
    public boolean supportBlePair;
    //support gesture
    public boolean supportGesture;
    //support music
    public boolean supportRingMusic;
    //support video
    public boolean supportRingVideo;
```

```
    //support ebook
    public boolean supportRingEbook;
    //support camera
    public boolean supportRingCamera;
    //support phone call
    public boolean supportRingPhoneCall;
    //support game
    public boolean supportRingGame;
}
```

bracelet battery

```
       CommandHandle.getInstance().executeReqCmd(new
SimpleKeyReq(Constants.CMD_GET_DEVICE_ELECTRICITY_VALUE), new
ICommandResponse<BatteryRsp>() {

@Override
public void onDataResponse(BatteryRsp resultEntity) {
    if (resultEntity.getStatus() == BaseRspCmd.RESULT_OK) {
        //Get battery successfully
    }
}

        });

//Callback Description
public class BatteryRsp extends BaseRspCmd {
    //battery【0-100】
    private int batteryValue;
}
```

Continuous heart rate, blood oxygen, blood pressure switch

```
     //Read continuous heart rate settings (too Low/High Reminder is notify in
DeviceNotifyListener 0x3A.need
BleOperateManager.getInstance().addOutDeviceListener(ListenerKey.Heart,yourListen
er))
      CommandHandle.getInstance()
          .executeReqCmd(HeartRateSettingReq.getReadInstance(),
              ICommandResponse<HeartRateSettingRsp> {
    //it.isEnable         switch
    //it.heartInterval    heart rate measurement interval
    //it.startInterval    minValue/minUnit 5、10、30
    //tooLowReminder low value reminder 0、40、45、50   0:tooLowReminder is off
other value: is on
    //tooHighReminder high value reminder 0、110、120、130、140、150
0:tooHighReminder is off   other value: is on
    // mainSwitch 0:no support main control 1:support main control and is on
2:1:support main control and is off
})
        //Read Continuous SpO2 settings
        CommandHandle.getInstance()
          .executeReqCmd(BloodOxygenSettingReq.getReadInstance(),
```

```
            ICommandResponse<BloodOxygenSettingRsp> {
    //it.isEnable         switch
    //it.interval    interval time unit min
})
        //Read continuous blood pressure settings
        CommandHandle.getInstance()
          .executeReqCmd(BpSettingReq.getReadInstance(),
              ICommandResponse<BpSettingRsp> {

})
        //Read pressure setting
        CommandHandle.getInstance()
          .executeReqCmd(PressureSettingReq.getReadInstance(),
              ICommandResponse<PressureSettingRsp>() {
        switch:it.isEnable
              })

                      //Write continuous heart rate switch isEnable: true on,
false: off  hrInterval: 10, 15, 20, 30, 60

                      CommandHandle.getInstance().executeReqCmd(
        HeartRateSettingReq.getWriteInstance(true,hrInterval),
          ICommandResponse<HeartRateSettingRsp> {

})
        //Write continuous heart rate switch isEnable: true on, false: off
hrInterval: 10, 15, 20, 30, 60
        //startInterval minValue/minUnit 5、10、30
        //tooLowReminder low value reminder 0、40、45、50   0:tooLowReminder is
off   other value: is on
        //tooHighReminder high value reminder 0、110、120、130、140、150
0:tooHighReminder is off   other value: is on
        // mainSwitch 0:no support main control 1:support main control and is on
2:support main control and is off
        // too Low/High Reminder is notify in DeviceNotifyListener 0x3A(need
BleOperateManager.getInstance().addOutDeviceListener(ListenerKey.Heart,yourListen
er)
        CommandHandle.getInstance().executeReqCmd(
        HeartRateSettingReq.getWriteInstance(
                isEnable,
        hrInterval,
        startInterval,
        tooLowReminder,
        tooHighReminder
        ),
                        ICommandResponse<HeartRateSettingRsp> {

})
        //add  Heart rate tooLow/High Reminder Listener
        BleOperateManager.getInstance()
                        .addOutDeviceListener(ListenerKey.Heart,
myDeviceNotifyListener)

//Write continuous blood oxygen switch isEnable: true on, false: off
        CommandHandle.getInstance().executeReqCmd(
        BloodOxygenSettingReq.getWriteInstance(boolean isEnable),
```

```
          ICommandResponse<BloodOxygenSettingRsp> {

})
        //Write continuous interval blood oxygen switch isEnable: true on, false:
off. interval: interval time unit min
        CommandHandle.getInstance().executeReqCmd(
        BloodOxygenSettingReq.getWriteInstance(boolean isEnable,int interval),
          ICommandResponse<BloodOxygenSettingRsp> {

})
        //write blood pressure switch

 CommandHandle.getInstance().executeReqCmd(BpSettingReq.getWriteInstance(boolean
isEnable, StartEndTimeEntity startEndTimeEntity, int multiple),
 ICommandResponse<BpSettingRsp> {

})
BpSettingRsp,Parameter Description
isEnable: true on false off
StartEndTimeEntity The parameter description is the same as above
multiple  default 60

        //Write pressure setting switch,switch isEnable: true on, false: off
        CommandHandle.getInstance().executeReqCmd(
        PressureSettingReq.getWriteInstance(isEnable),
          ICommandResponse<PressureSettingRsp> {

})


        //hrv readme
        CommandHandle.getInstance()
          .executeReqCmd(HrvSettingReq.getReadInstance(),
              ICommandResponse<HRVSettingRsp>() {

        })

        //hrv write
        CommandHandle.getInstance().executeReqCmd(
        HrvSettingReq(true),
          ICommandResponse<HRVSettingRsp> {

})
        //hrv write  interval
        CommandHandle.getInstance().executeReqCmd(
        HrvSettingReq(true,int interval),
          ICommandResponse<HRVSettingRsp> {

})


        //body temperature write
        CommandHandle.getInstance().executeReqCmd(
        SugarLipidsSettingReq.getWriteInstance(0x03,isEnable,interval),
          ICommandResponse<BloodSugarLipidsSettingRsp> {  })
        // temperature interval 2-120s
        CommandHandle.getInstance().executeReqCmd(
        SugarLipidsSettingReq.getWriteInstance(0x03,isEnable,0,interval),
```

```
          ICommandResponse<BloodSugarLipidsSettingRsp> {  })

body temperature read
          CommandHandle.getInstance()
          .executeReqCmd(SugarLipidsSettingReq.getReadInstance(0x03),
              ICommandResponse<BloodSugarLipidsSettingRsp>() {
        if (it.type == 0x03.toByte()) {
            //it.isEnable
                 }
              })

                      //ring bluetooth enable  0:disenable 1:enable
                      CommandHandle.getInstance()
          .executeReqCmd(BluetoothEnableReq.getWriteInstance(int enable)
```

Set watch sports goals

```
        CommandHandle.getInstance().executeReqCmd(
        TargetSettingReq.getWriteInstance(
                final int step, final int calorie, final int distance, final int
sportMinute, final int sleepMinute
), null
        )

Parameter Description
step: step target
calorie: Calorie target target, unit card, write kcal to a*1000
distance: Distance to target, in meters
sportMinute: Exercise duration target, in minutes
sleepMinute: Sleep duration target, in minutes
```

Find equipment

```
CommandHandle.getInstance().executeReqCmd(FindDeviceReq(), null)
```

R11 RING Display settings

```
// isEnable: whether to display the switch
// isLeft: default left-hand mode (true = left hand, false = right hand)
// light: brightness level, default is 1
// maxLight: maximum brightness, default is 5
// start / end: screen-on and screen-off time
//              Full day: start = 0, end = 0
//              Otherwise, time is the number of minutes since midnight
//              For example, 3:00 AM = 180


CommandHandle.getInstance().executeReqCmd(palmScreen,ICommandResponse<PalmScreenR
sp>{

            })

    //read
```

```
        CommandHandle.getInstance()
            .executeReqCmd(
                PalmScreenReq.getRingReadInstance(),
                ICommandResponse<PalmScreenRsp>() {

                })
```

PalmScreenRsp description:

```
/**
 * Whether the feature is enabled
 */
private boolean isEnable;

/**
 * Whether the device is worn on the left hand
 */
private boolean isLeft;

/**
 * Whether palm gesture (wrist flip) is required
 */
private boolean needPalm;

/**
 * Screen brightness level
 */
private int screenLight;

/**
 * Maximum brightness level
 */
private int maxLight;

/**
 * All-day mode (Do Not Disturb)
 */
private boolean dnd;

/**
 * Start time
 */
private int start;

/**
 * End time
 */
private int end;
```

Ring photo control

```
 //The bracelet enters the camera interface

CommandHandle.getInstance().executeReqCmd(CameraReq(CameraReq.ACTION_INTO_CAMARA_
UI), null)
        //The wristband is controlled by the bright screen on the camera
interface. The APP will send the bright screen command to keep the watch bright.
It is recommended to send it every 2 seconds.
        CommandHandle.getInstance().executeReqCmd(
        CameraReq(CameraReq.ACTION_KEEP_SCREEN_ON),
                            null
                                    )
                                    //Bracelet click to take a photo event
monitoring

 BleOperateManager.getInstance().addNotifyListener(Constants.CMD_TAKING_PICTURE,n
ew ICommandResponse<CameraNotifyRsp>(){

    @Override
    public void onDataResponse(CameraNotifyRsp resultEntity) {


    }
});
        resultEntity.getAction()
Parameter Description
//The watch exits the camera interface
CameraNotifyRsp.ACTION_FINISH
//The watch clicked on the photo event
CameraNotifyRsp.ACTION_TAKE_PHOTO


CommandHandle.getInstance().executeReqCmd(CameraReq(CameraReq.ACTION_FINISH),
null)
```

Set the Ring to factory reset

```
CommandHandle.getInstance().executeReqCmd(RestoreKeyReq(Constants.CMD_RE_STORE),n
ull)
```

- message push

```
 //The watch message push switch is fully turned on, and the APP should be
actively opened
  CommandHandle.getInstance().executeReqCmd(
        SetANCSReq(), null
        )
//Send message push to watch
        MessPushUtil.pushMsg(type,message:String)

PushMsgUintReq parameter description
```

```
type:
        0x00: Call reminder 0x01: SMS reminder 0x02: QQ reminder 0x03: WeChat
reminder,
       0x04: incoming call to answer or hang up 0x05: Facebook message reminder
0x06: WhatsApp message reminder
       0x07: Twitter message reminder 0x08: Skype message reminder 0x09: Line
message reminder 0x0a: Linkedln
       0x0b: Instagram 0x0c: TIM message 0x0d: Snapchat
       0x0e: others other types of notifications
```

Shutdown the Ring

```
BleOperateManager.getInstance().shutdown()
```

Wearing Calibration

```
 start :enable:true
end :enable:false
        BleOperateManager.getInstance().ringCalibration(false
) {
    //2:Measuring  1:success 3:fail
    it.success
}

  ```


          #### firmware version number, hardware version number
```java
//hardware information

CommandHandle.getInstance().execReadCmd(CommandHandle.getInstance().getReadHwRequ
est());
        //firmware information

 CommandHandle.getInstance().execReadCmd(CommandHandle.getInstance().getReadFmReq
uest());

Receiving implements this QCBluetoothCallbackCloneReceiver refer to demo
MyBluetoothReceiver
Judging UUID in the callback onCharacteristicRead

override fun onCharacteristicRead(uuid: String?, data: ByteArray?) {
    if (uuid != null && data != null) {
        val version = String(data, Charsets.UTF_8)
        when(uuid){
            Constants.CHAR_FIRMWARE_REVISION.toString() -> {
                //Firmware version number version
            }
            Constants.CHAR_HW_REVISION.toString() -> {
                //hardware version number version number version
            }
        }
    }
```

```
}
```

Close Bluetooth

```
//Generally, charging will restart the Bluetooth.
```java
     CommandHandle.getInstance()
            .executeReqCmd(
        BluetoothCloseReq.getWriteInstance(),
object : ICommandResponse<BluetoothCloseRsp> {
    override fun onDataResponse(bluetoothCloseRsp: BluetoothCloseRsp) {
        //nothing
    }
})
```

2.3.3 Data synchronization:

Synchronize steps, distance, kcal for the day

```
    CommandHandle.getInstance().executeReqCmd(
        SimpleKeyReq(Constants.CMD_GET_STEP_TODAY),
           ICommandResponse<TodaySportDataRsp> {})

TodaySportDataRsp parameter description
// days ago
private int daysAgo;
// date: year
private int year;
// date: month
private int month;
// date: day
private int day;
// total steps
private int totalSteps;
// running steps/aerobic steps
private int runningSteps;
// calorie value
private int calorie;
// walking distance
private int walkDistance;
// Movement time, in seconds
private int sportDuration;
// sleep time in seconds
private int sleepDuration;
```

Synchronized step data details

```
    dayOffset 0: Today 1: Yesterday 2: The day before yesterday, supports
synchronization for up to 7 days
    CommandHandle.getInstance().executeReqCmd(
        ReadDetailSportDataReq(dayOffset, 0, 95),
```
```

```
            ICommandResponse<ReadDetailSportDataRsp> {

})
BleStepDetails parameter description
//year
private int year;
//moon
private int month;
//day
private int day;
//15 minutes a point, the total number of points in a day is 96 points, [0, 95],
used to calculate the details of the number of steps per hour
private int timeIndex=0;
// calorie unit card
private int calorie=0;
//Step count
private int walkSteps=0;
//distance in meters
private int distance=0;
// keep for now
private int runSteps=0;
```

Synchronize the details of new sleep data and return according to SetTimeRsp

```
    //offset 0 today 1 yesterday
public void syncSleepList(int offset, final ILargeDataSleepResponse response)
SleepNewProtoResp   parameter description
//sleep start time
private int st;
//sleep end time
private int et;
//sleep collection
private List<DetailBean> list;
DetailBean parameter description
// duration of a sleep type
private int d;
//0未睡眠，1摘下，2浅睡，3深睡，4眼动，5清醒；
private int t;

//support lunch
public void syncSleepList(int offset, final ILargeDataSleepResponse
response,final ILargeDataLaunchSleepResponse lunchSleepResponse){

}
SleepNewProtoResp
lunch start time
private int lunchSt;
lunch end time
private int lunchEt;
private boolean lunchBreak;
private List<DetailBean> list;
```

Sedentary data synchronization

```
    //offset 0 today 1 yesterday
public void syncLongSitList(int offset, final ILargeSettingForLongDataResponse
response)
LongSitResp   parameter description
// offset 0 today 1 yesterday
private int index;
//Sedentary collection
private List<DetailBean> list;
DetailBean parameter description
// duration of a sedentary type
private int d;
//0 static (less than 30 steps in 1 minute), 1 trigger sedentary, 2 movement
(more than 30 steps)
private int t;
```

Sync heart rate data

```
 nowTime current time zone * 3600 + unix second value of current time
Sync yesterday: nowTime-(24*3600)*1,
Sync the day before yesterday: nowTime-(24*3600)*2
Data can be synchronized for up to three days
val time = (getTimeZone() * 3600).toInt()
val nowTime = date.unixTimestamp + time

  CommandHandle.getInstance().executeReqCmd(
        ReadHeartRateReq(nowTime),
             ICommandResponse<ReadHeartRateRsp> {

})

ReadHeartRateRsp parameter description
//nothing yet
private int size = 0;
//nothing yet
private int index = 0;
// unix second value of heart rate data
private int mUtcTime;
//The heart rate data array is one point every 5 minutes, the data subscript *5
is equal to the number of minutes of the day
private byte[] mHeartRateArray;
private boolean endFlag = false;
```

Synchronized blood pressure function

```
//Synchronized automatic blood pressure, measured once an hour
   CommandHandle.getInstance()
            .executeReqCmd(SimpleKeyReq(Constants.CMD_BP_TIMING_MONITOR_DATA),
ICommandResponse<BpDataRsp> {}
BpDataEntity parameter description
//year
private int year;
```

```
//moon
private int mouth;
//day
private int day;
private int timeDelay;
private ArrayList<BpValue> bpValues;

BpValue parameter description
//The minute of the day, usually the whole hour
int timeMinute;
//measured heart rate value
int value;

Get the blood pressure value calculated from the measured value
//The heart rate value returned by the hr callback, age is the age of the user
val sbp= CalcBloodPressureByHeart.cal_sbp(hr, age) (systolic blood pressure)
//sbp heart rate calculated value
val dbp=CalcBloodPressureByHeart.cal_dbp(sbp) (diastolic pressure)

        //Confirm blood pressure synchronization, call after receiving the
callback, the watch will delete the records that have been synchronized
        CommandHandle.getInstance().executeReqCmd(BpReadConformReq(true),null)

        //synchronize manual blood pressure
        CommandHandle.getInstance()
            .executeReqCmd(ReadPressureReq(0),
ICommandResponse<ReadBlePressureRsp> {}

    ReadBlePressureRsp.getValueList() parameter description
BlePressure parameter descriptionw
//time seconds value
public long time;
//(Diastolic pressure)
public int dbp;
//(systolic blood pressure)
public int sbp;
```

Synchronized blood oxygen function

```
 LargeDataHandler.getInstance().syncBloodOxygenWithCallback(new
IBloodOxygenCallback() {
    @Override
    public void readBloodOxygen(List<BloodOxygenEntity> data) {

    }
});

BloodOxygenEntity parameter description
//data date
private String dateStr;
//Data minimum value array, one data per hour, a total of 24
private List<Integer> minArray;
//Data maximum value array, one data per hour, a total of 24
private List<Integer> maxArray;
//Data value at 0:00 on a certain day
```

```
private long unix_time;
```

Synchronized Time Interval blood oxygen function

```
//One data point per minute, 1,440 data points per day
//dayIndex:Index of the day (0 = today, 1 = yesterday, and so on)
LargeDataHandler.getInstance().syncIntervalBloodOxygenWithCallback(dayIndex,new
IBloodOxygenCallback() {
    @Override
    public void readIntervalBloodOxygen(IntervalBloodOxygenEntity data) {

    }
});

IntervalBloodOxygenEntity parameter description
//dayIndex
private int dayIndex;
//interval
private int interval;
//Data value array, one data per min
private List<Integer> array;
```

Synchronized Time Interval temperature function

```
//One data point per minute, 1,440 data points per day
//dayIndex:Index of the day (0 = today, 1 = yesterday, and so on)
LargeDataHandler.getInstance().syncIntervalTemperatureWithCallback(dayIndex,new
IIntervalTemperatureCallback() {
    @Override
    public void readIntervalTemperature(IntervalTemperatureEntity data) {

    }
});

IntervalTemperaturenEntity parameter description
//dayIndex
private int dayIndex;
//interval
private int interval;
//Data value array, one data per min
private List<Float> array;
```

Synchronized pressure function

```
   //offset  0-6
//Synchronization pressure 7 days 0 only synchronizes today 1 yesterday 2
synchronizes the day before yesterday .... supports up to 7 days
  CommandHandle.getInstance()
            .executeReqCmd(PressureReq(offset),
```

```
                ICommandResponse<PressureRsp> {

})

//PressureRsp Parameter Description
//Pressure data, the return value divided by 10 is the value displayed by the
APP, and one data is generated every half hour.
private byte[] pressureArray;
//Stress test interval
private int range=30;
//Pressure data date
private DateUtil today;
```

Synchronized hrv function

```
   //offset  0-6
//Synchronization hrv 7 days 0 only synchronizes today 1 yesterday 2 synchronizes
the day before yesterday .... supports up to 7 days
  CommandHandle.getInstance()
            .executeReqCmd(HRVReq(0),
                ICommandResponse<HRVRsp> {

})

//HRVRsp Parameter Description
//hrv data, the return value divided by 10 is the value displayed by the APP, and
one data is generated every half hour.
private byte[] pressureArray;
//hrv test interval
private int range=30;
//hrv data date
private DateUtil today;
```

Synchronized skin(body) temperature function

```
    fun registerTempCallback(){
    //Register a callback
    FileHandle.getInstance().clearCallback()
    FileHandle.getInstance().registerCallback(Callback())
    FileHandle.getInstance().initRegister()
}
//Synchronized automatic skin temperature
fun syncAutoTemperature(){
    ////Synchronize automatic body temperature for 3 days 0 Synchronize only
today 1 Synchronize today and yesterday 2 Synchronize today, yesterday and the
day before yesterday .... Support up to 7 days
    FileHandle.getInstance().startObtainTemperatureSeries(2)
}
//Synchronous manual skin temperature
fun syncManual(){
    //Synchronize automatic body temperature for 3 days 0 Synchronize only today
1 Synchronize today and yesterday 2 Synchronize today, yesterday and the day
before yesterday .... Support up to 7 days
    FileHandle.getInstance().startObtainTemperatureOnce(0)
```

```
}

inner class Callback : SimpleCallback() {
    //Continuous temperature callback
    override fun onUpdateTemperature(data: TemperatureEntity) {

    }
    //Single temperature callback
    override fun onUpdateTemperatureList(array:
MutableList<TemperatureOnceEntity>) {

    }
}

TemperatureEntity Parameter Description
public int mIndex; represents today 1 represents yesterday ...6
public int mTimeSpan;  Test interval
public float[] mValues; Temperature array cursor*mTimeSpan = measurement time of
the day, in minutes
public float[] mSideValues;  Side temperature array cursor*mTimeSpan =
measurement time in a day, in minutes

TemperatureOnceEntity Parameter Description
public long mTime;  Measurement timestamp, seconds
public float mValue;Measurement value
```

Synchronous training records

```
val syncSport= SportPlusHandle()
syncSport.timeFormat="yyyy-MM-dd HH:mm"
syncSport.syncSportPlus(object :
BaseTransientBottomBar.BaseCallback<MutableList<SportPlusEntity>>() {
            public override fun result(
                errorCode: Int,
                sportPlusEntities: MutableList<SportPlusEntity>

            ) {
            }
        })
//Obtain all the data from the specified time until the present.0 is mean all.
//Note:The equipment only stores the latest ten pieces of data.
syncSport.cmdSummary(0)


SportPlusEntity
/**
 * Movement type index
 */
public int mSportType;
/**
 * Start time seconds
 */
public int mStartTime;
/**
```

```
     * Movement duration seconds
     */
    public int mDuration;
    /**
     * Movement mileage meters
     */
    public int mDistance;
    /**
     * Calories small calories
     */
    public float mCalories;
    /**
     * Average speed cm/s
     */
    public int mSpeedAvg;
    /**
     * Maximum speed cm/s
     */
    public int mSpeedMax;
    /**
     * Average heart rate beats/minute
     */
    public int mRateAvg;
    /**
     * Minimum heart rate beats/minute
     */
    public int mRateMin;
    /**
     * Maximum heart rate beats/minute
     */
    public int mRateMax;
    /**
     * Average altitude cm
     */
    public int mElevation;
    /**
     * Cumulative climbing cm
     */
    public int mUphill;
    /**
     * Cumulative downhill cm
     */
    public int mDownhill;
    /**
     * Average cadence steps/minute
     */
    public int mStepRate;
    /**
     * Number of exercises times
     */
    public int mSportCount;

    public int steps;

    public List<SportLocation> mLocations = new ArrayList<>();
```

```
SportLocation {

/**
 * Real-time heart rate beats/minute
 */
    public int mRateReal;
}

typedef NS_ENUM(NSInteger, OdmSportPlusExerciseModelType) {
    OdmSportPlusExerciseModelTypeGpsRun = 1, //gps running
            OdmSportPlusExerciseModelTypeGpsBike = 2, //gps cycling
            OdmSportPlusExerciseModelTypeGpsWalk = 3, //gps walking
            OdmSportPlusExerciseModelTypeIndoorRun = 4, //bracelet walking
            OdmSportPlusExerciseModelTypeRopeSkipping = 5, //bracelet rope
skipping
            OdmSportPlusExerciseModelTypeSwimming = 6, //bracelet swimming
            OdmSportPlusExerciseModelTypeRun = 7, //bracelet running-outdoor
            OdmSportPlusExerciseModelTypeWalk = 8, //Bracelet hiking
            OdmSportPlusExerciseModelTypeBike = 9, //Bracelet cycling
            OdmSportPlusExerciseModelTypeExercise = 10, //Others
            OdmSportPlusExerciseModelTypeSwing = 11, //Bracelet swinging
            OdmSportPlusExerciseModelTypeClimb = 20, //Bracelet climbing
            OdmSportPlusExerciseModelTypeBadminton = 21, //Bracelet badminton
            OdmSportPlusExerciseModelTypeYoga = 22, //Bracelet yoga
            OdmSportPlusExerciseModelTypeAerobics = 23, //Bracelet aerobics
            OdmSportPlusExerciseModelTypeSpinningBike = 24, //Bracelet spinning
bike
            OdmSportPlusExerciseModelTypeKayaking = 25, // Bracelet kayaking
            OdmSportPlusExerciseModelTypeEllipticalMachine = 26, // Bracelet
elliptical machine
            OdmSportPlusExerciseModelTypeRowingMachine = 27, // Bracelet rowing
machine
            OdmSportPlusExerciseModelTypePingpong = 28, // Bracelet table tennis
            OdmSportPlusExerciseModelTypeTennis = 29, // Bracelet tennis
            OdmSportPlusExerciseModelTypeGolf = 30, // Bracelet golf
            OdmSportPlusExerciseModelTypeBasketball = 31, // Bracelet basketball
            OdmSportPlusExerciseModelTypeFootball = 32, // Bracelet football
            OdmSportPlusExerciseModelTypeVolleyball = 33, // Bracelet volleyball
            OdmSportPlusExerciseModelTypeRockClimbing = 34, // Bracelet rock
climbing
            OdmSportPlusExerciseModelTypeDance = 35, // Bracelet dance
            OdmSportPlusExerciseModelTypeRollerSkating = 36, // Bracelet roller
skating


            OdmSportPlusExerciseModelTypeTreadmill = 40, // Running-treadmill
            OdmSportPlusExerciseModelTypeIndoorWalking = 41, // Running-indoor
walking
            OdmSportPlusExerciseModelTypeTrailRunning = 42, // Running-cross-
country running
            OdmSportPlusExerciseModelTypeRaceWalk = 43, // Running-race walking
            OdmSportPlusExerciseModelTypePlaygroundRunning = 44, // Running -
playground running
            OdmSportPlusExerciseModelTypeFatLossRunning = 45, // Running - fat
loss running
```

```
            OdmSportPlusExerciseModelTypeOutdoorCycling = 50, // Cycling -
outdoor cycling
            OdmSportPlusExerciseModelTypeIndoorCycling = 51, // Cycling - indoor
cycling
            OdmSportPlusExerciseModelTypeMountainBiking = 52, // Cycling -
mountain biking
            OdmSportPlusExerciseModelTypeBMX = 53, // Cycling - BMX

            OdmSportPlusExerciseModelTypeSwimmingPool = 55, // Swimming -
swimming pool swimming
            OdmSportPlusExerciseModelTypeOutdoorSwimming = 56, // Swimming -
outdoor swimming
            OdmSportPlusExerciseModelTypeFinSwimming = 57, // Swimming-Fin
Swimming
            OdmSportPlusExerciseModelTypeSynchronizedSwimming = 58, // Swimming-
Synchronized Swimming

            OdmSportPlusExerciseModelTypeOutdoorHiking = 60, // Outdoor Sports-
Outdoor Hiking
            OdmSportPlusExerciseModelTypeOrienteering = 61, // Outdoor Sports-
Orienteering
            OdmSportPlusExerciseModelTypeFishing = 62, // Outdoor Sports-Fishing
            OdmSportPlusExerciseModelTypeHunt= 63, // Outdoor Sports-Hunting
            OdmSportPlusExerciseModelTypeSkateboard = 64, // Outdoor Sports-
Skateboarding
            OdmSportPlusExerciseModelTypeParkour = 65, // Outdoor Sports-Parkour
            OdmSportPlusExerciseModelTypeATV = 66, // Outdoor sports - ATV
            OdmSportPlusExerciseModelTypeMotocross = 67, // Outdoor sports -
Motocross
            OdmSportPlusExerciseModelTypeRacing = 68, // Outdoor sports - Racing
            OdmSportPlusExerciseModelTypeHandCrank = 69,
// Outdoor sports - hand bike
            OdmSportPlusExerciseModelTypeMarathon = 70, // Outdoor sports -
marathon
            OdmSportPlusExerciseModelTypeObstacleCourse = 71, // Outdoor sports -
obstacle course

            OdmSportPlusExerciseModelTypeStairClimber = 80, // Indoor sports -
stair climbing machine
            OdmSportPlusExerciseModelTypeStairStepper = 81, // Indoor sports -
stepper
            OdmSportPlusExerciseModelTypeMixedAerobic = 82, // Indoor sports -
mixed aerobics
            OdmSportPlusExerciseModelTypeKickboxing = 83, // Indoor sports -
kickboxing
            OdmSportPlusExerciseModelTypeCoreTraining = 84, // Indoor sports -
core training
            OdmSportPlusExerciseModelTypeCrossTraining = 85, // Indoor sports -
cross training
            OdmSportPlusExerciseModelTypeIndoorFitness = 86, // Indoor sports -
indoor fitness
            OdmSportPlusExerciseModelTypeGroupGymnastics = 87, // Indoor sports -
group gymnastics
            OdmSportPlusExerciseModelTypeStrengthTraining = 88, // Indoor sports
- strength training
```

```
            OdmSportPlusExerciseModelTypeGapTraining = 89, // Indoor sports - gap
training
            OdmSportPlusExerciseModelTypeFreeTraining = 90, // Indoor sports -
free training
            OdmSportPlusExerciseModelTypeFlexibilityTraining = 91, // Indoor
sports - flexibility training
            OdmSportPlusExerciseModelTypeGymnastics = 92, // Indoor sports -
gymnastics
            OdmSportPlusExerciseModelTypeStretch = 93, // Indoor exercise -
stretching
            OdmSportPlusExerciseModelTypePilates = 94, // Indoor exercise -
Pilates
            OdmSportPlusExerciseModelTypeHorizontalBar = 95, // Indoor exercise -
horizontal bar
            OdmSportPlusExerciseModelTypeParallelBars = 96, // Indoor exercise -
parallel bars
            OdmSportPlusExerciseModelTypeBattleRope = 97, // Indoor exercise -
battle rope
            OdmSportPlusExerciseModelTypeFitness = 98, // Indoor exercise -
fitness
            OdmSportPlusExerciseModelTypeBalanceTraining = 99, // Indoor exercise
- balance training
            OdmSportPlusExerciseModelTypeStepTraining = 100, // Indoor exercise -
step training

            OdmSportPlusExerciseModelTypeSquareDance = 110, // Dance sports-
square dance
            OdmSportPlusExerciseModelTypeBallroomDancing = 111, // Dance sports-
social dance
            OdmSportPlusExerciseModelTypeBellyDance = 112, // Dance sports-belly
dance
            OdmSportPlusExerciseModelTypeBallet = 113, // Dance sports-ballet
            OdmSportPlusExerciseModelTypeStreetDance = 114, // Dance sports-
street dance
            OdmSportPlusExerciseModelTypeZumba = 115, // Dance sports-Zumba
            OdmSportPlusExerciseModelTypeLatinDance = 116, // Dance sports-Latin
dance
            OdmSportPlusExerciseModelTypeLatinJazz = 117, // Dance sports-Jazz
dance
            OdmSportPlusExerciseModelTypeHipHopDance = 118, // Dance sports -
hip-hop dance
            OdmSportPlusExerciseModelTypePoleDancing = 119, // Dance sports -
pole dance
            OdmSportPlusExerciseModelTypeBreakDance = 120, // Dance sports -
break dance
            OdmSportPlusExerciseModelTypeFolkDance = 121, // Dance sports - folk
dance
            OdmSportPlusExerciseModelTypeNewDance = 122, // Dance sports - music
dance
            OdmSportPlusExerciseModelTypeModernDance = 123, // Dance sports -
modern dance
            OdmSportPlusExerciseModelTypeDisco = 124, // Dance sports - disco
            OdmSportPlusExerciseModelTypeTapDance = 125, // Dance sports - tap
dance
            OdmSportPlusExerciseModelTypeOtherDance = 126, // Dance sports -
other dances
```

```
            OdmSportPlusExerciseModelTypeBoxing = 130, // Combat sports - boxing
            OdmSportPlusExerciseModelTypeWrestling = 131, // Combat sports -
wrestling
            OdmSportPlusExerciseModelTypeMartialArts= 132, // Combat sports -
martial arts
            OdmSportPlusExerciseModelTypeTaiChi= 133, // Combat sports - Tai Chi
            OdmSportPlusExerciseModelTypeMuayThai= 134, // Combat sports - Muay
Thai
            OdmSportPlusExerciseModelTypeJudo= 135, // Combat sports - Judo
            OdmSportPlusExerciseModelTypeTaekwondo= 136, // Combat Sports -
Taekwondo
            OdmSportPlusExerciseModelTypeKarate= 137, // Combat Sports - Karate
            OdmSportPlusExerciseModelTypeFreeSparring= 138, // Combat Sports -
Free Fighting
            OdmSportPlusExerciseModelTypeSwordsmanship= 139, // Combat Sports -
Swordsmanship
            OdmSportPlusExerciseModelTypeJujitsu= 140, // Combat Sports - Jiu-
Jitsu
            OdmSportPlusExerciseModelTypeFencing= 141, // Combat Sports - Fencing
            OdmSportPlusExerciseModelTypeKendo= 142, // Combat Sports - Kendo

            OdmSportPlusExerciseModelTypeBeachFootball=150, //!<150Ball Sports -
Beach Soccer
            OdmSportPlusExerciseModelTypeBeachVolleyball=151, //!<151Ball Sports-
Beach Volleyball
            OdmSportPlusExerciseModelTypeBaseball=152, //!<152Ball Sports-
Baseball
            OdmSportPlusExerciseModelTypeSoftball=153, //!<153Ball Sports-
Softball
            OdmSportPlusExerciseModelTypeNewFootball=154, //!<154Ball Sports-
Rugby
            OdmSportPlusExerciseModelTypeHockey=155, //!<155Ball Sports-Hockey
            OdmSportPlusExerciseModelTypeSquash=156, //!<156Ball Sports-Squash
            OdmSportPlusExerciseModelTypeDoorKick=157, //!<157Ball Sports-
Gateball
            OdmSportPlusExercis
    eModelTypeCricket=158, //!<158Ball Sports-Cricket
            OdmSportPlusExerciseModelTypeHandball=159, //!<159Ball Sports-
Handball
            OdmSportPlusExerciseModelTypeBowling=160, //!<160Ball Sports-Bowling
            OdmSportPlusExerciseModelTypePolo=161, //!<161Ball Sports-Polo
            OdmSportPlusExerciseModelTypeRacquetball=162, //!<162Ball Sports-
Racquetball
            OdmSportPlusExerciseModelTypeBilliards=163, //!<163Ball Sports-Table
Tennis
            OdmSportPlusExerciseModelTypeTakraw=164, //!<164Ball Sports-Sepak
Rattan Ball
            OdmSportPlusExerciseModelTypeDodgeBall=165, //!<165Ball Sports-
Dodgeball
            OdmSportPlusExerciseModelTypeWaterPolo=166, //!<166Ball Sports-Water
Polo
            OdmSportPlusExerciseModelTypePuck=167, //!<167Ball Sports-Ice Hockey
            OdmSportPlusExerciseModelTypeShuttlecock=168, //!<168Ball Sports-
Shuttlecock
```

```
            OdmSportPlusExerciseModelTypeIndoorSoccer=169, //!<169Ball Sports-
Indoor Football
            OdmSportPlusExerciseModelTypeSandbag=170, //!<170Ball Sports-Sandbag
Ball
            OdmSportPlusExerciseModelTypeBocce=171, //!<171Ball Sports-Floor
Bocce
            OdmSportPlusExerciseModelTypeJaiBall=172, //!<172Ball Sports-Paiai
            OdmSportPlusExerciseModelTypeFloorBall=173, //!<173Ball Sports-
Floorball
            OdmSportPlusExerciseModelTypeAustralianRulesFootball=174, //!<174Ball
Sports-Australian Rules Football
            OdmSportPlusExerciseModelTypePickering=175, //!<175Ball Sports-
Pickering

            OdmSportPlusExerciseModelTypeOutdoorBoating=180, //!<180Water Sports-
Outdoor Rowing
            OdmSportPlusExerciseModelTypeSailing=181, //!<181Water Sports-Sailing
            OdmSportPlusExerciseModelTypeDragonBoat=182, //!<182 Water Sports-
Dragon Boat
            OdmSportPlusExerciseModelTypeSurf=183, //!<183 Water Sports-Surfing
            OdmSportPlusExerciseModelTypeKitesurfing=184, //!<184 Water Sports-
Kite Surfing
            OdmSportPlusExerciseModelTypePaddling=185, //!<185 Water Sports-
Paddling
            OdmSportPlusExerciseModelTypePaddleboard=186, //!<186 Water Sports-
Paddleboard Surfing
            OdmSportPlusExerciseModelTypeIndoorSurfing=187, //!<187 Water Sports-
Indoor Surfing
            OdmSportPlusExerciseModelTypeDrifting=188, //!<188 Water Sports-
Rafting
            OdmSportPlusExerciseModelTypeSnorkeling=189, //!<189 Water Sports -
Snorkeling

            OdmSportPlusExerciseModelTypeSkis=190, //!<190 Snow Sports - Double
Skiing
            OdmSportPlusExerciseModelTypeSnowboard=191, //!<191 Snow Sports -
Singleboarding
            OdmSportPlusExerciseModelTypeAlpineSkiing=192, //!<192 Snow Sports -
Alpine Skiing
            OdmSportPlusExerciseModelTypeCrossCountrySkiing=193, //!<193 Snow
Sports - Cross-Country Skiing
            OdmSportPlusExerciseModelTypeSkiOrientserlng=194, //!<194 Snow Sports
- Orienteering
            OdmSportPlusExerciseModelTypeBiathlon=195, //!<195 Snow Sports -
Biathlon
            OdmSportPlusExerciseModelTypeOutdoorSkating=196, //!<196Snow and ice
sports-outdoor skating
            OdmSportPlusExerciseModelTypeIndoorSkating=197, //!<197Snow and ice
sports-indoor skating
            OdmSportPlusExerciseModelTypeCurling=198, //!<198Snow and ice sports-
curling
            OdmSportPlusExerciseModelTypeBobsleigh=199, //!<199Snow and ice
sports-bobsleigh
            OdmSportPlusExerciseModelTypeSled=200, //!<200Snow and ice sports-
sled
```

```
            OdmSportPlusExerciseModelTypeSnowmobile=201, //!<201Snow and ice
sports-snowmobile
            OdmSportPlusExerciseModelTypeSnowshoeing=202, //!<202 Ice and Snow
Sports-Snowshoe Hiking

            OdmSportPlusExerciseModelTypeHulaHoop=210, //!<210 Leisure Sports-
Hula Hoop
            OdmSportPlusExerciseModelTypeFrisbee=211, //!<211 Leisure Sports-
Frisbee
            OdmSportPlusExerciseModelTypeDarts=212, //!<212 Leisure Sports-Darts
            OdmSportPlusExerciseModelTypeFlyAKite=213, //!<213 Leisure Sports-
Kite Flying
            OdmSportPlusExerciseModelTypeTugOfWar=214, //!<214 Leisure Sports-Tug
of War
            OdmSportPlusExerciseModelTypeEsports=215, //!<215 Leisure Sports-
Esports
            OdmSportPlusExerciseModelTypeStroller=216, //!<216 Leisure Sports-
Walking Machine
            OdmSportPlusExerciseModelTypeNewSwing=217, //!<217 Leisure Sports-
Swing
            OdmSportPlusExerciseModelTypeShuffleboard=218, //!<218 Leisure
Sports-Shuffleboard
            OdmSportPlusExerciseModelTypeTableSoccer=219, //!<219 Leisure Sports-
Table Football
            OdmSportPlusExerciseModelTypeSomatosensoryGame=220, //!<220 Leisure
Sports-Somatics
            OdmSportPlusExerciseModelTypeBungeeJumping=221, //!<221 Leisure
Sports-Bungee Jumping
            OdmSportPlusExerciseModelTypeParachute=222, //!<222 Leisure Sports-
Skydiving
            OdmSportPlusExerciseModelTypeAnusara=223, //!<223 Leisure Sports-
Anusara
            OdmSportPlusExerciseModelTypeYinYoga=224, //!<224 Leisure Sports-Yin
Yoga
            OdmSportPlusExerciseModelTypePregnancyY
```

2.3.6 OTA upgrade function:

```java
    //dfu upgrade instance
    val fuHandle= DfuHandle.getInstance()
    //initialize callback
    dfuHandle.initCallback()
    //DFU file verification, path firmware file path
    if (dfuHandle.checkFile(path)) {
        dfuHandle.start(dfuOpResult)
    }
    //dfuOpResult callback description
    private val dfuOpResult: DfuHandle.IOpResult = object : DfuHandle.IOpResult {
        override fun onActionResult(type: Int, errCode: Int) {
            if (errCode == DfuHandle.RSP_OK) {
                when (type) {
                    1 -> dfuHandle.init()
                    2 -> dfuHandle.sendPacket()
                    3 -> dfuHandle.check()
```

```
                    4 -> {
                        //The upgrade is successful, wait for the device to
restart
                        dfuHandle.endAndRelease()
                    }
                }
            } else {
                //Upgrade exception or failure
            }
        }

        override fun onProgress(percent: Int) {
            // file upgrade progress
        }
    }
```

2.3.7Manual measurement

```
//StartHeartRateRsp parameter description
private byte type; type 1: heart rate 2: blood pressure 3: blood oxygen
private byte errCode; measurement error code 0: normal 1: measurement failed 2:
measurement failed
private byte value; measurement value: heart rate or blood oxygen
private byte sbp; blood pressure sbp
private byte dbp; blood pressure dbph

Boolean stop  true stop false start

// manual heart rate
 BleOperateManager.getInstance().manualModeHeart(new
ICommandResponse<StartHeartRateRsp>() {
    @Override
    public void onDataResponse(StartHeartRateRsp resultEntity) {

    }
},Boolean stop);
        //manual blood pressure
        BleOperateManager.getInstance().manualModeBP(new
ICommandResponse<StartHeartRateRsp>() {
    @Override
    public void onDataResponse(StartHeartRateRsp resultEntity) {

    }
},Boolean stop);
        //manual blood oxygen
        BleOperateManager.getInstance().manualModeSpO2(new
ICommandResponse<StartHeartRateRsp>() {
    @Override
    public void onDataResponse(StartHeartRateRsp resultEntity) {

    }
},Boolean stop);
        //manual pressure
```

```
        BleOperateManager.getInstance().manualModePressure(new
ICommandResponse<StartHeartRateRsp>() {
    @Override
    public void onDataResponse(StartHeartRateRsp resultEntity) {

    }
},Boolean stop);
        //manual hrv
        BleOperateManager.getInstance().manualModeHrv(new
ICommandResponse<StartHeartRateRsp>() {
    @Override
    public void onDataResponse(StartHeartRateRsp resultEntity) {

    }
},Boolean stop);
        //manual temperature
        BleOperateManager.getInstance().manualTemperature(new
ICommandResponse<StartHeartRateRsp>() {
    @Override
    public void onDataResponse(StartHeartRateRsp resultEntity) {
        //The temperature value obtained should be divided by 10 to get the
normal temperature.
    }
},Boolean stop);
        //One-click measurement

        BleOperateManager.getInstance().oneClickMeasurement(new
ICommandResponse<StopHeartRateRsp>() {
    @Override
    public void onDataResponse(StopHeartRateRsp resultEntity) {

    }
},Boolean stop);


 BleOperateManager.getInstance().manualModeHeartRateRawData(ICommandResponse<Stop
HeartRateRsp> response,int seconds,boolean stop) (new
ICommandResponse<StopHeartRateRsp>() {
@Override
public void onDataResponse(StopHeartRateRsp resultEntity) {

}
        },Boolean stop);


 BleOperateManager.getInstance().manualModeBloodOxygenRawData(ICommandResponse<St
opHeartRateRsp> response,int seconds,boolean stop) (new
ICommandResponse<StopHeartRateRsp>() {
@Override
public void onDataResponse(StopHeartRateRsp resultEntity) {

}
        },Boolean stop);

//StopHeartRateRsp params
private int sbp;
```

```
private int dbp;
private int rri;
private int heart;
private int hrv;
private int stress;
private int temperature;

private int ppgCount;
private int greenLightPpgL;
private int greenLightPpgH;
private int XL;
private int XH;
private int YL;
private int YH;
private int ZL;
private int ZH;
private int heartRate;
private int redLightPpgL;
private int redLightPpgH;
private int infraredPpgL;
private int infraredPpgH;
private int bloodOxygen;
```

2.3.8 Touch and gestures

```
    read
touch：true  touch  false:gestures

 CommandHandle.getInstance().executeReqCmd(TouchControlReq.getReadInstance(false)
,
            ICommandResponse<TouchControlResp> {
    appType:  0:close  1:music 2:video 3:muslim 4:ebook 5:camera 6:phone call
7:game  8:heart
    strength: Dynamics
}

write:
appType:  0:close  1:music 2:video 3:muslim 4:ebook 5:camera 6:phone call 7:game
 8:heart 9:touch event
touch：true  touch  false:gestures
Strength:1-10

 CommandHandle.getInstance().executeReqCmdNoCallback(TouchControlReq.getWriteInst
ance(appType,false,currStrength))

        //Set RT11 Touch Function

 CommandHandle.getInstance().executeReqCmd(TouchControlReq.getWriteTpSleepInstanc
e(appType,1),null)
```

2.3.9 Changes in Ring measurement data are proactively reported to the APP

```
 //Add listener

BleOperateManager.getInstance().addOutDeviceListener(ListenerKey.Heart,myDeviceNo
tifyListener)

ListenerKey Parameter Description
public class ListenerKey {
    public static int Heart=1;          Heart
    public static int BloodPressure=2;  Blood Pressure
    public static int BloodOxygen=3;    Blood Oxygen
    public static int Temperature=5;    Temperature
    public static int SportRecord=7;    Sport Record
    public static int All=7;            All
}

Monitoring instructions
inner class MyDeviceNotifyListener : DeviceNotifyListener() {
    override fun onDataResponse(resultEntity: DeviceNotifyRsp?) {
        if (resultEntity!!.status == BaseRspCmd.RESULT_OK) {
            BleOperateManager.getInstance().removeOthersListener()
            when (resultEntity.dataType) {
                1 -> {
                    //Watch heart rate test changes
                }
                2 -> {
                    //Watch blood pressure test changes
                }
                3 -> {
                    //Watch blood oxygen test changes
                }
                4 -> {
                    //Changes in watch step counting details
                }
                5 -> {
                    //Watch body temperature changes on the day
                }
                7 -> {
                    //Generate new exercise records
                }
                0x0c -> {
                    //730c57010000000000000000000d7
                    val charging = BLEDataFormatUtils.bytes2Int(
                            byteArrayOf(
                                    resultEntity.loadData[2]
                            )
                    )
                    if (charging > 0) {
                        //charging
                    } else {
                        val battery = BLEDataFormatUtils.bytes2Int(
                                byteArrayOf(
```

```
                                        resultEntity.loadData[1]
                                )
                        )
                        //battery power

                    }
                }

                0x2d ->{
                    //The custom function button is triggered
                    val event = BLEDataFormatUtils.bytes2Int(
                            byteArrayOf(
                                    resultEntity.loadData[1]
                            )
                    )
                    event:0 null  1:decline  2: Slide up  3: Single  4: Long press

                }
                0x12 -> {
                    //7312 000052000251000003c0000000066
                    AwLog.i(Author.HeZhiYuan,
ByteUtil.bytesToString(resultEntity.loadData))

                    val step = BLEDataFormatUtils.bytes2Int(
                            byteArrayOf(
                                    resultEntity.loadData[1],
                                    resultEntity.loadData[2],
                                    resultEntity.loadData[3]
                            )
                    )

                    val calorie = BLEDataFormatUtils.bytes2Int(
                            byteArrayOf(
                                    resultEntity.loadData[4],
                                    resultEntity.loadData[5],
                                    resultEntity.loadData[6]
                            )
                    )
                    val distance = BLEDataFormatUtils.bytes2Int(
                            byteArrayOf(
                                    resultEntity.loadData[7],
                                    resultEntity.loadData[8],
                                    resultEntity.loadData[9]
                            )
                    )
                    deviceNotification(step, distance, calorie)
                }
                0x3A -> {//must add  Heart rate tooLow/High Reminder
Listener:ListenerKey.Heart

                    //BleOperateManager.getInstance()
                    //.addOutDeviceListener(ListenerKey.Heart,
myDeviceNotifyListener)

                    //Heart Rate too low/high Reminder
                    val type = BLEDataFormatUtils.bytes2Int(
```

```
                            byteArrayOf(
                                    resultEntity.loadData[1]
                            )
                    )
                    if (type == 1) {//too low

                    } else if (type == 2) {//too high

                    }
                    //Heart Rate value
                    val value = BLEDataFormatUtils.bytes2Int(
                            byteArrayOf(
                                    resultEntity.loadData[2]
                            )
                    )
                }
            }
        }
    }
}

//Remove the listener. It must be removed after registration, otherwise multiple
callbacks will appear.
    BleOperateManager.getInstance().removeNotifyListener(ListenerKey.Heart)
//Remove all listeners
    BleOperateManager.getInstance().removeNotifyListener(ListenerKey.All)
```

2.3.10 APP opens exercise type

```
 // status 1 Start movement 2 Pause 3 Continue 4 End 6 Movement start timestamp
//Sport type 4 Walking 5 Jumping rope 7 Running 8 Hiking 9 Cycling 10 Other
sports 20 Hiking 21 Badminton
     22 Yoga 23 Aerobics 24 Spinning 25 Kayaking 26 Elliptical machine 27 Rowing
machine 28 Table tennis 29 Tennis
     30 Golf 31 Basketball 32 Football 33 Volleyball 34 Rock climbing 35 Dance 36
Roller skating 60 Outdoor hiking
     CommandHandle.getInstance().executeReqCmd(
        PhoneSportReq.getSportStatus(
                1, sportType
        ), gpsResponse
            )

private var gpsResponse: ICommandResponse<AppSportRsp> =
        ICommandResponse<AppSportRsp> { resultEntity ->
        AwLog.i(Author.HeZhiYuan, resultEntity)
    if (resultEntity != null) {
        when (resultEntity.gpsStatus) {
            6 -> {
                //Exercise start time (Unit second)
            }

            2 -> {
                //Exercise pause
            }
```

```
            3 -> {
                // //Exercise continues
            }

            4 -> {
                //Exercise end
            }
            0x25 -> {
                //Muslim ring click real-time data
                //732500000013000700000000000000b2
                val count = BLEDataFormatUtils.bytes2Int(
                        byteArrayOf(
                                resultEntity.loadData[1],
                                resultEntity.loadData[2],
                                resultEntity.loadData[3],
                                resultEntity.loadData[4],
                                )
                )
            }
        }
    }


    //Report data during exercise
    //Add motion data reporting and monitoring
    BleOperateManager.getInstance().addSportDeviceListener(0x78,
myDeviceSportNotifyListener)
    //Remove sports data reporting monitoring
    BleOperateManager.getInstance().removeSportDeviceListener(0x78)
    //Listening to inner classes
    inner class MyDeviceNotifyListener : DeviceSportNotifyListener() {
        override fun onDataResponse(resultEntity: DeviceNotifyRsp?) {
            super.onDataResponse(resultEntity)

            if (resultEntity!!.status == BaseRspCmd.RESULT_OK) {
                //Movement duration, unit seconds
                val sportTime = bytes2Int(
                        byteArrayOf(
                                resultEntity.loadData[2],
                                resultEntity.loadData[3]
                        )
                )
                //Exercise real-time heart rate
                val heart = bytes2Int(
                        byteArrayOf(
                                resultEntity.loadData[4]
                        )
                )

                //The number of steps generated during exercise will only have
data when the exercise type is 4, 7, or 8, otherwise it will be 0
                val step = bytes2Int(
                        byteArrayOf(
                                resultEntity.loadData[5],
                                resultEntity.loadData[6],
                                resultEntity.loadData[7]
```

```
                        )
                )
                //The distance generated during exercise will only have data when
the exercise type is 4, 7, or 8, and the others will be 0.
                val distance = bytes2Int(
                        byteArrayOf(
                                resultEntity.loadData[8],
                                resultEntity.loadData[9],
                                resultEntity.loadData[10]
                        )
                )
                //Calories generated during exercise
                val calorie = bytes2Int(
                        byteArrayOf(
                                resultEntity.loadData[11],
                                resultEntity.loadData[12],
                                resultEntity.loadData[13]
                        )

                )

                //Error status during exercise
                val status = bytes2Int(
                        byteArrayOf(
                                resultEntity.loadData[1]
                        )
                )
                val sportType = BLEDataFormatUtils.bytes2Int(
                        byteArrayOf(
                                resultEntity.loadData[0]
                        )
                )
                if (status == 0x03) {
                    //Not wearing
                }
            }
        }
    }


    /**
     * Convert the byte array to int type, with the high byte of the array first
     *
     * @param data
     * @return
     */
    public static int bytes2Int(byte[] data) {
        int length = data.length;
        int res = 0;
        for (int i = 0; i < length; i++) {
            res |= (data[i] & 0xFF) << (8 * (length - 1 - i));
        }
        return res;
    }
```

Muslim Data Synchronization

```
//dayOffset 0: Today 1: Yesterday 2: The day before yesterday, supports
synchronization for up to 7 days
 CommandHandle.getInstance()
            .executeReqCmd(
        MuslimReq(0),
                ICommandResponse<MuslimRsp> {

})

//MuslimRsp
private int size = 0;
private int index = 0;
private byte[] pressureArray;  //Muslim data one data point per hour
private boolean endFlag = false;
private int range=60;
private DateUtil today;
private int offset=-1;



Real-time data reporting, ring single reporting, please refer to 2.3.9 Type 0x25
```

###

Setting Ring user Id

```
//read
 CommandHandle.getInstance()
    .executeReqCmd(
        PhoneIdReq.getReadInstance(),
object : ICommandResponse<PhoneIdRsp> {
    override fun onDataResponse(phoneIdRsp: PhoneIdRsp) {
        //userId:phoneIdRsp.getUserId()
    }
})
//write
        CommandHandle.getInstance()
        .executeReqCmd(
        PhoneIdReq.getWriteInstance("123456789123"),
object : ICommandResponse<PhoneIdRsp> {
    override fun onDataResponse(phoneIdRsp: PhoneIdRsp) {

    }
})
```

设置个人信息

```
CommandHandle.getInstance().executeReqCmd(
        TimeFormatReq.getWriteInstance(
                12,24小时切换，0为24 1为12，
        公英，0公 1英，
        男 true,女 false, age, height, weight, 120, 90, 160
), ICommandResponse<TimeFormatRsp>() {})
```