Android SDK Development Documentation

`InfoWear`

- Android SDK Development Documentation
  - Install
  - use
    - 1.API interface callback management
    - 1.Bluetooth first connection
    - 2.The first Bluetooth connection is successful (non-binding is successful)
    - 3.Features
      - 1.Synchronous Data
      - 2.set heart rate
      - 3.Set device language
      - 4.set unit
      - 5.Device notifications
      - 6.event reminder
      - 7.reminder function
        - 1.Alarm clock

Install

- First, under the dependencies of the project root build.gradle file,Add protobuf library：classpath 'com.google.protobuf:protobuf-gradle-plugin:0.8.8'；
- Then add under the main module, such as the build.gradle file under the app: apply plugin: 'com.google.protobuf';
- Also add under dependencies: implementation 'com.google.protobuf:protobuf-java:3.5.1';
- Then add at the end of the build.gradle file:

```
protobuf {
    //Configure the protoc compiler
    protoc {
```

```
        artifact = 'com.google.protobuf:protoc:3.5.1'
    }
    //The generation directory is configured here. After compilation, the corresponding java file will be generated in the build directory.
    generateProtoTasks {
        all().each { task ->
            task.builtins {
                remove java
            }
            task.builtins {
                java {}
            }
        }
    }
}
```

use

1.API interface callback management

- ControlBleTools: First initialize through ControlBleTools.getInstance().init (context, notifyTittle, notifyText, notifyAppRes, BleStateCallBack); the parameters are context, notification title, notification text, notification app icon resource id, Bluetooth connection status callback interface;
  `API interface callback operation mode` : The SDK uses the CallBackUtils class to add various callback information

1.Bluetooth first connection

- Bluetooth scanning entity class ScanDeviceBean

> `name` : bluetooth name
> `isBind` : Is it bound
> `isUserMode` : user mode
> `rssi` : Bluetooth signal value
> `address` : MAC address

> `hasBindingFunction` : Whether it is directly connected and bound, whether the device displays the confirmation binding interface
> `deviceType` : Equipment type
> `serviceDataString` : Bluetooth Service Characteristic Data

- Common standard parameter BleCommonAttributes

> `STATE_CONNECTED` = 2,///< connected
> `STATE_CONNECTING` = 1,///< connecting
> `STATE_DISCONNECTED` = 0,///< Disconnect
> `STATE_TIME_OUT` = 4,///< time out
> `CONNECT_TIMEOUT` = 10 * 1000,///< connection timeout
> `CONNECT_TIMES` = 3,/// `SCAN_PERIOD_BLE_SERVICE` = 3 * 1000,///< Bluetooth scan time
> `SEND_CMD_TIME_OUT` = 30 * 1000, /// `UPLOAD_BIG_DATA_OTA` = "ota",///
> `UPLOAD_BIG_DATA_WATCH` = "watch",///

- call method (Automatic reconnection has been done inside the SDK)

```
1. Search for bluetooth devices -ControlBleTools.getInstance().startScanDevice (ScanDeviceCallBack) ;
2. Stop searching for bluetooth devices  -ControlBleTools.getInstance().stopScanDevice();
3. connect bluetooth  -ControlBleTools.getInstance().connect(device.deviceName, device.deviceAddress);
4. disconnect bluetooth  - ControlBleTools.getInstance().disconnect(true);
```

- bluetooth callback

```
1. Bluetooth callback class, When sdk is initialized: ControlBleTools.getInstance().init(), register BleStateCallBack; the callback information refers to the public standard parameter BleCommonAttributes;
```

2.The first Bluetooth connection is successful (non-binding is

successful)

- Query binding status

```
ControlBleTools.getInstance().requestDeviceBindState();Add callback information:CallBackUtils.requestDeviceBindStateCallBack = new RequestDeviceBindStateCallBack() {
            @Override
            public void onBindState(boolean isBind) {
                Query the binding status callback result: isBind is false means the device is not bound, is true

                if (!isBind) {
                    //Not bound, initiate binding
                } else {
                //The device has been bound, to verify that the userid is the same
                }
```

- Not bound, initiate binding

> ControlBleTools.getInstance().bindDevice();

```
Initiate binding result callback:
CallBackUtils.bindDeviceStateCallBack = new BindDeviceStateCallBack() {
            @Override
            public void onDeviceInfo(String serialNumber, String deviceNumber, boolean deviceVerify, String firmwareVersion, String mac) {
                The device responds to the binding callback result: deviceVerify is true: the binding is successful, otherwise it fails;
                //At the same time, the sendAppBindResult method is called to send the account id to the device.
                ControlBleTools.getInstance().sendAppBindResult(DeviceCache.getUserId());
            }
        };
```

- The device has been bound, to verify that the userid is the same

> ControlBleTools.getInstance().verifyUserId(DeviceCache.getUserId());

```
Validation result callback：
CallBackUtils.verifyUserIdCallBack =
                new VerifyUserIdCallBack() {
                    @Override
                    public void onVerifyState(int state) {
                        if (state != 0) {
                            //If the userid of the verification result is inconsistent, it indicates that the user's device has been bound by another account.
                        } else {
                            //If the userid of the verification result is the same, it indicates that the user's device has been bound by the current account. Prompt that the connection is successful.
                        }
                    }
                };
```

- Query device MTU value

```
/// Query device MTU value
-ControlBleTools.getInstance().getMtu()
```

- The App sends the current time (the current time of the mobile phone) to the device and synchronizes it

```
/// The app sends time to synchronize to the device
/// @param  current timestamp
/// @param  Is it a 12-hour clock
/// @param  Send command callback
ControlBleTools.getInstance().setTime(System.currentTimeMillis(), is12, new ParsingStateManager.SendCmdStateListener(null) {
                @Override
                public void onState(SendCmdState sendCmdState) {

                }
            });
```

- App sets user basic information

```
/// The app sets basic user information
/// @param distanceUnit Distance unit Metric: 0x00; //Metric (distance); 0x01; //Imperial (distance)
```

```
/// @param temperatureUnit Temperature unit: 0x00; //Celsius; 0x01; //
Fahrenheit
/// @param UserInfo Set user basic information result
ControlBleTools.getInstance().setUserInformation ()
```

- UserInfo entity class

```
public class UserInfo implements Serializable {
    public int height;//Height (cm cm)
    public float weight; //Weight (kg kg, accurate to two decimal
places, 120.15KG)
    public int birthday;//Birthday, timestamp, hour, minute, second, al
l 0
    public int sex;   //Male 0x01; //Female 0x02;
    public int age;
    public int maxHr;//Maximum heart rate (beats/min)
    public int calGoal;//Target Calories (kcal)
    public int stepGoal;//Target steps (steps)
    public int distanceGoal;//Target distance (m)
    public int standingTimesGoal;//Target effective standing times
(times)
    public int goalSleepMinute;//Target sleep duration (minutes)
}
```

3.Features

1.Synchronous Data

- App actively obtains daily data

> ControlBleTools.getInstance().getDailyHistoryData();

```
Daily data callback
CallBackUtils.fitnessDataCallBack = new FitnessDataCallBack() {
            @Override
            public void onProgress(int progress, int total) {
                //progress callback progress, total total;
            }

            @Override
            public void onDailyData(DailyBean dailyBean) {
                //Steps, calories, distance data throughout the day
```

```
            }

            @Override
            public void onSleepData(SleepBean sleepBean) {
               //All day sleep data
            }

            @Override
            public void
onContinuousHeartRateData(ContinuousHeartRateBean
continuousHeartRateBean){
                //Continuous heart rate data
            }

            @Override
            public void onOfflineHeartRateData(OfflineHeartRateBean off
lineHeartRateBean) {
                //Heart rate data for manual testing
            }

            @Override
            public void
onContinuousBloodOxygenData(ContinuousBloodOxygenBean
continuousBloodOxygenBean) {
                //Continuous blood oxygen data
            }

            @Override
            public void onOfflineBloodOxygenData(OfflineBloodOxygenBean
offlineBloodOxygenBean) {
                //Blood oxygen data for manual tests
            }

            @Override
            public void onContinuousPressureData(ContinuousPressureBean
continuousPressureBean) {

            }

            @Override
            public void onOfflinePressureData(OfflinePressureDataBean o
fflinePressureDataBean) {

            }

```

```
            @Override
            public void onContinuousTemperatureData(ContinuousTemperatureBean continuousTemperatureBean) {
            }

            @Override
            public void onOfflineTemperatureData(OfflineTemperatureDataBean offlineTemperatureDataBean) {

            }

            @Override
            public void onEffectiveStandingData(EffectiveStandingBean effectiveStandingBean) {
                //Valid standing data
            }

            @Override
            public void onActivityDurationData(ActivityDurationBean activityDurationBean) {
                //activity duration data
            }
        };
```

- DailyBean daily data entity class

```
public class DailyBean extends BaseBean implements Serializable {
    public int stepsFrequency;//Step frequency
    public List<Integer> stepsData;//step data
    public int distanceFrequency;//distance frequency
    public List<Integer> distanceData;//distance data
    public int calorieFrequency;//Calorie Frequency
    public List<Integer> calorieData;//Calorie data
}
```

- SleepBean sleep data entity class

```
public class SleepBean extends BaseBean implements Serializable {
    public long startSleepTimestamp;//Total sleep onset time: time to fall asleep
    public long endSleepTimestamp; //Total Sleep End Time: Wake Time
    public int sleepDuration; //Total Sleep Duration: Length of Sleep
```

```
    public int sleepScore;//sleep score
    public int awakeTime; //total waking time :
    public int awakeTimePercentage;//% of total waking hours :
    public int lightSleepTime;//total light sleep duration :
    public int lightSleepTimePercentage; //% of total light sleep time
:
    public int deepSleepTime;//total deep sleep time :
    public int deepSleepTimePercentage;//% of total deep sleep time :
    public int rapidEyeMovementTime;//Total REM duration :
    public int rapidEyeMovementTimePercentage;//% of total REM
duration :
    public boolean isNightSleep;//sleep type
    public List<SleepDistributionData> list = new ArrayList<>();

    public static class SleepDistributionData implements Serializable {
        public long  startTimestamp; //start time of sleep
        public int sleepDuration; //sleep duration
        public int  sleepDistributionType; //sleep distribution type
    }
}
```

- ContinuousHeartRateBean Continuous heart rate entity class

```
public class ContinuousHeartRateBean extends BaseBean implements
Serializable {
    public int continuousHeartRateFrequency;//continuous heart rate
    public List<Integer> heartRateData;//Continuous heart rate data
    public int max;
    public int min;
}
```

- OfflineHeartRateBean Heart rate entity class for manual testing

```
public class OfflineHeartRateBean extends BaseBean implements
Serializable {
    public List<MeasureData> list = new ArrayList<>();
    public static class MeasureData implements Serializable {
        public  int measureTimestamp; //time of measurement
        public  int measureData ; //Measurement data
    }
}
```

- ContinuousBloodOxygenBean Continuous blood oxygen entity class

```
public class ContinuousBloodOxygenBean extends BaseBean implements
Serializable {
    public int bloodOxygenFrequency;//continuous blood oxygen
frequency
    public List<Integer> bloodOxygenData;//Continuous blood oxygen
data
}
```

- ContinuousBloodOxygenBean Manually tested blood oxygen entity class

```
public class OfflineBloodOxygenBean extends BaseBean implements
Serializable {
    public List<OfflineBloodOxygenBean.MeasureData> list = new ArrayLis
t<>();
    public static class MeasureData implements Serializable {
        public  int measureTimestamp; //time of measurement
        public  int measureData ; //Measurement data
    }
}
```

- EffectiveStandingBean Valid standing entity class

```
public class EffectiveStandingBean extends BaseBean implements
Serializable {
    public int effectiveStandingFrequency; //effective standing
frequency
    public List<Integer> effectiveStandingData; //Valid standing data
}
```

- ActivityDurationBean Activity duration entity class

```
public class ActivityDurationBean extends BaseBean implements
Serializable {
    public int activityDurationFrequency; //activity duration
frequency
    public List<Integer> activityDurationData; //activity duration
data
}
```

- The device actively refreshes real-time data

App sets the device to start uploading real-time basic data

```
-(void)deviceStartUplaodBasicData;
```

App sets the device to stop uploading real-time basic data

```
-(void)deviceStopUplaodBasicData;
```

The device actively reports real-time basic data

```
//Device active upload (total steps, calories, distance, heart rate, etc.)
-(void)deviceUploadMessageDidReviceBasicData:(ZHBasicData *)basicData;
```

2.set heart rate

> **ControlBleTools.getInstance().setHeartRateMonitor ( )**

```
Heart rate parameter entity class
public class HeartRateMonitorBean implements Serializable {
    public int mode;           // mode 0 on 1 off
    public int frequency;      //Intervals
    public boolean isWarning;  //Whether there is a heart rate warning
    public int warningValue;   // Heart rate warning value

    public HeartRateMonitorBean() {
    }

    public HeartRateMonitorBean(int mode, int frequency, boolean isWarning, int warningValue) {
        this.mode = mode;
        this.frequency = frequency;
        this.isWarning = isWarning;
        this.warningValue = warningValue;
    }

    public HeartRateMonitorBean(SettingMenuProtos.HeartRateMonitor rateMonitor) {
        mode = rateMonitor.getMode().getNumber();
        frequency = rateMonitor.getFrequency();
        isWarning = rateMonitor.getWarning();
        warningValue = rateMonitor.getWarningValue();
    }
```

```
}
```

App gets heart rate setting parameters

```
ControlBleTools.getInstance().getHeartRateMonitor ()
```

3.Set device language

```
Device language ID corresponding value
    enum LanguageId {
    ALBANIAN = 0X01; //Albanian
    ARABIC = 0X02; //Arabic
    AM_HARIC= 0X03; //Amharic
    IRISH = 0X04; //Irish
    ORIYA = 0X05; //Oriya
    BASQUE = 0X06; //Basque
    BELARUSIAN = 0X07; //Belarusian
    BULGARIAN = 0X08; //Bulgarian
    POLISH = 0X09; //Polish
    PERSIAN = 0X10; //Persian
    BOOLEAN = 0X11; //Boolean
    DANISH = 0X12; //Danish
    GERMAN = 0X13; //German
    RUSSIAN = 0X14; //Russian
    FRENCH = 0X15; //French
    FILIPINO = 0X16; //Filipino
    FINNISH = 0X17; //Finnish
    CAMBODIAN = 0X18; //Cambodian
    GEORGIAN = 0X19; //Georgian
    GUJARATI = 0X20; //Gujarati
    KAZAKH = 0X21; //Kazakh
    JAITAN_CREOLE = 0X22; //Haitian Creole
    KOREAN = 0X23; //Korean
    DUTCH = 0X24; //Dutch
    GALICIAN = 0X25; //Galician
    CATALAN = 0X26; //Catalan
    CZECH = 0X27; //Czech
    KANNADA = 0X28; //Kannada
    CROATIAN = 0X29; //Croatian
    KURDISH = 0X30; //Kurdish
    LATIN = 0X31; //Latin
    LAO = 0X32; //Lao
    KINYARWANDA = 0X33; //Rwandan
```

```
    ROMANIAN = 0X34; //Romanian
    MALAGASY = 0X35; //Malagasy
    MARATHI = 0X36; //Marathi
    MALAYALAM = 0X37; //Malayalam
    MALAY = 0X38; //Malay
    MONGOLIAN = 0X39; //Mongolian
    BENGALI = 0X40; //Bengali
    BURMESE = 0X41; //Burmese
    HMONG = 0X42; //Hmong language
    ZULU_SOYTH_AFRICA = 0X43; //Zulu, South Africa
    NEPALI = 0X44; //Nepali
    NORWEGIAN = 0X45; //Norwegian
    PORTUGUESE = 0X46; //Portuguese
    JAPANESE = 0X47; //Japanese
    SWEDISH = 0X48; //Swedish
    SERBIAN = 0X49; //Serbian
    SINHALA = 0X50; //Sinhala
    SLOVAK= 0X51; //Slovak
    SOMALI = 0X52; //Somali
    TAJIK = 0X53; //Tajik
    TELUGU = 0X54; //Telugu
    TAMIL = 0X55; //Tamil
    THAI = 0X56; //Thai
    TURKISH = 0X57; //Turkish
    URDU = 0X58; //Urdu
    UKRAINIAN = 0X59; //Ukrainian
    UZBEK = 0X60; //Uzbek
    SPANISH = 0X61; //Spanish
    GREEK = 0X62; //Greek
    HUNGARIAN = 0X63; //Hungarian
    IGBO = 0X64; //Igbo
    ITALIAN = 0X65; //Italian
    HINDI= 0X66; //Hindi
    INDONESIAN = 0X67; //Indonesian
    ENGLISH = 0X68; //English
    VIETNAMESE = 0X69; //Vietnamese
    TRADITIONAL_CHINESE = 0X70; //traditional Chinese
    SIMPLIFIED_CHINESE = 0X71; //Simplified Chinese
}
```

- App sets device language

```
/// The app sends the language type to the device
/// @param languageId language id
```

```
ControlBleTools.getInstance().setLanguage()
```

- App gets the list of languages supported by the device

```
/// Get the list of languages supported by the device
ControlBleTools.getInstance().getLanguageList()
Callback language information
CallBackUtils.languageCallback = new LanguageCallBack() {
                @Override
                public void onResult(LanguageListBean languageListBean)
{
                    //Query device language results
                }
            };

language entity class
public class LanguageListBean implements Serializable {
    public int selectLanguageId;//The currently selected language id o
f the device
    public int defaultLanguageId;//device default language id
    public List<Integer> languageList = new ArrayList<>();//List of lan
guage ids supported by the device
}
```

4.set unit

- set metric imperial

```
/// Set metric and imperial (such as distance, weight, etc.)
/// @param distanceUnit Unit Type: 0, // Metric; 1, // Imperial
ControlBleTools.getInstance().setDistanceUnit()
```

- Set temperature units

```
/// Temperature unit setting
/// @param weatherUnit Temperature units 0, // Celsius; 1, //
Fahrenheit
ControlBleTools.getInstance().setTemperatureUnit()
```

5.Device notifications

- Send notification to device

```
/// @param appName Third-party app name
/// @param pageName Third-party app package name
/// @param title title
/// @param text content
/// @param tickerText
ControlBleTools.getInstance().sendAppNotification()
```

- Send incoming calls, missed calls, SMS notifications to the device

```
/// @param type     0x00; //Incoming call. 0x01; //Missed call. 0x02; //Short message
/// @param phoneNumber //Phone number
/// @param contactsInfo //Query the contact name, mobile phone number/name according to the address book
/// @param messageText //When there is only "MESSAGE", use
ControlBleTools.getInstance().sendSystemNotification()
```

6.event reminder

- Get device event reminders

```
ControlBleTools.getInstance().getEventInfoList()
```

- Set device event reminders

```
/// Set device event reminder parameters
/// @param List<EventInfoBean> list
ControlBleTools.getInstance().setEventInfoList()
Event reminder parameter entity class
public class EventInfoBean implements Serializable {
    public String description;  //Event description
    public TimeBean time;       //event reminder time

    public EventInfoBean(){

    }

    public EventInfoBean(String description, TimeBean time) {
        this.description = description;
        this.time = time;
    }

    public EventInfoBean(SettingMenuProtos.EventInfo eventInfo){
```

```
            description = eventInfo.getDescription();
            time = new TimeBean(eventInfo.getTime());
        }
    }
```

7.reminder function

1.Alarm clock

- Get device alarm

```
/// Get device alarm reminder parameters
ControlBleTools.getInstance().getClockInfoList()
```

- Set device alarm

```
/// Set device alarm reminder parameters
/// @param List<ClockInfoBean> list
ControlBleTools.getInstance().setClockInfoList()

Parameter alarm clock reminder entity class
public class ClockInfoBean implements Serializable {
    public int id = -1; //list
    public DataBean data;

    public ClockInfoBean() {
    }

    public ClockInfoBean(int id, DataBean data) {
        this.id = id;
        this.data = data;
    }

    public ClockInfoBean(SettingMenuProtos.ClockInfo clockInfo) {
        id = clockInfo.getId();
        data = new DataBean(clockInfo.getData());
    }

    public static class DataBean implements Serializable{
        public SettingTimeBean time;
        public int weekDays;            //The weekday of the alarm clock, 1bit represents a week
        public boolean isEnable;        //Is it on
        public String clockName;        //alarm clock name

```

```
29.          public boolean isMonday;//Does it repeat on monday
30.          public boolean isTuesday;//Does it repeat on Tuesday and so on
31.          public boolean isWednesday;
32.          public boolean isThursday;
33.          public boolean isFriday;
34.          public boolean isSaturday;
35.          public boolean isSunday;
36.
37.          public DataBean() {
38.          }
39.
40.          public DataBean(SettingTimeBean time, boolean isMonday,
41.                          boolean isTuesday, boolean isWednesday, boolean
     isThursday,
42.                          boolean isFriday, boolean isSaturday, boolean i
     sSunday,
43.                          boolean isEnable, String clockName) {
44.              this.time = time;
45.              this.isMonday = isMonday;
46.              this.isTuesday = isTuesday;
47.              this.isWednesday = isWednesday;
48.              this.isThursday = isThursday;
49.              this.isFriday = isFriday;
50.              this.isSaturday = isSaturday;
51.              this.isSunday = isSunday;
52.              this.isEnable = isEnable;
53.              this.clockName = clockName;
54.              this.weekDays = BleUtils.getBinaryValue(false,isSunday,isSa
     turday,isFriday,isThursday,isWednesday,isTuesday,isMonday);
55.          }
56.
57.          public void calculateWeekDays(){
58.              this.weekDays = BleUtils.getBinaryValue(false,isSunday,isSa
     turday,isFriday,isThursday,isWednesday,isTuesday,isMonday);
59.          }
60.
61.          public void analysisWeekDays(int weekDaysData){
62.              this.weekDays = weekDaysData;
63.              this.isMonday = BleUtils.getBinaryBit(weekDays,0);
64.              this.isTuesday = BleUtils.getBinaryBit(weekDays,1);
65.              this.isWednesday = BleUtils.getBinaryBit(weekDays,2);
66.              this.isThursday = BleUtils.getBinaryBit(weekDays,3);
67.              this.isFriday = BleUtils.getBinaryBit(weekDays,4);
68.              this.isSaturday = BleUtils.getBinaryBit(weekDays,5);
69.              this.isSunday = BleUtils.getBinaryBit(weekDays,6);
```

```
70.             }
71.
72.             public DataBean(SettingMenuProtos.ClockInfo.Data data) {
73.                 this.time = new SettingTimeBean(data.getTime());
74.                 analysisWeekDays(data.getWeekDays());
75.                 this.isEnable = data.getEnable();
76.                 this.clockName = data.getClockName();
77.             }
78.         }
79.     }
```