|  |
| --- |
|  |
| **P102T/P103T Android SDK Description** |
| **V1.0** |
|  |
| **Author: WangYong** |
| **2021/10/09** |

# Change log

|  |  |  |
| --- | --- | --- |
| **Date** | **Log** | **Author** |
| 2021-10-09 | Create | WangYong |
| 2024-10-28 | 1. Fix data3 error in historical data details  2. Update the intelligent selection algorithm version to 1.0.39  3. Optimize the code structure and remove baseautopillowsdk. jar  4. Increase real-time data listener, registRealtimeDataListener and unregistRealtimeDataListener | WangYong |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |
| --- |
|  |

# Catalog

[Change log 1](#_Toc85204240)

[Catalog 2](#_Toc85204241)

[Android SDK Intro 6](#_Toc85204242)

[1. Function and Purpose 6](#_Toc85204243)

[Integration 6](#_Toc85204244)

[1 .SDK framework 6](#_Toc85204245)

[2 .Integration 6](#_Toc85204246)

[Eclipse Config 6](#_Toc85204247)

[API 8](#_Toc85204248)

[1.API initialization 8](#_Toc85204249)

[Description 8](#_Toc85204250)

[Parameters 8](#_Toc85204251)

[2. Connnect Device 8](#_Toc85204252)

[Description 8](#_Toc85204253)

[Parameters 8](#_Toc85204254)

[3. Get Battery 9](#_Toc85204255)

[Description 9](#_Toc85204256)

[Parameters 9](#_Toc85204257)

[4. Get Device Version 9](#_Toc85204258)

[Description 9](#_Toc85204259)

[Parameters 9](#_Toc85204260)

[5. Query device collection status 10](#_Toc85204261)

[Description 10](#_Toc85204262)

[Parameters 10](#_Toc85204263)

[6. Stop Monitoring/Collecting 10](#_Toc85204264)

[Description 10](#_Toc85204265)

[Parameters 10](#_Toc85204266)

[7. Get Sleep Data (Real-time) 10](#_Toc85204267)

[Description 10](#_Toc85204268)

[Parameters 11](#_Toc85204269)

[8. Stop Getting Sleep Data 11](#_Toc85204270)

[Description 11](#_Toc85204271)

[Parameters 11](#_Toc85204272)

[9. Get The Signal Strength (Real-time) 11](#_Toc85204273)

[Description 11](#_Toc85204274)

[Parameters 11](#_Toc85204275)

[10. Stop Getting The Signal Strength 11](#_Toc85204276)

[Description 12](#_Toc85204277)

[Parameters 12](#_Toc85204278)

[11. Get Sleep Report 12](#_Toc85204279)

[Description 12](#_Toc85204280)

[Parameters 12](#_Toc85204281)

[12. Firmware Update 1 12](#_Toc85204282)

[Description 12](#_Toc85204283)

[Parameters 12](#_Toc85204284)

[13. Firmware Update 2 13](#_Toc85204285)

[Description 13](#_Toc85204286)

[Parameters 13](#_Toc85204287)

[14. Wait for device confirmation 13](#_Toc85204288)

[Description 13](#_Toc85204289)

[Parameters 14](#_Toc85204290)

[15. Monitor the status messages reported by the device 14](#_Toc85204291)

[Description 14](#_Toc85204292)

[Parameters 14](#_Toc85204293)

[16. Log off the status message listening reported by the device 14](#_Toc85204294)

[Description 14](#_Toc85204295)

[Parameters 14](#_Toc85204296)

[Object Description 15](#_Toc85204297)

[StatusCode 15](#_Toc85204298)

[Description 15](#_Toc85204299)

[Fields 15](#_Toc85204300)

[IResultCallback<T> 15](#_Toc85204301)

[Description 15](#_Toc85204302)

[Function 15](#_Toc85204303)

[CallbackData<T> 16](#_Toc85204304)

[Description 16](#_Toc85204305)

[Fields 16](#_Toc85204306)

[LoginBean 16](#_Toc85204307)

[Description 16](#_Toc85204308)

[Fields 16](#_Toc85204309)

[BatteryBean 16](#_Toc85204310)

[Description 16](#_Toc85204311)

[Fields 16](#_Toc85204312)

[CollectStatus 17](#_Toc85204313)

[Description 17](#_Toc85204314)

[Fields 17](#_Toc85204315)

[RealTimeData 17](#_Toc85204316)

[Description 17](#_Toc85204317)

[Fields 17](#_Toc85204318)

[OriginalData 18](#_Toc85204319)

[Description 18](#_Toc85204320)

[Fields 18](#_Toc85204321)

[HistoryData 18](#_Toc85204322)

[Description 18](#_Toc85204323)

[Fields 18](#_Toc85204324)

[Summary 18](#_Toc85204325)

[Description 18](#_Toc85204326)

[Fields 19](#_Toc85204327)

[Detail 19](#_Toc85204328)

[Description 19](#_Toc85204329)

[Fields 19](#_Toc85204330)

[Analysis 20](#_Toc85204331)

[Description 20](#_Toc85204332)

[Fields 20](#_Toc85204333)

[SleepStatusType 25](#_Toc85204334)

[Description 25](#_Toc85204335)

[Fields 25](#_Toc85204336)

[DeviceStateListener 25](#_Toc85204337)

[Description 25](#_Toc85204338)

[Function 25](#_Toc85204339)

[DeviceState 26](#_Toc85204340)

[Description 26](#_Toc85204341)

[Fields 26](#_Toc85204342)

# Android SDK Intro

## Function and Purpose

The sleeppace Android SDK is a software development kit for rapid Internet of things app development on the Android platform launched by sleeppace. The SDK encapsulates the communication process between app and hardware, and provides external functions such as device configuration, device control and data query. Using the SDK, users do not need to care about complex communication protocols and underlying implementation, but only need to focus on the interaction and business level of the app.

# Integration

## 1 .SDK framework

|  |  |
| --- | --- |
| **Framework** | **Description** |
| sdkcore.jar | SDK base library |
| p102tsdk.jar | p102t library |
| sdkalgorithm.jar | Algorithm library |
| libalgorithm.so | Algorithm library |

## 2 .Integration

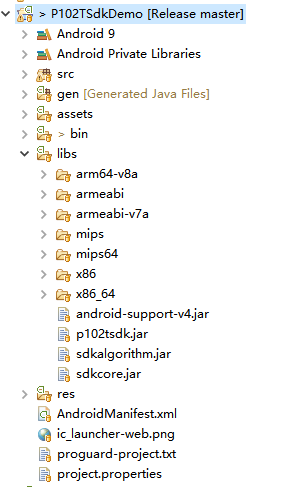
There are many Android development tools, and here we introduce the engineering configuration method of Sleepace SDK with Eclipse.

## Eclipse Config

**Step 1**：

In the project to create a "libs" folder, copy sdkcore.jar，p102tsdk.jar，sdkalgorithm.jar to "libs" folder, copy libalgorithm.so to "libs \ armeabi" folder.

Like this:



**Setp 2:**

Config the “AndroidManifest.xml”

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

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

<uses-feature android:name=*"android.hardware.bluetooth\_le"* android:required=*"true"*/>

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

# API

## 1.API initialization

P102THelper.getInstance(Context mContext);

### Description

P102THelper Initialization

### Parameters

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| mContext | Context | Conetxt |

## Connnect Device

**public** **void** login(String deviceName, String address, DeviceType deviceType, **int** userId, **int** timeout, IResultCallback<LoginBean> cb)

### Description

Connect Pillow and setting userId, Before connecting the device, the MTU of the device can be set using the setMTU method. P102T supports MTU, while P103T does not support MTU. For specific usage, please refer to the demo.

### Parameters

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| deviceName | String | deviceName |
| address | String | Bluetooth address |
| deviceType | enum | The device type depends on the access device. Refer to the DeviceType enum. The current device type is:  DEVICE\_ TYPE\_ P102T  DEVICE\_ TYPE\_ P103T |
| userId | int | userId does not belong to Sleepace.  userId belong to partner  **Why need it：**  Pillow separates the data according to userId.  It mean user A connects to device, generates and gets the data which only belong to user A. User A can’t get the data of user B |
| timeout | int | Timeout, Unit(Millisecond) |
| cb | [IResultCallback](#_IDataCallback<T>)<[LoginBean](#_LoginBean)> | Callback function, if success,return [LoginBean](#_LoginBean) Obj |

## Get Battery

**public** **void** getBattery(**int** timeout, IResultCallback<BatteryBean> cb)

### Description

Get battery

### Parameters

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| timeout | int | Timeout, Unit(Millisecond) |
| cb | [IResultCallback](#_IDataCallback<T>)<[BatteryBean](#_BatteryBean)> | Callback function, if success,return [BatteryBean](#_BatteryBean) Object |

## Get Device Version

**public** **void** getDeviceVersion(**int** timeout, IResultCallback<String> cb)

### Description

Get current version of device

### Parameters

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| timeout | int | Timeout, Unit(Millisecond) |
| cb | [IResultCallback](#_IDataCallback<T>)<String> | Callback function, if success,return the version of device |

## Query device collection status

**public** **void** queryCollectStatus(**int** timeout, IResultCallback<[CollectStatus](#_CollectStatus)> cb)

### Description

It is used to query whether the device is currently being monitored and how long it has been monitored

### Parameters

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| timeout | int | Timeout, Unit(Millisecond) |
| cb | [IResultCallback](#_IDataCallback<T>)<[CollectStatus](#_CollectStatus)> | Callback function |

## Stop Monitoring/Collecting

**public** **void** stopCollection(**int** timeout, IResultCallback<Void> cb)

### Description

Stop Monitoring/Collecting

### Parameters

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| timeout | int | Timeout, Unit(Millisecond) |
| cb | IResultCallback<Void> | Callback function |

## Get Sleep Data (Real-time)

**public** **void** startRealTimeData(**int** timeout, IResultCallback<Void> cb)

### Description

Get Real-time Data

### Parameters

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| timeout | int | Timeout, Unit(Millisecond) |
| cb | [IResultCallback](#_IDataCallback<T>)<Void> | Callback function |

## Stop Getting Sleep Data

**public** **void** stopRealTimeData(**int** timeout, IResultCallback<Void> cb)

### Description

Stop getting real-time data

### Parameters

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| timeout | int | Timeout, Unit(Millisecond) |
| cb | [IResultCallback](#_IDataCallback<T>)<Void> | Callback function |

## Get The Signal Strength (Real-time)

**public** **void** startOriginalData(**int** timeout, IResultCallback<OriginalData> cb)

### Description

Get the signal strength

### Parameters

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| timeout | int | Timeout, Unit(Millisecond) |
| cb | [IResultCallback](#_IDataCallback<T>)<[OriginalData](#_OriginalData)> | Callback function |

## Stop Getting The Signal Strength

**public** **void** stopOriginalData(**int** timeout, IResultCallback<Void> cb)

### Description

Stop Getting The Signal Strength

### Parameters

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| timeout | int | Timeout, Unit(Millisecond) |
| cb | [IResultCallback<Void>](#_IDataCallback<T>) | Callback function |

## Get Sleep Report

**public** **void** historyDownload(**int** startTime, **int** endTime, **int** sex, IResultCallback<List<HistoryData>> cb)

### Description

Get sleep report

### Parameters

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| starTime | int | Start time(timestamp), Unit(second) |
| endTime | int | End time(timestamp), Unit(second) |
| sex | int | Gender,1:male 0:female |
| timeout | int | Timeout, Unit(Millisecond) |
| cb | [IResultCallback](#_IDataCallback<T>)<List<[HistoryData](#_HistoryData)>> | Callback function |

## Firmware Update 1

**public** **void** upgradeDevice(**final** String pKey, **final** String hashCode, **final** File file, **final** IResultCallback<Integer> cb)

### Description

Firmware Update

### Parameters

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| pKey | String | Get it from Sleepace |
| hashCode | String | Get it from Sleepace |
| file | File | Firmware object |
| timeout | int | Timeout, Unit(Millisecond) |
| cb | [IResultCallback](#_IDataCallback<T>)<Integer> | Callback function, Return upgrade progress |

## Firmware Update 2

**public** **void** upgradeDevice(**final** String pKey, **final** String hashCode, **final** InputStream is, **final** IResultCallback<Integer> cb)

### Description

Firmware Update

### Parameters

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| pKey | String | Get it from Sleepace |
| hashCode | String | Get it from Sleepace |
| is | InputStream | File data stream |
| timeout | int | Timeout, Unit(Millisecond) |
| cb | [IResultCallback](#_IDataCallback<T>)<Integer> | Callback function, Return upgrade progress |

## Wait for device confirmation

**public** **void** waitConfirm(IResultCallback cb)

### Description

It is used to open the device confirmation notification. After calling the interface successfully, press the power button on the device twice continuously, and the device will report the status message 0xf0. For specific usage, refer to demo

Application scenario: when the user binds the device, in order to prevent the wrong device from being bound, call the interface, then press the power button on the device twice, and bind after the app receives the confirmation notice. (this step is not necessary, and the customer determines whether it is necessary according to his own needs)

### Parameters

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| cb | IResultCallback | Callback function |

## Register real-time data listener

**public** **void** registRealtimeDataListener([RealtimeDataListener](#_RealtimeDataListener) realtimeDataListener)

### Description

Used to monitor real-time data

### Parameters

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| [RealtimeDataListener](#_RealtimeDataListener) | Custom interface | Real-time data monitoring, refer to: [RealtimeDataListener](#_RealtimeDataListener) |

## unregister real-time data listener

**public** **void** unregistRealtimeDataListener([RealtimeDataListener](#_RealtimeDataListener) realtimeDataListener)

### Description

Used to log off real-time data monitoring

### Parameters

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| [RealtimeDataListener](#_RealtimeDataListener) | Custom interface | Real-time data monitoring, refer to: [RealtimeDataListener](#_RealtimeDataListener) |

## Monitor the status messages reported by the device

**public** **void** addDeviceStateListener([DeviceStateListener](#_DeviceStateListener) deviceStateListener)

### Description

Used to add device status message listening

### Parameters

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| deviceStateListener | [DeviceStateListener](#_DeviceStateListener) | Callback function |

## Log off the status message listening reported by the device

**public** **void** removeDeviceStateListener([DeviceStateListener](#_DeviceStateListener) deviceStateListener)

### Description

Used to remove device status message listening

### Parameters

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| deviceStateListener | [DeviceStateListener](#_DeviceStateListener) | Callback function |

# Object Description

## StatusCode

### Description

Status of execution

### Fields

|  |  |
| --- | --- |
| **Field** | **Description** |
| SUCCESS | success |
| DISCONNECT | failed |
| TIMEOUT | timeout |
| FAIL | Bluetooth is disconnected |
| NOT\_ENABLE | Bluetooth is not open |
| PARAMETER\_ERROR | Parameter error |

## IResultCallback<T>

### Description

Callback interface

### Function

**void** onDataCallback(CallbackData<T> cd)

callback function

## CallbackData<T>

### Description

Callback object

### Fields

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| status | Int | [Status](#_StatusCode) of execution |
| type | String | Interface Type, used to distinguish between operating interface |
| result | T | The result of execution |

## LoginBean

### Description

The result of Connnect Device.

### Fields

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| hardwareVersion | String | Device version |
| deviceId | String | Device id |

## BatteryBean

### Description

The result of getting battery

### Fields

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| chargingState | int | Charging or not，  0：not  1: charging |
| quantity | int | Percentage of battery, It’s disable when charging |

## CollectStatus

### Description

Device collection data status information

### Fields

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| collectState | byte | Collection status  0: not collected  1: collecting |
| timestamp | int | Time stamp of starting collection, in seconds, current time stamp – start time stamp, which shows how long the equipment has collected |

## RealTimeData

### Description

The result of getting sleep data (Real-time)

### Fields

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| heartRate | short | Heart rate |
| breathRate | short | Breath rate |
| status | byte | [SleepStatusType](#_SleepStatusType) |
| statusValue | int | The value of status. Unit(Millisecond) |
| sleepFlag | int | Asleep or not  1: asleep  0: not |
| wakeFlag | int | Awake or not  1: awake  0: not |

## OriginalData

### Description

The result of getting signal strength

### Fields

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| heartRate | float[] | Heart rate |
| breathRate | float[] | Breath rate |

## HistoryData

### Description

The result of getting sleep report

### Fields

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| summary | Summary | History data [Summary](#_Summary) |
| detail | Detail | History data [Detail](#_Detail) |
| analy | Analysis | [Analysis](#_Analysis) |

## Summary

### Description

Summary of sleep report

### Fields

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| recordCount | int | The length of Collecting.  Unit(minute)  Eg:  300. It means you collect for 300 minutes |
| startTime | int | Start time(timestamp). Unit(second) |
| stopMode | int | How to stop collecting：  0: Call the method “Stop Collecting”  1: stop automatically if you leave the bed for an hour  2: Error(a、Collect more than 24 hours，b、Pillow shutdown c、upgrade)  3: restart |
| timeStep | int | Record interval (default 60s, ie: 1 minute a time a point) |
| timezone | int | Timezone |

## Detail

### Description

Detail of sleep report

### Fields

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| data1 | int[] | raw data1 |
| data2 | int[] | raw data2 |
| data3 | int[] | raw data3 |
| data4 | int[] | raw data4 |
| breathRate | int[] | Breath rate |
| heartRate | int[] | Heart rate |
| status | int[] | Status |
| statusValue | int[] | The value of status |

## Analysis

### Description

Analysis of sleep report

### Fields

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| avgBreathRate | int | Average breath rate(n counts per min) |
| avgHeartRate | int | Average heart rate(n counts per min) |
| fallAlseepAllTime | int | Fall asleep time(Unit:min) |
| wakeAndLeaveBedBeforeAllTime | int | Duration of awake before getting up(Unit:min) |
| leaveBedTimes | int | Counts of leaving bed |
| trunOverTimes | int | Counts of turning over |
| bodyMovementTimes | int | Counts of body movement |
| heartBeatPauseTimes | int | Counts of Heat beat Pause |
| breathPauseTimes | int | Counts of apnea |
| deepSleepPerc | int | Deep sleep percentage |
| inSleepPerc | int | Mid sleep percentage |
| lightSleepPerc | int | Light sleep percentage |
| wakeSleepPerc | int | Awake percentage |
| duration | int | Sleep duration(Unit:min) |
| wakeTimes | int | Awake times |
| lightSleepAllTime | int | Duration of Light sleep(Unit:min) |
| inSleepAllTime | int | Duration of Mid sleep(Unit:min) |
| deepSleepAllTime | int | Duration of Deep sleep)(Unit:min) |
| wake | int | Duration of Awake)(Unit:min) |
| breathPauseAllTime | int | Duration of Apnea(Unit:seconds) |
| heartBeatPauseAllTime | int | Duration of heart beat pause)(Unit:seconds) |
| outOfBedDuration | int | Duration of leaving bed(Unit:min) |
| maxHeartBeatRate | int | Maximum heart rate(n counts per min) |
| maxBreathRate | int | Maximum breath rate(n counts per min) |
| minHeartBeatRate | int | Minimum heart rate)(n counts per min) |
| minBreathRate | int | Minimum breath rate(n counts per min) |
| heartBeatRateFastAllTime | int | Duration of tachycardia(Unit:seconds) |
| heartBeatRateSlowAllTime | int | Duration of bradycardia(Unit:seconds) |
| breathRateFastAllTime | int | Duration of tachypnea(Unit:seconds |
| breathRateSlowAllTime | int | Duration of bradypnea(Unit:seconds) |
| sleepScore | int | Score  90>=score<=100 Bravo!  80>=score<90 Good!  60>=score<80, average!  score <60 Bad |
| sleepCurveArray | float[] | Example:  [0.212,1.231,2.111,0.212,1.231,2.111,....]  0: awake 0 ~ 1: light sleep 1 ~ 2: moderate sleep 2 ~ 3: deep sleep  Drawing sleep curve (Unit:min) |
| sleepCurveStatusArray | ~~short[]~~ | ~~Sleep Event Flag (Unit:min)~~ |
| breathRateStatusAry | int[] | Apnea, It used to draw the graph  Example:  [0,0,1,0,2]  0: nothing  Other: Duration of Apnea in this minute(Unit:seconds) |
| heartRateStatusAry | int[] | Heart beat pause, It used to draw the graph  Example:  [0,0,1,0,2]  0: nothing  Other: Duration of Heat beat Pause in this minute(Unit:seconds) |
| leftBedStatusAry | int[] | Leave bed, It used to draw the graph  Example:  [0,0,1,0,2]  0: nothing  Other: Duration of leaving bed in this minute(Unit:seconds) |
| turnOverStatusAry | int[] | Turn over, It used to draw the graph  Example:  [0,0,1,0,2]  0: nothing  Other: the times of turning over |
| scaArray | byte[] | Analytical Sleep Status  0:awake, 1:light Sleep, 2:middle sleep, 3:deep sleep |
| algorithmVer | String | Algorithm version |
| fallsleepTimeStamp | int | The time you fall asleep(timestamp) |
| wakeupTimeStamp | int | The time you wake up(timestamp) |
| reportFlag | int | 1. Long report(>3h) 2. Short report(>10m && <3h) |
| md\_body\_move\_decrease\_scale | short | Score Deduction:Score Deduction due to body movement |
| md\_leave\_bed\_decrease\_scale |  | Score Deduction:Score Deduction due to the times of leaving bed |
| md\_wake\_cnt\_decrease\_scale | short | Score Deduction:Score Deduction due to the wake count |
| md\_start\_time\_decrease\_scale | short | Score Deduction:Score Deduction due to sleeping time (too late) |
| md\_fall\_asleep\_time\_decrease\_scale | short | Score Deduction:Score Deduction due to long falling sleep time |
| md\_perc\_deep\_decrease\_scale | short | Score Deduction:Score Deduction due to the deep sleep |
| md\_sleep\_time\_decrease\_scale | short | Score Deduction due to sleeping time too short |
| md\_sleep\_time\_increase\_scale | short | Score Deduction due to sleeping time too long |
| md\_breath\_stop\_decrease\_scale | short | Score Deduction:Score Deduction due to breathing stop |
| md\_heart\_stop\_decrease\_scale | short | Score Deduction:Score Deduction due to Heart beat stop |
| md\_heart\_low\_decrease\_scale | short | Score Deduction:Score Deduction due to slow heart beat |
| md\_heart\_high\_decrease\_scale | short | Score Deduction:Score Deduction due to Rapid heart beat |
| md\_breath\_low\_decrease\_scale | short | Score Deduction:Score Deduction due to slow breathing |
| md\_breath\_high\_decrease\_scale | short | Score Deduction:Score Deduction due to rapid breathing |
| md\_perc\_effective\_sleep\_decrease\_scale | short | Score Deduction: score deduction for benign sleep |
| exceptioncode | short | Error code  0: report normal  50: short report less than 3 hours  51: short report of more than 3 hours, but the opposite direction of awake point and sleep point is greater than or equal to 3 hours  52: more than 3 hours, sleeping point and waking point less than or equal to 20 minutes  53: prompt report for more than 3 hours and higher off bed density  100: wrong version number  101: clipping function allocation failed  102: the data pointer is null  103: the internal allocation char array is empty  104: one of the input arrays is all 0  105: failed to allocate the collation status array  106: sleep point calculation failed  107: sober point calculation failed  110: Universal allocation failed  111: heart rate respiratory structure dynamic allocation failure  112: the average value of heart rate and respiratory rate is 0, and the data is invalid |

## SleepStatusType

### Description

Status value of monitoring

### Fields

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Value** | **Description** |
| SLEEP\_OK | byte | 0x00 | normal |
| SLEEP\_INIT | byte | 0x01 | init |
| SLEEP\_B\_STOP | byte | 0x02 | apnea |
| SLEEP\_H\_STOP | byte | 0x03 | Heartbeat pause |
| SLEEP\_BODYMOVE | byte | 0x04 | Body movement |
| SLEEP\_LEAVE | byte | 0x05 | Leaving bed |
| SLEEP\_TURN\_OVER | byte | 0x06 | Turning over |
| SLEEP\_BODYMOVE\_TEMP | byte | 0x07 | Amplitude of body motion |
| SLEEP\_INVALID | byte | -1 | invalid |

## DeviceStateListener

### Description

Device status callback interface

### Function

**public** **void** onDeviceStateChanged([DeviceState](#_DeviceState) deviceState)

This method is called back when the state of the device side changes

## DeviceState

### Description

Device status information class

### Fields

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| type | byte | Device status message type:  0x01: low power  0x02: Collection start (there will be no such notification when normal start)  0x03: end of collection (there will be no such notification after normal end)  0x04: in bed  0x05: out of bed  0x06: sleep  0x07: wake up  0x08: awake  0xf0: Add Device confirmation |
| value | byte | Status value. When the status is 0x01, it indicates the percentage of power (0-100)  In other states, the value is 0, which is meaningless |

## RealtimeDataListener

### Interface Introduction

Real-time data monitoring

### Method description

**void** onReceive([RealTimeData](#_RealTimeData) realTimeData);

realTimeData：Real-time data reported by the device, including heart rate and breath rate, etc. refer to：[RealTimeData](#_RealTimeData)