2301API说明文档

版本历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| 修订号 | 作者 | 日期 | 说明 |
| V1.0 | 沈钰权 | 2023.12.11 | 建立 |
|  |  |  |  |
|  |  |  |  |

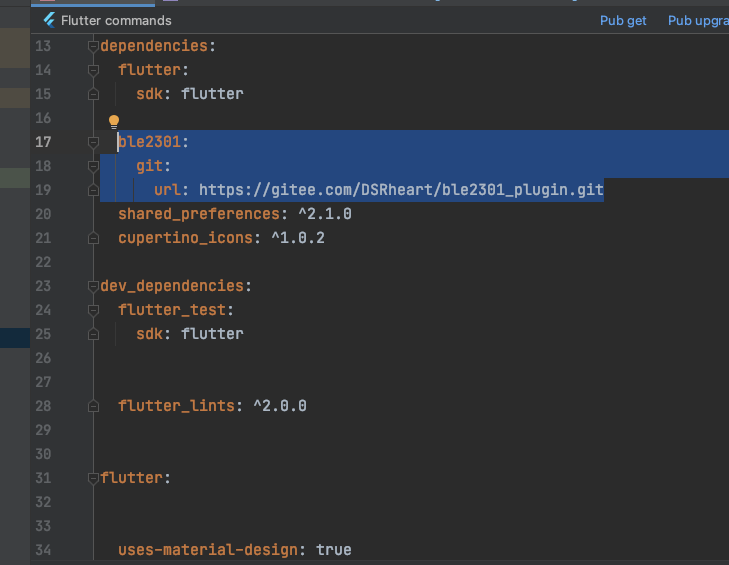
1. 接入流程（Access Process）
2. 导入库（add to library）

ble2301:

git:

url: <https://gitee.com/DSRheart/ble2301_plugin.git>

图例（illustrative）：



1. AndroidManifest添加权限（AndroidManifest Add Permissions）（android）

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

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

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

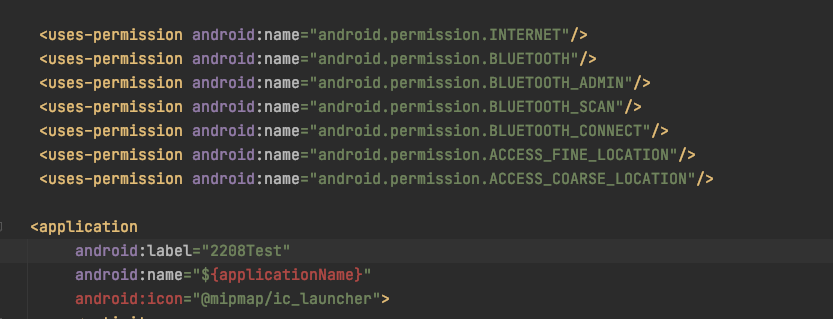
<uses-permission android:name="android.permission.BLUETOOTH\_SCAN"/>

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

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

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

图例（illustrative）:



1. Info.plist添加权限（Info.plist Add Permissions）（ios）

<key>NSBluetoothAlwaysUsageDescription</key>

<string>蓝牙权限提示</string>

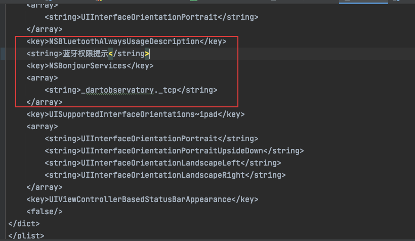
<key>NSBonjourServices</key>

<array>

<string>\_dartobservatory.\_tcp</string>

</array>

图例：



1. 接口说明
   1. 设置时间（Set Device Time）

|  |  |  |  |
| --- | --- | --- | --- |
| Method | BleSDK.SetDeviceTime(DateTime time) | | |
| Bluetooth instruction head | 0x01 | | |
| Description |  | | |
| Input parameter | | | |
| Parameter Name | | Type | Description |
| time | | DateTime | current time |
| Output parameters | | | |
| dataType | | Int | Data type return id |
| dataEnd | | Bool | End of data flag |
| dicData | | Map | useless parameter |
| Input Case | | | |
| DateTime dateTime = DateTime.now();  //Bluetooth data writing method  controller.writeData(BleSDK.SetDeviceTime(dateTime)); | | | |
| Export Cases | | | |
| {dataType: 1, dataEnd: true, dicData: {KPhoneDataLength: 244}} | | | |

* 1. 获取时间（Get device time）

|  |  |  |  |
| --- | --- | --- | --- |
| Method | BleSDK.GetDeviceTime() | | |
| Bluetooth instruction head | 0x41 | | |
| Description |  | | |
| Input parameter | | | |
| Parameter Name | | Type | Description |
|  | |  |  |
| Output parameters | | | |
| dataType | | Int | Data type return id |
| dataEnd | | Bool | End of data flag |
| dicData | | Map |  |
| dicData-strDeviceTime | | String | Current time of the device |
| dicData-GPSTime | | String | useless parameter |
| Input Case | | | |
| //Bluetooth data writing method  controller.writeData(BleSDK.GetDeviceTime()); | | | |
| Export Cases | | | |
| {dataEnd : true,  dataType : 0, dicData : {  strDeviceTime : 2023-04-26 14:41:49,  GPSTime : 00.00.00}} | | | |

* 1. 设置用户个人信息（Set user profile）

|  |  |  |  |
| --- | --- | --- | --- |
| Method | BleSDK.SetPersonalInfo(MyPersonalInfo info) | | |
| Bluetooth instruction head | 0x02 | | |
| Description |  | | |
| Input parameter | | | |
| Parameter Name | | Type | Description |
| info | | MyPersonalInfo | User Information Entity Class |
| info-sex | | Int | 1: male  0 : female |
| info-age | | Int | Age (1-220) |
| info-height | | Int | Height (cm) |
| info-weight | | Int | Weight (kg) |
| Output parameters | | | |
| dataType | | Int | Data type return id |
| dataEnd | | Bool | End of data flag |
| dicData | | Map |  |
| Input Case | | | |
| var info = MyPersonalInfo(sex: 0,age: 1,height: height,weight: weight);  //Bluetooth data writing method  controller.writeData(BleSDK.SetPersonalInfo(info)); | | | |
| Export Cases | | | |
| {dataType : 3, dicData : {}, dataEnd : true} | | | |

* 1. 获取用户个人信息（Get user's personal information）

|  |  |  |  |
| --- | --- | --- | --- |
| Method | BleSDK.GetPersonalInfo() | | |
| Bluetooth instruction head | 0x42 | | |
| Description |  | | |
| Input parameter | | | |
| Parameter Name | | Type | Description |
|  | |  |  |
| Output parameters | | | |
| dataType | | Int | Data type return id |
| dataEnd | | Bool | End of data flag |
| dicData | | Map |  |
| dicData-MyStride | | Int | useless parameter |
| dicData-MyAge | | Int | Age |
| dicData-MyWeight | | Int | Weight |
| dicData-MyGender | | Int | Gender :  1: male  0 : female |
| dicData-MyHeight | | Int | Height |
| dicData-deviceId | | String | useless parameter |
| Input Case | | | |
| //Bluetooth data writing method  controller.writeData(BleSDK.GetPersonalInfo()); | | | |
| Export Cases | | | |
| {dataEnd : true, dataType : 2, dicData : {MyStride : 66, MyAge : 38, MyWeight : 66, MyGender : 0, MyHeight : 166, deviceId : 888888}} | | | |

* 1. 读取设备电量（**Read device power**）

|  |  |  |  |
| --- | --- | --- | --- |
| Method | BleSDK.GetDeviceBatteryLevel() | | |
| Bluetooth instruction head | 0x13 | | |
| Description |  | | |
| Input parameter | | | |
| Parameter Name | | Type | Description |
|  | |  |  |
| Output parameters | | | |
| dataType | | Int | Data type return id |
| dataEnd | | Bool | End of data flag |
| dicData | | Map |  |
| dicData-batteryLevel | | Int | Electricity of equipment(1-100) |
| Input Case | | | |
| //Bluetooth data writing method  controller.writeData(BleSDK.GetDeviceBatteryLevel()); | | | |
| Export Cases | | | |
| {dataType : 9, dicData : {batteryLevel : 10  }, dataEnd : true} | | | |

* 1. 读取MAC地址（**Read MAC address**）

|  |  |  |  |
| --- | --- | --- | --- |
| Method | BleSDK.GetDeviceMacAddress() | | |
| Bluetooth instruction head | 0x22 | | |
| Description |  | | |
| Input parameter | | | |
| Parameter Name | | Type | Description |
|  | |  |  |
| Output parameters | | | |
| dataType | | Int | Data type return id |
| dataEnd | | Bool | End of data flag |
| dicData | | Map |  |
| dicData-macAddress | | String | MAC address of the device |
| Input Case | | | |
| //Bluetooth data writing method  controller.writeData(BleSDK.GetDeviceMacAddress()); | | | |
| Export Cases | | | |
| {dataEnd : true, dataType : 10, dicData : {macAddress : FE:53:8C:0D:73:FB}} | | | |

* 1. 读取软件版本号（**Read software version number**）

|  |  |  |  |
| --- | --- | --- | --- |
| Method | BleSDK.GetDeviceVersion() | | |
| Bluetooth instruction head | 0x27 | | |
| Description |  | | |
| Input parameter | | | |
| Parameter Name | | Type | Description |
|  | |  |  |
| Output parameters | | | |
| dataType | | Int | Data type return id |
| dataEnd | | Bool | End of data flag |
| dicData | | Map |  |
| dicData-deviceVersion | | String | software version number |
| Input Case | | | |
| //Bluetooth data writing method  controller.writeData(BleSDK.GetDeviceVersion()); | | | |
| Export Cases | | | |
| {dataType : 11, dicData : {deviceVersion : 0.3.2.1}, dataEnd : true} | | | |

* 1. 恢复出厂设置（**Restore factory settings**）

|  |  |  |  |
| --- | --- | --- | --- |
| Method | BleSDK.Reset() | | |
| Bluetooth instruction head | 0x12 | | |
| Description |  | | |
| Input parameter | | | |
| Parameter Name | | Type | Description |
|  | |  |  |
| Output parameters | | | |
| dataType | | Int | Data type return id |
| dataEnd | | Bool | End of data flag |
| dicData | | Map |  |
| Input Case | | | |
| //Bluetooth data writing method  controller.writeData(BleSDK.Reset()); | | | |
| Export Cases | | | |
| {dataType : 12, dataEnd : true} | | | |

* 1. MCU软复位指令（**MCU soft reset command**）

|  |  |  |  |
| --- | --- | --- | --- |
| Method | BleSDK.MCUReset() | | |
| Bluetooth instruction head | 0x2E | | |
| Description |  | | |
| Input parameter | | | |
| Parameter Name | | Type | Description |
|  | |  |  |
| Output parameters | | | |
| dataType | | Int | Data type return id |
| dataEnd | | Bool | End of data flag |
| dicData | | Map |  |
| Input Case | | | |
| //Bluetooth data writing method  controller.writeData(BleSDK.MCUReset()); | | | |
| Export Cases | | | |
| {dataType : 13, dataEnd : true} | | | |

* 1. 读取自动检测心率时段（**Read auto detect heart rate period**）

|  |  |  |  |
| --- | --- | --- | --- |
| Method | BleSDK.GetAutomaticHRMonitoring(int type) | | |
| Bluetooth instruction head | 0x2B | | |
| Description |  | | |
| Input parameter | | | |
| Parameter Name | | Type | Description |
| type | | Int | 1 : Heart rate  2 : Blood oxygen  3 : Temperature  4 : HRV |
| Output parameters | | | |
| dataType | | Int | Data type return id |
| dataEnd | | Bool | End of data flag |
| dicData | | Map |  |
| dicData-heartStartHour | | String | The hour parameter for measuring the start time of a time period |
| dicData-heartStartMinter | | String | The minute parameter for measuring the start time of a time period |
| dicData-heartEndHour | | String | The hour parameter for measuring the end time of a time period |
| dicData-heartEndMinter | | String | The minute parameter for measuring the end time of a time period |
| dicData-workModel | | Int | 0 close 2 open |
| dicData-weekValue | | String | Bit1 : 0 means Monday is not enabled, Bit1 : 1 means Monday is enabled.  Bit2 : 0 means Tuesday is not enabled, bit2 : 1 means Tuesday is enabled.  Bit3 : 0 means Wednesday is not enabled, bit3 : 1 means Wednesday is enabled.  Bit4 : 0 means Thursday is not enabled, bit4 : 1 means Thursday is enabled.  Bit5 : 0 means Friday is not enabled, bit5 : 1 means Friday is enabled.  Bit6 : 0 means Saturday is not enabled, bit6 : 1 means Saturday is enabled.  Bit7 : 0 means Sunday is not enabled, bit7 : 1 means Sunday is enabled. |
| dicData-workTime | | String | Detection interval time(min) |
| Input Case | | | |
| //Bluetooth data writing method  controller.writeData(BleSDK.GetAutomaticHRMonitoring(2)); | | | |
| Export Cases | | | |
| {dataType : 17, dicData : {  heartStartHour : 00,  heartStartMinter : 00,  workModel : 2,  heartEndHour : 23,  heartEndMinter : 59,  workTime : 5,  weekValue : 1-1-1-1-1-1-1},  dataEnd : true} | | | |

* 1. 设置自动检测心率时段（**Set automatic heart rate detection period**）

|  |  |  |  |
| --- | --- | --- | --- |
| Method | BleSDK.MCUReset() | | |
| Bluetooth instruction head | 0x2A | | |
| Description |  | | |
| Input parameter | | | |
| Parameter Name | | Type | Description |
| info | | MyAutomaticHRMonitoring | Set entity classes for automatic heart rate detection during time periods |
| info-mode | | Int | 0 close  2 open |
| info-startHour | | Int | The hour parameter for measuring the start time of a time period(Throughout the day, usually fixed without modification) |
| info-startMinute | | Int | The minute parameter for measuring the start time of a time period(Throughout the day, usually fixed without modification) |
| info-endHour | | Int | The hour parameter for measuring the end time of a time period(Throughout the day, usually fixed without modification) |
| info-endMinute | | Int | The minute parameter for measuring the end time of a time period(Throughout the day, usually fixed without modification) |
| info-time | | Int | Time interval between each measurement |
| info-type | | Int | 1 : Heart rate  2 : Blood oxygen  3 : Temperature  4 : HRV |
| info-week | | Int | 7 days a week, which days need to be measured (usually all need to be measured) |
| Output parameters | | | |
| dataType | | Int | Data type return id |
| dataEnd | | Bool | End of data flag |
| dicData | | Map |  |
| Input Case | | | |
| import 'dart:math';  var weeks = [1,1,1,1,1,1,1];  int week = 0;  for (int i = 0; i < 7; i++) {  if (weeks[i] == 1) {  week + = pow(2, i).toInt();  }  }  var info = MyAutomaticHRMonitoring(  open: 2, startHour:0,  startMinute: 0,  endHour: 23,  endMinute: 59,  week: week, time: 30,  type:type  );  //Bluetooth data writing method  controller.writeData(BleSDK.SetAutomaticHRMonitoring(info)); | | | |
| Export Cases | | | |
| {dataType : 17, dicData : {}, dataEnd : true} | | | |

* 1. 开始实时计步（**Start real time step**）

|  |  |  |  |
| --- | --- | --- | --- |
| Method | BleSDK.RealTimeStep(bool enable,bool tempEnable ) | | |
| Bluetooth instruction head | 0x09 | | |
| Description |  | | |
| Input parameter | | | |
| Parameter Name | | Type | Description |
| enable | | Bool | true : Start real-time step counting  false : Turn off real-time step counting |
| tempEnable | | Bool | true : Enable real-time temperature  false : Turn off real-time temperature |
| Output parameters | | | |
| dataType | | Int | Data type return id |
| dataEnd | | Bool | End of data flag |
| dicData | | Map |  |
| dicData-heartRate | | String | Heart rate value(bpm) |
| dicData-distance | | String | Walking distance(km) |
| dicData-ExerciseTime | | String | Fast movement time(min) |
| dicData-calories | | String | caloric value(Kcal) |
| dicData-TempData | | String | Temperature value(℃)(Needs /10) |
| dicData-step | | String | Total Steps |
| dicData-exerciseMinutes | | String | Exercise time(min) |
| Input Case | | | |
| //Bluetooth data writing method  controller.writeData(BleSDK.RealTimeStep(true,true)); | | | |
| Export Cases | | | |
| {dataType : 19, dicData : {  heartRate : 89,  distance : 0.12,  step : 193,  ExerciseTime : 0,  calories : 11.3,  exerciseMinutes : 1,  TempData:356  }, dataEnd : true} | | | |

* 1. 根据时间获得计步总数据(Obtain total step count data based on time）

|  |  |  |  |
| --- | --- | --- | --- |
| Method | BleSDK.GetTotalActivityDataWithModeForTime(int mode,String time) | | |
| Bluetooth instruction head | 0x51 | | |
| Description |  | | |
| Input parameter | | | |
| Parameter Name | | Type | Description |
| mode | | Int | 0 : Read the latest data (up to 50 groups of data)  2 : Continue reading from the previous position to the next section of data (up to 50 groups of data)  99 : Delete historical total data |
| time | | String | Time:  The first method is to pass in the '' string and query all data to the latest one  The second method is to pass in the specified date 'yyyy MM dd HH: mm' and query the data from the specified date to the latest data |
| Output parameters | | | |
| dataType | | Int | Data type return id |
| dataEnd | | Bool | End of data flag |
| dicData | | Map |  |
| dicData-[]-date | | String | Date of data (year, month, day) |
| dicData-[]-goal | | String | Target(useless parameter) |
| dicData-[]-distance | | String | Walking distance(km) |
| dicData-[]-step | | String | Total Steps |
| dicData-[]-ExerciseTime | | Int | Fast movement time(min) |
| dicData-[]-calories | | String | caloric value(Kcal) |
| dicData-[]-exerciseMinutes | | String | Exercise time(min) |
| Input Case | | | |
| If the value of the command mode sent is 0 or 2, and the number of Bluetooth data returns exceeds 50 times, it is necessary to send the command with a mode value of 2 again, repeating the previous judgment steps. If the value of dataEnd is true, it means that the data has been received and completed  //Bluetooth data writing method  The first type:  controller.writeData(BleSDK.GetTotalActivityDataWithMode(0,””));  Second type:  controller.writeData(BleSDK.GetTotalActivityDataWithMode(0,”2023-12-10 00:00”)); | | | |
| Export Cases | | | |
| {dataEnd : false, dataType : 24, dicData : [{date : 2023.04.18, goal : 0, distance : 0.00, step : 0, ExerciseTime : 0, calories : 0.00, exerciseMinutes : 0}, {date : 2023.04.17, goal : 26, distance : 1.61, step : 2647, ExerciseTime : 19, calories : 79.64, exerciseMinutes : 1339}, {date : 2023.04.16, goal : 0, distance : 0.00, step : 0, ExerciseTime : 0, calories : 0.00, exerciseMinutes : 0}, {date : 2023.04.15, goal : 46, distance : 3.70, step : 4689, ExerciseTime : 23, calories : 244.83, exerciseMinutes : 1712}]}  {dataEnd : true, dataType : 24, dicData : []} | | | |

* 1. 获得步数详细数据（**Get step details**）

|  |  |  |  |
| --- | --- | --- | --- |
| Method | BleSDK.GetDetailActivityDataWithModeForTime(int mode,String time) | | |
| Bluetooth instruction head | 0x52 | | |
| Description |  | | |
| Input parameter | | | |
| Parameter Name | | Type | Description |
| mode | | Int | 0 : Read the latest data (up to 50 groups of data)  2 : Continue reading from the previous position to the next section of data (up to 50 groups of data)  99 : Delete historical total data |
| time | | String | Time:  The first method is to pass in the '' string and query all data to the latest one  The second method is to pass in the specified date 'yyyy MM dd HH: mm' and query the data from the specified date to the latest data |
| Output parameters | | | |
| dataType | | Int | Data type return id |
| dataEnd | | Bool | End of data flag |
| dicData | | Map |  |
| dicData-[]-date | | String | Date of data (year, month, day) |
| dicData-[]-detailMinterStep | | String | Total Steps |
| dicData-[]-distance | | String | Walking distance(km) |
| dicData-[]-calories | | String | caloric value(Kcal) |
| dicData-[]-arraySteps | | Int[] | The number of steps taken per minute within these 10 minutes |
| Input Case | | | |
| If the value of the command mode sent is 0 or 2, and the number of Bluetooth data returns exceeds 50 times, it is necessary to send the command with a mode value of 2 again, repeating the previous judgment steps. If the value of dataEnd is true, it means that the data has been received and completed  //Bluetooth data writing method  The first type:  controller.writeData(BleSDK.GetDetailActivityDataWithModeForTime(0,””));  Second type:  controller.writeData(BleSDK.GetDetailActivityDataWithModeForTime(0,”2023-12-10 00:00”)); | | | |
| Export Cases | | | |
| {dataEnd : true, dataType : 25, dicData : [{date : 2023.04.15 17:00:53, detailMinterStep : 102, distance : 0.05, calories : 2.77, arraySteps : 16 0 18 10 36 0 0 22 0 0}, {date : 2023.04.15 15:48:24, detailMinterStep : 1175, distance : 1.08, calories : 75.99, arraySteps : 196 192 194 193 195 183 22 0 0 0}, {date : 2023.04.15 15:38:19, detailMinterStep : 1931, distance : 1.77, calories : 124.93, arraySteps : 183 196 195 196 192 196 194 194 192 193}]} | | | |

* 1. 获得睡眠详细数据（**Get sleep details**）

|  |  |  |  |
| --- | --- | --- | --- |
| Method | BleSDK.GetDetailSleepDataWithModeForTime(int mode,String time) | | |
| Bluetooth instruction head | 0x53 | | |
| Description |  | | |
| Input parameter | | | |
| Parameter Name | | Type | Description |
| mode | | Int | 0 : Read the latest data (up to 50 groups of data)  2 : Continue reading from the previous position to the next section of data (up to 50 groups of data)  99 : Delete historical total data |
| time | | String | Time:  The first method is to pass in the '' string and query all data to the latest one  The second method is to pass in the specified date 'yyyy MM dd HH: mm' and query the data from the specified date to the latest data |
| Output parameters | | | |
| dataType | | Int | Data type return id |
| dataEnd | | Bool | End of data flag |
| dicData | | Map |  |
| dicData-[]-date | | String | The start time of this sleep period |
| dicData-[]-arraySleepQuality | | String | Each number represents the quality of one minute of sleep |
| dicData-[]-sleepUnitLength | | String | Fixed to 1, each value in the index group represents one minute of data |
| Input Case | | | |
| If the value of the command mode sent is 0 or 2, and the number of Bluetooth data returns exceeds 50 times, it is necessary to send the command with a mode value of 2 again, repeating the previous judgment steps. If the value of dataEnd is true, it means that the data has been received and completed  //Bluetooth data writing method  The first type:  controller.writeData(BleSDK.GetDetailSleepDataWithModeForTime(0,””));  Second type:  controller.writeData(BleSDK.GetDetailSleepDataWithModeForTime(0,”2023-12-10 00:00”)); | | | |
| Export Cases | | | |
| {dataEnd : true, dataType : 26, dicData : [{date: 2023-04-15 15:59:59, arraySleepQuality : 1 2 2 2 2 2 2 2 2 3 1 1 1 2,sleepUnitLength:1 }]}  {DataType : 26, Data : [], DataEnd : true} | | | |

* 1. 获得心率数据（**Get heart rate data**）

|  |  |  |  |
| --- | --- | --- | --- |
| Method | BleSDK.GetDynamicHRWithMode(int Number) | | |
| Bluetooth instruction head | 0x54 | | |
| Description |  | | |
| Input parameter | | | |
| Parameter Name | | Type | Description |
| Number | | Int | 0 : Read the latest data (up to 50 groups of data)  2 : Continue reading from the previous position to the next section of data (up to 50 groups of data)  99 : Delete historical total data |
| Output parameters | | | |
| dataType | | Int | Data type return id |
| dataEnd | | Bool | End of data flag |
| dicData | | Map |  |
| dicData-[]-date | | String | Date of detection data |
| dicData-[]-arrayDynamicHR | | String | Save a heart rate value every minute |
| Input Case | | | |
| If the value of the command mode sent is 0 or 2, and the number of Bluetooth data returns exceeds 50 times, it is necessary to send the command with a mode value of 2 again, repeating the previous judgment steps. If the value of dataEnd is true, it means that the data has been received and completed  //Bluetooth data writing method  controller.writeData(BleSDK.GetDynamicHRWithMode(0)); | | | |
| Export Cases | | | |
| {dataEnd : false, dataType : 27, dicData : [{date : 2023.04.15 19:19:59, arrayDynamicHR : 84 89 94 86 86 87 89 87 86 87 90 83 81 81 80}, {date : 2023.04.15 18:27:59, arrayDynamicHR : 75 77 0 0 0 0 0 0 0 0 0 0 0 0 0}, {date : 2023.04.15 17:15:59, arrayDynamicHR : 78 75 79 0 0 0 0 0 0 0 0 0 0 0 0}, {date : 2023.04.15 16:52:59, arrayDynamicHR : 79 80 85 85 81 90 83 77 79 80 74 72 75 76 81}, {date : 2023.04.15 16:09:59, arrayDynamicHR : 70 72 0 0 0 0 0 0 0 0 0 0 0 0 0}]} | | | |

* 1. 获得单次心率数据（间隔测试心率）（**Obtain single heart rate data (interval test heart rate)**

|  |  |  |  |
| --- | --- | --- | --- |
| Method | BleSDK.GetStaticHRWithMode(int mode) | | |
| Bluetooth instruction head | 0x55 | | |
| Description |  | | |
| Input parameter | | | |
| Parameter Name | | Type | Description |
| mode | | Int | 0 : Read the latest data (up to 50 groups of data)  2 : Continue reading from the previous position to the next section of data (up to 50 groups of data)  99 : Delete historical total data |
| Output parameters | | | |
| dataType | | Int | Data type return id |
| dataEnd | | Bool | End of data flag |
| dicData | | Map |  |
| dicData-[]-date | | String | Date of detection data |
| dicData-[]-onceHeartValue | | String | Heart rate value |
| Input Case | | | |
| If the value of the command mode sent is 0 or 2, and the number of Bluetooth data returns exceeds 50 times, it is necessary to send the command with a mode value of 2 again, repeating the previous judgment steps. If the value of dataEnd is true, it means that the data has been received and completed  //Bluetooth data writing method  controller.writeData(BleSDK.GetStaticHRWithMode(0)); | | | |
| Export Cases | | | |
| {dataEnd : true, dataType : 28, dicData : [{date : 2023.04.15 17:20:30, onceHeartValue : 108}, {date : 2023.04.15 17:00:30, onceHeartValue : 78}, {date : 2023.04.15 16:55:30, onceHeartValue : 71}, {date : 2023.04.15 16:50:30, onceHeartValue : 80}]} | | | |

* 1. 获得HRV数据（**Get HRV test data**）

|  |  |  |  |
| --- | --- | --- | --- |
| Method | BleSDK.GetHRVDataWithMode(int mode) | | |
| Bluetooth instruction head | 0x56 | | |
| Description |  | | |
| Input parameter | | | |
| Parameter Name | | Type | Description |
| mode | | Int | 0 : Read the latest data (up to 50 groups of data)  2 : Continue reading from the previous position to the next section of data (up to 50 groups of data)  99 : Delete historical total data |
| Output parameters | | | |
| dataType | | Int | Data type return id |
| dataEnd | | Bool | End of data flag |
| dicData | | Map |  |
| dicData-[]-date | | String | Date of detection data |
| dicData-[]-hrv | | String | HRV value |
| dicData-[]-stress | | String | fatigue |
| dicData-[]-highBP | | String | High blood pressure |
| dicData-[]-lowBP | | String | Low blood pressure |
| dicData-[]-heartRate | | String | Heart rate value |
| dicData-[]-vascularAging | | String | Vascular aging degree value |
| Input Case | | | |
| If the value of the command mode sent is 0 or 2, and the number of Bluetooth data returns exceeds 50 times, it is necessary to send the command with a mode value of 2 again, repeating the previous judgment steps. If the value of dataEnd is true, it means that the data has been received and completed  //Bluetooth data writing method  controller.writeData(BleSDK.GetHRVDataWithMode(0)); | | | |
| Export Cases | | | |
| {dataType: 42, dataEnd: true, dicData: [{date: 2023.12.12 15:28:28, hrv: 65, vascularAging: 0, stress: 33, highBP: 118, lowBP: 63, heartRate: 98}, {date: 2023.12.12 14:58:29, hrv: 129, vascularAging: 0, stress: 41, highBP: 111, lowBP: 61, heartRate: 101}, {date: 2023.12.12 14:28:41, hrv: 74, vascularAging: 0, stress: 34, highBP: 119, lowBP: 64, heartRate: 99}, {date: 2023.12.11 19:58:38, hrv: 101, vascularAging: 0, stress: 34, highBP: 114, lowBP: 64, heartRate: 94}, {date: 2023.12.11 19:28:25, hrv: 24, vascularAging: 0, stress: 55, highBP: 115, lowBP: 60, heartRate: 95}, {date: 2023.12.11 16:58:33, hrv: 144, vascularAging: 0, stress: 33, highBP: 118, lowBP: 63, heartRate: 98}, {date: 2023.12.11 16:28:33, hrv: 137, vascularAging: 0, stress: 34, highBP: 114, lowBP: 64, heartRate: 94}]} | | | |

* 1. 获得血氧数据（Get blood oxygen data）

|  |  |  |  |
| --- | --- | --- | --- |
| Method | BleSDK.GetAutoBloodOxygen(int status) | | |
| Bluetooth instruction head | 0x66 | | |
| Description |  | | |
| Input parameter | | | |
| Parameter Name | | Type | Description |
| status | | Int | 0 : Read the latest data (up to 50 groups of data)  2 : Continue reading from the previous position to the next section of data (up to 50 groups of data)  99 : Delete historical total data |
| Output parameters | | | |
| dataType | | Int | Data type return id |
| dataEnd | | Bool | End of data flag |
| dicData | | Map |  |
| dicData-[]-date | | String | Date of detection data |
| dicData-[]-Blood\_oxygen | | String | Blood oxygen value |
| Input Case | | | |
| If the value of the command mode sent is 0 or 2, and the number of Bluetooth data returns exceeds 50 times, it is necessary to send the command with a mode value of 2 again, repeating the previous judgment steps. If the value of dataEnd is true, it means that the data has been received and completed  //Bluetooth data writing method  controller.writeData(BleSDK.GetAutoBloodOxygen(0)); | | | |
| Export Cases | | | |
| {dataType: 70, dataEnd: true, dicData: [{date: 2023.04.04 11:14:05, Blood\_oxygen: 99}, {date: 2023.04.04 11:13:44, Blood\_oxygen: 99}, {date: 2023.04.04 11:13:21, Blood\_oxygen: 99}, {date: 2023.04.04 11:13:00, Blood\_oxygen: 100}, {date: 2023.04.04 11:12:38, Blood\_oxygen: 98}]} | | | |

* 1. 自动测试温度数据（Automatic test temperature data）

|  |  |  |  |
| --- | --- | --- | --- |
| Method | BleSDK.GetTemperature\_historyDataWithMode(int mode) | | |
| Bluetooth instruction head | 0x62 | | |
| Description |  | | |
| Input parameter | | | |
| Parameter Name | | Type | Description |
| mode | | Int | 0 : Read the latest data (up to 50 groups of data)  2 : Continue reading from the previous position to the next section of data (up to 50 groups of data)  99 : Delete historical total data |
| Output parameters | | | |
| dataType | | Int | Data type return id |
| dataEnd | | Bool | End of data flag |
| dicData | | Map |  |
| dicData-[]-date | | String | Date of detection data |
| dicData-[]-temperature | | String | Temperature value |
| Input Case | | | |
| If the value of the command mode sent is 0 or 2, and the number of Bluetooth data returns exceeds 50 times, it is necessary to send the command with a mode value of 2 again, repeating the previous judgment steps. If the value of dataEnd is true, it means that the data has been received and completed  //Bluetooth data writing method  controller.writeData(BleSDK.GetTemperature\_historyDataWithMode(0)); | | | |
| Export Cases | | | |
| {dataType: 59, dataEnd: true, dicData: [{date: 2023.11.22 02:29:59, temperature: 26.0}, {date: 2023.11.22 01:59:59, temperature: 26.1}, {date: 2023.11.22 01:29:59, temperature: 26.1}, {date: 2023.11.22 00:59:59, temperature: 26.2}, {date: 2023.11.22 00:29:59, temperature: 26.3}, {date: 2023.11.21 23:59:59, temperature: 26.4}, {date: 2023.11.21 23:29:59, temperature: 26.6}, {date: 2023.11.21 22:59:59, temperature: 26.8}, {date: 2023.11.21 22:29:59, temperature: 27.0}, {date: 2023.11.21 21:59:59, temperature: 27.0}, {date: 2023.11.21 21:29:59, temperature: 27.1}, {date: 2023.11.21 20:59:59, temperature: 27.3}, {date: 2023.11.21 20:29:59, temperature: 27.3}, {date: 2023.11.21 19:59:59, temperature: 27.3}, {date: 2023.11.21 19:29:59, temperature: 27.4}, {date: 2023.11.21 18:59:59, temperature: 27.6}, {date: 2023.11.21 18:29:59, temperature: 27.9}, {date: 2023.11.21 17:59:59, temperature: 29.9}, {date: 2023.11.21 17:41:59, temperature: 48.0}, {date: 2023.11.21 17:29:59, temperature: 33.4}]} | | | |

* 1. 健康测量控制（Health measurement control）

|  |  |  |  |
| --- | --- | --- | --- |
| Method | BleSDK.HealthMeasurementWithDataType(int mode,bool enable) | | |
| Bluetooth instruction head | 0x28 | | |
| Description |  | | |
| Input parameter | | | |
| Parameter Name | | Type | Description |
| mode | | Int | 1 : hrv  2 : HeartRate  3 : Blood oxygen |
| enable | | Bool | true : open  false : close |
| Output parameters | | | |
| dataType | | Int | Data type return id |
| dataEnd | | Bool | End of data flag |
| dicData | | Map |  |
| Input Case | | | |
| If the value of the command mode sent is 0 or 2, and the number of Bluetooth data returns exceeds 50 times, it is necessary to send the command with a mode value of 2 again, repeating the previous judgment steps. If the value of dataEnd is true, it means that the data has been received and completed  //Bluetooth data writing method  controller.writeData(BleSDK.HealthMeasurementWithDataType(1,false)); | | | |
| Export Cases | | | |
| 1,hrv,  OPEN : ture {dataEnd : true, dataType : 63, dicData : {}}  OPEN : false {dataEnd : true, dataType : 66, dicData : {}}  2 HeartRate,  OPEN : ture {dataEnd : true, dataType : 64, dicData : {}}  OPEN : false {dataEnd : true, dataType : 67, dicData : {}}  3 Blood oxygen  OPEN : ture {dataEnd : true, dataType : 65, dicData : {}}  OPEN : false {dataEnd : true, dataType : 68, dicData : {}} | | | |

* 1. 运动模式（Sports mode）

Use Flow：

1. Activate sports mode
2. Start Timer (Send Current Seconds to Ring)

3. Pause (timer stops)/Continue (timer restarts even if)/Stop motion mode (timer ends)

|  |  |  |  |
| --- | --- | --- | --- |
| Method | BleSDK.EnterActivityMode(int activityMode, int WorkMode) | | |
| Bluetooth instruction head | 0x19 | | |
| Description |  | | |
| Input parameter | | | |
| Parameter Name | | Type | Description |
| activityMode | | Int | 0:Run  1: Cycling  2:Badminton  3:Soccer  4:Tennis  5:Yoga  6:Breathe  7:Dance  8:Basketball  9:Walk  10:Workout  11:Cricket  12:Hiking  13:Aerobics  14:Ping-Pong  15:Rope Jump  16:Sit-ups  17:Volleyball |
| WorkMode | | Int | 1:start  2:pause  3:contuine  4:finish |
| Output parameters | | | |
| dataType | | Int | Data type return id |
| dataEnd | | Bool | End of data flag |
| dicData | | Map |  |
| dicData-[]-date | | String | Date of detection data |
| dicData-[]-temperature | | String | Temperature value |
| Input Case | | | |
| //Bluetooth data writing method  //start sport  controller.writeData(BleSDK.EnterActivityMode(1,1));  Int second = 1;  While(true){//Simulated timer,+1 per second  second++;  controller.writeData(BleSDK.sendHeartPackage(0, second, 2));  }  //stop sport  controller.writeData(BleSDK.EnterActivityMode(1,4));  //Timer stopped | | | |
| Export Cases | | | |
|  | | | |