



鼎桥 Watch4 健康服务 SDK 接口说明

文档版本: V2.16
发布日期: 2024-4-15

鼎桥通信技术有限公司


网址: <http://www.td-tech.com>

客户服务电话: 400 0608 000

版权所有©鼎桥通信技术有限公司 2015。 保留一切权利。

非经本公司书面许可, 任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部, 并不得以任何形式传播。

商标声明

 **TD Tech** 和其他商标均为鼎桥通信技术有限公司的商标。

本文档提及的其他所有商标或注册商标, 由各自的所有人拥有。

注意

由于产品版本升级或其他原因, 本文档内容会不定期进行更新。除非另有约定, 本文档



仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目 录

目 录	1
前 言	5
一 变更说明	6
二 概述	7
2.1 接口管理类介绍	7
2.2 使用流程及注意事项	7
2.2.1 添加依赖	7
2.2.2 服务的初始化及释放	8
三 接口详细说明	9
3.1 历史数据查询	9
3.1.1 查询历史心率	9
3.1.2 查询历史血氧	10
3.1.3 查询历史压力数据	12
3.1.4 获取锻炼记录	13
3.1.5 获取睡眠记录	14
3.1.6 获取体温记录	16
3.1.7 获取步数	17
3.1.8 获取距离	19
3.1.9 获取卡路里	20
3.1.10 获取睡眠状态	21
3.1.11 获取当日活动数据	22
3.1.12 获取运动轨迹数据	23
3.1.13 获取步数汇总数据	25

3.1.14 获取距离汇总数据	26
3.1.15 获取卡路里汇总数据	28
3.1.16 获取一周日常活动数据	29
3.1.17 获取科学睡眠记录	31
3.1.18 查询心率告警参数	35
3.1.19 查询血氧告警参数	36
3.1.20 查询体温告警参数	37
3.2 实时数据监听	39
3.2.1 监听心率数据	39
3.2.2 停止监听心率数据	41
3.2.3 监听血氧数据	42
3.2.4 停止血氧心率数据	44
3.2.5 监听压力数据	45
3.2.6 停止监听压力数据	47
3.2.7 监听体温数据	48
3.2.8 停止监听体温数据	50
3.2.9 监听实时传感器数据	52
3.2.10 停止监听实时传感器数据	54
3.3 开关及参数设置	56
3.3.1 设置心率实时测量开关	56
3.3.2 设置血氧自动测量开关	56
3.3.3 设置压力自动测量开关	57
3.3.4 设置心率告警阈值	57
3.3.5 取消心率告警	58
3.3.6 设置血氧告警阈值	58
3.3.7 取消血氧告警	59
3.3.8 设置压力告警阈值	59
3.3.9 取消压力告警	60
3.3.10 设置压力校准值	60
3.3.11 获取是否设置压力校准	62
3.3.12 设置体温自动测量开关	63

3.3.13 设置体温告警阈值	63
3.3.14 取消体温告警	64
3.3.15 设置心率周期（自动）测量的频率	64
3.3.16 设置血氧周期（自动）测量的频率	65
3.3.17 设置压力周期（自动）测量的频率	66
3.3.18 设置体温周期（自动）测量的频率	67
3.3.19 设置心率告警触发次数	68
3.3.20 设置血氧告警触发次数	69
3.3.21 设置体温告警触发次数	69
3.3.22 设置运动健康连接开关	70
3.3.23 设置紧急联系人	70
3.3.24 设置实时传感器实时测量开关	71
3.4 系统事件监听	72
3.4.1 心率异常事件监听	72
3.4.2 血氧异常事件监听	74
3.4.3 压力异常事件监听	75
3.4.4 跌倒事件监听	76
3.4.5 功能键双击事件监听	77
3.4.6 设置定位服务开关修改事件监听	78
3.4.7 体温异常事件监听	79
3.4.8 一键告警事件监听	80
3.4.9 佩戴状态事件监听	82
3.4.10 来电事件监听	83
3.4.11 SOS 事件监听	84
3.5 系统事件启动服务	85
3.5.1 心率异常事件启动服务	85
3.5.2 血氧异常事件启动服务	87
3.5.3 压力异常事件启动服务	88
3.5.4 跌倒事件启动服务	89
3.5.5 功能键双击事件启动服务	91
3.5.6 设置定位服务开关修改事件启动服务	92

3.5.7 体温异常事件启动服务.....	93
3.5.8 一键告警事件启动服务.....	95
3.5.9 开机启动事件启动服务.....	96
3.5.10 佩戴状态改变事件启动服务	97
3.5.11 来电事件启动服务	99
3.5.12 SOS 事件启动服务.....	100
3.6 消息交互接口.....	101
3.6.1 注册消息接收器	101
3.6.2 发送消息	102
3.6.3 反注册消息接收器	103
3.7 附录.....	104
3.7.1 错误码对照表	104
3.7.2 功能模块名称对照表.....	105
3.7.3 历史和实时数据类型对照表.....	105
3.7.4 锻炼记录数据类型对照表.....	106
3.7.5 睡眠状态类型	106

前言

1.1 概述

本文档主要描述了三方鸿蒙 APP 在 Watch4 上使用鼎桥健康服务所使用到的 java 接口示例代码、数据类型定义、错误码等。

注意该文档接口仅仅支持鸿蒙 APP 使用。

1.2 读者对象

鼎桥健康管理平台开发人员、三方 APP 开发人员。

1.3 内容简介

1 变更说明

这里描述了各个版本间的变更信息。

2 概述

介绍 SDK 定制场景、API 层以及定义的接口。

3 API 参考

介绍系统向应用提供的 API。

一 变更说明

这里描述了各个版本间的变更信息。

1. V1.0_20211218, 第一次刷新, 提供心率, 血氧, 压力的历史数据查询和实时数据监听, 锻炼记录查询, 心率, 血氧, 压力的告警门限设置及开关。
2. V2.0_20211231, 增加压力校准, 体温数据查询和监听, 设置体温告警阈值, 体温告警开关和接口, 增加体温异常事件。
3. V2.1_20220214, 修正血氧, 压力, 睡眠等模块历史记录获取的调用事例。
4. V2.2_20220328, 增加心率, 血氧, 压力和体温模块的周期频率设置偶, 新增一键报警系统消息(事件)
5. V2.3_20220329, 增加佩戴系统事件, 睡眠状态获取接口
6. V2.4_20220412, 修正文档没有提供停止心率, 血氧, 压力和体温模块实时监听的接口问题。
7. V2.5_20220525, 新增支持多次点击 func 键方式触发一键报警。
8. V2.5.1_20220530, 添加一键报警事件多种触发方式的版本支持说明。
9. V2.7, 添加获取当日活动数据接口, 添加锻炼记录结束时间、静息心率、皮肤温度功能。
10. V2.8, 添加获取运动轨迹数据接口。
11. V2.9, 添加获取步数距离卡路里汇总数据、一周日常活动数据、查询是否设置压力校准值接口。
12. V2.10, 添加获取科学睡眠数据接口。
13. V2.11, 添加获取心率、血氧、体温告警参数, 设置心率、血氧、体温告警次数接口。
14. V2.12, 添加消息交互接收与发送接口。
15. V2.16, 更新 watch4 文档。

二 概述

示例代码重点展示调用 API 接口，为了简化代码省略了其他无关细节。

本文主要介绍鼎桥穿戴健康管理平台对外提供的获取穿戴设备运动及健康数据的 API 接口。应用通过使用这些接口，可以获取穿戴设备侧的设备信息、运动数据、健康数据以及向穿戴设备侧下发某些开关状态、配置信息等一系列功能。同时给出了各个接口的详细说明和使用限制。

2.1 接口管理类介绍

运动健康相关接口大部分都通过 `com.tdtech.ohos.health.HealthManager` 类提供。该类为单例模式，应用通过静态方法 `getInstance(Context context)` 获取实例对象。

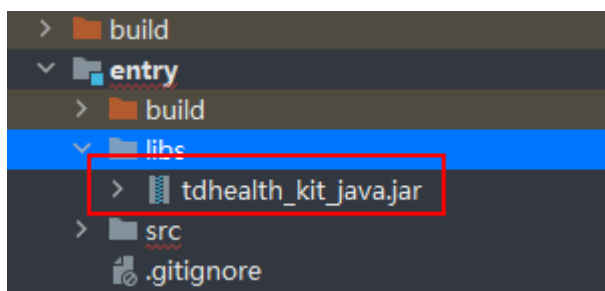
示例:

```
HealthManager healthManager = HealthManager.getInstance(getContext());
```

2.2 使用流程及注意事项

2.2.1 添加依赖

将提供的 Jar 包放入模块的 `libs` 目录下



在模块的 build.gradle 文件中依赖 jar 包（注意：这里依赖只能是编译时依赖，不能将 jar 包打包进自己应用的 hap 中，否则可能会产生冲突和异常）

```
dependencies {  
    //implementation fileTree(dir: 'libs', include: ['*.jar', '*.har'])  
    testImplementation 'junit:junit:4.13'  
    ohosTestImplementation 'com.huawei.ohos.testkit:runner:1.0.0.100'  
    compileOnly files('libs/tdhealth_kit_java.jar')  
}
```

2.2.2 服务的初始化及释放

在使用其他接口之前，需要先调用 init 方法进行初始化和连接服务。收到 onServiceReady 回调说明已经成功连接服务。确认连接到服务后再进行其他接口调用，否则可能由于服务没有初始化完成，导致 ServiceStateException 异常。

完成业务不再需要该服务的时候，可以调用 release 方法和服务断开连接。

示例：

```
@Override  
public void onStart(Intent intent) {  
    super.onStart(intent);  
    super.setUIContent(ResourceTable.Layout_ability_main);  
    // 初始化操作  
    HealthManager healthManager = HealthManager.getInstance(getContext());  
    healthManager.init(new HealthInitListener() {  
        @Override  
        public void onServiceReady() {  
            mIsServiceReady = true;  
            HiLog.info(logtable, "onServiceReady");  
        }  
    });  
  
    @Override  
    public void onServiceDisconnect(int errCode) {  
        mIsServiceReady = false;  
        HiLog.info(logtable, "onServiceDisconnect, errCode: %{public}d",  
errCode);  
    }  
});  
// 进行release操作  
@Override  
protected void onStop() {  
    super.onStop();  
    HealthManager.getInstance(getContext()).release();  
}
```

三 接口详细说明

3.1 历史数据查询

注意：目前历史数据的查询由于数据量传输大小的限制，起止时间间隔不能超过 **24 小时**，比如起始时间是 2021-11-22 10:00，结束时间就不能超过 2021-11-23 10:00

3.1.1 查询历史心率

- 接口原型

```
public String queryHeartRateRecords(HeartRateQueryConfig config) throws ServiceStateException
```

方法描述：查询历史心率数据

- 请求参数

参数名称	参数类型	参数描述	可选选项
config	HeartRateQueryConfig	查询的条件配置	M

查询参数配置选项

参数名称	参数类型	参数描述	可选选项
startTime	long	查询期间开始时间，默认为当天 0 点	O
endTime	long	查询区间截止时间，默认为当前时间	O

- 返回数据

返回具体的心率信息。为 JSON 格式数据。外层为 Json 对象，有以下几个参数：

name: String 类型，数据的名称，详细介绍参考[附录表格](#)

type: int 类型，数据类型，详细介绍参考[附录表格](#)

data: Json 数组，具体数据，里面每个对象包含两个数据(data: 具体心率值，silenceValue: 静息心率，timeStamp: 当前心率对应的时间戳)

code: 错误码，详见[错误码表格](#)

示例:

```
{
  "data": [{
    "data": 85,
    "silenceValue": 81,
    "timeStamp": 1637562406000
  }, {
    "data": 80,
    "silenceValue": 81,
    "timeStamp": 1637562046000
  }],
  "name": "heart_rate",
  "type": "history",
  "code": 0
}
```

● 调用代码实例

```
String data = HealthManager.getInstance(getContext())
    .queryHeartRateRecords(new HeartRateQueryConfig.Builder()
        .setStartTime(System.currentTimeMillis() - 647437)
        .setEndTime(System.currentTimeMillis())
        .build());
```

● 接口适配 SDK 版本
SP24

3.1.2 查询历史血氧

● 接口原型

public String querySpo2Records(Spo2QueryConfig config) throws
ServiceStateException

方法描述：查询历史血氧数据

● 请求参数

参数名称	参数类型	参数描述	可 选选项
config	Spo2QueryConfig	查询的条件配 置	M

查询参数配置选项

参数名称	参数 类型	参数描述	可选 选项
startTime	long	查询期间开始时间，默认为当天 0 点	○
endTime	long	查询区间截止时间，默认为当前时间	○

- 返回数据

返回具体的血氧信息。为 JSON 格式数据。外层为 Json 对象，有以下几个参数:

name: String 类型，数据的名称，详细介绍参考[附录表格](#)

type: int 类型，数据类型，详细介绍参考[附录表格](#)

data: Json 数组，具体数据，里面每个对象包含两个数据(data: 具体血氧值，timeStamp: 当前血氧对应的时间戳)

code: 错误码，详见[错误码表格](#)

示例:

```
{
  "data": [{
    "data": 97,
    "timeStamp": 1637562405000
  }, {
    "data": 98,
    "timeStamp": 1637562048000
  }],
  "name": "spo2",
  "type": "history",
  "code": 0
}
```

- 调用代码实例

```
String data = HealthManager.getInstance(getContext())
    .querySpo2Records(new Spo2QueryConfig.Builder()
        .setStartTime(System.currentTimeMillis() - 647437)
        .setEndTime(System.currentTimeMillis())
        .build());
```

- 接口适配 SDK 版本
SP24

3.1.3 查询历史压力数据

- 接口原型

public String queryStressRecords(StressQueryConfig config) throws
ServiceStateException

方法描述：查询历史压力数据

- 请求参数

参数名称	参数类型	参数描述	可 选选项
config	StressQueryConfig	查询的条件配 置	M

查询参数配置选项

参数名称	参数 类型	参数描述	可选 选项
startTime	long	查询期间开始时间，默认为当 天 0 点	O
endTime	long	查询区间截止时间，默认为当 前时间	O

- 返回数据

返回具体的压力信息。为 JSON 格式数据。外层为 Json 对象，有以下几个参数：

name: String 类型，数据的名称，详细介绍参考[附录表格](#)

type: int 类型，数据类型，详细介绍参考[附录表格](#)

data: Json 数组，具体数据，里面每个对象包含三个数据(data: 具体压力值，
startTimeStamp: 当前压力值的开始时间，endTimeStamp: 当前压力值的截止时间)

code: 错误码，详见[错误码表格](#)

示例:

```
{
  "data": [{
    "startTimeStamp": 1637562405000,
    "endTimeStamp": 1637562405000,
    "data": 50
  }, {
    "startTimeStamp": 1637562405000,
    "endTimeStamp": 1637562405000,
    "data": 45
  }],
  "name": " stress",
  "type": "history",
  "code": 0
}
```

● 调用代码实例

```
String data = HealthManager.getInstance(getContext())
    .queryStressRecords(new StressQueryConfig.Builder()
        .setStartTime(System.currentTimeMillis() - 647437)
        .setEndTime(System.currentTimeMillis())
        .build());
```

● 接口适配 SDK 版本
SP24

3.1.4 获取锻炼记录

● 接口原型

```
public String queryWorkoutRecords(WorkoutQueryConfig config) throws
ServiceStateException
```

方法描述: 查询历史锻炼数据

● 请求参数

参数名称	参数类型	参数描述	可 选选项
config	WorkoutQueryConfig	查询的条件配置	M

查询参数配置选项

参数名称	参数类型	参数描述	可选选项
------	------	------	------

startTime	long	查询期间开始时间，默认为当天 0 点	O
endTime	long	查询区间截止时间，默认为当前时间	O

● 返回数据

返回具体的锻炼信息。为 JSON 格式数据。外层为 Json 对象，有三个参数：

name: String 类型，数据的名称，详细介绍参考[附录表格](#)

type: String 类型，数据类型，详细介绍参考[附录表格](#)

data: Json 数组，具体数据，里面每个对象包含四个数据：calorie: 卡路里，distance: 运动距离，startTimeStamp: 运动开始时间，endTimeStamp: 运动结束时间，recordId: 运动记录 Id，workoutType: 运动类型（参考附录表格：[锻炼记录数据类型对照表格](#)）。

code: 错误码，详见[错误码表格](#)

示例：

```
{"code":0,"data":[{"calorie":60,"distance":900," startTimeStamp":1638793635000,
endTimeStamp":1638793645000,"workoutType":10},{"calorie":20,"distance":440,"
startTimeStamp":1638793166000,"endTimeStamp":1638793186000,"recordId":1,"workoutType":11}, {"calorie":44,"distance":650," startTimeStamp":1638786425000,
endTimeStamp":1638786455000,"workoutType":2}], "name":"workout", "type":"history"}
```

● 调用代码实例

```
String data = HealthManager.getInstance(getContext())
    .queryWorkoutRecords(new WorkoutQueryConfig.Builder()
        .setStartTime(System.currentTimeMillis() - 647437)
        .setEndTime(System.currentTimeMillis())
        .build());
```

● 接口适配 SDK 版本
SP24

3.1.5 获取睡眠记录

● 接口原型

```
public String querySleepRecords(SleepQueryConfig config) throws  
ServiceStateException
```

方法描述：查询历史睡眠数据

● 请求参数

参数名称	参数类型	参数描述	可选选项
config	SleepQueryConfig	查询的条件配置	M

查询参数配置选项

参数名称	参数类型	参数描述	可选选项
startTime	long	查询期间开始时间，默认为当天 0 点	O
endTime	long	查询区间截止时间，默认为当前时间	O
type	int	睡眠数据类型 (TYPE_SCIENCE(1)为科学睡眠记录，默认值，目前暂时只支持该类型)	O

● 返回数据

返回具体的睡眠信息。为 JSON 格式数据。外层为 Json 对象，有三个参数：

name: String 类型，数据的名称，详细介绍参考[附录表格](#)

type: int 类型，数据类型，详细介绍参考[附录表格](#)

data: Json 数组，具体数据，里面每个对象包含三个数据：endTimeStamp：睡眠截至时间，startTimeStamp：睡眠开始时间，type：睡眠类型（1：短睡，2：长睡）。

code: 错误码，详见[错误码表格](#)

示例：

- {"code":0,"data":[{"endTimeStamp":1638905820000,"startTimeStamp":1638895500000,"type":1},{"endTimeStamp":1638917160000,"startTimeStamp":1638905940000,"type":2}],"name":"sleep","type":"history"}

- 调用代码实例

```
String data = HealthManager.getInstance(getContext())
    .querySleepRecords(new SleepQueryConfig.Builder()
        .setStartTime(System.currentTimeMillis() - 647437)
        .setEndTime(System.currentTimeMillis())
        .setType(SleepQueryConfig.TYPE_SCIENCE)
        .build());
```

- 接口适配 SDK 版本
SP24

3.1.6 获取体温记录

- 接口原型

public String queryBodyTemperatureRecords(BodyTemperatureQueryConfig config)
throws ServiceStateException
方法描述：查询历史体温数据

- 请求参数

参数名称	参数类型	参数描述	可选选项
config	BodyTemperatureQueryConfig	查询的条件配置	M

查询参数配置选项

参数名称	参数类型	参数描述	可选选项
startTime	long	查询期间开始时间，默认为当天 0 点	O
endTime	long	查询区间截止时间，默认为当前时间	O

- 返回数据

返回具体的温度信息。为 JSON 格式数据。外层为 Json 对象，有三个参数：

name: String 类型，数据的名称，详细介绍参考[附录表格](#)

type: int 类型，数据类型，详细介绍参考[附录表格](#)

data: Json 数组，具体数据，里面每个对象包含一下几个数据：data：体温，skinTempera：体表温度；,timeStamp：时间戳；

code：错误码，详见[错误码表格](#)

示例：

```
{
  "data": [{
    "data": 36.5,
    "skinTempera": 34.5,
    "timeStamp": 1637562405000
  }, {
    "data": 36.7,
    "skinTempera": 34.6,
    "timeStamp": 1637562048000
  }],
  "name": "temperature",
  "type": "history",
  "code": 0
}
```

- 调用代码实例

```
HealthManager.getInstance(getContext())
    .queryBodyTemperatureRecords(new BodyTemperatureQueryConfig.Builder()
        .setStartTime(startTime)
        .setEndTime(endTime)
        .build());
```

- 接口适配 SDK 版本
SP24

3.1.7 获取步数

- 接口原型

```
public String queryStepsInRange(StepsQueryConfig config) throws ServiceStateException
```

方法描述：查询步数数据

- 请求参数

参数名称	参数类型	参数描述	可选选项
config	StepsQueryConfig	查询的条件配置	M

查询参数配置选项

参数名称	参数类型	参数描述	可选选项
startTime	long	查询期间开始时间，默认为当天 0 点	O
endTime	long	查询区间截止时间，默认为当前时间	O

- 返回数据

返回具体的步数信息。为 JSON 格式数据。外层为 Json 对象，有三个参数：

name: String 类型，数据的名称，详细介绍参考[附录表格](#)

type: int 类型，数据类型，详细介绍参考[附录表格](#)

data: Json 数组，具体数据，每个对象里面每个对象包含数据：data：步数

code: 错误码，详见[错误码表格](#)

示例：

```
{
  "data": [{
    "data": 101
  }],
  "name": "steps",
  "type": "history",
  "code": 0
}
```

- 调用代码实例

```
HealthManager.getInstance(getContext())
    .queryStepsInRange(new StepsQueryConfig.Builder()
        .setStartTime(startTime)
        .setEndTime(endTime)
        .build());
```

- 接口适配 SDK 版本
SP24

3.1.8 获取距离

- 接口原型

public String queryDistanceInRange(DistanceQueryConfig config) throws ServiceStateException

方法描述：查询距离数据

- 请求参数

参数名称	参数类型	参数描述	可选选项
config	DistanceQueryConfig	查询的条件配置	M

查询参数配置选项

参数名称	参数类型	参数描述	可选选项
startTime	long	查询期间开始时间，默认为当天 0 点	O
endTime	long	查询区间截止时间，默认为当前时间	O

- 返回数据

返回具体的距离信息。为 JSON 格式数据。外层为 Json 对象，有三个参数：

name: String 类型，数据的名称，详细介绍参考[附录表格](#)

type: int 类型，数据类型，详细介绍参考[附录表格](#)

data: Json 数组，具体数据，里面每个对象包含数据：data：距离。

code: 错误码，详见[错误码表格](#)

示例：

```
{
```

```
"data": [{
  "data": 396
}],
"name": "distance",
"type": "history",
"code": 0
}
```

● 调用代码实例

```
HealthManager.getInstance(getContext())
    .queryDistanceInRange (new DistanceQueryConfig.Builder()
        .setStartTime(startTime)
        .setEndTime(endTime)
        .build());
```

● 接口适配 SDK 版本
SP24

3.1.9 获取卡路里

● 接口原型

public String queryCalorieInRange(CalorieQueryConfig config) throws
ServiceStateException

方法描述：查询卡路里数据

● 请求参数

参数名称	参数类型	参数描述	可选选项
config	CalorieQueryConfig	查询的条件配置	M

查询参数配置选项

参数名称	参数类型	参数描述	可选选项
startTime	long	查询期间开始时间，默认为当天 0 点	O
endTime	long	查询区间截止时间，默认为当前时间	O

- 返回数据

返回具体的卡路里信息。为 JSON 格式数据。外层为 Json 对象，有三个参数：

name: String 类型，数据的名称，详细介绍参考[附录表格](#)

type: int 类型，数据类型，详细介绍参考[附录表格](#)

data: Json 数组，具体数据，里面每个对象包含一下几个数据：data：卡路里；

code: 错误码，详见[错误码表格](#)

示例：

```
{
  "data": [{
    "data": 300
  }],
  "name": "calorie",
  "type": "history",
  "code": 0
}
```

- 调用代码实例

```
HealthManager.getInstance(getContext())
    .queryCalorieInRange (new CalorieQueryConfig.Builder()
        .setStartTime(startTime)
        .setEndTime(endTime)
        .build());
```

- 接口适配 SDK 版本

SP24

3.1.10 获取睡眠状态

- 接口原型

public boolean isInSleep(long time) throws ServiceStateException,
GenericException

方法描述：查询指定时间点是否在入睡状态

- 请求参数

参数名称	参数类型	参数描述	可选选项
time	long	待查询的时间点的时间戳	M

- 返回数据

返回是否在入睡状态：

true：代表在入睡状态；

false：代表没有在睡眠状态，或者指定的时间点没有睡眠数据

- 调用代码实例

```
boolean isInSleep =  
HealthManager.getInstance(getContext()).isInSleep(System.currentTimeMillis());
```

- 接口适配 SDK 版本
SP24

3.1.11 获取当日活动数据

public String queryExerciseDailyRecord() throws ServiceStateException

方法描述：查询当日活动数据

- 返回数据

返回具体的卡路里信息。为 JSON 格式数据。外层为 Json 对象，有三个参数：

name: String 类型，数据的名称，详细介绍参考[附录表格](#)

type: int 类型，数据类型，详细介绍参考[附录表格](#)

data: Json 数组，具体数据，里面每个对象包含一下几个数据：**strengthTimes**：中高强度运动时间（分钟）；**totalTime**：总活动时长（小时）；

code: 错误码，详见[错误码表格](#)

示例：

```
{  
  "data": [{  
    "strengthTimes": 300,  
    "totalTime": 12,  
  }],  
  "name": "daily",
```

```
"type": "history",  
"code": 0  
}
```

- 调用代码实例

```
HealthManager.getInstance(getContext())  
    .queryCalorieInRange ();
```

- 接口适配 SDK 版本
SP24

3.1.12 获取运动轨迹数据

public String queryTracePointData(int recordId, TracePointListener listener) throws ServiceStateException

- 注意事项

- 1、recordId 需要从获取历史锻炼数据中获取
- 2、tracePointListener 每次获取 1200 个运动轨迹点
- 3、需要在获得完一次记录的轨迹数据后，再去请求下一个记录的运动轨迹

- 请求参数

参数名称	参数类型	参数描述	可选选项
recordId	Int	记录 Id	M
listener	TracePointListener	运动轨迹的 Listener	M

- 响应参数

运动轨迹数据会通过 TracePointListener 回调回来，正常会回调 onRevice 方法，如果出现异常或者问题会回调 onError 方法

参数名称	参数类型	参数描述	可选选项
code	int	错误码(0: 成功; 1: 完成所有数据接收; 其他失	—

		败，参考 错误码表格	
data	String	运动轨迹数据 (Json 格式)	—

data 数据格式说明：

返回具体的心率信息。为 JSON 格式数据。外层为 Json 对象，有以下几个参数：

name: String 类型，数据的名称，详细介绍参考[附录表格](#)

type: int 类型，数据类型，详细介绍参考[附录表格](#)

data: Json 数组，具体数据，，里面每个对象包含以下下几个数据：
altitude: 海拔；
latitude: 纬度； longitude: 经度； utcTime: 时间戳； valid: 数据是否合法；

code: 错误码，详见[错误码表格](#)

示例：

```
{
  "code": 0,
  "data": [
    {
      "altitude": 0,
      "latitude": 30.511451911223723,
      "longitude": 104.0868821409558,
      "utcTime": 1673503346,
      "valid": true
    },
    {
      "altitude": 0,
      "latitude": 30.511451680769554,
      "longitude": 104.08688253615551,
      "utcTime": 1673503347,
      "valid": true
    }
  ],
}
```

```
"name": "trace_point",  
"type": "history"  
}
```

- 接口适配 SDK 版本
暂不支持

3.1.13 获取步数汇总数据

- 接口原型

public String querySegmentStepInRange(StepsQueryConfig config) throws
ServiceStateException
方法描述：查询步数数据

- 请求参数

参数名称	参数类型	参数描述	可选选项
config	StepsQueryConfig	查询的条件配置	M

查询参数配置选项

参数名称	参数类型	参数描述	可选选项
startTime	long	查询期间开始时间，默认为当天 0 点	O
endTime	long	查询区间截止时间，默认为当前时间	O

- 返回数据

返回具体的步数信息。为 JSON 格式数据。外层为 Json 对象，有三个参数：

name: String 类型，数据的名称，详细介绍参考[附录表格](#)

type: int 类型，数据类型，详细介绍参考[附录表格](#)

data: Json 数组，具体数据，每个对象里面每个对象包含数据：data：步数, timestamp：时间戳

code: 错误码，详见[错误码表格](#)

示例:

```
{
  "data": [{
    "data": 85,
    "timeStamp": 1637562406000
  }, {
    "data": 80,
    "timeStamp": 1637562046000
  }],
  "name": "steps",
  "type": "history",
  "code": 0
}
```

● 调用代码实例

```
HealthManager.getInstance(getContext())
    .querySegmentStepsInRange (new StepsQueryConfig.Builder()
        .setStartTime(startTime)
        .setEndTime(endTime)
        .build());
```

● 接口适配 SDK 版本
SP24

3.1.14 获取距离汇总数据

● 接口原型

public String querySegmentDistanceInRange(DistanceQueryConfig config) throws
ServiceStateException

方法描述：查询距离数据

● 请求参数

参数名称	参数类型	参数描述	可选选项
config	DistanceQueryConfig	查询的条件配置	M

查询参数配置选项

参数名称	参数类型	参数描述	可选 选项
startTime	long	查询期间开始时间，默认为当天 0 点	○
endTime	long	查询区间截止时间，默认为当前时间	○

- 返回数据

返回具体的距离信息。为 JSON 格式数据。外层为 Json 对象，有三个参数：

name: String 类型，数据的名称，详细介绍参考[附录表格](#)

type: int 类型，数据类型，详细介绍参考[附录表格](#)

data: Json 数组，具体数据，里面每个对象包含数据：data：距离, timestamp：时间戳。

code: 错误码，详见[错误码表格](#)

示例：

```
{
  "data": [{
    "data": 85,
    "timeStamp": 1637562406000
  }, {
    "data": 80,
    "timeStamp": 1637562046000
  }],
  "name": "distance",
  "type": "history",
  "code": 0
}
```

- 调用代码实例

```
HealthManager.getInstance(getContext())
    .querySegmentDistanceInRange (new DistanceQueryConfig.Builder()
        .setStartTime(startTime)
        .setEndTime(endTime)
        .build());
```

- 接口适配 SDK 版本
SP24

3.1.15 获取卡路里汇总数据

● 接口原型

public String querySegmentCalorieInRange(CalorieQueryConfig config) throws ServiceStateException

方法描述：查询卡路里数据

● 请求参数

参数名称	参数类型	参数描述	可选选项
config	CalorieQueryConfig	查询的条件配置	M

查询参数配置选项

参数名称	参数类型	参数描述	可选选项
startTime	long	查询期间开始时间，默认为当天 0 点	O
endTime	long	查询区间截止时间，默认为当前时间	O

● 返回数据

返回具体的卡路里信息。为 JSON 格式数据。外层为 Json 对象，有三个参数：

name: String 类型，数据的名称，详细介绍参考[附录表格](#)

type: int 类型，数据类型，详细介绍参考[附录表格](#)

data: Json 数组，具体数据，里面每个对象包含一下几个数据：data：卡路里, timestamp：时间戳；

code: 错误码，详见[错误码表格](#)

示例:

```
{
  "data": [{
    "data": 85,
    "timeStamp": 1637562406000
  }]
```

```
    }, {  
      "data": 80,  
      "timeStamp": 1637562046000  
    }],  
    "name": "calorie",  
    "type": "history",  
    "code": 0  
  }  
}
```

- 调用代码实例

```
HealthManager.getInstance(getContext())  
    .querySegmentCalorieInRange(new CalorieQueryConfig.Builder()  
        .setStartTime(startTime)  
        .setEndTime(endTime)  
        .build());
```

- 接口适配 SDK 版本
SP24

3.1.16 获取一周日常活动数据

`public String queryExerciseDailyWeekRecord() throws ServiceStateException`

方法描述：查询当日活动数据

- 注意事项

1、手表第一次激活后可能会出现时间戳数据为 0 的情况，数据会跟随天数的增加而填充。

- 返回数据

返回具体的卡路里信息。为 JSON 格式数据。外层为 Json 对象，有三个参数：

name: String 类型，数据的名称，详细介绍参考[附录表格](#)

type: int 类型，数据类型，详细介绍参考[附录表格](#)

data: Json 数组，具体数据，里面每个对象包含一下几个数据：**timeStamps**：时间戳，**totalSteps**：总步数，**strengthTimes**：中高强度运动时间（分钟）；**totalTimes**：总活动时长（小时）；

code: 错误码，详见[错误码表格](#)

示例：

```
{
```



```
{
  "data": [{
    "timeStamps": 1679155199000,
    "totalSteps ": 4931,
    "strengthTimes": 300,
    "totalTime": 12,
  },
  {
    "timeStamps": 1679241599000,
    "totalSteps ": 3931,
    "strengthTimes": 300,
    "totalTime": 12,
  },
  {
    "timeStamps": 1679327999000,
    "totalSteps ": 3831,
    "strengthTimes": 300,
    "totalTime": 12,
  },
  {
    "timeStamps": 0,
    "totalSteps ": 0,
    "strengthTimes": 0,
    "totalTime": 0,
  },
  {
    "timeStamps": 0,
    "totalSteps ": 0,
    "strengthTimes": 0,
    "totalTime": 0,
  },
  {
    "timeStamps": 0,
    "totalSteps ": 0,
    "strengthTimes": 0,
    "totalTime": 0,
  },
  {
    "timeStamps": 0,
    "totalSteps ": 0,
    "strengthTimes": 0,
    "totalTime": 0,
  }
],
  "name": "daily",
  "type": "history",
  "code": 0
}
```

- 调用代码实例

```
HealthManager.getInstance(getContext())
    .queryExerciseDailyWeekRecord();
```

- 接口适配 SDK 版本

SP24

3.1.17 获取科学睡眠记录

- 接口原型

```
public String queryScientificSleepRecords(SleepQueryConfig config,
ScientificSleepListener listener) throws ServiceStateException
```

方法描述：查询历史睡眠数据

- 注意事项

每次只能请求 24 小时内数据

该接口为耗时操作，请放在子线程中执行

需要在获得完一次睡眠数据后，再去请求下一个记录

- 请求参数

参数名称	参数类型	参数描述	可选选项
config	SleepQueryConfig	查询的条件配置	M
listener	ScientificSleepListener	睡眠数据回调	M

查询参数配置选项

参数名称	参数类型	参数描述	可选选项
startTime	long	查询期间开始时间	M
endTime	long	查询区间截止时间	M
type	int	睡眠数据类型 (TYPE_SCIENCE(1)为科学睡眠记	M

		录，默认值，目前暂时只支持该类型)	
--	--	-------------------	--

科学睡眠数据会通过 TracePointListener 回调回来，正常会回调 onReceive 方法，如果出现异常或者问题会回调 onError 方法

参数名称	参数类型	参数描述	可选项
code	int	错误码(0：成功；其他失败，参考 错误码表格)	—
data	String	科学睡眠数据 (Json 格式)	—

● 返回数据

返回具体的睡眠信息。为 JSON 格式数据。外层为 Json 对象，有三个参数:

name: String 类型，数据的名称，详细介绍参考[附录表格](#)

type: int 类型，数据类型，详细介绍参考[附录表格](#)

data: 数据说明 (JSON 格式)：

errCode
睡眠结果错误码，0：成功
errCodeArr
睡眠结果错误时段
statusInDayArr
当天睡眠结果列表
deepSleepPartCnt
睡梦时长：分钟

fallAsleepTime
入睡时间点：时间戳（毫秒）
goBedTime
上床时间：时间戳（毫秒）
sleepEfficiency
睡眠效率，百分比，0~100
sleepLatency
睡眠潜伏期：毫秒
sleepScore
睡眠得分 0~100
sleepScoreOrigin
原始睡眠得分 0~100
snoreFreq
鼾声：每小时多少次
startTime
当天的时间戳（毫秒）
validData
数据有效性 0~100
wakeUpTime
醒来时间：时间戳（毫秒）
statusInMinuteArr
每分钟状态列表：具体含义请参照 3.6.5 睡眠状态类型表

code：错误码，详见[错误码表格](#)

示例:

[illegible]

```
    ],  
    "name": "scientific_sleep",  
    "type": "history",  
    "code": 0  
}
```

- 调用代码实例

```
HealthManager.getInstance(getContext())  
                .queryScientificSleepRecords(new  
SleepQueryConfig.Builder()  
                .setStartTime(startTime)  
                .setEndTime(endTime)  
                .build(), new ScientificSleepListener()  
{  
    @Override  
    public void onReceive(int i, String s) {  
        HiLog.debug(TAG, "onReceive i: " + i +  
" s: " + s);  
        BaseBean<ScientificSleepData>  
scientificSleepData = new Gson().fromJson(s,  
        new  
TypeToken<BaseBean<ScientificSleepData>>() {  
        }.getType());  
        ScientificSleepData data1 =  
scientificSleepData.data.get(0);  
        HiLog.debug(TAG, "onReceive i: " + i +  
" s: " + data1.statusInMinuteArr.get(0).toString());  
        mHandler.postTask(() -> {  
showQueryResult(scientificSleepData);  
        });  
    }  
    @Override  
    public void onError(int i) {  
        HiLog.debug(TAG, "onError i: " + i);  
    }  
});
```

- 接口适配 SDK 版本
SP24

3.1.18 查询心率告警参数

- 接口原型

```
public String queryHeartRemindParam() throws ServiceStateException
```

方法描述：查询心率告警参数

- 返回数据

返回具体的卡路里信息。为 JSON 格式数据。外层为 Json 对象，有三个参数：

name: String 类型，数据的名称，详细介绍参考[附录表格](#)

type: int 类型，数据类型，详细介绍参考[附录表格](#)

data: Json 数组，具体数据，里面每个对象包含一下几个数据：raiseSwitch 心率过高告警开关；raiseValue 上限阈值；downSwitch 下限开关； downValue 下限阈值； remindCnt 触发告警次数

code: 错误码，详见[错误码表格](#)

示例：

```
{
  "data": [{
    "raiseSwitch": 1,
    "raiseValue": 120,
    "downSwitch": 1,
    "downValue": 50,
    "remindCnt": 3
  }],
  "name": "heart_remind",
  "type": "history",
  "code": 0
}
```

- 调用代码实例
- `HealthManager.getInstance(getContext()).queryHeartRemindParam();`
- 接口适配 SDK 版本
SP24

3.1.19 查询血氧告警参数

- 接口原型

`public String querySpo2RemindParam() throws ServiceStateException`
方法描述：查询血氧告警参数

- 返回数据

返回具体的卡路里信息。为 JSON 格式数据。外层为 Json 对象，有三个参数：

name: String 类型，数据的名称，详细介绍参考[附录表格](#)

type: int 类型，数据类型，详细介绍参考[附录表格](#)

data: Json 数组，具体数据，里面每个对象包含一下几个数据：switchValue 告警开关；
remindValue 告警阈值； remindCnt 触发告警次数

code: 错误码，详见[错误码表格](#)

示例:

```
{
  "data": [{
    "switchValue": 1,
    "remindValue": 95
    "remindCnt": 3
  }],
  "name": "spo2_remind",
  "type": "history",
  "code": 0
}
```

- 调用代码实例
- `HealthManager.getInstance(getContext()).querySpo2RemindParam();`
- 接口适配 SDK 版本
SP24

3.1.20 查询体温告警参数

- 接口原型

`public String queryTemperatureRemindParam() throws ServiceStateException`
方法描述：查询心率告警参数

- 返回数据

返回具体的卡路里信息。为 JSON 格式数据。外层为 Json 对象，有三个参数:

name: String 类型，数据的名称，详细介绍参考[附录表格](#)

type: int 类型，数据类型，详细介绍参考[附录表格](#)

data: Json 数组，具体数据，里面每个对象包含一下几个数据：remindHighSwitch 体温过高告警开关； highRemindValue 上限阈值； remindLowSwitch 下限开关； lowRemindValue 下限阈值； remindCnt 触发告警次数

code: 错误码，详见[错误码表格](#)

示例:


```
{
  "data": [{
    "remindHighSwitch ": 1,
    "highRemindValue ": 380,
    "remindLowSwitch ": 1,
    "lowRemindValue ": 350,
    "remindCnt": 3
  }],
  "name": "temperature_remind ",
  "type": "history",
  "code": 0
}
```

- 调用代码实例
- `HealthManager.getInstance(getContext()).queryTemperatureRemindParam();`
- 接口适配 SDK 版本
SP24

3.2 实时数据监听

3.2.1 监听心率数据

- 接口原型

`public HeartRateSensorAgent getHeartRateSensor()`

方法描写：获取心率 SensorAgent

`public void startMeasureHeartRate(IBodySensorDataListener listener)`

方法描述：开始监听心率实时数据

- 注意事项

需要在开启心率自动(连续)测量开关才能正常监听到心率的实时数据

- 请求参数

参数名称	参数类型	参数描述	可 选选项
listener	IBodySensorDataListener	实时心率的 Listener	M

- 响应参数

心率数据会通过 IBodySensorDataListener 回调回来，正常会回调 onSensorDataUpdate 方法，如果出现异常或者问题会回调 onError 方法

参数名称	参数类型	参数描述	可 选选项
code	int	错误码(0：成功，其他失败，参考 错误码表格)	— —
data	String	心率数据(Json 格式)	— —

data 数据格式说明：

返回具体的心率信息。为 JSON 格式数据。外层为 Json 对象，有以下几个参数：

name: String 类型, 数据的名称, 详细介绍参考[附录表格](#)

type: int 类型, 数据类型, 详细介绍参考[附录表格](#)

data: Json 数组, 具体数据, 里面每个对象包含两个数据(data: 具体心率值, timeStamp: 当前心率对应的时间戳)

code: 错误码, 详见[错误码表格](#)

示例:

```
{
  "data": [{
    "data": 85,
    "timeStamp": 1637562406000
  }],
  "name": "heart_rate",
  "type": "realtime",
  "code": 0
}
```

- 调用代码实例

```
mListener = new IBodySensorDataListener() {
    @Override
    public void onSensorDataUpdate(int errCode, String s) {
        HiLog.info(loglable, "realtime heart rate, code is %{public}d, data
is : %{public}s", errCode, s);
        if (errCode == HealthManager.ERR_SUCCESS) {
            JSONObject object = new JsonParser().parse(s).getAsJsonObject();
            String name = object.get("name").getString();
            String type = object.get("type").getString();
            String data = object.get("data").getAsJsonArray().toString();
            HiLog.info(loglable, "realtime heart rate, name: %{public}s,
type: %{public}s, data: %{public}s",
                        name, type, data);
        }
    }
    @Override
    public void onError(int errCode) {
        HiLog.info(loglable, "realtime heart rate error : %{public}d", errCode);
    }
};
HealthManager.getInstance(getContext())
    .getHeartRateSensor()
    .startMeasureHeartRate(mListener);
```

- 接口适配 SDK 版本
SP24

3.2.2 停止监听心率数据

- 接口原型

`public HeartRateSensorAgent getHeartRateSensor()`

方法描写：获取心率 SensorAgent

`public void stopMeasureHeartRate(IBodySensorDataListener listener)`

方法描述：停止监听心率实时数据

- 请求参数

参数名称	参数类型	参数描述	可 选 项
listener	IBodySensorDataListener	开启监听实时心率的Listener	M

- 响应参数

如果停止出现异常或者问题会回调 `onError` 方法

参数名称	参数类型	参数描述	可 选选项
code	int	错误码(0: 成功, 其他失败, 参考 错误码表格)	—

- 调用代码实例

```
mListener = new IBodySensorDataListener() {
    @Override
    public void onSensorDataUpdate(int errCode, String s) {
        HiLog.info(loglable, "realtime heart rate, code is %{public}d, data is : %{public}s", errCode, s);
        if (errCode == HealthManager.ERR_SUCCESS) {
            JsonObject object = new JsonParser().parse(s).getAsJsonObject();
            String name = object.get("name").getAsString();
            String type = object.get("type").getAsString();
            String data = object.get("data").getAsJsonArray().toString();
            HiLog.info(loglable, "realtime heart rate, name: %{public}s, type: %{public}s, data: %{public}s", name, type, data);
        }
    }
    @Override
```

```
public void onError(int errCode) {  
    HiLog.info(loglable, "realtime heart rate error : %{public}d", errCode);  
}  
};  
HealthManager.getInstance(getContext())  
    .getHeartRateSensor()  
    .stopMeasureHeartRate (mListener);
```

- 接口适配 SDK 版本
SP24

3.2.3 监听血氧数据

- 接口原型

```
public Spo2SensorAgent getSpo2Sensor()  
方法描写：获取血氧 SensorAgent  
  
public void startMeasureSpo2(IBodySensorDataListener listener)  
方法描述：开始监听血氧实时数据
```

- 注意事项

需要在开启血氧自动测量开关才能正常监听到血氧的实时数据

- 请求参数

参数名称	参数类型	参数描述	可选选项
listener	IBodySensorDataListener	实时数据的 Listener	M

- 响应参数

数据会通过 IBodySensorDataListener 回调回来，正常会回调 onSensorDataUpdate 方法，如果出现异常或者问题会回调 onError 方法

参数名称	参数类型	参数描述	可选选项
code	int	错误码(0：成功，其他失败，参考 错误码表格)	—

data	String	血氧数据(Json 格式)	—
------	--------	------------------	---

data 数据格式说明:

返回具体的血氧信息。为 JSON 格式数据。外层为 Json 对象，有以下几个参数:

name: String 类型，数据的名称，详细介绍参考[附录表格](#)

type: int 类型，数据类型，详细介绍参考[附录表格](#)

data: Json 数组，具体数据，里面每个对象包含两个数据(data: 具体血氧值，timeStamp: 当前血氧对应的时间戳)

code: 错误码，详见[错误码表格](#)

示例:

```
{
  "data": [{
    "data": 95,
    "timeStamp": 1637562406000
  }],
  "name": "spo2 ",
  "type": "realtime",
  "code": 0
}
```

● 调用代码实例

```
mListener = new IBodySensorDataListener() {
    @Override
    public void onSensorDataUpdate(int errCode, String s) {
        HiLog.info(loglable, "realtime spo2, code is %{public}d, data
is : %{public}s", errCode, s);
        if (errCode == HealthManager.ERR_SUCCESS) {
            JsonObject object = new JsonParser().parse(s).getAsJsonObject();
            String name = object.get("name").getString();
            String type = object.get("type").getString();
            String data = object.get("data").getAsJsonArray().toString();
            HiLog.info(loglable, "realtime spo2, name: %{public}s,
type: %{public}s, data: %{public}s",
                        name, type, data);
        }
    }
    @Override
    public void onError(int errCode) {
        HiLog.info(loglable, "realtime spo2 error : %{public}d", errCode);
    }
}
```

```
};  
HealthManager.getInstance(getContext())  
    .getSpo2Sensor()  
    .startMeasureSpo2(mListener);
```

- 接口适配 SDK 版本
SP24

3.2.4 停止血氧心率数据

- 接口原型

```
public Spo2SensorAgent getSpo2Sensor()
```

方法描写：获取血氧 SensorAgent

```
public void stopMeasureSpo2(IBodySensorDataListener listener)
```

方法描述：停止血氧心率实时数据

- 请求参数

参数名称	参数类型	参数描述	可 选 选 项
listener	IBodySensorDataListener	开启监听实时血氧 的Listener	M

- 响应参数

如果停止出现异常或者问题会回调 onError 方法

参数名称	参数类型	参数描述	可 选选项
code	int	错误码(0: 成功, 其他失败, 参考 错误码表格)	—

- 调用代码实例

```
mListener = new IBodySensorDataListener() {  
    @Override  
    public void onSensorDataUpdate(int errCode, String s) {  
        HiLog.info(loglable, "realtime spo2, code is %{public}d, data  
is : %{public}s", errCode, s);  
        if (errCode == HealthManager.ERR_SUCCESS) {  
            JsonObject object = new JsonParser().parse(s).getAsJsonObject();
```

```
String name = object.get("name").getAsString();
String type = object.get("type").getAsString();
String data = object.get("data").getAsJsonArray().toString();
HiLog.info(loglable, "realtime spo2, name: %{public}s,
type: %{public}s, data: %{public}s",
            name, type, data);
}
}
@Override
public void onError(int errCode) {
    HiLog.info(loglable, "realtime spo2 error : %{public}d", errCode);
}
};
HealthManager.getInstance(getContext())
    .getSpo2Sensor()
    .stopMeasureSpo2 (mListener);
```

- 接口适配 SDK 版本
SP24

3.2.5 监听压力数据

- 接口原型

public StressSensorAgent getStressSensor()

方法描写：获取压力 SensorAgent

public void startMeasureStress(IBodySensorDataListener listener)

方法描述：开始监听压力实时数据

- 注意事项

需要在开启压力自动测量开关才能正常监听到压力的实时数据，在开启自动测量之前需要先通过“运动健康”进行压力校准。详细信息参考[设置压力自动测量开关接口](#)

- 请求参数

参数名称	参数类型	参数描述	可 选选项
listener	IBodySensorDataListener	实时数据的 Listener	M

- 响应参数

数据会通过 IBodySensorDataListener 回调回来，正常会回调 onSensorDataUpdate 方法，如果出现异常或者问题会回调 onError 方法

参数名称	参数类型	参数描述	可选选项
code	int	错误码(0: 成功, 其他失败, 参考 错误码表格)	—
data	String	压力数据(Json 格式)	—

data 数据格式说明:

返回具体的压力信息。为 JSON 格式数据。外层为 Json 对象，有以下几个参数:

name: String 类型，数据的名称，详细介绍参考[附录表格](#)

type: int 类型，数据类型，详细介绍参考[附录表格](#)

data: Json 数组，具体数据，里面每个对象包含三个数据(data: 具体压力值, startTimeStamp: 当前压力值的开始时间, endTimeStamp: 当前压力值的截止时间)

code: 错误码，详见[错误码表格](#)

示例:

```
{
  "data": [{
    "startTimeStamp": 1637562405000,
    "endTimeStamp": 1637562405000,
    "data": 50
  }],
  "name": " stress",
  "type": "realtime",
  "code": 0
}
```

● 调用代码实例

```
mListener = new IBodySensorDataListener() {
    @Override
    public void onSensorDataUpdate(int errCode, String s) {
        HiLog.info(loglable, "realtime stress, code is %{public}d, data is : %{public}s", errCode, s);
        if (errCode == HealthManager.ERR_SUCCESS) {
            JsonObject object = new JsonParser().parse(s).getAsJsonObject();
        }
    }
}
```

```
String name = object.get("name").getAsString();
String type = object.get("type").getAsString();
String data = object.get("data").getAsJsonArray().toString();
HiLog.info(loglable, "realtime stress, name: %{public}s,
type: %{public}s, data: %{public}s",
            name, type, data);
}
}
@Override
public void onError(int errCode) {
    HiLog.info(loglable, "realtime stress error : %{public}d", errCode);
}
};
HealthManager.getInstance(getContext())
    .getStressSensor()
    .startMeasureStress(mListener);
```

- 接口适配 SDK 版本
SP24

3.2.6 停止监听压力数据

- 接口原型

```
public StressSensorAgent getStressSensor()
方法描写：获取压力 SensorAgent

public void stopMeasureStress(IBodySensorDataListener listener)
方法描述：停止监听压力实时数据
```

- 请求参数

参数名称	参数类型	参数描述	可 选 项
listener	IBodySensorDataListener	开启实时数据的 Listener	M

- 响应参数

如果停止出现异常或者问题会回调 onError 方法

参数名称	参数类型	参数描述	可 选选项
------	------	------	----------

code	int	错误码(0: 成功, 其他失败, 参考 错误码表格)	—
------	-----	---	---

● 调用代码实例

```
mListener = new IBodySensorDataListener() {
    @Override
    public void onSensorDataUpdate(int errCode, String s) {
        HiLog.info(loglable, "realtime stress, code is %{public}d, data
is : %{public}s", errCode, s);
        if (errCode == HealthManager.ERR_SUCCESS) {
            JsonObject object = new JsonParser().parse(s).getAsJsonObject();
            String name = object.get("name").getString();
            String type = object.get("type").getString();
            String data = object.get("data").getAsJsonArray().toString();
            HiLog.info(loglable, "realtime stress, name: %{public}s,
type: %{public}s, data: %{public}s",
                        name, type, data);
        }
    }
    @Override
    public void onError(int errCode) {
        HiLog.info(loglable, "realtime stress error : %{public}d", errCode);
    }
};
HealthManager.getInstance(getContext())
    .getStressSensor()
    .stopMeasureStress(mListener);
```

● 接口适配 SDK 版本
SP24

3.2.7 监听体温数据

● 接口原型

```
public BodyTemperatureSensorAgent getBodyTemperatureSensor()
方法描写：获取温度 SensorAgent

public void startMeasureBodyTemperature(IBodySensorDataListener listener)
方法描述：开始监听温度实时数据
```

● 注意事项

需要在开启温度自动测量开关才能监听到正常的体温数据

● 请求参数

参数名称	参数类型	参数描述	可选选项
listener	IBodySensorDataListener	实时数据的 Listener	M

- 响应参数

数据会通过 IBodySensorDataListener 回调回来，正常会回调 onSensorDataUpdate 方法，如果出现异常或者问题会回调 onError 方法

参数名称	参数类型	参数描述	可选选项
code	int	错误码(0: 成功, 其他失败, 参考 错误码表格)	—
data	String	体温数据(Json 格式)	—

data 数据格式说明:

返回具体体温信息。为 JSON 格式数据。外层为 Json 对象，有以下几个参数:

name: String 类型，数据的名称，详细介绍参考[附录表格](#)

type: int 类型，数据类型，详细介绍参考[附录表格](#)

data: Json 数组，具体数据，内部具体的数据信息，data: 体温，timeStamp: 时间戳;

code: 错误码，详见[错误码表格](#)

示例:

```
{
  "data": [{
    "timeStamp": 1637562405000,
    "data": 36.5
  }],
  "name": "temperature",
  "type": "realtime",
  "code": 0
}
```

● 调用代码实例

```
mListener = new IBodySensorDataListener() {
    @Override
    public void onSensorDataUpdate(int errCode, String s) {
        HiLog.info(loglable, "realtime temperature, code is %{public}d, data
is : %{public}s", errCode, s);
        if (errCode == HealthManager.ERR_SUCCESS) {
            JsonObject object = new JsonParser().parse(s).getAsJsonObject();
            String name = object.get("name").getString();
            String type = object.get("type").getString();
            String data = object.get("data").getAsJsonArray().toString();
            HiLog.info(loglable, "realtime temperature, name: %{public}s,
type: %{public}s, data: %{public}s",
                        name, type, data);
        }
    }
    @Override
    public void onError(int errCode) {
        HiLog.info(loglable, "realtime temperature error : %{public}d",
errCode);
    }
};
HealthManager.getInstance(getContext())
    .getBodyTemperatureSensor ()
    .startMeasureBodyTemperature (mListener);
```

- 接口适配 SDK 版本
SP24

3.2.8 停止监听体温数据

● 接口原型

```
public BodyTemperatureSensorAgent getBodyTemperatureSensor()
方法描写：获取温度 SensorAgent

public void stopMeasureBodyTemperature(IBodySensorDataListener listener)
方法描述：停止监听温度实时数据
```

● 请求参数

参数名称	参数类型	参数描述	可 选 项
listener	IBodySensorDataListener	开启实时数据的 Listener	M

● 响应参数

如果停止出现异常或者问题会回调 onError 方法

参数名称	参数类型	参数描述	可选选项
code	int	错误码(0: 成功, 其他失败, 参考 错误码表格)	—

● 调用代码实例

```
mListener = new IBodySensorDataListener() {
    @Override
    public void onSensorDataUpdate(int errCode, String s) {
        HiLog.info(loglable, "realtime temperature, code is %{public}d, data is : %{public}s", errCode, s);
        if (errCode == HealthManager.ERR_SUCCESS) {
            JsonObject object = new JsonParser().parse(s).getAsJsonObject();
            String name = object.get("name").getAsString();
            String type = object.get("type").getAsString();
            String data = object.get("data").getAsJsonArray().toString();
            HiLog.info(loglable, "realtime temperature, name: %{public}s, type: %{public}s, data: %{public}s", name, type, data);
        }
    }
    @Override
    public void onError(int errCode) {
        HiLog.info(loglable, "realtime temperature error : %{public}d", errCode);
    }
};
HealthManager.getInstance(getContext())
    .getBodyTemperatureSensor()
    .stopMeasureBodyTemperature(mListener);
```

● 接口适配 SDK 版本
SP24

3.2.9 监听实时传感器数据

- 接口原型

`public MotionSensorAgent getMotionSensor()`

方法描写：获取实时传感器 SensorAgent

`public void startMeasureMotion(IBodySensorDataListener listener)`

方法描述：开始监听心率实时数据

- 注意事项

需要在开启实时传感器自动(连续)测量开关才能正常监听到实时传感器的实时数据

- 请求参数

参数名称	参数类型	参数描述	可 选选项
listener	IBodySensorDataListener	实时传感器的 Listener	M

- 响应参数

实时传感器数据会通过 IBodySensorDataListener 回调回来，正常会回调 onSensorDataUpdate 方法，如果出现异常或者问题会回调 onError 方法

参数名称	参数类型	参数描述	可 选选项
code	int	错误码(0：成功，其他失败，参考 错误码表格)	— —
data	String	实时传感器数据(Json 格式)	— —

data 数据格式说明：

返回具体的心率信息。为 JSON 格式数据。外层为 Json 对象，有以下几个参数：

name: String 类型，数据的名称，详细介绍参考[附录表格](#)

type: int 类型，数据类型，详细介绍参考[附录表格](#)

data: Json 数组，具体数据，里面每个对象包含两个数据(data: 具体心率值，timeStamp: 当前心率对应的时间戳)

code: 错误码，详见[错误码表格](#)

示例:

```
{
  "data": [{
    "accs": [          加速度数据，此处使用一组示意。
      {
        "x": -80,
        "y": 991,
        "z": 4020
      }
    ],
    "gyros": [         陀螺仪数据，此处使用一组示意
      {
        "a": 56,
        "b": -45,
        "r": -5
      }
    ],
    "ppgs": [          ppg数据，此处使用一组示意。
      {
        "ambgreen1": 438,
        "ambgreen2": 11602,
        "ambir": 1471,
        "ambred": 946,
        "green1": -326931,
        "green2": -144584,
        "ir": 641080,
        "red": 1219
      }
    ],
    "timestamps": {    时间戳组
      "acc": 1605184729997,  ppg unix时间戳
      "gyro": 1605184729999,  加速度unix时间戳
      "ppg": 1605184730082    陀螺仪unix时间戳
    }
  }],
  "name": "heart_rate",
  "type": "realtime",
  "code": 0
}
```

- 调用代码实例

```
mListener = new IBodySensorDataListener() {
    @Override
```



```
public void onSensorDataUpdate(int errCode, String s) {
    HiLog.info(loglable, "realtime heart rate, code is %{public}d, data
is : %{public}s", errCode, s);
    if (errCode == HealthManager.ERR_SUCCESS) {
        JsonObject object = new JsonParser().parse(s).getAsJsonObject();
        String name = object.get("name").getAsString();
        String type = object.get("type").getAsString();
        String data = object.get("data").getAsJsonArray().toString();
        HiLog.info(loglable, "realtime heart rate, name: %{public}s,
type: %{public}s, data: %{public}s",
name, type, data);
    }
}
@Override
public void onError(int errCode) {
    HiLog.info(loglable, "realtime heart rate error : %{public}d",
errCode);
}
};
HealthManager.getInstance(getContext())
    .getMotionSensor()
    .startMeasureMotion(mListener);
```

- 接口适配 SDK 版本
暂不支持

3.2.10 停止监听实时传感器数据

- 接口原型

```
public MotionSensorAgent getMotionSensor()
方法描写：获取实时传感器 SensorAgent

public void stopMeasureMotion(IBodySensorDataListener listener)
方法描述：停止监听实时传感器实时数据
```

- 请求参数

参数名称	参数类型	参数描述	可 选 项
listener	IBodySensorDataListener	开启监听实时心率的Listener	M

- 响应参数

如果停止出现异常或者问题会回调 onError 方法

参数名称	参数类型	参数描述	可 选选项
code	int	错误码(0: 成功, 其他失败, 参考 错误码表格)	— —

● 调用代码实例

```
mListener = new IBodySensorDataListener() {
    @Override
    public void onSensorDataUpdate(int errCode, String s) {
        HiLog.info(loglable, "realtime heart rate, code is %{public}d, data
is : %{public}s", errCode, s);
        if (errCode == HealthManager.ERR_SUCCESS) {
            JsonObject object = new JsonParser().parse(s).getAsJsonObject();
            String name = object.get("name").getString();
            String type = object.get("type").getString();
            String data = object.get("data").getAsJsonArray().toString();
            HiLog.info(loglable, "realtime heart rate, name: %{public}s,
type: %{public}s, data: %{public}s",
                        name, type, data);
        }
    }
    @Override
    public void onError(int errCode) {
        HiLog.info(loglable, "realtime heart rate error : %{public}d",
errCode);
    }
};
HealthManager.getInstance(getContext())
    .getMotionSensor()
    .stopMeasureMotion(mListener);
```

● 接口适配 SDK 版本

暂不支持

3.3 开关及参数设置

3.3.1 设置心率实时测量开关

- 接口原型

`public void switchOnHeartRateContinueMeasure() throws ServiceStateException`

方法描述：开启心率连续测量

`public void switchOffHeartRateContinueMeasure() throws ServiceStateException`

方法描述：关闭心率连续测量

- 调用代码实例

```
// 开启实时测量开关
HealthManager.getInstance(getContext()).switchOnHeartRateContinueMeasure();
```

```
// 关闭实时测量开关
HealthManager.getInstance(getContext()).switchOffHeartRateContinueMeasure();
```

- 接口适配 SDK 版本

SP24

3.3.2 设置血氧自动测量开关

- 接口原型

`public void switchOnSpo2AutoMeasure() throws ServiceStateException`

方法描述：开启血氧自动测量

`public void switchOffSpo2AutoMeasure() throws ServiceStateException`

方法描述：关闭血氧自动测量

- 调用代码实例

```
// 开启自动测量开关
HealthManager.getInstance(getContext()).switchOnSpo2AutoMeasure();
```

```
// 关闭自动测量开关
HealthManager.getInstance(getContext()).switchOffSpo2AutoMeasure();
```

- 接口适配 SDK 版本

SP24

3.3.3 设置压力自动测量开关

- 接口原型

`public void switchOnStressAutoMeasure() throws WrongStateException, ServiceStateException`

方法描述：开启压力自动测量

`public void switchOffStressAutoMeasure() throws ServiceStateException`

方法描述：关闭压力自动测量

- 注意事项

开启压力自测量之前，需要先通过设置压力校准参数，否则可能会出现压力测试数据不准确甚至发生异常导致压力无法出值的情况。

- 调用代码实例

```
// 开启自动测量开关
HealthManager.getInstance(getContext()).switchOnStressAutoMeasure();
```

```
// 关闭自动测量开关
HealthManager.getInstance(getContext()).switchOffStressAutoMeasure();
```

- 接口适配 SDK 版本
SP24

3.3.4 设置心率告警阈值

- 接口原型

`public void setHeartRateWarning(int high, int low) throws ServiceStateException`

方法描述：设置心率告警阈值

- 注意事项

该接口会自动打开心率告警的开关，设置的时候需要同时设置上限和下限。

- 请求参数

参数名称	参数类型	参数描述	可 选选项
high	int	心率告警的上 限(心率超过该值会	M

		触发告警), 范围 (0,255)	
low	int	心率告警的下 限 (心率低于该值 会触发告警) ,范围 (0,255)	M

- 调用代码实例

```
// 设置告警上限 100, 下限 50
HealthManager.getInstance(getContext()).setHeartRateWarning(100, 50);
```

- 接口适配 SDK 版本
SP24

3.3.5 取消心率告警

- 接口原型

```
public void cancelHeartRateWarning() throws ServiceStateException
```

方法描述：取消心率告警

- 调用代码实例

```
HealthManager.getInstance(getContext()).cancelHeartRateWarning();
```

- 接口适配 SDK 版本
SP24

3.3.6 设置血氧告警阈值

- 接口原型

```
public void setSpo2Warning(int value) throws ServiceStateException
```

方法描述：设置血氧告警阈值

- 注意事项

该接口会自动打开血氧告警的开关

- 请求参数

参数名称	参数类型	参数描述	可 选选项
value	int	血氧告警的阈值，范围(0,100]	M

- 调用代码实例

```
// 设置血氧低于 90 触发告警  
HealthManager.getInstance(getContext()).setSpo2Warning(90);
```

- 接口适配 SDK 版本
SP24

3.3.7 取消血氧告警

- 接口原型

```
public void cancelSpo2Warning() throws ServiceStateException  
方法描述：取消血氧告警
```

- 调用代码实例

```
HealthManager.getInstance(getContext()).cancelSpo2Warning();
```

- 接口适配 SDK 版本
SP24

3.3.8 设置压力告警阈值

- 接口原型

```
public void setStressWarning(int value) throws ServiceStateException  
方法描述：设置压力告警阈值
```

- 注意事项

该接口会自动打开压力告警的开关

- 请求参数

参数名称	参数类型	参数描述	可 选选项
------	------	------	----------

value	int	压力告警的阈值, 范围(0,100]	M
-------	-----	--------------------	---

- 调用代码实例

```
// 设置压力高于 66 触发告警
HealthManager.getInstance(getContext()).setStressWarning(66);
```

- 接口适配 SDK 版本

SP24

注意：SP8版本压力告警功能存在问题，设置压力阈值后收不到压力告警，在后续版本上解决。

3.3.9 取消压力告警

- 接口原型

```
public void cancelStressWarning() throws ServiceStateException
```

方法描述：取消压力告警

- 调用代码实例

```
HealthManager.getInstance(getContext()).cancelStressWarning();
```

- 接口适配 SDK 版本

SP24

3.3.10 设置压力校准值

- 接口原型

```
public void setStressInquiryData(int score, long startTime, float[] inquiryData) throws ServiceStateException, GenericException
```

方法描述：设置压力校准值

- 注意事项

注意参数的合法性，如果参数不正确可能会导致 `GenericException` 异常。

- 请求参数

参数名称	参数类型	参数描述	可选选项
------	------	------	------

score	int	校准的压力分数	M
startTime	int	校准的时间，一般传入当前时间即可	M
inquiryData	fload[]	校准的特征值，大小 12 的数组，依次代表： 1. AMo: 直方图高度，单位：个 2. HRVar: 心率的偏差，单位：beat/min 3. MxDMn: 直方图总底宽，单位：ms 4. VLF_percen: VLF 占总 PSD 比例 PSD_VLF/总 PSD 5. RMSSD: RMSSD 相邻 RR 间期之差的均方根值，单位：ms 6. MO: 根据对标设备计算方法所做出直方图，最高小直方的左边缘横坐标直方图范围 00-1400ms,间距 20ms, 单位：ms 7. SD1SD2: SD1SD2 SD1/SD2, SD1, 洛伦兹散点图的椭圆纵轴长度，	

		<p>SD2, 洛伦兹散点图的椭圆横轴长度</p> <p>8. PSD_HF: PSD_HF 0.15~.4 Hz 频段的功率, 单位: ms^2/Hz</p> <p>9. PSD_LF: PSD_LF 0.04~.15 Hz 频段的功率, 单位: ms^2/Hz</p> <p>10. LF_percent: LF 占总 PSD 比例 PSD_LF/总 PSD</p> <p>11.预留字段 1, 传 0 即可</p> <p>12.预留字段 2, 传 0 即可</p>	
--	--	---	--

● 调用代码实例

```
int score;  
float[] feature = new float[12];  
.....  
HealthManager.getInstance(getContext())  
                .setStressInquiryData(score, System.currentTimeMillis(),  
feature)
```

- 接口适配 SDK 版本
SP24

3.3.11 获取是否设置压力校准

● 接口原型

```
public int isSetStressInquiryData() throws ServiceStateException  
方法描述：查询卡路里数据
```

- 返回数据

结果 0 为设置成功，-1 为设置失败

- 调用代码实例

```
HealthManager.getInstance(getContext())  
    .isSetStressInquiryData();
```

- 接口适配 SDK 版本

暂不支持

3.3.12 设置体温自动测量开关

- 接口原型

public void switchOnBodyTemperatureAutoMeasure() throws ServiceStateException

方法描述：开启体温自动测量

public void switchOffBodyTemperatureAutoMeasure() throws ServiceStateException

方法描述：关闭体温自动测量

- 调用代码实例

```
// 开启实时测量开关  
HealthManager.getInstance(getContext()).switchOnBodyTemperatureAutoMeasure ();
```

```
// 关闭实时测量开关  
HealthManager.getInstance(getContext()).switchOffBodyTemperatureAutoMeasure ();
```

- 接口适配 SDK 版本

SP24

3.3.13 设置体温告警阈值

- 接口原型

public void setBodyTemperatureWarning(float high, float low) throws
ServiceStateException ,GenericException

方法描述：设置体温告警值阈值

- 注意事项

该接口会自动打开体温告警的开关，设置的时候需要同时设置上限和下限，注意参数，如果参数不正确可能会导致 `GenericException` 异常

- 请求参数

参数名称	参数类型	参数描述	可选选项
high	int	体温告警的上限(体温超过该值会触发告警)，范围(0,50]	M
low	int	体温告警的下限（体温低于该值会触发告警），范围(0,50]	M

- 调用代码实例

```
// 设置告警上限 37.5，下限 35
HealthManager.getInstance(getContext()).setBodyTemperatureWarning(37.5, 35.0);
```

- 接口适配 SDK 版本
SP24

3.3.14 取消体温告警

- 接口原型

```
public void cancelBodyTemperatureWarning() throws ServiceStateException
```

方法描述：取消体温告警

- 调用代码实例

```
HealthManager.getInstance(getContext()).cancelBodyTemperatureWarning();
```

- 接口适配 SDK 版本
SP24

3.3.15 设置心率周期（自动）测量的频率

- 接口原型

```
public void setHeartRateMeasurePeriod(long period, boolean isAutoStart)
```

方法描述：设置心率周期（自动）测量的频率

- 注意事项

该接口频率的单位为 ms，但是实际频率最大只能精确到秒级(s)，所以设置频率的时候最好也是按照 1s 的整数倍去设置。且由于环境，佩戴方式，活动状态等因素，可能造成测量失败，所以出值的频率不严格按照频率进行出值(可能出值间隔会大于频率)；

- 请求参数

参数名称	参数类型	参数描述	可选选项
period	long	心率测量的频率，单位：ms PS：该值必须大于等于 2s(2000ms)，建议5s及以上，最好也能是5s的倍数(如10s, 15s等)	M
isAutoStart	boolean	是否自动打开自动测量开关： true:如果自动测量开关关闭，自动打开心率自动测量 false:保持现有自动测量开关状态，不做任何改变	M

- 调用代码实例

```
HealthManager.getInstance(getContext()).setHeartRateMeasurePeriod(5000L, false);
```

- 接口适配 SDK 版本
SP24

3.3.16 设置血氧周期（自动）测量的频率

- 接口原型

```
public void setSpo2MeasurePeriod(long period, boolean isAutoStart) throws ServiceStateException, GenericException
```

方法描述：设置血氧周期（自动）测量的频率

- 注意事项

该接口频率的单位为 ms，但是实际频率最大只能精确到分钟级(min)，所以设置频率的时候最好也是按照 1min 的整数倍去设置。且由于环境，佩戴方式，活动状态等因素，可能造成测量失败，所以出值的频率不严格按照频率进行出值(可能出值间隔会大于频率)；

- 请求参数

参数名称	参数类型	参数描述	可选选项
period	long	血氧测量的频率，单位：ms PS：该值必须大于等于 2min(120000ms)，建议设置为1min 倍数(比如5min, 6min.....10min 等)，如无必要频率建议不要太频繁，10min比较好	M
isAutoStart	boolean	是否自动打开自动测量开关： true:如果自动测量开关关闭，自动打开心率自动测量 false:保持现有自动测量开关状态，不做任何改变	M

- 调用代码实例

```
HealthManager.getInstance(getContext()).setSpo2MeasurePeriod(300000L, false);
```

- 接口适配 SDK 版本
SP24

3.3.17 设置压力周期（自动）测量的频率

- 接口原型

```
public void setStressMeasurePeriod(long period, boolean isAutoStart) throws  
ServiceStateException, GenericException, WrongStateException
```

方法描述：设置压力周期（自动）测量的频率

- 注意事项

该接口频率的单位为 ms，但是实际频率最大只能精确到分钟级(min)，所以设置频率的时候最好也是按照 1min 的整数倍去设置。且由于环境，佩戴方式，活动状态等因素，可能造成测量失败，所以出值的频率不严格按照频率进行出值(可能出值间隔会大于频率)；

● 请求参数

参数名称	参数类型	参数描述	可选选项
period	long	压力测量的频率，单位：ms PS：该值必须大于等于 2min(120000ms)，建议设置为1min 倍数(比如5min, 6min.....10min 等)，如无必要频率建议不要太频繁，30min比较好	M
isAutoStart	boolean	是否自动打开自动测量开关： true:如果自动测量开关关闭，自动 打开心率自动测量 false:保持现有自动测量开关状态， 不做任何改变	M

● 调用代码实例

```
HealthManager.getInstance(getContext()).setStressMeasurePeriod(1800000L,  
false);
```

● 接口适配 SDK 版本
SP24

3.3.18 设置体温周期（自动）测量的频率

● 接口原型

```
public void setBodyTemperaMeasurePeriod(long period, boolean isAutoStart)  
throws ServiceStateException, GenericException
```

● 注意事项

该接口频率的单位为 ms，但是实际频率最大只能精确到秒级 (s)，且由于 sensor 器件的限制，只能按照 20s 的测量周期出值，所以如果设置的值不是 20s 倍数，也会被推迟 20s 倍数时间出值(比如设置 30s 的频率，但是实际出值可能是按照 40s 一次的数据出值)，且由于环境，佩戴方式，活动状态等因素，可能造成测量失败，这种情况也可能造成出值间隔会大于设置的频率。

● 请求参数

参数名称	参数类型	参数描述	可选选项
------	------	------	------

period	long	体温测量的频率，单位：ms PS：该值最小20s(20000ms)，建议设置为20s的倍数(比如20s, 40s,60s等)	M
isAutoStart	boolean	是否自动打开自动测量开关： true:如果自动测量开关关闭，自动打开心率自动测量 false:保持现有自动测量开关状态，不做任何改变	M

● 调用代码实例

```
HealthManager.getInstance(getContext()).setBodyTemperaMeasurePeriod(20000L, false);
```

● 接口适配 SDK 版本
SP24

3.3.19 设置心率告警触发次数

● 接口原型

```
public void setHeartWarningCnt(int count) throws ServiceStateException, GenericException
```

方法描述：设置心率告警触发次数

● 请求参数

参数名称	参数类型	参数描述	可选选项
count	int	触发告警的次数	M

● 调用代码实例

```
HealthManager.getInstance(getContext()).setHeartWarningCnt(5);
```

● 接口适配 SDK 版本
SP24

3.3.20 设置血氧告警触发次数

● 接口原型

public void setSpo2WarningCnt(int count) throws ServiceStateException, GenericException

方法描述：设置血氧告警触发次数

● 请求参数

参数名称	参数类型	参数描述	可 选选项
count	int	触发告警的次 数	M

● 调用代码实例

```
HealthManager.getInstance(getContext()).setSpo2WarningCnt(5);
```

● 接口适配 SDK 版本
SP24

3.3.21 设置体温告警触发次数

● 接口原型

public void setBodyTemperatureWarningCnt(int count) throws ServiceStateException, GenericException

方法描述：设置体温告警触发次数

● 请求参数

参数名称	参数类型	参数描述	可 选选项
count	int	触发告警的次 数	M

● 调用代码实例

```
HealthManager.getInstance(getContext()).setBodyTemperatureWarningCnt(5);
```


- 接口适配 SDK 版本
SP24

3.3.22 设置运动健康连接开关

- 接口原型

`public void switchHealthConnect(boolean switchValue) throws ServiceStateException, GenericException`

方法描述：设置运动健康连接开关

- 请求参数

参数名称	参数类型	参数描述	可 选选项
switchValue	boolean	是否允许连接 运动健康	M

- 调用代码实例

```
HealthManager.getInstance(getContext()).switchHealthConnect(false);
```

- 接口适配 SDK 版本
SP24

3.3.23 设置紧急联系人

- 接口原型

`public void addSosContactInfo(ArrayList<String> contactName, ArrayList<String> contactNum) throws ServiceStateException, GenericException`

方法描述：设置紧急联系人

- 注意事项

该接口两参数长度必须相同，且对紧急联系人总数有个数限制，该参数由版本配置决定。

3.3.24 设置实时传感器实时测量开关

● 接口原型

```
public void switchOnMotionSensorMeasure(int freqSwitch) throws  
ServiceStateException
```

方法描述：开启实时传感器连续测量

```
public void switchOffMotionSensorMeasure() throws ServiceStateException
```

方法描述：关闭实时传感器连续测量

● 请求参数

参数名称	参数类型	参数描述	可选选项
freqSwitch	Int	实时传感器测量的频率类型(数值只能为1或者2) 1：日常模式，不同场景下传感器采样率不同 2：测试模式，ppg、acc、gyro采样率均为100HZ	M

● 调用代码实例

```
// 开启实时测量开关  
HealthManager.getInstance(getContext()).switchOnMotionSensorMeasure(2);  
  
// 关闭实时测量开关  
HealthManager.getInstance(getContext()).switchOffMotionSensorMeasure();
```

● 接口适配 SDK 版本

暂不支持

● 请求参数

参数名称	参数类型	参数描述	可选选项
contactName	ArrayList<String>	紧急联系人姓名	M
contactNum	ArrayList<String>	紧急联系人电话号码	M

● 调用代码实例

```
ArrayList<String> contactNameList = new  
ArrayList<>(Arrays.asList(mToastDialog.mHighTextField.getText().split(",")));  
  
ArrayList<String> contactNumberList = new  
ArrayList<>(Arrays.asList(mToastDialog.mLowTextField.getText().split(",")));  
  
HealthManager.getInstance(getContext()).addSosContactInfo(contactNameList,  
contactNumberList);
```

接口适配 SDK 版本
SP24

3.4 系统事件监听

3.4.1 心率异常事件监听

● 适用范围:

事件监听只适用于应用处于运行中

● 监听方法

订阅鸿蒙公共事件

● 请求参数

参数名称	参数类型	参数描述	可选选项
com.tdtech.ohos.health.action.HEARTRATE_HIGH_ALERT	String	自定义事件	M
com.tdtech.ohos.health.action.HEARTRATE_LOW_ALERT	String	自定义事件	M

释意：调用 MatchingSkills 对象的 addEvent 将参数添加进去

● 返回数据

参数名称	参数类型	参数描述	可选选项
heartRate	String	心率异常数据值	——

释意：在 onReceiveEvent 回调方法中，通过

commonEventData.getIntent().getStringExtra("heartRate ")获取心率异常数据值

- 调用代码实例

心率过高异常监听：

```
MatchingSkills matchingSkills = new MatchingSkills();
matchingSkills.addEvent("com.tdtech.ohos.health.action.HEARTRATE_HIGH_ALERT");
CommonEventSubscribeInfo commonEventSubscribeInfo = new
CommonEventSubscribeInfo(matchingSkills);
try {
    CommonEventManager.subscribeCommonEvent(new
    CommonEventSubscriber(commonEventSubscribeInfo) {
        @Override
        public void onReceiveEvent(CommonEventData commonEventData) {
            String heartRate =
                commonEventData.getIntent().getStringExtra("heartRate");
        }
    });
} catch (RemoteException e) {
    e.printStackTrace();
}
```

心率过低异常监听：

```
MatchingSkills matchingSkills = new MatchingSkills();
matchingSkills.addEvent("com.tdtech.ohos.health.action.HEARTRATE_LOW_ALERT");
CommonEventSubscribeInfo commonEventSubscribeInfo = new
CommonEventSubscribeInfo(matchingSkills);
try {
    CommonEventManager.subscribeCommonEvent(new
    CommonEventSubscriber(commonEventSubscribeInfo) {
        @Override
        public void onReceiveEvent(CommonEventData commonEventData) {
            String heartRate =
                commonEventData.getIntent().getStringExtra("heartRate");
        }
    });
} catch (RemoteException e) {
    e.printStackTrace();
}
```

- 接口适配 SDK 版本
SP24

3.4.2 血氧异常事件监听

- 适用范围:

事件监听只适用于应用处于运行中

- 监听方法

订阅鸿蒙公共事件

- 请求参数

参数名称	参数类型	参数描述	可选选项
com.tdtech.ohos.health.action.SPO2_LOW_ALERT	String	自定义事件	M

释意：调用 MatchingSkills 对象的 addEvent 将参数添加进去

- 返回数据

参数名称	参数类型	参数描述	可选选项
spo2	String	血氧数据异常值	——

释意：在 onReceiveEvent 回调方法中，通过
commonEventData.getIntent().getStringParam("spo2 ")获取血氧异常数据值

- 调用代码实例

```
MatchingSkills matchingSkills = new MatchingSkills();
matchingSkills.addEvent("com.tdtech.ohos.health.action.SPO2_LOW_ALERT");
CommonEventSubscribeInfo commonEventSubscribeInfo = new
CommonEventSubscribeInfo(matchingSkills);
try {
    CommonEventManager.subscribeCommonEvent(new
    CommonEventSubscriber(commonEventSubscribeInfo) {
        @Override
        public void onReceiveEvent(CommonEventData commonEventData) {
            String spo2= commonEventData.getIntent().getStringParam("spo2");
        }
    });
} catch (RemoteException e) {
    e.printStackTrace();
}
```

- 接口适配 SDK 版本
SP24

3.4.3 压力异常事件监听

- 适用范围:

事件监听只适用于应用处于运行中

- 监听方法

订阅鸿蒙公共事件

- 请求参数

参数名称	参数类型	参数描述	可选选项
com.tdtech.ohos.health.action.STRESS_HIGH_ALERT	String	自定义事件	M

释意：调用 MatchingSkills 对象的 addEvent 将参数添加进去

- 返回数据

参数名称	参数类型	参数描述	可选选项
stress	String	压力异常数据	——

释意：在 onReceiveEvent 回调方法中，通过

commonEventData.getIntent().getStringParam("stress")获取压力异常数据值

- 调用代码实例

```
MatchingSkills matchingSkills = new MatchingSkills();
matchingSkills.addEvent("com.tdtech.ohos.health.action.STRESS_HIGH_ALERT");
CommonEventSubscribeInfo commonEventSubscribeInfo = new
CommonEventSubscribeInfo(matchingSkills);
try {
    CommonEventManager.subscribeCommonEvent(new
    CommonEventSubscriber(commonEventSubscribeInfo) {
        @Override
        public void onReceiveEvent(CommonEventData commonEventData) {
            String stressParam =
                commonEventData.getIntent().getStringParam("stress");
        }
    });
} catch (RemoteException e) {
    e.printStackTrace();
}
```

- 接口适配 SDK 版本

SP24

注意：SP8版本上该接口存在问题，设置压力阈值后收不到压力告警，在后续版本上解决。

3.4.4 跌倒事件监听

- 适用范围：

事件监听只适用于应用处于运行中

- 监听方法

订阅鸿蒙公共事件

- 前置条件

触发跌倒事件需要先开启跌倒监测开关

方法 1：手动开启设置-安全和隐私-SOS 紧急求助-跌倒监测开关

方法 2：参考 cust 配置文档配置，由配置人员配置生效后，开机自动开启跌倒监测

- 请求参数

参数名称	参数类型	参数描述	可选选项
com.tdtech.ohos.health.action.FALLDOWN_EVENT	String	自定义事件	M

释意：调用 MatchingSkills 对象的 addEvent 将参数添加进去

- 返回数据

参数名称	参数类型	参数描述	可选选项
alarmstate	String	接收到的参数，代表跌倒发生后在弹出的界面，点击不同的按钮。 “Emergency alarm”：表示点击了“拨打紧急电话”按钮； “Invalid alarm”：表示点击了“我跌倒了，我没事”按钮； “Cancel alarm”：表示点击了“我没跌倒”按钮；	——

释意：不点击跌倒后弹出的界面里的按钮，是没有返回值的，点击按钮后则可以

在 onCommand()方法中，通过 intent.getStringParam("alarmstate ")获取点击的按钮类型

● 调用代码实例

```
MatchingSkills matchingSkills = new MatchingSkills();
matchingSkills.addEvent("com.tdtech.ohos.health.action.FALLDOWN_EVENT");
CommonEventSubscribeInfo commonEventSubscribeInfo = new
CommonEventSubscribeInfo(matchingSkills);
try {
    CommonEventManager.subscribeCommonEvent(new
    CommonEventSubscriber(commonEventSubscribeInfo) {
        @Override
        public void onReceiveEvent(CommonEventData commonEventData) {
String alarmstate = commonEventData.getIntent().getStringParam("alarmstate");
        }
    });
} catch (RemoteException e) {
    e.printStackTrace();
}
```

● 接口适配 SDK 版本
SP24

3.4.5 功能键双击事件监听

● 适用范围:

事件监听只适用于应用处于运行中

● 监听方法

订阅鸿蒙公共事件

● 请求参数

参数名称	参数类型	参数描述	可选选项
com.tdtech.ohos.action.FUN_DOUBLE_CLICK	String	自定义事件	M

释意：调用 MatchingSkills 对象的 addEvent 将参数添加进去

● 返回数据

无，双击功能键时会回调 onReceiveEvent()方法

● 调用代码实例

```
MatchingSkills matchingSkills = new MatchingSkills();
matchingSkills.addEvent("com.tdtech.ohos.action.FUN_DOUBLE_CLICK");
CommonEventSubscribeInfo commonEventSubscribeInfo = new
CommonEventSubscribeInfo(matchingSkills);
```



```
try {
    CommonEventManager.subscribeCommonEvent(new
        CommonEventSubscriber(commonEventSubscribeInfo) {
            @Override
            public void onReceiveEvent(CommonEventData commonEventData) {

            }
        });
} catch (RemoteException e) {
    e.printStackTrace();
}
```

- 接口适配 SDK 版本
SP24

3.4.6 设置定位服务开关修改事件监听

- 适用范围：
事件监听只适用于应用处于运行中
- 监听方法
订阅鸿蒙公共事件

- 请求参数

参数名称	参数类型	参数描述	可选选项
com.tdtech.ohos.action.UI_SETTINGS_CHANGED	String	自定义事件	M

释意：调用 MatchingSkills 对象的 addEvent 将参数添加进去

- 返回数据

参数名称	参数类型	参数描述	可选选项
location_service_switch	String	设置定位服务开关 值，开：on 关：off	——

释意：在 onReceiveEvent 回调方法中，通过
commonEventData.getIntent().getStringParam("location_service_switch ")获取设置中定位服务修改时的开关值，开关为开返回 “on” ， 开关为关返回 “off” 。

- 调用代码实例

```
MatchingSkills matchingSkills = new MatchingSkills();
matchingSkills.addEvent("com.tdtech.ohos.action.UI_SETTINGS_CHANGED");
CommonEventSubscribeInfo commonEventSubscribeInfo = new
```

```
CommonEventSubscribeInfo(matchingSkills);
try {
    CommonEventManager.subscribeCommonEvent(new
        CommonEventSubscriber(commonEventSubscribeInfo) {
            @Override
            public void onReceiveEvent(CommonEventData commonEventData) {
                // 接收设置中定位服务开关修改后的值，on代表开，off代表关
                String location_service_switch =
                    commonEventData.getIntent().getStringExtra("location_service_
                        switch");
            }
        });
} catch (RemoteException e) {
    e.printStackTrace();
}
```

- 接口适配 SDK 版本
SP24

3.4.7 体温异常事件监听

- 适用范围：
事件监听只适用于应用处于运行中
- 监听方法
订阅鸿蒙公共事件

- 请求参数

参数名称	参数类型	参数描述	可选选项
com.tdtech.ohos.health.action.TEMPERATURE_HIGH_ALERT	String	自定义事件	M
com.tdtech.ohos.health.action.TEMPERATURE_LOW_ALERT	String	自定义事件	M

释意：调用 MatchingSkills 对象的 addEvent 将参数添加进去

- 返回数据

参数名称	参数类型	参数描述	可选选项
temperature	String	体温异常数据值	——

释意：在 onReceiveEvent 回调方法中，通过
commonEventData.getIntent().getStringExtra("temperature")获取体温异常数据值

- 调用代码实例

体温过高异常监听:

```
MatchingSkills matchingSkills = new MatchingSkills();
matchingSkills.addEvent("com.tdtech.ohos.health.action.TEMPERATURE_HIGH_ALERT");
CommonEventSubscribeInfo commonEventSubscribeInfo = new
CommonEventSubscribeInfo(matchingSkills);
try {
    CommonEventManager.subscribeCommonEvent(new
        CommonEventSubscriber(commonEventSubscribeInfo) {
            @Override
            public void onReceiveEvent(CommonEventData commonEventData) {
                String temperature =
                    commonEventData.getIntent().getStringParam("temperature");
            }
        });
} catch (RemoteException e) {
    e.printStackTrace();
}
```

体温过低异常监听:

```
MatchingSkills matchingSkills = new MatchingSkills();
matchingSkills.addEvent("com.tdtech.ohos.health.action.TEMPERATURE_LOW_ALERT");
CommonEventSubscribeInfo commonEventSubscribeInfo = new
CommonEventSubscribeInfo(matchingSkills);
try {
    CommonEventManager.subscribeCommonEvent(new
        CommonEventSubscriber(commonEventSubscribeInfo) {
            @Override
            public void onReceiveEvent(CommonEventData commonEventData) {
                String temperature =
                    commonEventData.getIntent().getStringParam("temperature");
            }
        });
} catch (RemoteException e) {
    e.printStackTrace();
}
```

- 接口适配 SDK 版本
SP24

3.4.8 一键告警事件监听

- 适用范围:

事件监听只适用于应用处于运行中

- 前提条件

目前支持 2 中触发方式：

长按功能键：从 sp9 版本开始支持，默认为该触发方式，该方式下长按 3.5s 触发，时间可以通过 cust 进行配置

多次点击功能键：从 sp13 版本开始支持，该方式默认不生效，需要进行 cust 配置才能生效，点击次数（至少 2 次）默认为 5 次，也可以通过 cust 进行配置。

提到的关于触发方式，时长，次数等相关 cust 的配置，如果需要定制，可以咨询鼎桥侧相关的服务或者项目接口人，由他们进行协助进行相关定制配置工作。

- 监听方法

订阅鸿蒙公共事件

- 请求参数

参数名称	参数类型	参数描述	可选选项
com.tdtech.ohos.action.ONE_KEY_ALARM	String	自定义事件	M

释意：调用 MatchingSkills 对象的 addEvent 将参数添加进去

- 返回数据

无，长按功能键时会回调 onReceiveEvent()方法

- 调用代码实例

```
MatchingSkills matchingSkills = new MatchingSkills();
matchingSkills.addEvent("com.tdtech.ohos.action.ONE_KEY_ALARM");
CommonEventSubscribeInfo commonEventSubscribeInfo = new
CommonEventSubscribeInfo(matchingSkills);
try {
    CommonEventManager.subscribeCommonEvent(new
        CommonEventSubscriber(commonEventSubscribeInfo) {
            @Override
            public void onReceiveEvent(CommonEventData commonEventData) {

            }
        });
} catch (RemoteException e) {
    e.printStackTrace();
}
```

- 接口适配 SDK 版本
SP24

3.4.9 佩戴状态事件监听

- 适用范围：
事件监听只适用于应用处于运行中
- 监听方法
订阅鸿蒙公共事件
- 请求参数

参数名称	参数类型	参数描述	可选选项
com.tdtech.ohos.action.WEAR_STATUS_CHANGED	String	自定义事件	M

释意：调用 MatchingSkills 对象的 addEvent 将参数添加进去

- 返回数据

参数名称	参数类型	参数描述	可选选项
wear_status	int	佩戴状态： 1：佩戴上 0：离腕，未佩戴	——

释意：在 onReceiveEvent 回调方法中，通过
commonEventData.getIntent().getIntParam("wear_status ")获取佩戴状态

- 调用代码实例

```
MatchingSkills matchingSkills = new MatchingSkills();
matchingSkills.addEvent("com.tdtech.ohos.action.WEAR_STATUS_CHANGED ");
CommonEventSubscribeInfo commonEventSubscribeInfo = new
CommonEventSubscribeInfo(matchingSkills);
try {
    CommonEventManager.subscribeCommonEvent(new
    CommonEventSubscriber(commonEventSubscribeInfo) {
        @Override
        public void onReceiveEvent(CommonEventData commonEventData) {
            int status = commonEventData.getIntent().getIntParam("wear_status
            ");
        }
    });
} catch (RemoteException e) {
```

```
        e.printStackTrace();  
    }
```

- 接口适配 SDK 版本
SP24

3.4.10 来电事件监听

- 适用范围:

事件监听只适用于应用处于运行中

- 监听方法

订阅鸿蒙公共事件

- 请求参数

参数名称	参数类型	参数描述	可选选项
com.tdtech.ohos.action.CALL_STATE	String	自定义事件	M

释意：调用 MatchingSkills 对象的 addEvent 将参数添加进去

- 返回数据

参数名称	参数类型	参数描述	可选选项
incoming_callId incoming_phonenumber	String String	来电电话callId 来电号码	——

释意：在 onReceiveEvent 回调方法中，通过

commonEventData.getIntent().getStringParam("incoming_callId")获取来电电话 callId，通过
commonEventData.getIntent().getStringParam("incoming_phonenumber")获取来电号码。

- 调用代码实例

```
MatchingSkills matchingSkills = new MatchingSkills();  
matchingSkills.addEvent("com.tdtech.ohos.action.CALL_STATE");  
CommonEventSubscribeInfo commonEventSubscribeInfo = new  
CommonEventSubscribeInfo(matchingSkills);  
try {  
    CommonEventManager.subscribeCommonEvent(new  
        CommonEventSubscriber(commonEventSubscribeInfo) {  
            @Override  
            public void onReceiveEvent(CommonEventData commonEventData) {  
                String callId= commonEventData.getIntent().getStringParam  
                    ("incoming_callId");  
                String phonenumber= commonEventData.getIntent().getStringParam  
                    ("incoming_phonenumber");  
            }  
        })  
}
```

```
    }  
    });  
} catch (RemoteException e) {  
    e.printStackTrace();  
}
```

- 接口适配 SDK 版本
暂不支持

3.4.11 SOS 事件监听

- 适用范围：
事件监听只适用于应用处于运行中
- 监听方法
订阅鸿蒙公共事件
- 前置条件
定制版本需配置 SOS 告警倒计时几秒后发送 SOS 告警事件才会发送 SOS 事件
- 请求参数

参数名称	参数类型	参数描述	可选选项
com.tdtech.ohos.health.action.SOS_EVENT	String	自定义事件	M

释意：调用 MatchingSkills 对象的 addEvent 将参数添加进去

- 调用代码实例

```
MatchingSkills matchingSkills = new MatchingSkills();  
matchingSkills.addEvent("com.tdtech.ohos.health.action.SOS_EVENT");  
CommonEventSubscribeInfo commonEventSubscribeInfo = new  
CommonEventSubscribeInfo(matchingSkills);  
try {  
    CommonEventManager.subscribeCommonEvent(new  
CommonEventSubscriber(commonEventSubscribeInfo) {  
        @Override  
        public void onReceiveEvent(CommonEventData commonEventData) {  
        }  
    });  
} catch (RemoteException e) {  
    e.printStackTrace();  
}
```
- 接口适配 SDK 版本

3.5 系统事件启动服务

3.5.1 心率异常事件启动服务

- 适用范围:

事件启动服务适用于应用处于运行中、应用未运行。

- 前置条件

提供需要启动的服务应用的包名以及服务类名，参考 cust 配置文档配置，由配置人员配置生效后，事件发生后会启动该服务

- 参数

参数名称	参数类型	参数描述	可选选项
com.tdtech.ohos.health.action.HEARTRATE_HIGH_ALERT	String	自定义事件	M
com.tdtech.ohos.health.action.HEARTRATE_LOW_ALERT	String	自定义事件	M

释意：在 config.json 文件中注册 service 并配置 service 的 action，在 onCommand()方法中，通过 intent.getAction()判断启动 service 的具体事件

- 返回数据

参数名称	参数类型	参数描述	可选选项
heartRate	String	心率异常数据值	——

释意：在 onCommand()方法中，通过 intent. getStringParam("heartRate ")获取心率异常数据值

- 调用代码实例

心率过高异常：

在 config.json 文件中配置服务以及 action

```
{
  "name": "com.tdtech.testsdk.MyService",
  "type": "service",
  "visible": true,
```



```
    "skills": [  
      {  
        "actions": [  
          "com.tdtech.ohos.health.action.HEARTRATE_HIGH_ALERT"  
        ]  
      }  
    ]  
  }  
}
```

```
public class MyService extends Ability {  
  
    @Override  
    public void onCommand(Intent intent, boolean restart, int startId) {  
        super.onCommand(intent, restart, startId);  
        if(intent.getAction().  
equals("com.tdtech.ohos.health.action.HEARTRATE_HIGH_ALERT")) {  
            String heartRate = intent.getStringParam("heartRate ");  
        }  
    }  
}
```

心率过低异常:

在 config.json 文件中配置服务以及 action

```
{  
  "name": "com.tdtech.testsdk.MyService",  
  "type": "service",  
  "visible": true,  
  "skills": [  
    {  
      "actions": [  
        "com.tdtech.ohos.health.action.HEARTRATE_LOW_ALERT"  
      ]  
    }  
  ]  
}
```

```
public class MyService extends Ability {  
  
    @Override  
    public void onCommand(Intent intent, boolean restart, int startId) {  
        super.onCommand(intent, restart, startId);  
        if(intent.getAction().  
equals("com.tdtech.ohos.health.action.HEARTRATE_LOW_ALERT")) {  
            String heartRate = intent.getStringParam("heartRate ");  
        }  
    }  
}
```

- 接口适配 SDK 版本
SP24

3.5.2 血氧异常事件启动服务

- 适用范围:

事件启动服务适用于应用处于运行中、应用未运行。

- 前置条件

提供需要启动的服务应用的包名以及服务类名，参考 cust 配置文档配置，由配置人员配置生效后，事件发生后会启动该服务

- 参数

参数名称	参数类型	参数描述	可选选项
com.tdtech.ohos.health.action.SPO2_LOW_ALERT	String	自定义事件	M

释意：在 config.json 文件中注册 service 并配置 service 的 action，在 onCommand()方法中，通过 intent.getAction()判断启动 service 的具体事件

- 返回数据

参数名称	参数类型	参数描述	可选选项
spo2	String	血氧数据异常值	——

释意：在 onCommand()方法中，通过 intent.getStringParam("spo2 ")获取血氧异常数据值

- 调用代码实例

在 config.json 文件中配置服务以及 action

```
{
  "name": "com.tdtech.testsdk.MyService",
  "type": "service",
  "visible": true,
  "skills": [
    {
      "actions": [
        "com.tdtech.ohos.health.action.SPO2_LOW_ALERT"
      ]
    }
  ]
}
```

```
public class MyService extends Ability {  
  
    @Override  
    public void onCommand(Intent intent, boolean restart, int startId) {  
        super.onCommand(intent, restart, startId);  
  
        if(intent.getAction().equals("com.tdtech.ohos.health.action.SPO2_LOW_ALERT")) {  
            String spo2= intent.getStringParam("spo2");  
        }  
    }  
}
```

- 接口适配 SDK 版本
SP24

3.5.3 压力异常事件启动服务

- 适用范围：
事件启动服务适用于应用处于运行中、应用未运行。
- 前置条件
提供需要启动的服务应用的包名以及服务类名，参考 cust 配置文档配置，由配置人员配置生效后，事件发生后会启动该服务
- 参数

参数名称	参数类型	参数描述	可选选项
com.tdtech.ohos.health.action.STRESS_HIGH_ALERT	String	自定义事件	M

释意：在 config.json 文件中注册 service 并配置 service 的 action，在 onCommand()方法中，通过 intent.getAction()判断启动 service 的具体事件

- 返回数据

参数名称	参数类型	参数描述	可选选项
stress	String	压力异常数据	——

释意：在 onCommand()方法中，通过 intent.getStringParam("stress")获取压力异常数据值

- 调用代码实例
在 config.json 文件中配置服务以及 action
{
 "name": "com.tdtech.testsdk.MyService",

```
    "type": "service",
    "visible": true,
    "skills": [
      {
        "actions": [
          "com.tdtech.ohos.health.action.STRESS_HIGH_ALERT"
        ]
      }
    ]
  }
}
```

```
public class MyService extends Ability {

    @Override
    public void onCommand(Intent intent, boolean restart, int startId) {
        super.onCommand(intent, restart, startId);

        if(intent.getAction().equals("com.tdtech.ohos.health.action.STRESS_HIGH_ALERT")) {
            String stress = intent.getStringParam("stress");
        }
    }
}
```

- 接口适配 SDK 版本
SP24

3.5.4 跌倒事件启动服务

- 适用范围：

事件启动服务适用于应用处于运行中、应用未运行。

- 前置条件

提供需要启动的服务应用的包名以及服务类名，参考 cust 配置文档配置，由配置人员配置生效后，事件发生后会启动该服务

触发跌倒事件需要先开启跌倒监测开关

方法 1：手动开启设置-安全和隐私-SOS 紧急求助-跌倒监测开关

方法 2：参考 cust 配置文档配置，由配置人员配置生效后，开机自动开启跌倒监测

- 参数

参数名称	参数类型	参数描述	可选选项
com.tdtech.ohos.health.action.FALLDOWN_EVENT	String	自定义事件	M

释意：在 config.json 文件中注册 service 并配置 service 的 action，在 onCommand()方法中，通过 intent.getAction()判断启动 service 的具体事件

● 返回数据

参数名称	参数类型	参数描述	可选选项
alarmstate	String	接收到的参数，代表跌倒发生后在弹出的界面，点击不同的按钮。 “Emergency alarm”：表示点击了“拨打紧急电话”按钮； “Invalid alarm”：表示点击了“我跌倒了，我没事”按钮； “Cancel alarm”：表示点击了“我没跌倒”按钮；	——

释意：不点击跌倒后弹出的界面的按钮，是没有返回值的，点击按钮后则可以

在 onCommand()方法中，通过 intent.getStringParam("alarmstate ")获取点击的按钮类型

● 调用代码实例

在 config.json 文件中配置服务以及 action

```
{
  "name": "com.tdtech.testsdk.MyService",
  "type": "service",
  "visible": true,
  "skills": [
    {
      "actions": [
        "com.tdtech.ohos.health.action.FALLDOWN_EVENT"
      ]
    }
  ]
}

public class MyService extends Ability {
  @Override
  public void onCommand(Intent intent, boolean restart, int startId) {

    super.onCommand(intent, restart, startId);
    if(intent.getAction().equals("com.tdtech.ohos.health.action.FALLDOWN_EVENT")) {
      String alarmstate = intent.getStringParam("alarmstate");
```

```
    }  
  }  
}
```

- 接口适配 SDK 版本
SP24

3.5.5 功能键双击事件启动服务

- 适用范围：

事件启动服务适用于应用处于运行中、应用未运行。

- 前置条件

提供需要启动的服务应用的包名以及服务类名，参考 cust 配置文档配置，由配置人员配置生效后，事件发生后会启动该服务

- 参数

参数名称	参数类型	参数描述	可选选项
com.tdtech.ohos.action.FUN_DOUBLE_CLICK	String	自定义事件	M

释意：在 config.json 文件中注册 service 并配置 service 的 action，在 onCommand()方法中，通过 intent.getAction()判断启动 service 的具体事件

- 调用代码实例

在 config.json 文件中配置服务以及 action

```
{  
  "name": "com.tdtech.testsdk.MyService",  
  "type": "service",  
  "visible": true,  
  "skills": [  
    {  
      "actions": [  
        "com.tdtech.ohos.action.FUN_DOUBLE_CLICK"  
      ]  
    }  
  ]  
}  
  
public class MyService extends Ability {  
  @Override  
  public void onCommand(Intent intent, boolean restart, int startId) {  
    super.onCommand(intent, restart, startId);  
    if(intent.getAction().equals("com.tdtech.ohos.action.FUN_DOUBLE_CLICK")) {  
    }  
  }  
}
```

```
}  
}
```

- 接口适配 SDK 版本
SP24

3.5.6 设置定位服务开关修改事件启动服务

- 适用范围：

事件启动服务适用于应用处于运行中、应用未运行。

- 前置条件

提供需要启动的服务应用的包名以及服务类名，参考 cust 配置文档配置，由配置人员配置生效后，事件发生后会启动该服务

- 参数

参数名称	参数类型	参数描述	可选选项
com.tdtech.ohos.action.UI_SETTINGS_CHANGED	String	自定义事件	M

释意：在 config.json 文件中注册 service 并配置 service 的 action，在 onCommand()方法中，通过 intent.getAction()判断启动 service 的具体事件

- 返回数据

参数名称	参数类型	参数描述	可选选项
location_service_switch	String	设置定位服务开关值，开：on 关：off	——

释意：在 onCommand()方法中，通过 intent.getStringParam("location_service_switch")获取设置中定位服务修改时的开关值，开关为开返回 “on” ，开关为关返回 “off” 。

- 调用代码实例

在 config.json 文件中配置服务以及 action

```
{  
  "name": "com.tdtech.testsdk.MyService",  
  "type": "service",  
  "visible": true,  
  "skills": [  
    {  
      "actions": [  
        "com.tdtech.ohos.action.UI_SETTINGS_CHANGED"  
      ]  
    }  
  ]  
}
```

```
    }  
  ]  
}  
  
public class MyService extends Ability {  
  
    @Override  
    public void onCommand(Intent intent, boolean restart, int startId) {  
        super.onCommand(intent, restart, startId);  
  
        if(intent.getAction().equals("com.tdtech.ohos.action.UI_SETTINGS_CHANGED")) {  
            String isSwitch = intent.getStringParam("location_service_switch");  
        }  
    }  
}
```

- 接口适配 SDK 版本
SP24

3.5.7 体温异常事件启动服务

- 适用范围：
事件启动服务适用于应用处于运行中、应用未运行。
- 前置条件
提供需要启动的服务应用的包名以及服务类名，参考 cust 配置文档配置，由配置人员配置生效后，事件发生后会启动该服务
- 参数

参数名称	参数类型	参数描述	可选选项
com.tdtech.ohos.health.action.TEMPERATURE_HIGH_ALERT	String	自定义事件	M
com.tdtech.ohos.health.action.TEMPERATURE_LOW_ALERT	String	自定义事件	M

释意：在 config.json 文件中注册 service 并配置 service 的 action，在 onCommand()方法中，通过 intent.getAction()判断启动 service 的具体事件

- 返回数据

参数名称	参数类型	参数描述	可选选项
temperature	String	体温异常数据值	——

释意：在 onCommand()方法中，通过 intent.getStringParam("temperature")获取体温异常数据值

- 调用代码实例

体温过高异常：

在 config.json 文件中配置服务以及 action

```
{
  "name": "com.tdtech.testsdk.MyService",
  "type": "service",
  "visible": true,
  "skills": [
    {
      "actions": [
        "com.tdtech.ohos.health.action.TEMPERATURE_HIGH_ALERT "
      ]
    }
  ]
}
```

```
public class MyService extends Ability {

    @Override
    public void onCommand(Intent intent, boolean restart, int startId) {
        super.onCommand(intent, restart, startId);
        if(intent.getAction().
equals(" com.tdtech.ohos.health.action.TEMPERATURE_HIGH_ALERT ")) {
            String temperature = intent.getStringParam("temperature");
        }
    }
}
```

体温过低异常：

在 config.json 文件中配置服务以及 action

```
{
  "name": "com.tdtech.testsdk.MyService",
  "type": "service",
  "visible": true,
  "skills": [
    {
      "actions": [
        "com.tdtech.ohos.health.action.TEMPERATURE_LOW_ALERT "
      ]
    }
  ]
}
```

```
public class MyService extends Ability {
```

```
@Override
public void onCommand(Intent intent, boolean restart, int startId) {
    super.onCommand(intent, restart, startId);
    if(intent.getAction().
equals("com.tdtech.ohos.health.action.TEMPERATURE_LOW_ALERT")) {
        String temperature = intent.getStringParam("temperature");
    }
}
```

- 接口适配 SDK 版本
SP24

3.5.8 一键告警事件启动服务

- 适用范围:

事件启动服务适用于应用处于运行中、应用未运行。

- 触发条件

目前支持以下 2 种方式触发一键告警事件

长按功能键：默认为该方式触发，从 SP9 版本开始支持，该方式下默认长按 3.5s 触发，时间可以通过 cust 进行配置

多次点击功能键：该触发方式从 SP13 版本开始支持，默认不生效，需要进行 cust 配置才能生效，点击的次数(至少 2 次)默认为 5 次，也可以通过 cust 项进行配置

上面提到的关于触发方式，时长，次数等相关 cust 配置项，如果需要定制的，可以咨项目鼎桥侧相关的服务或者接口人，由他们提供协助进行相关定制的配置。

- 参数

参数名称	参数类型	参数描述	可选选项
com.tdtech.ohos.action.ONE_KEY_ALARM	String	自定义事件	M

释意：在 config.json 文件中注册 service 并配置 service 的 action，在 onCommand()方法中，通过 intent.getAction()判断启动 service 的具体事件

- 调用代码实例

在 config.json 文件中配置服务以及 action

```
{
  "name": "com.tdtech.testsdk.MyService",
  "type": "service",
  "visible": true,
  "skills": [
    {
      "actions": [
        " com.tdtech.ohos.action.ONE_KEY_ALARM "
      ]
    }
  ]
}

public class MyService extends Ability {
  @Override
  public void onCommand(Intent intent, boolean restart, int startId) {
    super.onCommand(intent, restart, startId);
    if(intent.getAction().equals("com.tdtech.ohos.action.ONE_KEY_ALARM ")) {
    }
  }
}
```

- 接口适配 SDK 版本
SP24

3.5.9 开机启动事件启动服务

- 适用范围：
事件启动服务适用于应用处于运行中、应用未运行。
- 前置条件
提供需要启动的服务应用的包名以及服务类名，参考 cust 配置文档配置，由配置人员配置生效后，事件发生后会启动该服务
- 参数

参数名称	参数类型	参数描述	可选选项
com.tdtech.ohos.action.BOOT_COMPLETED	String	自定义事件	M

释意：在 config.json 文件中注册 service 并配置 service 的 action，在 onCommand()方法中，通过 intent.getAction()判断启动 service 的具体事件

- 调用代码实例
在 config.json 文件中配置服务以及 action

```
{
  "name": "com.tdtech.testsdk.MyService",
  "type": "service",
  "visible": true,
  "skills": [
    {
      "actions": [
        "com.tdtech.ohos.action.BOOT_COMPLETED"
      ]
    }
  ]
}

public class MyService extends Ability {
  @Override
  public void onCommand(Intent intent, boolean restart, int startId) {
    super.onCommand(intent, restart, startId);
    if(intent.getAction().equals("com.tdtech.ohos.action.BOOT_COMPLETED")) {
    }
  }
}
```

- 接口适配 SDK 版本
SP24

3.5.10 佩戴状态改变事件启动服务

- 适用范围：
事件启动服务适用于应用处于运行中、应用未运行。
- 前置条件
提供需要启动的服务应用的包名以及服务类名，参考 cust 配置文档配置，由配置人员配置生效后，事件发生后会启动该服务
- 参数

参数名称	参 数 类 型	参数描述	可 选 选项
com.tdtech.ohos.action.WEAR_STATUS_CHANGED	String	自定义事件	M

释意：在 config.json 文件中注册 service 并配置 service 的 action，在 onCommand()方法中，通过 intent.getAction()判断启动 service 的具体事件

- 返回数据

参数名称	参数类型	参数描述	可选选项
------	------	------	------

wear_status	int	佩戴状态： 1：佩戴上 0：离腕，未佩戴	——
-------------	-----	----------------------------	----

释意：在 onCommand()方法中，通过 intent.getIntParam("wear_status")获取佩戴状态

● 调用代码实例

在 config.json 文件中配置服务以及 action

```
{
  "name": "com.tdtech.testsdk.MyService",
  "type": "service",
  "visible": true,
  "skills": [
    {
      "actions": [
        "com.tdtech.ohos.action.WEAR_STATUS_CHANGED"
      ]
    }
  ]
}
```

```
public class MyService extends Ability {

  @Override
  public void onCommand(Intent intent, boolean restart, int startId) {
    super.onCommand(intent, restart, startId);
    if(intent.getAction().
equals("com.tdtech.ohos.action.WEAR_STATUS_CHANGED ")) {
      int status = intent.getIntParam("wear_status");
    }
  }
}
```

● 接口适配 SDK 版本
SP24

3.5.11 来电事件启动服务

- 适用范围：

事件启动服务适用于应用处于运行中、应用未运行。

- 前置条件

提供需要启动的服务应用的包名以及服务类名，参考 cust 配置文档配置，由配置人员配置生效后，事件发生后会启动该服务

- 参数

参数名称	参 数 类 型	参数描述	可 选 选项
com.tdtech.ohos.action.CALL_STATE	String	自定义事件	M

释意：在 config.json 文件中注册 service 并配置 service 的 action，在 onCommand()方法中，通过 intent.getAction()判断启动 service 的具体事件

- 返回数据

参数名称	参数类型	参数描述	可选选项
incoming_callId	String	来电callId	——
incoming_phonenumber	String	来电号码	

释意：在 onCommand()方法中，通过 intent. getStringParam("incoming_callId")获取来电 callId，通过 intent. getStringParam("incoming_phonenumber")获取来电号码。

- 调用代码实例

在 config.json 文件中配置服务以及 action

```
{
  "name": "com.tdtech.testsdk.MyService",
  "type": "service",
  "visible": true,
  "skills": [
    {
      "actions": [
        "com.tdtech.ohos.action.CALL_STATE"
      ]
    }
  ]
}
```

```
public class MyService extends Ability {
```

```
@Override
public void onCommand(Intent intent, boolean restart, int startId) {
    super.onCommand(intent, restart, startId);
    if(intent.getAction().
equals("com.tdtech.ohos.action.CALL_STATE")) {
        String callId = intent. getStringParam("incoming_callId");
        String phonenumber= intent. getStringParam("incoming_phonenumber");
    }
}
```

- 接口适配 SDK 版本
暂不支持

3.5.12 SOS 事件启动服务

- 适用范围：
事件启动服务适用于应用处于运行中、应用未运行。
- 前置条件
定制版本需配置 SOS 告警倒计时几秒后发送 SOS 告警事件才会发送 SOS 事件
- 参数

参数名称	参数类型	参数描述	可选选项
com.tdtech.ohos.health.action.SOS_EVENT	String	自定义事件	M

释意：在 config.json 文件中注册 service 并配置 service 的 action，在 onCommand()方法中，通过 intent.getAction()判断启动 service 的具体事件

- 调用代码实例
在 config.json 文件中配置服务以及 action

```
{
  "name": "com.tdtech.testsdk.MyService",
  "type": "service",
  "visible": true,
  "skills": [
    {
      "actions": [
        "com.tdtech.ohos.health.action.SOS_EVENT"
      ]
    }
  ]
}
```

```
    }  
    public class MyService extends Ability {  
        @Override  
        public void onCommand(Intent intent, boolean restart, int startId) {  
            super.onCommand(intent, restart, startId);  
            if(intent.getAction().equals("com.tdtech.ohos.health.action.SOS_EVENT")) {  
                }  
            }  
        }  
    }
```

- 接口适配 SDK 版本
SP24

3.6 消息交互接口

3.6.1 注册消息接收器

- 接口原型

```
public void registerReceiver(IReceiveMessageListener listener)  
方法描述：注册消息接收监听
```

- 请求参数

参数名称	参数类型	参数描述	可 选选项
listener	IReceiveMessageListener	消息接收的 Listener	M

- 响应参数

手机端数据会通过 IBodySensorDataListener 回调回来，正常会回调 onReceiveData 方法

参数名称	参数类型	参数描述	可 选选项
------	------	------	----------

errCode	int	错误码(0: 成功, 其他失败, 参考 错误码表格)	— —
data	String	消息数据(Json 格式)	— —

data 数据格式说明:

返回具体的心率信息。为 JSON 格式数据。外层为 Json 对象, 有以下几个参数:

type: int 类型, 数据类型, 1 为文本信息

from: string, 来源包名

data: String, 传递的消息

示例:

```
{  
  "type": 1,  
  "from": "com.xxx.xxx.test",  
  "data": "test msg"  
}
```

- 调用代码实例

```
HealthManager.getInstance(getContext())  
    .registerReceiver((errCode, payload) -> {  
        if (errCode == HealthManager.ERR_SUCCESS) {  
            JsonObject object = new  
JsonParser().parse(payload).getAsJsonObject();  
            int type = object.get("type").getAsInt();  
            String from = object.get("from").getString();  
            String data = object.get("data").getString();  
        }  
    });
```

- 接口适配 SDK 版本
SP24

3.6.2 发送消息

- 接口原型

```
public void sendMessage(String dstPackageName, String payload,  
ISendMessageListener listener)
```

方法描述：注册消息接收监听

● 请求参数

参数名称	参数类型	参数描述	可 选选项
dstPackageName	String	目标包名	M
payload	String	发送的文本消 息	M
listener	ISendMessageListener	消息发送的 Listener	M

● 响应参数

手机端响应会通过 ISendMessageListener 回调回来，正常会回调 onSendResult 方法

参数名称	参数类型	参数描述	可 选选项
errCode	int	错误码(0: 成 功, 其他失败, 参 考 错误码表格)	— —

● 调用代码实例

```
HealthManager.getInstance(getContext())  
    .sendMessage("com.xxx.xxx.test", "test msg", errCode -> {  
        HiLog.info(TAG, "sendMessage result :#{public}d", errCode);  
    });
```

● 接口适配 SDK 版本
SP24

3.6.3 反注册消息接收器

● 接口原型

```
public boolean unregisterReceiver() throws ServiceStateException
```

方法描述：反注册消息接收监听

● 响应参数

参数名称	参数类型	参数描述	可 选选项
Result	boolean	反注册结果	— —

● 调用代码实例

```
try {
    boolean result =
HealthManager.getInstance(getContext()).unregisterReceiver();
} catch (ServiceStateException e) {
    e.printStackTrace();
}
```

- 接口适配 SDK 版本
SP24

3.7 附录

3.7.1 错误码对照表

错误码	描述
0	成功
1000	健康服务异常
999	同健康服务进行 IPC 通信异常
998	非法参数
997	类型错误
996	结果错误
-1	未知错误

3.7.2 功能模块名称对照表

名称	描述
heart_rate	心率
spo2	血氧
stress	压力
sleep	睡眠
workout	锻炼
temperature	温度
steps	步数
distance	距离
calorie	卡路里
daily	每日活动数据
trace_point	运动轨迹

3.7.3 历史和实时数据类型对照表

类型值	类型描述
history	历史数据
realtime	实时数据

3.7.4 锻炼记录数据类型对照表

类型值	类型描述
workoutType	0:未设置运动类型默认值 1:户外跑步;2:户外步行;3:户外骑行;4:登山;5:室内跑步;6:泳池游泳;7:室内单车;8:开放水域;9:自由训练;10:徒步;11:越野跑;12:铁人三项;13:划船机;14:椭圆机;15:室内步行;16:智能单车器材;17:铁三换项类型;18:越野滑雪;19:场地滑雪/滑雪;20:雪板滑雪;21:高尔夫练习场模式;101:瑜伽;102:健身操;103:力量训练;104:动感单车;105:踏步机;106:漫步机;107:HIIT (高强度间歇训练);108:团体操;109:普拉提;110:Cross fit(交叉训练);111:功能性训练;112:体能训练;113:跆拳道;114:拳击;115:自由搏击;116:空手道;117:击剑;118:肚皮舞;119:爵士舞;120:拉丁舞;121:芭蕾;122:核心训练;123:搏击操;124:剑道;125:单杠;126:双杠;127:街舞;128:轮滑;129:武术;130:广场舞;131:太极拳;132:其他舞蹈;133:呼啦圈;134:飞盘;135:飞镖;136:射箭;137:骑马;138:对战游戏;139:放风筝;140:拔河;141:秋千;142:爬楼;143:障碍赛;144:羽毛球;145:乒乓球;146:网球;147:台球;148:保龄球;149:排球;150:毽球;151:手球;152:棒球;153:垒球;154:板球;155:橄榄球;156:沙滩足球;157:沙滩排球;158:门球;159:曲棍球;160:壁球;161:藤球;162:躲避球;163:帆船;164:冲浪;165:钓鱼;166:漂流;167:龙舟;168:皮划艇;169:赛艇;170:摩托艇;171:桨板冲浪;172:滑冰;173:冰球;174:冰壶;175:雪车;176:雪橇;177:冬季两项;178:滑板;179:攀岩;180:蹦极;181:跑酷;182:BMX;183:定向越野;184:跳伞;185:赛车;

3.7.5 睡眠状态类型

数据类型 Type	数据定义
1	浅睡
2	快速眼动
3	深睡
4	清醒

5	零星小睡、午睡
---	---------