HarmonyOS 设备应用开发文档

V1.0

鸿蒙学堂 hmxt.org 整理

2020 年 9 月 10 日

目 录

1 车机

1.1 概述

HarmonyOS 针对汽车场景提供了驾驶安全管控和车辆控制能力集，帮助开发者构建车载控制平台上可以使用的应用。开发者通过这些能力集，可以构建出更加适合于车载控制系统上运行的应用，提高驾驶员体验，也让乘客在旅途中享受优质的乘车服务。

基本概念

驾驶模式与非驾驶模式

在汽车行业，不同地域、国家对于车载中控系统有限制，例如汽车行驶过程中不允许播放视频和消息弹框，以避免影响驾驶员安全。HarmonyOS 针对汽车定义了“驾驶模式”和“非驾驶模式”用来标识车辆状态：

- 驾驶模式：汽车行驶过程中，当车辆状态达到或者超过车厂定义的限制标准后，当前车辆的状态就定义为“驾驶模式”状态。
- 非驾驶模式：与“驾驶模式”状态相对，即车辆没有达到车厂规定的限制标准，则认为是处于“非驾驶模式”状态。

在驾驶模式状态下，HarmonyOS 系统会根据当前车辆限制标准，对系统能力做约束，例如不允许播放视频和弹框，而在非驾驶模式状态下，系统能力则不受影响。

驾驶模式支持应用

HarmonyOS 在应用增加了“驾驶模式”状态支持。对于“驾驶模式”状态支持的应用，在车辆行驶过程中可以正常运行，而对于“驾驶模式”状态不支持的应用，则在车辆行驶过程中做限制，例如禁止播放视频，禁止文本弹框等，不同的厂商限制不同，具体详情请参考车厂说明。

HarmonyOS 应用市场在应用上架时会进行审核，对于“驾驶模式”状态支持的应用，HarmonyOS 规定开发者要遵守汽车行业应用开发规范要求，具体参考驾驶安全管控章节。

约束与限制

- HarmonyOS 车载应用要求支持“驾驶模式”和“非驾驶模式”状态切换。
- 驾驶模式下，默认不允许执行影响驾驶安全的所有操作，例如播放视频，弹框等。不同车厂、地域、国家对影响驾驶安全的操作限制不同，开发者需要基于具体限制开发应用，以确保驾驶员驾驶安全，共同营造安全的驾驶体验。

1.2 驾驶安全管控

1.2.1 开发驾驶模式支持应用

场景介绍

HarmonyOS 除了限制系统能力来保证驾驶员安全，同时提供了驾驶模式相关接口，允许开发者使用第三方能力库来开发驾驶模式下可用的安全应用，本章节主要简述如何开发驾驶模式下安全应用。

接口说明

HarmonyOS 提供了驾驶模式管理类 DrivingSafetyManager，开发者可以使用该类的开放能力，开发符合驾驶模式安全要求的应用。

接口名	描述
getRestraint()	获取当前系统在“驾驶模式”状态下的约束条件。
isDrivingMode()	查询当前车辆是否处于“驾驶模式”状态。
isDrivingSafety()	判断当前的应用是否是安全的。

表 1 DrivingSafetyManager 的主要接口

开发步骤

开发一个应用具备如下能力：

- 音乐播放能力。
- 通过弹框来显示通知信息。
- 视频播放能力（三方视频播放开发库）。
- 遵守地区法规，在车辆行驶过程中不能弹框和播放视频。

1. 在开始构建应用之前，请务必遵守 HarmonyOS 的约束和限制。
2. 为应用添加驾驶模式支持。

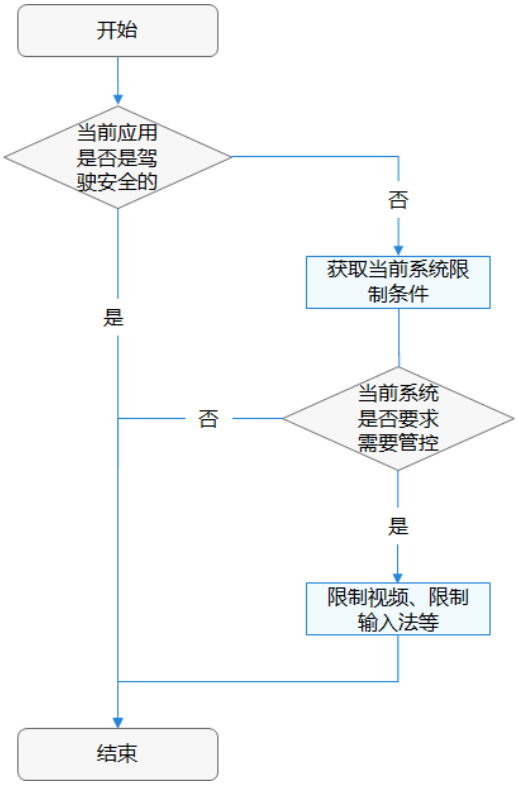
   HarmonyOS 车载应用需要开发者指定当前应用是否支持“驾驶模式”状态。对于不支持驾驶模式状态的应用，在汽车进入“驾驶模式”状态后，不允许启动，对于已经启动的应用也会冻结操作并退出。因此，开发者需要在应用配置文件（config.json）中"supported-modes"配置项中增加"drive"模式，以表

示该应用支持“驾驶模式”状态，保证车辆在行驶过程中，应用可以正常运行。如下所示：

```
"abilities": {
    "name": ".carlink",
    "icon": "$carlink:icon",
    "label": "carlink",
    "supported-modes": ["drive"],
}
```

3. 判断当前系统是否处于“驾驶模式”状态，应用后台通过调 isDrivingSafety()接口，判断当前应用是否是驾驶安全的：

- 如果是非驾驶安全的，则通过 getRestraint()接口获取当前系统的限制条件，根据系统限制条件，对当前的应用做处理，例如禁止视频播放，禁止输入法弹框；
- 如果是驾驶安全的，则无需处理。




```
if (isDrivingSafety(context)) { // 判断应用当前状态是否是驾驶安全的
    int restraintCode = DrivingSafetyManager.getRestraint(); // 获取当前系统限制条件
    if (restraintCode < 0) {
```

```
4.              HiLog.error("GetRestraint error: %d", restraintCode);
5.              return;
6.          }
7.          if (restraintCode == 0) { // 当前系统不受限
8.              HiLog.error("No restraint");
9.              return;
10.         }
11.         // 限制视频播放
12.         if (0x2 & restraintCode != 0) {
13.             Player play = new Player(content); // 第三方视频播放器
14.             play.stop();
15.         }
16.         // 限制输入法弹窗
17.         if (0x4 & restraintCode != 0) {
18.             InputMethodController mIMController = InputMethodController.getInstance(); // 第三
    方输入法
19.             mIMController.stopInput(InputMethodController.STOP_IM_NORMAL);
20.         }
21.         // 其他限制
22.         ...
23. }
```

1.2.2 定制化系统能力约束

场景介绍

HarmonyOS 提供了系统能力管控接口，允许车厂开发类似“系统设置”类应用，基于当前车型限制条件下，车厂可以提供一些系统能力，允许用户进行自定义管控策略。例如，某车型默认在“驾驶模式”状态下不允许播放视频，但可以允许消息弹出框正常弹出。用户可以根据习惯，为了驾驶安全，将消息弹出框也做限制，不允许在“驾驶模式”状态下弹出。本章节主要指导车厂如何使用定制化管控系统能力。

接口说明

HarmonyOS 提供的驾驶安全管控能力支持定制化管理，三方车厂可以通过 DrivingSafetyConfig 类的能力来开发管控类应用。

说明

1. 不同的车厂提供的能力不同，具体需要参考三方车厂能力限制说明；
2. 该开放能力只对 OEM 车厂开放，普通三方开发者不可调用。

接口名	描述
getSysDrivingSafetyConfigure()	查询指定的系统能力是否被管控。
setSysDrivingSafetyConfigure()	设定指定的系统能力是否被管控，具体需要参考三方车厂能力限制说明，不同车厂提供的限制能力不同。

表 1 DrivingSafetyConfig 的主要接口

目前，HarmonyOS 提供了两种系统能力管控的能力：

- SysDrivingSafetyControlItems.DM_IME：对系统输入法做管控
- SysDrivingSafetyControlItems.DM_Video：对系统视频播放器做管控
- SysDrivingSafetyControlItems.DM_AUTO_RUN：对自启动做管控
- SysDrivingSafetyControlItems.DM_REMOTE_CONTROL：对远程控制做管控
- SysDrivingSafetyControlItems.DM_UPGRADE：对升级做管控

开发步骤

1. 当开发者要查询当前的系统策略时，可以通过 getSysDrivingSafetyConfigure()接口获取。
2. 当开发者需要修改策略时，可以通过 setSysDrivingSafetyConfigure()接口修改当前系统能力管控策略。

```
// 构造查询结果对象
DrivingSafetyConfigResult result = new DrivingSafetyConfigResult();
// 调查询能力接口
try{
    int errorCode = DrivingSafetyConfig.getSysDrivingSafetyConfigure(SysDrivingSafetyControlItems.DM_IME, result);
    if (errorCode != 0) {
        HiLog.error("Get DrivingSafetyConfig Error: %d", errorCode));
        return;
    }
    Boolean isOpen = false;
    if (!result.isOpen()){ // 当前输入法策略为非管控状态
        isOpen = true; // 修改当前输入法策略为管控状态
    }
    // 调用修改管控能力接口，修改管控策略
    errorCode = DrivingSafetyConfig.setSysDrivingSafetyConfigure(SysDrivingSafetyControlItems.DM_IME, isOpen);
    if (errorCode != 0) {
        HiLog.error("Set DrivingSafetyConfigre Error: %d", errorCode);
        return;
    }
} catch (RemoteException | IllegalArgumentException e) {
    HiLog.error("System Error: %s", e.getMessage())
    return;
}
```

1.3 车辆控制

1.3.1 开发车辆控制应用

场景介绍

HarmonyOS 提供了车辆控制的能力接口，开发者可以基于其能力接口，开发相关的控制应用。例如，通过应用来控制车内空调温度、车窗开合程度、雨刷器、左右后视镜，查询发动机运行状况、转速等。

说明

车辆控制能力与车厂车型息息相关，HarmonyOS 提供统一的标准接口，具体能力请参考各个车辆说明。

接口说明

- 车机专有硬件服务连接类 Vehicle，支持车机专有硬件所有服务连接能力，同时携带自动重试机制。当车机专有硬件服务连接或者断开时，支持开发者实现自定义回调，具体开放能力如下：

接口名	描述
connect()	连接指定车机专有硬件服务。
disconnect()	断开指定车机专有硬件服务。
isConnected()	判断指定车机专有硬件服务是否已连接。

表 1 Vehicle 的主要接口

- 车辆座舱管理类 VehicleCabinManager，提供了车辆座舱信号访问控制方法，例如车门、空调。开发者可以通过定义的车辆信号标识来获取或者设置对应的信号值，完成对车辆座舱的控制，具体开放能力如下：

接口名	描述
getVehicleSignal()	获取座舱相关设备的信号值。
getVehicleSignalMultiAreas()	获取座舱指定信号的多区域值。
setVehicleActuator()	设置车辆座舱信号值。
subscribeVehicleSignal()	订阅指定的座舱信号。
unsubscribeVCabinSignal()	取消订阅指定的座舱信号。
unsubscribeVCabinSignalAll()	取消订阅全部座舱信号。

表 2 VehicleCabinManager 的主要接口

- 车辆车身管理类 VehicleBodyManager，提供了车辆车身设备控制相关的方法，例如雨刷器、挡风玻璃、清洁剂、车灯、引擎盖、行李箱等设备控制信息，具体开放能力如下：

接口名	描述
getVehicleSignal()	获取车身相关设备的信号值。
getVehicleSignalMultiAreas()	获取车身指定信号的多区域值。
setVehicleActuator()	设置车辆车身的信号值。
subscribeVehicleSignal()	订阅指定的车身信号。
unsubscribeVBodySignal()	取消订阅指定的车身信号。
unsubscribeVBodySignalAll()	取消订阅全部车身信号。

接口名	描述

表 3 VehicleBodyManager 的主要接口

- 车辆底盘管理类 VehicleChassisManager，提供了车辆底盘设备控制相关的方法，例如获取车辆重量、轴距、方向盘转向角度等。具体开放能力如下：

接口名	描述
getVehicleSignal()	获取车辆底盘相关设备信号值。
getVehicleSignalMultiAreas()	获取车辆底盘指定信号的多区域值。
setVehicleActuator()	设置车辆底盘相关设备的状态值。
subscribeVehicleSignal()	订阅指定的车辆底盘信号。
unsubscribeVChassisSignal()	取消订阅指定的车辆底盘信号。
unsubscribeVChassisSignalAll()	取消订阅全部的车辆底盘信号。

表 4 VehicleChassisManager 的主要接口

- 车辆引擎管理类 VehicleDriveTrainManager，提供了车辆引擎相关控制方法，例如控制变速箱模式，获取发动机转速等，具体开放能力如下：

接口名	描述
getVehicleSignal()	获取车辆引擎相关设备信号值。
getVehicleSignalMultiAreas()	获取车辆引擎指定信号的多区域值。
setVehicleActuator()	设置车辆引擎信号相关参数值。
subscribeVehicleSignal()	订阅指定的车辆引擎信号。

接口名	描述
unsubscribeVDriveTrainSignal()	取消订阅指定的车辆引擎信号。
unsubscribeVDriveTrainSignalAll()	取消订阅全部车辆引擎信号。

表 5 VehicleDriveTrainManager 的主要接口

- 通常在汽车使用过程中，驾驶员需要实时了解车辆的健康状态，从而判断车辆是那个部位出现故障，因此 HarmonyOS 提供了 OBD(on-board diagnostics)相关接口，供三方开发者开发车辆健康监测相关应用，更好服务于大众。

接口名	描述
getVehicleSignal()	获取 OBD 相关实时信号值。
getVehicleSignalMultiAreas()	获取 OBD 指定信号的多区域值。
setVehicleActuator()	设置 OBD 相关设备值。
subscribeVehicleSignal()	订阅指定的 OBD 设备信号。
unsubscribeVOBDSignal()	取消订阅指定的 OBD 设备信号。
unsubscribeVOBDSignalAll()	取消订阅全部的 OBD 设备信号。

表 6 VehicleOBDManager 的主要接口

- 车辆配置属性管理类 VehicleConfigurationManager，提供了车辆静态属性信息查询接口，例如车辆燃油类型，车辆外观尺寸等基本属性信息，具体开放能力如下：

接口名	描述
getVehicleSize()	获取车辆尺寸，包括：长、宽、高等信息。
getVehicleFuelType()	获取车辆燃油类型。

接口名	描述
getVehiclereFuelPosition()	获取燃油口位置信息。
getVehiclereTransmissionConfiguration()	获取变速器类型。
getVehicleWheelDiameter()	获取轮胎尺寸。
getVehicleSteeringWheelConfiguration()	获取车辆方向盘配置信息。
getVehicleACRISS()	获取汽车租赁公司使用的 ACRISS 汽车分类代码。
getVehicleMcuVersion()	获取车辆 MCU 版本号。
getVehicleModel()	获取车辆制造型号。
getVehicleModelYear()	获取车辆生产时间。
getVehicleBrand()	获取车辆品牌信息。
getVehicleVIN()	获取车辆识别号。
getVehicleWMI()	获取世界制造厂识别代码。
getDriverZone()	获取驾驶位信息。

表 7 VehicleConfigurationManager 的主要接口

开发步骤

1. 连接指定车机专有硬件服务。

```
1.  // 获取服务连接状态变化
2.  ServiceConnectionListener listener = new ServiceConnectionListener(){
3.      @Override
4.      public void onServiceConnected(VehicleServiceName serviceName) {
5.      }
6.      @Override
```

```
        public void onServiceDisconnected(VehicleServiceName serviceName) {
        }
    };
    // 连接指定车机专有硬件服务
    try {
        Vehicle.connect(VehicleServiceName.VEHICLECONTROL_SERVICE, listener);
        Thread.sleep(2000);
        return true;
    } catch (IllegalStateException | InterruptedException e) {
        Logger.info("Exception:" + e.toString());
        return false;
    }
```

2. 根据不同管理入口类，调用对应接口。

```
// VehicleCabinManager 类，座舱天窗管理
String propId = VehicleCabinManager.ID_CABIN_SUNROOF_SWITCH;
int zoneId = VehicleZone.ZONE_NONE;
String value = "Inactive";
VehicleActuatorCallback callback = new VehicleActuatorCallback() {
    @Override
    public void onErrorActuator(String propId, int zoneId, int outResult) {
    }
};
boolean result = false;
try {
    VehicleCabinManager.setVehicleActuator(propId, zoneId, callback, value);
    result = true;
} catch (RemoteException | IllegalArgumentException e) {
    result = false;
}
if (!result) {
    System.out.println(String.format("Set sunroof error: %d", result));
}

// VehicleBodyManager 类，获取车身前挡风玻璃雨刷器状态
zoneId = VehicleZone.ZONE_FRONT;
String signal Value = VehicleBodyManager.getVehicleSignal(String.class,
VehicleBodyManager.ID_BODY_WINDSHIELD_WIPING_STATUS, zoneId);
```

```
24.
25. // VehicleChassisManager 类, 获取车辆轮胎宽度
26. zoneId = VehicleZone.ZONE_ROW1;
27. Short signalValue = VehicleChassisManager.getVehicleSignal(Short.class,
    VehicleChassisManager.ID_CHASSIS_AXLE_WHEELWIDTH, zoneId);
28.
29. // VehicleDriveTrainManager 类，设置车辆变速箱模式
30. propId = VehicleDriveTrainManager.ID_DRIVETRAIN_TRANSMISSION_PERFORMANCEMODE;
31. zoneId = VehicleZone.ZONE_NONE;
32. String transmissionValue = "sport";
33. VehicleActuatorCallback tmCallback = new VehicleActuatorCallback() {
34.       @Override
35.       public void onErrorActuator(String propId, int zoneId, int outResult) {
36.       }
37. };
38. try {
39.       VehicleDriveTrainManager.setVehicleActuator(propId, zoneId, tmCallback, transmissValue);
40.       result = true;
41. } catch (RemoteException | IllegalArgumentException e) {
42.       result = false;
43. }
44. if(!result) {
45.       System.out.println(String.format("Set transmiss performance mode error: %d", result));
46. }
47. // VehicleConfigurationManager 类，获取车辆识别码
48. String vin = VehicleConfigurationManager.getVehicleVIN();
```

1.3.2 OEM 扩展接口

场景介绍

为了支持不同 OEM 车型信号矩阵定制化需求，HarmonyOS 提供了 OEM 扩展接口，用于访问/设置/订阅/去订阅 OEM 自定义信号。

说明

该功能针对不同的 OEM 车厂/车型，提供了统一的 OEM 扩展接口。

接口说明

目前 OEM 扩展接口提供的功能有如下表所示：

接口名	描述
getVehicleSignal()	获取 OEM 自定义信号实时取值。
getVehicleSignalMultiAreas()	获取指定 OEM 自定义信号的多区域值。
setVehicleActuator()	设置 OEM 自定义执行器参数值。
subscribeVehicleSignal()	订阅指定的 OEM 自定义信号。
unsubscribeVehicleSignal()	取消订阅指定的 OEM 自定义信号。
unsubscribeVehicleSignalAll()	取消订阅全部的 OEM 自定义信号。

表 1 VehicleVendorExtensionManager 的主要接口

开发步骤

根据不同管理入口类，调对应接口。

```
// 设置辅助输入信号值
String propId = "OEM_Status_DTCCountTest";
int zoneId = VehicleZone.ZONE_NONE;
Boolean value = true;
VehicleActuatorCallback callback = new VehicleActuatorCallback() {
    @Override
    public void onErrorActuator(String propId, int zoneId, int outResult) {
    }
};
bool result = true;
try {
    VehicleVendorExtensionManager.setVehicleActuator(propId, zoneId, callback, value);
} catch(RemoteException | IllegalArgumentException e) {
    result = false;
}
if(!result) {
```

```
17.         System.out.println(String.format("Set transmiss performance mode error: %d", result));
18.     }
```

1.3.3 开发 TBOX 相关应用

场景介绍

如果某款车型上装载了车载 T-BOX（Telematics BOX）盒子，开发者可以通过 HarmonyOS 提供的 T-BOX 相关接口获取或设置相关信息，如访问 T-BOX 的 xCall、定时充电等信息。

说明

该功能与具体的车厂车型相关，部分低配车型可能不具备该项功能。

接口说明

目前 TBOX 提供的功能有如下表所示：

接口名	描述
getProperty()	获取指定 TBOX 信号值。
setActuator()	设置指定 TBOX 执行器的信号值。
subscribeProperty()	订阅指定 TBOX 信号。
unsubscribeProperty()	取消订阅指定的 TBOX 信号。
unsubscribeAllProperty()	取消所有订阅的 TBOX 信号。
subscribeBatchProperties()	批量订阅 TBOX 信号。

表 1 TBoxManager 的主要接口

开发步骤

根据不同管理入口类，调对应接口。

```
// 设置 TBOX 属性值
String incorrectPath = TBoxManager.ID_TBOX_BCALL_STATUS;
byte[] result = null;
TBoxPropertyManager manager = new TBoxPropertyManager();
boolean isTrue = false;
try {
    result = manager.getBuffer(tboxPropPath);
    isTrue = true;
} catch (RemoteException | IllegalArgumentException | UnsupportedOperationException e) {
    isTrue = false;
}
```

1.3.4 开发 CLUSTER 相关应用

场景介绍

通常在汽车使用过程中，驾驶员需要设置仪表屏亮度、时间单位等参数，将电台、音乐等娱乐数据或导航数据显示在仪表屏上，因此 HarmonyOS 提供了和仪表交互相关的接口，供三方开发者开发仪表设置、显示等相关应用。

说明

该功能与具体的车厂车型相关，部分低配车型可能不具备该项功能。

接口说明

目前 Cluster 提供的功能有如下表所示：

接口名	描述
getClusterSignal()	获取指定 Cluster 信号值。
setClusterActuator()	设置指定 Cluster 执行器值。

接口名	描述
sendClusterSignal()	发送指定字节数组类型的 Cluster 信号请求信息。
subscribeClusterSignal()	订阅指定 Cluster 信号。
subscribeBatchProperties()	批量订阅 Cluster 信号。
unsubscribeClusterSignal()	取消订阅指定的 Cluster 信号。
unsubscribeClusterSignalAll()	取消所有订阅的 Cluster 信号。

表 1 ClusterManager 的主要接口

开发步骤

1. 根据不同管理入口类，调对应接口。

```
// 设置 Cluster 属性值
String propId = ClusterManager.ID_CLUSTER_SETTINGS_BRIGHTNESS;
ClusterActuatorCallback callback = new ClusterActuatorCallback() {
    @Override
    public void onErrorActuator(String propId, int errorCode) {}
};
boolean result = false;
byte[] value = new byte[1];
try {
    ClusterManager.sendClusterSignal(propId, callback, value);
    result = true;
} catch (RemoteException | IllegalArgumentException | UnsupportedOperationException e) {
    result = false;
}
```

1.3.5 开发 ADAS 相关应用

场景介绍

通常在汽车使用过程中，驾驶员希望通过显示、声音、预警、故障告警等方式感知行车危险或规划行驶路线，因此 HarmonyOS 提供了 ADAS 辅助交互相关的接口，供三方开发者开发 ADAS 设置、自动泊车等相关应用。

说明

该功能与具体的车厂车型相关，部分低配车型可能不具备该项功能。

接口说明

目前 ADAS 提供的功能主要有以下三类：

- 驾驶辅助管理类 DrivingAssistManager，提供了驾驶辅助相关方法，例如设置前向/后向碰撞预警开关、设置盲点检测开关、设置导航目的地及导航路径等；
- 公共信息管理类 InfoAssistManager，提供了 ADAS 公共信息管理的相关方法，例如获取障碍物信息、行车记录仪信息、车道线信息、驾驶员状态信息等；
- 自主泊车管理类 ParkingAssistManager，提供了泊车控制的相关方法，例如启动泊车、暂停泊车、设置泊车车位、获取泊车状态等。

接口名	描述
byte[] getAdasSignal()	获取指定字节数组类型的驾驶辅助信号值。
<T> T getAdasSignal()	获取指定驾驶辅助信号值。
setAdasActuator()	设置指定驾驶辅助信号值。
sendAdasSignal()	发送指定字节数组类型的驾驶辅助信号请求信息。
subscribeAdasSignal()	订阅指定驾驶辅助信号。
subscribeBatchProperties()	批量订阅指定驾驶辅助信号。

接口名	描述
unsubscribeAdasSignal()	取消订阅指定的驾驶辅助信号。
unsubscribeAdasSignalAll()	取消所有订阅的驾驶辅助信号。

表 1 DrivingAssistManager 的主要接口

接口名	描述
byte[] getAdasSignal()	获取指定字节数组类型的 Adas 信号值。
<T> T getAdasSignal	获取指定 Adas 信号值。
setAdasActuator()	设置指定 Adas 信号值。
sendAdasSignal()	发送指定字节数组类型的 Adas 信号请求信息。
subscribeAdasSignal()	订阅指定 Adas 信号。
subscribeBatchProperties()	批量订阅指定 Adas 信号。
unsubscribeAdasSignal()	取消订阅指定的 Adas 信号。
unsubscribeAdasSignalAll()	取消所有订阅的 Adas 信号。

表 2 InfoAssistManager 的主要接口

接口名	描述
byte[] getAdasSignal()	获取指定字节数组类型泊车信号值。
<T> T getAdasSignal()	获取指定泊车信号值。
setAdasActuator()	设置指定泊车信号值。
sendAdasSignal()	发送指定字节数组类型泊车信号请求值。

接口名	描述
subscribeAdasSignal()	订阅指定泊车信号。
subscribeBatchProperties()	批量订阅指定的泊车信号。
unsubscribeAdasSignal()	取消订阅指定的泊车信号。
unsubscribeAdasSignalAll()	取消所有订阅的泊车信号。

表 3 ParkingAssistManager 的主要接口

开发步骤

根据不同管理入口类，调对应接口。

```
// DrivingAssistManager 类使用
boolean result = false;
try {
    Boolean signalValue = DrivingAssistManager.getAdasSignal(Boolean.class,
DrivingAssistManager.ID_DRIVING_FCW_WARNING_SWITCH);
    result = true;
} catch (RemoteException | IllegalArgumentException | UnsupportedOperationException e) {
    result = false;
}

// ParkingAssistManager 类使用
String propId = ParkingAssistManager.ID_PARKING_APA_FUNCTION_SWITCH;
Boolean value = true;
AdasActuatorCallback callback = new AdasActuatorCallback() {
    @Override
    public void onErrorActuator(String propId, int outResult) {}
};
boolean result = false;
try {
     ParkingAssistManager.setAdasActuator(propId, callback, value);
     result = true;
} catch (RemoteException | IllegalArgumentException | UnsupportedOperationException e) {
     result = false;
```

```
}
// InfoAssistManager 类使用
boolean result = false;
byte[] request = {'q', 'w'};
try {
        byte[] response =
InfoAssistManager.getAdasSignal(InfoAssistManager.ID_INFO_HDMINFO, request);
        result = true;
} catch (RemoteException | IllegalArgumentException | UnsupportedOperationException e) {
        result = false;
}
```

1.4 打造车载系统应用

1.4.1 创建车载应用项目

说明

开始前，请参考 DevEco Studio 快速开始完成环境搭建、创建并运行一个项目。

配置 config.json

1. 添加访问车机硬件信息权限申请。

```
"reqPermissions": [
  {
    "name": "ohos.permission.vehicle.READ_VEHICLE_HMI_INFO",
    "reason": "",
    "usedScene": {
      "ability": [
        ".MainAbility"
      ],
      "when": "inuse"
    }
  }
]
```

2. 添加支持驾驶模式标签。

```
"abilities": {
  "name": ".carlink",
  "icon": "$carlink:icon",
  "label": "carlink",
  "supported-modes": ["drive"],
}
```

说明

1. 创建车机应用需要添加支持驾驶模式标签 **"supported-modes": ["drive"],** // 驾驶模式支持

2. 创建车辆控制应用需要申请车机信号对应的权限群组，例如读取车辆燃油类型，需要申请群组 READ_VEHICLE_FUEL_INFO。

1.4.2 添加多媒体支持

本小节主要说明 HarmonyOS 车载多媒体的使用方法，以音乐 Demo 开发为例，开发步骤如下：

1. 在布局中添加音乐播放控件。

```
<?xml version="1.0" encoding="utf-8"?>
<DirectionalLayout xmlns:ohos="http://schemas.huawei.com/res/ohos"
                   ohos:id="$+id:play_music_root"
                   ohos:width="-1"
                   ohos:height="-1"
                   ohos:left_padding="24vp"
                   ohos:right_padding="24vp"
                   ohos:orientation="1">
    <AdaptiveBoxLayout ohos:id="$+id:title_bar"
                       ohos:width="-1"
                       ohos:height="-2"
                       ohos:top_margin="24vp">
        <Image ohos:id="$+id:arrow_down_btn"
```

```
                ohos:width="24vp"
                ohos:height="24vp"
                ohos:align_parent_left="$+id:title_bar"
                ohos:image_src="$media:default.png"/>
        <Image ohos:id="$+id:music_heart_btn"
                ohos:width="24vp"
                ohos:height="24vp"
                ohos:left_of="$+id:music_hiplay_btn"
                ohos:image_src="$media:default.png"/>
        <Image ohos:id="$+id:music_hiplay_btn"
                ohos:width="24vp"
                ohos:height="24vp"
                ohos:left_margin="16vp"
                ohos:align_parent_right="$+id:title_bar"
                ohos:image_src="$media:default.png"/>
    </AdaptiveBoxLayout>
    <DirectionalLayout ohos:id="$+id:cover_container"
                        ohos:width="-1"
                        ohos:height="-2"
                        ohos:weight="1"
                        ohos:orientation="1">
        <AdaptiveBoxLayout
                ohos:id="$+id:music_cover_adapt"
                ohos:width="-1"
                ohos:height="-1">
            <DirectionalLayout
                    ohos:id="$+id:music_cover_wrap1"
                    ohos:width="-2"
                    ohos:height="-2"
                    ohos:padding="20vp"
                    ohos:orientation="1">
                <Image ohos:id="$+id:music_cover"
                        ohos:width="300vp"
                        ohos:height="300vp"
                        ohos:layout_alignment="17"
                        ohos:image_src="$media:default.png"/>
            </DirectionalLayout>
            <DirectionalLayout
```

```
52.                     ohos:id="$+id:music_cover_wrap2"
53.                     ohos:width="-1"
54.                     ohos:height="-1"
55.                     ohos:orientation="1">
56.                 <DirectionalLayout
57.                         ohos:width="-1"
58.                         ohos:height="-2"
59.                         ohos:layout_alignment="17"
60.                         ohos:top_margin="20vp"
61.                         ohos:bottom_margin="20vp"
62.                         ohos:orientation="1">
63.                     <Text ohos:id="$+id:music_title"
64.                           ohos:text_size="20vp"
65.                           ohos:shape="0"
66.                           ohos:text_color="#FF000000"
67.                           ohos:text_alignment="72"
68.                           ohos:width="-1"
69.                           ohos:height="-2"
70.                           ohos:multiple_lines="false"/>
71.                     <Text ohos:id="$+id:music_auth"
72.                           ohos:text_size="14vp"
73.                           ohos:shape="0"
74.                           ohos:top_margin="4vp"
75.                           ohos:text_color="#FF000000"
76.                           ohos:text_alignment="72"
77.                           ohos:width="-1"
78.                           ohos:height="-2"
79.                           ohos:multiple_lines="false"/>
80.                 </DirectionalLayout>
81.                 <Text ohos:id="$+id:music_lrc"
82.                       ohos:width="-1"
83.                       ohos:height="-2"
84.                       ohos:layout_alignment="17"
85.                       ohos:text="See the lights see the party the ball grows"
86.                       ohos:text_size="13vp"
87.                       ohos:text_color="#FF000000"
88.                       ohos:text_alignment="72"/>
89.             </DirectionalLayout>
```

```
90.         </AdaptiveBoxLayout>
91.     </DirectionalLayout>
92.
93.     <DirectionalLayout ohos:id="$+id:foot_wrap"
94.                        ohos:width="-1"
95.                        ohos:height="-2"
96.                        ohos:orientation="1">
97.         <DirectionalLayout ohos:id="$+id:progress_container"
98.                            ohos:width="-1"
99.                            ohos:height="-2"
100.                           ohos:top_margin="10vp"
101.                           ohos:orientation="0">
102.            <Text ohos:id="$+id:play_progress_time"
103.                  ohos:width="-2"
104.                  ohos:height="-2"
105.                  ohos:layout_alignment="16"
106.                  ohos:right_margin="6vp"
107.                  ohos:text_size="13vp"
108.                  ohos:text_color="#FF000000"
109.                  ohos:text_alignment="72"/>
110.            <SeekBar ohos:id="$+id:play_progress_bar"
111.                     ohos:width="-1"
112.                     ohos:height="14vp"
113.                     ohos:layout_alignment="16"
114.                     ohos:weight="1"/>
115.            <Text ohos:id="$+id:play_total_time"
116.                  ohos:width="-2"
117.                  ohos:height="-2"
118.                  ohos:layout_alignment="16"
119.                  ohos:left_margin="6vp"
120.                  ohos:text_size="13vp"
121.                  ohos:text_color="#FF000000"
122.                  ohos:text_alignment="72"/>
123.        </DirectionalLayout>
124.        <DirectionalLayout ohos:id="$+id:control_container"
125.                           ohos:width="-1"
126.                           ohos:height="96vp"
127.                           ohos:orientation="0">
```

```
128.        <DirectionalLayout ohos:id="$+id:control_box1"
129.                           ohos:width="-2"
130.                           ohos:height="-2"
131.                           ohos:weight="1"
132.                           ohos:layout_alignment="17"
133.                           ohos:orientation="1">
134.            <Image ohos:id="$+id:volume_down_btn"
135.                   ohos:width="24vp"
136.                   ohos:height="24vp"
137.                   ohos:layout_alignment="17"
138.                   ohos:image_src="$media:default.png"/>
139.        </DirectionalLayout>
140.        <DirectionalLayout ohos:id="$+id:control_box2"
141.                           ohos:width="-2"
142.                           ohos:height="-2"
143.                           ohos:weight="1"
144.                           ohos:layout_alignment="17"
145.                           ohos:orientation="1">
146.            <Image ohos:id="$+id:prev_btn"
147.                   ohos:width="40vp"
148.                   ohos:height="40vp"
149.                   ohos:layout_alignment="17"
150.                   ohos:image_src="$media:default.png"/>
151.        </DirectionalLayout>
152.        <DirectionalLayout ohos:id="$+id:control_box3"
153.                           ohos:width="-2"
154.                           ohos:height="-2"
155.                           ohos:weight="1"
156.                           ohos:layout_alignment="17"
157.                           ohos:orientation="1">
158.            <Image ohos:id="$+id:play_btn"
159.                   ohos:width="64vp"
160.                   ohos:height="64vp"
161.                   ohos:layout_alignment="17"
162.                   ohos:image_src="$media:default.png"/>
163.        </DirectionalLayout>
164.        <DirectionalLayout ohos:id="$+id:control_box4"
165.                           ohos:width="-2"
```

```
                                    ohos:height="-2"
                                    ohos:weight="1"
                                    ohos:layout_alignment="17"
                                    ohos:orientation="1">
                    <Image ohos:id="$+id:next_btn"
                            ohos:width="40vp"
                            ohos:height="40vp"
                            ohos:layout_alignment="17"
                            ohos:image_src="$media:default.png"/>
                </DirectionalLayout>
                <DirectionalLayout ohos:id="$+id:control_box5"
                                    ohos:width="-2"
                                    ohos:height="-2"
                                    ohos:weight="1"
                                    ohos:layout_alignment="17"
                                    ohos:orientation="1">
                    <Image ohos:id="$+id:volume_up_btn"
                            ohos:width="24vp"
                            ohos:height="24vp"
                            ohos:layout_alignment="17"
                            ohos:image_src="$media:default.png"/>
                </DirectionalLayout>
            </DirectionalLayout>
        </DirectionalLayout>
</DirectionalLayout>
```

2. 加载播放控件。

```
super.setUIContent(ResourceTable.Layout_play_music_layout);
```

3. 实现音乐播放管理类。

```
public class PlayManager {
    ...
    private Player player;
    public synchronized boolean play(String filePath, int startMilliSecond) {
        ...
        FileDescriptor fd = IoUtil.getFileDescriptor(filePath);
        Source source = new Source(fd);
        player.setSource(source);
        boolean isSuccess = player.prepare();
```

```
10.
11.             isSuccess = player.rewindTo(startMilliSecond * MICRO_MILLI_RATE,
    REWIND_NEXT_SYNC);
12.             // 播放
13.             isSuccess = player.play();
14.             isPlaying.set(isSuccess);
15.             return isSuccess;
16.         }
17.
18.         public synchronized void pause(int startMilliSecond) {
19.             ...
20.             player.pause();
21.         }
22.
23.         public synchronized void stop() {
24.             if (player == null) {
25.                 return;
26.             }
27.             player.stop();
28.             isPlaying.set(false);
29.             LogUtil.info(TAG, "stop success");
30.             player.release();
31.             player = null;
32.         }
33. }
```

4. 调用音乐播放管理类的接口播放音乐。

```
1.  // 指定歌曲播放
2.  String path = "/data/music/files/data/wonderful_life.mp3";
3.  PlayManager.getInstance().play(path,1);
```

5. 在布局中增加视频播放控件。

```
1.  // 视频布局实现方法
2.  public class MySurfaceSlice extends AbilitySlice {
3.      ...
4.      public void makeSurfaceView() {
5.          ...
6.          mySurfaceProvider = new SurfaceProvider(this);
```

```
7.
    adaptiveBoxLayoutSurfaceView.AdaptiveBoxLayout.LayoutConfig().addComponent(mySurfacePr
    ovider);
8.      }
9.  }
```

6. 实现视频播放管理类。

```
1.  public class VideoPlay {
2.      public synchronized void startPlay() {
3.          ...
4.          ret = playImpl.play();
5.      }
6.
7.      public synchronized void preParePlay() {
8.          ...
9.          ret = playImpl.prepare();
10.     }
11.
12.     public synchronized void pausePlay() {
13.         ...
14.         boolean pauseRet = playImpl.pause();;
15.     }
16.
17.     public synchronized void setSourcePlay(String filePath) {
18.          ...
19.         FileDescriptor fd = IoUtil.getFileDescriptor(filePath);
20.         Source source = new Source(fd);
21.         playImpl.setSource(source);
22.     }
23.
24.     @Override
25.     public synchronized void onStop() {
26.         ...
27.         super.onStop();
28.     }
29. }
```

7. 调用视频播放管理类的接口播放视频。

```
// 调用视频播放类进行播放
String filePath = "/data/video/files/data/festival.mp4";
VideoPlay videoPlay = new VideoPlay()
videoPlay.setSourcePlay(filePath);
videoPlay.startPlay();
```

2 智能穿戴

2.1 概述

对于智能穿戴，应用可以通过 HarmonyOS 提供的接口实现音频、传感器、网络连接、UI 交互、消息提醒等常规业务的开发。开发者也可以根据智能穿戴的特点，打造针对智能穿戴的独特应用。

说明

本文档适用于智能穿戴应用开发，针对轻量级智能穿戴请参考轻量级智能穿戴开发。

基于 HarmonyOS，开发者既可以在智能穿戴上开发独立工作的应用，也可以开发跨设备协同工作的应用，为消费者带来更加灵活、智慧的分布式交互体验。

当开发者需要新建一个工程开发智能穿戴应用时，请参考打造智能穿戴应用。

当开发者已有一个工程需要添加一个智能穿戴模块时，请参考添加智能穿戴模块。

功能	描述
应用向系统发送通知	用于提醒用户有来自应用的信息。当应用向系统发出通知时，通知将按照应用维度在通知中心聚合显示，详情请参考创建智能穿戴应用通知。
降低应用功耗	智能穿戴电池容量有限，为了让应用对用户更友好，开发者应当尽可能降低应用的功耗开销，详情请参考降低应用功耗。

表 1 智能穿戴应用开发的增强功能

基本概念

- **表盘**

智能穿戴配对完成后，开机首界面即为表盘，系统会预置一些表盘，让用户可以通过表盘快速地查看时间、计步、心率、天气等关键信息。用户也可以根据自己的喜好，长按表盘界面以选择切换自己喜爱的表盘。

2.2 打造智能穿戴应用

在开始进行智能穿戴应用开发前，请参考 DevEco Studio 快速开始完成环境搭建、创建并运行一个项目。设备类型选择“Wearable”。

智能穿戴应用支持 Java 和 JS 两种开发模式。但以下两种场景，暂时仅支持使用 Java 开发：

1. 如果开发的应用内嵌算法，需要通过 JNI （Java Native Interface，Java 本地接口）调用 so 库中的函数。
2. 应用需要较高的运算效率。

下面将介绍如何使用 JS 和 JAVA 开发一个睡眠检测应用界面。

适配圆形屏幕

在 HarmonyOS 智能穿戴应用的开发中，请使用通用的 UI 控件。针对圆形的智能穿戴，开发者需要将应用界面适配圆形屏幕，以带来更好的用户体验。应用在实际显示时仅会显示界面设计中的部分圆形界面，如示例图所示。

开发一个宽 400 高 1200 的竖长型界面，当上下滑动的时候，用户只能看到橘色圆圈内部的样式，其余部分不会展示。所以开发者在进行界面设计时，需要根据智能穿戴形状进行设计适配。

另外，穿戴设备的界面一般支持右滑退出，所以需要在 PageAblity 启动时进行设置，在 onStart 里调用 setSwipeToDismiss(true)，便于右滑退出。

图 1 圆形屏幕内容展示示例图

调试应用

在开启应用调试之前，需要在智能穿戴上开启开发者模式。

1. 进入“设置 > 关于手表”，查看智能穿戴设备信息。滑动到“版本号”的位置，点击 3 次，开启开发者模式。
2. 返回设置界面，进入新出现的“开发人员选项”界面，打开“开发人员选项”开关。

部分智能穿戴仅支持无线充电，开发者无法通过 USB 连接方式去开发和调试应用，可以通过 WLAN 进行调试。调试方法如下：

1. 使用路由器设置一个无密码的 WLAN 网络，将开发应用的 PC 接入该路由器。

2. 打开智能穿戴的“设置 > WLAN”，打开 WLAN 开关，将智能穿戴接入上述 WLAN 网络。
3. 进入开发人员选项，查看 IP 地址。
4. 在 PC 端打开 DevEco Studio，在上方导航栏选择“Tools > IP > Connect”。
5. 在弹出的窗口中，输入智能穿戴的 IP 地址。完成连接，可以开始进行应用调试。

当不需要进行应用调试时，可以进入“设置 > 开发人员选项”，关闭“开发人员选项”开关，退出开发者模式。

2.3 添加智能穿戴模块

以下根据实际的开发样例来展示如何在已有的 HarmonyOS 工程中添加一个智能穿戴模块。

如图所示，这是一个睡眠检测应用，应用分为主界面和详情界面，可以选择使用 PageSlider 实现界面间的切换。PageSlider 是一个布局管理器，用于实现左右滑动以及上下滑动的翻页效果。

图 1 开发样例效果图

1. 在 DevEco Studio 上方的导航栏中，依次点击“File > New > Module”，在“Device”中选择“Wearable”，添加一个新模块。
2. 在左侧的“Project”窗口，打开“entry > src > main > resources > base”，右键点击“base”文件夹，选择“New > Directory”，命名为“layout”。

3. 右键点击“layout”文件夹，选择“New > File”，新建两个 UI 布局文件，分别命名为“layout_sleep.xml”和“layout_detail.xml”。

主界面的 UI 布局文件是“layout_sleep.xml”，其完整示例代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<DirectionalLayout xmlns:ohos="http://schemas.huawei.com/res/ohos"
                   ohos:width="match_parent"
                   ohos:height="match_parent"
                   ohos:background_element="#FF000000"
                   ohos:orientation="vertical">

    <Image
            ohos:id="$+id:sleep_moon_img"
            ohos:width="46vp"
            ohos:height="46vp"
            ohos:top_margin="11vp"
            ohos:layout_alignment="horizontal_center"/>

    <Text
            ohos:width="match_parent"
            ohos:height="19.5vp"
            ohos:alpha="0.66"
            ohos:layout_alignment="horizontal_center"
            ohos:text_alignment="center"
            ohos:text="$string:sleep"
            ohos:text_color="$color:sleep_text_white"
            ohos:text_size="16vp"/>

    <DirectionalLayout xmlns:ohos="http://schemas.huawei.com/res/ohos"
                       ohos:width="match_content"
                       ohos:height="65vp"
                       ohos:top_margin="8vp"
                       ohos:layout_alignment="horizontal_center"
                       ohos:orientation="horizontal">
        <Text
                ohos:id="$+id:sleep_hour_text"
                ohos:width="match_content"
```

```
34.                 ohos:height="match_content"
35.                 ohos:layout_alignment="center"
36.                 ohos:text_alignment="center"
37.                 ohos:text="$string:dash"
38.                 ohos:text_color="$color:sleep_text_white"
39.                 ohos:text_size="58vp"
40.         />
41.         <Text
42.                 ohos:width="match_content"
43.                 ohos:height="match_content"
44.                 ohos:left_margin="2vp"
45.                 ohos:alpha="0.66"
46.                 ohos:layout_alignment="bottom"
47.                 ohos:bottom_padding="9.5vp"
48.                 ohos:text="$string:hour"
49.                 ohos:text_color="$color:sleep_text_white"
50.                 ohos:text_size="16vp"
51.         />
52.         <Text
53.                 ohos:id="$+id:sleep_min_text"
54.                 ohos:width="match_content"
55.                 ohos:height="match_content"
56.                 ohos:left_margin="2vp"
57.                 ohos:layout_alignment="center"
58.                 ohos:text_alignment="center"
59.                 ohos:text="$string:double_dash"
60.                 ohos:text_color="$color:sleep_text_white"
61.                 ohos:text_size="58vp"
62.         />
63.         <Text
64.                 ohos:width="match_content"
65.                 ohos:height="match_content"
66.                 ohos:left_margin="2vp"
67.                 ohos:alpha="0.66"
68.                 ohos:layout_alignment="bottom"
69.                 ohos:bottom_padding="9.5vp"
70.                 ohos:text="$string:minute"
71.                 ohos:text_color="$color:sleep_text_white"
```

```
                ohos:text_size="16vp"
        />
    </DirectionalLayout>
    <DirectionalLayout xmlns:ohos="http://schemas.huawei.com/res/ohos"
                ohos:width="match_content"
                ohos:height="25vp"
                ohos:top_margin="20.5vp"
                ohos:layout_alignment="horizontal_center"
                ohos:orientation="horizontal">
        <Text
                ohos:width="match_content"
                ohos:height="19.5vp"
                ohos:text="$string:goal"
                ohos:alpha="0.66"
                ohos:text_alignment="bottom"
                ohos:bottom_margin="1vp"
                ohos:text_color="$color:sleep_text_white"
                ohos:text_size="16vp"
        />
        <Text
                ohos:id="$+id:sleep_goal_text"
                ohos:width="match_content"
                ohos:height="match_parent"
                ohos:left_margin="2vp"
                ohos:text="$string:target_sleep_time"
                ohos:text_weight="600"
                ohos:text_color="$color:sleep_text_white"
                ohos:bottom_padding="2vp"
                ohos:text_size="21vp"
        />
        <Text
                ohos:width="match_content"
                ohos:height="19.5vp"
                ohos:left_margin="2vp"
                ohos:alpha="0.66"
                ohos:text="$string:hour"
                ohos:text_color="$color:sleep_text_white"
                ohos:text_size="16vp"
```

```
        />
    </DirectionalLayout>
</DirectionalLayout>
```

详情界面的 UI 布局文件是“layout_detail.xml”，其完整示例代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<DirectionalLayout xmlns:ohos="http://schemas.huawei.com/res/ohos"
                   ohos:width="match_parent"
                   ohos:height="match_parent"
                   ohos:orientation="vertical"
                   ohos:background_element="#FF000000">
    <Text
            ohos:id="$+id:detail_nodata_date"
            ohos:width="match_content"
            ohos:height="23vp"
            ohos:top_margin="20vp"
            ohos:layout_alignment="horizontal_center"
            ohos:text_alignment="bottom"
            ohos:text_color="$color:sleep_text_white"
            ohos:text_weight="600"
            ohos:text_size="19vp"/>

    <Image
            ohos:id="$+id:detail_nodata_img"
            ohos:width="46vp"
            ohos:height="46vp"
            ohos:top_margin="25vp"
            ohos:layout_alignment="horizontal_center"
            ohos:scale_type="scale_to_center"/>

    <Text
            ohos:width="match_content"
            ohos:height="match_content"
            ohos:alpha="0.66"
            ohos:top_margin="12vp"
            ohos:layout_alignment="horizontal_center"
            ohos:text_alignment="center"
            ohos:text="$string:no_data"
```

```
                ohos:text_color="$color:sleep_text_white"
                ohos:text_size="16vp"/>
        <Text
                ohos:width="match_content"
                ohos:height="match_content"
                ohos:alpha="0.66"
                ohos:layout_alignment="horizontal_center"
                ohos:text_alignment="center"
                ohos:text="$string:wearing_watch_tips"
                ohos:text_color="$color:sleep_text_white"
                ohos:text_size="16vp"/>
</DirectionalLayout>
```

2. 在左侧项目文件栏中，选择“entry > src > main > java > 应用包名 > slice”，在对应的 AbilitySlice 文件的 onStart 里，使用代码创建 PageSlider，添加这两个相应的界面。

```
public class SleepPageSlice extends AbilitySlice {
    private static final String TAG = "SleepPageSlice";

    private static final HiLogLabel LABEL = new HiLogLabel(HiLog.LOG_APP, 0, TAG);

    private List<ComponentOwner> list = new ArrayList<>();

    private PageSliderProvider provider = new PageSliderProvider() {
        @Override
        public int getCount() {
            return list.size();
        }

        @Override
        public Object createPageInContainer(ComponentContainer componentContainer, int index) {
            if (index >= list.size() || componentContainer == null) {
                HiLog.error(LABEL, "instantiateItem index error");
                return Optional.empty();
            }
            ComponentOwner container = list.get(index);
            componentContainer.addComponent(container.getComponent());
```

```
22.                 container.instantiateComponent();
23.                 return container.getComponent();
24.             }
25.
26.             @Override
27.             public void destroyPageFromContainer(ComponentContainer componentContainer, int
    index, Object object) {
28.                 HiLog.info(LABEL, "destroyItem index:" + index);
29.                 if (index >= list.size() || componentContainer == null) {
30.                     return;
31.                 }
32.                 Component content = list.get(index).getComponent();
33.                 componentContainer.removeComponent(content);
34.                 return;
35.             }
36.
37.             @Override
38.             public boolean isPageMatchToObject(Component component, Object object) {
39.                 return component == object;
40.             }
41.
42.             @Override
43.             public void startUpdate(ComponentContainer container) {
44.                 super.startUpdate(container);
45.                 HiLog.info(LABEL, "startUpdate");
46.             }
47.         };
48.
49.         @Override
50.         public void onStart(Intent intent) {
51.             super.onStart(intent);
52.             HiLog.info(LABEL, "onStart");
53.
54.             // 添加子页面
55.             list.add(new SleepComponentOwner(this));
56.             list.add(new DetailComponentOwner(this));
57.
58.             // 设置主界面
```

```
59.             DirectionalLayout layout = new DirectionalLayout(this);
60.             ComponentContainer.LayoutConfig config = new ComponentContainer.LayoutConfig(
61.                     ComponentContainer.LayoutConfig.MATCH_PARENT,
62.                     ComponentContainer.LayoutConfig.MATCH_PARENT);
63.             layout.setLayoutConfig(config);
64.
65.             // 使用 PageSlider 做滑动效果
66.             PageSlider slider = new PageSlider(this);
67.             ComponentContainer.LayoutConfig sliderConfig = new
    ComponentContainer.LayoutConfig(
68.                     ComponentContainer.LayoutConfig.MATCH_PARENT,
69.                     ComponentContainer.LayoutConfig.MATCH_PARENT);
70.             slider.setLayoutConfig(sliderConfig);
71.             slider.setOrientation(DirectionalLayout.VERTICAL);
72.             slider.setProvider(provider);
73.
74.             layout.addComponent(slider);
75.
76.             setUIContent(layout);
77.         }
78. }
```

3. 增加 ComponentOwner 容器接口和两个页面的实现方式，如下是容器的接口 ComponentOwner.java。

```
1.  public interface ComponentOwner {
2.      // 获取存放的 component
3.      Component getComponent();
4.
5.      // 当包含的 component 被添加到容器时回调
6.      void instantiateComponent();
7.  }
```

4. 增加第一个页面 SleepComponentOwner 和第二个页面 DetailComponentOwner，这两个页面从 xml 里加载。以下是首页 SleepComponentOwner 的实现。

```
1.  public class SleepComponentOwner implements ComponentOwner {
2.      private static final String TAG = "SleepViewContainer";
3.
```

```
4.      private static final HiLogLabel LABEL = new HiLogLabel(HiLog.LOG_APP, 0, TAG);
5.
6.      // 目标睡眠时长默认值,单位：分钟
7.      private static final int DEFAULT_SLEEP_TARGET_TIME = 480;
8.
9.      // 睡眠时长默认值,单位：分钟
10.     private static final int DEFAULT_SLEEP_TIME = 0;
11.
12.     private CircleProgressDrawTask drawTask;
13.
14.     private AbilityContext myContext;
15.
16.     private Component root;
17.
18.     public SleepComponentOwner(AbilityContext context) {
19.         init(context);
20.     }
21.
22.     private void init(AbilityContext context) {
23.         myContext = context;
24.         LayoutScatter scatter = LayoutScatter.getInstance(context);
25.         root = scatter.parse(ResourceTable.Layout_layout_sleep, null, false);
26.         drawTask = new CircleProgressDrawTask(root);
27.         drawTask.setMaxValue(DEFAULT_SLEEP_TARGET_TIME);
28.
29.         Component imageView = root.findComponentById(ResourceTable.Id_sleep_moon_img);
30.         imageView.setBackground(new VectorElement(context,
    ResourceTable.Graphic_ic_icon_moon));
31.     }
32.
33.     @Override
34.     public Component getComponent() {
35.         return root;
36.     }
37.
38.     @Override
39.     public void instantiateComponent() {
40.         return;
```

```
    }
}
```

以下是第二个页面 DetailComponentOwner 的实现。

```
public class DetailComponentOwner implements ComponentOwner {
    private static final String TAG = "DetailViewContainer";

    private static final HiLogLabel LABEL = new HiLogLabel(HiLog.LOG_APP, 0, TAG);

    private AbilityContext myContext;

    private ComponentContainer root;

    public DetailComponentOwner(AbilityContext context) {
        init(context);
    }

    private void init(AbilityContext context) {
        root = new DirectionalLayout(context);
        ComponentContainer.LayoutConfig config = new ComponentContainer.LayoutConfig(
                ComponentContainer.LayoutConfig.MATCH_PARENT,
                ComponentContainer.LayoutConfig.MATCH_PARENT);
        root.setLayoutConfig(config);
        myContext = context;
    }

    @Override
    public Component getComponent() {
        return root;
    }

    @Override
    public void instantiateComponent() {
        return;
    }
}
```

2. 增加一个类型为 Page 的 Ability，并在 config.json 里进行注册。需要在 onStart 里调用 setSwipeToDismiss(true)，来设置右滑退出。示例代码如下：

```
public class PageAbility extends Ability {
    @Override
    public void onStart(Intent intent) {
        super.onStart(intent);
        super.setMainRoute(SleepPageSlice.class.getName());
        setSwipeToDismiss(true);
    }
}
```

如下是配置文件 config.json，注册 PageAbility 的时候，需要指明 action.system.home，用来保证该 Ability 能在 launcher 上显示对应的图标。

```
{
    "app": {
        "bundleName": "com.huawei.health.sleep",
        "vendor": "huawei",
        "version": {
            "code": 1,
            "name": "1.0.8.27"
        },
        "apiVersion: {
            "compatible": 3,
            "target": 3
        }
    },
    "deviceConfig": {
        "default": {
            }
        }
    },
    "module": {
        "package": "com.huawei.health.sleep",
        "name": ".SleepApplication",
        "distro": {
            "moduleType": "entry",
```

```
            "deliveryWithInstall": true,
            "moduleName": "entry"
        },
        "deviceType": [
            "wearable"
        ],
        "reqCapabilities": [
            "video_support"
        ],
        "abilities": [
            {
                "name": ".PageAbility",
                "description": "$string:ability_description",
                "icon": "$media:icon_app",
                "label": "$string:app_name",
                "launchType": "standard",
                "orientation": "portrait",
                "visible": true,
                "permissions": [],
                "skills": [
                    {
                        "actions": [
                            "action.system.home"
                        ],
                        "entities": [
                            "entity.system.home"
                        ],
                    }
                ],
                "type": "page",
                "formEnabled": false
            }
        ]
    }
}

```

在睡眠界面中，我们用到了圆环效果，这里我们看一下圆环效果是如何实现的，如何实现自定义 Component 的效果。调用方代码如下：

```
drawTask = new CircleProgressDrawTask(root);
```

Componet 类提供了 UI 的基本组件，包括方法 addDrawTask(Component.DrawTask task)。该方法可以给任意一个 Componet 添加一段自定义绘制的代码。自定义 Component 的实现方法如下：

1. 创建一个自定义 DrawTask，包含与该 Componet 相关的自定义属性和绘制的代码。
2. 在构造方法里传入宿主 Component，跟自定义的 DrawTask 绑定。

实现睡眠圆环效果的示例代码如下。

```
public class CircleProgressDrawTask implements Component.DrawTask {
    // 用于配置圆环的粗细，具体参数可以在 xml 文件中配置
    private static final String STROKE_WIDTH_KEY = "stroke_width";

    // 用于配置圆环的最大值，具体参数可以在 xml 文件中配置
    private static final String MAX_PROGRESS_KEY = "max_progress";

    // 用于配置圆环的当前值，具体参数可以在 xml 文件中配置
    private static final String CURRENT_PROGRESS_KEY = "current_progress";

    // 用于配置起始位置的颜色，具体参数可以在 xml 文件中配置
    private static final String START_COLOR_KEY = "start_color";

    // 用于配置结束位置的颜色，具体参数可以在 xml 文件中配置
    private static final String END_COLOR_KEY = "end_color";

    // 用于配置背景色，具体参数可以在 xml 文件中配置
    private static final String BACKGROUND_COLOR_KEY = "background_color";

```

```
20.     // 用于配置起始位置的角度，具体参数可以在 xml 文件中配置
21.     private static final String START_ANGLE = "start_angle";
22.
23.     private static final float MAX_ARC = 360f;
24.
25.     private static final int DEFAULT_STROKE_WIDTH = 10;
26.
27.     private static final int DEFAULT_MAX_VALUE = 100;
28.
29.     private static final int DEFAULT_START_COLOR = 0xFFB566FF;
30.
31.     private static final int DEFAULT_END_COLOR = 0xFF8A2BE2;
32.
33.     private static final int DEFAULT_BACKGROUND_COLOR = 0xA8FFFFFF;
34.
35.     private static final int DEFAULT_START_ANGLE = -90;
36.
37.     private static final float DEFAULT_LINER_MAX = 100f;
38.
39.     private static final int HALF = 2;
40.
41.     // 圆环的宽度，默认 10 个像素
42.     private int myStrokeWidth = DEFAULT_STROKE_WIDTH;
43.
44.     // 最大的进度值，默认是 100
45.     private int myMaxValue = DEFAULT_MAX_VALUE;
46.
47.     // 当前的进度值，默认是 0
48.     private int myCurrentValue = 0;
49.
50.     // 起始位置的颜色，默认浅紫色
51.     private Color myStartColor = new Color(DEFAULT_START_COLOR);
52.
53.     // 结束位置的颜色，默认深紫色
54.     private Color myEndColor = new Color(DEFAULT_END_COLOR);
55.
56.     // 背景颜色，默认浅灰色
57.     private Color myBackgroundColor = new Color(DEFAULT_BACKGROUND_COLOR);
```

```
58.
59.     // 当前的进度值，默认从-90 度进行绘制
60.     private int myStartAngle = DEFAULT_START_ANGLE;
61.
62.     private Component myComponent;
63.
64.     // 传入要进行修改的 component
65.     public CircleProgressDrawTask(Component component) {
66.         myComponent = component;
67.         myComponent.addDrawTask(this);
68.     }
69.
70.     // 设置当前进度并且刷新 component，value 值为当前进度
71.     public void setValue(int value) {
72.         myCurrentValue = value;
73.         myComponent.invalidate();
74.     }
75.
76.     public void setMaxValue(int maxValue) {
77.         myMaxValue = maxValue;
78.         myComponent.invalidate();
79.     }
80.
81.     @Override
82.     public void onDraw(Component component, Canvas canvas) {
83.         // 通过 canvas 实现绘制圆环的功能
84.         ......
85.     }
86. }
87.
```

2.4 创建智能穿戴设备应用通知

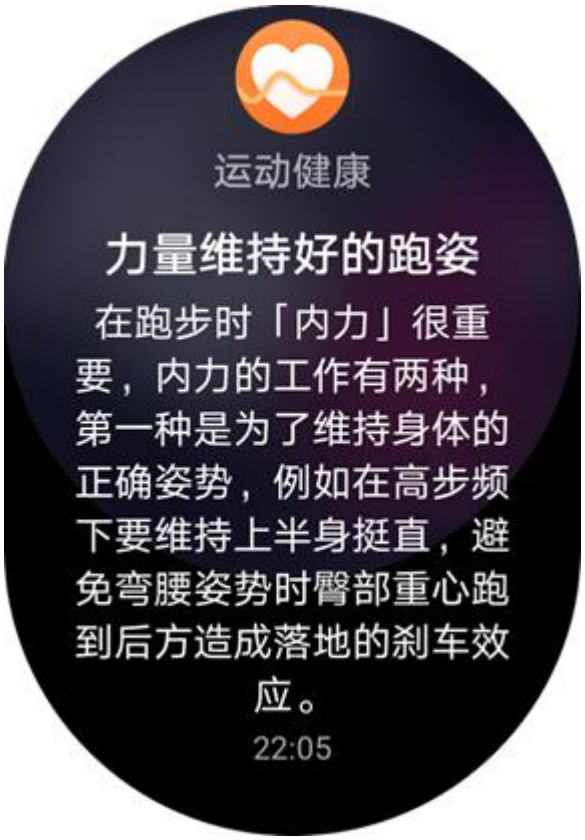
2.4.1 介绍

HarmonyOS 提供了通知功能，提醒用户有来自应用的信息。当应用向系统发出通知时，通知将以弹窗的形式显示，并同时出现在通知中心。用户可以在表盘界面上拉，通过通知中心查看通知的详细信息。常见的使用场景有：

- 显示接收到短消息、即时消息等。
- 显示应用的推送消息，如广告、版本更新等。
- 显示当前正在进行的事件，如勿扰模式等。

通知的样式

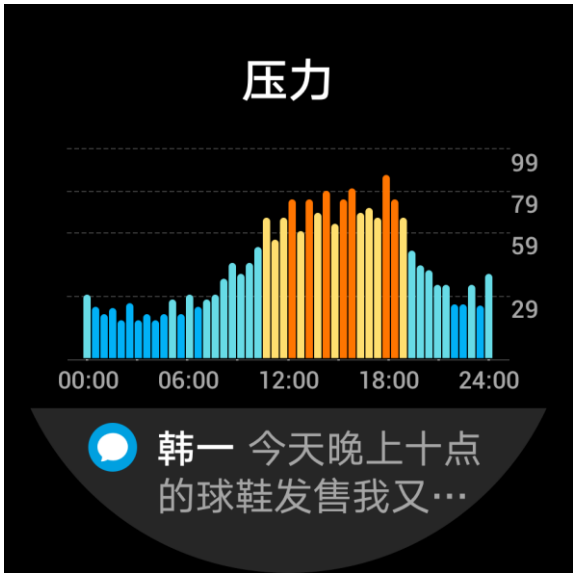
- 当在表盘界面收到消息通知，会全屏显示，示例如下：



- 在表盘界面上拉可以打开通知中心，消息通知会自动按应用维度进行聚合展示，示例如下：

- 聚合的消息会有小红点提示折叠的消息数量。点击消息可打开折叠的消息详情，示例如下：

- 当在非表盘界面收到消息通知，会在屏幕下方弹出浮窗显示，示例如下：



约束与限制

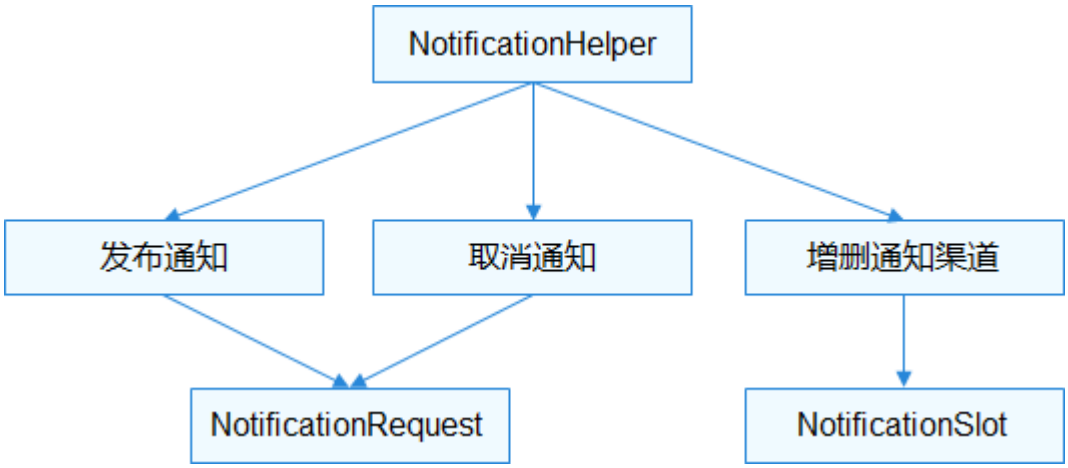
- 通知目前支持三种样式：普通文本、长文本、图片。
- 通知不支持快捷回复，不支持自定义布局。
- 目前通知订阅不支持多用户。
- 通知的订阅目前仅支持系统应用，不支持第三方应用。

2.4.2 开发指导

接口介绍

通知相关基础类包含 NotificationSlot、NotificationRequest 和 NotificationHelper。详细的接口信息请参考通知开发指导。基础类之间的关系如下所示：

图 1 通知基础类关系图



- **NotificationSlot**

  NotificationSlot 可以对提示音、振动和重要级别等进行设置。一个应用可以创建一个或多个 NotificationSlot，在发送通知时，通过绑定不同的 NotificationSlot，实现不同用途。

  说明

  NotificationSlot 需要先通过 NotificationHelper 的 addNotificationSlot(NotificationSlot)方法发布后，通知才能绑定使用；所有绑定该 NotificationSlot 的通知在发布后都具备相应的特性，对象在创建后，将无法更改其设置属性，对于是否启动相应设置，用户有最终控制权。

  不指定 NotificationSlot 时，当前通知会使用默认的 NotificationSlot，默认的 NotificationSlot 优先级为 LEVEL_DEFAULT，声音为系统默认提示音。

  NotificationSlot 的级别目前支持如下几种：

  - LEVEL_NONE： 表示通知不发布。
  - LEVEL_MIN/LEVEL_LOW/LEVEL_DEFAULT/LEVEL_HIGH：表示通知发布后可在通知中心显示，自动弹出，触发提示音。

- **NotificationRequest**

NotificationRequest 用于设置具体的通知对象，包括设置通知的属性，如：通知的小图标、自动删除等参数，以及设置具体的通知类型，如普通文本、长文本等。

通知的常用属性：

- 小图标

标识说明：为通过 NotificationRequest.setLittleIcon(PixelMap)设置的小图标。

- 从通知启动 Ability：点击通知栏的通知，可以通过启动 Ability，触发新的事件。

  通知设置 NotificationRequest 的 setIntentAgent(IntentAgent)后，点击通知栏上发布的通知，将触发通知中的 IntentAgent 承载的事件。IntentAgent 的设置请参考 IntentAgent 开发指导。

- 通知设置 ActionButton：通过点击通知按钮，可以触发按钮承载的事件。

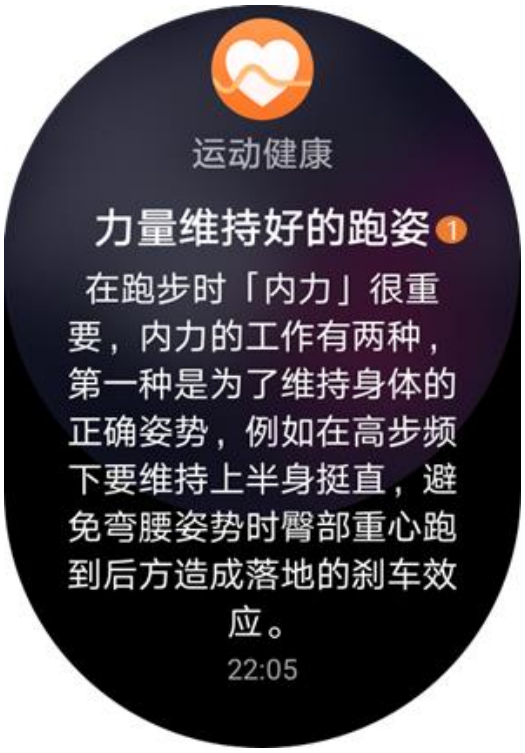
标识说明：、为两个通过NotificationRequest.addActionButton(NotificationActionButton)设置的通知附加按钮，点击按钮后可以触发相关的事件，具体事件内容如何设置需要参考 NotificationActionButton。

具体的通知类型：目前支持三种类型，包括普通文本 NotificationNormalContent、长文本 NotificationLongTextContent、图片 NotificationPitctureContent。

- 普通文本通知样式（NotificationNormalContent）

标识说明：为通知的标题，通过 NotificationNormalContent.setTitle(String)设置。为通知的内容，通过 NotificationNormalContent.setText(String)设置。通知标题和内容至少要设置一个。

- 长文本通知样式（NotificationLongTextContent）



标识说明：为通知的长文本，通过 NotificationLongTextContent.setLongText(String)设置，文本长度最大支持 1024 个字符。

- 图片通知样式（NotificationPictureContent）

标识说明： 为图片通知样式的图片，通过

NotificationPictureContent.setBigPicture(PixelMap bigPicture)设置。

说明

通知发布后，通知的设置不可修改。如果下次发布通知使用相同的 ID，就会更新之前发布的通知。

- **NotificationHelper**

NotificationHelper 封装了发布、更新、订阅、删除通知等静态方法。订阅通知、退订通知和查询系统中所有处于活跃状态的通知，有权限要求需为系统应用或具有订阅者权限。

开发步骤

通知的开发指导分为创建 NotificationSlot、发布通知和取消通知等开发场景。

创建 NotificationSlot

NotificationSlot 可以设置公共通知的提示声等，并通过调用 NotificationHelper.addNotificationSlot()发布 NotificationSlot 对象。

```
// 创建 notificationSlot 对象
NotificationSlot slot = new NotificationSlot("slot_001","slot_default",
NotificationSlot.LEVEL_DEFAULT);
slot.setDescription("NotificationSlotDescription");
try {
    NotificationHelper.addNotificationSlot(slot);
} catch (RemoteException ex) {
    HiLog.warn(LABEL, "addNotificationSlot occur exception.");
}
```

发布通知

1. 构建 NotificationRequest 对象，应用发布通知前，通过 NotificationRequet 的 setSlot()方法与 NotificationSlot 绑定，使该通知在发布后都具备该对象的特征。

```
int notification_id = 1;
NotificationRequest request = new NotificationRequest(notification_id);
request.setSlotId(slot.getId());
```

2. 调用 setContent()设置通知的内容。

```
String title = "title";
String text = "There is a normal notification content.";
NotificationRequest.NotificationNormalContent content = new
NotificationRequest.NotificationNormalContent();
content.setTitle(title)
        .setText(text);
NotificationRequest.NotificationContent notificationContent = new
NotificationRequest.NotificationContent(content);
// 设置通知的小图标
request.setLittleIcon(PixelMap);
// 设置通知的内容
```

```
request.setContent(notificationContent);
```

3. 调用 setIntentAgent()设置通知可以触发的事件。

```
// 指定要启动的 ability 的 ElementName 字段
ElementName elementName = new ElementName("", "com.example.testintentagent",
"com.example.testintentagent.IntentAgentAbility");
// 将 ElementName 字段添加到 Intent 中
Intent intent = new Intent();
intent.setElement(elementName);
List<Intent> intentList = new ArrayList<>();
intentList.add(intent);
// 指定启动一个有页面的 ability
IntentAgentInfo intenAgentinfo = new IntentAgentInfo(request.getNotificationId(),
IntentAgentConstant.OperationType.START_ABILITY,
IntentAgentConstant.Flags.UPDATE_PRESENT_FLAG, intentList, null);
// 获取 IntentAgent 实例
IntentAgent intentAgent = IntentAgentHelper.getIntentAgent(mContext, intenAgentinfo);
request.setIntentAgent(intentAgent);
request.setTapDismissed(true);
```

4. 调用 publishNotification()发送通知。

```
try {
    NotificationHelper.publishNotification(request);
} catch (RemoteException ex) {
    HiLog.warn(LABEL, "publishNotification occur exception.");
}
```

取消通知

取消通知分为取消指定单条通知和取消所有通知，应用只能取消自己发布的通知。

- 调用 cancelNotification()取消指定的单条通知。

```
int notification_id = 1;
try {
    NotificationHelper.cancelNotification(notification_id);
} catch (RemoteException ex) {
```

```
5.          HiLog.warn(LABEL, "cancelNotification occur exception.");
6.      }
```

2.5 降低应用功耗

当需要针对智能穿戴开发低功耗应用时，推荐开发者使用深色主题模式。

示例如下：

开发注意事项

智能穿戴电池容量有限，为了让应用对用户更友好，开发者应当尽可能降低应用的功耗开销。以下注意事项供开发者参考：

1. 避免长时间的屏幕常亮从而阻止系统休眠：除视频、游戏、导航等用户可感知的业务场景外，原则上应用不允许做屏幕常亮的设计。同时，禁止任何后台应用设置屏幕常亮。
2. 灭屏情况下，避免频繁唤醒系统：心跳类唤醒系统的频率建议每小时不超过 12 次，闹钟、日程提醒、邮件、IM 类应用按需唤醒系统，其他类的应用禁止唤醒系统。
3. 避免应用频繁自启：除被前台应用拉起的情况外，闹钟、日程提醒、邮件、IM 类应用按需自启，其他类的应用禁止自启。
4. 应用不应在后台长时间使用 GPS：除导航类、轨迹跟踪类、运动健康类应用，禁止非用户可感知业务进行后台定位。
5. 除用户可感知的业务外，禁止应用在后台造成 CPU 高负载耗电。
6. 除用户可感知的业务外，禁止应用在灭屏状态下长时间进行网络定位。

7. 除用户可感知的业务外，禁止应用 WLAN 在后台长时间处于扫描状态。

8. 除用户可感知的业务外，禁止应用在灭屏时后台频繁收发数据。

9. 除用户可感知的业务外，禁止应用运行不必要的后台服务，后台服务会被系统管控和约束。

3 智慧屏

3.1 概述

基于 HarmonyOS，开发者可以开发智慧屏应用，提供丰富的分布式体验。应用可以通过 HarmonyOS 的 API 实现摄像头拍摄、多模输入、分布式应用等能力，典型场景包括：

- 摄像头拍摄：示例参见相机。
- 多模输入：示例参见多模输入。
- 分布式应用：示例参见分布式任务调度。

约束与限制

- 智慧屏是依靠遥控器操作的设备，在智慧屏上应当始终存在一个焦点，告知用户当前可操作的位置。当焦点变化时，应当有明显的动效反馈。
- 智慧屏应用的交互设计必须按照表 1 中的要求来响应遥控器的操作。其中，“-”表示长按无特殊的功能设置，效果仅相当于连续多次点击该按键。

按键	点击	长按
方向键/滑动触摸板	移动控制焦点。	-
确认键	进入当前获焦内容。	-

按键	点击	长按
返回键	返回上一级。	-
首页键	返回首页。	调出多任务。
音量键	调节音量大小。	-
菜单键	调出对应界面的功能菜单。	调出控制中心。
开关机键	打开或关闭智慧屏。	-

表 1 标准遥控器按键规范