车机应用设计文档

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。

驾驶安全管控

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。

```
1. if (!isDrivingSafety(context)) { // 判断应用当前状态是否是驾驶安全的
2.     int restraintCode = DrivingSafetyManager.getRestraint(); // 获取当前系统限制条件
3.     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_N
    ORMAL);
20.         }
21.         // 其他限制
22.         ...
23. }
```

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。

```
1. // 构造查询结果对象
2. DrivingSafetyConfigResult result = new DrivingSafetyConfigResult();
3. // 调查询能力接口
4. try{
5.     int errorCode = DrivingSafetyConfig.getSysDrivingSafetyConfigure(SysDrivingSafetyControlItems.DM_IME, result);
6.     if (errorCode != 0) {
7.         HiLog.error("Get DrivingSafetyConfig Error: %d", errorCode));
8.         return;
```

```
9.          }
10.         Boolean isOpen = false;
11.         if (!result.isOpen()){ // 当前输入法策略为非管控状态
12.             isOpen = true; // 修改当前输入法策略为管控状态
13.         }
14.         // 调用修改管控能力接口，修改管控策略
15.         errorCode =
    DrivingSafetyConfig.setSysDrivingSafetyConfigure(SysDrivingSafety
    ControlItems.DM_IME, isOpen);
16.         if (errorCode != 0) {
17.             HiLog.error("Set DrivingSafetyConfigre Error: %d",
    errorCode);
18.             return;
19.         }
20. } catch (RemoteException | IllegalArgumentException e) {
21.         HiLog.error("System Error: %s", e.getMessage())
22.         return;
23. }
```

车辆控制

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。

接口名	描述
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。

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。

接口名	描述
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。
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。

接口名	描述

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。
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。

接口名	描述
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
7.          public void onServiceDisconnected(VehicleServiceName
    serviceName) {
8.          }
9.  };
10. // 连接指定车机专有硬件服务
11. try {
12.
    Vehicle.connect(VehicleServiceName.VEHICLECONTROL_SERVICE,
    listener);
13.         Thread.sleep(2000);
14.         return true;
15. } catch (IllegalStateException | InterruptedException e) {
16.         Logger.info("Exception:" + e.toString());
17.         return false;
18. }
```

2. 根据不同管理入口类，调用对应接口。

```
1.  // VehicleCabinManager 类，座舱天窗管理
2.  String propId =
    VehicleCabinManager.ID_CABIN_SUNROOF_SWITCH;
3.  int zoneId = VehicleZone.ZONE_NONE;
4.  String value = "Inactive";
```

```
5.  VehicleActuatorCallback callback = new VehicleActuatorCallback() {
6.      @Override
7.      public void onErrorActuator(String propId, int zoneId, int
    outResult) {
8.      }
9.  };
10. boolean result = false;
11. try {
12.     VehicleCabinManager.setVehicleActuator(propId, zoneId,
    callback, value);
13.     result = true;
14. } catch (RemoteException | IllegalArgumentException e) {
15.     result = false;
16. }
17. if (!result) {
18.     System.out.println(String.format("Set sunroof error: %d",
    result));
19. }
20.
21. // VehicleBodyManager 类，获取车身前挡风玻璃雨刷器状态
22. zoneId = VehicleZone.ZONE_FRONT;
```

```
23. 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_PERFO
    RMANCEMODE;
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();
```

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。

```
1. // 设置辅助输入信号值
2. String propId = "OEM_Status_DTCCountTest";
3. int zoneId = VehicleZone.ZONE_NONE;
```

```
4.  Boolean value = true;
5.  VehicleActuatorCallback callback = new VehicleActuatorCallback() {
6.          @Override
7.          public void onErrorActuator(String propId, int zoneId, int outResult) {
8.          }
9.  };
10. bool result = true;
11. try {
12.         VehicleVendorExtensionManager.setVehicleActuator(propId, zoneId,
    callback, value);
13. } catch(RemoteException | IllegalArgumentException e) {
14.         result = false;
15. }
16. if(!result) {
17.         System.out.println(String.format("Set transmiss performance mode
    error: %d", result));
18. }
```

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。

接口名	描述

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。

```
1.  // 设置 TBOX 属性值
2.  String incorrectPath = TBoxManager.ID_TBOX_BCALL_STATUS;
3.  byte[] result = null;
4.  TBoxPropertyManager manager = new TBoxPropertyManager();
5.  boolean isTrue = false;
6.  try {
7.      result = manager.getBuffer(tboxPropPath);
8.      isTrue = true;
9.  } catch (RemoteException | IllegalArgumentException |
    UnsupportedOperationException e) {
10.     isTrue = false;
11. }
```

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。
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。

接口名	描述
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;
```

```
12. } catch (RemoteException | IllegalArgumentException |
    UnsupportedOperationException e) {
13.         result = false;
14. }
```

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。
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

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。

接口名	描述
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。
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。

```
1. // DrivingAssistManager 类使用
2. boolean result = false;
3. try {
4.         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;
}
```

打造车载系统应用

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": [
    {
```

```
3.         "name": "ohos.permission.vehicle.READ_VEHICLE_HMI_INFO",
4.         "reason": "",
5.         "usedScene": {
6.           "ability": [
7.             ".MainAbility"
8.           ],
9.           "when": "inuse"
10.       }
11.    }
12. ]
```

2. 添加支持驾驶模式标签。

```
1.  "abilities": {
2.     "name": ".carlink",
3.     "icon": "$carlink:icon",
4.     "label": "carlink",
5.     "supported-modes": ["drive"],
6.  }
```

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。

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"
```

```
17.                 ohos:image_src="$media:default.png"/>
18.         <Image ohos:id="$+id:music_heart_btn"
19.                 ohos:width="24vp"
20.                 ohos:height="24vp"
21.                 ohos:left_of="$+id:music_hiplay_btn"
22.                 ohos:image_src="$media:default.png"/>
23.         <Image ohos:id="$+id:music_hiplay_btn"
24.                 ohos:width="24vp"
25.                 ohos:height="24vp"
26.                 ohos:left_margin="16vp"
27.                 ohos:align_parent_right="$+id:title_bar"
28.                 ohos:image_src="$media:default.png"/>
29.     </AdaptiveBoxLayout>
30.     <DirectionalLayout ohos:id="$+id:cover_container"
31.                         ohos:width="-1"
32.                         ohos:height="-2"
33.                         ohos:weight="1"
34.                         ohos:orientation="1">
35.         <AdaptiveBoxLayout
36.                 ohos:id="$+id:music_cover_adapt"
37.                 ohos:width="-1"
38.                 ohos:height="-1">
```

```
39.             <DirectionalLayout
40.                 ohos:id="$+id:music_cover_wrap1"
41.                 ohos:width="-2"
42.                 ohos:height="-2"
43.                 ohos:padding="20vp"
44.                 ohos:orientation="1">
45.                 <Image ohos:id="$+id:music_cover"
46.                     ohos:width="300vp"
47.                     ohos:height="300vp"
48.                     ohos:layout_alignment="17"
49.                     ohos:image_src="$media:default.png"/>
50.             </DirectionalLayout>
51.             <DirectionalLayout
52.                 ohos:id="$+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"
```

```
                ohos:height="-2"
                ohos:layout_alignment="17"
                ohos:text="See the lights see the party the
    ball grows"
                ohos:text_size="13vp"
                ohos:text_color="#FF000000"
                ohos:text_alignment="72"/>
            </DirectionalLayout>
        </AdaptiveBoxLayout>
    </DirectionalLayout>

    <DirectionalLayout ohos:id="$+id:foot_wrap"
                ohos:width="-1"
                ohos:height="-2"
                ohos:orientation="1">
        <DirectionalLayout ohos:id="$+id:progress_container"
                ohos:width="-1"
                ohos:height="-2"
                ohos:top_margin="10vp"
                ohos:orientation="0">
            <Text ohos:id="$+id:play_progress_time"
                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"
166.                                    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();

        isSuccess = player.rewindTo(startMilliSecond * MICRO_MILLI_RATE, REWIND_NEXT_SYNC);
        // 播放
        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().a
    ddComponent(mySurfaceProvider);
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。

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