Rockchip_Developer_Guide_Automotive_Audio_CN

文件标识：RK-KF-YF-784

发布版本：V1.0.0

日期：2024-01-18

文件密级：□绝密 □秘密 □内部资料 ■公开

免责声明

本文档按"现状"提供，瑞芯微电子股份有限公司（"本公司"，下同）不对本文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。本文档仅作为使用指导的参考。

由于产品版本升级或其他原因，本文档将可能在未经任何通知的情况下，不定期进行更新或修改。

商标声明

"Rockchip"、"瑞芯微"、"瑞芯"均为本公司的注册商标，归本公司所有。

本文档可能提及的其他所有注册商标或商标，由其各自拥有者所有。

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址： 福建省福州市铜盘路软件园A区18号

网址： www.rock-chips.com

客户服务电话： +86-4007-700-590

客户服务传真： +86-591-83951833

客户服务邮箱： fae@rock-chips.com

前言

概述

本文档主要介绍 RockChip AAOS Audio模块 音区配置、音量调节、焦点管理、AudioControl等内容

产品版本

芯片名称	内核版本
RK3588/RK3576	5.10/6.10

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
V0.1.0	黄晓明/郑志棋	2024-01-18	初始版本
V0.2.0	黄晓明	2024-05-31	添加Android14相关功能介绍

目录

基本概念

AAOS 管理来自 Android 应用的声音，同时控制这些应用，并根据其声音类型将声音路由到 HAL 中的输出设备。

- 逻辑声音流

使用音频属性进行标记，应用可以有一个或多个通过标准 Android API（如用于控制焦点的AudioManager或用于在线播放的 MediaPlayer）发出一个或多个音频数据逻辑声音流。应用声音流与AudioAttributes相关联。逻辑声音流通过 AudioService 发送，并路由到一个（并且只有一个）可用的物理输出声音流。

- 输出设备

在音频 HAL 级别，设备类型 AUDIO_DEVICE_OUT_BUS 提供用于车载音频系统的通用输出设备。

- 详细信息可参考 https://source.android.com/docs/automotive/audio/audio-policy-configuration?hl=zh-cn

音区配置

音区是AAOS中管理的独立音频区域的概念，每个音区拥有独立的焦点管理、音量控制，同时对应到车载各个区域当中。

- 开启动态音区管理

  在device 对应的产品目录overlay下添加开启动态路由开关

  ```
  <bool name="audioUseDynamicRouting">true</bool>
  ```

- 音频区域配置

  音频区域可通过修改device对应的产品目录下的car_audio_configuration.xml 和 audio_policy_configuration.xml进行配置。

  - car_audio_configuration.xml 配置

    默认的产品的xml配置4个音频区分别和4个区域的用户对应。

    以下以主音频区为例，占用区域ID为0。定义了3个音量组，同一音量组的设备有相同的音频增益。第一个音量组定义了两个输出设备bus0_media_out和bus6_notification_out，其中bus0_media_out的输出设备对应的音频场景是music和announcement。
    第二、第三、第四以此类推。 可以根据需要删减音量组、区域。

    注意点：

    1. 主音频区的 audioZoneId 始终为零。
    2. audioZoneId 和 occupantZoneId 的编号不可重复。其中occupantZoneId是用来映射 UserID 和DisplayID。 比如car_audio_configuration.xml定义了3个occupantZoneId 分别为0,1,2。就定义了三个音频区相关的配置。
    3. audioZoneId 和 occupantZoneId 之间只能是一对一的映射。
    4. 所有的context都必须分配一个device address，并且只能分配给一个device address。

```
<carAudioConfiguration version="2">
    <zones>
        <zone name="primary zone" isPrimary="true" occupantZoneId="0">
            <volumeGroups>
                <group>
                    <device address="bus0_media_out">
                        <context context="music"/>
                        <context context="announcement"/>
                    </device>
                    <device address="bus6_notification_out">
                        <context context="notification"/>
                        <context context="system_sound"/>
                        <context context="emergency"/>
                        <context context="safety"/>
                        <context context="vehicle_status"/>
                        <context context="alarm"/>
                    </device>
                </group>
                <group>
                    <device address="bus1_navigation_out">
                        <context context="navigation"/>
                    </device>
                    <device address="bus2_voice_command_out">
                        <context context="voice_command"/>
                    </device>
                </group>
                <group>
                    <device address="bus4_call_out">
                        <context context="call"/>
                        <context context="call_ring"/>
                    </device>
                </group>
            </volumeGroups>
        </zone>
    </zones>
</carAudioConfiguration>
```

- audio_policy_configuration.xml配置

  配置好上层service 使用的xml后，还需要配置框架中流和设备的信息，需要添加attchedDevices、mixport、deviceport以及mixport和deviceport之间的routes。如下配置主音频区域所需的设备、流端口、设备端口以及之间的路由。

```
<modules>
      <!-- Primary Audio HAL -->
      <module name="primary" halVersion="3.0">
          <attachedDevices>
              <!-- One bus per context -->
              <item>bus0_media_out</item>
              <item>bus1_navigation_out</item>
              <item>bus2_voice_command_out</item>
              <item>bus4_call_out</item>
              <item>bus6_notification_out</item>
          </attachedDevices>
          <defaultOutputDevice>bus0_media_out</defaultOutputDevice>
          <mixPorts>
              <mixPort name="mixport_bus0_media_out" role="source"
                       flags="AUDIO_OUTPUT_FLAG_PRIMARY">
                  <profile name="" format="AUDIO_FORMAT_PCM_16_BIT"
                           samplingRates="48000"
                           channelMasks="AUDIO_CHANNEL_OUT_STEREO"/>
              </mixPort>
              <mixPort name="mixport_bus1_navigation_out" role="source">
                  <profile name="" format="AUDIO_FORMAT_PCM_16_BIT"
                           samplingRates="48000"
                           channelMasks="AUDIO_CHANNEL_OUT_STEREO"/>
              </mixPort>
              <mixPort name="mixport_bus2_voice_command_out" role="source">
                  <profile name="" format="AUDIO_FORMAT_PCM_16_BIT"
                           samplingRates="48000"
                           channelMasks="AUDIO_CHANNEL_OUT_STEREO"/>
              </mixPort>
              <mixPort name="mixport_bus4_call_out" role="source">
                  <profile name="" format="AUDIO_FORMAT_PCM_16_BIT"
                           samplingRates="48000"
                           channelMasks="AUDIO_CHANNEL_OUT_STREO"/>
              </mixPort
              <mixPort name="mixport_bus6_notification_out" role="source">
                  <profile name="" format="AUDIO_FORMAT_PCM_16_BIT"
                           samplingRates="48000"
                           channelMasks="AUDIO_CHANNEL_OUT_STEREO"/>
              </mixPort>
          </mixPorts>
          <devicePorts>
              <devicePort tagName="bus0_media_out" role="sink"
type="AUDIO_DEVICE_OUT_BUS"
                          address="bus0_media_out">
                  <profile name="" format="AUDIO_FORMAT_PCM_16_BIT"
                           samplingRates="48000"
channelMasks="AUDIO_CHANNEL_OUT_STEREO"/>
                  <gains>
                      <gain name="" mode="AUDIO_GAIN_MODE_JOINT"
                            minValueMB="-3200" maxValueMB="600"
                            defaultValueMB="0" stepValueMB="100"/>
```

```
                </gains>
            </devicePort>
            <devicePort tagName="bus1_navigation_out" role="sink"
type="AUDIO_DEVICE_OUT_BUS"
                    address="bus1_navigation_out">
                <profile name="" format="AUDIO_FORMAT_PCM_16_BIT"
                         samplingRates="48000"
channelMasks="AUDIO_CHANNEL_OUT_STEREO"/>
                <gains>
                    <gain name="" mode="AUDIO_GAIN_MODE_JOINT"
                          minValueMB="-3200" maxValueMB="600"
                          defaultValueMB="0" stepValueMB="100"/>
                </gains>
            </devicePort>
            <devicePort tagName="bus2_voice_command_out" role="sink"
type="AUDIO_DEVICE_OUT_BUS"
                        address="bus2_voice_command_out">
                <profile name="" format="AUDIO_FORMAT_PCM_16_BIT"
                         samplingRates="48000"
channelMasks="AUDIO_CHANNEL_OUT_STEREO"/>
                <gains>
                    <gain name="" mode="AUDIO_GAIN_MODE_JOINT"
                          minValueMB="-3200" maxValueMB="600"
                          defaultValueMB="0" stepValueMB="100"/>
                </gains>
            </devicePort>
            <devicePort tagName="bus4_call_out" role="sink"
type="AUDIO_DEVICE_OUT_BUS"
                       address="bus4_call_out">
                <profile name="" format="AUDIO_FORMAT_PCM_16_BIT"
                         samplingRates="48000"
channelMasks="AUDIO_CHANNEL_OUT_STEREO"/>
                <gains>
                    <gain name="" mode="AUDIO_GAIN_MODE_JOINT"
                          minValueMB="-3200" maxValueMB="600"
                          defaultValueMB="0" stepValueMB="100"/>
                </gains>
            </devicePort>
            <devicePort tagName="bus6_notification_out" role="sink"
type="AUDIO_DEVICE_OUT_BUS"
                       address="bus6_notification_out">
                <profile name="" format="AUDIO_FORMAT_PCM_16_BIT"
                         samplingRates="48000"
channelMasks="AUDIO_CHANNEL_OUT_STEREO"/>
                <gains>
                    <gain name="" mode="AUDIO_GAIN_MODE_JOINT"
                          minValueMB="-3200" maxValueMB="600"
                          defaultValueMB="0" stepValueMB="100"/>
                </gains>
            </devicePort>
        </devicePorts>
        <!-- route declaration, i.e. list all available sources for a
given sink -->
        <routes>
            <route type="mix" sink="bus0_media_out"
sources="mixport_bus0_media_out"/>
            <route type="mix" sink="bus1_navigation_out"
sources="mixport_bus1_navigation_out"/>
```

```
                <route type="mix" sink="bus2_voice_command_out"
                       sources="mixport_bus2_voice_command_out"/>
                <route type="mix" sink="bus4_call_out"
sources="mixport_bus4_call_out"/>
                <route type="mix" sink="bus6_notification_out"
                       sources="mixport_bus6_notification_out"/>
           </routes>
</module>
```

- 乘客音频投放到主音频区（**Android14支持**）

  将乘客区的context 为Music的音频数据通过主屏区的设备进行播放。应用通过乘客区的userid 传递CarAudioManager提供的api allowMediaAudioOnPrimaryZone(long requestId, boolean allowed)中可实现相应的功能。

- 动态音频区配置（**Android14支持**）

  在同一个音频区域中 同一个context有两个输出的bus address，可以在播放过程中进行切换。音频区动态配置需要修改车载音频配置

  - 如下给zone id为1的音频区域配置 zone 1 config 0和 zone 1 config 1。

```
<carAudioConfiguration version="3">
  <zones>
      <zone name="Zone1" audioZoneId="1" occupantZoneId="1">
          <zoneConfigs>
              <zoneConfig name="Zone 1 Config 0" isDefault="true">
                  <volumeGroups>
                      <group>
                          <device address="bus_100">
                              <context context="music"/>
                                  ***
                          </device>
                      </group>
                  </volumeGroups>
              </zoneConfig>
              <zoneConfig name="Zone 1 Config 1">
                  <volumeGroups>
                      <group>
                          <device address="bus_101">
                              <context context="music"/>
                                  ***
                          </device>
                      </group>
                  </volumeGroups>
              </zoneConfig>
          </zoneConfigs>
      </zone>
</zones>
```

  注意点：

  1. 两个zoneConfig的地址不一样, 只能添加到非primary zone的区域。
  2. 添加的device address 在audio_policy_configuraion.xml需要定义。

- 音频区域镜像（**Android14支持**）

音频镜像功能使乘客能够共享音频。开启音频镜像能够将两个或两个以上的设备共享相同的音频数据。
如下车载音频配置定义了一个address 为bus_1000的mirror 设备，在外部应用调用enableMirrorForAudioZones传递mirror的zoneID 队列到系统。车载系统会将zoneID对应的所有bus address都输出相同的音频数据。

```
<carAudioConfiguration version="3">
   <mirroringDevices>
       <mirroringDevice address="bus_1000"/>
   </mirroringDevices>
  <zones>
    ....
  </zones>
</carAudioConfiguration>
```

音频焦点

- 音频焦点基本概念

  音频焦点的目标是确保在多个应用程序同时播放音频时，用户能够顺利地听到他们想要听的声音，并防止多个应用同时混合输出声音，造成用户困扰。也就是每次要去播放某个声音的时候 先去请求焦点，请求到焦点 后才能开始播放。

- 应用获取音频焦点的api

  android 的audioManager 提供了requestAudioFocus 的接口来获取焦点。

  - AAOS焦点请求处理
    CarAudioService根据传递进来的请求参数 和目前持有的场景来决定 是否让这个焦点请求成功。 这个具体的规则是基于交互矩阵来实现的。系统会根据请求的 `CarAudioContext` 和当前焦点持有者的 CarAudioContext 之间的预定义交互来处理音频焦点请求。交互类型分以下三种：独占、拒绝和并发。交互矩阵可以参考https://source.android.com/docs/devices/automotive/audio/audio-focus?hl=zh-cn

- car_audio_configuration.xml中配置audioUseHalDuckingSignals为ture，那么焦点的焦点类型为AUDIOFOCUS_GAIN_TRANSIENT_MAY_DUCK会需要调用audioControl hal的onDevicesToDuckChange来降低音量。 如果设置为false，那么duck的事件不会通知hal层， 上层也不会进行任何处理。

音量调节

音量调节

- 外部应用应使用CarAudioManger提供的相关api来获取和设置音量。
- 在device 对应的产品目录overlay下配置config_useFixedVolume为true，使用外部的硬件放大器进行音量的调节car_audio_configuration.xml中定义为同一个volumeGroup的音量会同步调节。
- 在device 对应的产品目录overlay下配置config_handleVolumeKeysInWindowManager 为true，按键处理不经过android框架而是回调到CarService进行处理。
- 在device 对应的产品目录overlay下配置audioUseCarVolumeGroupMuting为true（建议值），用于启用各个音量组的静音功能，可以单独将每个音量组静音。如果设为 false（默认值），系统会停用单个音量组的静音功能，静音功能会切换为主静音。
- 按键和ui 音量处理会回调到carAudioService而不经过android的系统音频框架进行处理， 其中按键只能控制主屏区域的音量。

PCM 插件

PCM插件扩展了PCM设备的功能和特性。这些插件负责各种采样率转换、通道之间复制等等。这些插件是通过配置vendor/etc/asound.conf配置文件来控制。asound.conf 格式可以参考ALSA project - the C library reference: PCM (digital audio) plugins (alsa-project.org)

- dshare plugin

  这个插件允许与多个的客户端共享多个通道，每个通道的访问是独一性的。与share插件不同，这个插件不需要显式的服务器程序，而是像dmix和dsnoop插件一样从每个客户端并发访问共享缓冲区。下面为插件支持的参数配置。

```
pcm.name {
    type dshare     # Direct sharing
    ipc_perm INT        # IPC permissions (octal, default 0600)
    slave STR
    # or
    slave {         # Slave definition
        pcm STR     # slave PCM name
        # or
        pcm { }     # slave PCM definition
        format STR  # format definition
        rate INT    # rate definition
        channels INT
        period_time INT # in usec
        # or
        period_size INT # in frames
        buffer_time INT # in usec
        # or
        buffer_size INT # in frames
        periods INT # when buffer_size or buffer_time is not specified
    }
    bindings {      # note: this is client independent!!!
        N INT       # maps slave channel to client channel N
    }
}
```

- dsnoop plugin

  这个插件将一个多通道捕获流拆分给多个客户端，每个通道的访问是独一性的。下面为插件支持的参数配置。

```
pcm.name {
    type dsnoop     # Direct snoop
    slave STR
    # or
    slave {         # Slave definition
        pcm STR     # slave PCM name
        # or
        pcm { }     # slave PCM definition
        format STR  # format definition
        rate INT    # rate definition
        channels INT
        period_time INT # in usec
        # or
        period_size INT # in frames
        buffer_time INT # in usec
        # or
        buffer_size INT # in frames
```

```
        periods INT # when buffer_size or buffer_time is not specified
    }
    bindings {      # note: this is client independent!!!
        N INT       # maps slave channel to client channel N
    }
}
```

Audio HAL

Android 管理来自应用程序的声音，控制这些应用程序并根据声音类型将它们的声音路由到HAL。在HAL上选择对应的输出设备，并将他们路由到合适的扬声器。

- 输出设备

  在Audio HAL层， 设备类型AUDIO_DEVICE_OUT_BUS提供用于车载音频系统的通用输出设备。AUDIO_DEVICE_OUT_BUS类型设备支持可寻址端口（其中每个端口都是一个物理设备的端口）。AUDIO_DEVICE_OUT_BUS 是目前预计车载唯一支持的输出设备类型。AUDIO_DEVICE_OUT_BUS类型，通过start_output_stream传下来， 而BUS address 是在开机时，调用open_output_stream打开stream时固定的。

```
if (out->devices[i] == AUDIO_DEVICE_OUT_SPEAKER ||
  out->devices[i] == AUDIO_DEVICE_OUT_WIRED_HEADSET ||
  out->devices[i] == AUDIO_DEVICE_OUT_WIRED_HEADPHONE ||
  out->devices[i] == AUDIO_DEVICE_OUT_BUS) {
  audio_devices_t route_device = out->devices[i];
  route_pcm_card_open(adev->dev_out[SND_OUT_SOUND_CARD_SPEAKER].card,
getRouteFromDevice(route_device));
  if (out->address == NULL) {
      card = adev->dev_out[SND_OUT_SOUND_CARD_SPEAKER].card;
      device = adev->dev_out[SND_OUT_SOUND_CARD_SPEAKER].device;
  } else {
      if (strcmp(out->address, "bus0_media_out") == 0) {
          card = 103;
          device = 0;
      } else if (strcmp(out->address, "bus1_navigation_out") == 0) {
           card = 103;
           device = 1;
      } else if (strcmp(out->address, "bus2_voice_command_out") == 0) {
           card = 103;
           device = 2;
      } else if (strcmp(out->address, "bus4_call_out") == 0) {
           card = 103;
           device = 3;
      } else if (strcmp(out->address, "bus6_notification_out") == 0) {
           card = 103;
           device = 4;
      } else if (strcmp(out->address, "bus100_audio_zone_1") == 0) {
          card = 103;
          device = 5;
      } else if (strcmp(out->address, "bus200_audio_zone_2") == 0) {
          card = 103;
          device = 6;
      } else if (strcmp(out->address, "bus300_audio_zone_3") == 0) {
          card = 103;
          device = 7;
      } else {
          card = 103;
```

```
            device = 0;
        }
    }
}
```

- 输入设备

  在录制音频时，Audio HAL层会收到openInputStream调用， 其中包含指示如何处理录制音频流的AudioSource参数。

- 注意点

1. 在asound.conf 中定义的声卡和设备的ID 需要 跟hal中配置的声卡 和设备 ID 一致。比如：分配给 bus0_media_out 的cardID 为 103 ，device 为 0。相对应的asound.conf中需要定义一个 pcmC103D0p的pcm节点。
2. 目前规定使用dshare/dsnoop插件声卡的编号需要从100开始。

AudioControl HAL

AudioControl HAL 是AAOS 系统中引入的车载音频硬件控制的一个hal。车载音频服务会与音频控制 HAL 进行通信，外部车载音频硬件如仪表、收音机等的声音状态会通过HAL 通知到车载音频服务，AAOS的声音的状态也可以通过HAL与外部音频硬件交互。

- 音量控制 (**Android12 支持**)

  Android12 车载音频服务 会通过AudioControl HAL 对 车载的外部音频硬件进行音量的控制。Android 14 暂未支持。其余音频闪避、平衡等功能暂未支持。

AAudio

- 基本概念

  AAudio 是 Android 8.0 及以上版本中引入的一种音频 API，提供了一个低延迟数据路径，在与支持 MMAP 的 HAL 和驱动程序结合使用时，可缩短音频播放和录制的延迟时间。

  AAudio可以分为shared模式和 exclusive 模式

  在 exclusive 模式下，使用该功能可将客户端应用代码直接写入与 ALSA 驱动程序共享的内存映射缓冲区。在 shared 模式下，MMAP 缓冲区由 AudioServer 中运行的混音器使用。在 exclusive 模式下，由于数据会绕过混音器，延迟时间会明显缩短。

- 配置添加

  默认sdk 中没有开启aaudio mmap的功能。

  1. 产品目录的build.prop 添加开启 AAudio的功能

```
PRODUCT_PROPERTY_OVERRIDES += aaudio.mmap_policy=2
PRODUCT_PROPERTY_OVERRIDES += aaudio.mmap_exclusive_policy=2
```

  可以根据情况设置值为1、2、3中的一个，建议设置为2。

  - 1 = AAUDIO_POLICY_NEVER - 仅使用旧的路径，不使用 MMAP。
  - 2 = AAUDIO_POLICY_AUTO - 尝试使用 MMAP。操作失败或 MMAP 不可用，则使用旧路径。
  - 3 = AAUDIO_POLICY_ALWAYS - 仅使用 MMAP 路径。不回退到旧路径。

  2. 配置AAudio 输出的声卡和设备

  如下属性配置用于输出的AAudio路径的声卡。没有配置则sdk默认输出到声卡1 的设备0。

```
PRODUCT_PROPERTY_OVERRIDES += aaudio.mmap_card=card:0dev:0
```

3. audio_policy_configuration.xml 添加mmap的流端口

如下所示添加mmap_no_irq_out 和 mmap_no_irq_in两个mixport，并和对应的deviceport建立起routes。
其中xml中channelMasks、dts中配置的TDM 通道数以及hal中配置给kernel的通道数需要保持一致。sdk默认配置为8通道。

```
<mixPort name="mmap_no_irq_out" role="source"
flags="AUDIO_OUTPUT_FLAG_DIRECT|AUDIO_OUTPUT_FLAG_MMAP_NOIRQ">
 <profile name="" format="AUDIO_FORMAT_PCM_16_BIT"
          samplingRates="48000"
channelMasks="AUDIO_CHANNEL_OUT_STEREO,AUDIO_CHANNEL_INDEX_MASK_8"/>
</mixPort>
<mixPort name="mmap_no_irq_in" role="sink"
flags="AUDIO_INPUT_FLAG_MMAP_NOIRQ">
 <profile name="" format="AUDIO_FORMAT_PCM_16_BIT"

samplingRates="8000,11025,12000,16000,22050,24000,32000,44100,48000"

channelMasks="AUDIO_CHANNEL_IN_MONO,AUDIO_CHANNEL_IN_STEREO,AUDIO_CHANNEL_IN
_FRONT_BACK"/>
</mixPort>
<routes>
 <route type="mix" sink="bus0_media_out"
sources="mixport_bus0_media_out,mmap_no_irq_out"/>
 <route type="mix" sink="mmap_no_irq_in"
    sources="Built-In Mic,Built-In Back Mic,Echo-Reference Mic"/>
</routes>
```