How to change audio device for Android
How to change audio device for Android

This article describes the integration steps required to connect your own audio device (inputs/outputs). It deals mainly with the Android impact of integrating the new audio solution. It is intended for Distribution Package users.

Contents

1 Prerequisites ....................................................................................................................................................................... 2
2 Android audio overview .................................................................................................................................................... 3
3 Integrating ........................................................................................................................................................................... 3

3.1 Integrating within the Linux kernel .............................................................................................................................. 4
  3.1.1 Compile the Linux audio driver .............................................................................................................................. 4
  3.1.2 Update the Linux device tree .................................................................................................................................... 4
3.2 Integrating within Android ............................................................................................................................................. 4
  3.2.1 Add Android permissions ........................................................................................................................................ 4
  3.2.2 Add the Android audio interface (Hardware Abstraction Layer) ........................................................................... 5
  3.2.3 Configure the Android service .................................................................................................................................. 6
    3.2.3.1 Audio policy configuration ................................................................................................................................... 7
    3.2.3.2 Audio policy customization .................................................................................................................................... 7
      3.2.3.2.1 Configurable AudioPolicyManager .................................................................................................................. 7
      3.2.3.2.2 Proprietary AudioPolicyManager ................................................................................................................... 8
3.3 Validating ......................................................................................................................................................................... 8
  3.3.1 Validate the Kernel Integration .................................................................................................................................. 8
  3.3.2 Validate the Android Integration ................................................................................................................................ 9
4 References ......................................................................................................................................................................... 10

1 Prerequisites

The environment must be installed using the Distribution Package adapted to your selected microprocessor device. See the list of Android Distribution Package.

The audio device (ex: Wolfson WM8994) must be already connected to the microprocessor device within your board.
2 Android audio overview

The Android audio software is structured in several layers:

- **Audio Drivers (hardware dependent)** interfaces providing a control and access to audio inputs or outputs
- **Audio Interface** (audio@4-0, audio-effect@4.0) hardware abstraction layers providing a standard way to configure the underlaying driver and audio-related functionality such as audio sources or sinks (as the Bluetooth stack for A2DP), based on the Android HIDL (Hardware Interface Definition Language) mechanism
- **Audio Server** (audioserver) native audio services managing routing, effects, mixing...
- **Audio JNI** (Java Native Interface) interfaces used to access the native audio services
- **Audio System service** several Android services useful for applications

3 Integrating

The audio solution must be integrated in three main steps, explained in the next chapters:

- Integrating within the Linux kernel
- Integrating within Android
- Validating
3.1 Integrating within the Linux kernel

The audio integration in the Linux kernel is performed in two steps:

- Add the audio driver within the compilation process
- Update the device tree

3.1.1 Compile the Linux audio driver

The audio subsystem is accessed through a dedicated driver (ALSA).

The Linux kernel audio driver documentation is available in the kernel sources, please refer to How to build kernel for Android for more information.

By default, audio is enabled within the kernel, adding the following lines within the android-soc.config available in kernel sources.

```
CONFIG_SND=y
CONFIG_SND_TIMER=y
CONFIG_SND_PCM=y
...
```

Two options:

- The audio driver is part of the Linux kernel source select its config (e.g.: CONFIG_SND_SOC_WM8994=y for the Wolfson WM8994 audio codec). Refer to Updating the kernel configuration for more information.
- The audio driver is provided separately add it in the build_kernel.sh script

3.1.2 Update the Linux device tree

The audio device tree documentation is available in the kernel sources.

Depending on the driver selected, it can be required to update the device tree (refer to Changing the Device Tree for more information).

3.2 Integrating within Android

The audio device integration within Android is performed in several steps:

- Add permissions to allow starting the required Android services
- Add the audio interface (Hardware Abstraction Layer)
- Configure the audio services (useful for audio policy)

3.2.1 Add Android permissions

Three permissions can be added to ensure that the audio services are correctly started:

- android.hardware.audio.output.xml for the audio track service
- android.hardware.audio.low_latency.xml for the audio low latency service ensure the use of a low latency audio card (not implemented by default)
- android.hardware.audio.pro.xml for the audio pro service ensure the use of a low latency audio card and audio high quality (not implemented by default)

For that purpose, it is required to add the following lines within the device.mk file:
How to change audio device for Android

```
PRODUCT_COPY_FILES += \
frameworks/native/data/etc/android.hardware.audio.output.xml:$(TARGET_COPY_OUT_VENDOR) \\
/etc/permissions/android.hardware.audio.output.xml \\
frameworks/native/data/etc/android.hardware.audio.low_latency.xml:$(TARGET_COPY_OUT_VENDOR)/etc/permissions/android.hardware.audio.low_latency.xml \\
frameworks/native/data/etc/android.hardware.audio.pro.xml:$(TARGET_COPY_OUT_VENDOR)/etc/permissions/android.hardware.audio.pro.xml \\
```

Note: the feature `android.hardware.microphone`, useful to start `AudioRecord`, is added by default within the different device core configurations (e.g.: `wearable_core_hardware.xml` for wearable devices)

### 3.2.2 Add the Android audio interface (Hardware Abstraction Layer)

The audio interface is defined in `hardware/interfaces/audio/`

The audio interface implementation is based on audio `<device>` so HAL libraries with the following devices:

- Audio primary device (used to manage built-in audio inputs and outputs) STMicroelectronics implementation available in `device/stm/<STM32Series>/peripheral/audio`
- Audio usb device (used to manage USB audio inputs and outputs) default implementation available in `hardware/libhardware/modules/usbaudio/`
- Audio remote-submix device (used to loop audio signal between input and output) default implementation available in `hardware/libhardware/modules/audio_remote_submix`
- Audio a2dp device (used to manage Bluetooth A2DP audio inputs and outputs) default implementation available in `system/bt/audio_a2dp_hw`

To select the required audio HAL (adapted to your requirements), first update the `device.mk` with the following information (STMicroelectronics implementation case):

```
TARGET_USB_AUDIO_HAL := default
TARGET_PRIMARY_AUDIO_HAL := stm
TARGET_REMOTE_SUBMIX_AUDIO_HAL := default
TARGET_A2DP_AUDIO_HAL := default

PRODUCT_PACKAGES += \ 
libtinyalsa \ 
tinyplay \ 
tinycap \ 
tinymix \ 
tinypcminfo \ 
libaudiohalcm \\

PRODUCT_PACKAGES += \ 
audio.usb.$(TARGET_USB_AUDIO_HAL) \ 
audio.primary.$(TARGET_PRIMARY_AUDIO_HAL) \ 
audio.r_submix.$(TARGET_REMOTE_SUBMIX_AUDIO_HAL) \ 
audio.a2dp.$(TARGET_A2DP_AUDIO_HAL) \\

PRODUCT_PROPERTY_OVERRIDES += \ 
ro.hardware.audio.usb=$(TARGET_USB_AUDIO_HAL) \ 
ro.hardware.audio.primary=$(TARGET_PRIMARY_AUDIO_HAL) \ 
ro.hardware.audio.r_submix=$(TARGET_REMOTE_SUBMIX_AUDIO_HAL) \ 
ro.hardware.audio.a2dp=$(TARGET_A2DP_AUDIO_HAL) \\

PRODUCT_PACKAGES += \ 
android.hardware.audio@2.0-service \ 
android.hardware.audio@4.0-impl \ 
android.hardware.audio.effect@4.0-impl \ 
android.hardware.soundtrigger@2.0-impl \ 
libeffects
```
Then set the audio interfaces within the manifest.xml of the device (if not already performed):

```xml
<manifest version="1.0" type="device">
  ...
  <hal format="hidl">
    <name>android.hardware.audio</name>
    <transport>hwbinder</transport>
    <version>4.0</version>
    <interface>
      <name>IDevicesFactory</name>
      <instance>default</instance>
    </interface>
  </hal>
  <hal format="hidl">
    <name>android.hardware.audio.effect</name>
    <transport>hwbinder</transport>
    <version>4.0</version>
    <interface>
      <name>IEffectsFactory</name>
      <instance>default</instance>
    </interface>
  </hal>
  ...
</manifest>
```

It is possible to use the provided **primary** device implementation: configure it creating an xml file (example and directives are given within the file device/stm/<STM32Series>/peripheral/audio/audio.example.xml).

In this xml configuration file, you must set:

- The mixer card number (e.g. /dev/snd/mixer0) and associated control settings executed at initialization
- The list of devices (linked to Android audio devices) and the list of associated control setting (if any) to be executed in case of activation or deactivation
- The list of stream card and device numbers (e.g. /dev/snd/pcmC1D0p) and the list of associated control setting (if any)

This file must be copied within the device in /vendor/etc/ directory adding these lines in the device.mk:

```makefile
BOARD_USES_TINYHAL_AUDIO := true
PRODUCT_COPY_FILES += \
  device/{{HighlightParam|''<STM32Series>''}}/{{HighlightParam|''<BoardId>''}}/media/audio/audio.stm.xml:$({TARGET_COPY_OUT_VENDOR})/etc/audio/{{HighlightParam|''<BoardId>''}}.xml \
```

A basic implementation is available in the file device/stm/<STM32Series>/peripheral/audio/audio.example.xml.

### 3.2.3 Configure the Android service

The audio is potentially configured in two steps:

- Create the audio policy configuration file giving information on audio capabilities of the platform (mandatory)
- Customize the audio policy manager changing the decision rules concerning the audio

Additional information is available in the Android porting guide.[1]
3.2.3.1 Audio policy configuration

A file named audio_policy_configuration.xml must be created to give detailed configuration/capabilities of the integrated modules.

This file must contain the following information for each module (e.g for primary, usb, a2dp or remote_submix modules):

- List of attached devices (ex: Speaker, HDMI...) and the default device (ex: Speaker) include only the lists of built-in audio devices which are always present:
- List of mixer ports and their possible configurations (ex: primary output and primary input with format PCM 16bits, 48kHz, stereo...)
- List of device ports and their possible configurations (ex: speaker output with format PCM 16bits, 48kHz, mono...)
- List of possible routings links between possible sources (ex: primary output mixer port) and possible sinks (ex: Speaker device port)

In general, the default implementations of the modules are associated with the default configuration files available in frameworks/av/services/audiopolicy/config/ directory. You can then just include them.

These files must be copied within the device at /vendor/etc/ directory. For that purpose, you must add the following lines within the device.mk depending on the list of audio interfaces implemented (e.g. Bluetooth audio devices):

```makefile
USE_XML_AUDIO_POLICY_CONF := 1
PRODUCT_COPY_FILES += \\
frameworks/av/services/audiopolicy/config/usb_audio_policy_configuration.xml:$ (TARGET_COPY_OUT_VENDOR)/etc/usb_audio_policy_configuration.xml \\
frameworks/av/services/audiopolicy/config/r_submix_audio_policy_configuration.xml:$ (TARGET_COPY_OUT_VENDOR)/etc/r_submix_audio_policy_configuration.xml \\
frameworks/av/services/audiopolicy/config/a2dp_audio_policy_configuration.xml:$ (TARGET_COPY_OUT_VENDOR)/etc/a2dp_audio_policy_configuration.xml \\
frameworks/av/services/audiopolicy/config/hearing_aid_audio_policy_configuration.xml:$ (TARGET_COPY_OUT_VENDOR)/etc/hearing_aid_audio_policy_configuration.xml \\
frameworks/av/services/audiopolicy/config/default_volume_tables.xml:$ (TARGET_COPY_OUT_VENDOR)/etc/default_volume_tables.xml \\
frameworks/av/services/audiopolicy/config/audio_policy_volumes.xml:$ (TARGET_COPY_OUT_VENDOR)/etc/audio_policy_volumes.xml \\
device/stm/{{HighlightParam|'STM32Series'}}/{{HighlightParam|'BoardId'}}/media/audio/audio_policy_configuration.xml:$ (TARGET_COPY_OUT_VENDOR)/etc/audio_policy_configuration.xml
```

3.2.3.2 Audio policy customization

The audio policy manager is required in Android, in charge of making all the audio related decisions (ex: audio output device selection, audio input device selection, audio routing, volume, mute...).

A default audio policy manager is implemented within frameworks/av/services/audiopolicy/managerdefault/.

It is possible to customize this policy (decision changes). For that purpose, there are two options:

- Use the configurable audio policy implementation
- Use your own implementation of the AudioPolicyManager (generating your own libaudiopolicymanager)

3.2.3.2.1 Configurable AudioPolicyManager

To enable this option, you must add the following line within the device.mk

```makefile
USE_CONFIGURABLE_AUDIO_POLICY := 1
```
This solution is based on the parameter-framework and an example of implementation is available for audio within frameworks/av/services/audiopolicy/engineconfigurable/parameter-framework/examples/.

### 3.2.3.2 Proprietary AudioPolicyManager

To enable this option, you must add the following line within the file `device.mk`

```bash
USE_CUSTOM_AUDIO_POLICY:= 1
```

This solution requires to implement:

- Your own `AndroidPolicyManager` class similar to the default one in frameworks/av/services/audiopolicy/managerdefault/ generate for example a library `libaudiopolicymanagerSTM`
- Your own `AndroidPolicyFactory` similar to the default one in frameworks/av/services/audiopolicy/managerdefault/ generate the required `libaudiopolicymanager` (statically linked with `libaudiopolicymanagerSTM`)

### 3.3 Validating

Audio validation can be performed in several steps:

- Validate the Kernel integration
- Check the audio Android compliance (VTS/CTS)

#### 3.3.1 Validate the Kernel Integration

> It is required to build the Android distribution in `eng` or `userdebug` mode

Several tools are available to test audio at kernel level:

- `tinycap` used to capture an audio stream input (card/device pair) within a .wav
- `tinymix` used to control inputs and outputs (execute without parameter to get back list of controls)
- `tinypcminfo` get capabilities of a stream input and output associated to a card/device pair
- `tinypad` used to play a .wav file on selected audio stream output (card/device pair)

If required, you can push a .wav file in the device storage:

```bash
PC $> adb push file.wav /data
```

Open a device console executing:

```bash
PC $> adb shell
```

Execute the following command to become root (needed to execute tiny tools):

```bash
Board $> su
```
Some examples of commands:

- **Example on how to get information on the stream output:**

  ```
  Board $> \text{tinypcminfo} -D 0 -d 0
  ```

  Info for card 0, device 0:

  ```
  PCM out:
  Access: 0x000009
  Format[0]: 0x000404
  Format[1]: 00000000
  Format Name: S16 LE, S32 LE
  Subformat: 0x000001
  Rate: min=32000Hz max=48000Hz
  Channels: min=2 max=2
  Sample bits: min=16 max=32
  Period size: min=32 max=16384
  Period count: min=2 max=4096
  ```

- **Example on how to capture an audio frame in a file (mono signal at 16kHz/16bits format):**

  ```
  Board $> \text{tinycap test.wav} -D 0 -d 1 -c 1 -r 16000 -b 16
  ```

  Capture for card 0, device 1 (need to insure that `/dev/snd/pcmC0D1c` exists)

  *Note: use Ctrl^C to stop capture*

- **Example on how to play an audio file:**

  ```
  Board $> \text{tinyplay test.wav} -D 0 -d 0
  ```

  Playback for card 0, device 0 (need to insure that `/dev/snd/pcmC0D0p` exists)

- **Example on how to control an audio card:**

  ```
  Board $> \text{tinymix} -D 0 → used to get list of controls (with ctl IDs)
  Board $> \text{tinymix} -D 0 1 → used to get value of control with ctl ID = 1
  Board $> \text{tinymix} -D 0 1 0 → used to set value 0 to the control with ctl ID = 1
  ```

3.3.2 Validate the Android Integration

Several tests are available, please refer to the official documents associated:

- **VTS** [2]
- **CTS** [3]
How to change audio device for Android

- CTS verifier for audio framework [4]

4 References

- https://source.android.com/compatibility/vts
- https://source.android.com/compatibility/cts

Advanced Audio Distribution Profile
HAL interface definition language (see https://source.android.com/devices/architecture/hidl)
Java Native Interface (for Android)
Advanced Linux sound architecture
Hardware Abstraction Layer
stm32mp1
Read Only
eval,disco (Generic term used, to complete configuration modules paths depending on used board)
High-Definition Multimedia Interface (HDMI standard)
Vendor Test Suite (Android specific)
Compatibility Test Suite (Android specific) or Clear to send (in UART context)