PulseAudio/Examples

< PulseAudio

Related articles

PulseAudio

PulseAudio/Troublesho

Contents

Set default input source

List available input sources

```
$ pacmd list-sources | grep -e 'index:' -e device.string -e 'name:'
```

```
  index: 0
    name: <input>
      device.string = "hw:2"
* index: 1
    name: <oss_input.dsp>
      device.string = "/dev/dsp"
  index: 2
    name: <alsa_output.pci-0000_04_01.0.analog-stereo.monitor>
```

The `*` in front of the index indicates the current default input.

To set a system wide default, add the source name in the `default.pa` file:

```
/etc/pulse/default.pa
```

```
...
set-default-source alsa_output.pci-0000_04_01.0.analog-stereo.monitor
...
```

For temporary use

```
$ pacmd "set-default-source alsa_output.pci-0000_04_01.0.analog-stereo.monitor"
```

Tip: The default source can be referred as `@DEFAULT_SOURCE@` in commands, for example: `$ pactl set-source-mute @DEFAULT_SOURCE@ toggle`.

Set the default output sink

To list the output sinks available, type the following command:

```
$ pacmd list-sinks | grep -e 'name:' -e 'index:'
```

```
  * index: 0
        name: <alsa_output.pci-0000_04_01.0.analog-stereo>
    index: 1
        name: <combined>
```

The `*` in front of the index indicates the current default output.

To set a system wide default, add the source name in the `default.pa` file:

```
/etc/pulse/default.pa
```

```
...
set-default-sink alsa_output.pci-0000_04_01.0.analog-stereo
...
```

When done then you can logout/login or restart PulseAudio manually for these changes to take effect.

Note:

- The numbering of sinks is not guaranteed to be persistent, so all sinks in the `default.pa` file should be identified by the name.
- For quick identification at runtime (e.g. to manage sound volume), you can use the sink index instead of the sink name:

  ```
  $ pactl set-sink-volume 0 +3%
  $ pactl set-sink-volume 0 -3%
  $ pactl set-sink-mute 0 toggle
  ```

- To avoid unnecessary overriding of 100% normal volume it is better to use alternative utilities for managing of sound. See the **forum thread (https://bbs.archlinux.org/viewtopic.php?id=124513)** for more information.

Tip: The default sink can be referred as `@DEFAULT_SINK@` in commands, for example: `$ pactl set-sink-volume @DEFAULT_SINK@ +5%`.

Set the default output sink profile

Sometimes PulseAudio neglects to load the desired profile on start (e.g. a profile for having **#Independent analog and digital outputs on the same card**). To change the default profile, append the following to `default.pa`:

```
/etc/pulse/default.pa
```

```
...
set-card-profile <symbolic-name> <profilename>
```

Note:

You could also use `<cardindex>` instead of `<symbolic-name>`, but using `<symbolic-name>` ensures referencing the correct device. `<cardindex>` is dynamic, and changes when a new device is plugged in.

Find `<symbolic-name>` by running `pacmd list-cards`:

```
$ pacmd list-cards
3 card(s) available.
    index: 0
        name: <alsa_card.pci-0000_01_00.1>
        driver: <module-alsa-card.c>
        owner module: 6
        (...)

    index: 1
        name: <alsa_card.usb-Sony_Interactive_Entertainment_Wireless_Controller-00>
        driver: <module-alsa-card.c>
        owner module: 7
        (...)

    index: 2
        name: <alsa_card.pci-0000_00_14.2>
        driver: <module-alsa-card.c>
        owner module: 8
        (...)
```

In this case, I want to use the device with index number 2, so `<symbolic-name>` should be `alsa_card.pci-0000_00_14.2`.

To find `<profilename>`, set the desired profile manually, then run `pacmd list-cards`:

```
$ pacmd list-cards | grep 'active profile'
        active profile: <off>
        active profile: <off>
        active profile: <output:analog-stereo+output:iec958-stereo+input:analog-stereo>
```

In this case, `default.pa` should now be changed to this:

```
/etc/pulse/default.pa
```

```
...
set-card-profile alsa_card.pci-0000_00_14.2 output:analog-stereo+output:iec958-stereo+input:analog-ste
reo
```

You can test your configuration by running `pactl set-card-profile <symbolic-name> <profilename>`.

Independent analog and digital outputs on the same card

Sound cards may have both analog and digital (iec958) outputs. Pulseaudio does not generate combined profiles by default, you can choose either digital or analog profiles.

The easiest way to make both outputs available is to add a combined profile to the end of default profile configuration file:

```
/usr/share/pulseaudio/alsa-mixer/profile-sets/default.conf
```

```
...

# Profile must be a '+' separated list of relevant mappings configured above
[Profile output:analog-stereo+output:iec958-stereo+input:analog-stereo]
# Human readable description
description = Analog and digital stereo output and analog stereo intput
output-mappings = analog-stereo iec958-stereo
input-mappings = analog-stereo
```

This way a defined profile is added to the end of the list of available profiles.

Although this works, pulseaudio has a nasty habit of falling back to auto-generated profiles, so you may eventually need to set your card back to the combined profile. The best way to overcome this is by writing a custom config with disabled `auto-profiles`. Copy `default.conf` to `custom-profile.conf`, and edit it to suit your needs (this example is for stereo output/input):

```
/usr/share/pulseaudio/alsa-mixer/profile-sets/custom-profile.conf

[General]
auto-profiles = no # disable  profile auto-generation
# Leave only relevant mappings:
[Mapping analog-stereo]
device-strings = front:%f
channel-map = left,right
paths-output = analog-output analog-output-lineout analog-output-speaker analog-output-headphones anal
og-output-headphones-2
paths-input = analog-input-front-mic analog-input-rear-mic analog-input-internal-mic analog-input-dock
-mic analog-input analog-input-mic analog-input-linein analog-input-aux analog-input-video analog-inpu
t-tvtuner analog-input-fm analog-input-mic-line analog-input-headphone-mic analog-input-headset-mic
priority = 15

[Mapping iec958-stereo]
device-strings = iec958:%f
channel-map = left,right
paths-input = iec958-stereo-input
paths-output = iec958-stereo-output
priority = 5

[Profile output:analog-stereo+output:iec958-stereo+input:analog-stereo]
description = Analog and digital stereo output and analog stereo intput
output-mappings = analog-stereo iec958-stereo
input-mappings = analog-stereo
skip-probe=yes # since you know what your sound card has, there is no need for checking which sinks ar
e available
```

Now that you have your custom profile you need to tell pulseaudio to use it. This can be done by defining an **udev rule**:

First get relevant information about your sound card:

```
$ pactl list cards

Card #0
        Name: alsa_card.pci-0000_00_1b.0
--- snip ----
        Properties:
--- snip ---
                device.vendor.id = "8086" # This is a 'vendor' attribute for udev rule
                device.product.id = "1c20" # This is a 'device' attribute for udev rule
```

Now create a config file:

```
/usr/lib/udev/rules.d/91-pulseaudio-custom.rules
```

```
SUBSYSTEM!="sound", GOTO="pulseaudio_end"
ACTION!="change", GOTO="pulseaudio_end"
KERNEL!="card*", GOTO="pulseaudio_end"

SUBSYSTEMS=="pci", ATTRS{vendor}=="0x8086", ATTRS{device}=="0x1c20", ENV{PULSE_PROFILE_SET}="custom-pr
ofile.conf"

LABEL="pulseaudio_end"
```

Now tell udev to reload sound subsystem `udevadm trigger -ssound` (as the root user) and restart pulseaudio. Your sound card should now use only the defined profile and have both analog and digital outputs available.

Simultaneous HDMI and analog output

PulseAudio allows for simultaneous output to multiple sources. In this example, some applications are configured to use HDMI while others are configured to use analog. Multiple applications are able to receive audio at the same time.

```
$ aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: Intel [HDA Intel], device 0: ALC889A Analog [ALC889A Analog]
  Subdevices: 0/1
  Subdevice #0: subdevice #0
card 0: Intel [HDA Intel], device 1: ALC889A Digital [ALC889A Digital]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 0: Intel [HDA Intel], device 3: HDMI 0 [HDMI 0]
  Subdevices: 0/1
  Subdevice #0: subdevice #0
```

Or by using the the `pacmd` command:

```
$ pacmd list-sinks  | grep -e 'name:'  -e 'alsa.device ' -e 'alsa.subdevice '
```

```
        name: <alsa_output.pci-0000_00_1b.0.analog-stereo>
                alsa.subdevice = "0"
                alsa.device = "0"
```

The key to a configuration like this is to understand that whatever is selected in pavucontrol under *Configuration > Internal Audio* is the default device. Load *pavucontrol > Configuration* and select HDMI as the profile.

To setup the analog device as a secondary source, add the following to the `/etc/pulse/default.pa` configuration at the beginning, before any other modules are loaded:

```
### Load analog device
load-module module-alsa-sink device=hw:0,0
load-module module-combine-sink sink_name=combined
set-default-sink combined
```

Restart PulseAudio, run *pavucontrol* and select the "Output Devices" tab. Three settings should be displayed:

1. Internal Audio Digital Stereo (HDMI)
2. Internal Audio
3. Simultaneous output to Internal Audio Digital Stereo (HDMI), Internal Audio

Now start a program that will use PulseAudio such as MPlayer, VLC, mpd, etc. and switch to the "Playback" tab. A drop-down list should be available for the running program to select one of the three sources.

Also see **this thread (https://bbs.archlinux.org/viewtopic.php?id=118026)** for a variation on this theme and **PulseAudio FAQ (https://www.freedesktop.org/wiki/Software/PulseAudio/FAQ#Can_I_use_PulseAudio_to_playback_music_on_two** _sound_cards_simultaneously.3F).

HDMI output configuration

As outlined in https://download.nvidia.com/XFree86/gpu-hdmi-audio-document/index.html#_issues_in_pulseaudio unless the HDMI port is the first output, PulseAudio will not be able to have any audio when using certain graphics cards with HDMI audio support. This is because of a bug in PulseAudio where it will only select the first HDMI output on a device. A work around posted further down is to first find which HDMI output is working by using the aplay utility from ALSA.

The original title for this section indicated the problem is specific to nVidia cards. As seen in this forum thread (https://bbs.archlinux.org/viewtopic.php?id=133222) other cards are affected as well. The rest of the section will use an nVidia card as a case-study but the solution should carry over for people using other affected cards.

Finding HDMI output

Then find the working output by listing the available cards

```
# aplay -l
```

```
sample output:
 **** List of PLAYBACK Hardware Devices ****
 card 0: NVidia [HDA NVidia], device 0: ALC1200 Analog [ALC1200 Analog]
   Subdevices: 1/1
   Subdevice #0: subdevice #0
 card 0: NVidia [HDA NVidia], device 3: ALC1200 Digital [ALC1200 Digital]
   Subdevices: 1/1
   Subdevice #0: subdevice #0
 card 1: NVidia_1 [HDA NVidia], device 3: HDMI 0 [HDMI 0]
   Subdevices: 1/1
   Subdevice #0: subdevice #0
 card 1: NVidia_1 [HDA NVidia], device 7: HDMI 0 [HDMI 0]
   Subdevices: 0/1
   Subdevice #0: subdevice #0
 card 1: NVidia_1 [HDA NVidia], device 8: HDMI 0 [HDMI 0]
   Subdevices: 1/1
   Subdevice #0: subdevice #0
 card 1: NVidia_1 [HDA NVidia], device 9: HDMI 0 [HDMI 0]
   Subdevices: 1/1
   Subdevice #0: subdevice #0
```

In case your HDMI port is wired to the NVIDIA card, but aplay does not detect an NVIDIA audio card, follow NVIDIA/Troubleshooting#No audio over HDMI.

Testing for the correct card

Now a list of the detected cards is known, users will need to test for which one is outputting to the TV/monitor

```
# aplay -D plughw:1,3 /usr/share/sounds/alsa/Front_Right.wav
```

where 1 is the card and 3 is the device substitute in the values listed from the previous section. If there is no audio, then try substituting a different device (on my card I had to use card 1 device 7)

Manually configuring PulseAudio to detect the Nvidia HDMI

Having identified which HDMI device is working, PulseAudio can be forced to use it via an edit to

`/etc/pulse/default.pa` :

```
# load-module module-alsa-sink device=hw:1,7
```

where the 1 is the card and the 7 is the device found to work in the previous section

restart pulse audio

```
 $ pulseaudio -k
 $ pulseaudio --start
```

open the sound settings manager, make sure that under the hardware tab the graphics cards HDMI audio is set to "Digital Stereo (HDMI) Output" (My graphics card audio is called "GF100 High Definition Audio Controller").

Then, open the output tab. There should now be two HDMI outputs for the graphics card. Test which one works by selecting one of them, and then using a program to play audio. For example, use VLC to play a movie, and if it does not work, then select the other.

Automatically switch audio to HDMI

Create a script to switch to the desired audio profile if an HDMI cable is plugged in:

`/usr/local/bin/hdmi_sound_toggle.sh`

```
#!/bin/bash

export PATH=/usr/bin

USER_NAME=$(who | awk -v vt=tty$(fgconsole) '$0 ~ vt {print $1}')
USER_ID=$(id -u "$USER_NAME")
CARD_PATH="/sys/class/drm/card0/"
AUDIO_OUTPUT="analog-surround-40"
PULSE_SERVER="unix:/run/user/"$USER_ID"/pulse/native"

for OUTPUT in $(cd "$CARD_PATH" && echo card*); do
  OUT_STATUS=$(<"$CARD_PATH"/"$OUTPUT"/status)
  if [[ $OUT_STATUS == connected ]]
  then
    echo $OUTPUT connected
    case "$OUTPUT" in
      "card0-HDMI-A-1")
        AUDIO_OUTPUT="hdmi-stereo" # Digital Stereo (HDMI 1)
     ;;
      "card0-HDMI-A-2")
        AUDIO_OUTPUT="hdmi-stereo-extra1" # Digital Stereo (HDMI 2)
     ;;
    esac
  fi
done
echo selecting output $AUDIO_OUTPUT
sudo -u "$USER_NAME" pactl --server "$PULSE_SERVER" set-card-profile 0 output:$AUDIO_OUTPUT+input:anal
og-stereo
```

Make the script executable:

```
chmod +x /usr/local/bin/hdmi_sound_toggle.sh
```

Create a **udev** rule to run this script when the status of the HDMI change:

```
/etc/udev/rules.d/99-hdmi_sound.rules
```

```
KERNEL=="card0", SUBSYSTEM=="drm", ACTION=="change", RUN+="/usr/local/bin/hdmi_sound_toggle.sh"
```

To make the change effective don't forget to reload the udev rules:

```
udevadm control --reload-rules
```

A reboot might be required.

Surround sound systems

Many people have a surround sound card, but have speakers for just two channels, so PulseAudio cannot really default to a surround sound setup. To enable all of the channels, edit `/etc/pulse/daemon.conf` : uncomment the default-sample-channels line (i.e. remove the semicolon from the beginning of the line) and set the value to **6**. For a *5.1* setup, or **8** for a *7.1* setup etc.

```
# Default
default-sample-channels=2
# For 5.1
default-sample-channels=6
# For 7.1
default-sample-channels=8
```

If your channels are not correclty mapped or the volume controls for the individual channels do not work as expected in pavucontrol, and you have a HDMI and an analog soundcard, then try to add the following line to `/etc/pulse/default.pa`

```
load-module module-combine channels=6 channel_map=front-left,front-right,rear-left,rear-right,front-center,lfe
```

Note that this example is for a 5.1 setup.

After doing the edit, restart PulseAudio.

Splitting front/rear

Connect speakers to front analog output and headphones to rear output. It would be useful to split front/rear to separate sinks. Add to `/etc/pulse/default.pa` :

```
 load-module module-remap-sink sink_name=speakers sink_properties="device.description='Speakers'" remix=no master=alsa_output.pci-0000_05_00.0.analog-surround-40 channels=2 master_channel_map=front-left,front-right channel_map=front-left,front-right
 load-module module-remap-sink sink_name=headphones sink_properties="device.description='Headphones'" remix=no master=alsa_output.pci-0000_05_00.0.analog-surround-40 channels=2 master_channel_map=rear-left,rear-right channel_map=front-left,front-right
```

Make sure to replace alsa_output.pci-0000_05_00.0.analog-surround-40 with the sound card name shown in 'pacmd list-sinks'. Now you have 2 additional sinks which can be used separately. You can choose 'sink_name' freely, as long as there is no sink with that name already. The 'remix' parameter controls whether the audio should be down-/upmixed to match the channels in the sink.

Tip: If pulseaudio fails with `master sink not found` , comment out the remapping lines, start

PulseAudio and verify your card output is set to the one you specified (e.g. analog surround 4.0). Alternatively, try using a **sink index**[broken link: invalid section] instead of a sink name.

Splitting 7.1 into 5.1+2.0

Similar to the example above, you can also split a 7.1 configuration into 5.1 surround and stereo output devices. Set your card to 7.1 mode, then add the following lines to `/etc/pulse/default.pa`:

```
 load-module module-remap-sink sink_name=Surround sink_properties="device.description='Surround'" remix=no master=alsa_output.pci-0000_00_14.2.analog-surround-71 channels=6 master_channel_map=front-left,front-right,rear-left,rear-right,front-center,lfe channel_map=front-left,front-right,rear-left,rear-right,front-center,lfe
 load-module module-remap-sink sink_name=Stereo sink_properties="device.description='Stereo'" remix=no master=alsa_output.pci-0000_00_14.2.analog-surround-71 channels=2 master_channel_map=side-left,side-right channel_map=front-left,front-right
```

Make sure to replace alsa_output.pci-0000_00_14.2 with your sound card name, get it by running 'pacmd list-sinks'. This configuration will use the front/rear/center+lfe (green/black/orange) jacks for the 5.1 sink and the side (grey) jack for the stereo sink. It will also downmix any audio to stereo for the stereo sink, but will not touch the 5.1 output.

Tip: If pulseaudio fails with `master sink not found`, comment out the remapping lines, start PulseAudio and verify your card output is set to analog surround 7.1. Alternatively, try using a **sink index**[broken link: invalid section] instead of a sink name.

Disabling LFE remixing

By default, PulseAudio remixes the number of channels to the default-sample-channels and since version 7 it also remixes the LFE channel. If you wish to disable LFE remixing, uncomment the line:

```
; enable-lfe-remixing = yes
```

and replace yes with no:

```
enable-lfe-remixing = no
```

then restart Pulseaudio.

Binaural Headphones

`ladspa-bs2b (https://aur.archlinux.org/packages/ladspa-bs2b/)`AUR provides a plugin to simulate surround sound on stereo headphones. To use it, find your headphones with:

```
$ pacmd list-sinks | grep -e 'name:'
```

```
        name: <alsa_output.pci-0000_00_1b.0.iec958-ac3-surround-51>
        name: <alsa_output.pci-0000_00_1b.0.iec958-ac3-surround-51.equalizer>
        name: <bluez_sink.00_1F_82_28_93_51>
```

Load the plugin (*new* sink_name is up to you, master=*headphone's sink name*):

```
pacmd load-module module-ladspa-sink sink_name=binaural master=bluez_sink.00_1F_82_28_93_51 plugin=bs2b label=bs2b control=700,4.5
```

Use pavucontrol (https://www.archlinux.org/packages/?name=pavucontrol) to transfer streams to the new sink, or:

```
pactl move-sink-input $INPUTID $BINAURALSINKNAME
```

PulseAudio over network

One of PulseAudio's unique features is its ability to stream audio from clients over TCP to a server running the PulseAudio daemon reliably within a LAN. Ensure that client and server systems agree on the time (i.e., use NTP), or audio streams may be choppy or may not work at all.

TCP support (networked sound)

You need to enable the TCP module, on the server edit `/etc/pulse/default.pa` to add or uncomment:

```
load-module module-native-protocol-tcp
```

For this to work, it is a requirement that both the client and server share the same cookie. Ensure that the clients and server share the same cookie file found under `~/.config/pulse/cookie` . It does not matter whose cookie file you use (the server or a client's), just that the server and client(s) share the same one.

Note: If experiencing trouble connecting, use (on server)

```
pacmd list-modules
```

TCP support with anonymous clients

If it is undesirable to copy the cookie file from clients, anonymous clients can access the server by giving these parameters to module-native-protocol-tcp on the server (again in `/etc/pulse/default.pa`):

```
 load-module module-native-protocol-tcp auth-ip-acl=127.0.0.1;192.168.0.0/24 auth-anonymous=1
```

Change the LAN IP subnet to match that of those clients you wish to have access to the server.

Zeroconf (Avahi) publishing

For the remote PulseAudio server to appear in the PulseAudio Device Chooser (`pasystray`), load the appropriate zeroconf modules, and enable the Avahi daemon. On both machines, the client and server, install pulseaudio-zeroconf (https://www.archlinux.org/packages/?name=pulseaudio-zeroconf) then start and enable `avahi-daemon.service` .

On the server, add `load-module module-zeroconf-publish` to `/etc/pulse/default.pa` . On the client, add `load-module module-zeroconf-discover` to `/etc/pulse/default.pa` . Now redirect any stream or complete audio output to the remote PulseAudio server by selecting the appropriate sink.

If you have issues with the remote syncs appearing on the client, try restarting the Avahi daemon on the server to rebroadcast the available interfaces.

Selecting the Server

Run the graphical PulseAudio Volume Control `pavucontrol`. Under the **Output Devices** tab, you should see the local and remote output devices. Under the **Playback** tab, to the left of the "X" Mute Audio button, you should see a box containing the name of an output device. That box is *actually a button*, which will display a drop-down radio-button list of the available output devices, with one output device selected. Selecting an output device from the list will allow the audio stream to be switched to the PulseAudio server associated with that output device. This control is not at all obvious until you have used it, and is especially useful with a remote Headless sound server.

Similarly, under the **Input Devices** tab, local and remote input devices will be seen. And under the **Recording** tab, there will be a box, to the left of the "X" Mute Audio button, with the name of an input device which is actually a button which will display a drop-down radio-button list of available input devices.

Run `pavucontrol` on the local or remote host associated with the audio stream to be directed. For instance, run `pavucontrol` on the remote host to direct the remote audio output to the local host. Run `pavucontrol` on the local host to direct the local audio output to some remote host.

Setting up simultaneous inputs or outputs is a different thing. Search about "monitor" and "module-combine-sink" for that.

Switching the PulseAudio server used by local X clients

To switch between servers on the client from within X, the `pax11publish` command can be used. For example, to switch from the default server to the server at hostname foo:

```
$ pax11publish -e -S foo
```

Or to switch back to the default:

```
$ pax11publish -e -r
```

Instead of telling the PulseAudio server to stream audio (as described above), this will edit PulseAudio variables on the X11 root window, which will instruct the PulseAudio client libraries to connect to a PulseAudio server other than `localhost`. As such, the programs will no longer interact with the local `pulseaudio` process, which can then be **stopped**. Programs such as `pactl`, `pacmd` or `pavucontrol` will need to also run with the appropriate `PULSE_SERVER` environment/X variable to control the remote PulseAudio server.

Note that for the switch to become apparent, the programs using Pulse must be restarted, or their PulseAudio client library otherwise reinitialized (completely stopping and restarting playback may be enough). To make this setting permanent, edit `default-server` in `~/.config/pulse/client.conf` or `/etc/pulse/client.conf`.

When everything else seems to fail

The following is a quick fix and NOT a permanent solution

On the server:

```
 $ paprefs
```

Go to Network Access -> Enable access to local sound devices (Also check both 'Allow discover' and 'Don't require authentication').

On the client:

```
$ export PULSE_SERVER=server.ip && mplayer test.mp3
```

ALSA monitor source

To be able to record from a monitor source (a.k.a. "What-U-Hear", "Stereo Mix"), use `pactl list` to find out the name of the source in PulseAudio (e.g. `alsa_output.pci-0000_00_1b.0.analog-stereo.monitor`). Then add lines like the following to `/etc/asound.conf` or `~/.asoundrc`:

```
pcm.pulse_monitor {
  type pulse
  device alsa_output.pci-0000_00_1b.0.analog-stereo.monitor
}

ctl.pulse_monitor {
  type pulse
  device alsa_output.pci-0000_00_1b.0.analog-stereo.monitor
}
```

Now you can select `pulse_monitor` as a recording source.

Alternatively, you can use pavucontrol to do this: make sure you have set up the display to "All input devices", then select "Monitor of [your sound card]" as the recording source.

Monitor specific output

It is possible to monitor a specific output, for example to stream audio from a music player into a VOIP application. Simply create a null output device:

```
pactl load-module module-null-sink sink_name=<name>
```

In Pulseaudio Volume Control (pavucontrol), under the "Playback" tab, change the output of an application to <name>, and in the recording tab change the input of an application to "Monitor of <name>". Audio will now be outputted from one application into the other.

PulseAudio through JACK

The JACK Audio Connection Kit is popular for audio work, and is widely supported by Linux audio applications. It fills a similar niche as PulseAudio, but with more of an emphasis on professional audio work. It can offer lower latency audio monitoring along with greater control of input and output of multi-i/o sound devices.

The KXStudio method

This is the recommended method as it is officially endorsed by the JACK developers (https://github.com/jackaudio/jackaudio.github.com/wiki/WalkThrough_User_PulseOnJack).

This configuration requires the `jack2 (https://www.archlinux.org/packages/?name=jack2)` package.

JACK now has native features for bridging between ALSA, PulseAudio, and JACK. This will allow you to simultaneously have

JACK and PulseAudio running with both outputting at the same time, with no config editing or terminal commands requried.

If you are using `qjackctl` (https://www.archlinux.org/packages/?name=qjackctl), it is recommended to uninstall it before beginning this.

Begin by installing `cadence` (https://www.archlinux.org/packages/?name=cadence), as well as `pulseaudio-jack` (https://www.archlinux.org/packages/?name=pulseaudio-jack). Once installed and started, JACK bridge configuration is found in the bottom right of the window. The ALSA audio bridge should be set to ALSA -> PulseAudio -> JACK, and the PulseAudio bridge should be enabled. Make sure in `pavucontrol` that all output devices besides Jack sink are muted, and all input devices besides Jack input are muted. Start JACK using the Force Restart button, and if it starts successfully PulseAudio programs should begin outputting to JACK.

The manual sink configuration method

This configuration provides a method of allowing JACK and PulseAudio to run at the same time and output to each other. It uses manual configuration of the systems that bridge between JACK and PulseAudio. This configuration has no reliance on scripts or commands and is entirely based in configuration.

This configuration only works with jack2. To use this configuration, just install the `pulseaudio-jack` (https://www.archlinux.org/packages/?name=pulseaudio-jack) package. `/etc/pulse/default.pa` is already configured to load the modules in `pulseaudio-jack` (https://www.archlinux.org/packages/?name=pulseaudio-jack) if they are present. If you want to be sure, open the file and look for the line:

```
load-module module-jackdbus-detect options
```

Where *`options`* can be any options supported by this module, usually `channels=2` .

As described on the **Jack-DBUS Packaging (https://github.com/jackaudio/jackaudio.github.com/wiki/JackDbusPackaging)** page:

Server auto-launching is implemented as D-Bus call that auto-activates JACK D-Bus service, in case it is not already started, and starts the JACK server. Correct interaction with PulseAudio is done using a D-Bus based audio card "acquire/release" mechanism. When JACK server starts, it asks this D-Bus service to acquire the audio card and PulseAudio will unconditionally release it. When JACK server stops, it releases the audio card that can be grabbed again by PulseAudio.

`module-jackdbus-detect.so` dynamically loads and unloads module-jack-sink and module-jack-source when jackdbus is started and stopped.

If PulseAudio sound does not work, check with `pavucontrol` to see if the relevant programs appear in the playback tab. If not, add the following to `~/.asoundrc` or `/etc/asound.conf` to redirect ALSA to PulseAudio:

```
pcm.pulse {
    type pulse
}

ctl.pulse {
    type pulse
}

pcm.!default {
    type pulse
}
ctl.!default {
    type pulse
}
```

If it still does not work, check with `pavucontrol` in the playback tab and make sure the relevant programs are outputting to PulseAudio JACK Sink instead of your audio card (which JACK has control of, so it will not work). Also ensure that in the JACK graph the PulseAudio JACK Source is connected to the system audio output.

The shell script method

This method allows JACK and PulseAudio to output at the same time. It mostly relies on shell scripts that are automatically run by QJackCTL to manage aspects of how the JACK sinks and PulseAudio behave.

The basic idea is that killing PulseAudio is a bad idea because it may crash any apps using PulseAudio and disrupt any audio playing.

The flow of how this setup works:

1. PulseAudio releases the sound card
2. JACK grabs sound card and starts up
3. script redirects PulseAudio to JACK
4. manually send PulseAudio apps to JACK output (pavucontrol may come in helpful for this)
5. use JACK programs etc
6. via script, stop redirecting PulseAudio to JACK
7. stop JACK and release sound card
8. PulseAudio grabs sound card and reroutes audio to it directly

With QJackCTL, set up these scripts:

`pulse-jack-pre-start.sh` set it up as the execute script on startup script

```
#!/bin/bash
pacmd suspend true
```

`pulse-jack-post-start.sh` set this one up as execute script after startup

```
#!/bin/bash
pactl load-module module-jack-sink channels=2
pactl load-module module-jack-source channels=2
pacmd set-default-sink jack_out
pacmd set-default-source jack_in
```

`pulse-jack-pre-stop.sh` "execute script on shutdown"

```
#!/bin/bash
SINKID=$(LANG=C pactl list | grep -B 1 "Name: module-jack-sink" | grep Module | sed 's/[^0-9]//g')
SOURCEID=$(LANG=C pactl list | grep -B 1 "Name: module-jack-source" | grep Module | sed 's/[^0-9]//g')
pactl unload-module $SINKID
pactl unload-module $SOURCEID
sleep 5
```

`pulse-jack-post-stop.sh` "execute script after shutdown"

```
#!/bin/bash
pacmd suspend false
```

The PulseAudio kill method

This method relies on shell scripts to automatically kill PulseAudio when JACK is started, and automatically restart it when JACK is stopped. This will result in lower CPU usage than having both running, but can cause errors in already running PulseAudio application and does not allow simultaneous output of both.

Using the settings listed above, use QjackCtl to execute a script upon startup and shutdown to load/unload PulseAudio. Part of the reason users may wish to do this is that the above changes disable PulseAudio's automatic hardware detection modules. This particular setup is for using PulseAudio in an exclusive fashion with JACK, though the scripts could be modified to unload and load an alternate non-JACK setup, but killing and starting PulseAudio while programs might be using it would become problematic.

Note: padevchooser in the following example is deprecated. It is replaced by pasystray

The following example could be used and modified as necessary as a startup script that daemonizes PulseAudio and loads the *padevchooser* program (optional, needs to be built from AUR) called `jack_startup` :

```
#!/bin/bash
#Load PulseAudio and PulseAudio Device Chooser
pulseaudio -D
padevchooser&
```

as well as a shutdown script to kill PulseAudio and the Pulse Audio Device Chooser, as another example called `jack_shutdown` also in the home directory:

```
#!/bin/bash
#Kill PulseAudio and PulseAudio Device Chooser
pulseaudio --kill
killall padevchooser
```

Both scripts need to be made executable:

```
chmod +x jack_startup jack_shutdown
```

then with QjackCtl loaded, click on the *Setup* button and then the *Options* tab and tick both "Execute Script after Startup:" And "Execute Script on Shutdown:" and put either use the ... button or type the path to the scripts (assuming the scripts are in the home directory) `~/jack_startup` and `~/jack_shutdown` making sure to save the changes.

PulseAudio through JACK issues

When JACK is started Firefox, Chrome and other apps stop playing video and audio

Firefox/Chrome/etc. is using PulseAudio soundcard sink instead of the JACK sink. Open `pavucontrol` and on the *Playback* tab switch all audiostreams from something like "Built-in Audio Analog Stereo" to something like "Jack sink (PulseAudio JACK Sink)".

After I start JACK the sound from PulseAudio becomes distorted

In QjackCtl click *Setup* and on the *Settings* tab, *Parameters* subtab untick "Realtime". In addition, tweaking Sample Rate, Frames/Period and Period/Buffer may help. Look for latency in the bottom right corner, as you still want minimal latency for audio production. Also, I think Sample Rate should match one of the rates supported by your audio interface (`cat /proc/asound/cardN/codec\#M` and look for `rates` , there could be multiple occurrences).

PulseAudio through OSS

Add the following to `/etc/pulse/default.pa` :

```
load-module module-oss
```

Then start PulseAudio as usual, making sure that sinks and sources are defined for OSS devices.

PulseAudio from within a chroot

Since a chroot sets up an alternative root for the running/jailing of applications, PulseAudio must be installed within the chroot itself (`pacman -S pulseaudio` within the chroot environment).

PulseAudio, if not set up to connect to any specific server (this can be done in `/etc/pulse/client.conf` , through the PULSE_SERVER environment variable, or through publishing to the local X11 properties using module-x11-publish), will attempt to connect to the local pulse server, failing which it will spawn a new pulse server. Each pulse server has a unique ID based on the machine-id value in `/var/lib/dbus` . To allow for chrooted apps to access the pulse server, the following directories must be mounted within the chroot:-

```
/run
/var/lib/dbus
/tmp
~/config/.pulse
```

`/dev/shm` should also be mounted for efficiency and good performance. Note that mounting /home would normally also allow sharing of the `~/.pulse` folder.

PulseAudio selects the path to the socket via XDG_RUNTIME_DIR, so be sure to drag it along when you chroot as a normal user using **sudo** (see **Sudo#Environment variables**).

Disabling automatic spawning of PulseAudio server

Some users may prefer to manually start the PulseAudio server before running certain programs and then stop the PulseAudio server when they are finished. A simple way to accomplish this is to edit `~/.config/pulse/client.conf` or `/etc/pulse/client.conf` and change `autospawn = yes` to `autospawn=no` . Make sure the line is uncommented as well.

```
~/.config/pulse/client.conf #or /etc/pulse/client.conf
```

```
autospawn=no
```

Now you can manually start the pulseaudio server with

```
$ pulseaudio --start
```

and stop it with

```
$ pulseaudio --kill
```

This setting is also respected by the default pulseaudio dektop session startup script `start-pulseaudio-x11` which is executed from `/etc/xdg/autostart/pulseaudio.desktop`.

Disabling pulseaudio daemon altogether

To disable the pulseaudio daemon completely, and thereby preventing it from starting, one can add `daemon-binary=/bin/true` to the configuration file.

```
~/.config/pulse/client.conf #or /etc/pulse/client.conf
```

```
daemon-binary=/bin/true
```

Remap stereo to mono

Remap a stereo input-sink to a mono sink by creating a virtual sink. It would be useful if you only have one speaker. Add to `/etc/pulse/default.pa`:

```
load-module module-remap-sink master=alsa_output.pci-0000_00_1f.5.analog-stereo sink_name=mono sink_pr
operties="device.description='Mono'" channels=2 channel_map=mono,mono
# Optional: Select new remap as default
set-default-sink mono
```

(replace alsa_output.pci-0000_00_1f.5.analog-stereo in the sound card name shown from `pacmd list-sinks`)

Switch player between virtual mono sink and real stereo sink.

Remap left or right to mono

Particularly useful in the case an audio stream has different content in the left and right channels, such as Japanese television broadcasts with bilingual audio.

```
# For Japanese bilingual TV
load-module module-remap-sink sink_name=Left-to-Mono sink_properties="device.description='Left to Mono
(5.1 AC3 on ALC892 Digital)'" master=alsa_output.pci-0000_00_1b.0.iec958-ac3-surround-51 channels=2 ma
ster_channel_map=mono,mono channel_map=front-left,rear-left
load-module module-remap-sink sink_name=Right-to-Mono sink_properties="device.description='Right to Mo
no (5.1 AC3 on ALC892 Digital)'" master=alsa_output.pci-0000_00_1b.0.iec958-ac3-surround-51 channels=2
master_channel_map=mono,mono channel_map=front-right,rear-right
```

Replace `alsa_output.pci-0000_00_1b.0.iec958-ac3-surround-51` (5.1 AC3 on ALC892 Digital) with your own card (`pacmd list-sinks`).

Note:

- *master_channel_map* is a list of outputs to be remapped to.
- *channel_map* is a list of inputs to be remapped from.
- A stereo card will not have to specify as many channels, eg. `channels=1 master_channel_map=mono channel_map=right`

Remap for broadcasting software

If you do not want to capture sound from the application you need to create Remap sink:

```
### Create Remap sink
load-module module-remap-sink sink_name=Remap_sink master=SINK_NAME channels=2 remix=no
set-default-sink Remap_sink
```

Note: Replace `SINK_NAME` with the real name of the master sink `pacmd list-sinks`.

Then restart PulseAudio daemon:

```
$ pulseaudio -k
$ pulseaudio --start
```

Now you need set the `Remap_sink` as the default sound source in broadcast software

Note: Use environment variable `PULSE_SINK=SINK_NAME` for applications, sound that does not need to be captured.

Swap left/right channels

This is the same as "reverse stereo", where the left and right channels are to be swapped.

First, identify the card you want its channels swapped:

```
$ cat /proc/asound/cards
```

and use the name string for the device you wish to use (the one in square brackets, e.g. [Intel]).

Edit `/etc/pulse/default.pa` and comment out module-hal-detect and module-detect lines.

Search for the commented-out line that starts "#load-module module-alsa-sink", uncomment it and change it to

```
load-module module-alsa-sink device=hw:"device_name" channel_map=right,left
```

Restart the pulseaudio deamon by running

```
pulseaudio -k; pulseaudio -D
```

Pulseaudio FAQ: How can I reverse my left and right speaker channels? (https://www.freedesktop.org/wiki/Software/PulseAudio/FAQ/#index34h3)

Using default.pa

Another approach to swapping channels is suggested in [1] (https://superuser.com/a/144252/161008):

```
~/.config/pulse/default.pa
```

```
#!/usr/bin/pulseaudio -nF

.include /etc/pulse/default.pa
```

```
load-module module-remap-sink sink_name=reverse-stereo master=0 channels=2 master_channel_map=front-ri
ght,front-left channel_map=front-left,front-right
set-default-sink reverse-stereo
```

PulseAudio as a minimal unintrusive dumb pipe to ALSA

Some people do not want to run PulseAudio all the time for various reasons. This example will turn the full fledged audio server into an unobstrusive dumb pipe to ALSA devices that automatically starts **and** stops itself when done, allowing applications that requires PulseAudio to fully function while not touching any ALSA setting nor setting itself as the default ALSA device.

This configuration tells native PA clients to autospawn the daemon when they need it, then the daemon is configured to autoexit as soon as all clients have disconnected. The daemon itself uses a plain simple static configuration that uses your configured `pcm.!default` ALSA devices and nothing more. No replacement of ALSA's default, no playing with mixer levels, nothing but record/playback. Also make sure `pulseaudio-alsa (https://www.archlinux.org/packages/?name=pulseaudio-alsa)` is **not** installed so standard ALSA clients don't default to pulse. Since `pulseaudio-alsa (https://www.archlinux.org/packages/?name=pulseaudio-alsa)` contains only a configuration file `/etc/asound.conf`, if it's installed as dependency, one could simply comment all contents in `/etc/asound.conf`. `alsamixer` functions properly as well as any other ALSA clients. Also make sure common frameworks like Xine, Gstreamer and Phonon are configured to use ALSA: by default if they detect PulseAudio is installed they will try to use it before ALSA.

```
/etc/pulse/daemon.conf
```

```
# Replace these with the proper values
exit-idle-time = 0 # Exit as soon as unneeded
flat-volumes = yes # Prevent messing with the master volume
```

```
/etc/pulse/client.conf
```

```
# Replace these with the proper values

# Applications that uses PulseAudio *directly* will spawn it,
# use it, and pulse will exit itself when done because of the
# exit-idle-time setting in daemon.conf
autospawn = yes
```

```
/etc/pulse/default.pa
```

```
# Replace the *entire* content of this file with these few lines and
# read the comments

.fail
    # Set tsched=0 here if you experience glitchy playback. This will
    # revert back to interrupt-based scheduling and should fix it.
    #
    # Replace the device= part if you want pulse to use a specific device
    # such as "dmix" and "dsnoop" so it doesn't lock an hw: device.

    # INPUT/RECORD
    load-module module-alsa-source device="default" tsched=1

    # OUTPUT/PLAYBACK
    load-module module-alsa-sink device="default" tsched=1
    # Accept clients -- very important
    load-module module-native-protocol-unix

.nofail
.ifexists module-x11-publish.so
    # Publish to X11 so the clients know how to connect to Pulse. Will
    # clear itself on unload.
    load-module module-x11-publish
.endif
```

Having both speakers and headphones plugged in and switching in software on-the-fly

By design, Pulseaudio automatically turns off Line Out when headphones are plugged in and uses Headphone slider instead. You can observe this behavior in `alsamixer` . What we want is to have Headphone and Line Out sliders working separately and at the same time. This is extremely useful if you want to remap Realtek's jacks to have, say, Rear Green for headphones and Blue for speakers (with the help of `hdajackretask` from **alsa-tools (https://www.archlinux.org/packages/?name=alsa-tools)**).

To achieve this, you should directly edit Pulseaudio mixer's configuration.

1. We tell pulseaudio that headphones are always plugged in. Edit:

`/usr/share/pulseaudio/alsa-mixer/paths/analog-output-lineout.conf`

Find:

```
[Jack Headphone]
state.plugged = no
state.unplugged = unknown
```

Change `no` to `yes`

2. By default, Line Out's volume controlled only by Master, and not by Line Out slider itself. We want to merge Line Out with Master.

Add this snippet to the end of the file:

```
[Element Line Out]
switch = mute
volume = merge
```

3. We need to completely cut off Line Out when we use headphones. Edit:

`/usr/share/pulseaudio/alsa-mixer/paths/analog-output-headphones.conf`

Add this snippet to the end of the file:

```
[Element Line Out]
switch = off
volume = off
```

Note: On some systems you might need to use `[Element Front]` instead of `[Element Line Out]` .

4. Like Pulseaudio, Alsa itself cuts off speakers when headphones are plugged in. Open `alsamixer` (in case of Realtek HDA

`alsamixer -c0`) and change `Auto-Mute mode` to `disabled` .

5. Restart Pulseaudio

```
$ pulseaudio -k
$ pulseaudio --start
```

Now you have two separate ports on the same sink in pulseaudio. They mute each other, so you can switch to headphones and this will mute Line Out, and vice versa. To switch between ports you can use Gnome or Plasma sound mixer, or install appropriate desktop extension.

Allowing multiple users to use PulseAudio at the same time

It is sometimes desirable to run some programs as another user on the same desktop of the primary user in order to isolate the software. However, PulseAudio will not accept by default connections by the secondary users, since a PulseAudio daemon is already running for the primary user. However, a PulseAudio UNIX socket can be created in order to accept connections from other users to the main PulseAudio daemon run by the primary user.

First, edit `/etc/pulse/default.pa` or `~/.config/pulse/default.pa` and add a directive for the unix socket to be created:

```
~/.config/pulse/default.pa
-----------------------------
load-module module-native-protocol-unix auth-anonymous=1 socket=/tmp/pulse-socket
```

Afterwards, set PulseAudio as a client to the UNIX socket just created in the secondary user:

```
/home/secondaryuser/.config/pulse/client.conf
-----------------------------
default-server = unix:/tmp/pulse-socket
```

Now, after restarting the PulseAudio daemon, applications running as the secondary user should be able to play sound through the main PulseAudio daemon running as the primary user without problems.

Mixing additional audio into the microphone's audio

Using a setup of null sinks and loopbacks you can mix arbitrary applications' audio output into your microphone's audio, for example to play sound effects or music on voice chat applications.

The setup suggested here will also play your sound effects back to you and use **PulseAudio echo cancellation (https://www.freedesktop.org/wiki/Software/PulseAudio/Documentation/User/Modules/#module-echo-cancel)** to prevent the effects from feeding back into your microphone. Despite this, using headphones instead of loudspeakers is probably a good idea.

Overview of the targeted PulseAudio configuration

Symbology: `(Application)` , `{Audio source}` , `[Audio sink]` , `{m} = Monitor of audio sink`

```
{Microphone}
```

```
   ||                                             Input
{mic_ec} -------------> [vsink_fx_mic]{m} ------------> (Voice chat)
            Loopback               ^                         |
                           Loopback|                   Output|
                                   |                         |
             Output                |      Loopback           v
(Soundboard) ---------> [vsink_fx]{m} ----------------> [spk_ec]
                                                           ||
                                                        [Speakers]
```

{mic_ec}, [spk_ec]
: Echo cancellation "clones" of the microphone and speakers

[vsink_fx]
: Virtual sink into which the sound effects are played

[vsink_fx_mic]
: Virtual sink where the microphone and the sound effects are mixed together

Applications configuration

The applications providing the sound effects must

- Output to "vsink_fx"

All other applications, including the voice chat, must

- Record audio from "Monitor of vsink_fx_mic"
- Output to "spk_ec"

Accordingly, these devices will be set as default source and default sink; ultimately, controlling which application uses which audio source/sink can be done in the `pavucontrol` graphical PulseAudio volume control panel.

No application whatsoever must record from, or output to, the "real" microphone or speakers, as this would bypass the echo cancellation.

Any echo cancellation or other audio processing provided by the voice chat application should be disabled – PulseAudio is doing this already, and as the application is not aware of the sound effects being played on the speakers, it will likely be ineffective in filtering them from the microphone anyway.

Setup steps

As of August 2020 there is **pulse-autoconf (https://aur.archlinux.org/packages/pulse-autoconf/)**AUR, a PulseAudio server dynamic configuration daemon that supports this setup with its 'EchoCancellationWithSourcesMix' preset and that dynamically reacts to e.g. a headset being plugged in our unplugged.

If **pulse-autoconf (https://aur.archlinux.org/packages/pulse-autoconf/)**AUR does not work out for your use case, here is the manual way:

1. Connect your microphone and headphones and make sure PulseAudio is configured correctly for their use, for example in the "Configuration" tab in `pavucontrol`

2. *First time only:*
   1. Save the template script below to an executable file of your choice
   2. Find the `name:` s of your microphone and headphones with `pacmd list-sources` and `pacmd list-sinks` , respectively
   3. In the script, replace the values of "microphone" and "speakers" with the names of your microphone/headphones
3. Run the script
4. Run your voice chat application and, in `pavucontrol` , make it record audio from "Monitor of vsink_fx_mic" and output audio to "spk_ec"
5. Run your sound effects application(s) and, in `pavucontrol` , make them play to "vsink_fx"

As for applications that can play sound effects, `castersoundboard-git (https://aur.archlinux.org/packages/castersoundboard-git/)`AUR has been found to work quite well. It however needs to be closed and re-opened when PulseAudio is restarted.

Teardown

The changes that the script makes to the running PulseAudio server are not permanent and will be lost when PulseAudio terminates.

To ditch the custom configuration, just restart PulseAudio, e.g. with `systemctl --user stop pulseaudio.service` . (PulseAudio is socket-activated and will automatically start on demand.)

Template script

```
#!/bin/bash

microphone="alsa_input.pci-0000_00_1b.0.analog-stereo"
speakers="alsa_output.pci-0000_00_1b.0.analog-stereo"

echo "Setting up echo cancellation"
pactl load-module module-echo-cancel use_master_format=1 aec_method=webrtc \
      aec_args="analog_gain_control=0\\ digital_gain_control=1\\ experimental_agc=1\\ noise_suppressio
n=1\\ voice_detection=1\\ extended_filter=1" \
      source_master="$microphone" source_name=mic_ec source_properties=device.description=mic_ec \
        sink_master="$speakers"     sink_name=spk_ec   sink_properties=device.description=spk_ec

echo "Creating virtual output devices"
pactl load-module module-null-sink sink_name=vsink_fx     sink_properties=device.description=vsink_fx
pactl load-module module-null-sink sink_name=vsink_fx_mic sink_properties=device.description=vsink_fx_
mic

echo "Creating loopbacks"
pactl load-module module-loopback latency_msec=30 adjust_time=3 source=mic_ec           sink=vsink_fx_
mic
pactl load-module module-loopback latency_msec=30 adjust_time=3 source=vsink_fx.monitor sink=vsink_fx_
mic
pactl load-module module-loopback latency_msec=30 adjust_time=3 source=vsink_fx.monitor sink=spk_ec

echo "Setting default devices"
pactl set-default-source vsink_fx_mic.monitor
pactl set-default-sink   spk_ec
```

This script has been inspired by **https://askubuntu.com/a/915064** , for more in-depth information also see that post's author's **pulseaudio-config GitHub repository (https://github.com/toadjaune/pulseaudio-config)**.