Sound

For audio handling on Linux, PipeWire is the currently recommended framework.

PipeWire is a server and user space API that provides a platform to handle multimedia pipelines. It is a modern, low-latency audio and video server designed to work with the latest audio use cases and handle professional audio interfaces and applications.

PipeWire was created as a replacement for both the PulseAudio sound server and the Jack Audio Connection Kit (JACK) server. It provides a unified interface for handling video and audio streams and is intended to be flexible and extensible, allowing it to address not only audio but other multimedia tasks as well, such as video conferencing, screen capture, and other multimedia applications. It can also work with different hardware devices, including webcams, microphones, and professional audio devices.

Additionally, it integrates better with the security models of Flatpak and Wayland. It does so via sandboxing processes from one another, preventing an application from snooping on other applications' audio streams. Before allowing an application to record audio or sharing the screen(e.g. in a browser over WebRTC) it will ask the user for permission to do so.

PipeWire implements no connection logic internally, that is the responsibility of a program called a session manager. It watches for new streams and connects them to the appropriate output device or application.

There are two session managers to choose from:

- **PipeWire Media Session:** A very simple session manager that caters to some basic desktop use cases. It was mostly implemented for testing and as an example for building new session managers.
- **WirePlumber:** A more powerful manager and the current recommendation. It is based on a modular design, with Lua plugins that implement the actual management functionality.

WirePlumber is the recommended choice, as it is better maintained, receives regular updates and is more feature-rich.

Installation

The most basic PipeWire setup includes the following packages:

NOTE: PipeWire handles Bluetooth audio devices if the `pipewire-audio` package is installed.

```
pacman -S pipewire pipewire-audio wireplumber
```

Additional packages can be installed to extend PipeWire's compatibility and capabilities:

Package	Description
`pipewire-alsa`	Support for routing ALSA clients through PipwWire
`pipewire-jack`	Support for JACK clients
`pipewire-pulse`	Support for PulseAudio clients (recommended)
`pipewire-v4l2`	Support for handling video devices, e.g. webcams, tuners, etc.
`pipewire-zeroconf`	Support for streaming audio over the network, e.g. an AirPlay receiver

Streaming audio to an AirPlay receiver

PipeWire can send audio to an AirPlay receiver via the `pipewire-zeroconf` package, which includes the necessary RTSP/RAOP modules to create a sink to send audio data to. This requires the Avahi zeroconf daemon.

Refer to the Network section on how to install and setup Avahi.

Firewall ports

If you're using a firewall, make sure that the following ports are open:

TIP: `firewalld` has a preset for RTSP. Make sure to apply the firewall changes permanently.

Port	Protocol	Service
554	TCP	RTSP
554	UDP	RTSP

Port	Protocol	Service
6001	UDP	Some 3rd party AirPlay receivers use this
6002	UDP	Some 3rd party AirPlay receivers use this

Auto-load PipeWire RAOP discovery module

Create a new drop-in config file, e.g. `~/.config/pipewire/pipewire.conf.d/raop-discover.conf`:

```
context.modules = [
    {
        name = libpipewire-module-raop-discover
        args = {
            #raop.latency.ms = 1000
            stream.rules = [
                {
                    matches = [
                        {
                            raop.ip = "~.*"
                            #raop.ip.version = 4 | 6
                            #raop.ip.version = 4
                            #raop.port = 1000
                            #raop.name = ""
                            #raop.hostname = ""
                            #raop.domain = ""
                            #raop.device = ""
                            #raop.transport = "udp" | "tcp"
                            #raop.encryption.type = "RSA" | "auth_setup" | "none"
                            #raop.audio.codec = "PCM" | "ALAC" | "AAC" | "AAC-ELD"
                            #audio.channels = 2
                            #audio.format = "S16" | "S24" | "S32"
                            #audio.rate = 44100
                            #device.model = ""
                        }
                    ]
                    actions = {
```

```
                    create-stream = {
                        #raop.password = ""
                        stream.props = {
                            #target.object = ""
                            #media.class = "Audio/Sink"
                        }
                    }
                }
            }
        ]
    }
}
]
```

Restart the `pipewire` user unit to make pipewire read the new drop-in config file and load the RAOP module automatically upon login:

```
systemctl restart --user pipewire
```

Scan for devices on the network

You can use the `avahi-browse` utility to scan for devices on your network:

```
avahi-browse --all --ignore-local --terminate
```

This will produce a list of devices broadcasting mDNS services over the network (not only AirPlay, but also file sharing, Spotify, Home Kit and various others).

You should now be able to `ping` your AirPlay receiver using its `.local` DNS name:

```
ping my-airplay-receiver.local
```

If everything worked as intended `wpctl status` should list new sinks to output audio to:

```
...
Audio
 ├─ Devices:
 │      53. Starship/Matisse HD Audio Controller [alsa]
 │
 ├─ Sinks:
```

```
 │      47. My AirPlay Reciever                  [vol: 1.00]
 │  *   61. Starship/Matisse HD Audio Controller Analog Stereo [vol: 1.00]
...
```

Finally, use your desktop environment's audio settings panel to select your AirPlay receiver as audio output device.

Revision #5
Created 11 February 2022 19:38:26 by Sebin
Updated 6 March 2024 10:03:38 by Sebin