Guide

Why?

AVPro is the only video player that supports live streams, but it does not and will not support audio filters per this VRChat Canny. For live-music venues and worlds, who by nature typically care about audio, this is a problem because being able to apply audio effects is a great addition. Many worlds have excellent visuals, world design, detail, etc. but often have very basic audio. I hope to share a little knowledge to help with this. For example, it would be nice to have a low-pass filter in a bathroom, or when you approach the main venue area - hearing the bass rumble and increase in volume as you walk up, a slight reverb in the hallway, then perfectly fading into full-spectrum audio. With this method, you can apply a limited (6*) number of arbitrary effects to your live streams.

Use Cases

My use case is to have a low-pass filtered audio stream as the "main area" is approached (just hear the bass rumbling), which fabes into a reverb/echo filtered audio stream in the concrete hallway, and finally perfectly fading into full spectrum audio in the main area. One use-case is carrying multi-lingual audio streams, where each channel (or pairs if you want stereo) can carry a different language. Another use case could be to use the audio stream to carry data to control lighting, or effects in the world. Can also carry multiple separate audio tracks in one stream, i.e. channels 1+2 is one stereo mix, channels 3+4 is another stereo mix, and channel 5 is a mono podcast.

Work Around

AVPro supports up to 8* audio channels, we can use those to carry other audio. There are many use cases and directions this can go, which I'll only touch on briefly. This book will cover applying a low-pass filter to a third audio channel, as I think that is a common "want" in many music venues, and follow up with an example ffmpeg command to do an arbitrary number of channels, each with any effect.

8 Channel notes

Although AVPro can support up to 8 channels, the windows media foundation AAC decoder does not, effectively killing any live 8 channel audio support. This explains the unexpected behavior past 6 channels. 7 channels results in no audio, but video. 8 channels and the stream does not load at all. ¯_( )_/¯

However, the `eac3` codec does support all 8 channels, but not in RTSP. Although an 8channel EAC3 mp4 file seems to work in AVPro, AVPro just hands non-live videos to the unity video player. Live videos get handed to the windows media foundation, which as mentioned above does not support past 5.1 channels. I have not yet found a way to do this for live video, as ffmpeg does not support eac3 7.1, only 5.1. Furthermore, ffmpeg will silently downmix 8 channels into a 5.0(side) configuration if you use eac3. Plex includes an encoder that can do this though, but there are still the issues mentioned above. Finally, VRCDN does not support any ingest format that supports eac3, so you would likely have to make your own CDN (I doubt any other whitelisted service supports eac3 7.1 audio)

As implied above though, non-live video (i.e. just a MP4 file) encoded with eac 7.1 will work in vrchat.

How

Making multi-channel audio streams is a pain unfortunately. Specifically because we want to apply an effect to only select channels, most software doesn't support this. There are a few methods this will include, although I'm sure there are more.

Method #1: Nginx RTMP (the one that works best!)

A some-what better solution is to just host your own RTMP server. Part of this WIP is trying to make this as easy to adopt as possible. Although setting up a server is a big ask, low processing power is needed for simple EQ effects. A Raspberry Pi could handle this with ease. Setting up Nginx is as easy as a few commands as well. The huge benefit of this method is that artists can simply point their OBS towards your server, and everything will just work. I detect no noticeable latency or degradation in anyway by using this "middle man" audio processing. Artists will of course incur a latency penalty from connecting to your server first, just whatever ping time between the artist and the server.

See this tutorial to setup Nginx in depth if you desire (tutorial), but in linux it just boils down to running this:

```
sudo apt install libnginx-mod-rtmp nginx ffmpeg
```

Note: ffmpeg version 3.4.8 plays well with VRCDN, but later versions don't. Not sure why, but if using Ubuntu, the default version in 18.04 works, but later version of Ubuntu won't unless you compile yourself.

Nginx setup is very simple, just add this into `/etc/nginx/nginx.conf`

```
rtmp {
  server {
    listen 1935;
    ping 30s;
    chunk_size 4096;
    # allow publishing from your own network
    allow publish 192.168.1.0/24;
    # allow publishing from another IP.
    allow publish 1.2.3.4;
    deny publish all;

    notify_method get;

    application ingest {
      live on;
      allow play all;
      exec /home/username/rtmp/start_stream.sh $name;
      exec_kill_signal term;
      record off;
   }
  }
}
```

Then the "exec" script is what does the magic. Nginx will execute this script on stream start, and kill it when the stream disconnects. This is very handy. The script it executes:

```
#!/bin/bash

echo "$(date +"%Y/%m/%d %H:%M:%S"): starting"
# This is ultimately the code that runs on stream stop. This kills all children of this script
on_die() {
  # kill all children
  pkill -KILL -P $$
}
# Nginx will trigger this trap on stream end
trap 'on_die' TERM

# read in raw stream, upmix stereo to 5.1
echo "Upmixing to 5.1 surround and pushing to VRCDN"
ffmpeg -i rtmp://127.0.0.1/ingest/test \
        -err_detect ignore_err \
        -filter_complex "[0:a]pan=5.1(side)|FL=FL|FR=FR|LFE<FL+FR|SL=FL|SR=FR[a]" \
        -map 0 -map -0:a -map "[a]" -ac 6 \
        -c:a aac -c:v copy \
        -f flv rtmp://ingest.vrcdn.live/live/YOURVRCDNKEY &


wait
```

And that's it! With this example setup, any streamer just points their OBS to `rtmp://SERVERIP/ingest` with a stream key of `test`. You can of course expose this publicly with port forwarding. A trustworthy host could set it up so a streamer to use their VRCDN key for your server, and you in turn just push the stream out to that streamers own VRCDN account. Again, this would require trust, or maybe just use revocable guest keys.

Once a streamer hits "start stream" - Nginx will fire off that script, which will read in the streamers stream from the RTMP server, then up mix it to 5.1 surround sound, set the audio channels to 6, and pushes it right back out to VRCDN. Ffmpeg default up-mixing behavior sets the LFE channel to have a low-pass, so there is no need for any extra work in my use case. On my home network, I did not notice any additional latency or any negative effects from this additional processing, which is expected as *-pass filters do not transform any audio data, it only removes a frequency range so very low processing overhead.

Additionally, there is this ffmpeg command which will apply individual effects to each channel, after upmixing. The following command will upmix stereo to 4 channel, and apply a lowpass to ch3 and highpass to ch4.

```
ffmpeg -re -i rtmp://localhost/ingestv2/test -err_detect ignore_err -filter_complex
"[0:a]channelsplit[left][right];[left]asplit=2[l1][l2];[l2]lowpass[l2];[right]asplit=2[r1][r2]
;[r2]highpass[r2];[l1][r1][l2][r2]amerge=inputs=4[a]" -map 0:v -map "[a]" -acodec aac -c:v
copy -f flv rtmp://ingest.vrcdn.live/live/yourkey
```

Lastly;

Above and beyond, with a great deal of help from a friend (thanks Jazzy!) I also have a service setup that deals with some basic authentication / stream keys, ability to dual forward to both VRCDN & Twitch, as well as having options for which filters you want applied.

Example use case:
Note: I'm not an audio engineer and I'm totally winging it

My current world has an "main section" you walk up to, then walk down a conrete hallway, then into the main room. I use 6 channels of audio for this:

4 channels in the main room (two speakers in front, two speaker in the rear). Each speaker has the following ffmpeg aecho settings: aecho=0.8:0.88:40:0.4 which stands for

in_gain: 0.8
out_gain: 0.88
delays: 40
decays: 0.4
This seems sensible to have a slight echo/reverb as it's a concrete room.

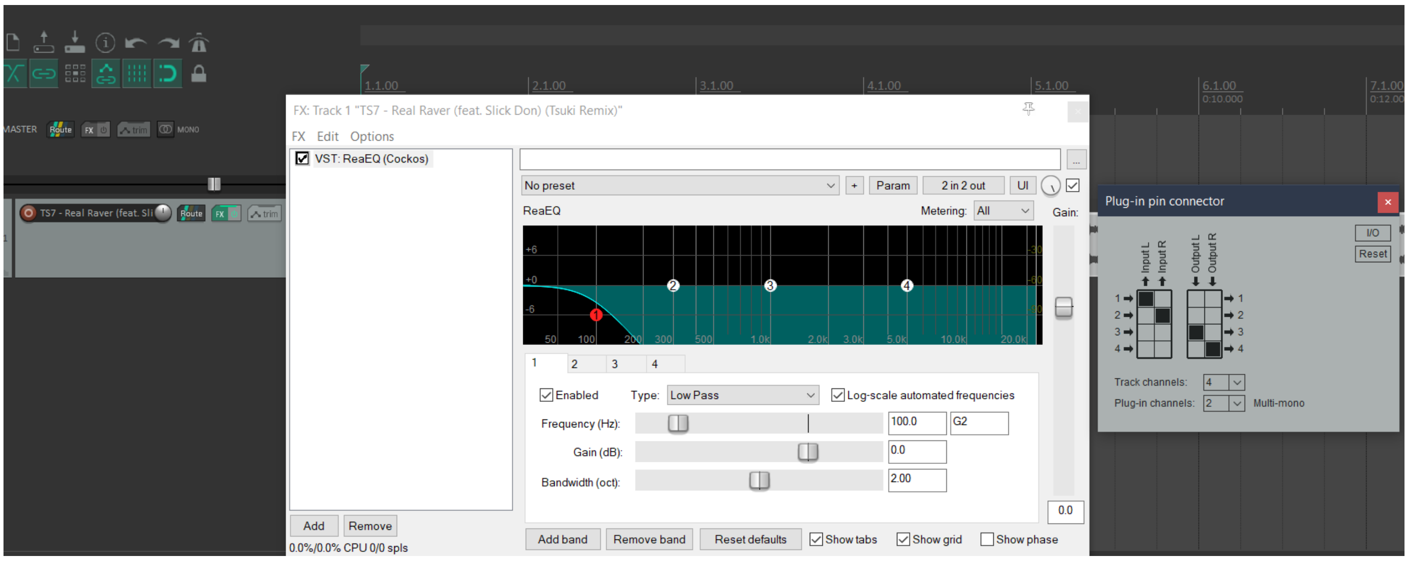
The rear two channels additionally have a 10ms delay. This also seems sensible as the time it takes audio to travel from each speaker is not going to be uniform. Although this isn't dynamically updated which would better reflect reality, it's still a step in the right direction as your brain will still perceive these slight differences, which gives a more spatalized effect.

An additional channel has a 550hz lowpass filter on it. This is used outside the "main section" so as you approach, you hear just the bass rumbling, getting louder as you get closer. The 6th channel has a 128hz lowpass filter on it against my will*. And because all of these channels are on the same audio stream, they all blend together flawlessly as you walk between each audio source
*ffmpeg assumes anytime you output 6 channels, that it _must_ be 5.1 audio, and it forces its upmixing scheme on your audio, which means that channel 4 has a 128hz low pass applied to it. This is a flaw IMO. I raised this on the ffmpeg dev mailing lists, and in a round about way that acknowledged this, and recommended using a different filter to combine audio, which allows me to specify the channel layout to 6.0 instead of 5.1. And this works, except AVPro doesn't acknowledge or respect 6.0 as a valid layout (I suspect) despite very much being a specified layout in several specifications. Lastly, the AAC specification is very lacking in what channel layouts can be specified. 6.0 is not a supported layout, however, this should not stop programs from generating or using a 6.0 layout - the layout bit flag is just a helper.

Method #2: Reaper DAW (takes a lot of client side setup)

Using a Digital Audio Workstation program like Reaper is capable of doing this. This a paid product, but has a free trial. During installation there is an optional checkbox to install ReaRoute which is required to make this work. This method required using a custom OBS fork that supports multi-channel audio found here, in conjunction with a custom audio driver ASIO4All, and lastly a plugin for OBS that can take advantage of the ASIO audio source called obs-asio. Make sure to install all of these.

These introduction videos to Reaper multi-channel audio show you pretty much all you need to know to make this happen (pt1 and pt2). Those videos just show simple up-mixing, routing, and typical surround sound features, but to do something like applying an effect to only certain channels is only slightly different. Instead of applying a ReaSurround filter, you can do something like adding a ReaEQ FX on a track, then for example set the "Plugin pin connector" to use channel 1 & 2 as input, and channel 3 & 4 as output. This would: duplicate channel 1&2 to 3&4, and apply whatever EQ effects you have to only channels 3&4. See the image below for an example setup.


FX: Track 1 "TS7 - Real Raver (feat. Slick Don) (Tsuki Remix)"

FX Edit Options

VST: ReaEQ (Cockos)

No preset | Param | 2 in 2 out | UI

ReaEQ Metering: All Gain:

Enabled Type: Low Pass Log-scale automated frequencies

Frequency (Hz): 100.0 G2

Gain (dB): 0.0

Bandwidth (oct): 2.00

Add Remove

0.0%/0.0% CPU 0/0 spls

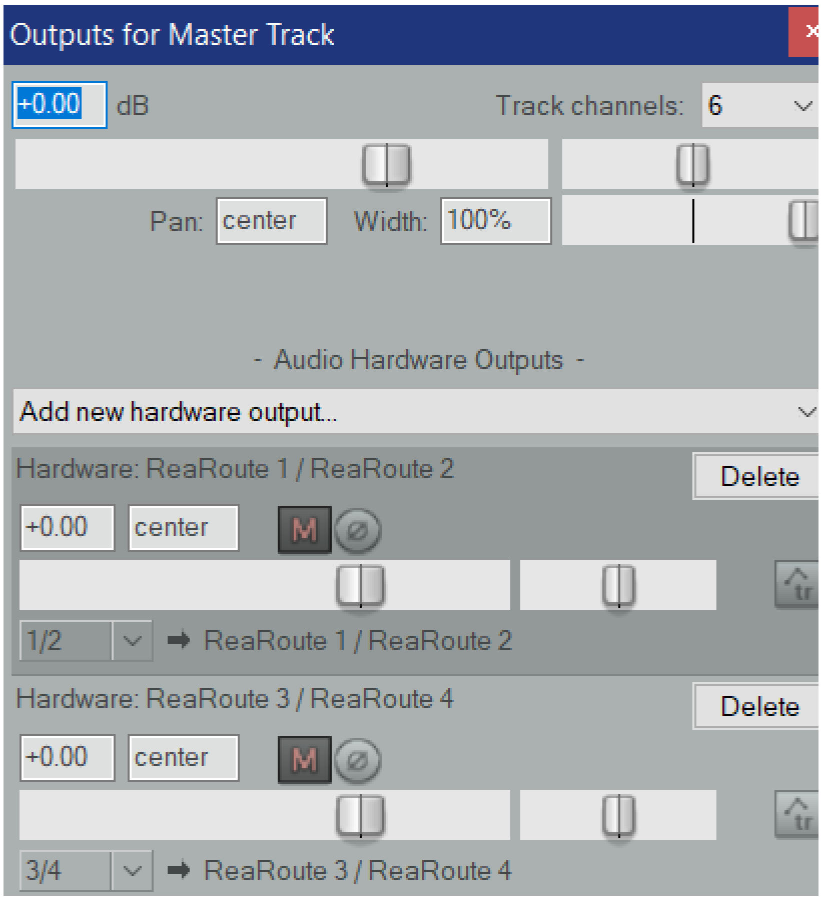
Add band | Remove band | Reset defaults | Show tabs | Show grid | Show phase

Plug-in pin connector

Track channels: 4

Plug-in channels: 2 Multi-mono

Then, with ReaRoute, map the channels in Reaper to ReaRoute output "devices" done in the Route button of the Master track.


Outputs for Master Track

+0.00 dB Track channels: 6

Pan: center Width: 100%

- Audio Hardware Outputs -

Add new hardware output...

Hardware: ReaRoute 1 / ReaRoute 2 Delete

+0.00 center

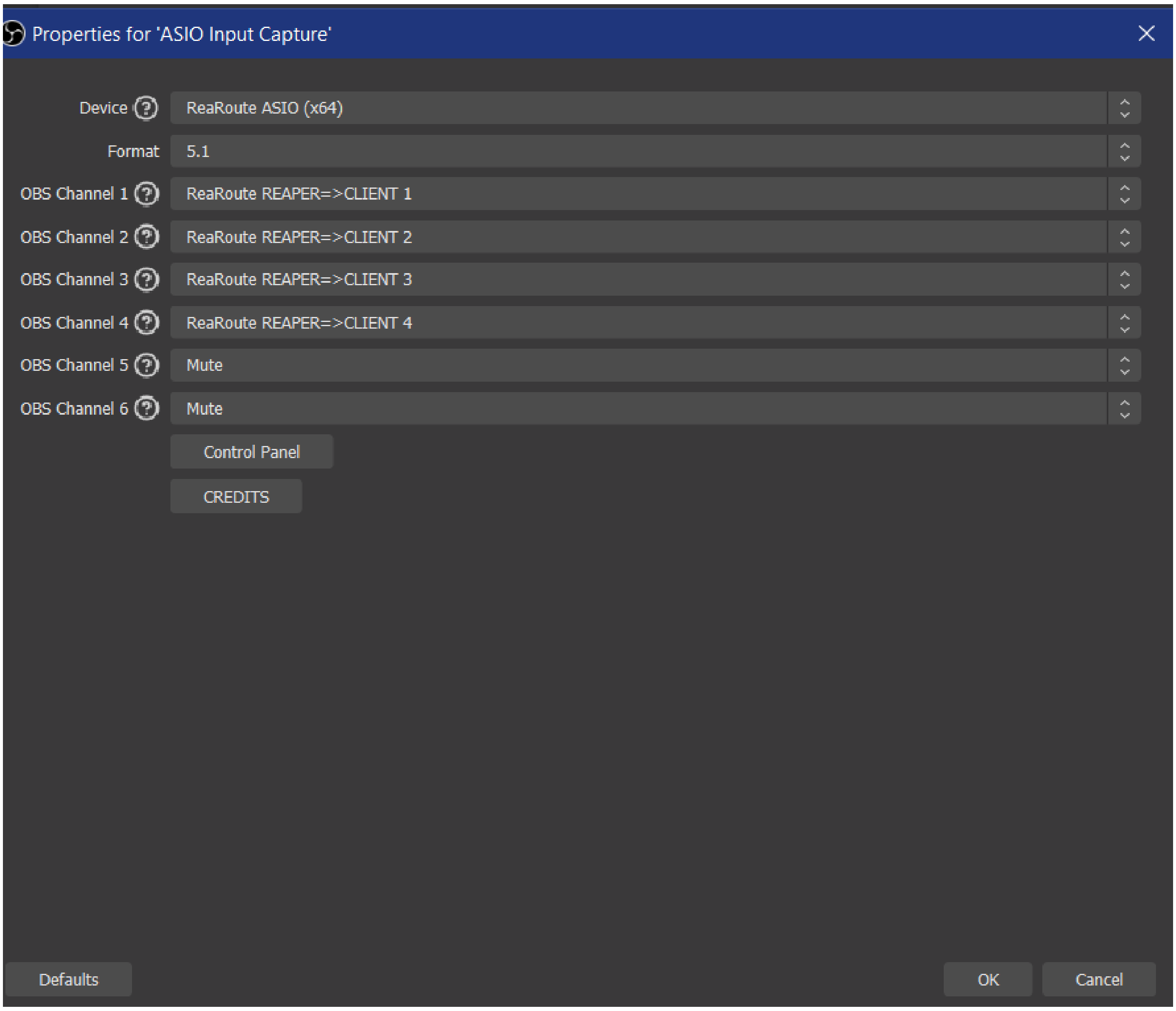
1/2 ➡ ReaRoute 1 / ReaRoute 2

Hardware: ReaRoute 3 / ReaRoute 4 Delete

+0.00 center

3/4 ➡ ReaRoute 3 / ReaRoute 4

This shows mapping channels "1/2" to ReaRoute 1/ReaRoute 2 and channels "3/4" to ReaRoute 3 / ReaRoute 4. This is about it for Reaper. Next, in the OBS Music Edition, first thing is to make sure you have multiple channels enabled (Settings -> Audio -> Channels), then add a new input, there should now be a source called "ASIO Input Capture". Set the channels/sources accordingly and that should be it.


Properties for 'ASIO Input Capture'

Device: ReaRoute ASIO (x64)

Format: 5.1

OBS Channel 1: ReaRoute REAPER=>CLIENT 1

OBS Channel 2: ReaRoute REAPER=>CLIENT 2

OBS Channel 3: ReaRoute REAPER=>CLIENT 3

OBS Channel 4: ReaRoute REAPER=>CLIENT 4

OBS Channel 5: Mute

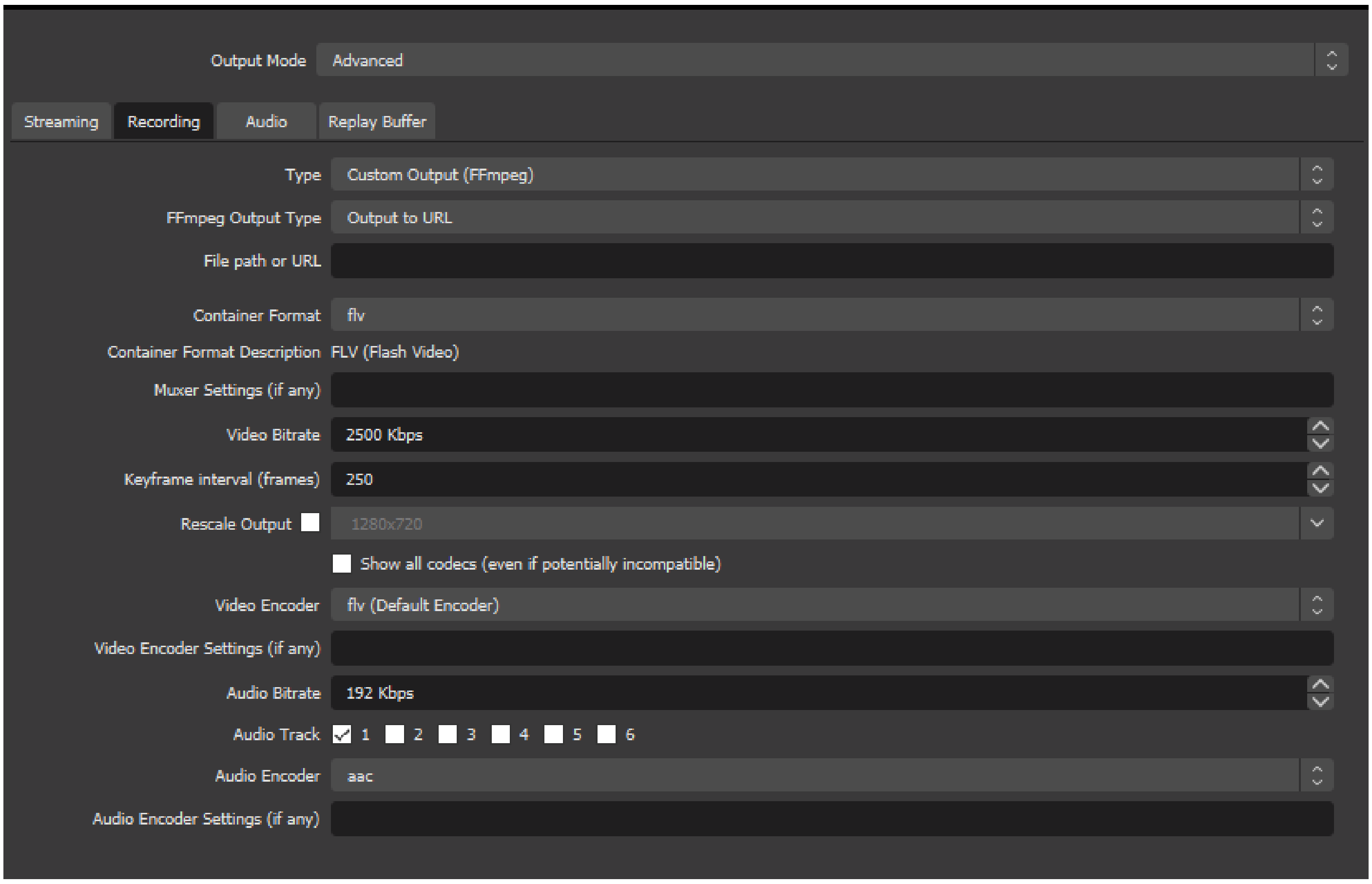
OBS Channel 6: Mute

This method is a lot to setup though, but it does work. I would still like to explore using Reaper to just read from a stream and pushing this out another stream i.e. receive a Twitch stream, manipulate audio, and push back out to either Twitch or VRCDN or something else. This method is unsustainable for multi-artist music venues otherwise as it's just not feasible to ask each artist to go through this kind of setup.

Method #3: Failed attempt to use OBS's native FFMPEG

This method is a dead-end. OBS has a crippled version of ffmpeg, which does not support audio filters. Another route to investigate is Stream-FX for OBS.

~~I haven't tried or tested this yet, but OBS's recording feature allows for ffmpeg output. It should be possible to run these commands direct from OBS, with no need for a server. Much easier than the first method, but still requires asking an artist to go through setup. I have not yet tried this method yet but may look into it in the future.~~



Output Mode	Advanced

Streaming | Recording | Audio | Replay Buffer

Type	Custom Output (FFmpeg)
FFmpeg Output Type	Output to URL
File path or URL	
Container Format	flv
Container Format Description	FLV (Flash Video)
Muxer Settings (if any)	
Video Bitrate	2500 Kbps
Keyframe interval (frames)	250
Rescale Output ☐	1280x720
	☐ Show all codecs (even if potentially incompatible)
Video Encoder	flv (Default Encoder)
Video Encoder Settings (if any)	
Audio Bitrate	192 Kbps
Audio Track	✓ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6
Audio Encoder	aac
Audio Encoder Settings (if any)	

Feel free to reach out on discord if you have any questions or input! lightbulb#0004

Additional References:

https://www.digitalocean.com/community/tutorials/how-to-set-up-a-video-streaming-server-using-nginx-rtmp-on-ubuntu-20-04

https://obsproject.com/wiki/Surround-Sound-Streaming-and-Recording

Revision #13
Created 13 March 2022 19:28:37 by lightbulb
Updated 6 July 2022 17:17:51 by lightbulb