Finding the Right Sound Device

> If you want to recreate TranscriptOMatic on a system different from a Raspberry Pi 500, you will need to find out the right devices and names on your own. This was my approach, that might give you an idea where to start finding the right settings for your environment:

Finding the System's Audio Sinks

Checking for the system's audio sinks:

```
pactl list short sinks
```

The result should look something like this:

```
mela@Cox:~ $ pactl list short sinks
35	auto_null	PipeWire	float32le 2ch 48000Hz	SUSPENDED
```

Finding the Discord Sound Device

After joining a Discord voice channel, checking for the Discord system sound device:

```
pactl list short sink-inputs
```

The result should be something like:

```
mela@Cox:~ $ pactl list short sink-inputs
184	35	183	PipeWire	float32le 2ch 48000Hz
```

Adding a Persistent Sink (And Using Its Monitor as a Stable Audio Source).

Create a dedicated null sink for Discord audio:

```
pactl load-module module-null-sink \
  sink_name=discord_sink \
```

```
  sink_properties=device.description=DiscordSink
```

After joining a Discord voice channel, the sink-input may disappear quickly if the channel is silent. To immediately move the active sink-input to the persistent sink:

```
pactl move-sink-input $(pactl list short sink-inputs | awk '{print $1}') discord_sink
```

This assumes that only a single sink-input is active, which is a reasonable assumption in a minimal setup, but may not hold true on a typical desktop system.

Controlling the result:

```
pactl list short sinks
```

The result should look something like this: `IDLE` instead of `SUSPENDED`.

`IDLE` means the sink exists persistently but currently receives no audio data.

```
mela@Cox:~ $ pactl list short sinks
199□discord_sink□PipeWire□float32le 2ch 48000Hz□IDLE
```

Getting the sound monitor source:

```
pactl list short sources
```

The result should look something like this:

```
mela@Cox:~ $ pactl list short sources
199□discord_sink.monitor□PipeWire□float32le 2ch 48000Hz□IDLE
```

Changing `meeting-start` to Reflect Your Environment

Based on your results, change the relevant parts of meeting-start accordingly:

```
# 1) Ensure discord_sink exists
if ! pactl list short sinks | awk '{print $2}' | grep -qx "$DISCORD_SINK"; then
  pactl load-module module-null-sink \
    sink_name="$DISCORD_SINK" \
    sink_properties=device.description=DiscordSink >/dev/null
fi

# 2) Try to move an active sink-input (Discord) to discord_sink
```

```
#    This may be transient in silent channels → poll briefly.
moved="no"
for _ in {1..40}; do
  SID="$(pactl list short sink-inputs 2>/dev/null | awk 'NF{print $1}' | head -n1 || true)"
  if [[ -n "${SID:-}" ]]; then
    if pactl move-sink-input "$SID" "$DISCORD_SINK" 2>/dev/null; then
      moved="yes"
      break
    fi
  fi
  sleep 0.25
done

if [[ "$moved" != "yes" ]]; then
  echo "⚠  Could not move a sink-input to $DISCORD_SINK."
  echo "    Make sure Legcord is connected to a voice channel, then re-run meeting-start."
fi

# 3) Remove existing whisper_mic remap-sources (idempotent)
for mid in $(pactl list short modules | awk '$0 ~ /module-remap-source/ && $0 ~
/source_name=whisper_mic/ {print $1}'); do
  pactl unload-module "$mid" >/dev/null 2>&1 || true
done
```

A Remapping Check

Under PipeWire, the master source of a remapped source may not be visible via pactl list sources . Successful capture via ffmpeg confirms correct wiring.

```
ffmpeg -f pulse -i whisper_mic -t 3 /tmp/whisper-mic-test.wav
ls -lh /tmp/whisper-mic-test.wav
```

A successful result:

```
-rw-rw-r-- 1 mela mela 565K Jan 11 07:23 /tmp/whisper-mic-test.wav
```

> PipeWire may ignore rate=16000; resampling can be handled downstream (e.g., by ffmpeg or whisper.cpp).

Created 2026-02-22 08:30:24 UTC by Mela
Updated 2026-02-22 09:37:29 UTC by Mela