

Fix your streaming audio quality issues

How to diagnose poor audio quality when streaming — covering input, network, browser, and ingestion method.

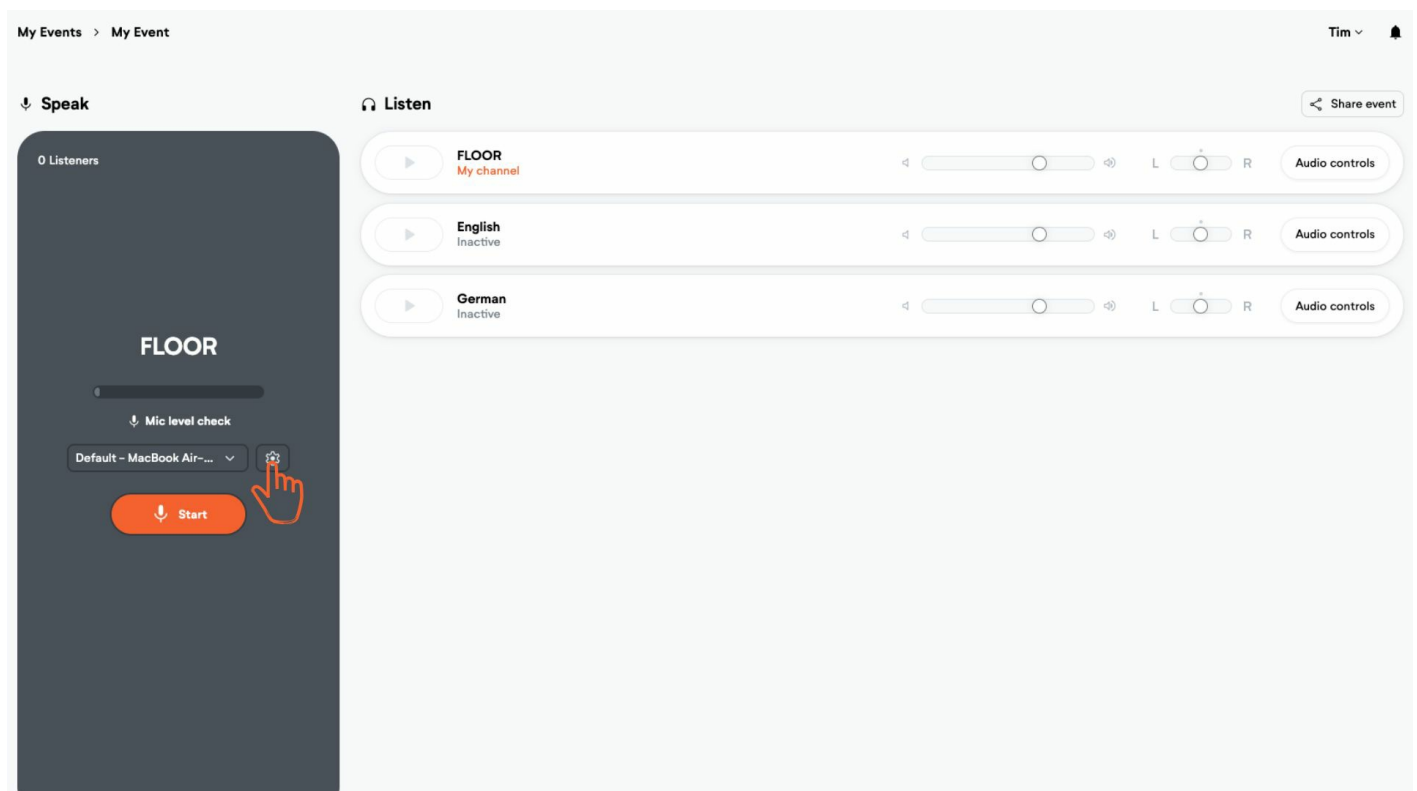
[Useful tips](#)[Live Translation](#)[Silent Conferencing](#)[Audio Description](#)[Guided Tours](#)[Admin](#)[Developer](#)

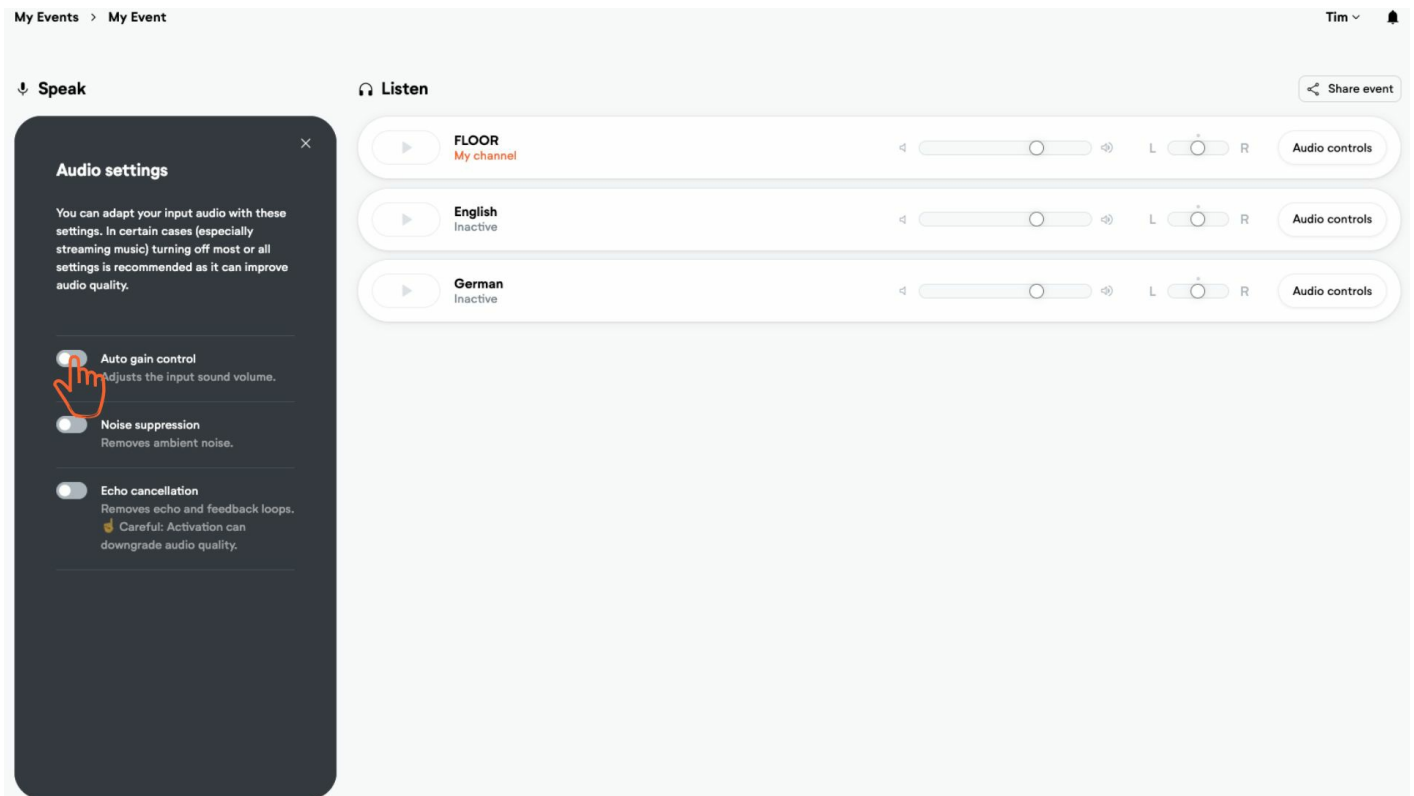
Poor audio quality and connections drops are almost always caused by factors outside LiveVoice. Work through the following checks to narrow it down.

Step 1: Disable browser audio processing (web only)

Turn off **auto-gain-control**, **noise-suppression**, and **echo-cancellation** in your browser's audio settings. These are on by default and can distort the signal.

Check whether the quality changes after disabling them.





Step 2: Try a different input device (web only)

Switch to a different microphone — your laptop's built-in mic instead of an external one, or vice versa. If the quality changes, the issue is with the original input device.

Step 3: Test a different network connection

Try streaming on a different connection: switch from WiFi to a cable, or from a cable to mobile data. If quality improves on one connection, the issue is network-side.

Step 4: Try a different browser (web only)

Open the stream in Chrome, Firefox, or another browser. If one browser sounds noticeably better, the issue is browser-specific.

Step 5: Test on a different operating system

Try streaming from a different device or OS (Windows vs. Mac, for example). A quality difference between them points to an OS-level audio issue.

Step 6: Switch the ingestion method

Change how audio gets into LiveVoice — try the web app, the mobile app, or Patchbay. If one method sounds better than another, that tells us where the problem sits.

Still seeing issues? [Fill in this form](#) and our tech team will take a look.

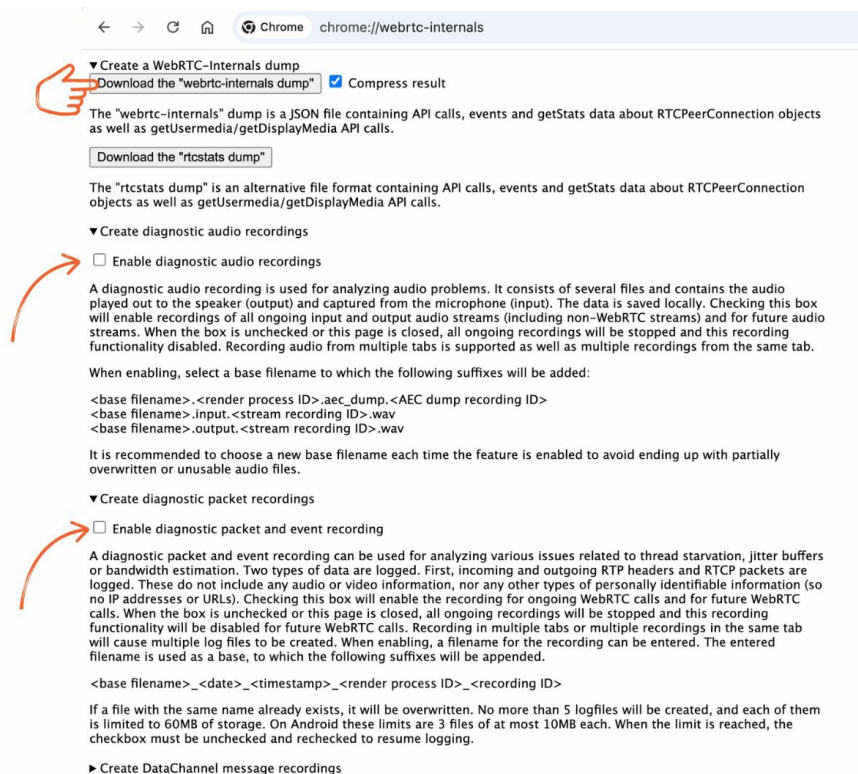
Optional: Collect a WebRTC dump while the problem is happening.

Do this while the audio issue is actively occurring — the data only captures what's happening in real time.

In Chrome, open a new tab and go to `chrome://webrtc-internals`. Click "**Download the \"webrtc-internals\" dump**". Enable "**diagnostic audio recordings**" and "**diagnostic packet and event recording**". To reduce the amount of collected data, this should only be done when the problem occurs (the problem should be

captured as precisely as possible).

Attach all files to the support form.



The screenshot shows the `chrome://webrtc-internals` page in a Chrome browser. It features several sections for debugging WebRTC. Red arrows highlight specific elements: one points to the 'Download the "webrtc-internals dump"' button, another to the 'Enable diagnostic audio recordings' checkbox, and a third to the 'Enable diagnostic packet and event recording' checkbox. The page includes detailed descriptions for each feature, such as the format of the 'webrtc-internals' JSON dump, the local storage of diagnostic audio recordings, and the logging of RTP/RTCP packets for diagnostic packet recording. It also provides filename templates and storage limitations for each recording type.

▼ Create a WebRTC-Internals dump
Download the "webrtc-internals dump" ☒ Compress result

The "webrtc-internals" dump is a JSON file containing API calls, events and getStats data about `RTCPeerConnection` objects as well as `getUserMedia/getDisplayMedia` API calls.

Download the "rtcstats dump"

The "rtcstats dump" is an alternative file format containing API calls, events and getStats data about `RTCPeerConnection` objects as well as `getUserMedia/getDisplayMedia` API calls.

▼ Create diagnostic audio recordings
☐ Enable diagnostic audio recordings

A diagnostic audio recording is used for analyzing audio problems. It consists of several files and contains the audio played out to the speaker (output) and captured from the microphone (input). The data is saved locally. Checking this box will enable recordings of all ongoing input and output audio streams (including non-WebRTC streams) and for future audio streams. When the box is unchecked or this page is closed, all ongoing recordings will be stopped and this recording functionality disabled. Recording audio from multiple tabs is supported as well as multiple recordings from the same tab.

When enabling, select a base filename to which the following suffixes will be added:

```
<base filename>.<render process ID>.aec_dump.<AEC dump recording ID>  
<base filename>.input.<stream recording ID>.wav  
<base filename>.output.<stream recording ID>.wav
```

It is recommended to choose a new base filename each time the feature is enabled to avoid ending up with partially overwritten or unusable audio files.

▼ Create diagnostic packet recordings
☐ Enable diagnostic packet and event recording

A diagnostic packet and event recording can be used for analyzing various issues related to thread starvation, jitter buffers or bandwidth estimation. Two types of data are logged. First, incoming and outgoing RTP headers and RTCP packets are logged. These do not include any audio or video information, nor any other types of personally identifiable information (so no IP addresses or URLs). Checking this box will enable the recording for ongoing WebRTC calls and for future WebRTC calls. When the box is unchecked or this page is closed, all ongoing recordings will be stopped and this recording functionality will be disabled for future WebRTC calls. Recording in multiple tabs or multiple recordings in the same tab will cause multiple log files to be created. When enabling, a filename for the recording can be entered. The entered filename is used as a base, to which the following suffixes will be appended.

```
<base filename>_<date>_<timestamp>_<render process ID>_<recording ID>
```

If a file with the same name already exists, it will be overwritten. No more than 5 logfiles will be created, and each of them is limited to 60MB of storage. On Android these limits are 3 files of at most 10MB each. When the limit is reached, the checkbox must be unchecked and rechecked to resume logging.

► Create DataChannel message recordings