I tried to record live stream from youtube - See all messages including command at the beginning in Msys2 bash:

$ yt-dlp -f bestaudio -o - "<https://www.youtube.com/watch?v=1_Ih0JYmkjI>" | ffmpeg -i - -acodec pcm\_s16le -ar 44100 -ac 2 "/d/Downloads/cnbctv18-10062025.wav"

Edit

Looking at your FFmpeg output, I can see several issues that are causing the live stream recording to fail:

**Main Problems**

1. **HLS Stream Issues**: The error shows problems with YouTube's HLS (HTTP Live Streaming) segments - FFmpeg is having trouble loading the stream segments from Google's servers.
2. **Format Mismatch**: There's a detected format mismatch between AAC audio and the expected format.
3. **Piping Issues**: The pipeline between yt-dlp and FFmpeg is breaking down, causing FFmpeg to receive invalid data.

**Solutions to Try**

**Option 1: Use yt-dlp's built-in FFmpeg integration**

bash

yt-dlp -f bestaudio --audio-format wav --audio-quality 0 "https://www.youtube.com/watch?v=1\_Ih0JYmkjI" -o "/d/Downloads/cnbctv18-10062025.%(ext)s"

**Option 2: Record with different audio formats**

bash

*# Try with MP3 first (more reliable for live streams)*

yt-dlp -f bestaudio -o - "https://www.youtube.com/watch?v=1\_Ih0JYmkjI" | ffmpeg -i - -acodec libmp3lame -ab 192k "/d/Downloads/cnbctv18-10062025.mp3"

**Option 3: Add buffering and error handling**

bash

yt-dlp -f bestaudio -o - "https://www.youtube.com/watch?v=1\_Ih0JYmkjI" | ffmpeg -i pipe:0 -acodec pcm\_s16le -ar 44100 -ac 2 -f wav "/d/Downloads/cnbctv18-10062025.wav"

**Option 4: Use a different approach for live streams**

bash

*# Record for a specific duration (e.g., 1 hour)*

timeout 3600 yt-dlp -f bestaudio -o - "https://www.youtube.com/watch?v=1\_Ih0JYmkjI" | ffmpeg -i pipe:0 -acodec pcm\_s16le -ar 44100 -ac 2 -t 3600 "/d/Downloads/cnbctv18-10062025.wav"

**Additional Troubleshooting Tips**

1. **Check if the stream is still live**: The URL might have expired or the stream ended.
2. **Try a different quality**: Instead of bestaudio, try worst or a specific format:

bash

yt-dlp -F "https://www.youtube.com/watch?v=1\_Ih0JYmkjI" *# List available formats*

1. **Update yt-dlp**: Make sure you have the latest version:

bash

yt-dlp -U

The most reliable approach for live YouTube streams is usually Option 1, as it lets yt-dlp handle the complexities of the live stream internally rather than trying to pipe through FFmpeg.

New error with option 1 above:

Error opening input files: Invalid data found when processing input ERROR: ffmpeg exited with code 3199971767

Edit

The persistent FFmpeg error suggests there are deeper issues. Let's try several alternative approaches:

**Option 1: Direct yt-dlp download (no FFmpeg piping)**

bash

yt-dlp -f bestaudio --extract-audio --audio-format wav --audio-quality 0 "https://www.youtube.com/watch?v=1\_Ih0JYmkjI" -o "/d/Downloads/cnbctv18-10062025.%(ext)s"

**Option 2: Check if the stream is actually live**

bash

*# First, check available formats and if stream is live*

yt-dlp -F "https://www.youtube.com/watch?v=1\_Ih0JYmkjI"

**Option 3: Use a temporary file instead of piping**

bash

*# Download to temp file first, then convert*

yt-dlp -f bestaudio "https://www.youtube.com/watch?v=1\_Ih0JYmkjI" -o "/tmp/temp\_audio.%(ext)s"

ffmpeg -i "/tmp/temp\_audio.\*" -acodec pcm\_s16le -ar 44100 -ac 2 "/d/Downloads/cnbctv18-10062025.wav"

**Option 4: Try with streamlink (alternative tool)**

bash

*# Install streamlink if you don't have it*

pip install streamlink

*# Record the stream*

streamlink "https://www.youtube.com/watch?v=1\_Ih0JYmkjI" audio\_only -o "/d/Downloads/cnbctv18-10062025.mp4"

**Option 5: Use youtube-dl instead of yt-dlp**

bash

youtube-dl -f bestaudio --extract-audio --audio-format wav "https://www.youtube.com/watch?v=1\_Ih0JYmkjI" -o "/d/Downloads/cnbctv18-10062025.%(ext)s"

**Option 6: Simpler FFmpeg approach**

bash

*# Direct FFmpeg recording (if the stream URL is still valid)*

ffmpeg -i "https://www.youtube.com/watch?v=1\_Ih0JYmkjI" -acodec pcm\_s16le -ar 44100 -ac 2 -t 3600 "/d/Downloads/cnbctv18-10062025.wav"

**Diagnostic Steps**

First, run the format check (Option 2) to see:

1. If the stream is still live
2. What audio formats are available
3. The actual stream URLs

The error code 3199971767 suggests a fundamental issue with either:

* The stream URL being invalid/expired
* Network connectivity issues
* The stream having ended
* YouTube blocking the download attempt

Try Option 1 first (direct yt-dlp), then Option 2 to diagnose what's happening with the stream.

j1304@dhawal123 MSYS ~ $ yt-dlp -F "<https://www.youtube.com/watch?v=1_Ih0JYmkjI>" [youtube] Extracting URL: <https://www.youtube.com/watch?v=1_Ih0JYmkjI> [youtube] 1\_Ih0JYmkjI: Downloading webpage [youtube] 1\_Ih0JYmkjI: Downloading tv client config [youtube] 1\_Ih0JYmkjI: Downloading tv player API JSON [youtube] 1\_Ih0JYmkjI: Downloading ios player API JSON [youtube] 1\_Ih0JYmkjI: Downloading m3u8 information [youtube] 1\_Ih0JYmkjI: Downloading m3u8 information [info] Available formats for 1\_Ih0JYmkjI: ID EXT RESOLUTION FPS │ TBR PROTO │ VCODEC VBR ACODEC MORE INFO ───────────────────────────────────────────────────────────────────────────────── 233 mp4 audio only │ m3u8 │ audio only unknown Default, low 234 mp4 audio only │ m3u8 │ audio only unknown Default, high 269 mp4 256x144 15 │ 290k m3u8 │ avc1.42C00B 290k video only 91 mp4 256x144 15 │ 290k m3u8 │ avc1.42C00B mp4a.40.5 229 mp4 426x240 30 │ 546k m3u8 │ avc1.4D4015 546k video only 92 mp4 426x240 30 │ 546k m3u8 │ avc1.4D4015 mp4a.40.5 230 mp4 640x360 30 │ 1210k m3u8 │ avc1.4D401E 1210k video only 93 mp4 640x360 30 │ 1210k m3u8 │ avc1.4D401E mp4a.40.2 231 mp4 854x480 30 │ 1569k m3u8 │ avc1.4D401F 1569k video only 94 mp4 854x480 30 │ 1569k m3u8 │ avc1.4D401F mp4a.40.2 232 mp4 1280x720 30 │ 2969k m3u8 │ avc1.4D401F 2969k video only 95 mp4 1280x720 30 │ 2969k m3u8 │ avc1.4D401F mp4a.40.2 270 mp4 1920x1080 30 │ 5421k m3u8 │ avc1.640028 5421k video only 96 mp4 1920x1080 30 │ 5421k m3u8 │ avc1.640028 mp4a.40.2

Edit

Perfect! The stream is live and available. I can see the formats are working. The issue is likely with the piping method. Let's try these approaches:

**Option 1: Download audio-only format directly**

bash

yt-dlp -f 234 --extract-audio --audio-format wav "https://www.youtube.com/watch?v=1\_Ih0JYmkjI" -o "/d/Downloads/cnbctv18-10062025.%(ext)s"

**Option 2: Use the high-quality audio format with conversion**

bash

yt-dlp -f 234 "https://www.youtube.com/watch?v=1\_Ih0JYmkjI" -o "/d/Downloads/cnbctv18-%(title)s.%(ext)s" --postprocessor-args "ffmpeg:-acodec pcm\_s16le -ar 44100 -ac 2"

**Option 3: Download to temporary file then convert**

bash

*# Download the audio stream first*

yt-dlp -f 234 "https://www.youtube.com/watch?v=1\_Ih0JYmkjI" -o "/d/Downloads/temp\_audio.%(ext)s"

*# Then convert it to WAV*

ffmpeg -i "/d/Downloads/temp\_audio.mp4" -acodec pcm\_s16le -ar 44100 -ac 2 "/d/Downloads/cnbctv18-10062025.wav"

**Option 4: Try with a specific timeout for live recording**

bash

yt-dlp -f 234 "https://www.youtube.com/watch?v=1\_Ih0JYmkjI" --live-from-start --wait-for-video 10 -o "/d/Downloads/cnbctv18-10062025.%(ext)s"

**Option 5: Record for a specific duration**

bash

*# Record for 1 hour (3600 seconds)*

yt-dlp -f 234 "https://www.youtube.com/watch?v=1\_Ih0JYmkjI" --external-downloader ffmpeg --external-downloader-args "ffmpeg:-t 3600 -acodec pcm\_s16le -ar 44100 -ac 2" -o "/d/Downloads/cnbctv18-10062025.wav"

The key issue was that piping live HLS streams can be unreliable. These methods avoid the piping problem by either:

1. Using yt-dlp's built-in conversion
2. Downloading first, then converting
3. Using yt-dlp's external downloader integration with FFmpeg

Try **Option 1** first as it's the simplest and most reliable approach for your needs.

I think we need to look at other options - I mean instead of python library yt-dlp, java script option or even java if nodejs / npm does not work. Will these options work flawlessly instead of python?

Edit

You're right to consider alternatives! Let me show you several non-Python options that often work more reliably for YouTube live streams:

**JavaScript/Node.js Options**

**Option 1: youtube-dl-exec (Node.js wrapper)**

bash

*# Install*

npm install -g youtube-dl-exec

*# Use*

npx youtube-dl-exec "https://www.youtube.com/watch?v=1\_Ih0JYmkjI" --extract-audio --audio-format wav -o "/d/Downloads/cnbctv18-10062025.%(ext)s"

**Option 2: node-ytdl-core (Pure JavaScript)**

bash

*# Install*

npm install -g ytdl-cli

*# Use*

ytdl "https://www.youtube.com/watch?v=1\_Ih0JYmkjI" --quality=highestaudio --output="/d/Downloads/cnbctv18-10062025.%(ext)s"

**Java Options**

**Option 3: YoutubeDL-java**

bash

*# Download the JAR file*

wget https://github.com/sapher/youtubedl-java/releases/download/v1.0.0/youtubedl-java-1.0.0.jar

*# Use*

java -jar youtubedl-java-1.0.0.jar "https://www.youtube.com/watch?v=1\_Ih0JYmkjI" --audio-only --output="/d/Downloads/cnbctv18-10062025"

**Browser-Based Options**

**Option 4: Browser Extensions + Recording**

* **Audio Capture Extension** (Chrome/Edge)
* **Video DownloadHelper** (Firefox)

**Native/Compiled Options**

**Option 5: Streamlink (C++ based)**

bash

*# Often more reliable than Python tools*

streamlink "https://www.youtube.com/watch?v=1\_Ih0JYmkjI" audio\_only -o "/d/Downloads/cnbctv18-10062025.mp4"

**Option 6: FFmpeg direct (if you can get the stream URL)**

bash

*# Get stream URL first with yt-dlp*

yt-dlp -g -f 234 "https://www.youtube.com/watch?v=1\_Ih0JYmkjI"

*# Then use that URL directly with FFmpeg*

ffmpeg -i "STREAM\_URL\_HERE" -acodec pcm\_s16le -ar 44100 -ac 2 -t 3600 "/d/Downloads/cnbctv18-10062025.wav"

**PowerShell/Windows Options**

**Option 7: PowerShell with Invoke-WebRequest**

powershell

*# More complex but sometimes works when other tools fail*

**My Recommendations:**

1. **Try Streamlink first** - It's often the most reliable for live streams
2. **Node.js ytdl-core** - Very actively maintained and handles YouTube changes well
3. **Direct FFmpeg** approach (Option 6) - Bypasses most wrapper issues

Would you like me to help you set up any of these alternatives? **Streamlink** is probably your best bet as it's specifically designed for live streaming and handles YouTube's anti-bot measures better than most Python tools.

Which option appeals to you most, or should I provide detailed setup instructions for Streamlink?

npm needed to be updated. But it finally says like this: $ npm install -g youtube-dl-exec added 53 packages in 15s 19 packages are looking for funding run npm fund for details npm notice npm notice New major version of npm available! 10.9.2 -> 11.4.1 npm notice Changelog: <https://github.com/npm/cli/releases/tag/v11.4.1> npm notice To update run: npm install -g npm@11.4.1 npm notice (venv313) j1304@dhawal123 MSYS ~ $ npm install -g npm@11.4.1 removed 14 packages, and changed 65 packages in 5s 25 packages are looking for funding run npm fund for details

Edit

Great! You've successfully installed youtube-dl-exec and updated npm. Now let's try using it to record the live stream:

**Try the Node.js option now:**

bash

npx youtube-dl-exec "https://www.youtube.com/watch?v=1\_Ih0JYmkjI" --extract-audio --audio-format wav -o "/d/Downloads/cnbctv18-10062025.%(ext)s"

**Alternative Node.js commands to try:**

**Option 1: Specify audio quality**

bash

npx youtube-dl-exec "https://www.youtube.com/watch?v=1\_Ih0JYmkjI" -f bestaudio --extract-audio --audio-format wav -o "/d/Downloads/cnbctv18-10062025.%(ext)s"

**Option 2: Use specific format (234 from your earlier output)**

bash

npx youtube-dl-exec "https://www.youtube.com/watch?v=1\_Ih0JYmkjI" -f 234 --extract-audio --audio-format wav -o "/d/Downloads/cnbctv18-10062025.%(ext)s"

**Option 3: If the above doesn't work, try without extraction (keep as MP4)**

bash

npx youtube-dl-exec "https://www.youtube.com/watch?v=1\_Ih0JYmkjI" -f 234 -o "/d/Downloads/cnbctv18-10062025.%(ext)s"

The Node.js version might handle the live stream better than the Python version. If it downloads as MP4, you can still convert it to WAV afterward with FFmpeg:

bash

ffmpeg -i "/d/Downloads/cnbctv18-10062025.mp4" -acodec pcm\_s16le -ar 44100 -ac 2 "/d/Downloads/cnbctv18-10062025.wav"

Try the first command and let me know what happens!

What is this (updating npm) 25 packages are looking for funding AND the errors on actual execution of your command: $ npx youtube-dl-exec "<https://www.youtube.com/watch?v=1_Ih0JYmkjI>" --extract-audio --audio-format wav -o "/d/Downloads/cnbctv18-10062025.%(ext)s" npm error could not determine executable to run npm error A complete log of this run can be found in: C:\Users\j1304\AppData\Local\npm-cache\\_logs\2 025-06-10T06\_20\_20\_167Z-debug-0.log

Edit

**About the funding message:**

The "25 packages are looking for funding" is just a notice that some of the npm packages you installed have developers seeking financial support. It's completely harmless - just informational. You can ignore it.

**About the npm error:**

The error suggests youtube-dl-exec might not have installed correctly or doesn't have an executable. Let's try different approaches:

**Option 1: Try the direct package name**

bash

npx youtube-dl-exec --version

If that fails, let's try other Node.js options:

**Option 2: Install ytdl-cli instead**

bash

npm install -g ytdl-cli

ytdl "https://www.youtube.com/watch?v=1\_Ih0JYmkjI" --quality=highestaudio

**Option 3: Install node-youtube-dl**

bash

npm install -g node-youtube-dl

youtube-dl "https://www.youtube.com/watch?v=1\_Ih0JYmkjI" -f bestaudio --extract-audio --audio-format wav

**Option 4: Let's try Streamlink instead (often more reliable)**

bash

*# Install streamlink (if you have Python/pip)*

pip install streamlink

*# Use it*

streamlink "https://www.youtube.com/watch?v=1\_Ih0JYmkjI" audio\_only -o "/d/Downloads/cnbctv18-10062025.mp4"

**Option 5: Go back to the direct URL approach**

bash

*# Get the direct stream URL*

yt-dlp -g -f 234 "https://www.youtube.com/watch?v=1\_Ih0JYmkjI"

*# Then use that URL with FFmpeg directly*

Try **Option 1** first to see if the package installed correctly, then we can decide on the best approach. Streamlink (Option 4) is often the most reliable for live streams if you're open to that.

$ npm install -g node-youtube-dl npm warn deprecated node-youtube-dl@0.0.2: No longer maintained npm warn deprecated har-validator@5.1.5: this library is no longer supported npm warn deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math .random() in certain circumstances, which is known to be problematic. See <https://v8.dev/blog/math-> random for details. npm warn deprecated request@2.88.2: request has been deprecated, see <https://github.com/request/requ> est/issues/3142 added 48 packages in 3s 3 packages are looking for funding run npm fund for details

Edit

The package installed but with deprecation warnings (it's no longer maintained). Let's try it, but if it doesn't work, we'll move to better alternatives:

**Try the deprecated package:**

bash

youtube-dl "https://www.youtube.com/watch?v=1\_Ih0JYmkjI" -f bestaudio --extract-audio --audio-format wav

**Better alternatives to try:**

**Option 1: Install Streamlink (most reliable for live streams)**

bash

pip install streamlink

streamlink "https://www.youtube.com/watch?v=1\_Ih0JYmkjI" audio\_only -o "/d/Downloads/cnbctv18-10062025.mp4"

**Option 2: Use the direct URL method (bypasses most issues)**

bash

*# Step 1: Get the direct stream URL*

yt-dlp -g -f 234 "https://www.youtube.com/watch?v=1\_Ih0JYmkjI"

*# Step 2: Copy that URL and use it directly with FFmpeg*

*# ffmpeg -i "PASTE\_URL\_HERE" -acodec pcm\_s16le -ar 44100 -ac 2 -t 3600 "/d/Downloads/cnbctv18-10062025.wav"*

**Option 3: Try a modern Node.js alternative**

bash

npm install -g @microlink/youtube-dl

youtube-dl "https://www.youtube.com/watch?v=1\_Ih0JYmkjI"

**Option 4: Browser-based recording**

If all command-line tools fail, you could use browser extensions like:

* **Audio Capture** (Chrome extension)
* **Video DownloadHelper** (Firefox)

**I'd recommend trying Option 1 (Streamlink) first** - it's specifically designed for live streaming and handles YouTube's complexities better than most tools.

Which approach would you like to try first?

Can this be resolved? $ streamlink "<https://www.youtube.com/watch?v=1_Ih0JYmkjI>" audio\_only -o "/d/Downloads/cnbctv18-10062025.mp4" [cli][info] Found matching plugin youtube for URL <https://www.youtube.com/watch?v=1_Ih0JYmkjI> error: The specified stream(s) 'audio\_only' could not be found. Available streams: 144p (worst), 240p, 360p, 480p, 720p, 1080p (best)

Edit

Yes! Streamlink is working but doesn't have an audio\_only stream. Let's fix this:

**Option 1: Use the lowest quality video stream (it still contains audio)**

bash

streamlink "https://www.youtube.com/watch?v=1\_Ih0JYmkjI" 144p -o "/d/Downloads/cnbctv18-10062025.mp4"

Then extract audio afterward:

bash

ffmpeg -i "/d/Downloads/cnbctv18-10062025.mp4" -vn -acodec pcm\_s16le -ar 44100 -ac 2 "/d/Downloads/cnbctv18-10062025.wav"

**Option 2: Use streamlink with FFmpeg pipe to extract audio directly**

bash

streamlink "https://www.youtube.com/watch?v=1\_Ih0JYmkjI" 144p -O | ffmpeg -i pipe:0 -vn -acodec pcm\_s16le -ar 44100 -ac 2 "/d/Downloads/cnbctv18-10062025.wav"

**Option 3: Record for specific duration (e.g., 1 hour)**

bash

streamlink "https://www.youtube.com/watch?v=1\_Ih0JYmkjI" 144p -O | ffmpeg -i pipe:0 -vn -acodec pcm\_s16le -ar 44100 -ac 2 -t 3600 "/d/Downloads/cnbctv18-10062025.wav"

**Option 4: Use the direct URL method (most reliable)**

bash

*# Get the direct stream URL for audio format 234*

yt-dlp -g -f 234 "https://www.youtube.com/watch?v=1\_Ih0JYmkjI"

This will output a direct URL that you can then use with FFmpeg:

bash

ffmpeg -i "DIRECT\_URL\_HERE" -acodec pcm\_s16le -ar 44100 -ac 2 -t 3600 "/d/Downloads/cnbctv18-10062025.wav"

**Try Option 2 first** - it should work well and extract only the audio while recording. The -vn flag tells FFmpeg to ignore the video stream and only process audio.

You have last piped the command. How I can delete cached data of audio + video?

Edit

You're asking about clearing cached data from the piped command. Here are several ways to clear different types of cache:

**Clear YouTube-DL/yt-dlp Cache**

bash

*# Clear yt-dlp cache*

yt-dlp --rm-cache-dir

*# Or manually delete cache directory*

rm -rf ~/.cache/yt-dlp/

**Clear FFmpeg temporary files**

bash

*# FFmpeg usually doesn't cache, but check temp directory*

ls /tmp/ | grep ffmpeg

rm /tmp/ffmpeg\*

*# Windows temp (if using Windows paths)*

rm -rf /c/Users/j1304/AppData/Local/Temp/ffmpeg\*

**Clear Streamlink cache**

bash

*# Clear streamlink cache*

rm -rf ~/.cache/streamlink/

**Clear npm cache**

bash

npm cache clean --force

**Clear system temporary files**

bash

*# Linux/MSYS2 temp*

rm -rf /tmp/\*

*# Your Downloads directory (be careful!)*

ls /d/Downloads/

*# Remove specific files if needed*

rm /d/Downloads/cnbctv18-\*

**Kill any running processes**

bash

*# Kill any hanging ffmpeg processes*

pkill ffmpeg

pkill yt-dlp

pkill streamlink

*# Check what's running*

ps aux | grep -E "(ffmpeg|yt-dlp|streamlink)"

**Clear browser cache (if you used browser tools)**

* **Chrome**: Ctrl+Shift+Delete
* **Firefox**: Ctrl+Shift+Delete

**Check disk space usage**

bash

*# See what's taking space*

du -sh /d/Downloads/

df -h

**Most likely you want to run:**

bash

yt-dlp --rm-cache-dir

rm /d/Downloads/cnbctv18-\*

pkill ffmpeg

Which specific cache/data are you looking to clear?

streamlink "<https://www.youtube.com/watch?v=1_Ih0JYmkjI>" 144p -O | ffmpeg -i pipe:0 -vn -acodec pcm\_s16le -ar 44100 -ac 2 "/d/Downloads/cnbctv18-10062025.wav"

Worked. But whether it will also record advertisements which are shown in between the stream?

Edit

YouTube live stream ads recorded streamlink ffmpeg

I recorded live stream from youtube using: streamlink "<https://www.youtube.com/watch?v=1_Ih0JYmkjI>" 144p -O | ffmpeg -i pipe:0 -vn -acodec pcm\_s16le -ar 44100 -ac 2 "/d/Downloads/cnbctv18-10062025.wav" It Worked. But whether it will also record advertisements which are shown in between the stream?

Edit

Yes, your command will likely record advertisements that appear during the live stream. Here's why:

When you use streamlink to capture a YouTube live stream, it's essentially downloading the video stream as it's being broadcast. Since advertisements are typically inserted directly into the live stream feed (server-side ad insertion), they become part of the continuous stream data that streamlink receives.

Your command setup:

* streamlink captures the live stream
* The -vn flag in ffmpeg discards video, keeping only audio
* Ads that are part of the stream will be recorded as audio

However, there are a few considerations:

1. **Ad type matters**: If YouTube uses client-side ad insertion (where ads are loaded separately by the web player), those might be skipped. But most live streams use server-side insertion where ads are baked into the stream.
2. **Ad blocking**: If you're using streamlink with certain configurations or if your network has ad blocking, some ads might be filtered out, but this isn't guaranteed for live streams.
3. **Stream continuity**: Since you're capturing a continuous audio stream, any audio content (including ads) that's part of the broadcast will be recorded sequentially.

If you want to minimize ads, you could:

* Use streamlink with --twitch-disable-ads (though this is Twitch-specific)
* Try different quality settings, as some may have fewer ads
* Consider that some live streams have fewer ads than others

The recording will capture whatever audio is being broadcast at the time, so ads will likely be included in your final WAV file.