Encoding for streaming sites

Such as twitch.tv, YouTube Live, Facebook, and other RTMP streaming providers. Since FFmpeg development is very active it is recommend to use a current version. See compilation guides or FFmpeg Download to take advantage of bug fixes and new features.

This guide assumes that you will be using the `libx264` encoder which offers high quality, fast encoding speeds, and high compatibility. Other encoders may be more efficient, but slower, e.g. `libx265`. For more info, please read the H.264 Encoding Guide.

> **Note:** You may have to tweak the commands and settings listed below, e.g. by customizing the `-b:v`/`-crf`, `-preset`, `-maxrate`, `-bufsize`, and `-g` options. Make sure you understand what they mean, and visually inspect your output.

Options

`-b:v`

Video bitrate. Value is in bits. Refer to the documentation of your specific streaming service for bitrate recommendations.

`-preset`

This provides the compression to encoding speed ratio. Use the slowest preset you can: `ultrafast`, `superfast`, `veryfast`, `faster`, `fast`, `medium`, `slow`, `slower`, `veryslow`, `placebo`. Try `-preset veryfast` if you are unsure of what to choose, then watch the console output or the video to see if the encoder is keeping up with your desired output frame rate: if it is not then use a faster preset and/or reduce your width x height or frame rate.

`-maxrate`

Anytime you are encoding video with bandwidth as a limiting factor you should be using VBV (Video Buffer Verifier) with the `-maxrate` and `-bufsize` options:

- Assuming your upload rate is 1024kbit/s (1 megabit/s), and assuming you can reliably utilize 80% of that = 820 kbit/s. Audio will consume 128 kbit/s (64k/channel for stereo, but you can of course use a different audio bitrate) leaving ~692 kbit/s for video: this is your `-maxrate` value.
- If you have a sane upload rate, or do not know what to choose then a `-maxrate` value of up to 3000k-4000k will probably be fine if your upload rate can handle it (depending on your input complexity and `-video_size`). Refer to your streaming service for any limitations that may apply.
- Note that the claimed data rate pay your ISP for may not actually be what you get.

`-bufsize`

`-bufsize` sets the buffer size, and can be 1-2 seconds for most gaming screencasts, and up to 5 seconds for more static content. If you use `-maxrate 960k` then use a `-bufsize` of 960k-1920k. You will have to experiment to see what looks best for your content. Refer to your streaming service for the recommended buffer size (it may be shown in seconds or bits).

`-g`

Use a 2 second GOP (Group of Pictures), so simply multiply your output frame rate * 2. For example, if your input is `-framerate 30`, then use `-g 60`.

Examples

Streaming your desktop

Examples below use x11grab for Linux. Windows users can use dshow or gdigrab. macOS can use avfoundation. See FFmpeg Wiki: Capture Desktop for additional examples.

Without scaling the output

If you want the output video frame size to be the same as the input:

```
$ ffmpeg -f alsa -ac 2 -i hw:0,0 -f x11grab -framerate 30 -video_size 1280x720 \
-i :0.0+0,0 -c:v libx264 -preset veryfast -b:v 1984k -maxrate 1984k -bufsize 3968k \
-vf "format=yuv420p" -g 60 -c:a aac -b:a 128k -ar 44100 \
-f flv rtmp://live.twitch.tv/app/<stream key>
```

Scaling the output

If you want the output video frame size to be smaller than the input then you can use the scale video filter:

```
$ ffmpeg -f alsa -ac 2 -i hw:0,0 -f x11grab -framerate 30 -video_size 1680x1050 \
-i :0.0+0,0 -c:v libx264 -preset veryfast -b:v 3000k -maxrate 3000k -bufsize 3000k \
-vf "scale=1280:-1,format=yuv420p" -g 60 -c:a aac -b:a 128k -ar 44100 \
-f flv rtmp://live.twitch.tv/app/<stream key>
```

The `-1` in the scale filter example will automatically calculate the correct value to preserve the height. In this case the output will have a frame size of 1280x800.

With webcam overlay/picture-in-picture (PiP)

This will place your webcam overlay in the top right:

```
$ ffmpeg -f x11grab -video_size 1680x1050 -framerate 30 -i :0.0 \
-f v4l2 -video_size 320x240 -framerate 30 -i /dev/video0 \
-f alsa -ac 2 -i hw:0,0 -filter_complex \
"[0:v]scale=1024:-1,setpts=PTS-STARTPTS[bg]; \
 [1:v]scale=120:-1,setpts=PTS-STARTPTS[fg]; \
 [bg][fg]overlay=W-w-10:10,format=yuv420p[v]" \
-map "[v]" -map 2:a -c:v libx264 -preset veryfast \
-b:v 3000k -maxrate 3000k -bufsize 4000k -c:a aac -b:a 160k -ar 44100 \
-f flv rtmp://live.twitch.tv/app/<stream key>
```

- You can see additional details your webcam with something like: `ffmpeg -f v4l2 -list_formats all -i /dev/video0` or with `v4l2-ctl --list-formats-ext`. See the documentation on the video4linux2 (v4l2) input device for more info.
- Your webcam may already support whatever frame size you want to overlay onto the main video, so scaling the webcam video as shown in this example can be omitted (just set the appropriate v4l2 `-video_size` and remove the `scale=120:-1,`).

With webcam overlay and logo

This will place your webcam overlay in the top right, and a logo in the bottom left:

```
$ ffmpeg -f x11grab -video_size 1680x1050 -framerate 30 -i :0.0 \
-f v4l2 -video_size 320x240 -framerate 30 -i /dev/video0 \
-f alsa -ac 2 -i hw:0,0 -i logo.png -filter_complex \
```

```
"[0:v]scale=1024:-1,setpts=PTS-STARTPTS[bg]; \
 [1:v]scale=120:-1,setpts=PTS-STARTPTS[fg]; \
 [bg][fg]overlay=W-w-10:10[bg2]; \
 [bg2][3:v]overlay=W-w-10:H-h-10,format=yuv420p[v]"
-map "[v]" -map 2:a -c:v libx264 -preset veryfast \
-maxrate 3000k -bufsize 4000k -c:a aac -b:a 160k -ar 44100 \
-f flv rtmp://live.twitch.tv/app/<stream key>
```

Streaming a file

```
$ ffmpeg -re -i input.mkv -c:v libx264 -preset veryfast -b:v 3000k -maxrate 3000k \
-bufsize 6000k -pix_fmt yuv420p -g 50 -c:a aac -b:a 160k -ac 2 \
-ar 44100 -f flv rtmp://live.twitch.tv/app/<stream key>
```

Encoding a file for streaming

If your computer is too slow to encode the file on-the-fly like the example above then you can re-encode it first:

```
$ ffmpeg -i input.mkv -c:v libx264 -preset medium -b:v 3000k -maxrate 3000k -bufsize 6
-vf "scale=1280:-1,format=yuv420p" -g 50 -c:a aac -b:a 128k -ac 2 -ar 44100 file.flv
```

Then stream copy it to the streaming service:

```
$ ffmpeg -re -i file.flv -c copy -f flv rtmp://live.twitch.tv/app/<stream key>
```

Outputting to multiple streaming services & local file

You can use the tee muxer to efficiently stream to multiple sites and save a local copy if desired. Using tee will allow you to encode only once and send the same data to multiple outputs. Using the `onfail` option will allow the other streams to continue if one fails.

Note: Since ffmpeg has a 'different relationship' with each rtmp server, the long-term headers need to be "out of band" at the container level inthe "extradata". This is achieved with the global_header flag which very likely will be necessary based upon your encoding options as documented below.

```
$ ffmpeg -i input -map 0 -c:v libx264 -c:a aac -b:v 1000k -maxrate 1000k -bufsize 2000
"[f=flv:onfail=ignore]rtmp://facebook|[f=flv:onfail=ignore]rtmp://youtube|local_file.m
```

Some encoders may need different options depending on the output format; the auto-detection of this can not work with the tee muxer, so they need to be explicitly specified. The main example is the **global_header** flag.

```
$ ffmpeg -i input -map 0 -flags +global_header -c:v libx264 -c:a aac -b:v 1000k -maxra
"[f=flv:onfail=ignore]rtmp://facebook|[f=flv:onfail=ignore]rtmp://youtube|local_file.m
```

Notes

- Linux users can use `xwininfo | grep geometry` to select the target window and get placement coordinates. For example, an output of `-geometry 800x600+284+175` would result in using `-video_size 800x600 -i :0.0+284,175`. You can also use it to automatically enter the input screen size: `-video_size $(xwininfo -root | awk '/-geo/{print $2}')`.
- The pulse input device (requires `--enable-libpulse`) can be an alternative to the ALSA input device, as in: `-f pulse -i default`.

- Windows users can use dshow or gdigrab. macOS can use avfoundation. See FFmpeg Wiki: Capture Desktop for examples.

Help

Always use a recent `ffmpeg`. See the compilation guides for more information.

If you need additional help then send a message to the ffmpeg-user mailing list (see Mailing List FAQ first) or hang out in the *#ffmpeg* IRC channel on Freenode.

Dernière modification le 29 oct. 2020 07:16:10)

t a g s

rtmp screencasting streaming twitch
vbv youtube