|  |
| --- |
| Before you begin, please open the **“08 Report” Moodle Assignment Form** allowing you to obligatorily record your answers during the exercise.  **The answers do not need to be verbose; they are just needed to justify that a certain step was completed.** |

Exercise 1 – FFmpeg Refresher

**Foreword**:

(As some of you might recall from previous classes) **FFmpeg** serves as a very powerful framework allowing for various operations (transcoding (de), multiplexing, streaming to name a few) For this exercise, one will need the following tools installed:

* **ffmpeg**
* **ffprobe**

Initially, check on your computer if relevant instances (along with the required packages) of both tools have been installed. If not, follow the steps described in <https://www.ffmpeg.org/download.html> according to the OS version you are working on. Check with the following details provided for **Ubuntu OS**:

>> **apt-get** install yasm

>> **apt-get** install libx264-dev

>> **apt-get** -y install autoconf automake build-essential libass-dev libfreetype6-dev libsdl1.2-dev libtheora-dev libtool libva-dev libvdpau-dev libvorbis-dev libxcb1-dev libxcb-shm0-dev libxcb-xfixes0-dev pkg-config texinfo zlib1g-dev

>> **apt-get** update

>> **apt-get** install ffmpeg

Upon successful installation, you should be able to run the command as follows:

>> **ffmpeg** –help

If you encounter any issues, follow the steps provided on:

<https://trac.ffmpeg.org/wiki/CompilationGuide/Ubuntu#ffmpeg>

**Please take care while copying commands from this file - character encoding may substitute some text!**

**[ACTIVITY 1.0]**

Collect preliminary data. For the next exercise(s) you’ll need to identify the following parameters:

1. **Your display dimension & name** - you can gather that data using xdpyinfo command.
2. **Video size** - based on [FFMPEG documentation](https://ffmpeg.org/ffmpeg-utils.html), find the name corresponding (or similar - e.g., a bit smaller - if you are unable to find your resolution) to your screen dimension, i.e., if your screen size is *4096x2160* your video size would be ‘4K’ :-)
3. **video codecs** - find the names corresponding to H.264, H.265, and VP9 codecs

**[ACTIVITY 1.1]**

Screen capture. Run the following command. Remember about filling out all necessary parts (bolded) with the information that you gathered previously. For the **video, the codec** picks up H.264.

>> **ffmpeg** –framerate 30 -video\_size [**display\_dimension**] -f x11grab -i [**display\_name**] -codec:v [**video codec**] -acodec libfdk\_aac -preset ultrafast [**fileName**].mp4

***Note****! You need only around 15-20 seconds of recording. Press “Q” to stop capturing.*

It's also perfectly possible to capture the screen using ffmpeg using the "avfoundation" device.

To get a list of available sources:

ffmpeg -f avfoundation -list\_devices true -i ""

Then to capture:

ffmpeg –vsync 2 -f avfoundation -i "<screen device index>:<audio device index>" <destination filename>

**[ACTIVITY 1.2]**

Check on the details of the created derivative (fileName.mp4) by running the following command. We’ll come back to this section later.

>> **ffmpeg** -i fileName.mp4

***Note****! Please, remember about pasting the output of the command to Moodle Assignment Form with answers.*

Exercise 2 – Leading Codecs

**Foreword**: Every year, one of the leading transcoding companies, Encoding.com, publishes a very meaningful report highlighting key trends in the video transcoding & packaging market during the past 12 months.

Here you can access the latest report for the last year: [**Encoding.com report**](https://www.encoding.com/resources/#global-formats-report) **(Top of the list)**

**[ACTIVITY 2.0]**

Read through the report and figure out which **video** & **audio** **codecs** were leading during the last year.

***Note****! Please, remember about pasting your answer to the Moodle Assignment Form.*

**[ACTIVITY 2.1]**

Based on the lecture of the report, figure out the **top 3** Cloud Storage Providers.

***Note****! Please, remember about pasting your answer to the Moodle Assignment Form.*

**[ACTIVITY 2.2]**

There is a separate **paper** dedicated to the comparison of H.264 and H.265 (HEVC) codecs. You can find in its PDF version online here: [H.264-vs-H.265.pdf](https://github.com/0voice/audio_video_streaming/blob/main/paper/ffmpeg/H264/H.264-vs-H.265.pdf)

Please find in this paper, by how much (in %) the **target bandwidth** can be decreased while using H.265 as opposed to H.264.

***Note****! Please, remember about pasting your answer to the Moodle Assignment Form.*

**[ACTIVITY 2.3]**

Now, let’s come back to your original content (screen capture from Activity 1.1) that was initially encoded with H.264. Run the following command to create a new file encoded in H.265.

***Note****! Don’t forget to update the* ***video codec*** *name for H.265!*

>> **ffmpeg** -i [**fileName**].mp4 -codec:v [**video codec**] [**fileName\_h265**].mp4

Now, once it’s done, please compare the size of both files (by indicating % of compression) of both files: **fileName.mp4** and **fileName\_h265.mp4**. It is an indication of how many percent the new file (H.265) has a smaller size than the original file (H.264).

***Note****! Please, remember about pasting your answer on the Moodle Assignment Form.*

Exercise 3 – HLS Playlist

**Foreword:**

Apple maintains its own RFC for the HLS format. Timely updated for detailed reading can be found here: <https://tools.ietf.org/html/rfc8216>

FFMPEG contains a dedicated multiplexer that segments MPEG-TS files according to the HTTP Live Streaming (HLS) specification. It results in M3U8 playlist files and one or more segment (MPEG-TS) files created with respect to the RFC.

**[ACTIVITY 3.0]**

First of all, let’s download any exemplary video file that will serve us as “original content” we want to stream. You can either download some sample.MP4 from [Big Buck bunny movies](https://download.blender.org/peach/bigbuckbunny_movies/) OR you can make another screen capture attempt (similarly to how it was done in Activity 1.1).

>> **ffmpeg** -i [**sourceFile**].mp4 –s 1280x720 -hls\_segment\_filename 'segment\_%03d.ts' -hls\_allow\_cache 1 -hls\_time 6 –hls\_list\_size 0 hsl\_playlist.m3u8

***Note****! If you decide on the latter option, please record at least 40-50 seconds.*

***Note****! Please, remember about pasting your resulting HLS playlist (generated hls\_playlist.m3u8 file) to the Moodle Assignment Form.*

Exercise 4 – Micro-Project

**Foreword**: The goal of this mini project is to let you build and customize your own **Web Player** that allows for the playback of HTTP Live Streaming. We will work with **hls.js** – a JavaScript library that implements an HLS Client and relies on HTML5 video tags as well as MSE (Media-Source Extension).

**URL to hls.js README**: <https://github.com/video-dev/hls.js/blob/master/README.md>

**URL to hls.js DEMO player**: <http://video-dev.github.io/hls.js/demo>

High-level requirements:

**MUST HAVE**:

* Allows for successful HLS playback (e.g., playlists created during the class).
* Contains a predefined list of 2-3 HLS playlists (it can be a drop-down menu OR an ordered list).
* Allows for manual switch between bitrates (i.e., buttons).
* Displays basic information about the video (bitrate, height, width, video/audio codec).

**NICE HAVE:**

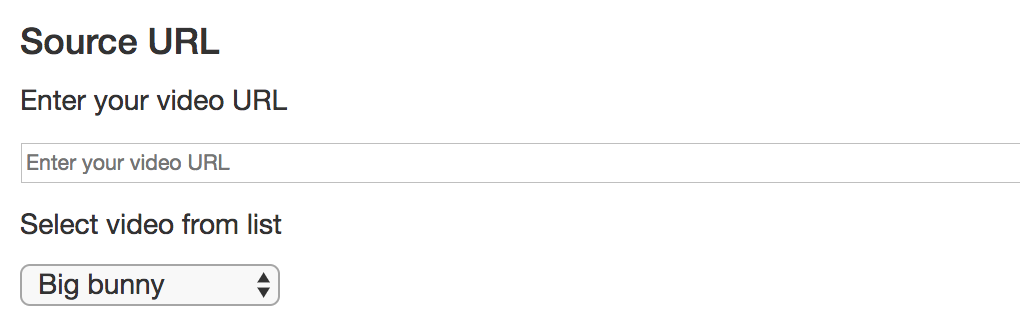
* displays all URLs being requested by the player “on-the-fly” (tracking requests);
* Provides access to the “Premium” Content by introducing GEO-based control.
  + Whitelist (or blacklist) access to the content based on IP of the requestor.

***Hint****! To implement the GEO logic, you can either use the geo-plugin API or parse the response:*

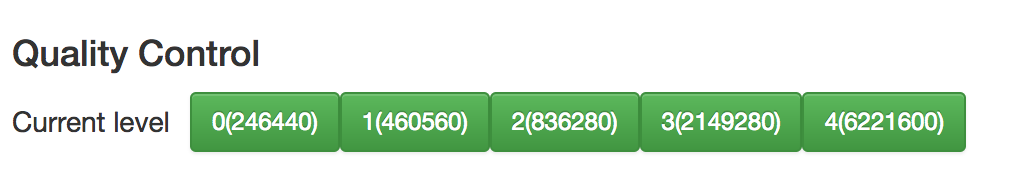
*(*[*http://www.geoplugin.net/json.gp?ip=1.2.3.4*](http://www.geoplugin.net/json.gp?ip=1.2.3.4)*) or leverage PHP GEO library: (*[*https://www.php.net/manual/en/book.geoip.php*](https://www.php.net/manual/en/book.geoip.php)*)*

Example of requirements:

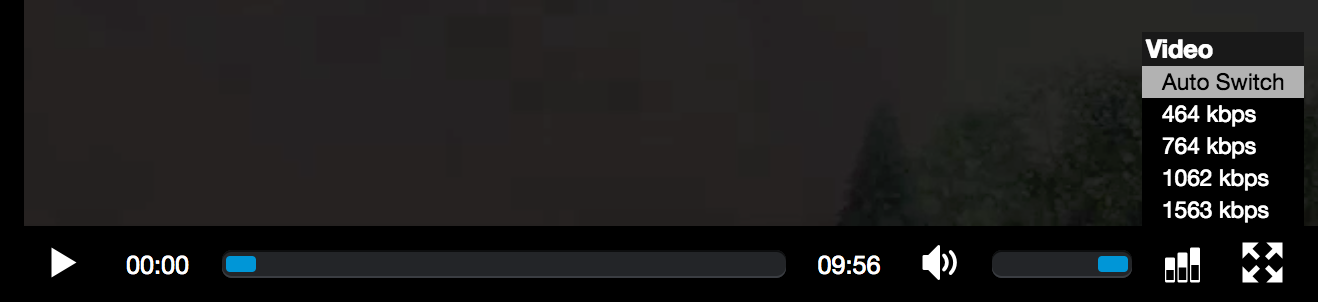
* Player contains a **selectable list** of predefined HLS playlists as well as **input fields**. The goal is to have at least 3-4 different videos available.



* The player allows for **manual switching** between available bitrates. It can either be a set of buttons (see screen 1) or built-in options in the form of a selectable list in the playback window (see screen 2):

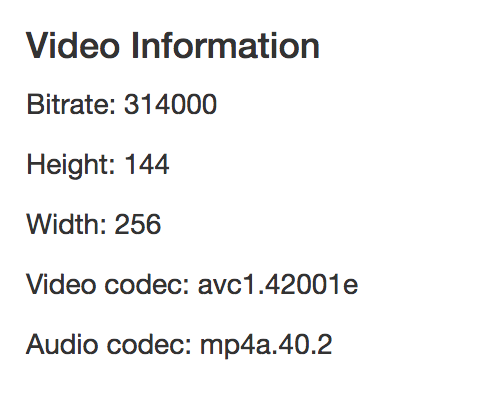


Screen 1. Multiple buttons for different bitrates.



Screen 2. A built-in selectable list of bitrates in the playback window.

* Displays basic information about the video. This includes at least: bitrate, height, width, video/audio codecs.



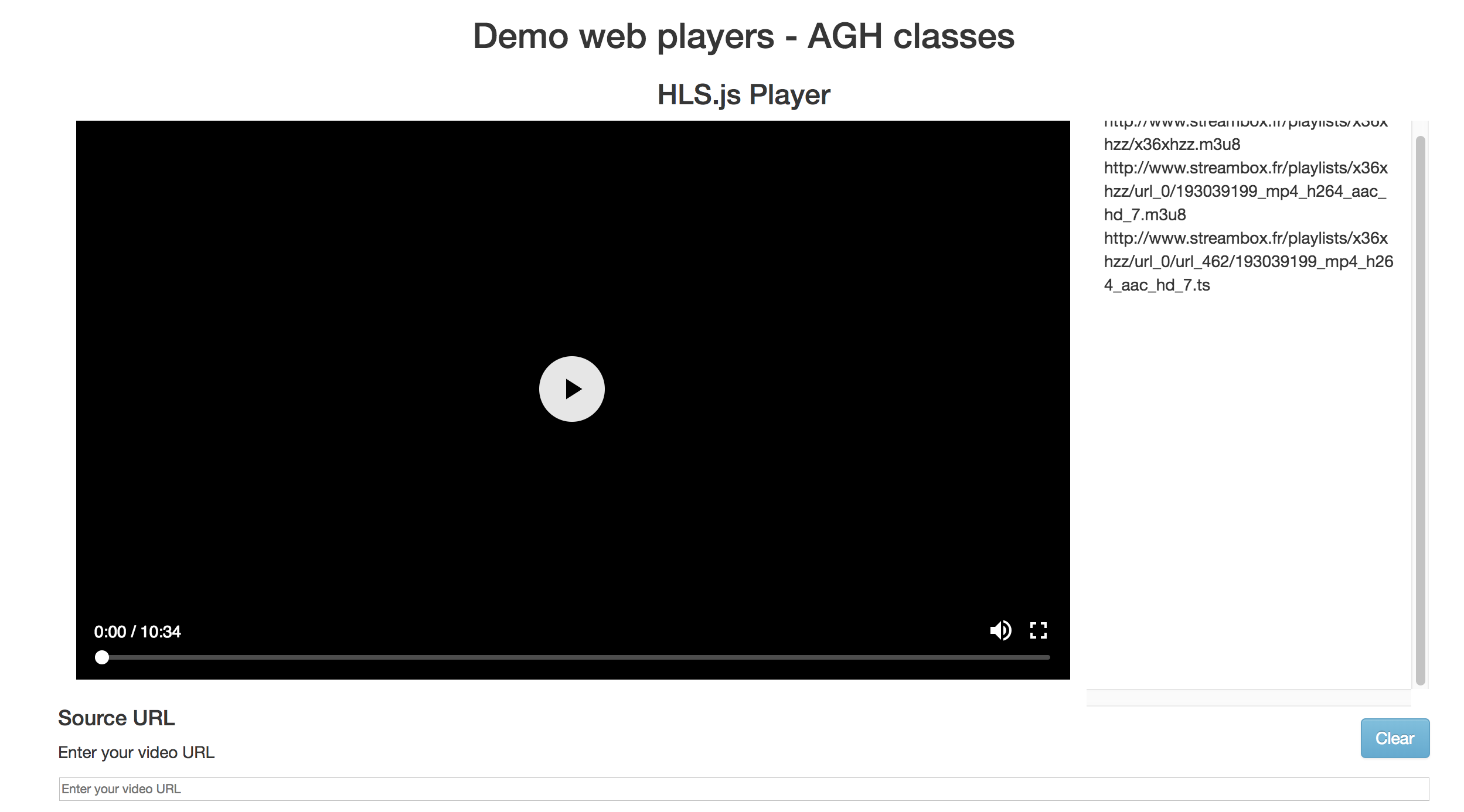
***Hint****! Use an Hls* ***object*** *that contains different controllers, including* ***level Controller****. For detailed use-case use developer tools (e.g., Firebug) to review HLS.jd Demo player.*

* Displays all URLs requested by the player “on-the-fly.”



**DEMO TIME**!

The presenters will now make a quick demo of the prepared exemplary players, including the features described above.



**How to send your project to us**:

* Upload your **player file** (along with all embedded objects like JavaScript or CSS) to any Web Server you have access to (or GitHub). For example, you can use pluton.kt.agh.edu.pl/~<yourLogin>. In the case of .php files, please make sure we can also see the source code.

* Don’t forget to upload your **HLS playlist files** created during the class (or any other created for the project);
* Ultimately, provide us with **the link to your repository** on the server by filling out the following **Moodle Assignment Form**:

|  |
| --- |
| **Deadline**: Hard deadline to send us your projects is **the last day of the semester** until 23:59 CEST |

--- APPENDIX ---

1. Here is an example of how you can try the playback of newly created HLS playlists. Firstly, we’ll use VLC Media Player (<http://www.videolan.org/index.pl.html>) with the following guidance to install it:

*Login root*

>> **apt-get** install vlc

*Logout from root*

>> vlc [**path to the created M3U8 playlist file**]

2. The Aim of this supplementary part is to explain how to track the HTTP request/response flow during the delivery of, i.e., HLS Streaming. For this purpose, one will need:

* Google Chrome or Mozilla Firefox (regardless of OS on your machine)
* Browser add-on allowing to track HTTP requests (i.e., Firebug for Firefox)
* Terminal on Your Machine

In any of - mentioned above - browsers, enter. <https://www.hlsplayer.net/> (Or any other sample HLS player available on the Internet). This page contains a web player capable of understanding and properly playback HLS streaming (both VOD and Live).

Before starting playback of the player URL, activate relevant add-ons:

* if working with Mozilla Firefox, run Firebug. It’s usually available through: **Firefox** -> **View** -> '**Firebug**' (or simply press **F12)**
* if working on Google Chrome, select **View** -> **Developer** -> '**Developer Tools**' (or try to press F12) In both cases, switch to '**Network**' tab, that will display all HTTP requests sent by your browser from now.

Start the playback of the test stream (press 'Play') and make note of the requests sent from the browser.

To understand the better the order that Web browser does, use ***the curl*** command to see how the playlist file looks like:

>> **curl** <http://www.streambox.fr/playlists/test_001/stream.m3u8>

or

>> **curl** <https://mnmedias.api.telequebec.tv/m3u8/29880.m3u8>