H.264 Recorder for Mac

Installing software

Download the application from https://github.com/learninglabdtu/OpenProjects/tree/master/H264%20Recorder

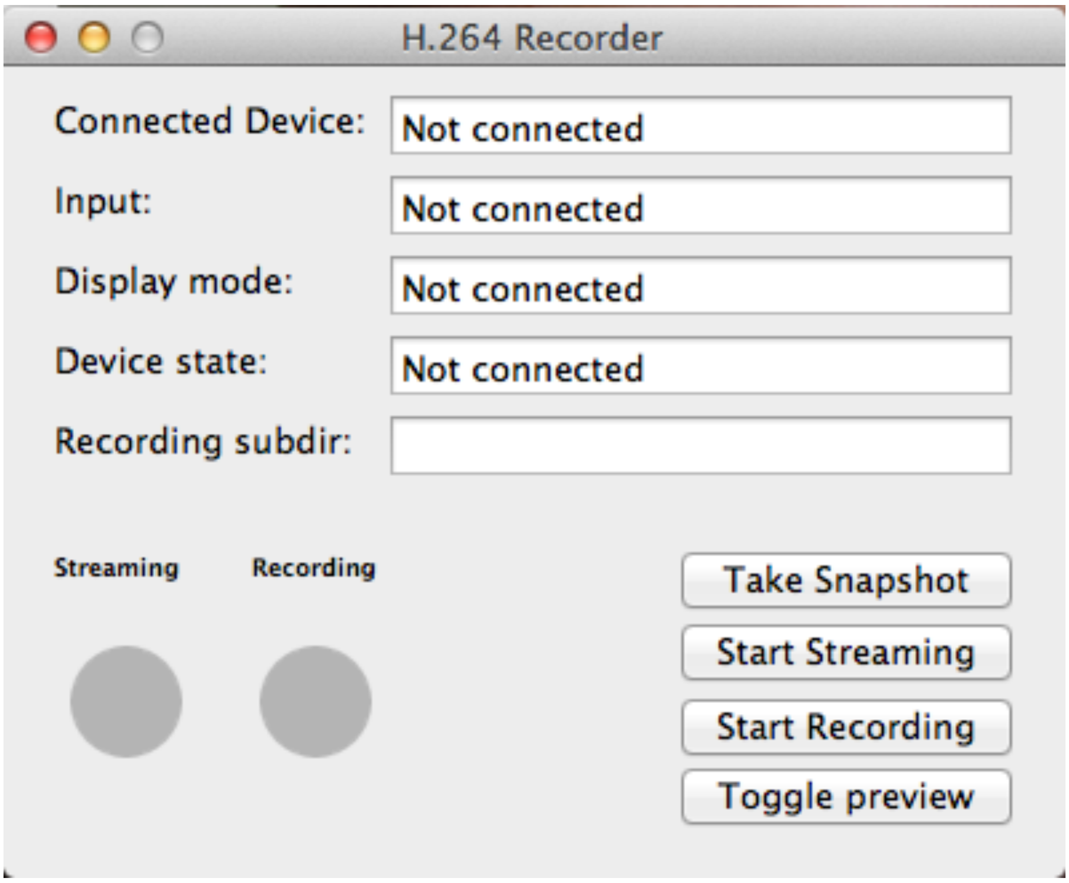
Copy "H.264 Recorder" to "Applications"

Install BlackMagic Designs software for the H.264 recorder (https://www.blackmagicdesign.com/support). This software download is called "Desktop Video" which includes Media Express that works as the middleman between the hardware and this application). The Mac will restart.

Install VLC (http://www.videolan.org/vlc/)

Connecting and confirming signal

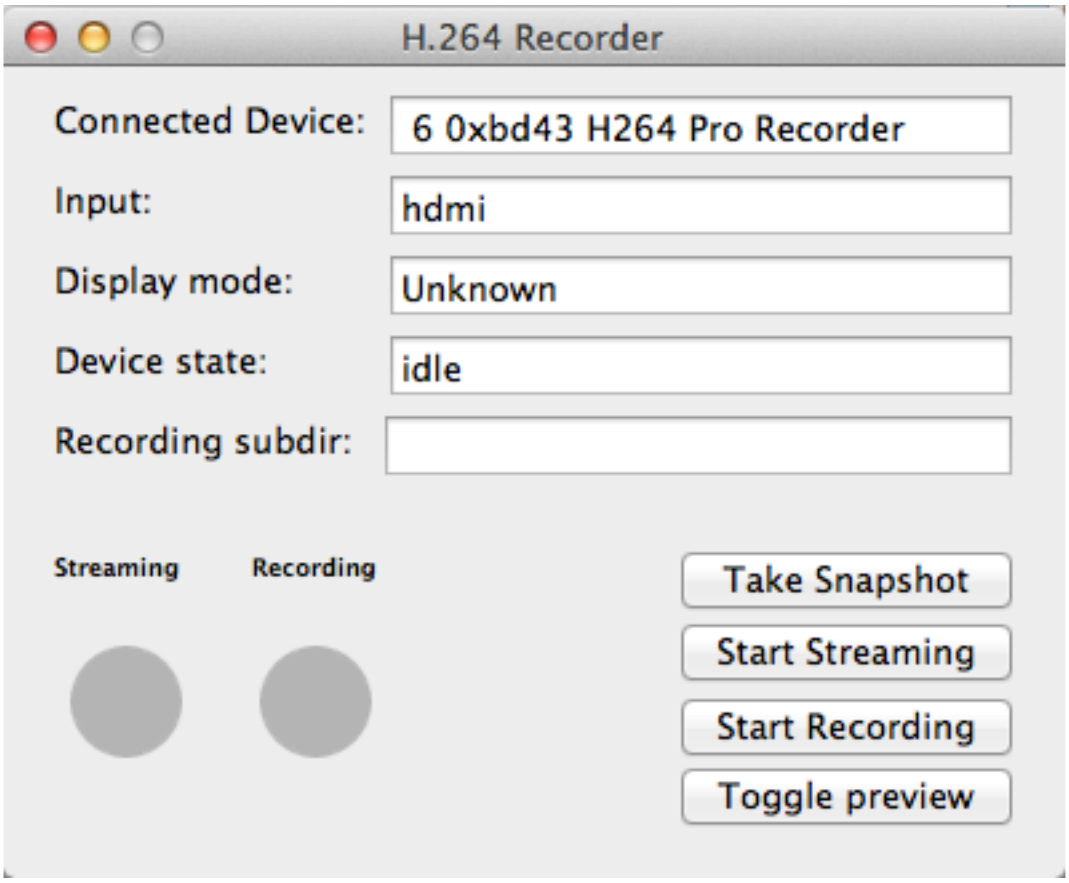
When you start "H.264 Recorder", you'll see this:



H.264 Recorder

Connected Device: Not connected
Input: Not connected
Display mode: Not connected
Device state: Not connected
Recording subdir:

After connecting a H.264 recorder or a ATEM Television Studio to the computers USB port, you'll see this:


H.264 Recorder

Connected Device: 6 0xbd43 H264 Pro Recorder

Input: hdmi

Display mode: Unknown

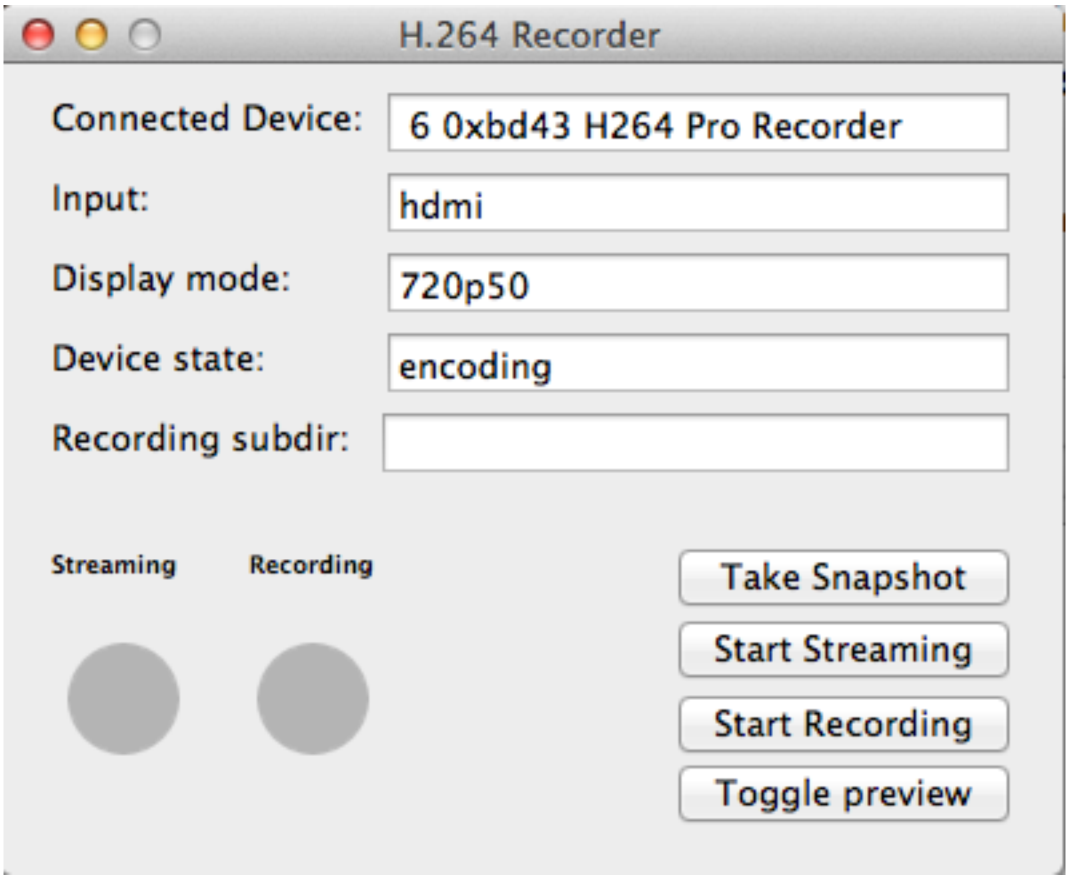
Device state: idle

Recording subdir:

(You may need to restart the application)

The Input field will show that the hardware is searching for an input signal.

When an input signal is present, it'll look something like this:



H.264 Recorder

Connected Device: 6 0xbd43 H264 Pro Recorder

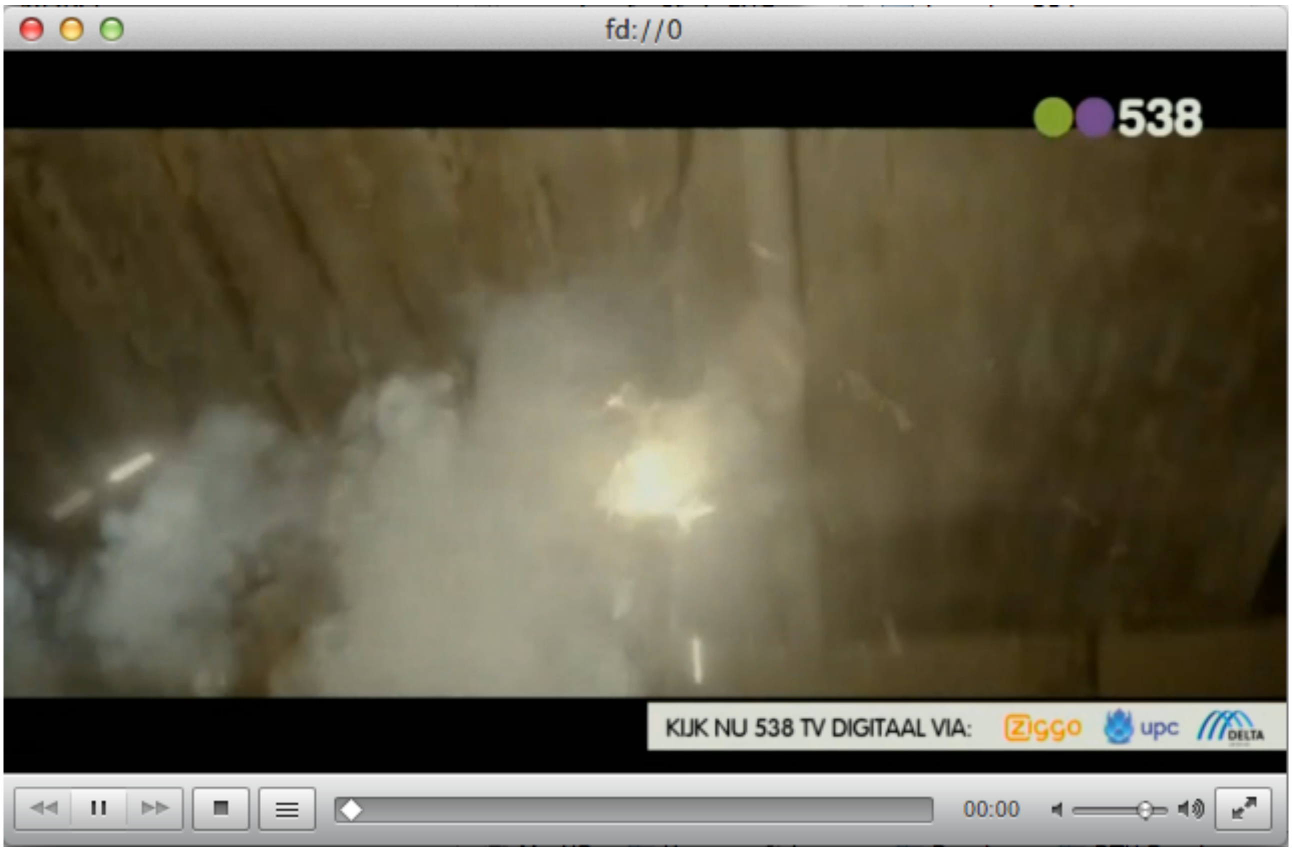
Input: hdmi

Display mode: 720p50

Device state: encoding

Recording subdir:

Press “Toggle Preview” to see what is coming in (requires VLC):

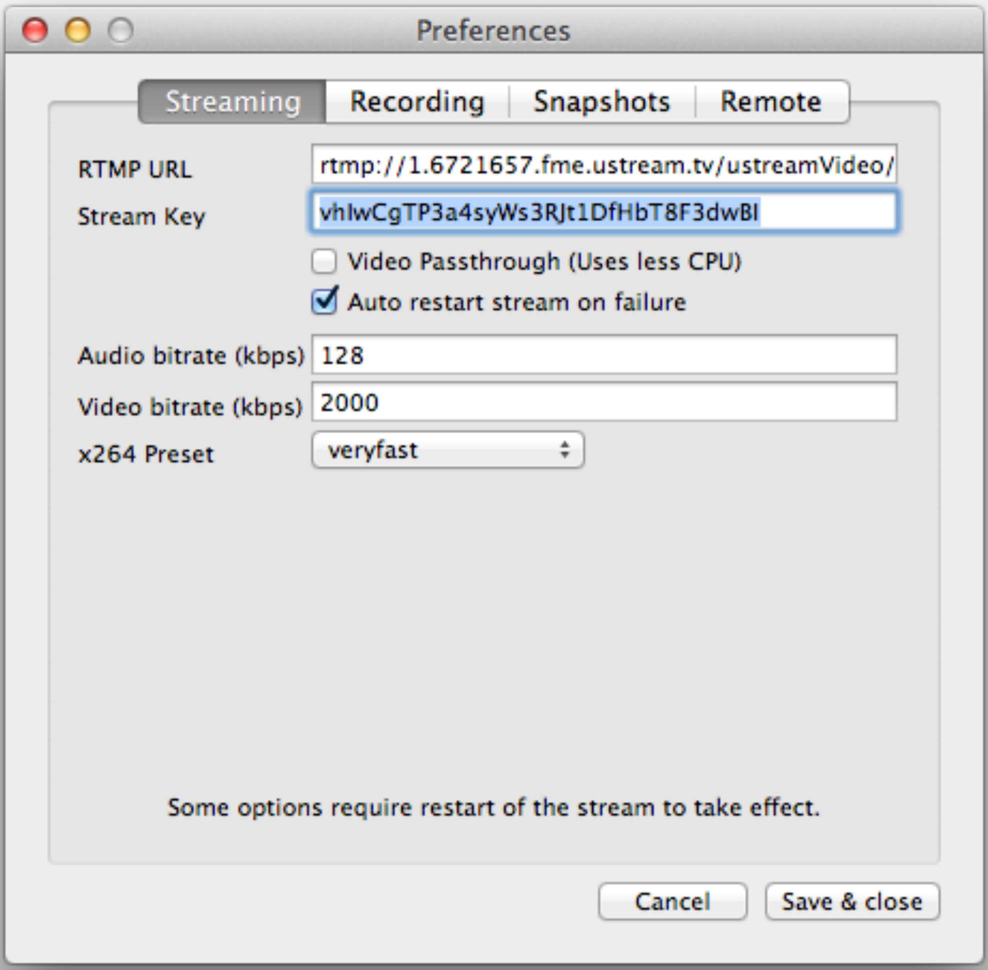


Warning: For technical reasons DO NOT press the pause button in the VLC preview window!

Preferences

Streaming preferences

Set up preferences



RTMP URL: rtmp://1.6721657.fme.ustream.tv/ustreamVideo/

Stream Key: vhlwCgTP3a4syWs3RJt1DfHbT8F3dwBI

☐ Video Passthrough (Uses less CPU)

☑ Auto restart stream on failure

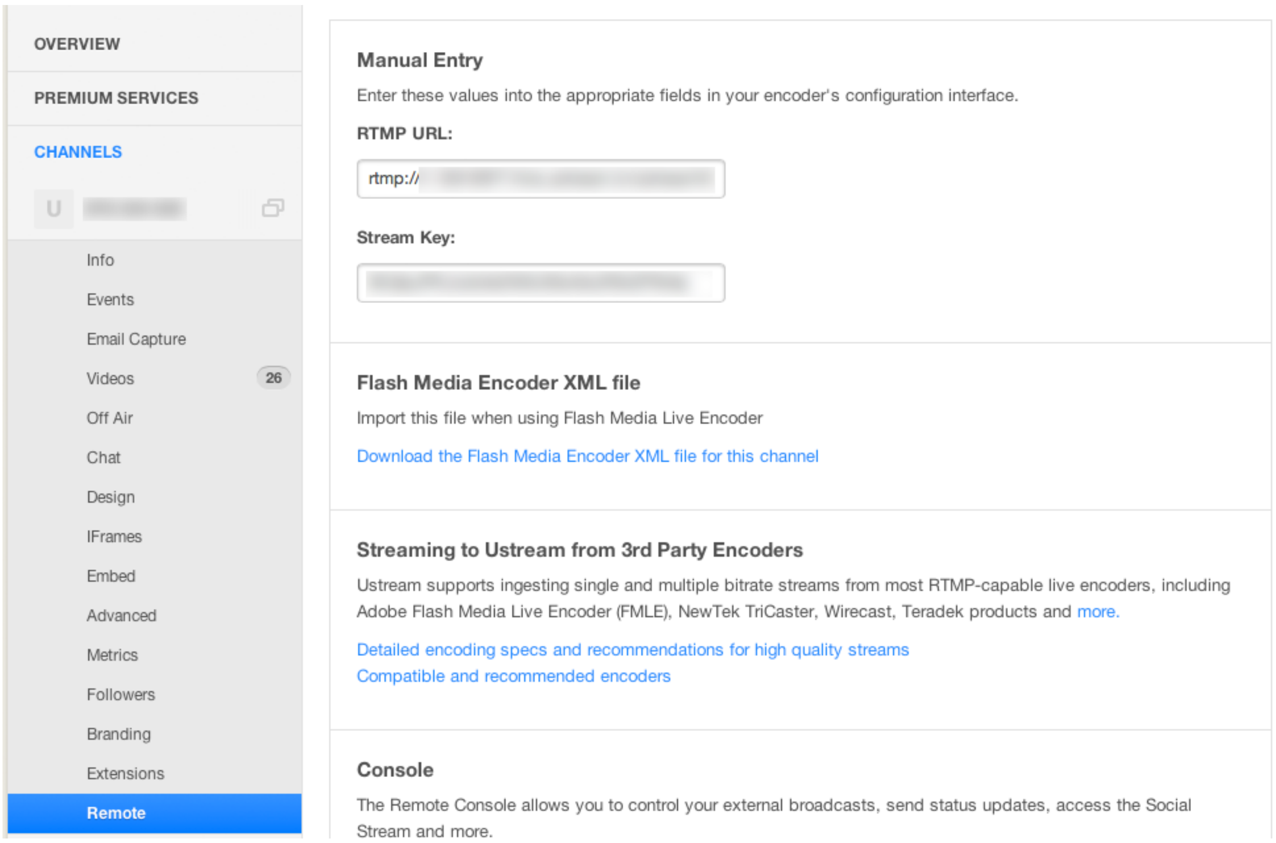
Audio bitrate (kbps): 128

Video bitrate (kbps): 2000

x264 Preset: veryfast

Some options require restart of the stream to take effect.

The RTMP URL and Streaming Key can typically be seen somewhere with the streaming server provider. In this case it’s ustream and they would display it like this:



Manual Entry

Enter these values into the appropriate fields in your encoder's configuration interface.

RTMP URL:

rtmp://

Stream Key:

Flash Media Encoder XML file

Import this file when using Flash Media Live Encoder

Download the Flash Media Encoder XML file for this channel

Streaming to Ustream from 3rd Party Encoders

Ustream supports ingesting single and multiple bitrate streams from most RTMP-capable live encoders, including Adobe Flash Media Live Encoder (FMLE), NewTek TriCaster, Wirecast, Teradek products and more.

Detailed encoding specs and recommendations for high quality streams

Compatible and recommended encoders

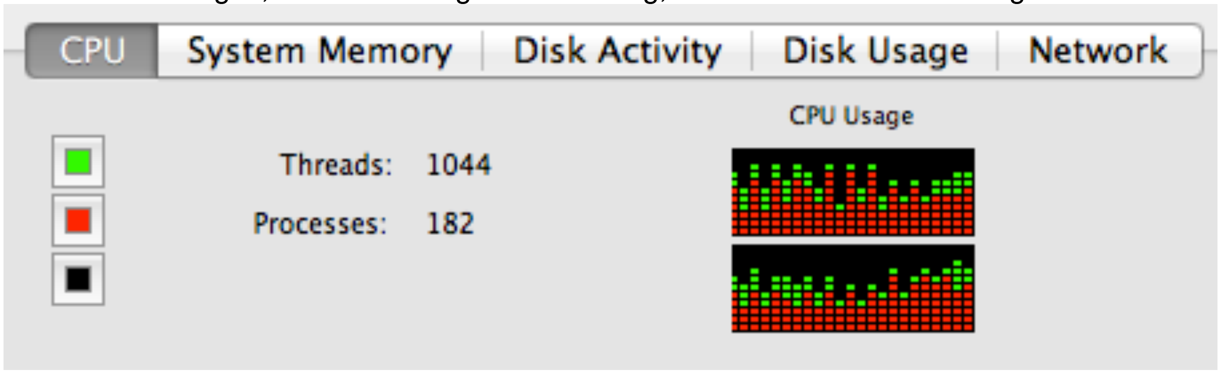
Console

The Remote Console allows you to control your external broadcasts, send status updates, access the Social Stream and more.

It's under Channels > Remote that you find it.

Using ffmpeg, the H.264 Recorder can transcode the input from H.264 recorder to a lower bitrate for streaming. This is what the Audio and Video bitrates are for in this screen. If "Video Passthrough" is enabled, no such transcoding will occur, thus saving processor cycles.

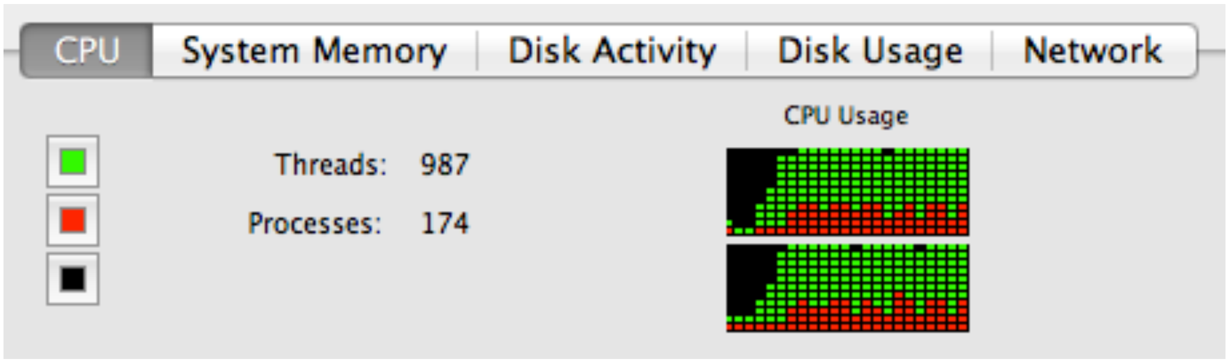
This is CPU usages, both streaming and recording, but with "Video Passthrough" enabled:



Threads: 1044

Processes: 182

This is CPU usages, both streaming and recording, but with "Video Passthrough" disabled (transcoding):



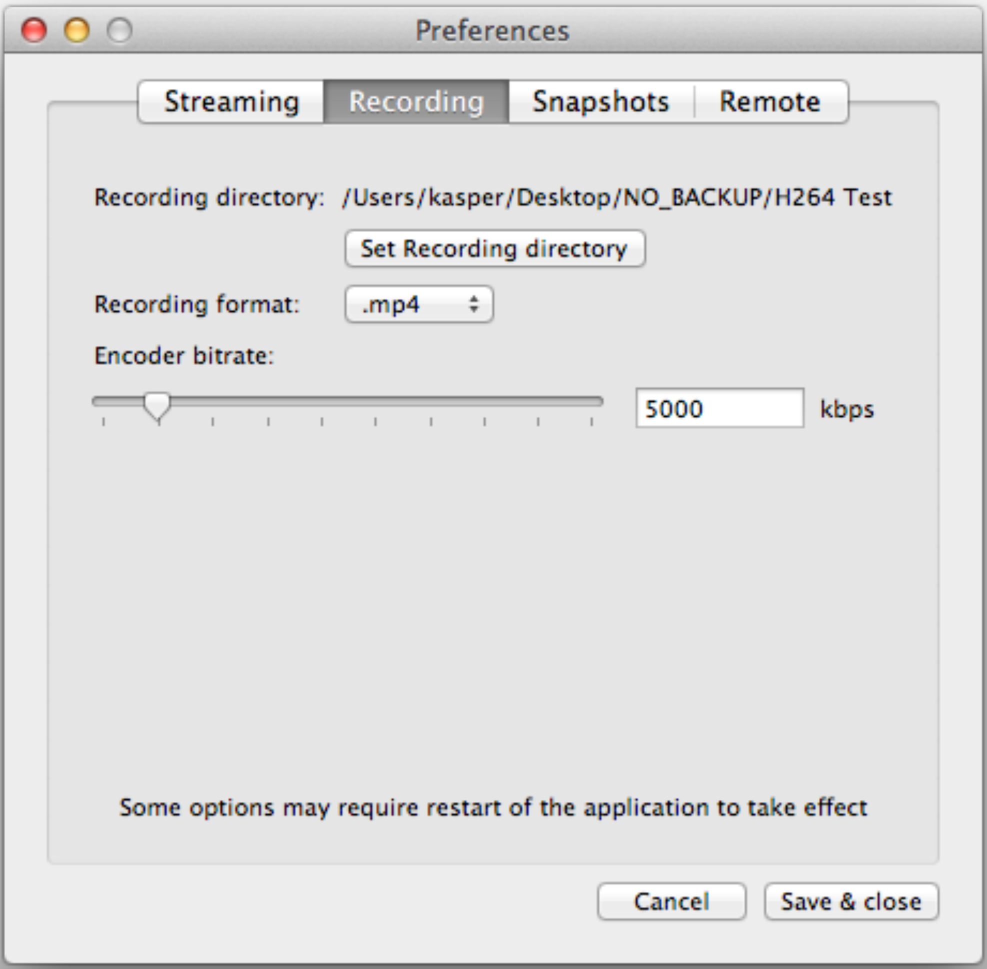
Threads: 987

Processes: 174

Make sure to play a bit with the various transcoding settings and see what results you get.

Recording Preferences

Settings for recording looks like this:



Recording directory: /Users/kasper/Desktop/NO_BACKUP/H264 Test

Recording format: .mp4

Encoder bitrate: 5000 kbps

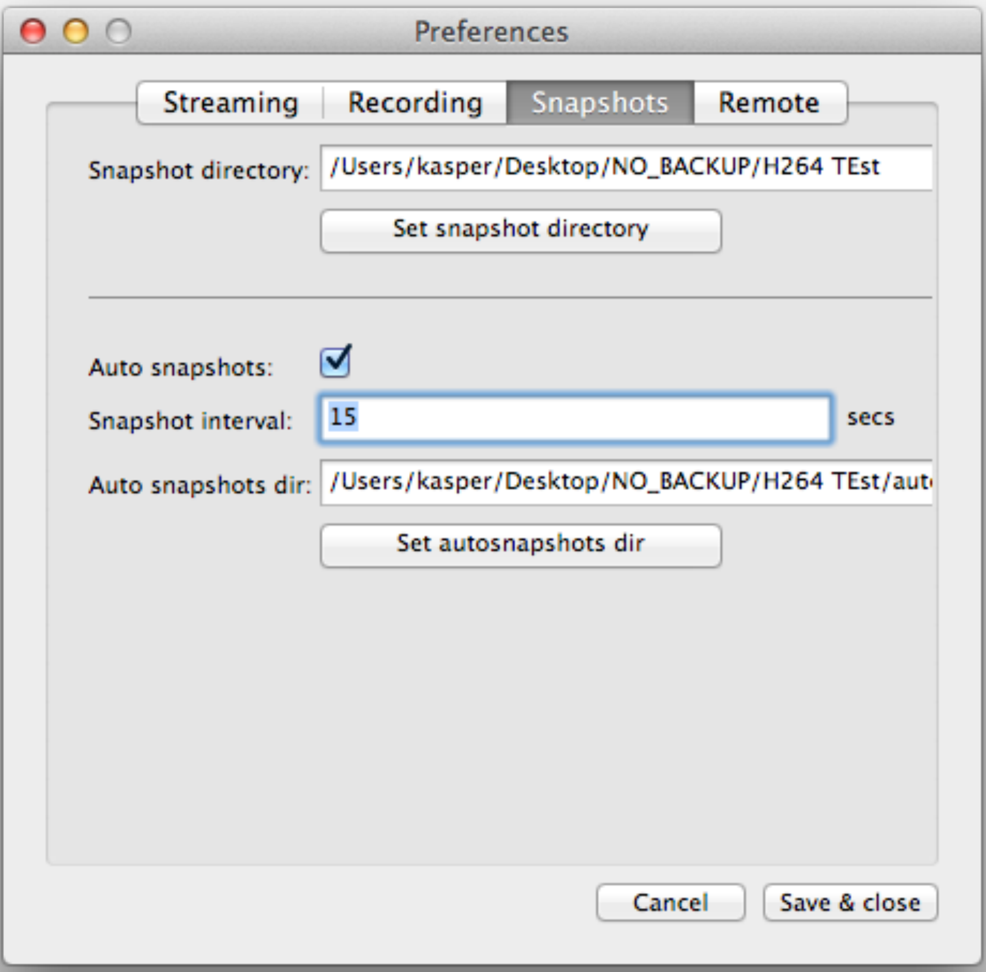
Some options may require restart of the application to take effect

You need to set a directory in which to store the MP4 files and snapshots.

When you start a recording, the file is named “Recording.[ext]”, but when you stop, the file is renamed to “video-[timestamp].[ext]”.

Snapshot Preferences

The app can make still snapshots as well as periodic snapshots with a time interval. You must configure a folder where these are stored.



Snapshot directory: /Users/kasper/Desktop/NO_BACKUP/H264 TEst

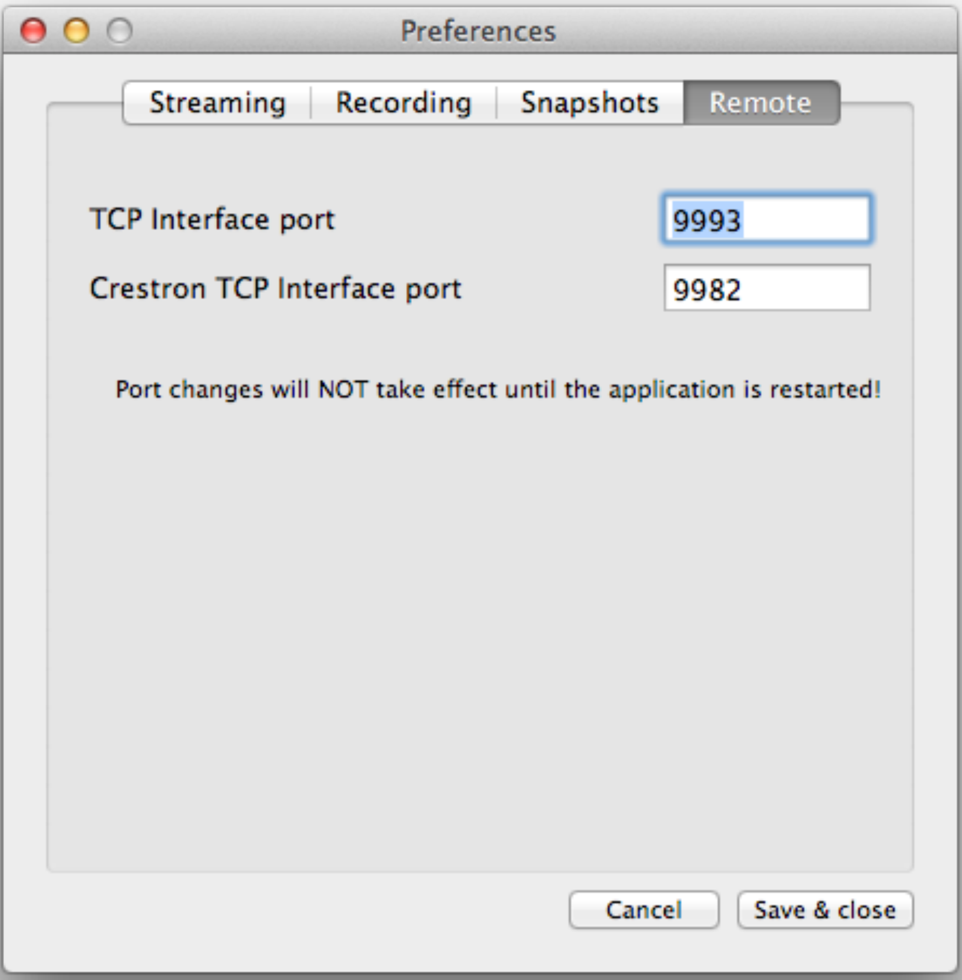
Auto snapshots: ✓

Snapshot interval: 15 secs

Auto snapshots dir: /Users/kasper/Desktop/NO_BACKUP/H264 TEst/aut

Remote Preferences

Here you configure which ports you can access the application on for remote control via a TCP interface (using a terminal).



TCP Interface port 9993

Crestron TCP Interface port 9982

Port changes will NOT take effect until the application is restarted!

"Recording subdir"

Recording subdir is a feature where you can have recordings and snapshots stored in a subfolder under the Recording Directory. The folder is created if it doesn't exist as soon as you start the recording and when you stop recording, the file "Recording.[ext]" is moved into that folder and renamed to "video-[timestamp].[ext]".

You will also find a file "notifyUser.json" in this folder which contains at least the information "username" = [name of subdir]. This is because the subdir feature is linked to the variable "username" used in the TCP interfaces (see below).

TCP control (One-Shot)

The app can be controlled via telnet. You must accept incoming network connections for the app to do so. The "One-Shot" connection will connect, wait for your command, respond and close the connection again.

Open a Terminal and type (assuming the port number is the default):

```
nc localhost 9993
```

Then you type and press enter:

```
IS_RECORDING
```

The response should reflect whether the application records or not.

To start a recording and check it has been started, a communication example could look like this:

```
Kaspers-MacBook-Pro-2:H.264 Recorder kasper$ nc 10.15.20.156 9993
IS_RECORDING
FALSE
Kaspers-MacBook-Pro-2:H.264 Recorder kasper$ nc 10.15.20.156 9993
START_RECORDING
OK
Kaspers-MacBook-Pro-2:H.264 Recorder kasper$ nc 10.15.20.156 9993
IS_RECORDING
TRUE
Kaspers-MacBook-Pro-2:H.264 Recorder kasper$
```

Commands (One-Shot):

The server recognises a set of ASCII encoded commands, treated in detail below:

IS_STREAMING
returns "TRUE" if there is network activity indicating that application is streaming, and "FALSE" otherwise

IS_RECORDING
returns "TRUE" if the application is recording, "FALSE" otherwise.

START_STREAMING
Attempts to start streaming. If successful (same criteria as IS_STREAMING), "OK" is returned. Otherwise "ERROR" is returned.

STOP_STREAMING
Stops broadcasting. If the script cannot detect any broadcasting afterwards (same criteria as IS_STREAMING), "OK" is returned. Otherwise "ERROR" is returned.

START_RECORDING
Attempts to start recording. If successful (same criteria as IS_RECORDING), "OK" is returned. Otherwise "ERROR" is returned.
You can specify additional parameters on the form [parameter1]=[value]:[parameter2]=[value]:… These parameters will end up in the "notifyUser.json" file in the recording subdir. The only *required* parameter is the one named "username" which will also set the recording subdirectory. "email" is another parameter which the system will recognize and let you query. Any other parameters is to your liking, this feature can be used in custom workflows. For example, "START_RECORDING:username=kasper:email=kaska@llab.dtu.dk" will create a recording in the subdir "kasper/" and put a json file in there with the contents

```
{
      "email" : "kaska@llab.dtu.dk",
      “username” : “kasper”
}
```

STOP_RECORDING

Stops recording. If successful "OK" is returned, and in other cases "ERROR" is returned.

START_AUTOSNAPSHOTS

Attempts to start creating periodic snapshots. If successful, "OK" is returned. Otherwise "ERROR" is returned. Can be used with parameters in the same way “START_RECORDING” is.

STOP_AUTOSNAPSHOTS

Stops periodic snapshots. "OK" or "ERROR" is returned.

IS_AUTOSNAPSHOTS_RUNNING

Returns TRUE if periodic snapshots are being taken.

GET_USER

Returns the 'username' json variable.

GET_EMAIL

Returns the 'email' json variable.

TAKE_SNAPSHOT

Saves a snapshot of the current output scene in wirecast. Returns "OK" if a file is created, and "ERROR" if a file could not be found. Can be used with parameters in the same way “START_RECORDING” is.

CLEAR_USER

Clears all json variables. Always returns "OK"

TCP control (Persistent)

The app can be controlled via telnet in a persistent (continously connected) way. You must accept incoming network connections for the app to do so. The “Persistent” connection will connect, let you type commands, respond to them and also update you of any external changes. It’s possible to have multiple simultaneous connections.

Open a Terminal and type (assuming the port number is the default):

```
telnet localhost 9982
```

Then you type and press enter:

```
RECORD_START
```

The response should be:

```
RECORD_STATUS=STARTED
```

A communication example could look like this:

```
PING
ACK
RECORD_START:username=kasper:email=kaska@llab.dtu.dk:somethingelse=My Text
RECORD_STATUS=STARTED
RECORD_STOP
```

```
RECORD_STATUS=STOPPED
STREAM_START
STREAMING_STATUS=STARTED
STREAMING_STATUS=STOPPED      (external event caused this!)
RECORD_STATUS=STOPPED         (external event caused this!)
```

(Commands typed are in red, black lines is the response from the application)

Commands (Persistent):

PING<cr+lf>

Responds "ACK<cr+lf>"

STREAMING_STATUS?<cr+lf>

returns "STREAMING_STATUS=STARTED" if there is network activity indicating that application is streaming, and "STREAMING_STATUS=STOPPED" otherwise

RECORD_STATUS?<cr+lf>

returns "RECORD_STATUS=STARTED" if the application is recording, "RECORD_STATUS=STOPPED" otherwise.

STREAM_START<cr+lf>

Attempts to start streaming. Returns the streaming status.

STREAM_STOP<cr+lf>

Stops broadcasting. Returns the streaming status.

RECORD_START<cr+lf>

Attempts to start recording. Returns the recording status.

You can specify additional parameters on the form [parameter1]=[value]:[parameter2]=[value]:… These parameters will end up in the "notifyUser.json" file in the recording subdir. The only *required* parameter is the one named "username" which will also set the recording subdirectory. "email" is another parameter which the system will recognize and let you query. Any other parameters is to your liking, this feature can be used in custom workflows. For example, "RECORD_START:username=kasper:email=kaska@llab.dtu.dk" will create a recording in the subdir "kasper/" and put a json file in there with the contents

```
{
      "email" : "kaska@llab.dtu.dk",
      "username" : "kasper"
}
```

RECORD_STOP<cr+lf>

Stops recording. Returns the recording status.

AUTOSNAPSHOTS_START<cr+lf>

Attempts to start creating periodic snapshots. Returns the status for making auto snapshots. Can be used with parameters in the same way "RECORD_START" is.

AUTOSNAPSHOTS_STOP<cr+lf>

Stops periodic snapshots. Returns the status for making auto snapshots.

AUTOSNAPSHOTS_STATUS?<cr+lf>

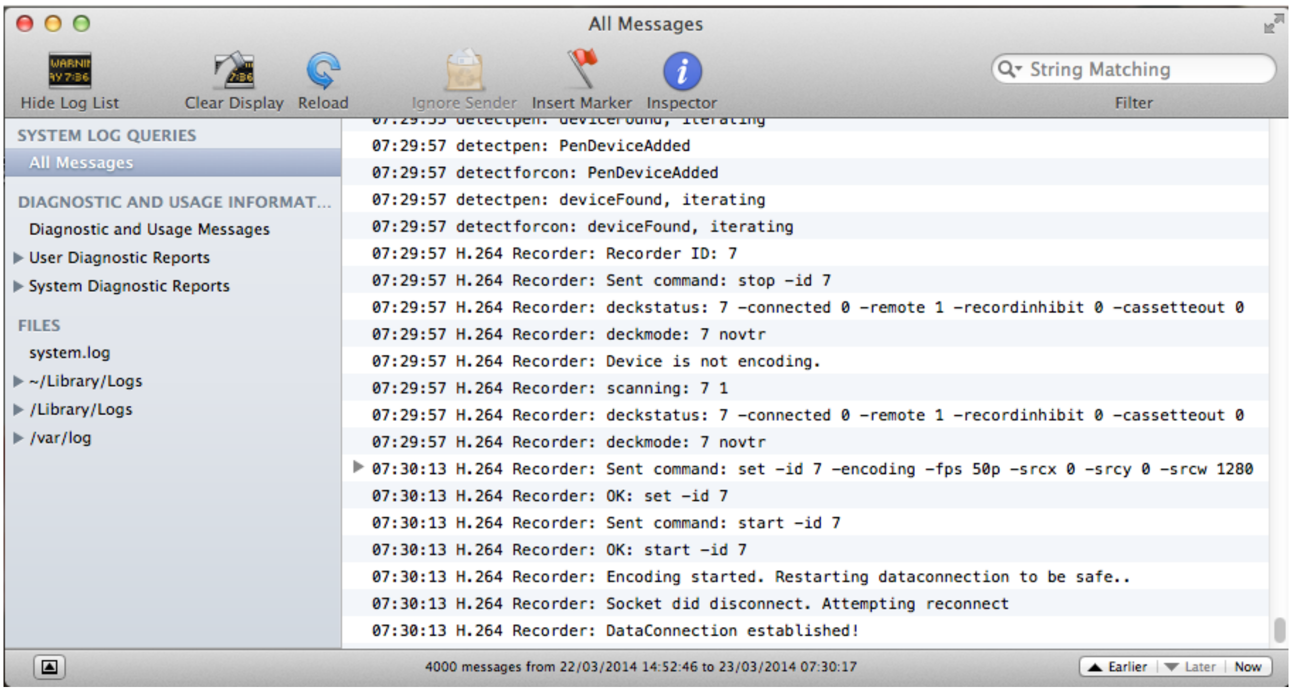
Returns "AUTOSNAPSHOTS_STATUS=STARTED" if periodic snapshots are being taken, otherwise "AUTOSNAPSHOTS_STATUS=STOPPED"

SNAPSHOT_TAKE<cr+lf>

Saves a snapshot of the current output scene in wirecast. Returns "SNAPSHOT_STATUS=EXECUTED" if a file is created. Can be used with parameters in the same way “RECORD_START” is.

Debugging

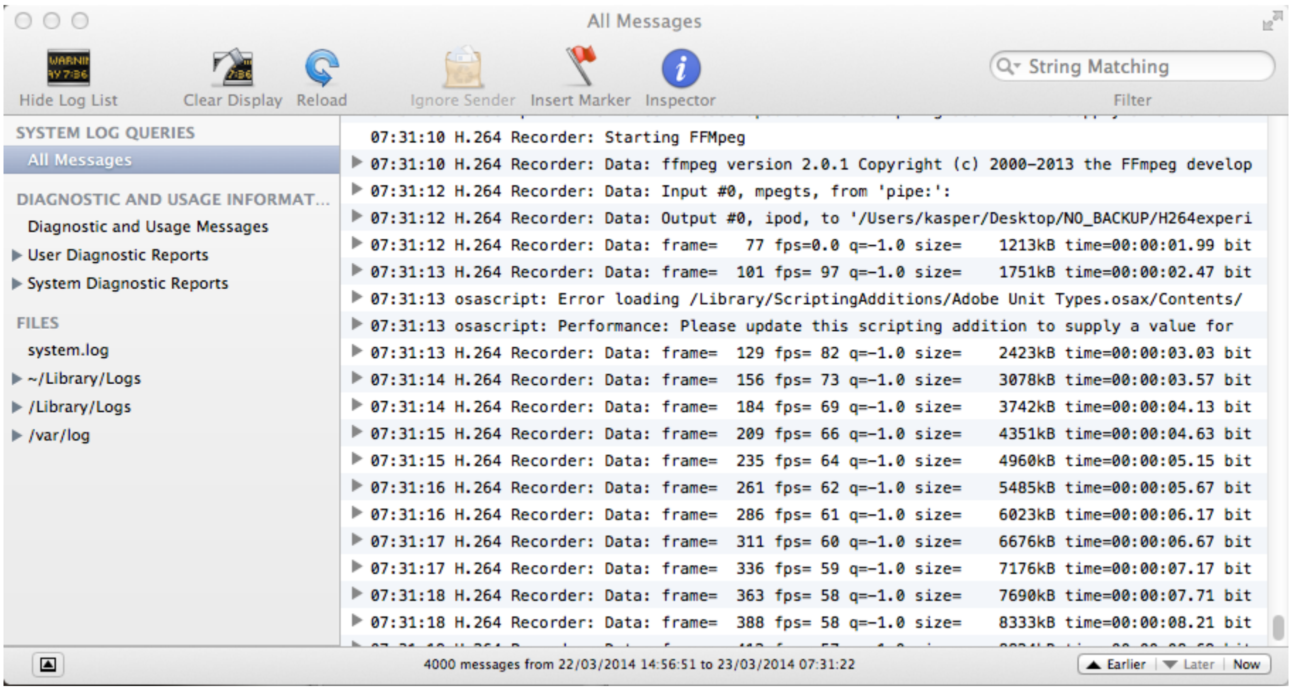
Open Console on your Mac:



```
07:29:57 detectpen: PenDeviceAdded
07:29:57 detectforcon: PenDeviceAdded
07:29:57 detectpen: deviceFound, iterating
07:29:57 detectforcon: deviceFound, iterating
07:29:57 H.264 Recorder: Recorder ID: 7
07:29:57 H.264 Recorder: Sent command: stop -id 7
07:29:57 H.264 Recorder: deckstatus: 7 -connected 0 -remote 1 -recordinhibit 0 -cassetteout 0
07:29:57 H.264 Recorder: deckmode: 7 novtr
07:29:57 H.264 Recorder: Device is not encoding.
07:29:57 H.264 Recorder: scanning: 7 1
07:29:57 H.264 Recorder: deckstatus: 7 -connected 0 -remote 1 -recordinhibit 0 -cassetteout 0
07:29:57 H.264 Recorder: deckmode: 7 novtr
07:30:13 H.264 Recorder: Sent command: set -id 7 -encoding -fps 50p -srcx 0 -srcy 0 -srcw 1280
07:30:13 H.264 Recorder: OK: set -id 7
07:30:13 H.264 Recorder: Sent command: start -id 7
07:30:13 H.264 Recorder: OK: start -id 7
07:30:13 H.264 Recorder: Encoding started. Restarting dataconnection to be safe..
07:30:13 H.264 Recorder: Socket did disconnect. Attempting reconnect
07:30:13 H.264 Recorder: DataConnection established!
```

4000 messages from 22/03/2014 14:52:46 to 23/03/2014 07:30:17

Starting a recording will reveal various information:



```
07:31:10 H.264 Recorder: Starting FFMpeg
07:31:10 H.264 Recorder: Data: ffmpeg version 2.0.1 Copyright (c) 2000-2013 the FFmpeg develop
07:31:12 H.264 Recorder: Data: Input #0, mpegts, from 'pipe:':
07:31:12 H.264 Recorder: Data: Output #0, ipod, to '/Users/kasper/Desktop/NO_BACKUP/H264experi
07:31:12 H.264 Recorder: Data: frame=   77 fps=0.0 q=-1.0 size=    1213kB time=00:00:01.99 bit
07:31:13 H.264 Recorder: Data: frame=  101 fps= 97 q=-1.0 size=    1751kB time=00:00:02.47 bit
07:31:13 osascript: Error loading /Library/ScriptingAdditions/Adobe Unit Types.osax/Contents/
07:31:13 osascript: Performance: Please update this scripting addition to supply a value for
07:31:13 H.264 Recorder: Data: frame=  129 fps= 82 q=-1.0 size=    2423kB time=00:00:03.03 bit
07:31:14 H.264 Recorder: Data: frame=  156 fps= 73 q=-1.0 size=    3078kB time=00:00:03.57 bit
07:31:14 H.264 Recorder: Data: frame=  184 fps= 69 q=-1.0 size=    3742kB time=00:00:04.13 bit
07:31:15 H.264 Recorder: Data: frame=  209 fps= 66 q=-1.0 size=    4351kB time=00:00:04.63 bit
07:31:15 H.264 Recorder: Data: frame=  235 fps= 64 q=-1.0 size=    4960kB time=00:00:05.15 bit
07:31:16 H.264 Recorder: Data: frame=  261 fps= 62 q=-1.0 size=    5485kB time=00:00:05.67 bit
07:31:16 H.264 Recorder: Data: frame=  286 fps= 61 q=-1.0 size=    6023kB time=00:00:06.17 bit
07:31:17 H.264 Recorder: Data: frame=  311 fps= 60 q=-1.0 size=    6676kB time=00:00:06.67 bit
07:31:17 H.264 Recorder: Data: frame=  336 fps= 59 q=-1.0 size=    7176kB time=00:00:07.17 bit
07:31:18 H.264 Recorder: Data: frame=  363 fps= 58 q=-1.0 size=    7690kB time=00:00:07.71 bit
07:31:18 H.264 Recorder: Data: frame=  388 fps= 58 q=-1.0 size=    8333kB time=00:00:08.21 bit
```

4000 messages from 22/03/2014 14:56:51 to 23/03/2014 07:31:22

Setting up a “Black Box”

Since the app can be a controller over IP, it’s a very useful to create a “black box” recorder solution with remote control. Various configuration of such a host installation includes:

- Daily automatic reboot
- Automatic reboot after power-failure
- Monitoring of the application and restarting it if it fails
- Pushing files to a webserver (for instance with rsync, ssh keys, crontab)
- Firewall (Icefloor)
- Remote Desktop and SSH
- Automatic Login
- Disable display/computer/harddrive sleep?
- Disable automatic updates
- Disable wireless, enable ethernet
- Set the application as a login item

Development TODO/Ideas:

- USB drive “mount, record, unmount” (option in settings)
  IS_USB_DRIVE_AVAILABLE networking option?
- UDP type interface