Webcast Encoder API v3.2

Overview

The webcasting system consists of encoders which reside onsite with the audio/video systems (typically behind the client's firewall) and a cloudhosted control system.

The webcast encoder API is provided by the control system at https://api.isilive.ca/api and wss://api.isilive.ca/api

API commands and queries are sent by POSTing JSON objects to the HTTPS URL, or by sending JSON objects through the web socket interface.

Responses are also JSON objects.

API command list

Required parameters

- client_id
- password

Optional parameters

- encoder_uid
- command
    - start
    - stop
    - start_recording
        - File_name
        - append
    - stop_recording
    - set_mute
        - channel
        - mute
    - set_volume
        - channel
        - level
    - select_source
        - source_id
    - skew_audio
        - audio_skew
    - send_preview_frame
        - channel

API command list (cont)

Optional parameters

- command
  - set_idle_image
    - idle_image_name
    - image_data
  - clear_idle_image
  - set_overlay_image
    - overlay_image_name
    - image_data
  - clear_overlay_image
  - start_preview
    - channel
  - stop_preview
    - channel

Examples

Request a list of all the webcast encoders registered for a particular client_id. This is the simplest command.

Command

curl -H "Content-Type: application/json" -X POST -d '**{"client_id":"demo", "password":"**********"}**' https://api.isilive.ca/api

Expected response

[{"uid":6136231407564882000,
"client_id":"demo",
"width":"854", "height":"480",
"video_bitrate":"100", "audio_bitrate":"32",
"stream_name":"live1",
"server":"cdn2.isilive.ca",
"application":"liverepeater",
"master_mute":true,
"camera_mute":false,
"projector_mute":false,
"name":"New Encoder",
"model":"dual-decklink",
"version":"3.0",
"selected_source":"1",
"master_vol":"4",
"projector_vol":"4",
"camera_vol":"4",
"webcasting_state":"idle",
"desired_webcasting_state":"false"}]

Additional fields may appear in responses. Any fields not documented here may change, disappear, or contain inaccurate information. Do NOT make use of undocumented fields!

Query the status of a particular webcast encoder.

Command

curl -H "Content-Type: application/json" -X POST -d '**{"client_id":"demo", "password":"**********", "encoder_uid":"6136231407564882000"}**'
https://api.isilive.ca/api

Expected response

{"**status**":"**online**",
"uid":6136231407564882000,
"client_id":"demo",
"width":"854", "height":"480",
"video_bitrate":"100", "audio_bitrate":"32",
"stream_name":"live1",
"server":"cdn2.isilive.ca",
"application":"liverepeater",
"master_mute":true,
"camera_mute":false,
"projector_mute":false,
"name":"New Encoder",
"model":"dual-decklink",
"version":"3.0",
"selected_source":"1",
"master_vol":"4",
"projector_vol":"4",
"camera_vol":"4",
"webcasting_state":"idle",
"desired_webcasting_state":"false"}

Instead of a list, we receive a single JSON object containing all the same information about the encoder from the database. This is what you will receive in response to most successful API commands. Additionally, we get a “status” field which is up to the second information about the encoder’s state.

Possible values are “online” and “offline”

Instruct the webcast encoder to begin streaming live video.

Command

curl -H "Content-Type: application/json" -X POST -d '**{"client_id":"demo", "password":"**********", "encoder_uid":"6136231407564882000", "command":"start"}**' https://api.isilive.ca/api

Expected response

Unless there is an error, the response will be the standard encoder information JSON object.

Instruct the server to start a server-side recording of the live video stream.

Command

curl -H "Content-Type: application/json" -X POST -d '**{"client_id":"demo", "password":"**********", "encoder_uid":"6136231407564882000", "command":"start_recording", "file_name":"test.mp4", "append":false}**' https://api.isilive.ca/api

When an encoder is streaming live, it will always be making a recording to its local disk. That local recording is considered a backup recording. With this command we can also instruct the Wowza Media Server to begin recording the live stream to disk on media servers in our datacentre. This is how we create the primary recording which will be published on the Internet. This video will be availalble the instant the recording is stopped at a URL like this: http://video.isilive.ca/demo/test.mp4

Set the volume on the primary HDMI/SDI input card.

Command

curl -H "Content-Type: application/json" -X POST -d '**{"client_id":"demo", "password":"**********", "encoder_uid":"6136231407564882000", "command":"set_volume", "channel":"camera", "level":"4"}**' https://api.isilive.ca/api

The encoder has 3 audio channels which can raised, lowered, and muted. The two input channels are "**camera**" and "**projector**". The output channel is "**master**".

The level is a range from **0 to 9**. 4 is "nominal", or pass-through audio level.

API command reference

Required parameters for all commands: **client_id, password, encoder_uid**

start

Commence streaming live audio/video to the Internet.

stop

Stop streaming live audio/video to the Internet

start_recording

Start recording, in the cloud, the live stream to an mp4 file.
The specified file_name must end in "**.mp4**" and should not contain any unusual characters that could cause problems in URLs or filesystems. Even spaces are discouraged because they must become %20 creating ugly, unreadable URLs.
You must also specify whether or not to append the new recording to an existing mp4 of the same name. Values for "**append** " are **true|false**. If append is set to false, and a file already exists with the specified file_name, the old file is renamed and the new file is created with the specified name.
Required parameters: **encoder_uid, file_name**, **append**

stop_recording

Stop the recording. This does not affect the live stream, only stops the mp4 file recording.
The recorded mp4 file will be immediately available at http(s)://video.isilive.ca/CLIENT_ID/FILE_NAME.mp4

set_mute

Mute or unmute one of the inputs, or the master output audio of the encoder.
channel options are: **camera | projector | master**
mute options are: **true | false**
Required parameters: **channel, mute**

set_volume

Set the volume level of one of the inputs, or the master audio output of the encoder.
Nominal, unmodified audio is level “4”.
channel options are: **camera | projector | master**
level options are: **0 - 9** (as a string: “4”)
Required parameters: **channel, level**

select_source

The webcast encoder has two audio/video inputs and includes a software video switcher. Use this command to select a source to broadcast. 99% of the time, this should stay on the camera video source “1”.
source_id options are: **1 | 2** (as a string: “1”)
Required parameters: **source_id**

skew_audio

Adjust the audio delay. Minor audio/video sync issues can be remedied by shifting the live audio stream forward or back slightly. Sending a positive value for audio_skew will insert extra audio samples into the stream, thus delaying the audio. Sending a negative value for audio_skew will drop audio samples, thus shifting the audio forward.
Audio_skew values in the range 30 - 200 (milliseconds) are recommended for active live streams. Listeners are unlikely to notice the shift if it is under 200ms. If you need to shift the audio by more than that, simply do so incrementally using 200ms shifts separated by a couple of seconds.
Note that this is a dynamic adjustment. It is not stored or retrievable. Audio skew returns to zero when the encoder is rebooted or reset.

audio_skew options are: **-200 to 200** (as a string: "-150")

Required parameters: **audio_skew**

send_preview_frame

Request a small preview image from one of the encoder's input cards.
This command is only useful when sent via the web socket interface because the image is returned asynchronously via a web socket message.

channel options are: **1 | 2** (as a string: "1")

Required parameters: **channel**

Using web sockets

Alternatively to simply posting commands to the API service using HTTPS, you can create a web socket (over ssl) connection and conduct your interactions with the API server over that persistent connection.

There are two advantages to this technique. One, responses may be slightly quicker because you don't have to go through the whole DNS->TCP->HTTP process for each command. More importantly, the second advantage is that you can receive "push" messages from the system.

From time to time, the server or the encoder may have status information or other messages which you can receive through the web socket and handle using your own callbacks. As of version 3.1 of the API (in addition to responses to your commands) there are only three types of push messages you are likely to receive.

1. Updates/changes to the encoder initiated by another user or interface. If you have two or more browser pages open, each with a web socket connection to the API service, when one of them posts a command to set a volume level for instance, all other connected pages will receive a push message with the new setting.

2. Input capture card preview images. The "send_preview_frame" command will cause the encoder to send a small base64 encoded jpg preview image to all connected web sockets.

3. Live stream progress messages. When the encoder is broadcasting live, it sends status updates to all connected web sockets approximately once per second. These messages are positive proof that the webcast is live.

In future versions, additional status and error messages will be pushed out via sockets.

There are two steps to create a web socket and receive the push messages.

1. Establish the web socket connection
2. Make a request to the API which includes an encoder UID

You must post at least one API command which includes an encoder UID in order to become subscribed to the push messages for that encoder.

An example page showing the basic usage of the API via web sockets is provided at the following URL:

http://video.isilive.ca/examples/api/web_sockets.html

Be there.

iSi LIVE (Integrated Solutions Inc.)
150 – C, Terence Matthews Cres.
Ottawa, Canada K2M 1X4

iSiLIVE.ca