MovieRecorder REST API

SOFTRON

The REST API we have built is used internally as well for the remote control between MovieRecorder as for the web page control. We have settled for JSON over XML for its simplicity and its easy support in the browser.

The documentation is organised per object (Sources, Destinations, …) that can be accessed through the API.

The fields of each object are described with their type and a comment followed by the list of all the methods that the object has. Each method has an HTTP verb (GET, POST, …) and an unique URL.

REST API Quick start guide

Here is an example of request that you can use in a simple browser :

```
http://localhost:8080/sources
```

This will return a dictionary with the list of sources available for a MovieRecorder, which should look like this:

```
{

     {
     "is_recording" : true,
     "is_paused" : false,
     "is_enabled" : true,
     "display_name" : "Io4K - 0 - 1",
     "description" : "Io4K - 0 - 1 (1080i 59.94)",
     "unique_id" : "4XT00292 - 0",
     "device_name" : "Io4K - 0 - 1"
     },
     {
     "is_recording" : false,
     "is_paused" : false,
     "is_enabled" : false,
     "display_name" : "Io4K - 0 - 2",
     "description" : "Io4K - 0 - 2 (1080i 59.94)",
     "unique_id" : "4XT00292 - 1",
     "device_name" : "Io4K - 0 - 2"
     },
     {
     "is_recording" : false,
     "is_paused" : false,
     "is_enabled" : false,
     "display_name" : "Io4K - 0 - 3",
     "description" : "Io4K - 0 - 3 (1080i 59.94)",
     "unique_id" : "4XT00292 - 2",
     "device_name" : "Io4K - 0 - 3"
     },
     {
     "is_recording" : false,
     "is_paused" : false,
     "is_enabled" : false,
     "display_name" : "Io4K - 0 - 4",
     "description" : "Io4K - 0 - 4 (1080i 59.94)",
     "unique_id" : "4XT00292 - 3",
     "device_name" : "Io4K - 0 - 4"
     }

}
```

About the authentication

If you have enabled the password requirement on MovieRecorder, you will have to send the password with each of your requests. so for example if you want to know the list of sources available, you will have to use something like this (1234 is the default password) .

```
http://localhost:/sources?password=1234
```

If you pass the wrong password it will return:

```
Authentication Failed
```

Note that in this document, instead of writing the url that you can type in a browser (http://localhost:8080/sources), we will indicate the REST command and it will look something like this :

```
GET /sources
```

Unique id or index ?

Each source, destination or schedule can be addressed using either its unique_id, or its index :

- The **unique_id** is the safest way to talk to a source as it will not change and remain unique, but you have to request first its value before being able to send arguments.
- The **index** is the (zero based) position of that source, destination or schedule in the list of sources, destinations or schedules presented in the MovieRecorder interface. It is easy to work with as you can just send arguments to the first source, destination or schedule. But its value can change depending on the other elements in the list.

Working with Sources

The API allows you to manipulate the sources (inputs). You can access their information, modify the recording settings (recording name, destinations), start/pause/stop recording. The following keys are the JSON object representation of a source :

Name	Type	Content
device_name	string	The name of the source
unique_id	string	A string that uniquely identifies the source
display_name	string	The name chosen by the user for the source
video_format	string	The name of the source format
frame_rate	float	The frame rate of the source
recording_name	string	The name of the recording
reel_name	string	The reel name of the source. Will be written in the TC track of the QuickTime movie
is_enabled	boolean	True if the source has been enabled, false otherwise
is_locked	boolean	True if the source has been locked, false otherwise. A source must be unlocked to be able to start/stop the recordings
is_recording	boolean	True if the source is recording (or paused), false otherwise
is_paused	boolean	True if the source is paused, false otherwise
can_be_paused	boolean	True if the source can be paused, false otherwise. Classic destinations can not be paused
recording_start_date	date	The start date of the current recording or empty
recording_end_date	date	The date at which the current recording will end. Only specified if a scheduled or timed

		recording is taking place
enabled_destinations	dictionary	returns a an array of dictionaries specifying which destinations are enabled for the source. Each destination will show the following keys: • "destination_name": The name of the destination • "destination_unique_id": A string that uniquely identifies the destination • "destination_recording_path": The path to the currently recording file (or the last recorded file if recording is stopped). This is useful when using tokens and/or counters in file names.

Accessing the information

`GET /sources`

Returns a list of sources as well as their information.

Example response:

```
[

     {

          "recording_name": "RecordingName",
          "recording_start_date": "2017-05-08T12:14:09.837Z",
          "is_paused": false,
          "is_locked": false,
          "can_be_paused": true,
          "enabled_destinations": [

               {
               "destination_name": "ProRes To Movies",
               "destination_unique_id": "D3501279-1B50-4532-84A0-0AB8AD78B02E",
               "destination_recording_path": "/Volumes/Video/MyFirstSource_RecordingName.mov"
               }

          ],
          "unique_id": "AJA_01",
          "display_name": "MyFirstSource",
          "device_name": "AJA_01",
          "reel_name": "001",
          "frame_rate": 29.97,
          "video_format": "1080i 59.94",
          "is_recording": false,
          "recording_end_date": "2017-05-08T13:35:05.539Z",
          "is_enabled": true

     },
     {

          "recording_name": "RecordingName",
          "recording_start_date": "2017-05-08T12:14:09.837Z",
          "is_paused": false,
          "is_locked": false,
          "can_be_paused": true,
          "enabled_destinations": [

               {
               "destination_name": "ProRes To Movies",
               "destination_unique_id": "D3501279-1B50-4532-84A0-0AB8AD78B02E",
               "destination_recording_path": "/Volumes/Video/MySecondSource_RecordingName.mov"
               }

          ],
          "unique_id": "Blackmagic-Design_01",
          "display_name": "MySecondSource",
          "device_name": "Blackmagic-Design_01",
            "reel_name": "001",
            "frame_rate": 29.97,
            "video_format": "1080i 59.94",
            "is_recording": false,
            "recording_end_date": "2017-05-08T13:35:05.539Z",
            "is_enabled": true

        }

]
```

GET /sources/{unique_id or index}

Returns the source and its settings identified by the unique id or the index.

Example response:

```
{

    "recording_name": "RecordingName",
    "recording_start_date": "2017-05-08T12:14:09.837Z",
    "is_paused": false,
    "is_locked": false,
    "can_be_paused": true,
    "enabled_destinations": [

        {
        "destination_name": "ProRes To Movies",
        "destination_unique_id": "D3501279-1B50-4532-84A0-0AB8AD78B02E",
        "destination_recording_path": "/Volumes/Video/MyFirstSource_RecordingName.mov"
        }

    ],
    "unique_id": "AJA_01",
    "display_name": "MyFirstSource",
    "device_name": "AJA_01",
    "reel_name": "001",
    "frame_rate": 29.97,
    "video_format": "1080i 59.94",
    "is_recording": false,
    "recording_end_date": "2017-05-08T13:35:05.539Z",
    "is_enabled": true

}
```

GET /sources/{unique_id or index}/thumbnail

Returns a JPEG thumbnail of the last received image.

GET /sources/{unique_id or index}/info

Returns information about the source feed: last received timecode (string), if it is recording (bool), last warning (string), vuMeterLevels (array of values from 0 to 1.0).

Control the recording

GET /sources/{unique_id or index}/record

Starts the recording of the source identified by the unique id or the index.

Optional parameters (to manage the effective recording duration or end date):

- duration: number of seconds the recording should last
- end_date: date at which the recording should end

`GET /sources/{unique_id or index}/pause`

Pauses the recording of the source identified by the unique id or the index.

`GET /sources/{unique_id or index}/resume`

Resumes (after a pause) the recording of the source identified by the unique id or the index..

`GET /sources/{unique_id or index}/stop`

Stops the recording of the source identified by the unique id or the index.

`GET /sources/{unique_id or index}/destinations`

Returns a list of unique ids of the enabled destinations for the source identified by the unique id or the index.

`PUT /sources/{unique_id or index}/destinations`

Pass an array of destination unique ids that the source identified by the unique id or the index should record to.

`GET /sources/{unique_id or index}/recording_name`

Returns the recording name of the source identified by the unique id or the index.

`PUT /sources/{unique_id or index}/recording_name`

Pass a string identified by the key recording_name to replace the recording name of the source identified by the unique id or the index.

`GET /sources/{unique_id or index}/reel_name`

Returns the reel name of the source identified by the unique id or the index.

`PUT /sources/{unique_id or index}/reel_name`

Pass a string identified by the key reel_name to replace the reel name of the source identified by the unique id or the index.

`PUT /sources/{unique_id or index}/recording_settings`

Pass a dictionary with the keys recording_name and destinations where recording name is a string and destinations is an array of destination unique ids. This method is there for your convenience and groups destinations and recording_name methods into one.

`GET /sources/{unique_id or index}/lock`

Disables the ability to start/pause/resume/stop recording on the source identified by the unique id or the index.

`GET /sources/{unique_id or index}/unlock`

Enables the ability to start/pause/resume/stop recording on the source identified by the unique id or the index.

`GET /sources/{unique_id or index}/destinations`

Returns a list of unique ids of the enabled destinations for the source identified by the unique id or the index.

Gang recording

Instead of controlling each source individually, you can control them together by simply sending an array of sources that you want to control.

`PUT /sources/record`

Pass an array of unique ids or indexes of sources to start recording these.

`PUT /sources/pause`

Pass an array of unique ids or indexes of sources to pause these.

`PUT /sources/resume`

Pass an array of unique ids or indexes of sources to resume recording (after a pause) these.

`PUT /sources/stop`

Pass an array of unique ids or indexes of sources to stop recording these.

Working with Destinations

The API allows you to access the list of destinations and update the path. The following keys are the JSON object representation of a destination :

Name	Type	Content
name	string	The name of the destination.
unique_id	string	A string that uniquely identifies the destination.
preset_display_name	string	The name of the destination’s audio/video preset.
path	string	The path the destination will record to.

`GET /destinations`

Returns a list of destinations.

`PUT /destinations/{unique_id or index}`

Pass a new path to update the destination’s path.

Working with Scheduled Recordings

The API allows you to manipulate the scheduled recordings. The following keys are the JSON object representation of a scheduled recording :

Name	Type	Content
name	string	The name of the scheduled recording.
unique_id	string	A string that uniquely identifies the scheduled recording. This property is read-only and will automatically be generated.
is_enabled	boolean	True if the scheduled recording is enabled, false otherwise.
source_unique_id	string	The unique id of the source that should record.
destination_unique_ids	array	An array of destination unique ids that should be used for the recording.
duration	integer	The duration of the recording in minutes.

start_time	integer	The offset in minutes from midnight.
weekly_repeated	boolean	True if the scheduled recording is to be repeated over time, false otherwise.
recording_days	array	An array of the days of the week (0 being Monday and 6 Sunday) which is present only when weekly_repeated is true.
date	string	The date of the recording which is present only when weekly_repeated is false.

`GET /scheduled_recordings`

Returns a list of scheduled recordings.

`GET /scheduled_recordings/{unique_id or index}`

Returns the scheduled recording identified by the unique id or the index.

`POST /scheduled_recordings`

Creates a new scheduled recording.

Example body:

```
{

     "name" : "My JSON recording",
     "source_unique_id" : "OnTheAir Video",
     "destination_unique_ids": ["928BEB6F-5342-4223-8CB9-5C4629B00288"],
     "duration" : 30,
     "start_time" : 60,
     "weekly_repeated" : true,
     "recording_days" : [3],
     "is_enabled": false

}
```

This creates a scheduled recording that will record every Wednesday from 1 AM to 1:30 AM.

Example response:

```
{

     "success" : true,
     "unique_id" : "BD6C4147-E4AC-4525-B7DE-1392A2382BEE"

}
```

`PUT /scheduled_recordings/{unique_id or index}`

Pass values to update the scheduled recording identified by the unique id or the index.

Working with Metadata

The API allows you to manipulate the metadata sets.

Please note that metadata sets can be used on sources and some keys/values of metadata fields are only relevant when the set is being used on a source. See the documentation of the source object for the usage of sets on sources.

The following keys are the JSON object representation of a metadata set:

Name	Type	Content
name	string	The name of the metadata set.
unique_id	string	A string that uniquely identifies the metadata set. This property is read-only and will automatically be generated.
fields	array	Array of metadata fields.

The following keys are the JSON object representation of a metadata field:

Name	Type	Content
display_name	string	The name of the metadata field.
unique_id	string	A string that uniquely identifies the metadata field. This property is read-only and will automatically be generated.
key	string	The key of the metadata field. Must be unique inside the set.
type	string	The type of the data the metadata field holds.
possible_values	array	An array of strings representing the possible values for the metadata field. Only if the field type is “Multiple Choice”.
current_date_by_default	boolean	True if the metadata field should always be the current date Only if the field type is “Date/Time”.
default_value	(type)	The default value for the metadata field.
use_manual_value	boolean	True if the value should be used instead of the default value Only if the set is being used on a source.
default_value	(type)	The value for the metadata field. Only if the set is being used on a source.

Valid values for the type key are: Boolean, Text, Number, Date/Time and Multiple Choice

`GET /metadata_sets`

Returns a list of metadata sets.

`GET /metadata_sets/{unique_id or index}`

Returns the metadata set identified by the unique id or the index.

`POST /metadata_sets`

Creates a new metadata set.

`PUT /metadata_sets/{unique_id or index}`

Pass values to update the metadata set identified by the unique id or the index.

`DELETE /metadata_sets`

Deletes the metadata set identified by the unique id or the index.

About Websockets

You have 2 options with our APIs to know the status of a recording:

- Using the **REST api**: You would be regularly polling the sources to know their status. This enables you to check the signal format but for dropped frames it would be less accurate. Indeed, as you can only get the last warning, you could miss one in between two polls.
- Using the **WebSockets**: This is much better to work with as the websockets will send you the notifications. There are 2 sockets. One main containing all the changes regarding the sources and destinations, and a second one with all the information for the activity & logs.

The advantage of the WebSockets is that every update is pushed live to it so you would be notified immediately and you won't miss any update. Contact us if you need more information about the WebSockets. You can also have a look at the sample webpage that we have created (it uses websockets). On the Mac that is running MovieRecorder, you can take a look at:

    http://localhost:8080/index.html

For more information

If you need more info or support about MovieRecorder, you can find it on our support desk. And if you need additional information on the API and/or the websockets, submit a ticket and we'll be happy to guide you.