media-nommer-api-python Documentation

Release 1.0dev

DUO Interactive, LLC

January 16, 2012

CONTENTS

This module simplifies accessing media-nommer's JSON API. Request formation, sending, and response parsing is handled for the developer. One should be able to read through the media-nommer documentation and expect to be productive with this API module in short order.

CHAPTER

ONE

PURPOSE

Using this API client along with media-nommer will allow you to do things such as:

- Submit encoding jobs
- Check the status of existing jobs

CHAPTER

TWO

LEARNING MORE

Make sure to read through the media-nommer documentation.

Project Status: Alpha

License: media-nommer-api-python is licensed under the BSD License.

These links may also be useful to you.

- Source repository: https://github.com/duointeractive/media-nommer-api-python
- Issue tracker: https://github.com/duointeractive/media-nommer-api-python/issues

CHAPTER

THREE

DOCUMENTATION

3.1 Installation

Due to the early state of this project, there is a good chance that parts of these instructions are out of date at any given time. If you run into any such issue please let us know on our issue tracker.

3.1.1 Requirements

- Python 2.6 or 2.7. Python 3.x is not supported (yet).

3.1.2 Installing

The easiest way is to install the package is through **pip** or **easy_install**:

```
pip install media-nommer-api
```

Note: This will only work once we start distributing media-nommer-api on PyPi. For now, you'll need to download a tarball/zip, or check the source out from our GitHub project and install via the enclosed `setup.py`. See the `requirements.txt` within the project for dependencies.

3.2 Getting Started

The first thing you'll need to do is import the API module and specify connection details for your `feederd` daemon. You will obviously need to change the hostname, and possibly the port if you overrode the default of 8001. After this call is made, you're ready to start submitting encoding jobs.

```
import media_nommer_api
from media_nommer_api.presets.video_basic import web_medium

api = media_nommer_api.connect('http://localhost:8001')

response = api.job_submit(
    # Source file to encode.
    's3://YOUR_AWS_ID:YOUR_AWS_SECRET@SOME_BUCKET/infilename.mp4',
    # Destination for the encoding.
    's3://YOUR_AWS_ID:YOUR_AWS_SECRET@SOME_BUCKET2/outfilename.mp4',
    # The encoding preset to use.
```

```
    web_medium(),
)
```

3.2.1 Encoding Presets

You'll notice in the example above that we use the `media_nommer_api.presets.video_basic.web_medium()` encoding preset. Presets determine the options that get passed to the encoder, ffmpeg in this case. While this package comes with a number of *Encoding Presets*, you are encouraged to create your own that are specialized to your usage case.

3.2.2 API Call Documentation

You'll now want to review the *API Reference* to see what calls are available.

3.3 Encoding Presets

Encoding presets are simple functions that return dicts that are passed to the selected Nommer. At the current time, this is almost always FFmpegNommer, which uses EC2 instances + ffmpeg to get encodings done.

To use a preset, simply import it and pass it as the `job_options` argument to `media_nommer_api.api.APIConnection.job_submit()` method.

```
import media_nommer_api
from media_nommer_api.presets.video_basic import web_medium

api = media_nommer_api.connect('http://localhost:8001')

response = api.job_submit(
    # Source file to encode.
    's3://YOUR_AWS_ID:YOUR_AWS_SECRET@SOME_BUCKET/infilename.mp4',
    # Destination for the encoding.
    's3://YOUR_AWS_ID:YOUR_AWS_SECRET@SOME_BUCKET2/outfilename.mp4',
    # The encoding preset to use.
    web_medium(),
)
```

3.4 Preset List

This preset list contains all presets included in the media-nommer-api-python package. These are subject to change or be re-named at any time, so you're typically better off using these as the foundation for your own custom presets.

3.4.1 Basic Video Presets

The following presets are basic video encoding presets. Right now, this is just ffmpeg, but may be expanded in the future with the addition of Nommers.

`media_nommer_api.presets.video_basic.`**`web_medium`**`()`

This preset is suitable for medium-quality encodings for web consumption. The two-pass encoding yields a smaller size, and we use qtfaststart.py to move the meta-data to the front for faster buffering.

3.4.2 Android Video Presets

The following presets are geared towards compatibility with Android devices. Some of the older handsets are very picky about certain settings like framerate, audio channels, and etc.

`media_nommer_api.presets.video_android.`**`android_low`**`()`
A lower-quality Android video encoding setting. This should be suitable for the vast majority of devices.

3.5 API Reference

3.5.1 media_nommer_api

Import this client module from within your Python application or script to use media-nommer. Note that this API requires `feederd` to be running and reachable by the host that your Python application is on.

`media_nommer_api.`**`connect`**`(`*api_hostname*`)`
Returns an `APIConnection` object, which is what you make API calls through. This is lazily loaded, so connect away without worry of overhead from instantiation alone. You shouldn't take much of a performance hit until you actually perform some API calls.

Parameters **api_hostname** (*str*) – A URL with protocol, hostname, and port. No trailing slash.

Returns An `APIConnection` object, which has methods on it that map to feederd's RESTful API.

3.5.2 media_nommer_api.api

This is the top-level API module. You should generally instantiate the `APIConnection` object in here by using `media_nommer_api.connect()` instead of direct initialization of `APIConnection`.

The `APIConnection` class is your link to `feederd`, which runs a JSON API.

class `media_nommer_api.api.`**`APIConnection`**`(`*api_hostname*`)`
Your application's means of communicating with feederd's JSON API. The public methods on this class correspond to API calls.

API calls return `APIResponse` objects with the results of your queries. Check the APIResponse.data attrib for your un-serialized results.

Tip: Do not instantiate this class directly. Use the `media_nommer_api.connect()` function for that purpose.

Variables **api_hostname** (*str*) – The protocol, hostname, and port in URI format.

This should generally only be called by media_nommer.client.connect().

Do any setup work to prepare this object for communication with feederd's API. This method constructor should be as lazy as possible.

Parameters **api_hostname** (*str*) – A URL with protocol, hostname, and port. No trailing slash.

`job_submit``(`*source_path*, *dest_path*, *job_options*, *notify_url=None*`)`
Submits an encoding job to feederd. This is an async call, so you may want to specify a notify_url for job state notifications to be sent to.

Parameters

- **source_path** (*str*) – The path string to the master file to encode.
- **dest_path** (*str*) – The path string to where you'd like the encoded files to be saved to.
- **preset** (*str*) – A preset string that corresponds to key in the settings.PRESETS dict.
- **job_options** (*dict*) – A dictionary with additional job options like bitrates, target encoding formats, etc. These options can vary based on the Nommer and the formats you're asking for.
- **notify_url** (*str*) – (Optional) A URL to send job state updates to.

Returns An `APIResponse` object containing `feederd`'s response.

3.5.3 media_nommer_api.server_io

API request and response abstraction and helpers. This module and its contents should not be instantiated directly.

class `media_nommer_api.server_io.`**`APIRequest`**(*api_hostname*, *request_path*, *data*)

An abstraction class that all outbound API requests to `feederd` are wrapped in. Handles some basic serialization and transport stuff.

Variables

- **data** (*dict*) – The dict to be urlencoded and sent to the API server.
- **api_hostname** (*str*) – The protocol, hostname, and port in URI format.
- **request_path** (*str*) – The URL path to the API method.

Parameters

- **api_hostname** (*str*) – The protocol, hostname, and port of your feederd's REST API. There should be no trailing slash.
- **request_path** (*str*) – The URL to query. No leading or trailing slash.
- **data** (*dict*) – A dict object of key/value pairs to urlencode and send to feederd's REST API. If some of your POST keys require JSON values, you'll need to serialize in your API method.

class `media_nommer_api.server_io.`**`APIResponse`**(*request*, *raw_response*)

A basic API response. Performs some simple error handling and provides helpers to check the response for success or failure.

Your application will be interested in this class's `data` instance variable, which contains the server's response.

Variables

- **data** (*dict*) – The un-serialized response from `feederd` in dict form.
- **request** (*APIRequest*) – The `APIRequest` object that this object originated from.
- **raw_response** (*str*) – The raw, serialized string returned from `feederd`.

Parameters

- **request** (`APIRequest`) – The request object that instantiated this response object.
- **raw_response** (*str*) – The raw server response's body.

`is_success`()

Indicates whether the call completed without errors.

Returns True if there were no errors, False otherwise.

Return type bool

3.6 Indices and tables

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX

m