Stupeflix Tasks API Documentation

Release 0.1

Luper Rouch

February 09, 2017

Contents

The REST API is the interface to access the Stupeflix tasks system.

The API is versioned. You should always use the newest version of the API, but older versions will never be removed.

The base URL of the tasks system is `https://dragon.stupeflix.com/`. So for example the full URL for the method `/foo/bar` in the API Reference API is `https://dragon.stupeflix.com/v2/foo/bar`.

Contents:

CHAPTER 1

API Reference

1.1 General notes

All requests are done over SSL.

All strings must be UTF-8 encoded.

POST requests parameters are passed in JSON-encoded bodies.

All dates are in ISO8601 format.

1.2 Authentication

Requests that create tasks such as *POST /v2/create* require authentication. There are two methods to authenticate requests:

1.2.1 Secret key

This kind of authentication should only be used for server to server requests, as it exposes your secret key.

For POST and DELETE requests, the secret key can be passed as a top level “secret” key in the JSON body:

```
{
    "secret": "123456",
    "tasks": {
        "task_name": "image.info",
        "url": "http://files.com/image.jpg"
    }
}
```

The secret key can also be passed via the `Authorization` header. The key should be prefixed by the string `Secret`, with a whitespace separating the two strings:

```
Authorization: Secret 123456
```

For GET requests, the secret key must be passed in the querystring:

```
https://dragon.stupeflix.com/v2/storage/files?secret=123456
```

1.2.2 Api key + referrer

As for *Secret key* authentication, the api key can be passed in the request JSON body or the `Authorization` header:

- in the JSON body:

```
{
    "api_key": "654321",
    "tasks": {
        "task_name": "image.info",
        "url": "http://files.com/image.jpg"
    }
}
```

- in the `Authorization` header:

```
Authorization: Api-Key 654321
```

The `Referrer` header of the request must also be in your account's whitelist.

This kind of authentication is what you should use in your javascript code, but be careful as requests can easilly be forged to fake the `Referrer` header.

1.3 HTTP Status codes

200 Operation was successful.

400 Invalid request. The response body contains a description of the errors, for example if you forgot the `tasks` parameter in a *`POST /v2/create`* request:

```
{
    "status": "error",
    "errors": [
        {
            "name": "tasks",
            "location": "body",
            "description": "tasks is missing"
        }
    ]
}
```

401 Invalid credentials, or account limits reached.

503 The system is temporarily down and the client should retry later.

1.4 Tasks definitions

Task are defined by objects with at least a "task_name" key. Storage systems parameters can be passed in the "task_store" key. Other keys contain the task parameters. Here is an example of a `image.info` task definition, with the result stored on the *volatile* storage system:

```
{
    "task_name": "image.info",
    "task_store": {
        "type": "volatile"
    },
    "url": "http://files.com/image.jpg"
}
```

1.5 Tasks statuses format

Tasks statuses are objects of the form:

```
{
    "status": "executing",
    "key": "5OYA5JQVFIAHYOMLQG5QV3U33M",
    "progress": 90,
    "events": {
        "started": "2013-04-03T15:47:27.707526+00:00",
        "queued": "2013-04-03T15:47:27.703674+00:00"
    }
}
```

Statuses contain at minimum the following keys:

- “status”: the current step of the task in the execution pipeline, one of “queued”, “executing, “success” or “error”
- “key”: the server-side key used to identify the task
- “progress”: a value representing task progress; its type depends on the task and could be anything that is JSON-encodable
- **“events”: an object containing chronological events of the task:**
    - “queued”: date at which the task was queued
    - “started”: date at which the task has been attributed to a worker
    - “completed”: completion date of the task

When a task completes successfully (with `"status": "success"`), its status contains an additional “result” key, for example:

```
{
    "status": "success",
    "progress": null,
    "events": {
        "completed": "2013-10-31T14:54:52.689272+00:00",
        "queued": "2013-10-31T14:54:51.987459+00:00"
    },
    "key": "UM6EFJKQWMVON5N3CBKPV52NHE",
    "result": {
        "exposure_time": 0.00156,
        "date_time": "2009:11:02 01:21:55",
        "content_type": "image/jpeg",
        "flash": false,
        "height": 1320,
        "width": 1918,
        "iso_speed": 640,
        "focal_length": 2800,
        "alpha": false,
        "rotation": null,
        "type": "image"
    }
}
```

Tasks that ended on an error (with `"status": "error"`) will return a status with an “error” key. “error” can have two forms:

- a string containing the error message
- for input parameters validation errors, an object describing the validation problems, for example:

```
{
    "status": "error",
    "progress": null,
    "events": {
        "completed": "2013-09-20T12:56:49.385937+00:00",
        "queued": "2013-09-20T12:56:49.369911+00:00"
    },
    "key": "QJZTXA3LNZKQ6X4RPGQ5EHRSMI",
    "error": {
        "parameters": {
            "url": [
                "this field is required"
            ]
        }
    }
}
```

Some tasks also support partial results, that are sent before the end of the task. Partial results are like full results, but their status is “executing” and the “result” mapping only contains a subset of the final result.

Here is an example partial result for the `video.create` task. Note that “result” only contains the “duration” and “preview” keys, while the final result would also contain the URLs of the final video and thumbnail image in the “export” and “thumbnail” keys:

```
{
    "status": "executing",
    "result": {
        "duration": 10,
        "preview": "http://bill.stupeflix.com/storage/flvstreamer/222/LY5XZIPILG6WKKIAGQAB4RLHBY/360
    },
    "key": "LY5XZIPILG6WKKIAGQAB4RLHBY",
    "progress": 100,
    "events": {
        "started": "2013-11-16T06:02:55.669278+00:00",
        "queued": "2013-11-16T06:02:55.667394+00:00"
    }
}
```

1.6 Tasks results

By default, output files are stored forever on Amazon S3 and served through Amazon Cloudflare. Other storage backends are available, see Storage systems for a complete reference.

1.7 Tasks API methods

`POST /v2/create`

Queue one or more tasks and return a list of tasks status.

Example request:

```
{
    "tasks": [
        {"task_name": "image.info", "url": "http://files.com/image.jpg"},
        {"task_name": "image.thumb", "url": "http://files.com/image.jpg"}
    ]
}
```

Example response:

```
[
    {
        "status": "queued",
        "events": {"queued": "2013-04-03T15:47:27.703674+00:00"},
        "key": "6GRQ3H5EHU7GXUTIOSS2GUDPGQ"
    },
    {
        "status": "queued",
        "events": {"queued": "2013-04-03T15:47:27.703717+00:00"},
        "key": "5OYA5JQVFIAHYOMLQG5QV3U33M"
    }
]
```

Request JSON Object

- **`tasks`** – a list containing the definitions of the tasks to execute. The method also accepts a single task definition for convenience.
- **`block`** – a boolean indicating if the call should return immediately with the current status of the tasks, or wait for all tasks to complete and return their final status.

`POST /v2/create_stream`

Queue one or more tasks, and stream their status updates.

Example request:

```
{
    "tasks": [
        {"task_name": "hello", "name": "John"},
        {"task_name": "hello", "name": "Jane"},
    ]
}
```

Example response:

```
[{"status": "queued", "events": {"queued": "2013-04-03T15:47:27.703674+00:00"}, "key": "6GRQ3H5E
{"status": "executing", "events": {"started": "2013-04-03T15:47:27.707526+00:00", "queued": "201
{"status": "executing", "events": {"started": "2013-04-03T15:47:27.710286+00:00", "queued": "201
{"status": "success", "result": "Hello John", "events": {"completed": "2013-04-03T15:47:27.72622
{"status": "success", "result": "Hello Jane", "events": {"completed": "2013-04-03T15:47:27.72902
```

The first line of the response contains a list with the immediate statuses of the tasks. The list is in the same order as the `tasks` parameter, to allow the client to know which key correspond to which task.

The next lines contains interleaved statuses of the two tasks. The response is closed when all the tasks have finished.

Request JSON Object

- **`tasks`** – a list containing the definitions of the tasks to execute.

GET /v2/status

Query the status of one or more tasks.

Example request:

```
https://dragon.stupeflix.com/v2/status?tasks=6GRQ3H5EHU7GXUTIOSS2GUDPGQ&tasks=5OYA5JQVFIAHYOMLQG
```

The response contains a list of task statuses, see *POST /v2/create* for a response example.

Query Parameters

- **tasks** – one or more tasks keys.
- **block** – a boolean indicating if the call should return immediately with the current status of the task, or wait for all tasks to complete and return their final status.
- **details** – if this boolean is true, return more details in the statuses objects (tasks parameters, storage details, etc...).

POST /v2/status

Same as *GET /v2/status* but using POST semantics. Useful when there are too much tasks to query and the querystring size limit is reached.

GET /v2/status_stream

Get status streams of one or more tasks.

Example request:

```
https://dragon.stupeflix.com/v2/stream?tasks=6GRQ3H5EHU7GXUTIOSS2GUDPGQ&tasks=5OYA5JQVFIAHYOMLQG
```

See *POST /v2/create_stream* for a description of the response.

Query Parameters

- **tasks** – one or more tasks keys.

POST /v2/status_stream

Same as *GET /v2/status_stream* but using POST semantics. Useful when there are too much tasks to query and the querystring size limit is reached.

1.8 Storage API methods

GET /v2/storage/files/(*path*)

List tasks output files.

The response is a JSON mapping containing the lists of files and directories, and storage space used by these files:

```
{
    "files": [
        {
            "name": "dragon-image.thumb-IeWutW",
            "size": 4293,
            "last_modified": "2013-10-28T20:22:21.000Z"
        }
    ],
    "directories": [
        "XDJC6DIS5UDSFBBOLXWMN27ORI/"
    ],
    "usage": 4293
}
```

Parameters

- **`path`** – the path of the directory to list.

Query Parameters

- **`recursive`** – a boolean value indicating if *path* sub-directories must be traversed too.

`DELETE /v2/storage/files/`(*path*)

Delete tasks output files.

If *path* is empty, recursively delete all output files.

If *path* points to a directory, recursively delete all output files under this directory.

If *path* points to a file, delete this file.

Files can also be targeted by date with the *from*, *to* and *max_age* parameters. *from* and *to* dates must be ISO8601 date time strings; if they don't include a timezone they will be interpreted as UTC.

The *urls* parameter also allows to delete files from absolute URLs.

The response is a JSON mapping containing the list of deleted files, the number of bytes freed and the (approximate) total space used on the persistent storage after the operation:

```
{
    "deleted": [
        "BCSIT5KDDQQTC7GZ6TBJE7NFIU/dragon-image.thumb-9b0E9P",
        "XDJC6DIS5UDSFBBOLXWMN27ORI/dragon-image.thumb-IeWutW"
    ],
    "freed": 1257,
    "usage": 6578
}
```

Parameters

- **`path`** – the path of the file or directory to delete.

Request JSON Object

- **`urls`** – a list of absolute URLs to delete. If this parameter is used, all other selection parameters (*path, from, to, max_age*) are ignored.
- **`dry_run`** – if this boolean is true, return the files that would be deleted, but don't actually delete them (default: `false`).
- **`from`** – the date from which point to delete files.
- **`to`** – the date up to which point to delete files.
- **`max_age`** – files older than *max_age* days are deleted.

`POST /v2/storage/expiration`

Set lifetime of tasks output files.

Request JSON Object

- **`days`** (*`int`*) – the number of days after which files are deleted in the tasks output storage. A value of 0 means that files are never deleted.

`GET /v2/storage/expiration`

Get the current lifetime of tasks output files.

The response is the current lifetime of files, in days. A value of 0 means that files are never deleted.

CHAPTER 2

Storage systems

Storage systems, introduced in the /v2 API, allow you to choose where your task output files are stored.

Here is an example request storing two thumbnails in the *persistent* and *volatile* storages:

```
{
    "tasks": [
        {
            "task_name": "image.thumb",
            "task_store": {
                "type": "volatile"
            },
            "url": "http://files.com/image.jpg"
        },
        {
            "task_name": "image.thumb",
            "task_store": {
                "type": "persistent"
            },
            "url": "http://files.com/image.jpg"
        }
    ]
}
```

2.1 persistent

This is the default storage system, it stores files permanently on Cloudfront.

You can change the lifetime of your files with *`POST /v2/storage/expiration`*.

You can manage the files in your persistent storage with the *Storage API methods*.

Additional costs are applied to the persistent storage, please read our pricing.

2.2 volatile

Files are stored for a week on S3, and permanently deleted.

2.3 youtube

Upload videos on Youtube. It requires the following parameters:

- **access_token** (string) - Target user's access token with upload authorization.
- **developer_key** (string) -Youtube developer key of a registered app.
- **title** (string) - Video title.

The following optional parameters are also accepted:

- **description** (string) - Video description.
- **tags** (list of strings) - List of video tags.
- **category_id** (integer) - Video category ID number.
- **privacy_status** (string) - Privacy status of the video. Accepted values are "public", "private" and "unlisted" *(default: "public")*.

2.4 s3_signed

Upload to S3 signed URLs. It requires the following parameters:

- **url** (string) - the S3 signed URL to upload to.

Callbacks and Errbacks

All tasks accept a `url_callback` and `url_errback` argument, that allows to specify a HTTP endpoint that will be called when the task is complete.

This alleviates the need to poll *`POST /v2/status`* to check if a task was completed and simplifies asynchronous code if you need to queue many tasks at once.

The callback endpoint will receive a POST containing the full task status as soon as the task ends (the same JSON that you would get on *`POST /v2/status`*). Note that the request is not form-encoded, the POST request body contains directly the status in JSON.

Warning: If you poll *`POST /v2/status`* just after receiving the callback, it will most likely return an "executing" status. This is expected, because it takes some time for the status to propagate to the database. This is also unneeded, since the final status is contained in the callback request body.

CHAPTER 4

Tasks Reference (v2)

4.1 audio.beats

Find beats in an audio file.

Parameters

- **url_callback** (*string*) – URL to callback when the task completes successfully. See Callbacks and Errbacks for details.
- **url_errback** (*string*) – URL to callback when the task fails. See Callbacks and Errbacks for details.
- **url** (*string*) – URL of the audio file

Output Values

- **beats** (*object*) – An array containing the timestamps of the detected beats, in seconds
- **duration** (*integer*) – The processed audio file duration in seconds
- **downbeats** (*object*) – An array containing the timestamps of the detected downbeats, in seconds

4.2 audio.convert

Transcode audio file (mp3, vorbis), and return audio duration.

Parameters

- **url_callback** (*string*) – URL to callback when the task completes successfully. See Callbacks and Errbacks for details.
- **url_errback** (*string*) – URL to callback when the task fails. See Callbacks and Errbacks for details.
- **url** (*string*) – URL of the audio file to be converted.
- **codec** (*string*) – Desired codec for the output file. *(choices:* `'mp3'`, `'vorbis'`, `'aac'` *) (default:* `u'mp3'` *)*
- **force** (*boolean*) – Force encoding. *(default:* `False` *)*

Output Values

- **`duration`** (*`float`*) – Duration of the audio file in seconds.
- **`content_type`** (*`string`*) – Output file content type.

Output Files

- **`output`** – URL of the output file.

4.3 audio.info

Return duration and codec of an audio file.

Parameters

- **`url_callback`** (*`string`*) – URL to callback when the task completes successfully. See Callbacks and Errbacks for details.
- **`url_errback`** (*`string`*) – URL to callback when the task fails. See Callbacks and Errbacks for details.
- **`url`** (*`string`*) – URL of the audio file to be scanned.

Output Values

- **`duration`** (*`float`*) – Duration of the audio file in seconds, rounded to 1/100th second.
- **`content_type`** (*`string`*) – Content-type of the audio file.
- **`codec`** (*`string`*) – Codec of the audio file.

4.4 audio.waveform

Create a waveform image from an audio file.

Parameters

- **`url_callback`** (*`string`*) – URL to callback when the task completes successfully. See Callbacks and Errbacks for details.
- **`url_errback`** (*`string`*) – URL to callback when the task fails. See Callbacks and Errbacks for details.
- **`url`** (*`string`*) – URL of the audio file to be scanned.
- **`width`** (*`integer`*) – *(default:* `1024` *)*
- **`height`** (*`integer`*) – *(default:* `60` *)*
- **`vmargin`** (*`integer`*) – Vertical margin. *(default:* `0` *)*
- **`fill`** (*`string`*) – Color of the wave-form. *(default:* `u'#000000'` *)*
- **`background`** (*`string`*) – Color of the background. *(default:* `u'#FFFFFF'` *)*
- **`start`** (*`float`*) – Seconds to start from. *(default:* `0.0` *)*
- **`end`** (*`float`*) – Generate waveform up to this point, in seconds.

- **format** (*string*) – Output image format. *(choices:* `'png'`, `'jpeg'` *) (default:* `u'jpeg'` *)*

Output Values

- **duration** (*float*) – Duration of the audio file in seconds.
- **width** (*integer*) –
- **height** (*integer*) –
- **content_type** (*string*) –

Output Files

- **output** – URL of the output file.

4.5 html.scrape

Scrape html webpage to return videos & images found

Parameters

- **url_callback** (*string*) – URL to callback when the task completes successfully. See Callbacks and Errbacks for details.
- **url_errback** (*string*) – URL to callback when the task fails. See Callbacks and Errbacks for details.
- **url** (*string*) – URL of the html page

Output Values

- **hits** (*object*) –
- **page_title** (*string*) –

4.6 image.gif

Create an animated GIF from a list of images.

Parameters

- **url_callback** (*string*) – URL to callback when the task completes successfully. See Callbacks and Errbacks for details.
- **url_errback** (*string*) – URL to callback when the task fails. See Callbacks and Errbacks for details.
- **images** (*list of strings*) – The list of image URLs that will be used to create the animated GIF.
- **loop** (*integer*) – The number of loops of the GIF, 0 means to loop forever, and -1 no loop. *(default:* `0` *)*
- **frame_duration** (*float*) – The duration in seconds during which each image will be shown when the GIF is playing, rounded to 1/100th of a second. *(default:* `0.1` *)*

- **`width`** (*`integer`*) – The pixel width of the output GIF. Leave empty to use source images width.
- **`height`** (*`integer`*) – The pixel height of the output GIF. Leave empty to use source images height.

Output Files

- **`output`** – The URL of the output GIF.

4.7 image.info

Return image file information.

Parameters

- **`url_callback`** (*`string`*) – URL to callback when the task completes successfully. See Callbacks and Errbacks for details.
- **`url_errback`** (*`string`*) – URL to callback when the task fails. See Callbacks and Errbacks for details.
- **`url`** (*`string`*) – URL of the image file to be scanned.

Output Values

- **`content_type`** (*`string`*) – Content-Type of the image file.
- **`type`** (*`string`*) – Type of the file.
- **`width`** (*`integer`*) –
- **`height`** (*`integer`*) –
- **`alpha`** (*`boolean`*) –
- **`exif_orientation`** (*`integer`*) – The exif orientation that should be applied to the image to see it as it was shot, as an integer x, where:
  - x=1: The 0th row is at the visual top of the image, and the 0th column is the visual left-hand side.
  - x=2: The 0th row is at the visual top of the image, and the 0th column is the visual right-hand side
  - x=3: The 0th row is at the visual bottom of the image, and the 0th column is the visual right-hand side.
  - x=4: The 0th row is at the visual bottom of the image, and the 0th column is the visual left-hand side.
  - x=5: The 0th row is the visual left-hand side of the image, and the 0th column is the visual top.
  - x=6: The 0th row is the visual right-hand side of the image, and the 0th column is the visual top.
  - x=7: The 0th row is the visual right-hand side of the image, and the 0th column is the visual bottom.
  - x=8: The 0th row is the visual left-hand side of the image, and the 0th column is the visual bottom.

- **`rotation`** (*`float`*) – The rotation that should be applied to the image to see it as it was shot, in degrees. (None if a flip is required or info is not present in exif)
- **`date_time`** (*`string`*) –
- **`flash`** (*`boolean`*) –
- **`focal_length`** (*`float`*) –
- **`iso_speed`** (*`float`*) –
- **`exposure_time`** (*`float`*) –

4.8 image.strip

Create an image strip of custom dimensions by concatenating images.

Parameters

- **`url_callback`** (*`string`*) – URL to callback when the task completes successfully. See Callbacks and Errbacks for details.
- **`url_errback`** (*`string`*) – URL to callback when the task fails. See Callbacks and Errbacks for details.
- **`urls`** (*`MultiHttpFileField`*) – Array of the source images URLs.
- **`width`** (*`integer`*) – Pixel width of each frame stitched into film strip.
- **`height`** (*`integer`*) – Pixel height of each frame stitched into film strip.
- **`crop`** (*`boolean`*) – If false, video frames fit each strip section. If true, video frames fill each strip section, aligning centers. *(default:* `False` *)*
- **`wrap`** (*`integer`*) – Number of images that can be stitched horizontally before stitching starts onto a new line. Use it to create a two dimensional film strip, with count = int * wrap. If left unspecified, all frames are stitched on a single line.
- **`format`** (*`string`*) – Output image file format *(choices:* `'jpeg'`, `'png'` *) (default:* `u'jpeg'` *)*

Output Values

- **`count`** (*`integer`*) – Actual number of frames in the output.
- **`width`** (*`integer`*) – Width of the output image in pixels.
- **`height`** (*`integer`*) – Height of the output image in pixels.
- **`content_type`** (*`string`*) – Mime-type of the output image.

Output Files

- **`output`** – URL of the output image.

4.9 image.thumb

Create a new image of custom dimensions and orientation from an original image.

Parameters

- **url_callback** (*string*) – URL to callback when the task completes successfully. See Callbacks and Errbacks for details.
- **url_errback** (*string*) – URL to callback when the task fails. See Callbacks and Errbacks for details.
- **width** (*integer*) – Desired thumbnail width, in pixels.
- **height** (*integer*) – Desired thumbnail height, in pixels
- **crop** (*boolean*) – If crop is true, original image fills new image dimensions. If crop is false, original image fits new image dimensions. *(default:* `False` *)*
- **url** (*string*) – URL of the source image
- **rotation** (*integer*) – A counter clockwise rotation rotation to apply to the thumbnail, in degrees. *(choices:* `0`, `90`, `180`, `270` *) (default:* `0` *)*
- **poster** (*boolean*) – If true, a play icon is added in the center. *(default:* `False` *)*
- **format** (*string*) – The output format. *(choices:* `'jpeg'`, `'gif'`, `'png'` *) (default:* `u'jpeg'` *)*

Output Values

- **width** (*integer*) – thumbnail width
- **height** (*integer*) – thumbnail height
- **original_width** (*integer*) – original image width
- **original_height** (*integer*) – original height

Output Files

- **output** – URL of the thumbnail.

4.10 video.convert

Create transcoded video file with custom dimensions, and return its video.info output values.

Parameters

- **url_callback** (*string*) – URL to callback when the task completes successfully. See Callbacks and Errbacks for details.
- **url_errback** (*string*) – URL to callback when the task fails. See Callbacks and Errbacks for details.
- **url** (*string*) – URL of the source video
- **width** (*integer*) –
- **height** (*integer*) –
- **crop** (*boolean*) – Allows croping the video to fit in the output size *(default:* `False` *)*
- **audio_codec** (*string*) – Desired audio audio. *(choices:* `'mp2'`, `'mp3'`, `'aac'`, `'wmav1'`, `'wmav2'` *) (default:* `u'aac'` *)*
- **video_codec** (*string*) – Desired video codec. *(choices:* `'h264'` *) (default:* `u'h264'` *)*
- **video_bitrate** (*integer*) – Desired video bitrate, in kbps. *(default:* `3000` *)*

- **audio_bitrate** (*integer*) – Desired audio bitrate, in kbps. *(default:* `128` *)*
- **sample_rate** (*integer*) – Desired audio sample rate, in kHz. *(choices:* `22050`, `44100`, `48000` *) (default:* `44100` *)*
- **crf** (*integer*) – Output constant rate factor (video) *(default:* `23` *)*
- **gop** (*integer*) – Output group of picture (GOP) size *(default:* `250` *)*

Output Values

- **content_type** (*string*) – Output file content type.
- **width** (*integer*) –
- **height** (*integer*) –
- **original_width** (*integer*) –
- **original_height** (*integer*) –
- **duration** (*float*) – Duration of the video file, in seconds.
- **frame_rate** (*float*) –
- **audio_codec** (*string*) –
- **video_codec** (*string*) –
- **alpha** (*boolean*) –
- **rotation** (*float*) – The counter clockwise rotation that should be applied to the video to see it as it was shot, in degrees.

Output Files

- **output** – URL of the converted file.

4.11 video.create

Create video file(s) from a SXML definition and video profile(s).

Parameters

- **url_callback** (*string*) – URL to callback when the task completes successfully. See Callbacks and Errbacks for details.
- **url_errback** (*string*) – URL to callback when the task fails. See Callbacks and Errbacks for details.
- **definition** (*string*) – SXML video definition
- **profile** (*string*) – *(default:* `u'360p'` *)*
- **preview** (*boolean*) – *(default:* `False` *)*
- **export** (*boolean*) – *(default:* `True` *)*
- **thumbnail_time** (*float*) – *(default:* `1.0` *)*
- **antialias** (*integer*) – *(choices:* `1`, `2`, `4` *) (default:* `4` *)*

Output Values

- **duration** (*float*) –

- **width** (*integer*) – video width
- **height** (*integer*) – video height

Output Files

- **preview** –
- **export** –
- **thumbnail** –

4.12 video.info

Return video file information.

Parameters

- **url_callback** (*string*) – URL to callback when the task completes successfully. See Callbacks and Errbacks for details.
- **url_errback** (*string*) – URL to callback when the task fails. See Callbacks and Errbacks for details.
- **url** (*string*) – URL of the video file to be scanned.

Output Values

- **content_type** (*string*) – Mime-type of the video file.
- **width** (*integer*) – Video width, in pixels.
- **height** (*integer*) – Video height, in pixels.
- **duration** (*float*) – Video duration, in seconds.
- **frame_rate** (*float*) – Video frame rate, in frames per second.
- **alpha** (*boolean*) – A boolean indicating if the video has an alpha channel.
- **rotation** (*float*) – The rotation that should be applied to the video to see it as it was shot, in degrees.
- **audio_codec** (*string*) – Audio codec name.
- **video_codec** (*string*) – Video codec name.

4.13 video.reverse

Create a reversed video file with custom dimensions, and return its video.info output values.

Parameters

- **url_callback** (*string*) – URL to callback when the task completes successfully. See Callbacks and Errbacks for details.
- **url_errback** (*string*) – URL to callback when the task fails. See Callbacks and Errbacks for details.
- **url** (*string*) – URL of the source video

- **width** (*integer*) –
- **height** (*integer*) –
- **crop** (*boolean*) – Allows croping the video to fit in the output size *(default:* `False` *)*
- **video_codec** (*string*) – Desired video codec. *(choices:* `'h264'` *) (default:* `u'h264'` *)*
- **video_bitrate** (*integer*) – Desired video bitrate, in kbps. Use source bitrate if left empty.
- **crf** (*integer*) – Output constant rate factor (video) *(default:* `23` *)*
- **gop** (*integer*) – Output group of picture (GOP) size *(default:* `250` *)*

Output Values

- **duration** (*float*) – Duration of the video file, in seconds.

Output Files

- **output** – URL of the converted file.

4.14 video.strip

Create a film strip image of custom dimensions showing stitched frames of a video, return video.info output values for original video.

Parameters

- **url_callback** (*string*) – URL to callback when the task completes successfully. See Callbacks and Errbacks for details.
- **url_errback** (*string*) – URL to callback when the task fails. See Callbacks and Errbacks for details.
- **url** (*string*) – URL of the source video.
- **width** (*integer*) – Pixel width of each frame stitched into film strip.
- **height** (*integer*) – Pixel height of each frame stitched into film strip.
- **crop** (*boolean*) – If false, video frames fit each strip section. If true, video frames fill each strip section, aligning centers. *(default:* `False` *)*
- **wrap** (*integer*) – Number of video frames that can be stitched horizontally before stitching starts onto a new line. Use it to create a two dimensional film strip, with count = int * wrap. If left unspecified, all frames are stitched on a single line.
- **start** (*float*) – Time of first frame extracted from video - by default first frame of video. *(default:* `0.0` *)*
- **end** (*float*) – Time of last frame extracted from video - by default last frame of video.
- **count** (*integer*) – Number of frames extracted from video, at equal time intervals between start and end times. *(default:* `10` *)*
- **format** (*string*) – Output image file format *(choices:* `'jpeg'`, `'png'` *) (default:* `u'jpeg'` *)*

Output Values

- **count** (*integer*) – Actual number of frames in the output.
- **width** (*integer*) – Width of the output image in pixels.
- **height** (*integer*) – Height of the output image in pixels.
- **original_width** (*integer*) – Width of the input video file, in pixels.
- **original_height** (*integer*) – Width of the input video file, in pixels.
- **duration** (*float*) – Duration of the input video file, in seconds.
- **frame_rate** (*float*) – Frame rate of the input video file, in frames per second.
- **content_type** (*string*) – Mime-type of the output image.

Output Files

- **output** – URL of the output image.

4.15 video.thumb

Create a reversed video file with custom dimensions, and return its video.info output values.

Parameters

- **url_callback** (*string*) – URL to callback when the task completes successfully. See Callbacks and Errbacks for details.
- **url_errback** (*string*) – URL to callback when the task fails. See Callbacks and Errbacks for details.
- **url** (*string*) – URL of the source video.
- **width** (*integer*) – Width of output image file, in pixels. The default is to use the original video width.
- **height** (*integer*) – Height of output image file, in pixels. The default is to use the original video height.
- **crop** (*boolean*) – If false, video frame fits output image. If true, video frame fills output image. *(default:* `False` *)*
- **time** (*float*) – Timestamp of the video frame to extract, in seconds. *(default:* `0.0` *)*
- **format** (*string*) – Output image file format. *(choices:* `'jpeg'`, `'png'` *) (default:* `u'jpeg'` *)*
- **poster** (*boolean*) – If true, a play icon is added in the center. *(default:* `False` *)*
- **quality** (*integer*) – Output quality, from 1 to 95. *(default:* `75` *)*

Output Values

- **width** (*integer*) – Width of the output image in pixels.
- **height** (*integer*) – Height of the output image in pixels.
- **original_width** (*integer*) – Width of the input video file.
- **original_height** (*integer*) – Width of the input video file.
- **duration** (*float*) – Duration of the input video file, in seconds.
- **content_type** (*string*) – Mime-type of the output image.

Output Files

- **`output`** – URL of the output image.

4.16 video.upload.fb

Upload a video to Facebook.

Parameters

- **`url_callback`** (*string*) – URL to callback when the task completes successfully. See Callbacks and Errbacks for details.
- **`url_errback`** (*string*) – URL to callback when the task fails. See Callbacks and Errbacks for details.
- **`url`** (*string*) – URL of the source video.
- **`access_token`** (*string*) – Target user's access token.
- **`title`** (*string*) – Video title.
- **`description`** (*string*) – Video description.
- **`privacy`** (*string*) – Privacy level of the video. *(choices:* `'AUTO'`, `'EVERYONE'`, `'ALL_FRIENDS'`, `'FRIENDS_OF_FRIENDS'`, `'SELF'` *) (default:* `u'AUTO'` *)*
- **`no_story`** (*boolean*) – If set to true, this will suppress feed and timeline story. *(default:* `False` *)*
- **`api_key`** (*string*) – Facebook API key. *(default:* `u'-'` *)*
- **`app_secret`** (*string*) – Facebook app secret. *(default:* `u'-'` *)*

Output Values

- **`duration`** (*float*) – Duration of the input video file, in seconds.

Output Files

- **`output`** – URL of the uploaded video on Facebook.

4.17 video.upload.vimeo

Upload a video from user url on Vimeo. Register your app to get a consumer key and secret. Then retrieve an access token key and a secret following these instructions on Oauth for the Vimeo API.

You can use either OAuth1 or OAuth2.

OAuth2 parameter (Vimeo API v3):

- oauth2_token

OAuth1 parameters (Vimeo API v2):

- consumer_key
- consumer_secret
- access_token_key

- access_token_secret

OAuth2 Token requires these privileges:

- Edit (to edit titles / descriptions)
- Upload

Parameters

- **`url_callback`** (*string*) – URL to callback when the task completes successfully. See Callbacks and Errbacks for details.
- **`url_errback`** (*string*) – URL to callback when the task fails. See Callbacks and Errbacks for details.
- **`url`** (*string*) – Video url to upload
- **`title`** (*string*) – Video title
- **`description`** (*string*) – Video description
- **`consumer_key`** (*string*) – OAuth1 Application consumer key
- **`consumer_secret`** (*string*) – OAuth1 Application consumer secret
- **`access_token_key`** (*string*) – OAuth1 User access token key
- **`access_token_secret`** (*string*) – OAuth1 User access token secret
- **`oauth2_token`** (*string*) – OAuth2 User access token secret

Output Values

- **`free_space`** (*integer*) –
- **`uploaded_file_size`** (*integer*) –
- **`output`** (*string*) – URL of the uploaded video on Vimeo.
- **`duration`** (*float*) – Duration of the input video file, in seconds.

4.18 video.upload.youtube

Upload a video to Youtube using the version 3 of the API with OAuth2 Bearer authentication. Register your app and retrieve an access token following these instructions.

Otherwise, you can also get a token with us from there

Parameters

- **`url`** (*string*) – URL of the source video.
- **`access_token`** (*string*) – Target user's access token with upload authorization.
- **`developer_key`** (*string*) – Youtube developer key of a registered app.
- **`title`** (*string*) – Video title.
- **`description`** (*string*) – Video description.
- **`tags`** (*list of strings*) – *(default:* `[]` *)*
- **`category_id`** (*integer*) – Video category ID number.The default value is 22, which refers to the People & Blogs category.

- **`privacy_status`** (*`string`*) – Privacy status of the video. *(choices:* `'public'`, `'private'`, `'unlisted'` *) (default:* `u'public'` *)*
- **`url_callback`** (*`string`*) – URL to callback when the task completes successfully. See Callbacks and Errbacks for details.
- **`url_errback`** (*`string`*) – URL to callback when the task fails. See Callbacks and Errbacks for details.

Output Values

- **`output`** (*`string`*) – URL of the uploaded video on Youtube.
- **`duration`** (*`float`*) – Duration of the input video file, in seconds.

CHAPTER 5

History

5.1 10/02/2013 – /v2 API

Added support for Storage systems.

Tasks result objects also slightly change in this versions. Details of errors are now returned in the “error” key, instead of “result”. This avoids mixing successes and errors in code that doesn’t check the “status” key. See API Reference for details.

5.2 09/04/2013 – /v1 API

This is the first release of our API.

CHAPTER 6

Indices and tables

- genindex
- search

HTTP Routing Table

/v2

Index