ustudio Video API

Overview of Video Creation

A Video resource in the uStudio platform represents metadata and files associated with a video. Videos are resources that you can share, publish and distribute to multiple people and platforms.

An Asset is the actual bits and bytes of a single video file. It is attached to a Video resource once it is ready for storage, review, and delivery to Destinations (see the Destination API for more information).

A Video resource *must* exist in order to perform an upload to attach an asset to it.

Authentication

All requests to the uStudio API *must* contain the access token which will be provided by uStudio. The token may be passed in the `X-Auth-Token: <token>` header or the `token=<token>` query parameter. (See the Authentication document for more information.)

Studios

A Studio is the central organizational unit in uStudio. It represents an organization, or a large project, which contains a number of videos, destinations, and users who have been granted access to the Studio’s resources.

To retrieve a list of accessible Studios, perform the following request:

```
GET /api/v2/studios
```

This will return a list of Studios that the user is able to access:

```
{
    "studios": [
        {
            "owner": "username",
            "name": "Studio Name",
            "studio_url": "/api/v2/studios/(Studio UID)",
            "uid": "(Studio UID)"
        }
    ]
}
```

To retrieve more information about a single Studio, perform a GET request on the `studio_url` returned from list of Studios:

```
GET /api/v2/studios/(STUDIO_UID)
```

Many of the URLs used throughout the API documentation are obtained from the Studio response:

```
{
    "collections_url": "/api/v2/studios/(Studio UID)/collections",
    "description": "",
    "created": "2012-05-01 06:07:38",
    "destinations_url": "/api/v2/studios/(Studio UID)/destinations",
    "studio_url": "/api/v2/studios/(Studio UID)",
    "name": "Studio Name",
    "videos_url": "/api/v2/studios/(Studio UID)/videos",
    "uid": "(Studio UID)"
}
```

NOTE The previous example has been truncated, there are other fields that are provided from the individual Studio response that are referenced in other API documentation.

Every Studio has a Videos endpoint URL in its view:

`"videos_url": "https://app.ustudio.com/api/v2/studios/(Studio UID)/videos"`

This will be used for retrieving and creating Video resources for a given Studio.

Creating a Video

All Videos are created by making a POST request to the Videos endpoint, with the video metadata provided as JSON in the request body.

```
POST /api/v2/studios/(Studio ID)/videos

{
  "title": "Baking 101",
  "description": "Learning the art of cooking.",
  "keywords": ["food", "cooking"],
  "category" : "entertainment",
}
```

This will create a Video resource in the system. The Video resource holds metadata about a video project, such as the title, description, keywords, etc. The `category` is used by uStudio to map to categories across all Destinations.

The request will return a simple representation of the new Video resource:

```
{
    "category": "entertainment",
    "description": "Learning the art of cooking.",
    "video_url": "/api/v2/studios/OZOYbUDKV5iB/videos/UBY7PLoHj53g",
    "title": "Baking 101",
    "web_page": "",
    "created": "Thu Oct 31 14:39:04 2013",
    "image_url": null,
    "upload_url": "https://upload03-app.ustudio.com/api/v2/studios/(Studio UID
)/videos/UBY7PLoHj53g/asset",
    "uid": "UBY7PLoHj53g"
}
```

NOTE These representations are abridged for clarity.

Uploading the Video Asset

The “upload_url”

It is important to note the “upload_url” in the response from video creation. This is the endpoint to which a video asset is uploaded. This URL *must* be used to upload the Video.

The Asset

All Assets are created by making a POST request to the upload endpoint, specified in the video creation response, with multipart/form-data containing the raw video data.

```
POST https://(upload-server)/api/v2/studios/(Studio ID)/videos/(Video ID)/asse
t

...Multipart file content...
```

This resource handles uploads of a video asset to the system.

The video file should be submitted with the multipart identifier of `file`. In an HTML form, this is accomplished with the `name` attribute of an `input` tag:

```
<input type=”file” name=”file”/>
```

An example using cURL (a common command-line interface for making HTTP requests):

```
curl -X POST 'https://app.ustudio.com/api/v2/studios/
    (Studio UID)/videos/(Video UID)/asset?token=(Token)'
    --form file=@movie.mp4
```

Progress can be tracked by appending an `?X-Progress-ID=(identifer)` query argument to the initial request, and then performing additional GET requests to the same asset URL plus “/progress”. Providing a `?callback=functionName` allows the progress request to be included as a `<script>` tag. This is useful for producing progress bars or other indicators in user interfaces.

For instance, a progress request would look like:

```
<script src='https://app.ustudio.com/api/v2/studios/
    (Studio UID)/videos/(Video UID)/asset/progress?
    X-Progress-ID=(identifier)&callback=functionName'></script>
```

The result, wrapped in a callback, looks like:

```
functionName({
    "state": "uploading",
    "received": 819201,
    "size": 6190806
});
```

The `state` value will be `done` when the upload is finished.

After the upload completes an Asset resource will exist. The Asset resource holds information about a video's master file. The video file will be inspected, and the system will email the studio owner if there are any issues. Any file that is not a recognizable format will fail inspection.

NOTES

By default, uStudio will reject any file larger than five gigabytes (5GB). Additionally, it is recommended that only video files with two audio tracks (stereo) are uploaded to avoid processing errors. This information should be displayed somewhere near the upload form, where applicable.

Asset Information

By requesting information from the Asset resource, you can monitor the progress of the Asset through processing and inspection.

```
GET /api/v2/studios/(Studio UID)/videos/(Video UID)/asset
```

A fully inspected Video may look like:

```
{
    "mimetype": "video/mp4",
    "status": {"state": "finished"},
    "video_uid": "(Video UID)",
    "uid": "(Asset UID)",
    "download_url": "https://app.ustudio.com/api/v2/studios/(Studio UID)/video
s/(Video UID)/asset/file",
    "filename": "movie.mp4",
    "size": 46696109,
    "metadata": {
        "audio_channels": 2.0,
        "audio_format": "AAC",
        "duration": 44,
        "video_codec": "avc1",
        "audio_sample_rate": 44100.0,
        "video_size": [
            1280, 720
        ]
    }
}
```

The Artwork

By default, uStudio will automatically extract a preview image from the provided video. However, any application may upload a preview image to the Video by performing a POST request to the `image_upload_url` from a Video resource:

```
POST /api/v2/studios/(Studio UID)/videos/(Video UID)/image

...Multipart data...
```

The image file should be submitted with the multipart identifier of `image`. Also, it is recommended that a high-quality image is used (600x600 or higher, but in the proper aspect ratio) and that the format be JPEG.

Getting Video Metadata

The metadata for a video can be retrieved at any time by performing a GET to the video's URL, as provided in the JSON response when first creating the video, or in the response for the individual video when getting a list of videos:

```
"video_url": "/api/v2/studios/OZOYbUDKV5iB/videos/UBY7PLoHj53g"
```

The returned data will include the metadata for the video, as well as information about some useful records related to the individual video:

```
GET /api/v2/studios/(Studio UID)/videos/(Video UID)

{
    "category": "entertainment",
    "description": "Learning the art of cooking.",
    "video_url": "/api/v2/studios/OZOYbUDKV5iB/videos/UBY7PLoHj53g",
    "title": "Baking 101",
    "web_page": "",
    "created": "Thu Oct 31 14:39:04 2013",
    "image_url": null,
    "upload_url": "https://upload03-app.ustudio.com/api/v2/studios/(Studio UID
)/videos/UBY7PLoHj53g/asset",
    "uid": "UBY7PLoHj53g"
}
```

Note The information above is abridged, and more information will be shown in the actual response.

Updating Video Metadata

The metadata for a video may be updated at any time by performing a POST to the video's URL (see above). Any fields which may be set when creating a video may be changed. For example:

```
POST /api/v2/studios/(Studio UID)/videos/(Video UID)

{
    "title": "A new title",
    "description": "A new description",
    "keywords": ["new", "keywords"],
    "category": "art",
    "web_page": "http://my.website",
    "notes": "Notes not published to destination"
}
```

Any fields which are not specified in the request body will not be changed, so it is possible to update just one field of a video. Fields which are not writable will not be modified, so it is also possible to perform a GET, change a value in the response, and then POST the entire JSON back without error.

Custom Video Metadata

It is possible to provide custom video metadata fields, in addition to those provided by the base platform. The available fields are defined per studio, and field values can be set and retrieved on individual videos, so additional data may be associated with a video.

Field Types

Each video field added to a studio must have a field type, which describes the type of data stored in that field. The list of available field types may be queried from the `field_types_url` route on the studio:

```
"field_types_url": "https://app.ustudio.com/api/v2/studios/OZOYbUDKV5iB/field_
types"
```

Performing a GET on this URL will return a JSON encoded list of available field types:

```
GET /api/v2/studios/(Studio UID)/field_types

{
    "field_types": [
        {
            "uid": "text"
        },
        {
            "uid": "long_text"
        }
    ]
}
```

Video Fields

Video fields describe the custom fields that may be set and retrieved on individual videos. The available fields for a given studio may be retrieved from the `video_fields_url` route on the studio:

```
"video_fields_url": "https://app.ustudio.com/api/v2/studios/OZOYbUDKV5iB/video
_fields"
```

Performing a GET on this URL will return a list of the current custom video fields on the studio:

```
GET /api/v2/studios/(Studio UID)/video_fields

{
    "video_fields": [
        {
            "uid": "ticket_number",
            "url": "https://app.ustudio.com/api/v2/studios/OZOYbUDKV5iB/video_
fields/ticket_number",
            "type_uid": "text",
            "created": 12345678,
            "public": true,
            "label": "Ticket Number",
            "help": "The ticket number for the video in the ticketing system"
        },
        {
            "uid": "production_notes",
            "url": "https://app.ustudio.com/api/v2/studios/OZOYbUDKV5iB/video_
fields/production_notes",
            "type_uid": "long_text",
            "created": 12345679,
            "public": false,
            "label": "Production Notes",
            "help": "Private production notes, not for public consumption"
        }
    ]
}
```

The values in video field data have the following meanings:

- `uid`: The unique ID of the video field. This will be the key of the field in the video data.
- `url`: The URL of the field, to get, update or delete the field.
- `type_uid`: The UID of the type of the video field (see above).
- `created`: The Unix timestamp when the field was created.
- `public`: True if the field data may be sent to destinations which support it, when a video is published, False otherwise.
- `label`: A short, human readable name for the video field.
- `help`: A longer, human readable description of the purpose of the field.

The fields `label` and `help` are intended for use when building UIs for video fields.

CREATING VIDEO FIELDS

In order to create a video field, perform a POST with the definition of the field to the studio's `video_fields_url`:

```
POST /api/v2/studios/(Studio UID)/video_fields

{
    "uid": "production_notes",
    "type_uid": "long_text",
    "public": false,
    "label": "Production Notes",
    "help": "Private production notes, not for public consumption"
}
```

Note that, unlike most resources in the system, the client must provide the UID of the video field being created. This field UID must be unique to the studio, and it must be a valid C-like programming language identifier: it must start with a letter and contain only letters, numerals or underscore. Field UIDs may not start with an underscore, because they are reserved for internal use, and they may not conflict with any built in attributes of the video.

The fields `public`, `label` and `help` are optional. `public` will default to `false` if not set, and `label` and `help` will default to empty strings.

UPDATING VIDEO FIELDS

Fields may be updated by performing a PUT to the URL of the video field to be updated:

```
PUT /api/v2/studios/(Studio UID)/video_fields/(Video Field UID)

{
    "label": "New Label",
    "help": "A new, helpful description of the field"
}
```

Only the `label` and `help` attributes of the video field may be changed. It is not an error to include the `uid`, `public`, or `type_uid` fields in the request body, as long as the values are not changed, so that you may GET a video field, change a value, and PUT the entire field description back.

DELETING VIDEO FIELDS

Fields may be deleted by performing a DELETE, with no body, to the URL of the video field to be deleted:

```
DELETE /api/v2/studios/(Studio UID)/video_fields/(Video Field UID)
```

When a field is deleted, the values of that field are deleted from all the videos in the studio.

Video Field Data

Video field data is available on the video in the root of the video JSON, keyed by `field_uid`. For example, if there is a video field with UID `ticket_number` on a studio, an (abridged) view of the video might look like:

```
{
    "uid": "UBY7PLoHj53g",
    "title": "Video Title",
    "ticket_number": "PROD45830238"
}
```

Every video also includes a `fields` parameter, which lists all the video fields for that studio. This is intended to aid in developing UI tools for editing or displaying video metadata.

To set the value of a field on a video, simply POST to the video's URL with the field set in the JSON:

```
POST /api/v2/studios/(Studio UID)/videos/(Video UID)

{
    "field_uid": "field value"
}
```

If the field value is invalid for the type of the field, an error will be returned and the video will not be updated. Video fields and normal attributes of the video may be updated at the same time. Video fields not specified in the request will be left unchanged.