ustudio API Overview

Introduction

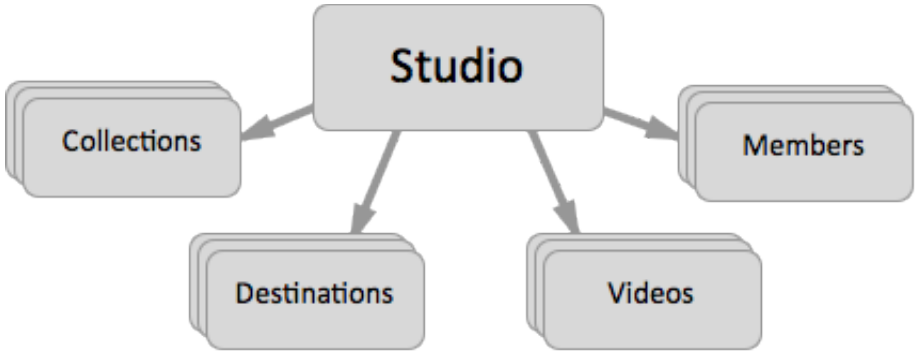
The uStudio Platform and APIs are designed to enable third-party applications to extend, customize, and white-label uStudio solutions for video management and distribution. By providing access to these products to selected partners, we hope to foster a strong community of uStudio-supported applications and provide extensive and growing integration potential.

The fundamental theme of the uStudio Platform is a separation between the various stages of a video's lifecycle, enabling complex but discrete workflows built from basic blocks of functionality. Everything from video management, collaboration, distribution, and measurement are exposed via various APIs and utilities, and partners can choose which tools make sense for their applications.

This document lays out the basic resources and operations of uStudio APIs, as well as a few implementation pointers.

Resource Overview

The uStudio Platform operates on several high-level resource types. The central resource in the system is the Studio – most resources are attached to a Studio either directly or through additional hierarchy. Since a user can be in multiple Studios, this enables simple organization management without complicated access control rules.



Studio Resource Hierarchy

As the above chart illustrates, each Studio owns a set of Collections, Destinations, and Videos in addition to the list of Users (identified as Members) who have access to the Studio itself.

Owners and Members

Every Studio has a single administrative user identified as an Owner. This user is able to add and remove other Users, as well as change high-level settings like the Studio name.

A Member has access to all other resources in the Studio. He or she is able to create and delete Videos and Collections, view statistics, create, update, delete, and publish to Destinations, etc. A Studio Member is considered a trusted user, most likely a member of the same organization as the Owner. For sharing resources with “untrusted” users, tools like Send Link and Video Sites are provided.

Videos

Arguably the most important resource in the system, a Video is the source of information about a video project. Although it is usually attached to a specific video file, the Video also tracks a thumbnail and various metadata about the contents of that project. This metadata is used when publishing in addition to various management and collaboration steps.

Collections

The Collection resource is a simple organizational unit. Functioning somewhere between a folder and a tag, Collections enable Videos to be placed into multiple buckets for purposes such as approval workflows, client management, or content grouping.

Destinations

A Destination resource is an abstract concept around a distribution endpoint. It might represent an Embeddable Player, a YouTube account, or an RSS feed. A Destination also manages all the requirements for authenticating users, publishing videos, and tracking activity, and removes the concerns of transcoding, packaging, and delivery from a partner application.

Destinations are derived from Destination Templates. This enables a single Studio to have multiple Players, YouTube accounts, or any other Destination Template type. The details of this relationship are identified in the Destination API document.

Other Resources

There are many other resources in the system such as Statistics, Assets, Destination Files, etc. that are not covered here. These are addressed in separate API documentation. In principal, with the exception of the User, most resources are attached to the Studio or to another Studio-based resource such as a Video or Destination.

API Fundamentals

The uStudio Platform is driven by a series of RESTful APIs that predominately “speak” in JSON. First, REST is a strong interface for representing resource relationships while simplifying complex interactions into straightforward, parallel operations on those resources. Second, REST is widely understood. JSON provides a ubiquitous, human-readable alternative to other formats such as XML, which pairs nicely with an interface like REST.

Most resources in the system function with the following pattern:

- Perform a `POST` request to a factory resource (such as `/api/v2/studios`) to create a new resource
- Perform a `GET` request on a factory resource to retrieve a list of simple JSON representations.
- Perform a `GET` request on a single resource (such as `/api/v2/studios/Ok82U2krWVzp`) to retrieve the full JSON representation of a resource.
- Perform a `PUT` request on a single resource to update attributes of the resource.
- Perform a `DELETE` request on a single resource to delete or remove the resource from that context.

Unless otherwise indicated, all requests should be JSON-encoded and all responses will be JSON-encoded as well.

Response Codes

For successful create requests, response codes of `201` indicate immediate creation, `202` indicate delayed creation based on external systems, and `204` indicates no response body. For successful updates, most resources will return a `200` or `204` (for no response body). Finally, successful deletes usually result in a `204`.

A response code of `400` is generally associated with an error in the response body indicating the issue with the request. `400` usually indicate malformed requests, missing required attributes, or other client issues. A `401` indicates a lack of authentication credentials, which often means the token was not provided in the request properly.

URLs

The base URL for all API operations is `https://app.ustudio.com/api/v2`.

Throughout the uStudio APIs, the resources provide URLs to other, associated resources or resource lists, so that an API client can easily

construct the next appropriate resource URL. Additionally, using these URLs instead of constructing them manually makes the integration code more “future-proof”.

For instance, performing a `GET` on the “current user” resource (https://app.ustudio.com/api/v2/users/me) provides the following response:

```
{
    "datastore_url": "/api/v2/users/A42BdvE3aC/datastore",
    "email": "john@ustudio.com",
    "name": "John McDuffy",
    "user_url": "/api/v2/users/A42BdvE3aC",
    "username": "A42BdvE3aC"
}
```

In this simplified example, there are two URLs provided by the User resource, the `datastore_url` and the `user_url`. They are provided so that an API client can access those resources for that user, without constructing them manually.

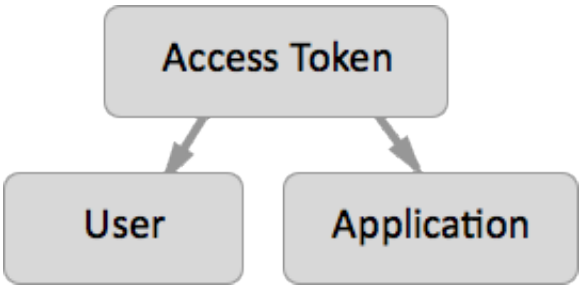
In that vein, we are currently moving from an older relative URL response (which you see above) to an absolute URL in the API responses. When using a URL from the system, it is always best to ensure it isn’t already fully-qualified. The following is a simple function in PHP to validate a URL:

```
function qualify($url) {
    if (!substr($url, 0, 4) == "http") {
        return "https://app.ustudio.com" . $url;
    }
    return $url;
}
```

This ensures that a.) current relative URLs work and b.) future absolute URLs won’t break a legacy implementation.

Authentication

In order to access the API, most operations require an Access Token to act on behalf of a User. Access Tokens are a combination of an Application's abilities and a User's permissions. For instance, if a User is allowed to delete a Video from a Studio, but the Application is limited to read-only status, then the action would be rejected.



User Access Token

Users are granted a general purpose Access Token when they are created. Many integrations do not require any more than a single user's token – for those services, simply contact support to get the access token for a specific owner or member account.

For services that need to manage multiple User accounts in uStudio, an Application will need to be created. uStudio Applications are configured for particular methods of authentication. A few are allowed to authenticate based on credentials (email and password), some are allowed to create users, and still others use an OAuth-based authentication mechanism.

Based on your integration needs, you will need to speak with uStudio support to get the Access Token for a specific user, or a custom Application with the appropriate credentials and permissions.

Once an Access Token has been obtained, all resource requests must provide it. There are two ways to send a Token with a request – via header or query argument. The query argument route simply involves appending it to the URL like this:

```
https://app.ustudio.com/api/v2/users/me?token=a6bf156153ea2b02a3a1a6b2111ef17a
```

Alternatively, providing the token as a header argument is an option. To do this, simply construct a header entry that looks like:

```
X-Auth-Token: a6bf156153ea2b02a3a1a6b2111ef17a
```