BACKLOT DEVELOPER GUIDE

CONTENTS

USING THE BACKLOT API

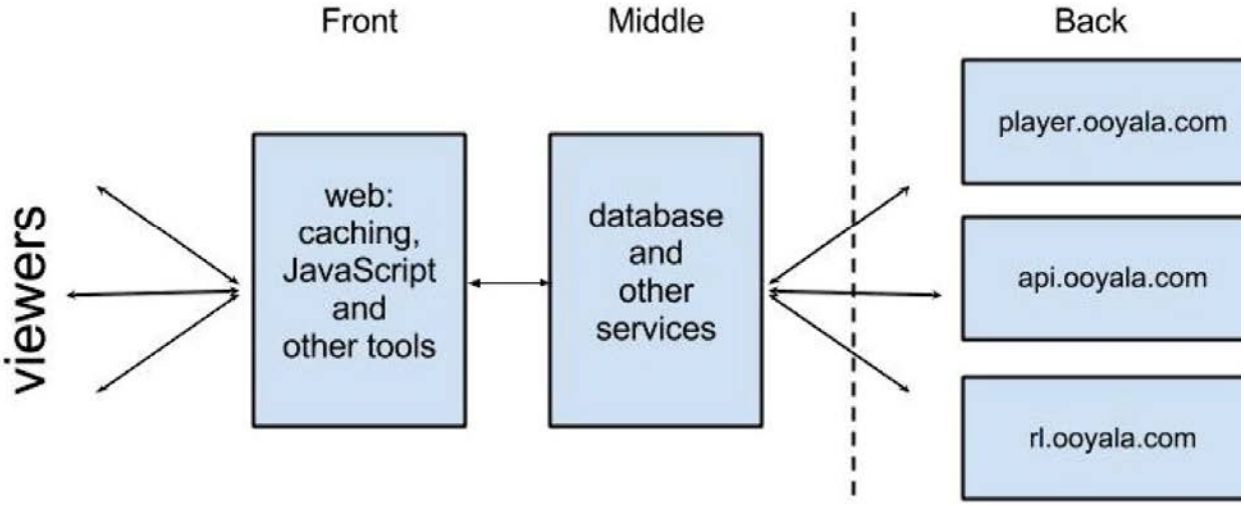
Ooyala provides a REST-based interface that enables you to make requests to the Backlot platform.

If API access is enabled for your user account, Ooyala provides you with an API Key and a Secret Key. If you use the Ooyala SDKs, they automatically compute the signature based on these credentials. If you make HTTP calls directly, you must write the code to compute the signature.

COMMON DEPLOYMENT ARCHITECTURE: MULTIPLE TIERS

Ooyala recommends a multi-tier design for your server deployments.

For best performance for your viewers and efficiency of your own services, plan a server architecture with "separation of concerns," a typical design with multiple tiers of servers, each tier dedicated to fulfilling some portion of the overall service. A simple conceptual view is shown below, with the commonplace tiers of front, middle, and back.



Front — At the front is your web site.

Ooyala recommends that you implement a caching subsystem on the front to hold results of database or other queries. This is the most performant way to serve your viewers. *Best Practices* on page 6 has more information.

A representative programming language on the front is JavaScript, with auxiliary tools like JQuery (for database retrieval) and others. (Of course, many different programming languages are used.)

Middle — Typically the middle tier is some sort of database or other services intermediate between front and back.

The middle communicates with Ooyala services on the back through application programming interfaces (APIs) and other mechanisms. Data retrieved from Ooyala is commonly stored in the middle database for serving the front.

	Not all architectures use or need a middle tier, which is probably most common in large scale deployments. If there is no middle, then the front itself commonly serves the functions that might commonly be relegated to the middle.
Back	In this simple recommended architecture, the back is composed of Ooyala's API and other services. (Not all services are shown, nor are your own back-end services.) For a list of API server names and an overview of their functions, see *API Server Endpoints* on page 9.

BEST PRACTICES

Ooyala provides a set of APIs that enable you to create rich and unique experiences on your web sites. Here are some best practices to follow.

SECURITY OF YOUR API KEYS

Keep your Ooyala API key, secret, and provider code safe and secure. Avoid embedding the keys or code directly in your applications. Allow only secure processes to read your secret and key.

Ooyala technical support or other personnel will never ask for security keys.

CACHING OR DATABASE FOR YOUR FRONT-END SERVER

- To provide speedy responses to your customers, set up caching or a database on your web site so that you avoid making many calls to the Backlot APIs during user interaction with your site. For more details, see *Advice for Web Programming* on page 6.
- In addition, take advantage of Ooyala's high performance API endpoint (cdn-api.ooyala.com). It caches repeated requests from large volume web sites. Depending on the type of content, round your request expiration times up to the next hour, four hours, eight hours, or day. For example, if your data changes frequently, round up to the next hour. If your data doesn't change often, round up to the next day. For more details, see *High Performance API Endpoint* on page 13

ERROR HANDLING

Make sure your code handles errors and that your web site can gracefully display content in case of a problem.

RATE-LIMIT MANAGEMENT

Rate-limited requests return the number of seconds before rate limiting is reset. Configure your code to wait the specified number of seconds before retrying. For more detail, see *API Rate Limiting* on page 13.

ADVICE FOR WEB PROGRAMMING

Here are some recommendations for developers who use the Ooyala APIs and who can benefit from recommendations about programming for the web.

These recommendations are based on the experience of Ooyala's engineering and professional services experts and are tips for putting in practice Ooyala's *Best Practices* on page 6.

SET UP WEB CACHING

For best performance for your customers, set up a caching system for the responses from the Backlot API. One very popular and free program for caching is memcache, which comes free on Linux and is available for Microsoft Windows or Apple systems.

For instance, in the middle of the night (or whenever your front-end system is least active), you can set up a cron job (on Linux) that uses the Backlot API's `/v2/assets` route to retrieve all your videos (assets) from Backlot. With memcache running, the references to these videos are stored in case of future requests. When a request for an asset is received from a user, your program checks if the requested asset has already been cached. If so, the video is displayed; if not, a call to Backlot API retrieves it, stores it for future reference, and displays it. An example of coding with memcache in PHP is viewable at *http://www.search-this.com/2007/07/24/an-introduction-to-memcached/*

There is also a wealth of information on the Internet about this subject, such as *http://www.mnot.net/cache_docs/*.

LEARN ABOUT PROGRAMMING FOR MOBILE WEB

Some recommendations[1]:

1. Use basic HTML and CSS.
2. Keep the size of images small for fast downloading.
3. Use redirects with browser detection.
4. Usability issues: do not overload your pages. Provide the necessary basics in an easy-to-use size.
    - Avoid too many navigation links.
    - Minimize the amount of content on a page.
    - Presentational items can get in the way.
    - Provide a link to a full-sized version, if desired.

Other sources of advice:

- *http://mobilewebbook.com/*
- *http://www.teehanlax.com/blog/iphone-gui-psd/*
- For iPhone:
    - *http://mobiledesign.org/*
    - *http://cssiphone.com/*

TAKE ADVANTAGE OF YOUR BROWSER TOOLS

Become familiar with your browser's debugging tools.

For example, if you are developing for the iPad or iPhone, you can use the Apple Safari or Mozilla Firefox web browser to emulate those devices by changing the user agent. See *http://www.mydigitallife.info/how-to-emulate-iphone-change-user-agent-in-safari-and-firefox-web-browser/*.

[1] *http://www.slideshare.net/nickwhitmoyer/mobile-web-design-the-basics*

REST OVERVIEW

Every resource in the Backlot API has its own URL. You can view, modify and delete those resources by sending requests to the resource's URL and using standard HTTP methods in your requests.

You can use any HTTP library or tool when making requests. The Backlot API supports the following HTTP methods, each having the conventionalized functions described below.

- GET: view a resource
- POST: create a new resource
- DELETE: delete a resource
- PUT: replace an existing resource; create a resource
- PATCH: update or modify an existing resource

Note:

All requests must be made using HTTPS.

Not all routes support all of the requests stated above.

In general, a POST request accepts the same parameters as a PATCH to that resource.

When creating a new resource, you use a POST request. Depending on the requirements of the resource, you configure its settings by including a JSON representation of an object in the body of the request. The following example creates a label, only specifying its name:

```
[POST]/v2/labels{
   "name":"Leisure"
}
```

Although you only configured the name, Backlot automatically configures the remaining settings using the defaults.

If the request is successful, the Backlot API returns a response similar to the following:

```
{
   "name":"Leisure",
   "id":"9d765ce4b3884f8",
   "full_name":"/Leisure",
   "parent_id":null
}
```

You can get this information at any time by issuing a GET request. For example, the following request returns the settings for the label with the `9d765ce4b3884f8` ID.

```
[GET]/v2/labels/9d765ce4b3884f8
```

With this information, you can modify the resource using a PATCH request. For example, the following request changes the name of the label to `Recreation`:

```
[PATCH]/v2/labels/9d765ce4b3884f8{
   "name":"Recreation"
}
```

If the request is successful, the Backlot API returns a response similar to the following:

```
{
   "name":"Recreation",
```


```
    "id":"9d765ce4b3884f8",
    "full_name":"/Recreation",
    "parent_id":null
}
```

CHARACTER ENCODING AND MIME TYPES

Encode your data in UTF-8 and set the proper MIME type on requests with bodies.

The API requires the following:

1. Make sure that your character data are encoded in UTF-8.
2. On HTTP requests that contain a request body in JSON format (POST, PUT, or PATCH), make sure to set the MIME type:

   ```
   Content-type: application/json
   ```

3. Make sure non-ASCII letters and ' characters are escaped.

API SERVER ENDPOINTS

Make your requests to these endpoints.

Ooyala has several API server endpoints. Use the one appropriate to your needs.

Table 1: Server Endpoints

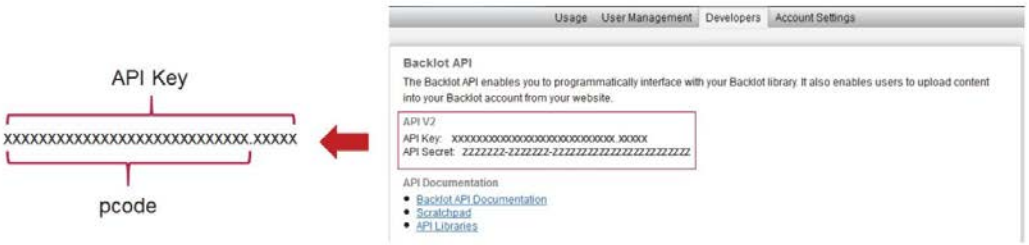
Endpoint	Service	Description
player.ooyala.com/	Player API	For video playback.
http://api.ooyala.com/	Backlot API (including Ooyala IQ v3 Analytics Reporting API)	For normal usage of the Backlot API. Requests to this endpoint are rate-limited; see *API Rate Limiting* on page 13.
https://cdn-api.ooyala.com/	Backlot API	High-performance endpoint for the Backlot API: unlimited `GET` requests against the cache. See *High Performance API Endpoint* on page 13.
http://rl.ooyala.com	Ooyala Rights Locker	Ooyala Rights Locker entitlements. See *http://support.ooyala.com/documentation/developers/chapter_rightslocker.html*.

YOUR API CREDENTIALS

Your Ooyala API credentials include your API key and secret, which you provide to sign HTTP requests and when using the scratchpad.

Regardless of which endpoint you use, all requests must be signed by your Ooyala API credentials (your API key and API secret), and a signature computed from these and the request arguments. The API key and secret are viewable in your Backlot UI account. In the Backlot UI, go to the **ACCOUNT** tab, **Developers** subtab; the key and secret are on the left.

In addition, if you are a publisher (sometimes called "provider" or "partner"), Ooyala assigns you a `pcode`, which is required on some API requests. An API key has the following format: *<pcode>.<string>*. Your `pcode` is the alphanumeric string that precedes the period. For example, if the API key is `xxxxxxxxxxxxxxxxxxxxxxxxxxxx.xxxxx`, then use the first 28 characters (`xxxxxxxxxxxxxxxxxxxxxxxxxxxx`) for the pcode.



For the general algorithm for signing requests, see *General Algorithm for Signing Requests* on page 10.

For code samples in various programming languages, see *Sample Code for Signing Requests* on page 173 on the Support Site.

In addition, you can use your API credentials to make Backlot and v3 Analytics requests on the scratchpad at *https://api.ooyala.com/docs/api_scratchpad*.

GENERAL ALGORITHM FOR SIGNING REQUESTS

Every request made to Backlot requires three query string parameters for authentication: the API Key, the request expiration, and the signature.

To sign a request:

1. Start with your 40 character secret key (see the **Developers** tab in the Backlot UI); it is unique for each user and should always be kept secure and private. For details, see *Your API Credentials* on page 9.

   This example uses`329b5b204d0f11e0a2d060334bfffe90ab18xqh5` as the secret key.

   ```
   fakef391e36243a94d54e5da667020d3
   ```

2. Append the HTTP method (e.g. "GET", "POST", "PUT"):

   ```
   fakef391e36243a94d54e5da667020d3GET
   ```

3. Append the request path or route:

   ```
   fakef391e36243a94d54e5da667020d3GET/v2/players/HbxJK
   ```

4. Append any query string parameters, sorted alphabetically by keys. This includes the required API Key (see the **Developers** tab in the Backlot UI) and the `expires` parameter.

   Note: Do not URL-encode these parameters.

   ```
   fakef391e36243a94d54e5da667020d3GET/v2/players/
   HbxJKMapi_key=7ab06expires=1299991855
   ```

5. If your request has a body, append the entire request body to the string.
6. From this string, you can now generate a SHA-256 digest in base64, truncate the string to 43 characters, and remove any trailing = signs. This example produces the following signature:

```
p9DG/+ummS0YcTNOYHtykdjw5N2n5s81OigJfdgHPTA
```

7. URL encode the signature:

```
p9DG%2F%2BummS0YcTNOYHtykdjw5N2n5s81OigJfdgHPTA
```

8. Append this signature to your request URL as a query string parameter. You can now visit this URL to make your request. The following example is the final signed URL:

```
https://api.ooyala.com/v2/players/HbxJKM?
api_key=7ab06&expires=1299991855&signature=p9DG%2F
%2BummS0YcTNOYHtykdjw5N2n5s81OigJfdgHPTA
```

SETTING AN EXPIRATION TIME ON REQUESTS

You must set the `expires` parameter on your HTTP requests.

Your HTTP requests must include the `expires` parameter to indicate when the request becomes invalid.

The value for the `expires` parameter is the time in seconds since January 1, 1970 ("the epoch") after which the request is invalid, as shown in the following example:

```
[GET]https://api.ooyala.com/v2/players/somePlayerId?
api_key=someKey&signature=someSignature&expires=1930294539
```

API PING URL

The **API Ping URL** is invoked whenever transcoding of a file completes.

In the Backlot UI, go to the **ACCOUNT** tab, **Developers** subtab, you can set the **API Ping URL**. The **API Ping URL** is for you to receive a GET request containing the `embedCode` (content ID or asset ID assigned to a video or other asset) whenever transcoding completes for an API-uploaded movie.

The notification usually occurs only once per movie unless the content is reprocessed or replaced. Multiple URLs separated by commas can be specified. Maximum character length is 4,096 bytes. The URL you enter must be reachable at the time you enter it; the system checks whether it is a valid URL or not.

For example, if you set the **API Ping URL** as follows, when processing is complete, Backlot executes it with embed code in the query string:

```
http://foobar.com?foo=bar
[GET] http://foobar.com?foo=bar&embedCode=ReturnedEmbedCodeHere
```


Example of a commonly named **API Ping URL**:

```
http://example.com/ooyalapostback
```

PRACTICE MAKING REQUESTS WITH THE SCRATCHPAD

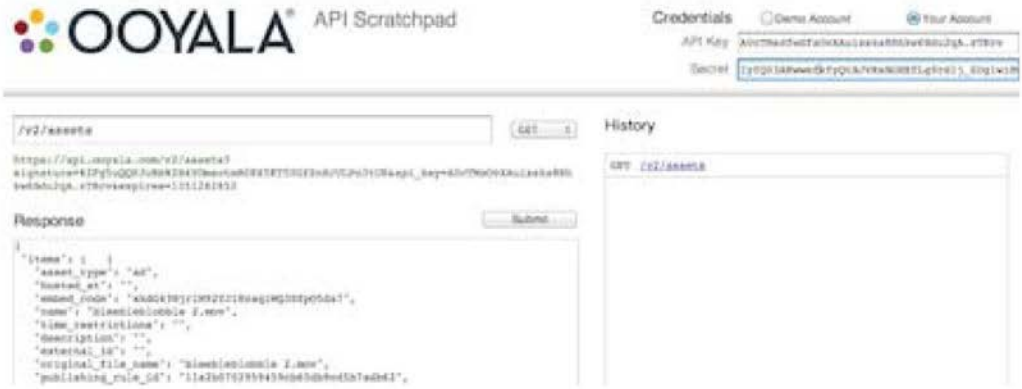
You can make practice requests on the scratchpad.

For your experimentation and learning, Ooyala provides a "scratchpad" where you can practice making requests, see the structure of responses, and familiarize yourself with Ooyala's APIs in general.

Note: The scratchpad accesses your live information and assets. Any changes you make in the scratchpad are reflected in production.

The scratchpad is located at *https://api.ooyala.com/docs/api_scratchpad*.

In the upper right, you can choose to use either a demo account (which allows only `GET` requests) or your own account (which allows all methods). To use your own account, provide your account's Ooyala API credentials, which are described in *Your API Credentials* on page 9.



v3 Analytics (Ooyala IQ) API

The following applies only if you are migrated to Ooyala IQ. For v3 Analytics (Ooyala IQ), prefix /v3 to your route. For example, to retrieve the performance report, select GET and input the following:

```
 /v3/analytics/reports/?
report_type=performance&dimensions=asset&start_date=2015-01-01&end_date=2015-01-07
```

WRAPPER SDKS/API LIBRARIES

Several SDKs for different languages are available.

Ooyala offers easy-to-use "wrapper" software development kits (SDKs) for the Backlot REST-based APIs in a variety of programming languages. For example, to create new request and sign it is as easy as the following (in Java):

```
OoyalaAPI api = new OoyalaAPI("apikey", "secret key");
```

The wrapper SDKs are available from GitHub for the following languages:

- *Java*

- *PHP*
- *Python*
- *Ruby*
- *C#*
- *Objective C*

In addition, sample code for computing signatures and signing requests in a number of programming languages (without the wrapper SDKs) is also available. See *Sample Code for Signing Requests* on page 173.

API RATE LIMITING

Through the Ooyala APIs, you are allowed 300 requests per minute. Each minute, the counter resets.

All API requests cost one credit unless otherwise specified. If you do not have enough credits to make an API request you will receive a 429 response.

Each request made to our APIs, successful or not, returns the following headers:

```
X-RateLimit-Credits  # The number of credits you have remaining.
X-RateLimit-Reset    # The number of seconds until your credits reset.
```

If you find yourself continually hitting the limit, try the following to decrease your API usage:

- Instead of making a request every time you need some information, try caching responses locally and only make requests when you expect that information to change.
- Keep an eye on your remaining credits, and if you are low try backing off the number of requests you are making until they reset.
- You might have some API users that you would like to prioritize. If so, only let your high priority API users make requests when you are low on credits.

HIGH PERFORMANCE API ENDPOINT

The high performance API endpoint caches repeated requests from large volume web sites.

To use the High Performance API endpoint, make `GET` requests to the content delivery network (CDN):

`https://cdn-api.ooyala.com`

The performance of the CDN high performance API endpoint often exceeds that of the normal service. The CDN is also geographically distributed, so response time is better in various parts of the world.

The CDN endpoint:

- Accepts either HTTPS (preferred) or HTTP requests.
- Accepts only `GET` requests.

CACHE-FRIENDLY `EXPIRES` PARAMETER

The CDN caching layer maintains a copy of each response for approximately five minutes. When making requests, the CDN caches your exact request to return the same cached results in the future. If you change any part of the request, the CDN treats it as a new request. To ensure the best performance from

the High Performance API, make sure the request expires parameter is "cache friendly"; that is, not too short. In the following Ruby snippet, the request always expires at the start of tomorrow:

```
today = Date.today
expires = Time.mktime(today.year, today.month, today.day) + 86400
```

If you are generating URLs to use in an embedded Flash or iOS application, you can generate a URL that expires years in the future.

JSONP

You can specify the name of a JavaScript callback function on `GET` requests to the high performance API endpoint.

To avoid cross-domain restrictions when invoking API requests from web pages, on `GET` requests from embedded JavaScript `<script>`s you can specify the name of a JavaScript callback function by using the callback query string parameter. This is supported only on `GET` requests.

Note: For performance, JSONP is available only from the high performance cached API. For more information, see *High Performance API Endpoint* on page 13.

In the following example, a standard `GET` request returns all labels within the account:

```
[GET]/v2/labels
```

Backlot returns a response similar to the following:


```
{
   "items":[
      {
         "full_name":"/Hobbies/Hockey",
         "id":"85042f300fc143c093e8f4ee01892af8",
         "name":"Hockey",
         "parent_id":"d5751b77a0c24972888bf906734d8c34"
      },
      {
         "full_name":"/Hobbies",
         "id":"d5751b77a0c24972888bf906734d8c34",
         "name":"Hobbies",
         "parent_id":null
      },
      {
         "full_name":"/Sports/Motorcycle Racing",
         "id":"bace921fdea44cc18a5a273155514522",
         "name":"Motorcycle Racing",
         "parent_id":"814efb109416490a98ee3f4fcd6784cf"
      },
      {
         "full_name":"/Sports",
         "id":"814efb109416490a98ee3f4fcd6784cf",
         "name":"Sports",
         "parent_id":null
      }
   ]
}
```

In the following example, when the `GET` request is triggered from a `<script>`, the results are returned with the `get_my_labels` function.

```
[GET]/v2/labels?callback=get_my_labels
```

Backlot returns a response similar to the following:

```
 get_my_labels({"items":
[{"name":"Hobbies","full_name":"/
Hobbies","id":"d5751b77a0c24972888bf906734d8c34","parent_id":null},
{"name":"Hockey","full_name":"/Hobbies/
Hockey","id":"85042f300fc143c093e8f4ee01892af8","parent_id":"d5751b77a0c24972888bf90673·
{"name":"Motorcycle Racing","full_name":"/Sports/Motorcycle
 Racing","id":"bace921fdea44cc18a5a273155514522","parent_id":"814efb109416490a98ee3f4fc·
{"name":"Sports","full_name":"/
Sports","id":"814efb109416490a98ee3f4fcd6784cf","parent_id":null}]})
```

HTTP RESPONSE CODES

Depending on request results, the Backlot API returns different HTTP response codes.

Response Code	Short Description	Comments
200	OK	The Backlot API successfully processed the request.
204	No content	The server has fulfilled the request, but does not need to return an entity-body. The response does not have a message body.
400	Bad request	The request is poorly formed or contains invalid data. For example, the Backlot API returns 400 response code if your request contains malformed JSON or invalid settings, such as a time setting of "25:00:00".
401	Not authorized	The request is not properly signed or signed with an invalid API key.
403	Forbidden	The request is not properly authenticated for the requested operation.
404	Not found	The resource does not exist. For example, if you tried to delete a video using the wrong content ID.
429		Insufficient "API credits" to fulfill request. For explanation, see *http://support.ooyala.com/developers/documentation/concepts/api_rate_limiting.html*

HOW TO EXPORT YOUR DATA WITH THE V2 ANALYTICS API

INTRODUCTION

The following information will guide you on retrieving and saving your analytics v2 data.

As we migrate customers from v2 Analytics to Ooyala IQ (v3 Analytics), you need to know the following:

- All customers will have access to reprocessed data starting from January 1, 2014.
- You will continue to have access to the old v2 Analytics API until March 31, 2016.

This means that if you need access to more than 1 year of historical analytics data, you need to export the data using the v2 APIs while they are still active.

OVERVIEW: THE OOYALA V2 ANALYTICS API

With the Ooyala v2 Analytics API you can easily create a report that will provide you with your analytics data. The results will be in JSON format. (JSON is a lightweight data-interchange format that is easy to read and write.)

To retrieve analytics results from a specific date range you simply need to define the type of result you need with the api call /v2/analytics/reports/, either in a terminal emulator such as Terminal on a Mac or in the Ooyala Scratchpad.

What is an API?

In computer programming. **API (Application Programming Interface)** is the name of a set of routines and protocols for software applications. An **API** expresses a software component in terms of its operations and results.

Where can I locate my API Credentials?

You can locate your API Key and Secret in the Backlot UI. Please use your API v2 credentials located in Backlot under the ACCOUNT>Developers tab.

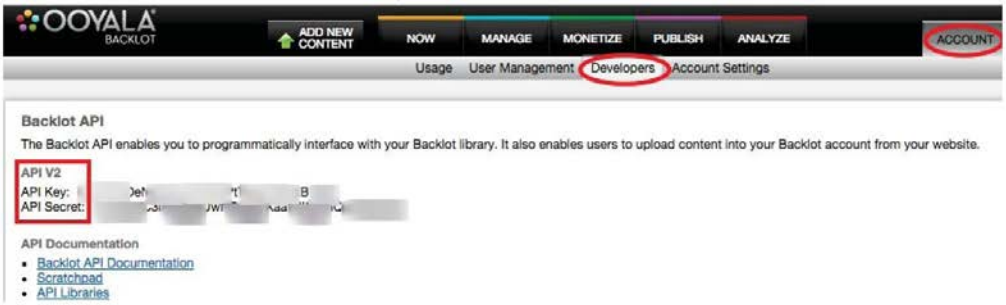


Figure 1: Finding Your API Keys

How Does the API Work?

An Ooyala API call requires 4 basic elements: API Key, API Secret, a Signature and an Expiration time.

API calls are made via *HTTP* methods. The GET API call is used to retrieve data without directly modifying it and allows you to get a typed JSON document response based on the id of the object.

What are the Available Results I can Retrieve with the Ooyala v2 Analytics API?

On the v2 Analytics API there are 4 key qualifiers that you need to identify in order to retrieve your results:

asset_id: This value is referred to by different names depending on where you look for it. In the Backlot API, asset_id is the identifier for a specific asset. asset_id has the same value as the content ID found in the Backlot UI that represents a piece of content. The value is the same for all scenarios. asset_id can be used if you want to retrieve results for a specific asset. Alternatively, you can retrieve results for your account, which would include all of your assets.

date_range: Defines the date range for this report. Analytics is based on dates. You can define the date following the format (YYYY-MM-DD) or you can define a date range with (YYYY-MM-DD...YYYY-MM-DD).

Report Request: Defines the type of report you want to retrieve. Valid values include *performance*, *sharing*, *engagement*, and *delivery*.

Dimension: Dimensions are common criteria that are used to aggregate data, such as the date when the user activity occurred or the country where the users were located. Every Dimension also has "**Drilldowns**", which allow you to filter your results by specific values for each dimension.

Common query string parameters and attributes can be found at *Common Attributes and Query String Parameters*.

HOW TO RETRIEVE YOUR V2 ANALYTICS DATA

What is the API Call Format?

The v2 Analytics API follows a specific order on the body of the API.

For example, the API call: `/v2/analytics/reports/account/performance/[:dimension/:drilldown]/:date_range`

Should appear like the following if you want to get a performance report from 2011-01-01 to 2014-01-01:

Figure 2: API Call For Performance Report

How Can I Use the Scratchpad to Save Reports?

The Scratchpad is a tool created by Ooyala that allows you to make API queries in your browser.

To retrieve an analytics report with the Scratchpad:

1. Go to *https://api.ooyala.com/docs/api_scratchpad?url=*.
2. Select "Your Account" in the Credentials section in the upper right corner of the page.
3. Enter your v2 API credentials (API Key and Secret) in the Credentials section in the upper right corner of the page.
4. In the Query field located on the left side of the page, enter your Analytics query. For example, if you would like to get the performance report from 2011 to 2014, copy and paste this query: `/v2/analytics/reports/account/performance/total/2011-01-01...2014-01-01`
5. Select GET.
6. Click Submit.

   Note: Your response appears in the response field.
7. If you prefer to see your results in a larger browser window, copy the API URL shown in green above the Submit button into your browser. In this case, the URL would be *https://api.ooyala.com/v2/analytics/reports/account/performance/total/2011-01-01...2014-01-01?api_key=yourApiKey&signature=yourSignature&expires=1418771221*.
8. Save the JSON by selecting File > Save As… in your browser. For information on how to convert JSON to CSV, see *Converting Analytics JSON to CSV* on page 128

Additional Query Examples

For more details on how to form queries in Scratchpad and for specific analytics report types, see:

- *Performance* query examples
- *Sharing* query examples
- *Engagement* query examples
- *Delivery* query examples

Note: To retrieve all data for a report type for your account, use the "total" query string parameter. You can find examples using "total" in each of the query example links mentioned above. "total" is used to retrieve all data for that particular report type for your account.

For example, the following query retrieves all performance data for the account over the date range 2011-08-01...2011-08-02.

```
[GET]/v2/analytics/reports/account/performance/total/2011-08-01...2011-08-02
```

How can I Create my own API Report?

You should only create your own API script if you are comfortable with the Ooyala API and have created scripts before or if you have the technical resources available who can modify the pre-made query to retrieve the data for you.

If you check the following snippet from our *sample code* that shows a terminal, you will be able to identify that we send the request of the API call using *cURL*. cURL makes http request where you can modify the parameters and the headers.

```
$apiSecret = "xxxxxx";          ---> Your API V2 Credentials
$apiKey = "xxxxxx";

$restMethod = 'GET';            ---> HTTP Method

$requestPath = '/v2/analytics/reports/account/performance/total/2011-01-01...2014-01-01';  ---> API Call

$expDate = time() + 1209600;    ---> Expiration time

$parameters = array( "expires" => $expDate, "api_key" => $apiKey, );

$content = "";

$signature = createSignature($apiSecret, $restMethod, $requestPath,  $parameters, $content);  ---> Signature

$url = 'https://api.ooyala.com'.$requestPath.'?&api_key=txxxxxxxxxxx&expires='.$expDate.'&signature='.$signature;

echo $url;

$ch = curl_init($url);          ---> Curl request to the api url
curl_setopt ($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
curl_setopt ($ch, CURLOPT_HTTPHEADER, Array("Content-Type: application/json"));
curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "GET");
curl_setopt($ch, CURLOPT_POSTFIELDS, $content);
```

Figure 3: Sample API Report

Note: For more script examples, please check our support site documentation at *Sample Code for Signing Requests* on page 173.

API REFERENCE

The Backlot API Reference has full documentation.

For full documentation for all Backlot API routes, query string parameters, and more, see the *Backlot API Reference*.

MANAGING ASSETS

In Backlot, an asset is a piece of content of various types, such as video, audio, live stream, and more.

It is identified by a unique identifier, which is called variously the *embed_code* (in the API), *content ID* (in the Backlot UI), or *asset_id* (in the documentation).

TYPES OF ASSETS AND CORRELATION TO /V2/ASSETS QUALIFIERS

The `/v2/assets` route deals with all types of assets.

When you create a new asset, the `asset_type` property in the request body can specify any of the following values.

- video
- ad: An ad is simply a video, marked as an ad in the Backlot UI.
- remote_asset
- youtube: A video destined for YouTube
- live_stream
- live_audio
- channel
- channel_set

After an asset is created with `[POST] /v2/assets` (as described in *Creating an Asset: POST* on page 20), in general you can update it in a variety of ways with `[PATCH] /v2/assets`. See the generalized process in *Editing Asset Information: PATCH* on page 21.

In addition, the `/v2/assets` route has qualifiers for working with assets in general or specific types of assets, such as associating labels, publishing rules, ad sets, and more. Exact use of these qualifiers is detailed in other sections.

Table 2: /v2/assets Qualifiers by Asset Type

	video	ad	audio	remote_ asset	you tube	live_ stream	live_ audio	channel	channel_ set
/ad_set	X	X							
/closed_captions	X	X	X	X	X	X	X	X	X
/drm_attributes	X	X	X	X	X	X	X	X	X
/generated_preview_images	X	X	X						
/labels	X	X	X	X	X	X	X	X	X
/lineup									
/metadata	X	X	X	X	X	X	X	X	X
/player	X	X	X	X	X	X	X	X	X
/preview_image_files	X	X	X	X	X	X	X	X	X
/preview_image_urls	X	X	X	X	X	X	X	X	X
/primary_preview_image	X	X	X	X	X	X	X	X	X

/publishing_rule	X	X	X	X	X	X	X	X	X
/source_file_info	X	X	X						
/streams	X	X	X						
/upload_status	X	X	X		X				
/uploading_urls	X	X	X		X				
/youtube	X			X	X				
/replacement	X	X	X		X				

CREATING AN ASSET: POST

To create an asset, use `POST /v2/assets` with properties in the body of the request.

Note: The following steps are for the creation of general assets.

- For Remote Assets, see *Remote Assets* on page 45.
- For Live Streams, see *Live Streams* on page 42.
- For Channel Sets, see *Channels and Channel Sets* on page 50.
- For YouTube videos, see *Uploading a YouTube Video* on page 47

Use the `POST` method with the `/v2/assets` route and properties in the request body.

Note: In addition to creating the asset for them, video and audio files must also be uploaded; see *Uploading a Video or Audio Asset: Four Steps* on page 35.

Note: Properties in the body of the POST vary by asset type.

```
[POST]/v2/assets{
   "asset_type":"video",
   "duration":0,
   "name":"My Video",
   "preview_image_url":null,
   "created_at":"2011-07-22T18:54:19+00:00",
   "embed_code":"Y1dTdvMjq9QtOMGrP-H59OIgiZ6-_Mrl",
   "time_restrictions":null,
   "updated_at":"2011-07-22T18:54:19+00:00",
   "external_id":null,
   "description":null,
   "status":"uploading",
}
```

Backlot returns a response similar to the following. Note the `embed_code` property. This is the key to the asset (referred to as `asset_id` in the documentation) and is required to update, delete, or work with the asset in general.

```
{
   "asset_type":"video",
   "duration":30,
   "name":"My Video",
   "preview_image_url":null,
   "created_at":"2011-07-22T18:54:19+00:00",
   "embed_code":"Y1dTdvMjq9QtOM",
   "time_restrictions":null,
   "updated_at":"2011-07-22T20:51:07+00:00",
   "external_id":null,
   "hosted_at":"http://www.mydomain.com",
```


```
    "original_file_name":"my_video.avi"
}
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

The asset is successfully created.

VIEWING ASSET INFORMATION: GET

To view an asset's information, use `GET /v2/assets/asset_id`. To view information about all assets, do not specify an *asset_id*.

To view asset information:

1. For a single asset, use the `GET` method with the `/v2/assets/asset_id` route.

```
[GET]/v2/assets/Y1dTdvMjq9QtOM
```

Backlot returns a response similar to the following.

```
{
   "asset_type":"video",
   "duration":30,
   "name":"My Awesome Video",
   "preview_image_url":null,
   "created_at":"2011-07-22T18:54:19+00:00",
   "embed_code":"Y1dTdvMjq9QtOM",
   "time_restrictions":null,
   "updated_at":"2011-07-22T20:51:07+00:00",
   "external_id":null,
   "hosted_at":"http://www.mydomain.com",
   "original_file_name":"my_video.avi",
   "description":"The next Internet meme.",
   "status":"live"
}
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

2. To view information about all assets, do not specify an *asset_id*.

```
[GET]/v2/assets
```

Backlot returns information about all assets.

EDITING ASSET INFORMATION: PATCH

To change an asset's information, use `PATCH /v2/assets/asset_id`.

To edit asset information:

Use the `PATCH` method with the `/v2/assets` route, with the properties to be changed in the request body.

The following example changes the name, hosted at URL, and description of an asset.

Note: Properties in the body of the `PATCH` vary by asset type.

```
[PATCH]/v2/assets/Y1dTdvMjq9QtOM{
   "name":"My Awesome Video",
   "hosted_at":"http://www.mydomain.com",
   "description":"The next Internet meme."
}
```

Backlot returns a response similar to the following.

```
{
   "asset_type":"video",
   "duration":30,
   "name":"My Awesome Video",
   "preview_image_url":null,
   "created_at":"2011-07-22T18:54:19+00:00",
   "embed_code":"Y1dTdvMjq9QtOM",
   "time_restrictions":null,
   "updated_at":"2011-07-22T20:51:07+00:00",
   "external_id":null,
   "hosted_at":"http://www.mydomain.com",
   "original_file_name":"my_video.avi",
   "description":"The next Internet meme.",
   "status":"live"
}
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

The asset's information is successfully changed.

DELETING AN ASSET: DELETE

To permanently remove an asset, use `DELETE /v2/assets/asset_id`.

To edit YouTube video information:

Use a `DELETE` with the `/v2/assets` route and the asset ID.

The following example performs a get to obtain the current settings for the asset with the `BnMDNxMjqmPTu` ID.

```
[DELETE]/v2/assets/BnMDNxMjqmPTu
```

Backlot returns a 200 response without a response body.

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

The asset is successfully deleted.

COMMON TASKS WITH ALL ASSETS: PUT OR PATCH AND QUALIFIERS

Some functions are common to all assets, regardless of their type.

Unless stated otherwise at the beginning of a section, the API tasks detailed here are common to all asset types.

Setting an Asset's Post-processing Status

After an asset has been uploaded and transcoded, you might not want it to go live immediately, which is the default. By setting the asset's post-processing status, you can instruct the system to not immediately publish the asset. There are only two statuses:

- `live` (default)
- `paused`

1. To set an asset's post-processing status to `paused`.

```
[PATCH]/v2/assets/BjMDVrOTrLeHriLAq43T5y{
   "status" : "paused"
}
```

Backlot returns a response similar to the following.

```
{
   "duration": "5038",
   "asset_type": "video",
   "name": "bleebleblobble.mov",
   "preview_image_url": "http://ak.c.ooyala.com/w3ZHc0Njr33Tdp-
RRcwfZMjaOrmzOP82/Ut_HKthATH4eww8X4xMDoxOmdtO40mAx",
   "original_file_name": "bleebleblobble.mov",
   "embed_code": "w3ZHc0Njr33Tdp-RRcwfZMjaOrmzOP82",
   "hosted_at": "",
   "publishing_rule_id": "11a2b0702959459cb60db9cd5b7adb62",
   "description": "geeble gobble",
   "status": "paused",
   "external_id": "ABC124",
   "time_restrictions": null,
   "created_at": "2012-10-10T15:24:40Z",
   "updated_at": "2013-08-06T15:32:17Z",
   "ad_set_id": null
}
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

2. To set an asset's post-processing status to `live`.

```
[PATCH]/v2/assets/BjMDVrOTrLeHriLAq43T5y{
   "status" : "live"
}
```


Backlot returns a response similar to the following.

```
{
   "time_restrictions": null,
   "original_file_name": "bleebleblobble.mov",
   "created_at": "2012-10-10T15:24:40Z",
   "external_id": "ABC124",
   "name": "bleebleblobble.mov",
   "preview_image_url": "http://ak.c.ooyala.com/w3ZHc0Njr33Tdp-
RRcwfZMjaOrmzOP82/Ut_HKthATH4eww8X4xMDoxOmdtO40mAx",
   "ad_set_id": null,
   "asset_type": "video",
   "duration": "5038",
   "status": "live",
   "updated_at": "2013-08-06T15:32:53Z",
   "hosted_at": "",
   "description": "geeble gobble",
   "embed_code": "w3ZHc0Njr33Tdp-RRcwfZMjaOrmzOP82",
   "publishing_rule_id": "11a2b0702959459cb60db9cd5b7adb62"
}
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

Working with Labels

After you associate a video, channel, remote asset, or channel set with a label, Backlot automatically publishes it based on how you configured syndication.

Before you can associate them with assets, you must create the labels. See .

To work with labels for an asset:

1. To associate a label with an asset, use `POST` with the `/v2/assets` route, the asset ID, the `/v2/labels` qualifier, and a label ID. You can also associate multiple labels at the same time.

   The following example adds the Motorcycle Racing label to the `JxbzdkMjqBEsO` asset.

   ```
   [PUT]/v2/assets/JxbzdkMjqBEsO/labels/bace921fdea44cc18a5a273155514522
   ```

   Backlot returns a response similar to the following.

   ```
   {
      "items":[
         {
            "name":"Sports",
            "id":"814efb109416490a98ee3f4fcd6784cf",
            "full_name":"/Sports",
            "parent_id":null
         },
         {
            "name":"Motorcycle Racing",
            "id":"bace921fdea44cc18a5a273155514522",
            "full_name":"/Sports/Motorcycle Racing",
            "parent_id":"814efb109416490a98ee3f4fcd6784cf"
         }
      ]
   }
   ```


Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

2. View the current labels for an asset with `GET /v2/assets/asset_id/labels`.

   The following example gets the labels for the `JxbzdkMjqBEsO` asset.

```
[GET]/v2/assets/JxbzdkMjqBEsO/labels
```

   Backlot returns a response similar to the following.

```
{
   "items":[
      {
         "name":"Sports",
         "id":"814efb109416490a98ee3f4fcd6784cf",
         "full_name":"/Sports",
         "parent_id":null
      }
   ]
}
```

   Note:

   Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

3. To remove a single label from an asset, use `DELETE` with the `/v2/assets` route, the asset ID, the `/labels` qualifier, and a label ID. To disassociate all labels from the asset, specify `/labels` without a label ID.

   Note: To permanently delete a label and remove it from all assets, see *http://support.ooyala.com/documentation/users/tasks/label_delete.html*.

   The following example removes the Motorcycle Racing label from the `JxbzdkMjqBEsO` asset.

```
[DELETE]/v2/assets/JxbzdkMjqBEsO/labels/814efb109416490a98ee3f4fcd6784cf
```

   Backlot returns a 200 response.

   Note:

   Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

Applying Ad Sets

The easiest way to manage advertising when you have multiple assets with advertisements that change is to use ad sets.

Note: Before you can associate them with assets, you must create the ad sets. See *.Working with Your Own Ad Assets, Ad Sets, and Ad Sources* on page 139.

To associate an ad set with an asset, use `PATCH` with the `/v2/assets` route, the `/ad_sets` qualifier, and an ad set ID.

Use the `/v2/assets` route to associate an ad set.

The following example associates the `cab127d23bf45c38` ad set with the `JxbzdkMjqB` asset:

```
[PATCH]/v2/assets/JxbzdkMjqB/ad_set/cab127d23bf45c38
```


Backlot returns a 200 response.

Assigning Publishing Rules to Assets

After you add an asset to Backlot, you can assign publishing rules that affect when and where the content can be played.

Note: When you first create a remote asset with POST, do not include the `external_id` or `time_retsrictions` properties in the initial POST. The remote asset must first be created so that the embed code (content ID or asset ID) exists in the Ooyala system. After you have received the returned embed code for the asset, then you can use PATCH with the embed code to add `external_id` or PUT with `time_retsrictions`. See an example in *Adding an External ID to an Existing Asset* on page 33.

For information about setting up publishing rules, see *Publishing Rules* on page 163.

To work with a publishing rules and assets:

1. Assign a rule using a `PUT` with the `/v2/assets` route, the asset ID, and the `/publishing_rules` qualifier.

   The following example adds the `79feefbe24d9424786` publishing rule to the `Y1dTdvMjq9QtOM` asset:

   ```
   [PUT]/v2/assets/Y1dTdvMjq9QtOM/publishing_rule/79feefbe24d9424786
   ```

   Backlot returns a 200 response.

   Note:

   Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

2. To verify that the publishing rule was added, use `GET` with the `/v2/assets` route, the asset ID, and the `/publishing_rules` qualifier.

   The following example gets the publishing rule for the asset with the `Y1dTdvMjq9QtOM` ID:

   ```
   [GET]/v2/assets/Y1dTdvMjq9QtOM/publishing_rule
   ```

   Backlot returns the details of the publishing rule. In this example, the rule allows playback on desktops/notebooks, at any time, from any domain, in the US and Great Britain:

   ```
   {
      "name":"My Publishing Rule",
      "allowed_devices":[
         "desktop"
      ],
      "time_restrictions":{
         "start_date":"2007-04-05T00:00:00+00:00",
         "type":"range",
         "end_date":null
      },
      "domain_restrictions":{
         "domains":[

         ],
         "type":"blacklist"
      },
      "geographic_restrictions":{
         "type":"whitelist",
         "locations":[
            "GB",
            "US"
   ```

```
        ]
    },
    "id":"79feefbe24d9424786"
}
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

3. To remove a publishing rule from an asset, use `DELETE` with the `/v2/assets` route, the asset ID, the `/publishing_rule` qualifier, and a label ID.

   Note: To permanently delete a publishing rule and remove it from all assets, see *http://support.ooyala.com/documentation/users/label_delete.html*.

   The following example deletes the `79feefbe24d9424786` publishing rule from the `JxbzdkMjqBEsO` asset.

```
[DELETE]/v2/assets/JxbzdkMjqBEsO/labels/
```

   Backlot returns a 200 response.

   Note:

   Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

Replacing an Asset

Content replacement enables you to replace old content with new content. For example, you might have a video with technical issues or you might need to change your logo on multiple videos.

If the old asset and new asset are nearly identical, replacing content can save you time over deleting the old asset and uploading its replacement—you do not need to update labels, syndication settings, publishing rules, and so on. However, replacing content can affect analytics. For example, if you replace a 5 minute video with a 15 minute video, the average time watched becomes less meaningful and your engagement report will be inaccurate. Content replacement does not create a new movie profile. Instead, the system transcodes a new video file and links it to already existing asset the with same embed code.

Note:

Content replacement is intended primarily for use with video assets. If you want to replace content for remote assets or live streams, simply change their settings.

If you are using the default preview images, they are automatically replaced. If you are using a custom preview image (uploaded or via remote URL), it continues to be associated with the new asset.

To replace content:

1. Use POST with the `/v2/assets` route, the asset ID, the `/replacement` qualifier, with properties in the body of the request.

   The following example replaces the content of the the `Y1dTdvMjq9QtOMGrP-H59OIgiZ6-_Mrl` asset.

   Note: To upload the entire file at once, do not specify a `chunk_size`.

```
[POST]/v2/assets/Y1dTdvMjq9QtOMGrP-H59OIgiZ6-_Mrl/replacement{
   "file_size": 199895,
   "chunk_size": 100000
}
```


Backlot returns a response similar to the following.

```
{
   "asset_type": "video",
   "duration": 0,
   "name": "My Video",
   "preview_image_url": null,
   "created_at": "2011-07-22T18:54:19+00:00",
   "embed_code": "Y1dTdvMjq9QtOMGrP-H59OIgiZ6-_Mrl",
   "time_restrictions": null,
   "updated_at": "2011-07-22T18:54:19+00:00",
   "external_id": null,
   "description": null,
   "status": "uploading"
}
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

2. Retrieve the URLs for uploading your file with GET to the `/v2/assets` route, the asset ID, the `/replacement` qualifier, and the `/uploading_urls` qualifier.

```
[GET]/v2/assets/Y1dTdvMjq9QtOMGrP-H59OIgiZ6-_Mrl/replacement/
uploading_urls
```

The URLs are returned in the response body.

3. Upload each chunk.

Note: If you didn't specify a `chunk_size`, do not specify a chunk range.

The following example uploads the first chunk of the `Y1dTdvMjq9QtOMGrP-H59OIgiZ6-_Mrl` asset.

```
[PUT] https://uploader-v2.ooyala.com/send?filename=Y1dTdvMjq9QtOMGrP-
H59OIgiZ6-_Mrl/0000000000000-0000000099999
&filesize=100000
&expires=1311447448
&signature=tUp+iEUq9O9oP6khuasvZVFecoECcoej0gycG/ZTZr0
<file>
```

4. Mark the `status` of the replacement asset as `uploaded` with the `/replacement/upload_status` qualifiers.

```
[PUT]/v2/assets/Y1dTdvMjq9QtOMGrP-H59OIgiZ6-_Mrl/replacement/
upload_status{
   "status": "uploaded"
}
```

The asset is marked as uploaded. Once processed, it replaces the current asset and becomes live.

Searching for Assets

You can search for assets by specifying search keywords with various characteristics, including labels and metadata, as query string parameters.

The `/v2/assets` route has general querying capabilities, with the query string parameters `where`, `include`, `includes` and `and` (labels only), and more.

- The query string parameter for sorting, `orderby`, has an optional argument: `ascending` or `descending`.

- With the `limit` query string parameter, you can restrict the results to only a desired number of entries.
- With the `and` operator you can search for intersections of labels; that is, for assets that have two or more different labels.

1. To find assets that have a metadata field named `speed` with a value of `bursty`:

```
[GET] /v2/assets?where=metadata.speed='bursty'&include=metadata
```

2. To find assets that have the label `Sports`:

```
[GET] /v2/assets?where=labels INCLUDES 'Sports'
```

3. To find assets that have both the label `Sports` and the label `Baseball`:

```
[GET] /v2/assets?where=labels INCLUDES 'Sports' AND labels INCLUDES
 'Baseball'
```

4. To find assets updated after a certain date/time:

```
[GET] /v2/assets?where=updated_at>'2011-06-04T01:22:50Z'
```

5. To sort with `orderby` and limit the results with `limit`:

```
[GET] /v2/assets?where=description='Under the sea.' AND duration >
 600&orderby=created_at descending&limit=5
```

Managing Custom Metadata

Custom metadata enables you to add useful information to your assets. You can associate custom metadata with any type of asset.

You can use custom metadata for any purpose. Some common uses include:

- Organizing and categorizing content
- Storing information about videos to display on pages on which they are embedded
- Specifying information about how to display videos, such as CSS data
- Processing data for workflow integration

You can add metadata to assets through the Backlot UI or Backlot API. Once added, you can consume metadata through the Backlot UI or Backlot API.

The limit is 100 name/value pairs per asset.

The name is limited to 128 characters. There are other restrictions on allowed characters in the name; see *http://support.ooyala.com/developers/documentation/api/asset_metadata.html*

The value is limited to 2,048 characters.

Adding Custom Metadata

You can add up to 100 name/value pairs to an asset.

The name is limited to 128 characters, and the value to 2,048 characters.

To add custom metadata to an asset:

1. To add metadata without replacing the current metadata, use PATCH with the `/v2/assets` route.

   The following example adds the director and copyright custom metadata to the `JxbzdkMjqBEsO` asset.

   ```
   [PATCH]/v2/assets/JxbzdkMjqBEsO/metadata{
      "director":"Peter Jackson",
      "copyright":"2002"
   }
   ```

   Backlot returns a response similar to the following.

   ```
   {
      "director":"Peter Jackson",
      "copyright":"2002",
      "category":"Adventure",
      "custom_id":"54678934"
   }
   ```

   Note:

   Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

2. To add metadata that entirely replaces the current metadata, use PUT with the `/v2/assets` route.

   The following example replaces custom metadata for the `JxbzdkMjqBEsO` asset.

   ```
   [PUT]/v2/assets/JxbzdkMjqBEsO/metadata{
      "director":"Not Really Peter Jackson",
   }
   ```

   Backlot returns a response similar to the following.

   ```
   {
      "director":"Not Really Peter Jackson",
   }
   ```

   Note:

   Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

3. To get the current custom metadata, you can perform a GET at any time.

Deleting Custom Metadata

You can delete metadata entries at any time.

To delete custom metadata:

To delete an asset's metadata, use `DELETE` with the `/v2/assets` route, the asset ID, and the `/metadata` qualifier.

The following example deletes custom metadata from the `JxbzdkMjqBEsO` asset.

```
[DELETE]/v2/assets/JxbzdkMjqBEsO/metadata
```

Backlot returns a 200 response.

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

The metadata is deleted.

Adding Closed Captions

Closed captions are additional text shown within a video during playback. The captions typically contain an audio transcript of the video.

Closed captions improve the viewing experience for hearing impaired viewers as well as viewers who are not fluent in the spoken language of the video. Additionally, adding closed captions has been shown to significantly increase the viewership of online content.

Beyond driving additional viewership, many government and educational sites are required to have closed captions available for their video content to comply with government regulations.

Closed captions are added to asset by uploading a file in the Distribution Format Exchange Profile (DFXP [now TTML]) closed caption format.

Note: Although you can technically add closed captions to any type of asset, they make sense primarily with videos, remote assets, and YouTube videos.

Uploading and Viewing a Closed Captions File

After preparing a DFXP (now TTML) closed captions file, you can upload it to Backlot.

Note: If you need to edit an already uploaded closed captions file, change it on your computer and then upload the updated file.

To upload a closed captions file:

1. Create a DFXP (now TTML) closed captions file.
2. To add closed captions, use PUT with the `/v2/assets` route, the asset ID, and the `/closed_captions` qualifier.

   The following example adds closed captions to the `JxbzdkMjqB` asset:

   ```
   [PUT]/v2/assets/JxbzdkMjqB/closed_captions
      <DFXP file contents>
   ```

   Backlot returns a 200 response.

   Note:

   Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.
3. To view the contents of the closed captions file, use `GET` with the `/v2/assets` route, the asset ID, and the `/closed_captions` qualifier.

   The following example gets closed captions for the `JxbzdkMjqB` asset:

   ```
   [GET]/v2/assets/JxbzdkMjqB/closed_captions
   ```

   You are redirected to the file through a HTTP 302 redirect. The location header contains the redirect URL.

   Note:

   Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

Deleting a Closed Captions File

You can delete a closed captions file at any time.

To delete a closed captions file:

Use DELETE, the `/v2/assets` route., the asset ID, and the `/closed_captions` qualifier.

The following example deletes closed captions from the `JxbzdkMjqB` asset:

```
[DELETE]/v2/assets/JxbzdkMjqB/closed_captions
```

Backlot returns a 200 response.

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

Using External Identifiers

External identifiers enable you to assign custom identifiers to your assets so they are easier to organize, update, and modify. They are sometimes used to correlate with your own internal identifiers.

An external ID can contain any characters except commas (",") or slashes ("/").

The same external ID cannot be assigned to two different assets.

When making a change to an asset, you can use either its content ID (asset ID or embed_code) or external ID.

Creating an Asset with an External ID

Creating an asset with an external ID is a two part process: first create the asset with POST; then add the external_id property with PATCH.

Note: When you first create a remote asset with POST, do not include the `external_id` or `time_retsrictions` properties in the initial POST. The remote asset must first be created so that the embed code (content ID or asset ID) exists in the Ooyala system. After you have received the returned embed code for the asset, then you can use PATCH with the embed code to add `external_id` or PUT with `time_retsrictions`. See an example in *Adding an External ID to an Existing Asset* on page 33.

1. Use POST with the `/v2/assets` route and be sure *not* to include the `external_id` property in the request body.

   The following example creates the "My Video" video

   ```
   [POST]/v2/assets{
      "name":"My Video",
      "file_name":"my_video.avi",
      "asset_type":"video",
      "file_size":199895,
      "chunk_size":100000
   }
   ```

   Backlot returns a response similar to the following.

   ```
   {
      "asset_type":"video",
      "duration":0,
      "name":"My Video",
      "preview_image_url":null,
      "created_at":"2011-07-22T18:54:19+00:00",
      "embed_code":"Y1dTdvMjq9QtOMGrP",
      "time_restrictions":null,
      "updated_at":"2011-07-22T18:54:19+00:00",
      "description":null,
      "status":"uploading"
   }
   ```

   Now use PATCH to update the asset with the `external_id` property:

Note: Do not create a null `external_id`, that is, an `external_id` with no value (`""`) or a value of `"null"`. Such null external IDs cannot be searched for later.

```
[PATCH]/v2/assets/Y1dTdvMjq9QtOMGrP{
   "external_id":"my_video",
}
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

2. Follow the procedure for the asset type to add the asset to Backlot.

Adding an External ID to an Existing Asset

You can add an external ID to an asset at any time.

To add an external ID to an existing asset:

1. Use `PATCH` with the `/v2/assets` route, the asset ID, and the `external_id` property in the request body.

   The following example adds the `my_video` external ID to the asset with the `Y1dTdvMjq9QtOMGrP` content ID.

   ```
   [PATCH]/v2/assets/Y1dTdvMjq9QtOMGrP{
      "external_id":"my_video"
   }
   ```

   Backlot returns a response similar to the following.

   ```
   {
      "asset_type":"video",
      "duration":0,
      "name":"My Video",
      "preview_image_url":null,
      "created_at":"2011-07-22T18:54:19+00:00",
      "embed_code":"Y1dTdvMjq9QtOMGrP",
      "time_restrictions":null,
      "updated_at":"2011-07-22T18:54:19+00:00",
      "external_id":"my_video",
      "description":null,
      "status":"uploading"
   }
   ```

   Note:

   Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

2. You can now modify the asset using the external ID.

   The following example changes the external ID to `my_cool_video` using the current external ID.

   ```
   [PATCH]/v2/assets/my_video{
      "external_id":"my_cool_video"
   }
   ```

   Backlot returns a response similar to the following.

   ```
   {
      "asset_type":"video",
      "duration":0,
   ```


```
    "name":"My Video",
    "preview_image_url":null,
    "created_at":"2011-07-22T18:54:19+00:00",
    "embed_code":"Y1dTdvMjq9QtOMGrP",
    "time_restrictions":null,
    "updated_at":"2011-07-22T18:54:19+00:00",
    "external_id":"my_cool_video",
    "description":null,
    "status":"uploading"
}
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

VIDEO

When you upload a video to Ooyala, Ooyala automatically transcodes it using the settings specified for your account and makes the videos available for playback through your player.

We recommend that you upload videos to Ooyala in H.264, but we also support many other common codecs.

Quality of and Recommendations on Source Material (Video and Audio)

You should always start with the best quality source material available.

For best results, Ooyala recommends the following.

Item	Recommendation
Video Codecs	• **Highly recommended and preferred**: H.264 • **Other supported formats**: MPEG-4, MPEG-1, MPEG-2, H.263, Motion-JPEG, VP-6, Windows Media Video 7 (WMV-7), Windows Media Video 8 (WMV-8), Windows Media Video 9 (WMV-9), Windows Media Video 9 Advanced Profile (WVC1, VC-1), Microsoft Mpeg4, DV Video, Apple ProRes 422, XDCAM SD, XDCAM HD, XDCAM EX, XDCAM HD422, Raw Video (uncompressed YCbCr), Microsoft Screen Scraper 2, CinePak, Motion JPEG, Quicktime Run Length Encoding.
Audio Codecs	• **Highly recommended and preferred**: AAC • **Other supported formats**: AC-3, MPEG-2 Audio, MP3, PCM, Windows Media Audio (WMAV1 and WMAV2), WMA Pro, WMA Lossless, WMA Voice, Vorbis, AD-PCM, AMR.
Muxed/Container Formats	MP4, Mpeg, MOV, MPEG-2 Transport Stream, OGG, FLV, AVI, ASF, Material Exchange Format (MXF), General Exchange Format (GXF), Matroska.
Frame Rate	The same rate at which you shot the video. We recommend shooting or encoding at 24 (23.98), 25,

Item	Recommendation
	or 30 (29.97) fps. If the frame rate is greater than 30 fps, reduce it to 30 fps or lower.
Bit Rate	Double the highest bit rate that you intend to display to viewers (e.g., use a 4,000 kbps (kilobits per second) source video if your highest bit rate is 2,000 kbps). If your highest resolution is 1080p, we recommend 15,000 kbps. If your highest resolution is 720p, we recommend 7,500 kbps.
Interlacing	If your video is interlaced, deinterlace it before uploading to avoid jagged lines during playback. If the video was recorded as progressive, upload it as is.
Keyframes	Set to automatic.
Pixels	Encode using isomorphic (square) pixels (1 x 1). Although, this is usually the default video software setting, adjusting to 16:9 might unknowingly change the encoding to (rectangular) anamorphic pixels.
Resolution	If you are delivering to televisions, upload at 1080p. For delivery to a large variety of devices, upload at 720p. If you are targeting computers only, upload at the target embed size. **Note:** If your source file is in the MPEG-2 format, make sure the height is an increment of 16 (e.g., 480, 720, and so on).

Uploading a Video or Audio Asset: Four Steps

Only certain assets require uploading: video and audio files.

You can upload a file all at once in a single request or in chunks with multiple requests.

Note: By default, after processing, assets are live (publicly viewable). If you do not want the asset to be immediately published after transcoding, set its `status`. See *Setting an Asset's Post-processing Status* on page 23.

Uploading a source file (video or audio) is a four-step process:

1. Create the asset as detailed in *Creating an Asset: POST* on page 20.

   This returns the asset's identifier (called the `embed_code`)
2. Retrieve the "uploading URLs" for the asset with `GET /v2/assets/asset_id/uploading_urls`
3. For each file chunk, `PUT` to the URLs returned in Step 2. Do not sign these requests; see the explanation below.
4. Mark the asset's status as `uploaded` with `PUT /v2/assets/asset_id/upload_status`.

To upload a video or audio file:

1. Use `POST` with the `/v2/assets` route and properties in the request body.

   The following example creates the "My Video" video.

   Note: To upload the entire file at once, do not specify a `chunk_size`.

```
[POST]/v2/assets{
   "name": "My Video",
```


```
    "file_name": "my_video.avi",
    "asset_type": "video",
    "file_size": 199895,
    "chunk_size": 100000
}
```

Backlot returns a response similar to the following.

```
{
    "asset_type": "video",
    "duration": 0,
    "name": "My Video",
    "preview_image_url": null,
    "created_at": "2011-07-22T18:54:19+00:00",
    "embed_code": "Y1dTdvMjq9QtOMGrP-H59OIgiZ6-_Mrl",
    "time_restrictions": null,
    "updated_at": "2011-07-22T18:54:19+00:00",
    "external_id": null,
    "description": null,
    "status": "uploading"
}
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

2. Get the uploading URLs with `GET /v2/assets`, the asset ID, and the `/uploading_urls` qualifier.

   Note: If you are replacing an asset with the `/replacement` qualifier, get the uploading URLs with `[GET] /v2/assets/`*`asset_id`*`/replacement/uploading_urls`. See *Replacing an Asset* on page 27.

   The following example gets the uploading URLs for the asset with the ID `Y1dTdvMjq9QtOMGrP-H59OIgiZ6-_Mrl`. This ID was returned as the `embed_code` of the previous response.

   ```
   [GET]/v2/assets/Y1dTdvMjq9QtOMGrP-H59OIgiZ6-_Mrl/uploading_urls
   ```

   Backlot returns a response similar to the following.

   ```
   ["https://uploader-v2.ooyala.com/send?filename=Y1dTdvMjq9QtOMGrP-
   H59OIgiZ6-_Mrl/0000000000000-0000000099999&filesize=100000
    &expires=1311447448 &signature=tUp+iEUq9O9oP6khuasvZVFecoECcoej0gycG/
   ZTZr0", "https://uploader-v2.ooyala.com/send?filename=Y1dTdvMjq9QtOMGrP-
   H59OIgiZ6-_Mrl/0000000100000-000000019989 &filesize=99895
    &expires=1311447448 &signature=1aJ3ojLTUDnfLiHxZt/1us7jZ0StbtMC
   +pOnqNSQtiw"]
   ```

   Note:

   Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

3. Upload each chunk.

   If you didn't specify a `chunk_size`, do not specify a chunk range. The following example uploads the first chunk of the `Y1dTdvMjq9QtOMGrP-H59OIgiZ6-_Mrl` asset.

   Note: ***Do not sign this request***, as you would other requests. Simply PUT the chunk contents to the exact URLs. These URLs have already been signed for you.

   ```
   [PUT]https://uploader-v2.ooyala.com/send?filename=Y1dTdvMjq9QtOMGrP-
   H59OIgiZ6-_Mrl/0000000000000-0000000099999 &filesize=100000
   ```


```
 &expires=1311447448 &signature=tUp+iEUq9O9oP6khuasvZVFecoECcoej0gycG/
ZTZr0 <file>
```

4. To indicate that the video is ready for processing, use `PUT /v2/assets`, the asset ID, the `/upload_status` qualifier, and the `status` property in the request body as `uploaded`.

   The following example marks the `Y1dTdvMjq9QtOMGrP-H59OIgiZ6-_Mrl` asset as uploaded.

```
[PUT]/v2/assets/Y1dTdvMjq9QtOMGrP-H59OIgiZ6-_Mrl/upload_status{
   "status": "uploaded"
}
```

   Backlot returns a response similar to the following and starts processing the video.

```
{
   "status": "uploaded"
}
```

   Note:

   Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

The video is successfully uploaded.

About Auto-generated Preview Images (Thumbnails)

During transcoding of a video, Backlot generates thumbnails of the video at 15 second intervals.

When a video is uploaded to Ooyala, Backlot produces a series of thumbnails (also known as a "preview image") for each encoding, regardless of type of encoding.

By default, a thumbnail is taken at 15-second intervals throughout the length of the video. If you want to change this interval, contact Ooyala Technical Support.

Thumbnails are sized at 1/3 of the encoding width.

However, the "preview image" is a global setting per video. Only one of the generated thumbnails (or one of the custom uploaded preview images) can be set as the one that is displayed in the player preview. By default, this is the first thumbnail from the highest-resolution encoding.

Managing the Preview Image

After you add an asset to Backlot, you can specify the image to display before viewers click the play button. The image can be in the GIF, PNG, or JPG format.

To select your desired thumbnail or preview images, you have the following choices:

- Use one of the auto-generated images. See the discussion *About Auto-generated Preview Images (Thumbnails)* on page 37.
- Upload your own custom image. Backlot only supports uploading one preview image. Subsequent uploads simply replace the previous upload.
- Indicate a URL where the desired image is.

To manage the preview image:

1. To view the preview images that are automatically generated through transcoding, use GET with the `/v2/assets` route, the asset ID, and the `/generated_preview_images` qualifier.

   The following example returns the preview automatically generated images for asset `JxbzdkMjqB`:

```
[GET]/v2/assets/JxbzdkMjqB/generated_preview_images
```


Backlot returns a response similar to the following:

```
[
   {
      "url":"http://ak.c.ooyala.com/JxbzdkMjqB/Ut_HKthATH4",
      "time":0
   },
   {
      "url":"http://ak.c.ooyala.com/JxbzdkMjqB/q78XH7vLqJJ",
      "time":4512
   }
]
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

2. To create a custom preview image, you can either upload a file or specify a URL where the image is.

- **Note:** A `POST` request adds the preview image file to the current set of files. A `PUT` request replaces the current preview image files.

  To upload a custom preview image, use `POST`, the `/v2/assets` route, the asset ID, and the image data in the request body.

  The following example adds a preview image to asset `Y1dTdvMjq9QtOM`:

  ```
  [POST]/v2/assets/Y1dTdvMjq9QtOM/preview_image_files
     <file_contents>
  ```

  Backlot returns a response similar to the following:

  ```
  {
     "width":800,
     "height":600,
     "url":"http://ak.c.ooyala.com/Y1dTdvMjq9QtOM/promo136839624",
     "id":"promo136839624"
  }
  ```

  Repeat this step for each size of the image to add.

  Note:

  Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

- To indicate a custom preview image by URL, use `PUT`, the `/v2/assets` route, the asset ID, and the size and location details in the request body.

  The following example adds multiple sizes of a preview image to asset `Y1dTdvMjq9QtOM`:

  ```
  [POST]/v2/assets/Y1dTdvMjq9QtOM/preview_image_urls[
     {
        "width":1920,
        "height":1080,
        "url":"http://www.mysite.com/my_big_image.png"
     },
     {
        "width":1280,
        "height":720,
        "url":"http://www.mysite.com/my_medium_image.png"
     },
     {
  ```


```
        "width":640,
        "height":480,
        "url":"http://www.mysite.com/my_small_image.png"
    }
]
```

Backlot returns a response similar to the following:

```
[
    {
        "url":"http://www.mysite.com/my_big_image.png",
        "width":1920,
        "height":1080
    },
    {
        "url":"http://www.mysite.com/my_medium_image.png",
        "width":1280,
        "height":720
    },
    {
        "url":"http://www.mysite.com/my_small_image.png",
        "width":640,
        "height":480
    }
]
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

3. For the primary preview image an asset, you can use either the generated image or the uploaded/ custom one.

   - To choose one of the automatically generated images, in the request body for the `type` property, specify `generated` and the time. In the following example, the primary preview image is set to the image that was generated at the 15-second mark.

     ```
     [PUT]/v2/assets/Y1dTdvMjq9QtOM/primary_preview_image{
        "type":"generated",
        "time":15
     }
     ```

     Note:

     Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

   - To use uploaded files, in the request body for the `type` property, specify `uploaded_file`. In the following example, the primary preview image is set to use one of the uploaded images. At the time of embed, Ooyala automatically selects an image closest to the size of the embed.

     ```
     [PUT]/v2/assets/Y1dTdvMjq9QtOM/primary_preview_image{
        "type":"uploaded_file"
     }
     ```

     Note:

     Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

- To include information about the primary preview image in a get request, specify `include=primary_preview_image`. In the following example, the primary preview image is returned in the response.

```
[GET]/v2/assets/Y1dTdvMjq9QtOM?include=primary_preview_image
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

- To use files at remote URLs, in the request body for the `type` property, specify `remote_url`. In the following example, the primary preview image is set to use one of the remote URLs. At the time of embed, Ooyala automatically selects an image closest to the size of the embed.

```
[PUT]/v2/assets/Y1dTdvMjq9QtOM/primary_preview_image{
   "type":"remote_url"
}
```

Note:

For displaying remote-URL type preview images in the Flash-based Ooyala player, be sure to set up the Flash-required cross-domain security file `crossdomain.xml` to include an entry for the *.ooyala.com domain and any other domains (web sites) that need to access the image. Otherwise, Ooyala player cannot display the remotely hosted image.

The file is described at *Setting a crossdomain.xml file for HTTP streaming*.

The entry you need to add looks like the following. In addition, you need to include entries for all domains (web sites) that need to access the image.

```
.
.
.
<allow-access-from domain="*.ooyala.com" />
.
.
.
```

Viewing Video Stream Information

Based on your account settings, videos are transcoded using different resolutions, codecs, and bit rates.

The term *streams* in this context means the internal-to-Ooyala processes that are started by the system to transcode your videos.

Note: You cannot control or manipulate these streams; they are controlled by the system. You can only view their names, not change them.

To view streams for a video:

To view streams, use `GET /v2/assets`, the asset ID, and the `/streams` qualifier. ABR streams can be distinguished by the "stream_type" field ("stream_type":"abr").

The following example displays streams for the `JxbzdkMjqB` asset:

```
[GET]/v2/assets/JxbzdkMjqB/streams
```

Backlot returns a response similar to the following.

```
[
   {
```


```
      "profile":"baseline",
      "video_height":360,
      "is_source":false,
      "video_width":640,
      "file_size":17937644,
      "audio_codec":"aac",
      "video_codec":"h264",
      "average_video_bitrate":300
   },
   {
      "profile":"baseline",
      "video_height":360,
      "is_source":false,
      "video_width":640,
      "file_size":41716636,
      "audio_codec":"aac",
      "video_codec":"h264",
      "average_video_bitrate":900
   }
]
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

Viewing Source File Information

To see information about the orginal source file of an asset, use `GET /v2/assets/asset_id/source_file_info`.

To view the information about the source file for videos or audios:

Use the `GET /v2/assets`, the asset ID, and the `/source_file_info` qualifier.

The following example displays ABR streams for the `JxbzdkMjqB` asset:

```
[GET]/v2/assets/JxbzdkMjqB/source_file_info
```

Backlot returns a response similar to the following.

```
[
   {
      "profile":"baseline",
      "video_height":360,
      "is_source":false,
      "video_width":640,
      "file_size":17937644,
      "audio_codec":"aac",
      "video_codec":"h264",
      "average_video_bitrate":300
   },
   {
      "profile":"baseline",
      "video_height":360,
      "is_source":false,
      "video_width":640,
      "file_size":41716636,
      "audio_codec":"aac",
      "video_codec":"h264",
      "average_video_bitrate":900
   }
]
```


Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

LIVE STREAMS

Live streaming is the delivery of video content in real time.

Live streaming is popular for sporting and music events and is a great way to drive traffic to your site and increase awareness of your brand. Ooyala allows publishers to manage both their live and on demand video assets within the same content management system. This includes obtaining analytics on live streams, setting syndication controls for live streams, and customizing / branding a live player. In addition, we support adaptive bit rate (ABR) live streaming, the ability to monetize your live streams using 3rd party ad networks and servers, and the ability to monetize your live streams using paywalls.

Live streams consist of content creation and encoding, content delivery through a CDN, and content consumption through Internet-connected devices. Backlot manages the set up, publishing rules, and monetization of your live streams.

To set up a live stream with Ooyala, you only need a video camera, an Internet connection, and an encoder. Popular encoders include hardware encoders such as Inlet Spinnaker HD and software encoders such as Flash Media Live (free).

Note:

If live streaming is not enabled on your account, contact Sales, your Customer Success Manager, or Technical Support.

Live streaming does not support Real Time Streaming Protocol (RTSP).

For general encoding information that you can use to choose live stream encodings, see *http://support.ooyala.com/users/documentation/concepts/chapter_encoding.html*.

Adding a Live Stream

When you add a live stream, you receive URLs from Ooyala that you configure in your encoder.

For the highest quality streaming, specify the encoder IP at the location from which you are broadcasting and make sure that you have enough dedicated bandwidth to encode and serve all of your live streams.

Note: It can take up to 30 minutes for a newly provisioned live stream to be usable.

To add a live stream:

1. Use the `/v2/assets` route. In the POST body, set asset_type to live_stream, is_flash to true, and specify the encodings.

   Note: For live streaming to iPhone and iPad (iOS devices), also set is_ios to true.

   The following example creates a live stream with three Flash encodings and iOS live support:

```
[POST]/v2/assets{
   "name":"Flash and iOS Live Stream",
   "asset_type":"live_stream",
   "primary_encoder_ip":"8.8.8.8",
   "password":"encoder_password",
   "is_flash":true,
   "is_ios":true,
   "encodings":[
      {
         "width":800,
```


```
            "height":600,
            "bitrate":600
         },
         {
            "width":800,
            "height":600,
            "bitrate":300
         },
         {
            "width":400,
            "height":300,
            "bitrate":200
         }
      ]
}
```

Backlot returns a response similar to the following.

```
{
   "name":"Flash and iOS Live Stream",
   "preview_image_url":null,
   "asset_type":"live_stream",
   "duration":0,
   "embed_code":"FlczhuMzocRFR3YwMD0bxxo7wOtQY8-w",
   "created_at":"2012-03-02T00:42:13+00:00",
   "encodings":[
      {
         "bitrate":200,
         "height":300,
         "width":400
      },
      {
         "bitrate":300,
         "height":600,
         "width":800
      },
      {
         "bitrate":600,
         "height":600,
         "width":800
      }
   ],
   "is_ios":true,
   "time_restrictions":null,
   "updated_at":"2012-03-02T00:42:15+00:00",
   "hosted_at":null,
   "external_id":null,
   "original_file_name":null,
   "is_flash":true,
   "ios_live_upload_url":"http://post.ooyala-
i.akamaihd.net/201184/1dayarchive/ VpNDqXTEbTE_5UgsKdpQcDjcJATg/
FlczhuMzocRFR3YwMD0bxxo7wOtQY8-w/master.m3u8",
   "ios_live_delivery_url":"http://ooyala-i.akamaihd.net/
hls/live/201184/1dayarchive/ VpNDqXTEbTE_5UgsKdpQcDjcJATg/
FlczhuMzocRFR3YwMD0bxxo7wOtQY8-w/master.m3u8",
   "backup_entry_point":"rtmp://b.ep54802.i.akamaientrypoint.net/
EntryPoint",
   "primary_entry_point":"rtmp://p.ep54802.i.akamaientrypoint.net/
EntryPoint",
   "description":null,
   "status":"live"
}
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

2. For setup in your encoder, for Flash, note the `primary_entry_point` and `backup_entry_point`.
3. For setup in your encoder, for iOS, note the `ios_live_upload_url` and `ios_live_delivery_url`.

   The values of these properties have the following formats. The `pcode` is your provider code or account identifier (for details, see *Your API Credentials* on page 9). The `embedCode` is the embed code or content ID for the live stream. Also shown are the corresponding field names on the Spinnaker.

   Table 3:

Property	On Spinnaker	Example or Format
`primary_entry_point`	Server Location URL	`rtmp://p.ep54802.i.akamaientrypoint.net/EntryPoint`
`backup_entry_point`		`rtmp://b.ep54802.i.akamaientrypoint.net/EntryPoint`
`ios_live_upload_url`	Publishing URL	`http://post.ooyala-i.akamaihd.net/201184/1dayarchive/pcode/embedCode/master.m3u8`
`ios_live_delivery_url`	Delivery URL	`http://ooyala-i.akamaihd.net/hls/live/201184/1dayarchive/pcode/embedCode/master.m3u8`

The live stream is successfully added.

Setting Up an Encoder

Setting up an encoder involves configuring it with the Backlot-provided URLs when you set up the live stream.

Before you can configure the encoder, you must have already create the video asset on Backlot or with the Backlot API as detailed in *Adding a Live Stream* on page 42.

To set up an encoder:

1. Open the encoder's configuration interface.
2. If you will be generating Flash streams, navigate to the page where you configure Flash streams and enter the primary and secondary URLs.

   On the Spinnaker 7100, do the following:

   a) Click the **H264** tab.
   b) Click the **MP4 Output** subtab.
   c) Enter `master` in the **Variant playlist filename(s)** field.
   d) In the **Streaming to Primary Flash Media Server** section, enter the value of the `primary_entry_point` property received in the response with the live stream was created in the **Server Location URL** field. (See *Adding a Live Stream* on page 42 for details.)
   e) In the **Streaming to Secondary Flash Media Server** section, enter the value of the `backup_entry_point` property received in the response with the live stream was created in the **Server Location URL** field. (See *Adding a Live Stream* on page 42 for details.)
   f) Click **Apply**.

3. If you will generate iOS streams, navigate to the page where you configure Flash streams and enter the upload and delivery URLs.

   On the Spinnaker 7100, do the following:

   a) Click the **H264** tab.
   b) Click the **iPhone Output** subtab.
   c) Enter `master` in the **Variant playlist filename(s)** field.
   d) Enter the value of the `ios_live_upload_url` property received in the response with the live stream was created in the **Storage URL** field, removing "/master.m3u8" from the end of the URL. (See *Adding a Live Stream* on page 42 for details.)
   e) Enter the value of the `ios_live_delivery_url` property received in the response with the live stream was created in the **Publishing URL** field, removing "/master.m3u8" from the end of the URL. (See *Adding a Live Stream* on page 42 for details.)
   f) Click **Apply**.

REMOTE ASSETS

A remote asset is a piece of content that you add to Backlot without uploading, processing, or storing the content with Ooyala.

Remote asset management provides all of the benefits of Backlot, including management, syndication, monetization, analytics, and access to the Ooyala APIs.

Ooyala supports the following formats for remote assets:

- flv (extra meta-data fields must be added in Backlot)
- mp4, m4a, mov, 3gp, 3g2, f4v
- SMIL via HTTP (single stream)
- SMIL via RTMP (multiple streams)
- HLS

Note: For iPad and iPhone streams, only MP4 or M3U8 formats are valid.

Remote assets can be added to Backlot using the Backlot UI or through the APIs. After a piece of content is added to Backlot, Ooyala stores basic information about it including the duration and title. Rules, customizations, preview images, and additional metadata can be applied to remote assets through the Backlot UI or APIs.

Once uploaded, remote assets can be delivered by Ooyala or a CDN of your choice.

Note: After a remote asset has been created, its propagation to the various CDNs might be delayed 60 seconds or more. If you request a remote asset too soon after its creation, the results will be cached by the CDNs, which might take several minutes to clear. Best practice: after creation, wait 30 or 60 seconds, query with the Backlot API `[GET] /v2/assets/asset_id` route, and after retrieving the remote asset's embed code (content ID or asset ID), then proceed to embed the asset.

Adding a Remote Asset

Before you can manage a Remote Asset, you must add it to Backlot.

To add a remote asset:

Use the `/v2/assets` route.

The following example creates the "My Remote Asset" remote asset.

```
[POST]/v2/assets{
   "name":"My Remote Asset",
   "asset_type":"remote_asset",
```


```
   "duration":120000,
   "stream_urls":{
      "flash":"http://mydomain.com/my_flash_file.flv",
      "iphone":"http://mydomain.com/iphone_compatible_file.mp4"
   }
}
```

Backlot returns a response similar to the following.

```
{
   "asset_type":"remote_asset",
   "duration":120000,
   "name":"My Remote Asset",
   "preview_image_url":null,
   "created_at":"2011-05-16T20:35:53+00:00",
   "embed_code":"Y1NWZnMjq-DPsM2",
   "stream_urls":{
      "flash":"http://mydomain.com/my_flash_file.flv",
      "iphone":"http://mydomain.com/iphone_compatible_file.mp4"
   },
   "time_restrictions":null,
   "updated_at":"2011-06-02T00:08:58+00:00",
   "external_id":"myExternalId",
   "hosted_at":null,
   "original_file_name":null,
   "description":null,
   "status":"live"
}
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

The remote asset is successfully added.

YOUTUBE VIDEOS

A YouTube video is a video hosted by YouTube and played through the Ooyala player.

This enables you to:

- Maintain a consistent look and feel across your site with the Ooyala player or your custom player.
- Leverage the use of Ooyala's rich APIs for an integrated and customized experience.
- Gain insight and maintain the benefits of Ooyala's leading analytics for all videos in your account.
- Discover supplemental content for your site from Youtube, the Internet's largest video library.
- Apply time, geographic, or domain-based rules to the content.
- Leverage Ooyala's intuitive and comprehensive tools to manage, syndicate, customize and analyze YouTube content along with your own content.

YouTube videos are sourced from YouTube, are not downloaded to your account, are not transcoded by Ooyala, and are not applied to your delivery or managed content allowances. As a result, all videos will contain a YouTube watermark and will be delivered using the best quality YouTube can provide based on the embed size of the player.

Because Backlot does not transcode YouTube videos, it does not generate any preview images and the content cannot be played back on iOS devices.

YouTube content might not always be available to the player. For example, videos might be removed, videos might be restricted based on YouTube publisher rules, or access to YouTube might be blocked by a viewer's company or institution.

Uploading a YouTube Video

Before you can manage a YouTube video, you must add it to Backlot.

Note: No two YouTube assets in your account can have the same YouTube ID.

If you attempt to add a YouTube video that has been previously added, you receive the following error message, where *asset_id* identifies the previously created asset:

```
"{"duplicate_youtube_id":["asset_id"]}"
```

To add a YouTube video:

Use POST with the `/v2/assets` route and `asset_type` of `youtube` and the `youtube_id` in the request body.

The following example adds the YouTube video with the `dQw4w9WgXcQ` YouTube ID.

```
[POST] /v2/assets{
   "name": "My Video",
   "asset_type": "youtube",
   "youtube_id": "dQw4w9WgXcQ"
}
```

Backlot returns a response similar to the following.

```
{
   "asset_type":"youtube",
   "duration":213,
   "name":"My YouTube Video",
   "preview_image_url":null,
   "created_at":"2011-08-10T00:04:05+00:00",
   "embed_code":"BnMDNxMjqmPTu",
   "time_restrictions":null,
   "updated_at":"2011-08-10T00:04:05+00:00",
   "youtube_id":"dQw4w9WgXcQ",
   "external_id":null,
   "hosted_at":null,
   "original_file_name":null,
   "description":"Music video by Rick Astley performing Never Gonna Give
 You Up. YouTube view counts pre-VEVO: 2,573,462 (C) 1987 PWL",
   "status":"live"
}
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

The video is successfully added.

Making Remote Assets Available on YouTube

To make a remote asset available on YouTube, you need to setup a YouTube syndication (only once), upload a video to YouTube, create a remote asset in Backlot, add the YouTube ID to the asset, and add a label to the asset that triggers the YouTube syndication.

To make a remote asset available on YouTube:

1. Setup a YouTube syndication (only once).

   The following example creates a YouTube syndication that uses the `syndicate_to_youtube` label (48951b2ed95a4e13b8a3b07d59ac6ec1).

```
[POST]/v2/syndications{
   "asset_types":[
      "remote_asset"
   ],
   "include_all_content":false,
   "include_labels":[
      "48951b2ed95a4e13b8a3b07d59ac6ec1"
   ],
   "name":"YouTube",
   "should_create_youtube_videos":true,
   "should_delete_youtube_videos":true,
   "type":"youtube",
   "username":"my_username",
   "password":"my_password"
}
```

   Backlot returns a response similar to the following.

```
{
   "name":"YouTube",
   "require_access_key":false,
   "created_at":"2011-03-31T22:38:34+00:00",
   "include_labels":[
      "48951b2ed95a4e13b8a3b07d59ac6ec1"
   ],
   "should_create_youtube_videos":true,
   "asset_types":[
      "remote_asset"
   ],
   "username":"my_username",
   "id":"716437dedfa443bbb69c7101cf3574c0",
   "include_all_content":false,
   "type":"youtube",
   "should_delete_youtube_videos":true
}
```

   Note:

   Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

2. Upload the video to YouTube and get the YouTube ID. For more information on uploading videos to YouTube, refer to the *YouTube Web Site*.

   In this example, the YouTube ID is `dQw4w9WgXcQ` for the following URL:

```
http://www.youtube.com/watch?v=dQw4w9WgXcQ
```

3. Create a remote asset in Backlot.

   The following example creates a remote asset that points to the original location of the remote asset (not the YouTube link).

```
[POST]/v2/assets{
   "name":"A new remote asset",
   "asset_type":"remote_asset",
   "duration":120000,
   "stream_urls":{
```


```
      "flash":"http://example.com/my_flash_file.flv",
      "iphone":"http://example.com/iphone_compatible_file.mp4"
   }
}
```

Backlot returns a response similar to the following.

```
{
   "asset_type":"remote_asset",
   "duration":120000,
   "name":"A new remote asset",
   "preview_image_url":null,
   "created_at":"2011-09-12T17:47:03+00:00",
   "embed_code":"c1cWxzMjo6AKb_NhHAQiXx9mvAY4mG-D",
   "stream_urls":{
      "ipad":null,
      "source_file":null,
      "iphone":"http://example.com/iphone_compatible_file.mp4",
      "flash":"http://example.com/my_flash_file.flv",
      "itunes":null
   },
   "time_restrictions":null,
   "updated_at":"2011-09-12T17:47:04+00:00",
   "external_id":null,
   "hosted_at":null,
   "original_file_name":null,
   "description":null,
   "status":"live"
}
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

4. Associate the YouTube ID with the asset.

   The following example associates the `dQw4w9WgXcQ` YouTube ID with the `c1cWxzMjo6AKb_NhHAQiXx9mvAY4mG-D` asset.

```
[PATCH]/v2/assets/c1cWxzMjo6AKb_NhHAQiXx9mvAY4mG-D/youtube{
   "youtube_id":"dQw4w9WgXcQ"
}
```

Backlot returns a response similar to the following.

```
{
   "author":"RickAstleyVEVO",
   "youtube_id":"dQw4w9WgXcQ",
   "youtube_syndication_settings":{
      "private":null,
      "mobile":null,
      "embeddable":null,
      "keywords":null
   }
}
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

5. Associate the label with the asset.

The following example associates the `syndicate_to_youtube` label (48951b2ed95a4e13b8a3b07d59ac6ec1) with the `c1cWxzMjo6AKb_NhHAQiXx9mvAY4mG-D` asset.

```
[PATCH] /v2/assets/c1cWxzMjo6AKb_NhHAQiXx9mvAY4mG-D/
labels/48951b2ed95a4e13b8a3b07d59ac6ec1
```

Backlot returns a response similar to the following.

```
{
   "items":[
      {
         "name":"syndicate_to_youtube",
         "id":"48951b2ed95a4e13b8a3b07d59ac6ec1",
         "full_name":"/syndicate_to_youtube",
         "parent_id":null
      }
   ]
}
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

The remote asset is available on YouTube.

CHANNELS AND CHANNEL SETS

Channels and channel sets let you group your content for delivery to your customers.

Note: For best performance, a maximum of 64 entries per channel or channel set is allowed. Channels or channels sets with more than 64 entries are truncated.

You cannot include audio content in channels, only video.

For information on how to create channels and channel sets, please see *Creating a Channel*and *Creating a Channel Set*.

Creating a Channel

Once a channel is created, you can add videos.

To create a channel:

Use `POST` and `/v2/assets` with the `asset_type` set to `channel` in the request body.

The following example creates the "My Channel" channel.

```
[POST] /v2/assets{
   "name": "My Channel",
   "asset_type": "channel"
}
```

Backlot returns a response similar to the following.

```
{
   "asset_type": "channel",
   "duration": 0,
   "name": "My Channel",
   "preview_image_url": null,
```


```
    "created_at": "2011-08-10T17:45:24+00:00",
    "embed_code": "dvNTVxMjrRktSlb5v",
    "time_restrictions": null,
    "updated_at": "2011-08-10T17:45:24+00:00",
    "external_id": null,
    "hosted_at": null,
    "original_file_name": null,
    "description": null,
    "status": "live"
}
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

The channel is successfully added.

Viewing and Editing Channel Information

You can change information about a channel at any time.

To view or edit channel information:

1. To view the channel's information, use the `GET /v2assets` route with the asset ID of the channel.

   The following example performs a get to obtain the current settings for channel `dvNTVxMjrRktSlb5v`.

   ```
   [GET]/v2/assets/dvNTVxMjrRktSlb5v
   ```

   Backlot returns a response similar to the following.

   ```
   {
       "asset_type": "channel",
       "duration": 0,
       "name": "My Channel",
       "preview_image_url": null,
       "created_at": "2011-08-10T17:45:24+00:00",
       "embed_code": "dvNTVxMjrRktSlb5v",
       "time_restrictions": null,
       "updated_at": "2011-08-10T17:45:24+00:00",
       "external_id": null,
       "hosted_at": null,
       "original_file_name": null,
       "description": null,
       "status": "live"
   }
   ```

   Note:

   Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

2. To change a channel's information, use `PATCH` with `/v2/assets`, the asset ID, and the updated properties in the request body.

   The following example changes the name, "hosted at" URL, and description.

   ```
   [PATCH]/v2/assets/dvNTVxMjrRktSlb5v{
       "name": "My Awesome Channel",
       "hosted_at": "http://www.mydomain.com",
       "description": "This channel is awesome, really."
   }
   ```


Backlot returns a response similar to the following.

```
{
   "asset_type": "channel",
   "duration": 0,
   "name": "My Awesome Channel",
   "preview_image_url": null,
   "created_at": "2011-08-10T17:45:24+00:00",
   "embed_code": "dvNTVxMjrRktSlb5v",
   "time_restrictions": null,
   "updated_at": "2011-08-10T17:48:25+00:00",
   "external_id": null,
   "hosted_at": "http://www.mydomain.com",
   "original_file_name": null,
   "description": "This channel is awesome, really.",
   "status": "live"
}
```

The channel is successfully edited.

Adding Videos to a Channel

After you create a channel, you can start adding videos.

You must have the same player assigned to all the videos on the channel, so the channel can play them as expected.

Note: You cannot add channels to a channel. To create collections of channels, use channel sets.

To add videos to a channel:

1. To view videos currently assigned to a channel, use `GET /v2/assets`, the asset ID, and the `/lineup` qualifier.

   The following example lists the videos for the `dvNTVxMjrRktSlb5v` channel.

   ```
   [GET]/v2/assets/dvNTVxMjrRktSlb5v/lineup
   ```

   Backlot returns a response similar to the following.

   ```
   [
      "Argh3lMjphu25tH",
      "RQlbdvMjq9QtGGG"
   ]
   ```

   Note:

   Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

2. To replace the current set of videos with a new set, use `PUT` with the `/v2/assets` route, the asset ID of the channel, the `/lineup` and the asset IDs of videos in the request body.

   The following example adds the `IzNnllMjphu2XF3` and `Y1dTdvMjq9QtOMG` video assets to the `dvNTVxMjrRktSlb5v` channel, replacing the current videos.

   ```
   [PUT]/v2/assets/dvNTVxMjrRktSlb5v/lineup[
      "IzNnllMjphu2XF3",
      "Y1dTdvMjq9QtOMG"
   ]
   ```


Backlot returns a response similar to the following.

```
[
   "IzNnllMjphu2XF3",
   "Y1dTdvMjq9QtOMG"
]
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

3. To append a single video to a channel without replacing the current set, use `PUT /v2/assets` with the asset ID of the channel, the `/lineup` qualifier, and the asset ID of the video to add.

   The following example adds the `BrdXVjMjrgtupU3M` video asset to the `dvNTVxMjrRktSlb5v` channel.

   ```
   [PUT]/v2/assets/dvNTVxMjrRktSlb5v/lineup/BrdXVjMjrgtupU3M
   ```

   Backlot returns a response similar to the following.

   ```
   [
      "BrdXVjMjrgtupU3M",
      "IzNnllMjphu2XF3",
      "Y1dTdvMjq9QtOMG"
   ]
   ```

   Try it

4. To remove a video from a channel, use `DELETE /v2/assets` with the asset ID of the channel, the `/lineup` qualifier, and the asset ID of the video to remove.

   The following example deletes the `BrdXVjMjrgtupU3M` video asset from the `dvNTVxMjrRktSlb5v` channel.

   ```
   [DELETE]/v2/assets/dvNTVxMjrRktSlb5v/lineup/BrdXVjMjrgtupU3M
   ```

   Backlot returns a response similar to the following.

   ```
   [
      "IzNnllMjphu2XF3",
      "Y1dTdvMjq9QtOMG"
   ]
   ```

   Note:

   Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

The videos within the channel are successfully updated.

Creating a Channel Set

Once a channel set is created, you can add channels.

To create a channel set:

Use `POST` and `/v2/assets` with the `asset_type` set to `channelset` in the request body.

The following example creates the "My Channel Set" channel set.

```
[POST]/v2/assets{
```


```
    "name": "My Channel Set",
    "asset_type": "channel_set"
}
```

Backlot returns a response similar to the following.

```
{
    "asset_type": "channel_set",
    "duration": 0,
    "name": "My Channel Set",
    "preview_image_url": null,
    "created_at": "2011-08-10T18:20:03+00:00",
    "embed_code": "IwYTVxMjqLtYHmwqy",
    "time_restrictions": null,
    "updated_at": "2011-08-10T18:20:04+00:00",
    "external_id": null,
    "hosted_at": null,
    "original_file_name": null,
    "description": null,
    "status": "live"
}
```

The channel set is successfully created.

Editing a Channel Set

You can change information about a channel set at any time.

To edit channel set information:

1. To view the channel's information, use the `GET /v2assets` route with the asset ID of the channel set. The following example performs a get to obtain the current settings for channel set `IwYTVxMjqLtYHmwqy`.

   ```
   [GET]/v2/assets/IwYTVxMjqLtYHmwqy
   ```

   Backlot returns a response similar to the following.

   ```
   {
       "asset_type": "channel_set",
       "duration": 0,
       "name": "My Channel Set",
       "preview_image_url": null,
       "created_at": "2011-08-10T18:20:03+00:00",
       "embed_code": "IwYTVxMjqLtYHmwqy",
       "time_restrictions": null,
       "updated_at": "2011-08-10T18:20:04+00:00",
       "external_id": null,
       "hosted_at": null,
       "original_file_name": null,
       "description": null,
       "status": "live"
   }
   ```

   Note:

   Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

2. To change a channel's information, use `PATCH` with `/v2/assets`, the asset ID, and the updated properties in the request body.

The following example changes the name, "hosted at" URL, and description.

```
[PATCH]/v2/assets/dvNTVxMjrRktSlb5v{
   "name": "My Awesome Channel Set",
   "hosted_at": "http://www.mydomain.com",
   "description": "This is a set of channels."
}
```

Backlot returns a response similar to the following.

```
{
   "asset_type": "channel_set",
   "duration": 0,
   "name": "My Awesome Channel Set",
   "preview_image_url": null,
   "created_at": "2011-08-10T18:20:03+00:00",
   "embed_code": "IwYTVxMjqLtYHmwqy",
   "time_restrictions": null,
   "updated_at": "2011-08-10T18:20:04+00:00",
   "external_id": null,
   "hosted_at": "http://www.mydomain.com",
   "original_file_name": null,
   "description": "This is a set of channels.",
   "status": "live"
}
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

The channel set is successfully edited.

Adding Channels to a Channel Set

After you create a channel set, you can start adding channels.

Note: You cannot add videos directly to a channel set. To create collections of videos, use channels; see *Creating a Channel* on page 50.

To add channels to a channel set:

1. To view channels currently assigned to a channel set, use `GET /v2/assets`, the asset ID of the channel set, and the `/lineup` qualifier.

   The following example lists the channels for the `dvNTVxMjrRktSlb5v` channel.

   ```
   [GET]/v2/assets/dvNTVxMjrRktSlb5v/lineup
   ```

   Backlot returns a response similar to the following.

   ```
   [
      "Argh3lMjphu25tH",
      "RQlbdvMjq9QtGGG"
   ]
   ```

   Note:

   Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

2. To add channels to a channel set or replace the current set of channels with a new set, use `PUT` with the `/v2/assets` route, the asset ID of the channel set, the `/lineup` and the asset IDs of channels in the request body.

   The following example adds the `IzNnllMjphu2XF3` and `Y1dTdvMjq9QtOMG` channels to the `dvNTVxMjrRktSlb5v` channel set.

   ```
   [PUT]/v2/assets/dvNTVxMjrRktSlb5v/lineup[
      "IzNnllMjphu2XF3",
      "Y1dTdvMjq9QtOMG"
   ]
   ```

   Backlot returns a response similar to the following.

   ```
   [
      "IzNnllMjphu2XF3",
      "Y1dTdvMjq9QtOMG"
   ]
   ```

   Note:

   Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

3. To append a single single to a channel set without replacing the current ones, use `PUT /v2/assets` with the asset ID of the channel set, the `/lineup` qualifier, and the asset ID of the channel to add.

   The following example adds the `BrdXVjMjrgtupU3M` channel to the `dvNTVxMjrRktSlb5v` channel set.

   ```
   [PUT]/v2/assets/dvNTVxMjrRktSlb5v/lineup/BrdXVjMjrgtupU3M
   ```

   Backlot returns a response similar to the following.

   ```
   [
      "BrdXVjMjrgtupU3M",
      "IzNnllMjphu2XF3",
      "Y1dTdvMjq9QtOMG"
   ]
   ```

4. To remove a channel from a channel set, use `DELETE /v2/assets` with the asset ID of the channel set, the `/lineup` qualifier, and the asset ID of the channel to remove.

   The following example deletes the `BrdXVjMjrgtupU3M` channel from the `dvNTVxMjrRktSlb5v` channel set.

   ```
   [DELETE]/v2/assets/dvNTVxMjrRktSlb5v/lineup/BrdXVjMjrgtupU3M
   ```

   Backlot returns a response similar to the following.

   ```
   [
      "IzNnllMjphu2XF3",
      "Y1dTdvMjq9QtOMG"
   ]
   ```

The channels within the channel set are successfully updated.

ALTERNATIVE ROUTES

The Backlot API has several ways to accomplish the same tasks.

In addition to the `/v2/assets` route and its special-purpose qualifiers, there are other routes to accomplish similar tasks.

/v2/assets	Alternative	Described in...
`/v2/assets/asset_id/labels`	`/v2/labels`	*Labels* on page 166
`/v2/assets/asset_id/publishing_rules`	`/v2/publishing_rules`	*Publishing Rules* on page 163
`/v2/assets/asset_id/ad_set`	`/v2/ad_set`	*Working with Your Own Ad Assets, Ad Sets, and Ad Sources* on page 139
`/v2/assets/asset_id/players`	`/v2/players`	*Working with Your Own Ad Assets, Ad Sets, and Ad Sources* on page 139

ANALYTICS

Please note that all our platform services are currently dependent on Ooyala players being the generator of analytics data.

V3 ANALYTICS (OOYALA IQ)

Ooyala IQ Release Notes

FEATURES

Ooyala IQ is Ooyala's new analytics product. Features in Ooyala IQ allow for in-depth analysis of your content. New features in Ooyala IQ include:

- ***Multidimensional analysis***: You can now analyze more than one set of data at a time. For example, you can analyze your data for the month of July and see what videos had the most plays in Canada on the iPhone device.
- ***Consolidated reporting***: Instead of having to switch between dimension sub-report views, you now have a consolidated view of all of the dimensions in one place.
- ***Redesigned user experience***: The new Ooyala IQ UI is more insightful and focuses on minimizing the clicks involved to view the data you care the most about.
- **New dimension**: Ooyala IQ has a player dimension. In this case, player refers to a Backlot player with its own embed code. For example, if you deploy different players for each of your brands, this dimension would let you separate the traffic coming from your different players.
- ***New metric***: There is a new metric called video_starts. video_starts represents the number of times the actual video (non-ad) content begins playback.
- ***Video Details page***: A new page for each video you have. This page gives you a 360 degree view of a video, including key performance indicators, statistics on performance across dimensions, and depth around a single video.

FULL RELEASE NOTES

UI

Reports

- Different reports are no longer contained on separate pages. All dimension data is located on one page (the *Business Intelligence page*).
- Performance metrics, called the performance report in v2 Analytics, can be found on the *Business Intelligence page* and the *Analytics Dashboard*.
- Engagement metrics, called the engagement report in v2 Analytics, can be found on the *Business Intelligence page* and the *Video Details page*.
- There is no sharing report in the UI. Data are available via the Ooyala IQ API. Most users have sharing mechanisms on their own sites outside of the Ooyala Player, so be aware that the sharing data you see in the API is only the sharing that has happened through the Ooyala Player.
- There is no direct equivalent to the External Publishing report found in v2 Analytics.

Note: The algorithms used to compute various metrics in Ooyala IQ might differ slightly from those used in v2 for the Analytics Dashboard. A comparison of numbers might reveal slight differences.

Metrics

- `plays_requested`: We have improved our calculation of this metric. You may see the number of plays requested from your HTML5 and SDK players increase after 11/11/2014 due to this implementation. This may also affect your data for play conversion rate, which is calculated as `plays_requested/displays`.
- `video_starts`: We have improved our calculation of this metric. You may see the number of video starts from your HTML5 and SDK players decrease after 11/11/2014 due to this implementation. This may also affect your data for video conversion rate, which is calculated as `video_starts/plays_requested`.

Dimensions

- The v2 Analytics `platforms` dimension has been replaced in Ooyala IQ with *`device_type`*, *`os`*, and *`browser`*.
- The v2 Analytics dimension called **Domain** has been renamed in the Ooyala IQ GUI as **Traffic Source**. This represents the sites where a video has been viewed.

v3 Analytics API

The v3 Analytics (Ooyala IQ) API is a new analytics API. For general differences between v2 Analytics and v3 (Ooyala IQ), see *Reporting Application Programming Interface (API)*. For details on potential differences in the data from v2 Analytics to v3 (Ooyala IQ), see *Potential Differences in the Data* on page 112.

Metrics

- We renamed the following metrics to align the metric name with the metric definition:
  - `plays` is now `plays_requested`.
  - `avg_time_watched_per_play` is now `avg_time_watched_per_video`.
  - `start_conversion_rate` is now `video_conversion_rate`.
  - All `unique_[string]` metrics are now `uniq_[string]`.
  - `unique_plays` is now `uniq_plays_requested`.
  - `avg_plays_per_viewer` is now `avg_plays_per_user`.
- `player_conversion_rate` has been redefined as the ratio of plays requested to displays.
- `video_conversion_rate` has been redefined as the ratio of video starts to plays requested.
- The `initial_plays_requested` metric is no longer shown in the UI and is not accessible through the API.
- The `player_loads` metric is no longer shown in the UI. You can only access this metric through the API.
- The daily, weekly, and monthly uniques have been consolidated into *a single unique metric*.
- You will now use `uniq_plays_requested` as a measure of Unique Users.
- Due to low customer usage, the following metrics are not supported in Ooyala IQ (they are no longer accessible through the UI or API): `unique_replays`, `unique_autoplays`, and `unique_player_loads`.

Dimensions

- To manage the size of the datacubes, we have removed the `city`, `tag`, and `url` dimensions.
- The *domain* dimension is used to capture internet domains.

DOCUMENTATION

New documentation for Ooyala IQ includes:

- *Ooyala IQ User Guide*
- *v3 Analytics (Ooyala IQ) API Reference*
- *Ooyala IQ FAQ*

Overview

The Ooyala IQ Representational State Transfer (REST) over HTTP Application Programming Interface (API) consists of two parts:

- A write, or input, component (HTTP `POST`) for recording playback events (not available for customer use).
- A read, or output, component (HTTP `GET`) for reporting, known as the v3 Analytics Reporting API.

These are shown as parts 1 and 3 below, which is from the *conceptual overview* to Ooyala IQ in the Ooyala IQ User Guide, which you might like to read in its entirety.

	Read/ Write	More details on this API	Description
Part 1	Write: `POST`		Occurrences of events during playback are the actual metrics on which Ooyala IQ is based.
Part 2			Ooyala IQ, behind the scenes, creates datacubes (computed correlations of the various dimensions) via *data aggregation and summarization*.
Part 3	Read: `GET`	*v3 Analytics Reporting API* on page 66	This API can be used for data downloading, for loading into your own reporting systems, or for building your own custom statistical graphics. **Note:** For queries with query parameters that would exceed the HTTP GET specification limit of 230 characters, please use a POST request. Some browsers and http clients may support more than 230 characters, but we will not provide official support for queries that violate the HTTP GET specification. **Prerequisites for Using the v3 Analytics Reporting API** You need your Ooyala-issued API key and secret to sign the `GET` requests. See *Setup/Mechanics for the v3 Analytics Reporting API* on page 68.

MIGRATION TIMELINE

Existing customers will be migrated in 4 batches, with the first set of migrations beginning in May 2015 and the subsequent batches scheduled during the remainder of 2015. Customers will be contacted with further information on their scheduled migration.

V3 ANALYTICS

To retrieve analytics reports from Ooyala IQ, use the v3 Analytics Reporting API, also referred to as the v3 Analytics API and the Ooyala IQ API.

DATA RETENTION

Ooyala will only retain up to 37 months of your data. This is calculated as the current month along with the past 36 months. For example, if today is Jan 29, 2015, you have data from current month up to now (Jan 1, 2015 - Jan 29, 2015) and the past 36 full months (Jan 1, 2012 - Dec 31, 2014). Ooyala will only present and allow queries on the most recent 37 months of valid data for you in UI and API. The rest of your data

will be archived. For information on how much historical data will be migrated from v2 Analytics to Ooyala IQ, please see *Data Migration* on page 109.

JSON

The bodies of the Analytics API requests and responses rely on JavaScript Object Notation (JSON). If you are unfamiliar with JSON, there are many resources available for learning about it, starting with *http://www.json.org*.

FAQ

To learn answers to frequently asked questions about Ooyala IQ, please see *Analytics FAQ*.

v3 Analytics Best Practices

The following best practices will ensure that you have optimal query performance and speed.

- Whenever possible, if you have to retrieve a lot of data, chunk your queries into smaller bits. This makes the query much faster. For example:
  - Instead of running a query on one year of data, run a query on every quarter (4 months) and combine the CSV data afterward.
  - Instead of running a query on all of the domains you want to investigate, run a query on each domain separately and combine the CSV data afterward.
  - Instead of running a query on all of the countries you want to investigate, run separate queries on small groups of countries and combine the CSV data afterward.
- Use multi-dimensional queries only as necessary. When you do perform multi-dimensional queries (group your results by up to 3 dimensions), try to restrict the query with filters to improve query performance.
- You can run up to 300 calls to the v3 Analytics API per minute. However, for optimal performance we recommend that you, run one query at a time as we build up capacity for more customers.

Common Metric Definitions and Examples

COMMON METRIC BREAKDOWN

Here are descriptions of some of the common metrics utilized in Ooyala IQ. This is not an inclusive list.

Displays: Measures the number of times that a piece of video content is loaded and displayed within the player before it gets played. Displays are related to each individual video. Each time an embed code is changed, this event gets triggered.

Plays Requested: Measures the number of times that the "Play" button is triggered either manually or automatically. The requested content could be ad content or the actual video content. A plays requested is counted regardless of which type of content is requested to play. Plays requested currently doesn't include the Replay event.

Video Starts: Measures the number of times that actual non-ad video content starts playing. If the user initiates the playback experience and only watches a pre-roll ad without continuing on to the actual video content, a Plays requested count is reported, but not a Video Start. Video Starts is only recorded if the user waits until the actual video starts playing back.

Average Time Watched per Video: The average time watched across all the users viewing this content across the different platforms. Today, we compute Average Time Watched (per video) = hours watched/ video starts, converted to HH:MM:SS format.

Note: This number will be smaller than the duration of the video in most cases. However, seeks within the player can influence this number. If the user seeks back and forth during a playback, this will push up the

time watched and can potentially cause the average time watched to be greater than the duration of the video.

Playthrough

- **Playthrough 25%**: The number of video plays for the selected video assets that reached the state of 25% of completion.
- **Playthrough 50%**: The number of video plays for the selected video assets that reached the state of 50% of completion.
- **Playthrough 75%**: The number of video plays for the selected video assets that reached the state of 75% of completion.
- **Playthrough 100%**: The number of video plays for the selected video assets that reached the state of 100% of completion.

Note: No matter how many times the user rewinds within the same view session, once the "state" is reached it won't be counted again.

Segments Watched: The number of times each segment of a piece of video content is watched. 1 segment is defined as 2.5% of video length.

Note: If a user rewinds and watches the same segment N times, Segments Watched for that segment will count as N times.

Unique Users:

- **Ooyala mobile SDK for iOS:** The Ooyala mobile SDK for iOS generates and stores a random unique ID which is application-specific. The unique ID is generated in the "OOClientID" class and is stored in the "standardUserDefaults" object. The unique ID is valid until the application is deleted. This unique ID cannot be erased or reset by the end user without deleting the app. The application developer can store a different ID than the generated ID by erasing the existing ID [OOClientID resetID] and setting a new ID [OOClientID setID:New_ID].
- **Ooyala mobile SDK for Android:** The Ooyala mobile SDK for Android generates and store a random unique ID which is application-specific. The unique ID is generated in the "OOClientID" class and is stored in the "SharedPreferences" file. The unique ID is valid until the application is deleted. This unique ID cannot be erased or reset by the end user without deleting the app. The application developer can store a different ID by erasing the existing ID [ClientID.resetID(context)] and setting a new ID [ClientID.resetID(NEW_ID)].
- **All other environments (HTML5, Flash, Chromecast):** In other environments, a unique user is identified by local storage or cookie. To generate the GUID, Flash players use the timestamp of when the GUID is generated and append random data to it. The string is then converted to base64. To generate the GUID, HTML5 players use the current time, browser information, and random data and hash it and convert it to base64. Within the same browser on desktop, once a GUID is set by one platform it is used for both platforms for the user. If a user clears their browser cache, that user/device's ID will get regenerated next time they watch video. Incognito modes will track a user for a single session, but once the browser is closed the GUID is erased.

The generated IDs are completely random and don't include any user-identifiable information. When such information is not available for a user (there is no local storage or cookie), a new unique identifier will be created for that user. We de-duplicate when calculating the number of unique users over time. For example, day 1 has users A, B, C; day 2 has users B, E., then when you pick date range = day 1 and day 2, then total unique users = 4 (A, B, C, E), Daily Avg. Unique Users = (3+2) / 2 = 3 (2.5 is converted to the closed integer).

EXAMPLES

The following examples show how many displays, plays requested and video starts Ooyala IQ would record in different situations.

Example 1: Pre-roll Ads

The publisher has pre-roll ads delivered during playback.

User A begins watching video X with 2 pre-roll ads but leaves after the first ad. User B begins watching video X (again with 2 pre-roll ads), and leaves after both ads and one minute of play time.

Table 4: Metric Tallies for Example 1

	Displays	Plays Requested	Video Starts	Avg. Time Watched/Video
User A	1	1	0	00:00:00
User B	1	1	1	00:01:00
Video X in total (User A + User B)	2	2	1	00:01:00

Example 2: Replays

The user clicks replay to play the content again during the same viewing experience.

User A begins watching Video X with 1 pre-roll ad, watches the ad and the content all the way through and then hits re-play and watches it all the way again. Alternatively, User B, in the second replay, watches the ad and then exits.

Table 5: Metric Tallies for Example 2

	Displays	Plays Requested	Video Starts
User A	1	1	1
User B	1	1	1

Example 3: Mid-roll Ads

The publisher also delivers mid-roll ads during the content playback experience.

User A begins watching Video X with 2 pre-roll ads, watches the ads, starts watching the content, sees a mid-roll ad and exits the video. User B begins watching Video X with 2 pre-roll ads, watches the ads, starts watching the content, watches the mid-roll ads and completes the video.

Table 6: Metric Tallies for Example 3

	Displays	Plays Requested	Video Starts
User A	1	1	1
User B	1	1	1
Video X in total (User A + User B)	2	2	2

Example 4: Seeks

The user can also seek through content back and forth during a single viewing experience within the player.

User A begins watching Video X with 2 pre-roll ads, watches both ads, starts watching the content, watches a mid-roll ad, continues watching and then seeks back to the beginning and plays the content. The user is now shown the mid-roll ad again. This results in 1 display, 1 play requested, 1 video start.

Table 7: Metric Tallies for Example 4

	Displays	Plays Requested	Video Starts
User A	1	1	1

Example 5: Autoplays

The publisher has autoplay turned on - this means that the video player starts playing the content automatically when the user visits the page containing the player.

User A visits the page (where autoplay is turned on) with 1 pre-roll ad, watches the ad and exits before the video begins playing. User B visits the page (where autoplay is turned on) with 1 pre-roll ad, and exits the page halfway through the video content.

Table 8: Metric Tallies for Example 5

	Displays	Plays Requested	Video Starts
User A	1	1	0
User B	1	1	1
Video X in total (User A + User B)	2	2	1

Example 6: Playthrough

This example shows how playthrough is counted. The video in this example is 4 minutes in length.

User A watches the video content from its beginning to 2 minutes and 30 seconds and then stops watching.

User B watches the video content from its beginning to 3 minutes and 30 seconds, then rewinds to the beginning and watches all the way to 1 minute and 30 seconds, then stops watching.

User C seeks to 2 minutes directly and watches from there to 2 minutes and 30 seconds then stops watching.

Table 9: Metric Tallies for Example 6

	Playthrough 25%	Playthrough 50%	Playthrough 75%	Playthrough 100%
User A	1	1	0	0
User B	1	1	1	0
User C	1	1	0	0

Example 7: Segments Watched, Percentage Watched

In this example a video is 20-minutes in length. This video has 40 buckets assigned to it, where each bucket is 30 seconds in length. Percentage watched is calculated as (buckets watched/total number of buckets) * 100.

User A plays the video and watches it through 1 minute.

User B plays the video and watches it through 1 minute, then rewinds to 40 seconds and watches through 1 minute again during the same video view session (the user didn't reload the player).

User C plays the video and watches it through 1 minute, then fast-forwards to 3 minutes and 31 seconds of the video content and watches through the end from there.

Table 10: Metric Tallies for Example 7

	Segments Watched	Percentage Watched
User A	1st bucket has 1 play 2nd bucket has 1 play	5%
User B	1st bucket has 1 play 2nd bucket has 2 plays	5%
User C	1st bucket has 1 play 2nd bucket has 1 play 40th bucket has 1 play	7.5%

In User B's scenario, the second bucket has 2 plays because the user rewinded within the second bucket. This is because segments watched captures the total number of plays each specific segment incurs in order to measure its popularity (no matter if it's from the same user rewinding or from different users watching it). So within the same video view session the segments watched number could increase if a user rewinds. Percentage watched is still 5% for user B because the user still only watched 2 buckets (1st and 2nd). Percentage watched is used to measure the percentage of the whole video content that ever got watched (the count won't increase if the same user rewinds within the bucket that already got watched during the same video view session).

Percentage watched measures how many percentages of the whole video content ever got watched (40% watched could mean the first 40% of the content got watched or it could mean first 20% of the content got watched and last 20% of the content got watched).

Example 8: Unique Users

This illustrates how unique users are counted. You have a video on your website "myExampleSite.com".

Situation A: User W watches your video on their iPhone with application 1. Later in the day they decide to watch the video again on their iPhone with application 1.

Situation B: User X watches your video on their laptop. They leave the browser window up and go about their day. Twelve hours later they come back to their laptop and re-watch the video.

Situation C: User Y watches your video on their tablet. They then watch the video on their laptop, while still using the same wifi network as before.

Situation D: User Z watches your video on their laptop in Browser 1. After the video finishes they clear their cookies. They then watch the video again in Browser 1 on their laptop.

Table 11: Metric Tallies for Example 8

	Unique Users
Situation A	1
Situation B	1
Situation C	2
Situation D	2

API Server Endpoints

Make your requests to these endpoints.

Ooyala has several API server endpoints. Use the one appropriate to your needs.

Table 12: Server Endpoints

Endpoint	Service	Description
player.ooyala.com/	Player API	For video playback.
http://api.ooyala.com/	Backlot API (including Ooyala IQ v3 Analytics Reporting API)	For normal usage of the Backlot API. Requests to this endpoint are rate-limited; see *API Rate Limiting* on page 13.
https://cdn-api.ooyala.com/	Backlot API	High-performance endpoint for the Backlot API: unlimited `GET` requests against the cache. See *High Performance API Endpoint* on page 13.
http://rl.ooyala.com	Ooyala Rights Locker	Ooyala Rights Locker entitlements. See *http://support.ooyala.com/documentation/developers/chapter_rightslocker.html*.

Practice Making Requests with the Scratchpad

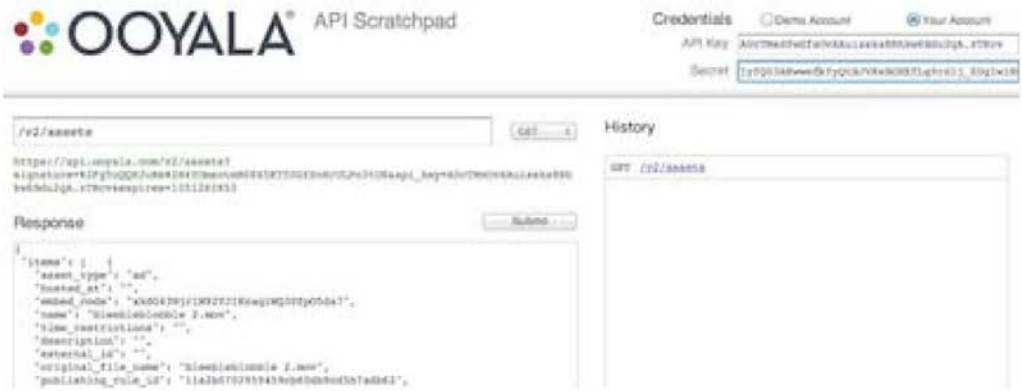
You can make practice requests on the scratchpad.

For your experimentation and learning, Ooyala provides a "scratchpad" where you can practice making requests, see the structure of responses, and familiarize yourself with Ooyala's APIs in general.

Note: The scratchpad accesses your live information and assets. Any changes you make in the scratchpad are reflected in production.

The scratchpad is located at *https://api.ooyala.com/docs/api_scratchpad*.

In the upper right, you can choose to use either a demo account (which allows only `GET` requests) or your own account (which allows all methods). To use your own account, provide your account's Ooyala API credentials, which are described in *Your API Credentials* on page 9.



v3 Analytics (Ooyala IQ) API

The following applies only if you are migrated to Ooyala IQ. For v3 Analytics (Ooyala IQ), prefix /v3 to your route. For example, to retrieve the performance report, select GET and input the following:

```
 /v3/analytics/reports/?
report_type=performance&dimensions=asset&start_date=2015-01-01&end_date=2015-01-07
```

v3 Analytics Reporting API

This section details the v3 Analytics Reporting API, also referred to as the v3 Analytics API and the Ooyala IQ API.

The v3 Analytics Reporting API relies on the following basic concepts, almost all of which are described in *How Ooyala IQ Works* in the Ooyala IQ User Guide:

- ***Dimensions***: common criteria that are used to aggregate data, such as the date on which the user activity occurred or the country where the users were located.
- ***Metrics***: measurements of individual events related to your content, such as video plays requested or pauses.
- ***Filters***: dimension values that constrain the retrieved data. For example, you can retrieve data for a specific type of device, a specific video, or specific dates.

BEST PRACTICES

The following best practices will ensure that you have optimal query performance and speed.

- Whenever possible, if you have to retrieve a lot of data, chunk your queries into smaller bits. This makes the query much faster. For example:
  - Instead of running a query on one year of data, run a query on every quarter (4 months) and combine the CSV data afterward.
  - Instead of running a query on all of the domains you want to investigate, run a query on each domain separately and combine the CSV data afterward.
  - Instead of running a query on all of the countries you want to investigate, run separate queries on small groups of countries and combine the CSV data afterward.
- Use multi-dimensional queries only as necessary. When you do perform multi-dimensional queries (group your results by up to 3 dimensions), try to restrict the query with filters to improve query performance.
- You can run up to 300 calls to the v3 Analytics API per minute. However, for optimal performance we recommend that you, run one query at a time as we build up capacity for more customers.

REQUEST SIGNING AND EXPIRATION

All reporting requests must be signed by your Ooyala-issued API key and secret, and your GET requests must have an explicit expiration time. How to do this is detailed in *Setup/Mechanics for the v3 Analytics Reporting API* on page 68.

Note: For ease of reading, the majority of examples here are *not* signed requests.

ENDPOINT FOR REQUESTS

All your reporting `GET` requests must be made to the `http://api.ooyala.com/` endpoint.

v3 Analytics Reporting API Syntax

The v3 Analytics Reporting API `GET` or `POST` request must be signed by your Ooyala-issued API key and secret as detailed in *Setup/Mechanics for the v3 Analytics Reporting API* on page 68 and must include the `expires` keyword and timestamp.

In addition, the individual values of query string parameters and the entire value for *`filters=`* must be URL-encoded.

General Syntax of Reporting GET

The base syntax of the route and query string is as follows. For ease of reading, the single-line request has been split across several lines.

```
[GET] /v3/analytics/reports/?
            report_type=type
            &dimensions=dimensions
```


```
&metrics=metrics
&filters=filter_type=='filter_value'
&start_date=date
&end_date=date
&other_parms
&api_key=your_api_key
```

- The **required** query string parameters (shown in bold) are *report_type*, start_date, and api_key.
- If no dimensions are specified, total values across all dimension are returned.
- You can specify up to 3 dimensions at a time.
- If no metrics are specified, all metrics are returned.

Note: At this time, the only valid value for *report_type* is performance.

General Syntax of Reporting Long Queries POST

For queries with query parameters that would exceed the HTTP GET specification limit of 230 characters , please use a POST request. Some browsers and http clients may support more than 230 characters, but we will not provide official support for queries that violate the HTTP GET specification. For POST requests, pass a JSON object in the request body instead of the query string parameters.

```
[POST] /v3/analytics/reports

{
    "report_type":"type",
    "dimensions":"dimensions",
    "metrics":"metrics",
    "filters":"(filter_type==\"filter_value\")",
    "start_date":"date",
    "end_date":"date",
    "other_parms":"other_param_value",
    "api_key":"your_api_key"
}
```

Setup/Mechanics for the v3 Analytics Reporting API

To use the v3 Analytics Reporting API, you need your Ooyala-supplied key and secret so you can sign your GET or POST requests, which must also have an explicit expiration time.

Your API Credentials

Your Ooyala API credentials include your API key and secret, which you provide to sign HTTP requests and when using the scratchpad.

Regardless of which endpoint you use, all requests must be signed by your Ooyala API credentials (your API key and API secret), and a signature computed from these and the request arguments. The API key and secret are viewable in your Backlot UI account. In the Backlot UI, go to the **ACCOUNT** tab, **Developers** subtab; the key and secret are on the left.

In addition, if you are a publisher (sometimes called "provider" or "partner"), Ooyala assigns you a pcode, which is required on some API requests. An API key has the following format: <*pcode*>.<*string*>. Your pcode is the alphanumeric string that precedes the period. For example, if the API key is xxxxxxxxxxxxxxxxxxxxxxxxxxxx.xxxxx, then use the first 28 characters (xxxxxxxxxxxxxxxxxxxxxxxxxxxx) for the pcode.

For the general algorithm for signing requests, see *General Algorithm for Signing Requests* on page 10.

For code samples in various programming languages, see *Sample Code for Signing Requests* on page 173 on the Support Site.

In addition, you can use your API credentials to make Backlot and v3 Analytics requests on the scratchpad at *https://api.ooyala.com/docs/api_scratchpad*.

General Algorithm for Signing Requests

Every request made to Backlot requires three query string parameters for authentication: the API Key, the request expiration, and the signature.

To sign a request:

1. Start with your 40 character secret key (see the **Developers** tab in the Backlot UI); it is unique for each user and should always be kept secure and private. For details, see *Your API Credentials* on page 9.

   This example uses `329b5b204d0f11e0a2d060334bfffe90ab18xqh5` as the secret key.

   ```
   fakef391e36243a94d54e5da667020d3
   ```

2. Append the HTTP method (e.g. "GET", "POST", "PUT"):

   ```
   fakef391e36243a94d54e5da667020d3GET
   ```

3. Append the request path or route:

   ```
   fakef391e36243a94d54e5da667020d3GET/v2/players/HbxJK
   ```

4. Append any query string parameters, sorted alphabetically by keys. This includes the required API Key (see the **Developers** tab in the Backlot UI) and the `expires` parameter.

   Note: Do not URL-encode these parameters.

   ```
   fakef391e36243a94d54e5da667020d3GET/v2/players/
   HbxJKMapi_key=7ab06expires=1299991855
   ```

5. If your request has a body, append the entire request body to the string.
6. From this string, you can now generate a SHA-256 digest in base64, truncate the string to 43 characters, and remove any trailing = signs. This example produces the following signature:

   ```
   p9DG/+ummS0YcTNOYHtykdjw5N2n5s81OigJfdgHPTA
   ```

7. URL encode the signature:

```
p9DG%2F%2BummS0YcTNOYHtykdjw5N2n5s81OigJfdgHPTA
```

8. Append this signature to your request URL as a query string parameter. You can now visit this URL to make your request. The following example is the final signed URL:

```
https://api.ooyala.com/v2/players/HbxJKM?
api_key=7ab06&expires=1299991855&signature=p9DG%2F
%2BummS0YcTNOYHtykdjw5N2n5s81OigJfdgHPTA
```

Setting an Expiration Time on Requests

You must set the `expires` parameter on your HTTP requests.

Your HTTP requests must include the `expires` parameter to indicate when the request becomes invalid.

The value for the `expires` parameter is the time in seconds since January 1, 1970 ("the epoch") after which the request is invalid, as shown in the following example:

```
[GET]https://api.ooyala.com/v2/players/somePlayerId?
api_key=someKey&signature=someSignature&expires=1930294539
```

Character Encoding and MIME Types

Encode your data in UTF-8 and set the proper MIME type on requests with bodies.

The API requires the following:

1. Make sure that your character data are encoded in UTF-8.
2. On HTTP requests that contain a request body in JSON format (POST, PUT, or PATCH), make sure to set the MIME type:

```
Content-type: application/json
```

3. Make sure non-ASCII letters and ' characters are escaped.

Preprogrammed Report Types

The types of preprogrammed reports in the Analytics V3 *Reporting API* are described below. "Preprogrammed" means that the various report types each include default dimensions and metrics, without your having to specify any. You can, however, modify the preprogrammed report with whatever dimensions, metrics, or filters you want.

The report type value is specified with the *`report_type=`* query string parameter.

Report Type	Description	Data generally available after event
`performance`	Shows comprehensive performance data for all your videos, along with sharing information, engagement information, and unique information. • Sharing information shows the number of shares (on Facebook and other social media) for all your videos. • The engagement information shows data related to user behavior. Can help you determine the effects of advertising	1 hour

Report Type	Description	Data generally available after event
	placements. It displays a count of the users at certain points as a video is played. (These are sometimes called "drop-off points.") With this information, you can identify those points in your video where users possibly lose interest and stop viewing. • Unique information shows data related to unique users.	

v3 Analytics Reporting API Request Examples

The business use case examples here illustrate a variety of API requests to return data suitable for analysis to gain business insights.

Be sure to *sign your requests* and URL-encode your parameter values. The examples here do not show signed or URL-encoded requests.

For additional examples, see *API Requests: v2 Analytics and v3 Analytics Comparison* on page 76.

How Are My Different Brands Performing?

Context: Your company has many different brands, so you use a different player for each brand. You want to see how each brand is performing.

HOW TO IN V3 ANALYTICS

Query against the `asset` and player_id dimensions, filtering to select only player 1 and player 2. The results include the `plays_requested` metric.

```
[GET] /v3/analytics/reports/?
report_type=performance
&dimensions=asset,player_id
&filters=player_id=='player1',player_id=='player2'
&metrics=plays_requested
&start_date=2014-10-10
&end_date=2014-10-20
&api_key=yourApiKey
```

```
https://api.ooyala.com/v3/analytics/reports/?
report_type=performance&dimensions=asset,player_id&filters=player_id=='player1'player_id=
signature=yourSignature
```

Response:

```
{
   "status": "OK",
   "results":[
      {
         "data":[
            {
              "metrics":{
                  "plays_requested":2500
               },
               "group":{
                  "asset":"asset_id_1",
                  "player_id":"player1",
                  "name":"unknown",
                  "status":"unknown",
```

```
                "type":"unknown"
            }
        },
        {
            "metrics":{
                "plays_requested":2200
            },
            "group":{
                "asset":"asset_id_2",
                "player_id":"player2",
                "name":"unknown",
                "status":"unknown",
                "type":"unknown"
            }
        }
    ],
    "start_date":"2014-10-10T00:00Z",
    "end_date":"2014-10-20T00:00Z"
  }
],
"status_code": "200",
"total_count": "100",
"result_count": "2"
}
```

What Geographies Are Driving My Content Traffic?

Context: Your content reaches viewers in multiple countries. You want to see what portion of your traffic is driven by each geography your content reaches.

HOW TO IN V3 ANALYTICS

Query against the *country* dimension and filter by domain='exampledomain.com' and device_type='tablet'. Sort by the *plays_requested* metric, as well, limiting the results to the month of February, 2015.

```
[GET] /v3/analytics/reports/?
report_type=performance
&dimensions=country
&filters=((device_type=='tablet')) AND ((domain=='exampledomain.com'))
&metrics=plays_requested
&start_date=2014-10-10
&end_date=2014-10-20
&sort=plays_requested
&api_key=yourApiKey
```

```
https://api.ooyala.com/v3/analytics/reports/?
report_type=performance&dimensions=country&amp;filters=%28%28device_type%3D
%3D%27tablet%27%29%29+AND+%28%28domain%3D%3D%27exampledomain.com%27%29%29
&metrics=plays_requested&start_date=2014-10-10&end_date=2014-10-20&sort=plays_requested&.
expires=yourExpiration&signature=yourSignature
```

Response:

```
{
   "status":"OK",
   "results":[
      {
         "data":[
            {
```

```
            "metrics":{
               "plays_requested":11000
            },
            "group":{
               "country":"us",
               "countryName":"United States"
            }
         },
         {
            "metrics":{
               "plays_requested":5100
            },
            "group":{
               "country":"fr",
               "countryName":"France"
            }
         },
         {
            "metrics":{
               "plays_requested":2000
            },
            "group":{
               "country":"gb",
               "countryName":"United Kingdom"
            }
         },
         {
            "metrics":{
               "plays_requested":2000
            },
            "group":{
               "country":"be",
               "countryName":"Belgium"
            }
         },
         {
            "metrics":{
               "plays_requested":1600
            },
            "group":{
               "country":"ca",
               "countryName":"Canada"
            }
         },
         {
            "metrics":{
               "plays_requested":1400
            },
            "group":{
               "country":"de",
               "countryName":"Germany"
            }
         },
         {
            "metrics":{
               "plays_requested":1100
            },
            "group":{
               "country":"ua",
               "countryName":"Ukraine"
            }
         },
         {
            "metrics":{
```


```
                "plays_requested":1000
            },
            "group":{
                "country":"se",
                "countryName":"Sweden"
            }
        },
        {
            "metrics":{
                "plays_requested":800
            },
            "group":{
                "country":"nl",
                "countryName":"Netherlands"
            }
        },
        {

            "metrics":{
                "plays_requested":600
            },
            "group":{
                "country":"it",
                "countryName":"Italy"
            }
        }
    ],
    "start_date":"2014-10-10T00:00Z",
    "end_date":"2014-10-20T00:00Z"
  }
],

"status_code":"200",
"total_count":100,
"result_count":10
}
```


What Filters Can I Use?

Some example filters are shown below. These are not fully formed requests (see *v3 Analytics Reporting API Syntax* on page 67 for complete syntax), only examples of the `filters` query parameter.

For queries with query parameters that would exceed the HTTP GET specification limit of 230 characters , please use a *POST request*.

Note: For up to 2 filters you can set a date range of up to 1 year (366 days). For 3 filters you can set a date range of up to 1 month (31 days).

GROUPING FILTERS

- Return data with the country Australia (AU; see *Country and Location Codes* on page 100) and the `mobile` device type:

```
...&filters=((country=='AU')) AND ((device_type=='mobile'))
```

- Mobile and tablet devices in Colombia:

```
...&filters=country=='CO',device_type=='mobile',device_type=='tablet'
```


- Only mobile in Columbia and the US:

```
...&filters=((device_type=='mobile') AND (country=='CO',country=='US'))
```

FILTERING BY LABEL

Ooyala labels are a mechanism for grouping videos. In Ooyala Analytics, labels can be used to filter. Use the `label_id` to identify the label filter.

Keep in mind the following points about working with labels in Ooyala Analytics:

- Correlations among labels and associated videos are not pre-aggregated or pre-summarized (rolled up) but are resolved at the time of an API request.
- You must have created an explicit association between a video and all of the individual labels you want. The Ooyala system does not aggregate or summarize based on a label's apparent hierarchy, only on explicit associations. For example, imagine two labels, the parent "Cycle World" and its child label "Sport Bikes". A video labeled with the child "Sport Bikes" is rolled up into the parent label "Cycle World" only if the video is also explicitly labeled "Cycle World".

This request returns totals for *all* metrics, grouped by a label with the label_id 123abc456yui123xyz.

```
...&filters=label_id=='123abc456yui123xyz'
```

How Can I Make a Long and Detailed Query?

Context: You want to use the API to make a long query that uses many text characters.

HOW TO IN V3 ANALYTICS

For queries with query parameters that would exceed the HTTP GET specification limit of 230 characters, use a POST request and pass a JSON object in the request body.

```
[POST] /v3/analytics/reports
{
    "report_type":"performance",
    "start_date":"2014-11-26",
    "end_date":"2014-12-04",
    "filters": "((device_type==\"desktop\")) AND ((country==\"us\"))",
    "metrics": "plays_requested,displays,video_starts",
    "sort":"plays_requested",
    "dimensions":"device_type",
    "limit": 12,
    "api_key":"(yourApiKey)"
}
```

```
~$: curl -H "Content-Type: application/json" -d '{"report_type":"performance",
"start_date":"2014-11-26", "end_date":"2014-12-04", "filters":
"((device_type==\"desktop\")) AND ((country==\"us\"))", "metrics":
"plays_requested,displays,video_starts", "sort":"plays_requested",
"dimensions":"device_type", "limit": 12, "api_key":"(YOUR API KEY HERE)"}'
http://api.ooyala.com/v3/analytics/reports
```


Response:


```
{
   "status":"OK",
   "results":[
      {
         "data":[
            {
               "metrics":{
                  "displays":11,
                  "plays_requested":7,
                  "video_starts":7
               },
               "group":{
                  "device_type":"desktop"
               }
            }
         ],
         "start_date":"2014-11-26T00:00Z",
         "end_date":"2014-12-4T00:00Z"
      }
   ],

   "status_code":"200",
   "total_count":"1",
   "result_count":"1"
}
```


API Requests: v2 Analytics and v3 Analytics Comparison

These examples show some commonly used queries in v2 Analytics and their equivalents in v3 Analytics (Ooyala IQ).

For ease of reading:

- The v3 query parameter values are not URL-encoded.
- Variable values are prefixed with a $ sign.
- Some requests are broken across lines.
- "&metrics=metrics" is left out of the syntax because by default all metrics are returned. Define specific metrics to return in your reports with "&metrics=yourDesiredMetrics".

Examples here include the following:

- *Total Dimensions*
- *Total with Breakdown by Day*
- *Geo Dimensions* on page 80
- *Device and Platform Dimensions* on page 83

TOTAL DIMENSIONS

v2 URL Syntax	v3 Equivalent
`/v2/analytics/reports/account/performance/total/$date_range`	`/v3/analytics/reports/?report_type=performance&start_date=$start&end_date=$end&api_key=yourApiKey`

```
https://api.ooyala.com/v3/analytics/reports/?
report_type=performance&start_date=YYYY-MM-DD&end_date=YYYY-MM-
DD&api_key=yourApiKey&expires=yourExpiration&signature=yourSignature
```

Sample Ooyala IQ Response:


```
{
     "status_code":"200",
     "result_count":1,
     "results":[
      {
          "start_date":"YYYY-MM-DD",
          "end_date":"YYYY-MM-DD",
          "data":[
            {
                 "group":{
                    },
                 "metrics":{
                     "playthrough_50": "40000",
                     "uniq_displays": "3000000",
                     "video_starts": "4000000",
                     "displays": "5000000",
                     "percentage_watched":[
                        18,
                        14,
                        13,
                        13,
                        15,
                        20,
                        12,
                        14,
                        12,
                        11,
                        19,
                        18,
                        8,
                        13,
                        11,
                        18,
                        16,
                        10,
                        15,
                        17,
                        15,
                        16,
                        8,
                        10,
                        12,
                        13,
                        8,
                        9,
                        8,
                        14,
                        8,
                        10,
                        11,
                        11,
                        12,
                        10,
                        11,
                        5,
                        0,
```


```
                    14
                ],
                "player_loads":5000000,
                "segment_watched":[
                    18,
                    14,
                    13,
                    13,
                    15,
                    20,
                    12,
                    14,
                    12,
                    11,
                    19,
                    18,
                    8,
                    13,
                    11,
                    18,
                    16,
                    10,
                    15,
                    17,
                    15,
                    16,
                    8,
                    10,
                    12,
                    13,
                    8,
                    9,
                    8,
                    14,
                    8,
                    10,
                    11,
                    11,
                    12,
                    10,
                    11,
                    5,
                    0,
                    14
                ],
                "urls_copied": "3",
                "plays_requested": "6500",
                "embeds_copied": "2",
                "facebook": "3",
                "replays": "145",
                "uniq_video_starts": "2000000",
                "autoplays": "0",
                "emails_sent": "0",
                "digg": "0",
                "playthrough_25": "50000",
                "playthrough_75": "40000",
                "twitter": "0",
                "uniq_plays_requested": "60000",
                "playthrough_100": "30000",
                "time_watched": "7500000000"

            }
        }
    ]
```


```
        }
      ],
        "total_count":1,
        "status":"OK"
   }
```

Note: There is no data in the group because there's no dimension specified, and the plays requested, displays is grand total for all the metrics across the time frame

TOTAL WITH BREAKDOWN BY DAY

v2 URL Syntax	v3 Equivalent
`/v2/analytics/reports/account/performance/total/$date_range?breakdown_by=day`	`/v3/analytics/reports/?report_type=performance&time_segment=day&start_date=$start&end_date=$end&api_key=yourApiKey`

```
https://api.ooyala.com/v3/analytics/reports/?report_type=performance&time_segment=day&start_date=YYYY-MM-DD&end_date=YYYY-MM-DD&api_key=yourApiKey&expires=yourExpiration&signature=yourSignature
```

Sample Ooyala IQ Response:

```
{
   "status_code":"200",
   "result_count":3,
   "results":[
      {
         "start_date":"YYYY-MM-DD",
         "end_date":"YYYY-MM-DD",
         "data":[
            {
               "group":{

               },
               "metrics":{
                 "playthrough_50":100,
                 "uniq_displays":2500,
                 "video_starts":3000,
                 ...
                 "time_watched":240000
         }
            }
         ]
      },
   {
      "start_date":"YYYY-MM-DD",
      "end_date":"YYYY-MM-DD",
      "data":[
         {
            "group":{

            },
            "metrics":{
              "playthrough_50":100,
              "uniq_displays":2500,
              "video_starts":3000,
```


```
                ...
                "time_watched":240000
            }
            }
        ]
    },
{
    "start_date":"YYYY-MM-DD",
    "end_date":"YYYY-MM-DD",
    "data":[
        {
            "group":{

            },
            "metrics":{
                "playthrough_50":100,
                "uniq_displays":2500,
                "video_starts":3000,
                ...
                "time_watched":240000
            }
        }
    ]
}
],
"status":"OK",
"total_count":3
}
```

This returns metric totals broken down by day.

GEO DIMENSIONS

v2 URL Syntax	v3 Equivalent
`/v2/analytics/reports/asset/$asset_id/performance/countries/$country/regions/$region/$date_range`	`/v3/analytics/reports/?report_type=performance&dimensions=asset,country,region&filters=asset=='asset_id',country=='country',r•&start_date=$start&end_date=$end&api_key=yourApiKey`

```
https://api.ooyala.com/v3/analytics/reports/?
report_type=performance&dimensions=asset,country,region&filters=asset=='asset_id',
country=='country',region=='region'&start_date=YYYY-MM-DD&end_date=YYYY-MM-DD
&api_key=yourApiKey&expires=yourExpiration&signature=yourSignature
```

Sample Ooyala IQ Response:

```
{
     "status_code":"200",
     "result_count":1,
     "results":[
      {
          "start_date":"YYYY-MM-DD",
          "end_date":"YYYY-MM-DD",
          "data":[
            {
                 "group":{
                     "asset":"asset_id",
```

```
            "country":"Country Code",
            "countryName":"Country Name",
            "region":"Region",
            "name":"Asset Name",
            "status":"live",
            "type":"video"
            },
        "metrics":{
            "playthrough_50": "40000",
            "uniq_displays": "3000000",
            "video_starts": "4000000",
            "displays": "5000000",
            "percentage_watched":[
               18,
               14,
               13,
               13,
               15,
               20,
               12,
               14,
               12,
               11,
               19,
               18,
               8,
               13,
               11,
               18,
               16,
               10,
               15,
               17,
               15,
               16,
               8,
               10,
               12,
               13,
               8,
               9,
               8,
               14,
               8,
               10,
               11,
               11,
               12,
               10,
               11,
               5,
               0,
               14
            ],
            "player_loads":5000000,
            "segment_watched":[
               18,
               14,
               13,
               13,
               15,
               20,
               12,
               14,
```


```
                        12,
                        11,
                        19,
                        18,
                        8,
                        13,
                        11,
                        18,
                        16,
                        10,
                        15,
                        17,
                        15,
                        16,
                        8,
                        10,
                        12,
                        13,
                        8,
                        9,
                        8,
                        14,
                        8,
                        10,
                        11,
                        11,
                        12,
                        10,
                        11,
                        5,
                        0,
                        14
                    ],
                    "urls_copied": "3",
                    "plays_requested": "6500",
                    "embeds_copied": "2",
                    "facebook": "3",
                    "replays": "145",
                    "uniq_video_starts": "2000000",
                    "autoplays": "0",
                    "emails_sent": "0",
                    "digg": "0",
                    "playthrough_25": "50000",
                    "playthrough_75": "40000",
                    "twitter": "0",
                    "uniq_plays_requested": "60000",
                    "playthrough_100": "30000",
                    "time_watched": "7500000000"

                }
            }
        ]
    }
],
    "total_count":1,
    "status":"OK"
}
```


DEVICE AND PLATFORM DIMENSIONS

v2 URL Syntax	v3 Equivalent
`/v2/analytics/reports/asset/$asset_id/performance/device_types/$device_type/platforms/$platform/$date_range`	`/v3/analytics/reports/?report_type=performance&dimensions=asset,device_type&filters=((device_type=='$device_type')) AND ((asset_id=='$asset_id'))&start_date=$start&end_date=$end&api_key=yourApiKey`

```
https://api.ooyala.com/v3/analytics/reports/?
report_type=performance&dimensions=asset,device_type&filters=
%28%28device_type%3D%3D%27device_type%27%29%29+AND+%28%28asset_id%3D
%3D%27$asset_id%27%29%29 &start_date=YYYY-MM-DD&end_date=YYYY-MM-DD
&api_key=yourApiKey&expires=yourExpiration&signature=yourSignature
```

Sample Ooyala IQ Response:


```
{
     "status_code":"200",
     "result_count":1,
     "results":[
      {
           "start_date":"YYYY-MM-DD",
           "end_date":"YYYY-MM-DD",
           "data":[
             {
                  "group":{
                      "asset":"asset_id",
                      "name":"Asset Name",
                      "status":"live",
                      "type":"video",
                      "device_type":"device_type"
                      },
                  "metrics":{
                      "playthrough_50": "40000",
                      "uniq_displays": "3000000",
                      "video_starts": "4000000",
                      "displays": "5000000",
                      "percentage_watched":[
                         18,
                         14,
                         13,
                         13,
                         15,
                         20,
                         12,
                         14,
                         12,
                         11,
                         19,
                         18,
                         8,
                         13,
                         11,
```


```
            18,
            16,
            10,
            15,
            17,
            15,
            16,
            8,
            10,
            12,
            13,
            8,
            9,
            8,
            14,
            8,
            10,
            11,
            11,
            12,
            10,
            11,
            5,
            0,
            14
        ],
        "player_loads":5000000,
        "segment_watched":[
            18,
            14,
            13,
            13,
            15,
            20,
            12,
            14,
            12,
            11,
            19,
            18,
            8,
            13,
            11,
            18,
            16,
            10,
            15,
            17,
            15,
            16,
            8,
            10,
            12,
            13,
            8,
            9,
            8,
            14,
            8,
            10,
            11,
            11,
            12,
            10,
```


```
                        11,
                        5,
                        0,
                        14
                    ],
                    "urls_copied": "3",
                    "plays_requested": "6500",
                    "embeds_copied": "2",
                    "facebook": "3",
                    "replays": "145",
                    "uniq_video_starts": "2000000",
                    "autoplays": "0",
                    "emails_sent": "0",
                    "digg": "0",
                    "playthrough_25": "50000",
                    "playthrough_75": "40000",
                    "twitter": "0",
                    "uniq_plays_requested": "60000",
                    "playthrough_100": "30000",
                    "time_watched": "7500000000"

                }
            }
        ]
    }
],
    "total_count":1,
    "status":"OK"
}
```

Parameter Reference

Reporting Query String Parameters

Here are the complete reference details for all query string parameters for the v3 Analytics *Reporting API*.

All parameter names are case-sensitive and lowercase.

The entire query string must be URL-encoded.

General Syntax of Reporting GET

The base syntax of the route and query string is as follows. For ease of reading, the single-line request has been split across several lines.

```
[GET] /v3/analytics/reports/?
            report_type=type
            &dimensions=dimensions
            &metrics=metrics
            &filters=filter_type=='filter_value'
            &start_date=date
            &end_date=date
            &other_parms
            &api_key=your_api_key
```

- The **required** query string parameters (shown in bold) are `report_type`, `start_date`, and api_key.
- If no dimensions are specified, total values across all dimension are returned.
- You can specify up to 3 dimensions at a time.
- If no metrics are specified, all metrics are returned.

Note: At this time, the only valid value for *report_type* is `performance`.

General Syntax of Reporting Long Queries POST

For queries with query parameters that would exceed the HTTP GET specification limit of 230 characters , please use a POST request. Some browsers and http clients may support more than 230 characters, but we will not provide official support for queries that violate the HTTP GET specification. For POST requests, pass a JSON object in the request body instead of the query string parameters.

```
[POST] /v3/analytics/reports

{
    "report_type":"type",
    "dimensions":"dimensions",
    "metrics":"metrics",
    "filters":"(filter_type==\"filter_value\")",
    "start_date":"date",
    "end_date":"date",
    "other_parms":"other_param_value",
    "api_key":"your_api_key"
}
```

Parameter	Description	Required
`report_type`	Specifies the type of report. Valid values: `performance` Default: None Example: `report_type=performance`	Yes
`start_date`	The start date is specified as *YYYY-MM-DD* The value of `start_date` is the provider's timezone. Default: None Limitations: You can query a range of up to 1 year (366 days) for up to 2 filters and up to 1 month (31 days) for 3 filters. Example: `start_date=2014-10-28`	Yes
`end_date`	The end date is specified as *YYYY-MM-DD* **Note:** The end date is not inclusive. That is, data in the response is up to but not including the end date. Default: Tomorrow's date in the provider's timezone.	No

Parameter	Description	Required
	Limitations: You can query a range of up to 1 year (366 days) for up to 2 filters and up to 1 month (31 days) for 3 filters. Example: To get data through the end of 2014-10-29: `end_date=2014-10-29`	
`metrics`	List of comma-separated metric names. Default: * (all metrics). Valid values: See *Metrics* on page 91 section for details. Example: `metrics=plays_requested,displays`	No
`dimensions`	List of comma-separated dimension names. Results are grouped by the specified dimensions. If no dimensions are specified, total values across all dimension are returned. Valid values: See *Dimensions* on page 89 section for details. You can specify up to 3 dimensions at a time. Default: None Example: `dimensions=country,region`	No
`filters`	Restricts the set of results by specified filter values. **Note:** For up to 2 filters you can set a date range of up to 1 year (366 days). For 3 filters you can set a date range of up to 1 month (31 days). Valid values: See *Filters* on page 96 for valid filter names and boolean operations. • The value of `filter_by` must be URL-encoded. • The value of the actual filter must be enclosed in single quotation marks. Default: None Examples:	No

Parameter	Description	Required
	• Filter by the country Australia: `filters=country=='AU'` • Filter by mobile devices in country Colombia `filters=country=='CO',device_type=='mobile'`	
`time_segment`	Specify the time-based segment for dimension data. See the discussion of behavior in *About time_segment and Data Persistence* on page 89. This will sort blocks of data. **Note:** A week is defined as Monday - Sunday. Valid values: • `month` | `week` | `day` Default: None Example: `time_segment=day`	No
`sort`	List of comma-separated metrics to sort by. For multiple metrics, sorts by the metrics in the order the metrics are placed in the query. You can explicitly use as many sort metrics as you want (given that you have the metric), but default sorting has a limit of two metrics. Default: Sort by first two metrics (if present) in query order. Default value ordering is descending order. For ascending order, prefix a given metric with the + character. Examples: • Plays requested, displays, and video starts in descending order: `sort=plays_requested,displays,video_starts` • Video starts in ascending order: `sort=+video_starts` **Note:** We recommend that you do not use segment_watched or percentage_watched for sorting. Please use other metrics, like plays_requested, instead. There are different sort semantics for different metrics. For	No

Parameter	Description	Required
	example, segment_watched and percentage_watched use the string (of the entire array) for sorting and do not use a numeric sort. This means that if you sort by segment_watched, your results may not appear in numeric order. Your results may appear with the first result having a lower count of segment_watched than the second result (e.g. [99,…] , [999,...]). In contrast, plays_requested uses a numeric sort, which will provide you with clear results.	
`limit`	Limit the records returned in the response. Maximum limit: 10,000. Default: 50 Example: `limit=100`	No
`page`	Integer for pagination. Starts with 0. Default: 0 Example: For the second page: `page=1`	No

ABOUT TIME_SEGMENT AND DATA PERSISTENCE

The time range automatically expands to the minimal time range to fulfill the request. For example, specifying a start and end date for only today but with a `time_segment` of `week` causes the specified the start/end date to expand to cover the entire current week (in the provider's timezone). When there is no specified `time_segment`, the aggregate total for the specified dimensions is returned. When there is neither `time_segment` nor any specified dimension, the grand total for the time range is returned.

Dimensions

Here are the reference details for all predefined dimensions for the v3 Analytics *Reporting API*.

Dimensions are specified with the *`dimensions=`* query string parameter. Some considerations:

- You can filter by up to 3 dimensions at a time.
- Multiple values must be comma-separated with no spaces.
- The order of multiple is not important:
  - `dimensions=device_type,dma` is the same as `dimensions=dma,device_type`

General Syntax of Reporting GET

The base syntax of the route and query string is as follows. For ease of reading, the single-line request has been split across several lines.

```
[GET] /v3/analytics/reports/?
```


```
report_type=type
&dimensions=dimensions
&metrics=metrics
&filters=filter_type=='filter_value'
&start_date=date
&end_date=date
&other_parms
&api_key=your_api_key
```

- The **required** query string parameters (shown in bold) are *report_type*, start_date, and api_key.
- If no dimensions are specified, total values across all dimension are returned.
- You can specify up to 3 dimensions at a time.
- If no metrics are specified, all metrics are returned.

Note: At this time, the only valid value for *report_type* is performance.

General Syntax of Reporting Long Queries POST

For queries with query parameters that would exceed the HTTP GET specification limit of 230 characters , please use a POST request. Some browsers and http clients may support more than 230 characters, but we will not provide official support for queries that violate the HTTP GET specification. For POST requests, pass a JSON object in the request body instead of the query string parameters.

```
[POST] /v3/analytics/reports

{
    "report_type":"type",
    "dimensions":"dimensions",
    "metrics":"metrics",
    "filters":"(filter_type==\"filter_value\")",
    "start_date":"date",
    "end_date":"date",
    "other_parms":"other_param_value",
    "api_key":"your_api_key"
}
```

Name	Meaning	Additional Info
asset	Video or other assets, including both Ooyala asset IDs (embed codes or content IDs) and external identifiers.	To filter the asset dimension by label, include the filter: filters=label_id=='yourLabelId' Response also includes: { embed_code,name,status,type } • name is the title of the video. • embed_code is the video identifier (content ID or asset ID). • status is the video's status in Ooyala Backlot: live or paused.

Name	Meaning	Additional Info
country	Country code. See *Country and Location Codes* on page 100.	Response includes full country name. In the Analytics UI, part of ***Geography***.
region	Geographic region. Our geographic reporting uses the definition of region provided by *Quova*. You cannot specify region alone; you must also specify its country.	In the Analytics UI, part of ***Geography***.
dma	Defined marketing area (a US-centric concept not prominent in other parts of the world), specified by *DMA ID*.	Response includes DMA ID and a descriptive name. In the Analytics UI, part of ***Geography***.
state	State name. For example, if you drill down by Geo on the United States, you will see California, Arizona, etc. Our geographic reporting uses the definition of state provided by *Quova* (state is "the first-level administrative division" of a country). You cannot specify state alone; you must also specify its country.	In the Analytics UI, part of ***Geography***.
device_type	Device type.	See possible values in *Codes for Platforms, Devices, and Operating Systems* on page 99.
domain	Internet domain (fully qualified domain name).	In the Analytics UI, part of ***Traffic Source***.
os	Operating system of the user's device. Must be exact string; partial matches not supported.	See possible values in *Codes for Platforms, Devices, and Operating Systems* on page 99.
browser	User's web browser. **Note:** The browser dimension applies only when device_type is desktop.	See possible values in *Codes for Platforms, Devices, and Operating Systems* on page 99.
pcode	Ooyala-supplied provider ID.	
player_id	The identifiers of Ooyala or other named video players used during playback.	**Note:** The Ooyala mobile SDKs (Android and iOS) currently do not send a player_id, so the player_id for the mobile SDKs is attributed as "unknown".

Metrics

Here are the complete reference details for all metrics for the v3 Analytics *Reporting API*.

Note: All metrics, except for player_loads, are associated with the asset pcode. The player_loads metric is associated with the player pcode.

Metrics are specified with the *metrics=* query string parameter. Multiple values must be comma-separated with no spaces.

Note: You can retrieve the entire set of metrics by not including `metrics` in your query. To retrieve a specific metric, specify it as in the following examples:

- `metrics=plays_requested`
- `metrics=uniq_plays_requested`
- `metrics=displays,uniq_displays`

General Syntax of Reporting GET

The base syntax of the route and query string is as follows. For ease of reading, the single-line request has been split across several lines.

```
[GET] /v3/analytics/reports/?
            report_type=type
            &dimensions=dimensions
            &metrics=metrics
            &filters=filter_type=='filter_value'
            &start_date=date
            &end_date=date
            &other_parms
            &api_key=your_api_key
```

- The **required** query string parameters (shown in bold) are *report_type*, `start_date`, and api_key.
- If no dimensions are specified, total values across all dimension are returned.
- You can specify up to 3 dimensions at a time.
- If no metrics are specified, all metrics are returned.

Note: At this time, the only valid value for *report_type* is `performance`.

General Syntax of Reporting Long Queries POST

For queries with query parameters that would exceed the HTTP GET specification limit of 230 characters , please use a POST request. Some browsers and http clients may support more than 230 characters, but we will not provide official support for queries that violate the HTTP GET specification. For POST requests, pass a JSON object in the request body instead of the query string parameters.

```
[POST] /v3/analytics/reports

{
    "report_type":"type",
    "dimensions":"dimensions",
    "metrics":"metrics",
    "filters":"(filter_type==\"filter_value\")",
    "start_date":"date",
    "end_date":"date",
    "other_parms":"other_param_value",
    "api_key":"your_api_key"
}
```

UNIQUE USERS

You may want to measure the number of unique users. To measure unique users, use the `uniq_plays_requested` metric described below. The Ooyala mobile SDKs generate and store a random unique ID which is application-specific. In other environments, a unique user is identified by local

storage or cookie. If a user clears their browser cache, that user/device's ID will get regenerated next time they watch video. The generated IDs are completely random and don't include any user-identifiable information. When such information is not available for a user, a new unique identifier will be created for that user. We de-duplicate when calculating the number of unique users over time. For example, day 1 has users A, B, C; day 2 has users B, E., then when you pick date range = day 1 and day 2, then total unique users = 4 (A, B, C, E), Daily Avg. Unique Users = (3+2) / 2 = 3 (2.5 is converted to the closed integer).

VIDEO AND ENGAGEMENT-RELATED METRICS

Metric Name	Meaning	Related Event, UI Label
`autoplays`	Number of automatic plays (without user intervention).	
`plays_requested`	The number of times that the "Play" button is triggered either manually or automatically for any content (including video and ad content). This metric does not include the replay event. **Note:** We have improved our calculation of this metric. You may see the number of plays requested from your HTML5 and SDK players increase after 11/11/2014 due to this implementation.	***Plays Requested*** on Business Intelligence page
`displays`	The number of times that the particular video content is loaded and displayed within the player before it gets played (displays are related to each individual asset). Each time an embed code is changed this event gets triggered.	***Displays*** on Business Intelligence page
`replays`	Number of times the replay button is clicked.	
`video_starts`	Number of times the actual non-ad video content begins playback. Thus, not fired until the completion of pre-roll ads and actual asset playback has begun. If a user clicks on the “Play” button, a pre-roll ad starts playing, and the user drops off before the actual video content starts playing, this event won’t be triggered. **Note:** We have improved our calculation of this metric. You may see the number of video starts from your HTML5 and SDK players decrease after 11/11/2014 due to this implementation.	***Video Starts*** on Business Intelligence page
`time_watched`	Amount of time the user spent playing back the video asset, in milliseconds.	
`playthrough_25`	Number of video plays for the selected assets that reached the state of 25% completion of the video asset. No matter how many times the user rewinds within the same view/play session, once the "state" is	

Metric Name	Meaning	Related Event, UI Label
	reached it won't be marked/counted again. For example, within the same video viewing session (i.e. a user didn't refresh the page or reload the player), if a user rewinds and watches through the 25% quartile multiple times, Playthrough 25% will only count once. Applies only to video-on-demand (VOD), not live streams.	
`playthrough_50`	Number of video plays for the selected assets that reached the state of 50% completion of the video asset. No matter how many times the user rewinds within the same view/play session, once the "state" is reached it won't be marked/counted again. For example, within the same video viewing session (i.e. a user didn't refresh the page or reload the player), if a user rewinds and watches through the 50% quartile multiple times, Playthrough 50% will only count once. Applies only to video-on-demand (VOD), not live streams.	
`playthrough_75`	Number of video plays for the selected assets that reached the state of 75% completion of the video asset. No matter how many times the user rewinds within the same view/play session, once the "state" is reached it won't be marked/counted again. For example, within the same video viewing session (i.e. a user didn't refresh the page or reload the player), if a user rewinds and watches through the 75% quartile multiple times, Playthrough 75% will only count once Applies only to video-on-demand (VOD), not live streams.	
`playthrough_100`	Number of video plays for the selected assets that reached the state of 100% completion of the video asset. No matter how many times the user rewinds within the same view/play session, once the "state" is reached it won't be marked/counted again. For example, within the same video viewing session (i.e. a user didn't refresh the page or reload the player), if a user rewinds and watches through the 100% quartile multiple times, Playthrough 100% will only count once. Applies only to video-on-demand (VOD), not live streams.	

Metric Name	Meaning	Related Event, UI Label
avg_time_watched_per_video	Average video viewing time, calculated as sum(time watched)/video_starts, converted to HH:MM:SS format. **Note:** We have improved our calculation of video starts. You may see avg_time_watched_per_video increase after 11/11/2014 due to this implementation.	
play_conversion_rate	Ratio of plays requested events to displays events. **Note:** We have improved our calculation of plays requested. You may see the play_conversion_rate metric from your HTML5 and SDK players increase after 11/11/2014 due to this implementation.	
video_conversion_rate	Ratio of video starts events to plays requested events. **Note:** We have improved our calculation of video starts and plays requested. You may see the number of video starts from your HTML5 and SDK players change after 11/11/2014 due to this implementation.	
player_loads	Triggered each time the player loads on the page. The player_loads metric is associated with the pcode for the player.	

OTHER METRICS AND THEIR RELATED REPORTS

Metric Name	Meaning	Used in Report Type
uniq_plays_requested	Number of unique users that trigger the "play" button (manually or automatically) for any content (see ***plays_requested***). **Note:** uniq_plays_requested is a measure of Unique Users (uniq_users). **Note:** We have improved our calculation of this metric. You may see the number of plays requested from your HTML5 and SDK players increase after 11/11/2014 due to this implementation.	Performance
uniq_displays	Number of unique users that see the video loaded and displayed within the player before it gets played (displays are related to each individual asset). Each time an embed code is changed the display event gets triggered (see ***displays***).	Performance
uniq_video_starts	Number of unique users that see playback of actual non-ad video content (see ***video_starts***).	Performance

Metric Name	Meaning	Used in Report Type
	Note: We have improved our calculation of this metric. You may see the number of video starts from your HTML5 and SDK players decrease after 11/11/2014 due to this implementation.	
`embeds_copied`	Number of times the embed code has been copied.	Performance
`digg`	Number of times the "Digg" button has been pressed.	Performance
`emails_sent`	Number of times the "share via email" button is pressed.	Performance
`facebook`	Number of times the Facebook "Like" button has been pressed.	Performance
`twitter`	Number times the Twitter "tweet" button is pressed.	Performance
`urls_copied`	Number of times the "Copy URL" button has been pressed.	Performance
`segment_watched`	The number of times each segment of a piece of video content is watched. 1 segment is defined as 2.5% of video length. Please note if a user rewinds and watches the same segment N times, Segments Watched for that segment will count as N times.	Performance
`percentage_watched`	Measures the percentage of a video (in its entirety) that ever got watched in increments of 2.5%. This metric will not increase if a user rewinds and re-watches a segment in the same video session.	Performance

Filters

Here are the complete reference details for filters for the v3 Analytics *Reporting API*.

The filter names are essentially the same as dimension names. Filters are specified with the *`filters=`* query string parameter. You can further refine the results with boolean operators. Surround filter values with single quotations (e.g. filters=asset=='asset_id'). Filter values are case-sensitive.

For examples of how to use filters, see *What Filters Can I Use?* on page 74.

Date Range Limitation

Note: For up to 2 filters you can set a date range of up to 1 year (366 days). For 3 filters you can set a date range of up to 1 month (31 days).

For queries with query parameters that would exceed the HTTP GET specification limit of 230 characters , please use a *POST request*.

General Syntax of Reporting GET

The base syntax of the route and query string is as follows. For ease of reading, the single-line request has been split across several lines.

```
[GET] /v3/analytics/reports/?
            report_type=type
            &dimensions=dimensions
            &metrics=metrics
            &filters=filter_type=='filter_value'
```


```
&start_date=date
&end_date=date
&other_parms
&api_key=your_api_key
```

- The **required** query string parameters (shown in bold) are *report_type*, start_date, and api_key.
- If no dimensions are specified, total values across all dimension are returned.
- You can specify up to 3 dimensions at a time.
- If no metrics are specified, all metrics are returned.

Note: At this time, the only valid value for *report_type* is performance.

General Syntax of Reporting Long Queries POST

For queries with query parameters that would exceed the HTTP GET specification limit of 230 characters , please use a POST request. Some browsers and http clients may support more than 230 characters, but we will not provide official support for queries that violate the HTTP GET specification. For POST requests, pass a JSON object in the request body instead of the query string parameters.

```
[POST] /v3/analytics/reports

{
    "report_type":"type",
    "dimensions":"dimensions",
    "metrics":"metrics",
    "filters":"(filter_type==\"filter_value\")",
    "start_date":"date",
    "end_date":"date",
    "other_parms":"other_param_value",
    "api_key":"your_api_key"
}
```

Filter Type	Meaning	Valid Values
asset	Filter data by asset.	Any asset id associated with the provider or subproviders. The asset id can be embed_code or external_ids that you have entered in Backlot.
country	Filter data by country.	Country code. See *Country and Location Codes* on page 100.
region	Filter data by geographic region. Our geographic reporting uses the definition of region provided by *Quova*.	Any region string. You cannot specify region alone; you must also specify its country.
dma	Filter by DMA (defined marketing area, a US-centric concept not prominent in other parts of the world).	Any provider-defined DMA.
state	Filter data by state (i.e. the states within a country). In our geographic reporting, state is "the first-level administrative division" of a country, as defined by *Quova*. For example, you can filter so you only see data for plays in California.	Any state string. You cannot specify state alone; you must also specify its country.

Filter Type	Meaning	Valid Values
`device_type`	Filter by device type.	See possible values in *Codes for Platforms, Devices, and Operating Systems* on page 99.
`domain`	Filter by internet domain.	Any domain string.
`os`	Filter data by operating system.	See possible values in *Codes for Platforms, Devices, and Operating Systems* on page 99.
`browser`	Filter data by browser. **Note:** The `browser` dimension applies only when `device_type` is `desktop`.	See possible values in *Codes for Platforms, Devices, and Operating Systems* on page 99.
`pcode`	Filter data by the provider's pcode (Ooyala-supplied provider ID). For details, see *Your API Credentials* on page 9.	Any pcode that is the current account (the provider executing the API call) or one of its sub-accounts.
`player_id`	Filter data by player id (of an Ooyala player).	
`label_id`	Filter data by label using the label_id. Keep in mind the following points about working with labels in Ooyala Analytics: • Correlations among labels and associated videos are not pre-aggregated or pre-summarized (rolled up) but are resolved at the time of an API request. • You must have created an explicit association between a video and all of the individual labels you want. The Ooyala system does not aggregate or summarize based on a label's apparent hierarchy, only on explicit associations. For example, imagine two labels, the parent "Cycle World" and its child label "Sport Bikes". A video labeled with the child "Sport Bikes" is rolled up into the parent label "Cycle World" only if the video is also explicitly labeled "Cycle World".	You can use any defined label in Backlot. The result will contain only metrics for assets that are tagged with the label. You can filter by multiple labels at a time.

BOOLEAN OPERATORS; (AND) AND | (OR)

The formal syntax of a filter is shown below.

```
filters= filter [boolOp filter]*
```

Symbol	Expansion
filter	*dimension-filter* or *composite-filter*

Symbol	Expansion
dimension-filter	*dimension op value* Where: • *dimension* is a dimension name listed in *Dimensions* on page 89 or a custom dimension name. • *op* is == (equal to) or != (not equal to). • *value* is a single-quoted dimension value, like os!='android' (data with operating system Android).
composite-filter	(*filter* [*boolOp filter*]*) Where: • *filter* is defined above. • *boolOp* is ; (AND) or | (OR). The boolean operator and second filter are optional and can be repeated without limit.

Grouping with Parentheses

Use parentheses to logically group components of the filter value. For an example, see *What Filters Can I Use?* on page 74.

Codes for Platforms, Devices, and Operating Systems

A device type is a category of the physical device on which a video was accessed. Platforms are a concatenation of a device type, an OS, and a browser. These tables shows the input you will use to query device or OS type.

DEVICE TYPES

deviceType
desktop
mobile
settop
tablet
unknown.devicetype

OPERATING SYSTEMS

osType
android
bada os
firefox os
ios
linux
mac
meego

osType
rim os
rim tablet os
sunos
unknown.os
webos
windows
windows rt
xrossmediabar

BROWSER TYPES

browserType
android_sdk
android webkit
aol explorer
blackberry
chrome
chromium
eventmachine
fennec
firefox
ie
ios_sdk
msie
netfront
opera
safari
seamonkey
ucweb
unknown.browser
webkit/webos

Country and Location Codes

These country codes are used in a variety of Ooyala features.

The following table lists supported location codes:

Alpha-2 Code	English Short Name	Country Name Returned by Analytics API (If Different Than English Short Name)
AD	Andorra	
AE	United Arab Emirates	
AF	Afghanistan	
AG	Antigua and Barbuda	
AI	Anguilla	
AL	Albania	
AM	Armenia	
AN	Netherlands Antilles	
AO	Angola	
AQ	Antarctica	
AR	Argentina	
AS	American Samoa	
AT	Austria	
AU	Australia	
AW	Aruba	
AZ	Azerbaijan	
BA	Bosnia and Herzegovina	
BB	Barbados	
BD	Bangladesh	
BE	Belgium	
BF	Burkina Faso	
BG	Bulgaria	
BH	Bahrain	
BI	Burundi	
BJ	Benin	
BM	Bermuda	
BN	Brunei Darussalam	
BO	Bolivia	
BR	Brazil	Brasil
BS	Bahamas	
BT	Bhutan	
BV	Bouvet Island	
BW	Botswana	
BY	Belarus	
BZ	Belize	

Alpha-2 Code	English Short Name	Country Name Returned by Analytics API (If Different Than English Short Name)
CA	Canada	
CC	Cocos (Keeling) Islands	
CD	Congo, The Democratic Republic of the	Democratic Republic of the Congo
CF	Central African Republic	
CG	Congo	
CH	Switzerland	
CI	Cote D'Ivoire	Cote d'Ivoire
CK	Cook Islands	
CL	Chile	
CM	Cameroon	
CN	China	People's Republic of China
CO	Colombia	
CR	Costa Rica	
CU	Cuba	
CV	Cape Verde	
CX	Christmas Island	
CY	Cyprus	
CZ	Czech Republic	
DE	Germany	
DJ	Djibouti	
DK	Denmark	
DM	Dominica	
DO	Dominican Republic	
DZ	Algeria	
EC	Ecuador	
EE	Estonia	
EG	Egypt	
EH	Western Sahara	
ER	Eritrea	
ES	Spain	
ET	Ethiopia	
FI	Finland	
FJ	Fiji	
FK	Falkland Islands (Malvinas)	

Alpha-2 Code	English Short Name	Country Name Returned by Analytics API (If Different Than English Short Name)
FM	Micronesia, Federated States of	Micronesia
FO	Faroe Islands	
FR	France	
FX	France, Metropolitan	
GA	Gabon	
GB	United Kingdom	
GD	Grenada	
GE	Georgia	
GF	French Guiana	
GG	Gurnesy	
GH	Ghana	
GI	Gibraltar	
GL	Greenland	
GM	Gambia	
GN	Guinea	
GP	Guadeloupe	
GQ	Equatorial Guinea	
GR	Greece	
GS	South Georgia and the South Sandwich Islands	
GT	Guatemala	
GU	Guam	
GW	Guinea-Bissau	
GY	Guyana	
HK	Hong Kong	
HM	Heard Island and McDonald Islands	
HN	Honduras	
HR	Croatia	
HT	Haiti	
HU	Hungary	
ID	Indonesia	
IE	Ireland	Republic of Ireland
IL	Israel	
IM	Isle of Man	
IN	India	

Alpha-2 Code	English Short Name	Country Name Returned by Analytics API (If Different Than English Short Name)
IO	British Indian Ocean Territory	
IQ	Iraq	
IR	Iran, Islamic Republic of	Iran
IS	Iceland	
IT	Italy	
JE	Jersey	
JM	Jamaica	
JO	Jordan	
JP	Japan	
KE	Kenya	
KG	Kyrgyzstan	
KH	Cambodia	
KI	Kiribati	
KM	Comoros	
KN	Saint Kitts and Nevis	
KP	Korea, Democratic People's Republic of	North Korea, Korea (North)
KR	Korea, Republic of	South Korea, Korea (South)
KW	Kuwait	
KY	Cayman Islands	
KZ	Kazakhstan	
LA	Lao People's Democratic Republic	Laos
LB	Lebanon	
LC	Saint Lucia	
LI	Liechtenstein	
LK	Sri Lanka	
LR	Liberia	
LS	Lesotho	
LT	Lithuania	
LU	Luxembourg	
LV	Latvia	
LY	Libyan Arab Jamahiriya	Libya
MA	Morocco	
MC	Monaco	
MD	Moldova, Republic of	Moldova

Alpha-2 Code	English Short Name	Country Name Returned by Analytics API (If Different Than English Short Name)
MG	Madagascar	
MH	Marshall Islands	
MK	Macedonia	
ML	Mali	
MM	Myanmar	
MN	Mongolia	
MO	Macau	
MP	Northern Mariana Islands	
MQ	Martinique	
MR	Mauritania	
MS	Montserrat	
MT	Malta	
MU	Mauritius	
MV	Maldives	
MW	Malawi	
MX	Mexico	
MY	Malaysia	
MZ	Mozambique	
NA	Namibia	
NC	New Caledonia	
NE	Niger	
NF	Norfolk Island	
NG	Nigeria	
NI	Nicaragua	
NL	Netherlands	
NO	Norway	
NP	Nepal	
NR	Nauru	
NU	Niue	
NZ	New Zealand	
OM	Oman	
PA	Panama	
PE	Peru	
PF	French Polynesia	
PG	Papua New Guinea	

Alpha-2 Code	English Short Name	Country Name Returned by Analytics API (If Different Than English Short Name)
PH	Philippines	
PK	Pakistan	
PL	Poland	
PM	Saint Pierre and Miquelon	
PN	Pitcairn Islands	
PR	Puerto Rico	
PS	Palestinian Territory, Occupied	Occupied Palestinian Territory
PT	Portugal	
PW	Palau	
PY	Paraguay	
QA	Qatar	
RE	Reunion	
RO	Romania	
RU	Russian Federation	Russia
RW	Rwanda	
SA	Saudi Arabia	
SB	Solomon Islands	
SC	Seychelles	
SD	Sudan	
SE	Sweden	
SG	Singapore	
SH	Saint Helena	
SI	Slovenia	
SJ	Svalbard and Jan Mayen	
SK	Slovakia	Slovak Republic
SL	Sierra Leone	
SM	San Marino	
SN	Senegal	
SO	Somalia	
SR	Suriname	
ST	Sao Tome and Principe	
SV	El Salvador	
SY	Syrian Arab Republic	Syria
SZ	Swaziland	
TC	Turks and Caicos Islands	

Alpha-2 Code	English Short Name	Country Name Returned by Analytics API (If Different Than English Short Name)
TD	Chad	
TF	French Southern Territories	
TG	Togo	
TH	Thailand	
TJ	Tajikistan	
TK	Tokelau	
TM	Turkmenistan	
TN	Tunisia	
TO	Tonga	
TP	East Timor	
TR	Turkey	
TT	Trinidad and Tobago	
TV	Tuvalu	
TW	Taiwan	Taiwan, Republic of China Taiwan
TZ	Tanzania, United Republic of	Tanzania
UA	Ukraine	
UG	Uganda	
UM	United States Minor Outlying Islands	
US	United States	
UY	Uruguay	
UZ	Uzbekistan	
VA	Holy See (Vatican City State)	
VC	Saint Vincent and the Grenadines	
VE	Venezuela	
VG	Virgin Islands, British	Virgin Islands (British)
VI	Virgin Islands, U.S.	Virgin Islands (U.S.)
VN	Vietnam **Note:** The country name is Viet Nam in Backlot.	
VU	Vanuatu	
WF	Wallis and Futuna	Wallis and Futuna Islands
WS	Samoa	
YE	Yemen	
YT	Mayotte	

Alpha-2 Code	English Short Name	Country Name Returned by Analytics API (If Different Than English Short Name)
YU	Yugoslavia	
ZA	South Africa	
ZM	Zambia	
ZR	Zaire	
ZW	Zimbabwe	
A1	Anonymous Proxy	
A2	Satellite Provider	
O1	Other	

HTTP Response Codes and Messages

The top of all responses contain the following message format. Possible values for *status message* and *code* are described in the table below. Unless otherwise indicated, these codes pertain to the Ooyala IQ Reporting API. In the case of an error, the response includes the errors key, with an explanatory message.

```
{
  "status": "status message",
  "statusCode": code
  "errors" : "explanatory message"
}
```

Example:

```
 {
    "status": "Invalid parameter",
    "statusCode": "400",
    "errors": ["filter has invalid syntax: invalid character . at position
 4"]
}
```

HTTP Response code	General Meaning	Detailed Description
200	Success	The request was successful.
204 No Content	Successfully received	Returned by the Event Ping API. No response body is returned.
400	Bad request	The request is incorrect. For example, it does not contain required query string parameters.
401	Not authorized	Either the credentials (Ooyala-provided API key and secret) are missing or their values are not valid.
403	Insufficient permission	The credentials in the request do not have sufficient permission for data requested.
429	Quota exceeded	The requester's quota of API calls has been exceeded. Quotas are described at *API Rate Limiting*.

HTTP Response code	General Meaning	Detailed Description
500	System error	Possible problem in the Ooyala system.
503	Service Unavailable	There is a temporary service outage.

Migrating from v2 Analytics

Here are some considerations for when you migrate from v2 Analytics to v3 Analytics (Ooyala IQ).

v2/v3 Fundamental Difference: Multidimensional Analysis

In v3 Analytics (Ooyala IQ), data are stored and presented multi-dimensionally.

In v2 Analytics, the data are shown uni-dimensionally (that is, a single dimension). This means that in v2 you could investigate only one type of data at a time; for example, by domain, or by geography, or by device, and so on.

In v3 Analytics, however, the data are presented multi-dimensionally (that is, more than a single dimension). What this means is that you can do analyses of more than one set of data at a time. For example, instead of looking at data only by geography OR by domain type, in v3 Analytics you can look at data by geography AND by domain (the intersection between these two dimensions). You can apply up to 3 dimension filters at a time.

In v3 Analytics (Ooyala IQ), multidimensional analysis is supported in both the GUI and the REST-based APIs.

Data Migration

All customers will have reprocessed data starting from January 1, 2014 to query with Ooyala IQ.

You will still be able to use the v2 Analytics API for single dimensional queries on all of your historical data until March 31, 2016.

To save your historical data from Backlot as CSV using the Backlot UI:

1. Log in to *Backlot*.
2. Click the ANALYZE tab.
3. Set filters and dimensions so that your report contains the desired content.
4. Click **Save as CSV** next to the date selector in the upper right portion of the UI.

To save your historical data from Backlot as CSV using the Backlot API, please see *How to Export Your Data With the v2 Analytics API* on page 15.

How to Export Your Data With the v2 Analytics API

INTRODUCTION

The following information will guide you on retrieving and saving your analytics v2 data.

As we migrate customers from v2 Analytics to Ooyala IQ (v3 Analytics), you need to know the following:

- All customers will have access to reprocessed data starting from January 1, 2014.
- You will continue to have access to the old v2 Analytics API until March 31, 2016.

This means that if you need access to more than 1 year of historical analytics data, you need to export the data using the v2 APIs while they are still active.

OVERVIEW: THE OOYALA V2 ANALYTICS API

With the Ooyala v2 Analytics API you can easily create a report that will provide you with your analytics data. The results will be in JSON format. (JSON is a lightweight data-interchange format that is easy to read and write.)

To retrieve analytics results from a specific date range you simply need to define the type of result you need with the api call /v2/analytics/reports/, either in a terminal emulator such as Terminal on a Mac or in the Ooyala Scratchpad.

What is an API?

In computer programming. **API (Application Programming Interface)** is the name of a set of routines and protocols for software applications. An **API** expresses a software component in terms of its operations and results.

Where can I locate my API Credentials?

You can locate your API Key and Secret in the Backlot UI. Please use your API v2 credentials located in Backlot under the ACCOUNT>Developers tab.

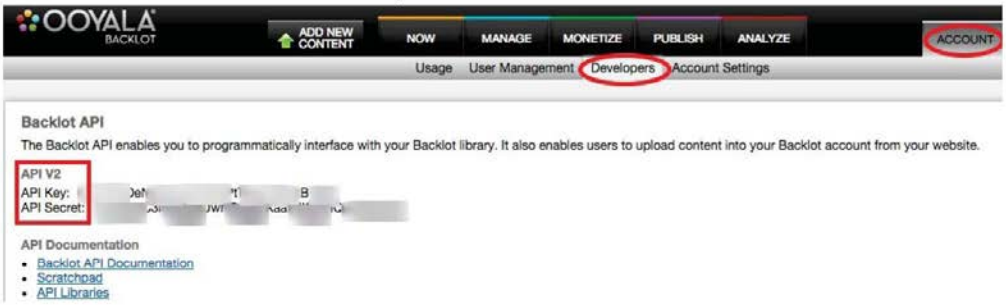


Figure 4: Finding Your API Keys

How Does the API Work?

An Ooyala API call requires 4 basic elements: API Key, API Secret, a Signature and an Expiration time.

API calls are made via *HTTP* methods. The GET API call is used to retrieve data without directly modifying it and allows you to get a typed JSON document response based on the id of the object.

What are the Available Results I can Retrieve with the Ooyala v2 Analytics API?

On the v2 Analytics API there are 4 key qualifiers that you need to identify in order to retrieve your results:

asset_id: This value is referred to by different names depending on where you look for it. In the Backlot API, asset_id is the identifier for a specific asset. asset_id has the same value as the content ID found in the Backlot UI that represents a piece of content. The value is the same for all scenarios. asset_id can be used if you want to retrieve results for a specific asset. Alternatively, you can retrieve results for your account, which would include all of your assets.

date_range: Defines the date range for this report. Analytics is based on dates. You can define the date following the format (YYYY-MM-DD) or you can define a date range with (YYYY-MM-DD...YYYY-MM-DD).

Report Request: Defines the type of report you want to retrieve. Valid values include *performance*, *sharing*, *engagement*, and *delivery*.

Dimension: Dimensions are common criteria that are used to aggregate data, such as the date when the user activity occurred or the country where the users were located. Every Dimension also has "**Drilldowns**", which allow you to filter your results by specific values for each dimension.

Common query string parameters and attributes can be found at *Common Attributes and Query String Parameters*.

HOW TO RETRIEVE YOUR V2 ANALYTICS DATA

What is the API Call Format?

The v2 Analytics API follows a specific order on the body of the API.

For example, the API call: `/v2/analytics/reports/account/performance/[:dimension/:drilldown]/:date_range`

Should appear like the following if you want to get a performance report from 2011-01-01 to 2014-01-01:

Figure 5: API Call For Performance Report

How Can I Use the Scratchpad to Save Reports?

The Scratchpad is a tool created by Ooyala that allows you to make API queries in your browser.

To retrieve an analytics report with the Scratchpad:

1. Go to *https://api.ooyala.com/docs/api_scratchpad?url=*.
2. Select "Your Account" in the Credentials section in the upper right corner of the page.
3. Enter your v2 API credentials (API Key and Secret) in the Credentials section in the upper right corner of the page.
4. In the Query field located on the left side of the page, enter your Analytics query. For example, if you would like to get the performance report from 2011 to 2014, copy and paste this query: `/v2/analytics/reports/account/performance/total/2011-01-01...2014-01-01`
5. Select GET.
6. Click Submit.

   Note: Your response appears in the response field.
7. If you prefer to see your results in a larger browser window, copy the API URL shown in green above the Submit button into your browser. In this case, the URL would be *https://api.ooyala.com/v2/analytics/reports/account/performance/total/2011-01-01...2014-01-01?api_key=yourApiKey&signature=yourSignature&expires=1418771221*.
8. Save the JSON by selecting File > Save As… in your browser. For information on how to convert JSON to CSV, see *Converting Analytics JSON to CSV* on page 128

Additional Query Examples

For more details on how to form queries in Scratchpad and for specific analytics report types, see:

- *Performance* query examples
- *Sharing* query examples
- *Engagement* query examples
- *Delivery* query examples

Note: To retrieve all data for a report type for your account, use the "total" query string parameter. You can find examples using "total" in each of the query example links mentioned above. "total" is used to retrieve all data for that particular report type for your account.

For example, the following query retrieves all performance data for the account over the date range 2011-08-01...2011-08-02.

```
[GET]/v2/analytics/reports/account/performance/total/2011-08-01...2011-08-02
```

How can I Create my own API Report?

You should only create your own API script if you are comfortable with the Ooyala API and have created scripts before or if you have the technical resources available who can modify the pre-made query to retrieve the data for you.

If you check the following snippet from our *sample code* that shows a terminal, you will be able to identify that we send the request of the API call using *cURL*. cURL makes http request where you can modify the parameters and the headers.

```
$apiSecret = "xxxxxx";
$apiKey = "xxxxxx";

$restMethod = 'GET';

$requestPath = '/v2/analytics/reports/account/performance/total/2011-01-01...2014-01-01';

$expDate = time() + 1209600;

$parameters = array( "expires" => $expDate, "api_key" => $apiKey, );

$content = "";

$signature = createSignature($apiSecret, $restMethod, $requestPath,  $parameters, $content);

$url = 'https://api.ooyala.com'.$requestPath.'?&api_key=txxxxxxxxxxx&expires='.$expDate.'&signature='.$signature;

echo $url;

$ch = curl_init($url);
curl_setopt ($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
curl_setopt ($ch, CURLOPT_HTTPHEADER, Array("Content-Type: application/json"));
curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "GET");
curl_setopt($ch, CURLOPT_POSTFIELDS, $content);
```

Your API V2 Credentials

HTTP Method

API Call

Expiration time

Signature

Curl request to the api url

Figure 6: Sample API Report

Note: For more script examples, please check our support site documentation at *Sample Code for Signing Requests* on page 173.

Potential Differences in the Data

There may be differences in the data in v3 Analytics (Ooyala IQ) when compared to data in v2 Analytics due to the following:

- Due to the re-processing of data to accommodate the multi-dimensional capabilities, the aggregation of the data may result in slightly different values than in v2 Analytics. We are also using a new robust architecture in Ooyala IQ which results in better error handling.
- Ooyala IQ uses Quova, a more accurate geo provider than in v2 Analytics.
- Ooyala IQ uses a more accurate device profiling provider than v2 Analytics, so the aggregation across devices and operating systems may differ. We are now using UADetector along with WURFL to optimize mobile and desktop coverage.
- The metrics are defined more accurately. Metrics with new definitions or bug fixes in calculation include:
    - **Playthrough**: v2 Analytics calculated playthrough on the server side based on the segments watched. There are 40 segments for each video. If the 10th segment got watched, Playthrough 25% was triggered; if the 30th segment got watched, Playthrough 75% was triggered. This behavior is problematic when people rewind and/or fast-forward. This calculation method did not always take deduplicating into account and did not report segments that users skipped, which at times led to playthrough 75% appearing higher than playthrough 25%, for example. The new analytics has a more accurate *definition of playthrough*, which is calculated on the client-side to indicate the furthest quartile point a user has reached within one viewing session. The new calculation deduplicates, reports for the segments that the user skipped, and has a more accurate measure of when the 100% mark is reached. This change in calculation provides more accurate metric data for playthrough. Data after August 1, 2014 is calculated using the new playthrough logic.

- **Uniques**: The calculation of unique metrics in v2 Analytics involved looking for cookies on the client side based on event timestamps. The *new calculation of uniques* is server-side and involves an algorithm called hyperloglog (HLL) and a guid that identifies browsers within a device. If the player does not see a guid a new guid is generated. This new calculation method reduces bugs, deduplicates, and is more accurate. You may see differences in your "unique" data, but the values should be similar, and any change is due to increased accuracy in "unique" calculation.
- **Plays Requested:** We have improved our calculation of this metric. You may see the number of plays requested from your HTML5 and SDK players increase after 11/11/2014 due to this implementation.
- **Video Starts:** We have improved our calculation of this metric. You may see the number of video starts from your HTML5 and SDK players decrease after 11/11/2014 due to this implementation.

Analytics Graphical User Interface (GUI) Differences

Here are general differences in the GUI.

1. The major difference between the GUIs for v2 Analytics and Ooyala IQ is that Ooyala IQ allows much richer investigation because of *multidimensional analysis*. In v2 Analytics you have multiple pages in the GUI where analysis has to be done, such as the geography page, the domain page, and others. In Ooyala IQ, you also have different pages to work with, but you should concentrate primarily on the **Business Intelligence** page, which gives you an at-a-glance view of all your dimensions.
2. UI reports

v2 Analytics	Ooyala IQ
The reports for dimensions (by device, by geography, and so on) across the top each have a separate page. You are essentially looking at a single dimension on each page.	The *Business Intelligence page* centralizes the investigation of all dimensions on a single page. When you filter by a dimension, the data for the other dimensions are constrained by that filter. Thus the reports are by nature multi-dimensional. For concrete examples, see *Business Use Case Examples of Analyses*, which illustrate how filtering works.
Performance	The *Analytics Dashboard* shows performance metrics, which you can also see on the *Business Intelligence* page.
Engagement	The *Business Intelligence page Performance tab* shows average time watched and other engagement-related metrics. The *Video Details page* gives a full view of the engagement metrics.
Sharing	No direct equivalent in GUI. Most users have sharing mechanisms on their own sites outside of the Ooyala Player, so be aware that the sharing data you see in the API is only the sharing that has happened through the Ooyala Player. Data are available via Ooyala IQ Reporting API.
External Publishing	No direct equivalent.

The reports in V2 were all contained on their own page:

In Ooyala IQ you can look at data for all of the dimensions at once:

3. The v2 Analytics `platforms` dimension has been replaced in Ooyala IQ with the following: *`device_type`*, *`os`*, and *`browser`*.
4. In Ooyala IQ you filter the report results by label using the Asset dimension box. *To filter the Business Intelligence page by label*, click the filter icon at the top right of the Asset dimension box and select "Filter by label". You can then select which label filter you want to apply to the entire Business Intelligence page.
5. The v2 Analytics dimension called **Domain** has been renamed in the Ooyala IQ GUI as **Traffic Source**. This represents the sites where an asset has been viewed. Use the Traffic Source dimension in lieu of the old Domain dimension.
6. Within the geography dimension panel in the Business Intelligence page you can filter your data by country. When you select a country you can filter even further by states or provinces. If states or provinces are not available for a country, "none" or "unknown" are shown.
7. To manage the size of the datacubes, we have removed the city, tag, and url dimensions.

v3 Analytics Application Programming Interface (API)

Here are the major differences in the programming interfaces between v2 Analytics and v3 Analytics (Ooyala IQ).

1. The major difference in the APIs between v2 Analytics and v3 Analytics is that v3 allows much richer investigation because of *multidimensional analysis*. This has certain ramifications on syntax, as described below.
2. The v2 Analytics API positionally expresses many options or filters directly on the route itself, often making the route difficult to parse. In addition, in v2 no more than a single dimension could be requested on the route. In the v3 Analytics API, however, *query string parameters (name/value pairs)* are used for greater clarity and up to 3 dimensions can be included in a request. For example, for the performance report:
   - In v2 Analytics: `/v2/analytics/reports/level/performance/total/date_range`
   - In v3 Analytics: `/v3/analytics/reports/?report_type=performance&start_date=beginning date`
3. In v3 Analytics, for queries with query parameters that would exceed the HTTP GET specification limit of 230 characters, please use a POST request. Some browsers and http clients may support more than 230 characters, but we will not provide official support for queries that violate the HTTP GET specification.
4. In general, in v2 Analytics, a date range to filter the data can be specified at the end of the route, which is confusing. Instead, in v3 Analytics, use the `start_date` query string parameter and (if desired) the `end_date` parameter.
5. In v2 Analytics, the `breakdown_by` parameter forces the response to show values by day, week, or month. In v3 Analytics there is no direct equivalent; all values are returned. You can limit the time segment returned with the `time_segment` parameter.
6. v3 Analytics features with no equivalents in v2: `filters=`, `metrics=`, `dimensions=`, and other *query string parameters*.
7. To manage the size of the datacubes, we have removed the city, tag, and url dimensions.
8. API reports. The following table correlates the general syntax of the v2 Analytics report types with their v3 Analytics general equivalents or near equivalents. This is not intended to be a thorough treatment of all syntactical possibilities, but a general view of the differences.

   Note: The engagement, sharing, and unique reports have been combined and included in the performance report.

Report Type	v2 Analytics Report Route	v3 Analytics Equivalent or Near Equivalent
Performance	`/v2/analytics/reports/level/performance/`	`/v3/analytics/reports/?report_type=performance`
Sharing	`/v2/analytics/reports/account/sharing/videos/`	`/v3/analytics/reports/?report_type=performance`
Engagement	`/v2/analytics/reports/asset/asset_id/engagement`	`/v3/analytics/reports/?report_type=performance`
Unique	No equivalent	`/v3/analytics/reports/?report_type=performance`
Delivery	`/v2/analytics/reports/level/delivery/`	No equivalent at this time.

9. Equivalences in query string parameters

v2 Analytics Query String Parameter	v3 Analytics Equivalent	Description
page_token	page	Specify desired page of next response
breakdown_by	time_segment	Categorize the data according to times
limit	limit	Limit the number of records in the response
order_by	sort	Sort the returned data

10. For even more examples of equivalences, see *API Requests: v2 Analytics and v3 Analytics Comparison* on page 76 in the Analytics Developer Guide.

Certain "Unique" Metrics Confusing in v2 Analytics

In v2 Analytics, certain "unique" metrics were unexpectedly returned in responses.

In the v2 Analytics API, you can request that the system return the "daily uniques" (unique viewership details) for a given date. Unfortunately, the API response included not just the "daily uniques" but sometimes unexpectedly the "weekly" and "monthly" uniques. This presented difficulty to the user, who had to determine exactly which metric was the one he wanted among those he had requested and those not expected in the response. These metrics are shown in the snippet of a v2 Analytics response shown below:

```
    .
    .
    .
"metrics": {
    "video": {
     "uniq_replays": {
      "monthly_uniqs": "40",
      "daily_uniqs": "45",
      "weekly_uniqs": "40"
     },
      .
      .
      .
```

In Ooyala IQ, this behavior has been corrected such that only a single "unique" metric is returned in the response. The single unique metric corresponds to the time frame specified on the request (daily, weekly, and so forth).

API Requests: v2 Analytics and v3 Analytics Comparison

These examples show some commonly used queries in v2 Analytics and their equivalents in v3 Analytics (Ooyala IQ).

For ease of reading:

- The v3 query parameter values are not URL-encoded.
- Variable values are prefixed with a $ sign.
- Some requests are broken across lines.
- "&metrics=metrics" is left out of the syntax because by default all metrics are returned. Define specific metrics to return in your reports with "&metrics=yourDesiredMetrics".

Examples here include the following:

- *Total Dimensions*

- *Total with Breakdown by Day*
- *Geo Dimensions* on page 120
- *Device and Platform Dimensions* on page 123

TOTAL DIMENSIONS

v2 URL Syntax	v3 Equivalent
`/v2/analytics/reports/account/performance/total/$date_range`	`/v3/analytics/reports/?report_type=performance&start_date=$start&end_date=$end&api_key=yourApiKey`

```
https://api.ooyala.com/v3/analytics/reports/?
report_type=performance&start_date=YYYY-MM-DD&end_date=YYYY-MM-
DD&api_key=yourApiKey&expires=yourExpiration&signature=yourSignature
```

Sample Ooyala IQ Response:


```
{
     "status_code":"200",
     "result_count":1,
     "results":[
      {
          "start_date":"YYYY-MM-DD",
          "end_date":"YYYY-MM-DD",
          "data":[
            {
                 "group":{
                     },
                 "metrics":{
                     "playthrough_50": "40000",
                     "uniq_displays": "3000000",
                     "video_starts": "4000000",
                     "displays": "5000000",
                     "percentage_watched":[
                        18,
                        14,
                        13,
                        13,
                        15,
                        20,
                        12,
                        14,
                        12,
                        11,
                        19,
                        18,
                        8,
                        13,
                        11,
                        18,
                        16,
                        10,
                        15,
                        17,
                        15,
                        16,
                        8,
```

```
            10,
            12,
            13,
            8,
            9,
            8,
            14,
            8,
            10,
            11,
            11,
            12,
            10,
            11,
            5,
            0,
            14
        ],
        "player_loads":5000000,
        "segment_watched":[
            18,
            14,
            13,
            13,
            15,
            20,
            12,
            14,
            12,
            11,
            19,
            18,
            8,
            13,
            11,
            18,
            16,
            10,
            15,
            17,
            15,
            16,
            8,
            10,
            12,
            13,
            8,
            9,
            8,
            14,
            8,
            10,
            11,
            11,
            12,
            10,
            11,
            5,
            0,
            14
        ],
        "urls_copied": "3",
        "plays_requested": "6500",
        "embeds_copied": "2",
```


```
                    "facebook": "3",
                    "replays": "145",
                    "uniq_video_starts": "2000000",
                    "autoplays": "0",
                    "emails_sent": "0",
                    "digg": "0",
                    "playthrough_25": "50000",
                    "playthrough_75": "40000",
                    "twitter": "0",
                    "uniq_plays_requested": "60000",
                    "playthrough_100": "30000",
                    "time_watched": "7500000000"

                }
            }
        ]
    }
  ],
    "total_count":1,
    "status":"OK"
}
```

Note: There is no data in the group because there's no dimension specified, and the plays requested, displays is grand total for all the metrics across the time frame

TOTAL WITH BREAKDOWN BY DAY

v2 URL Syntax	v3 Equivalent
`/v2/analytics/reports/account/performance/total/$date_range?breakdown_by=day`	`/v3/analytics/reports/?report_type=performance&time_segment=day&start_date=$start&end_date=$end&api_key=yourApiKey`

```
https://api.ooyala.com/v3/analytics/reports/?
report_type=performance&time_segment=day&start_date=YYYY-MM-DD&end_date=YYYY-
MM-DD&api_key=yourApiKey&expires=yourExpiration&signature=yourSignature
```

Sample Ooyala IQ Response:

```
{
   "status_code":"200",
   "result_count":3,
   "results":[
      {
         "start_date":"YYYY-MM-DD",
         "end_date":"YYYY-MM-DD",
         "data":[
            {
               "group":{

               },
               "metrics":{
                 "playthrough_50":100,
                 "uniq_displays":2500,
                 "video_starts":3000,
                 ...
                 "time_watched":240000
```


```
            }
               }
            ]
         },
      {
         "start_date":"YYYY-MM-DD",
         "end_date":"YYYY-MM-DD",
         "data":[
            {
               "group":{

               },
               "metrics":{
                 "playthrough_50":100,
                 "uniq_displays":2500,
                 "video_starts":3000,
                 ...
                 "time_watched":240000
            }
            }
         ]
      },
{
   "start_date":"YYYY-MM-DD",
   "end_date":"YYYY-MM-DD",
   "data":[
      {
         "group":{

         },
         "metrics":{
            "playthrough_50":100,
            "uniq_displays":2500,
            "video_starts":3000,
            ...
            "time_watched":240000
         }
      }
   ]
}
],
"status":"OK",
"total_count":3
}
```

This returns metric totals broken down by day.

GEO DIMENSIONS

v2 URL Syntax	v3 Equivalent
`/v2/analytics/reports/asset/$asset_id/performance/countries/$country/regions/$region/$date_range`	`/v3/analytics/reports/?report_type=performance&dimensions=asset,country,region&filters=asset=='asset_id',country=='country',r•&start_date=$start&end_date=$end&api_key=yourApiKey`

```
https://api.ooyala.com/v3/analytics/reports/?
report_type=performance&dimensions=asset,country,region&filters=asset=='asset_id',
```


```
country=='country',region=='region'&start_date=YYYY-MM-DD&end_date=YYYY-MM-DD
&api_key=yourApiKey&expires=yourExpiration&signature=yourSignature
```

Sample Ooyala IQ Response:


```
{
     "status_code":"200",
     "result_count":1,
     "results":[
      {
          "start_date":"YYYY-MM-DD",
          "end_date":"YYYY-MM-DD",
          "data":[
            {
                 "group":{
                     "asset":"asset_id",
                     "country":"Country Code",
                     "countryName":"Country Name",
                     "region":"Region",
                     "name":"Asset Name",
                     "status":"live",
                     "type":"video"
                     },
                 "metrics":{
                     "playthrough_50": "40000",
                     "uniq_displays": "3000000",
                     "video_starts": "4000000",
                     "displays": "5000000",
                     "percentage_watched":[
                        18,
                        14,
                        13,
                        13,
                        15,
                        20,
                        12,
                        14,
                        12,
                        11,
                        19,
                        18,
                        8,
                        13,
                        11,
                        18,
                        16,
                        10,
                        15,
                        17,
                        15,
                        16,
                        8,
                        10,
                        12,
                        13,
                        8,
                        9,
                        8,
                        14,
                        8,
                        10,
                        11,
                        11,
```


```
            12,
            10,
            11,
            5,
            0,
            14
        ],
        "player_loads":5000000,
        "segment_watched":[
            18,
            14,
            13,
            13,
            15,
            20,
            12,
            14,
            12,
            11,
            19,
            18,
            8,
            13,
            11,
            18,
            16,
            10,
            15,
            17,
            15,
            16,
            8,
            10,
            12,
            13,
            8,
            9,
            8,
            14,
            8,
            10,
            11,
            11,
            12,
            10,
            11,
            5,
            0,
            14
        ],
        "urls_copied": "3",
        "plays_requested": "6500",
        "embeds_copied": "2",
        "facebook": "3",
        "replays": "145",
        "uniq_video_starts": "2000000",
        "autoplays": "0",
        "emails_sent": "0",
        "digg": "0",
        "playthrough_25": "50000",
        "playthrough_75": "40000",
        "twitter": "0",
        "uniq_plays_requested": "60000",
        "playthrough_100": "30000",
```


```
                    "time_watched": "7500000000"

                }
            }
        ]
    }
  ],
    "total_count":1,
    "status":"OK"
}
```

DEVICE AND PLATFORM DIMENSIONS

v2 URL Syntax	v3 Equivalent
/v2/analytics/reports/asset/ $asset_id/ performance/ device_types/$device_type/ platforms/$platform/ $date_range	/v3/analytics/reports/? report_type=performance &dimensions=asset,device_type &filters=((device_type=='$device_type')) AND ((asset_id=='$asset_id')) &start_date=$start &end_date=$end&api_key=yourApiKey

```
https://api.ooyala.com/v3/analytics/reports/?
report_type=performance&dimensions=asset,device_type&filters=
%28%28device_type%3D%3D%27device_type%27%29%29+AND+%28%28asset_id%3D
%3D%27$asset_id%27%29%29 &start_date=YYYY-MM-DD&end_date=YYYY-MM-DD
&api_key=yourApiKey&expires=yourExpiration&signature=yourSignature
```

Sample Ooyala IQ Response:

```
{
     "status_code":"200",
     "result_count":1,
     "results":[
      {
           "start_date":"YYYY-MM-DD",
           "end_date":"YYYY-MM-DD",
           "data":[
             {
                  "group":{
                      "asset":"asset_id",
                      "name":"Asset Name",
                      "status":"live",
                      "type":"video",
                      "device_type":"device_type"
                      },
                  "metrics":{
                      "playthrough_50": "40000",
                      "uniq_displays": "3000000",
                      "video_starts": "4000000",
                      "displays": "5000000",
                      "percentage_watched":[
                         18,
                         14,
                         13,
                         13,
```


```
            15,
            20,
            12,
            14,
            12,
            11,
            19,
            18,
            8,
            13,
            11,
            18,
            16,
            10,
            15,
            17,
            15,
            16,
            8,
            10,
            12,
            13,
            8,
            9,
            8,
            14,
            8,
            10,
            11,
            11,
            12,
            10,
            11,
            5,
            0,
            14
        ],
        "player_loads":5000000,
        "segment_watched":[
            18,
            14,
            13,
            13,
            15,
            20,
            12,
            14,
            12,
            11,
            19,
            18,
            8,
            13,
            11,
            18,
            16,
            10,
            15,
            17,
            15,
            16,
            8,
            10,
            12,
```


```
                        13,
                        8,
                        9,
                        8,
                        14,
                        8,
                        10,
                        11,
                        11,
                        12,
                        10,
                        11,
                        5,
                        0,
                        14
                    ],
                    "urls_copied": "3",
                    "plays_requested": "6500",
                    "embeds_copied": "2",
                    "facebook": "3",
                    "replays": "145",
                    "uniq_video_starts": "2000000",
                    "autoplays": "0",
                    "emails_sent": "0",
                    "digg": "0",
                    "playthrough_25": "50000",
                    "playthrough_75": "40000",
                    "twitter": "0",
                    "uniq_plays_requested": "60000",
                    "playthrough_100": "30000",
                    "time_watched": "7500000000"

                }
            }
        ]
    }
],
    "total_count":1,
    "status":"OK"
}
```

Analytics Glossary

For definitions of other terms, see also the following reference material in the Ooyala IQ Developer Guide:

- *Dimensions* on page 89
- *Metrics* on page 91

Term	Definition
asset	Ooyala term for content of various kinds, typically videos.
cardinality	The maximum number of possible values of a dimension. For example, a dimension named "color" might have three possible values: red, green, and gold. This dimension has a cardinality of three.
completion	Amount (usually a percentage) of assets actually watched. See also *playthrough (%)*. Applies only to video-on-demand, not live assets.

Term	Definition
conversion rate	In Ooyala v2 Analytics terminology, the change from a "display" event to a "play" event; that is, how many times the video was actually played, not simply displayed. (Note that in common industry understanding, "conversion rate" can mean many different things, such as conversion of a user from viewing an advertisement to actually making a purchase.) In Ooyala IQ there are two conversion rates; see *conversion rate, play* and *conversion rate, video*.
conversion rate, play	Ratio of plays requested to display events. Videos that are displayed on high traffic pages, but are not played, will have low conversion rates.
conversion rate, video	Ratio of start events to plays requested events. The Video Conversion Rate is useful for any publisher that provides some form of "bumpers" before the video starts or runs pre-roll ads to track the abandonment rate of consumers that quit before the requested video begins.
derived metric	A metric that is calculated from other, primary metrics.
device	*Dimension* to record the type of device the user operates to plays your asset. This is derived from the `USER_AGENT` web server environment variable.
dimension	An aspect or attribute of a video. All videos have the predefined dimensions described in *Dimensions* on page 89 in the Analytics Developer Guide.
displays	A count of the number of display events for a given asset. A display event is triggered each time a new video asset is loaded by the player.
DMA	*Dimension* to record Designated Marketing Area.
domain	*Dimension* to record the fully qualified domain name (FQDN).
engagement	A word used to classify metrics related to how much time people spent watching a video. See the "playthrough" metrics described in *Metrics* on page 91 in the Analytics Developer Guide.
event	A defined occurrence of a phenomenon in the "life of an asset", such as display, playRequested, or playStarted.
filter	"Constrain", decrease the amount of data by excluding some metrics to focus on specific aspects of the data. Noun: an instance of such filtering.
geography	*Dimension* to record the geographic location. Calculated by the system based on the IP address of the user's device. Geography actually consists of three separate dimensions: *country*, *region*, and *DMA*.
identifier	The unique value that identifies any given object, such as a video. Sometimes called a "key."
kpi	Key performance indicator.
metric	A measurement.
location	An old term from Analytics v2, now called *"Geography"*.
os	The operating system (OS) dimension of devices.

Term	Definition
player loads	The number of times a video player has been loaded on a page or device.
players	Video playing software with a defined name. All Ooyala players have defined names. Also, the *dimension* to record this.
plays requested	The number of times the "play" button is triggered either manually or automatically. A count of the number of `playRequested` events for a given asset (this could be playing an ad or the actual video content). Plays requested does not include replays.
playthrough (%)	Percentages of completion of playing assets, 25%, 50%, 75%. Applies only to video-on-demand, not live assets. This is a measure of the furthest point a user has reached during one video view session. There will only ever be one count of each playthrough % per video view session per user (for example, even if the user rewinds and re-watches the first quarter of the video three times during the same video view session, there will only be one count of 25% playthrough). See the "playthrough" metrics described in *Metrics* on page 91 in the Analytics Developer Guide.
slice-and-dice	(verb) To filter, to constrain, to limit the analyzed data by constraining it in various ways.
summarize	(verb) To calculate totals of lower order items in "buckets" or defined groups. Sometimes called "roll up". For example, the count of all devices of type "iOS" is the "iOS rollup."
total hours watched	Sum of the time all videos were watched.
traffic source	Recorded in the *domain* dimension, the site that referred or redirected the user's player to your asset. Derived from the *HTTP_REFERER* web server environment variable. Note that this is not the site the user "came from" to reach the location where the player is. This is the site where the player is embedded.
trending	Most popular videos are measured as videos with the most momentum during the time window. Momentum is calculated using a combination of time watched and the hourly increase of a video's performance against its baseline.
user, unique user	In many cases, a synonym for *plays_requested*. Unique users are calculated in the following ways: • **Ooyala mobile SDK for iOS:** The Ooyala mobile SDK for iOS generates and store a random unique ID which is application-specific. The unique ID is generated in the "OOClientID" class and is stored in the "standardUserDefaults" object and is valid until the application is deleted. This unique ID cannot be erased or reset by the end user without deleting the app. The application developer can store a different ID than the generated ID by erasing the existing ID [OOClientID resetID] and setting a new ID [OOClientID setID:New_ID]. • **Ooyala mobile SDK for Android:** The Ooyala mobile SDK for Android generates and store a random unique ID which is application-specific. The unique ID is generated in the "OOClientID" class and is stored in the "SharedPreferences" file and is valid until the application is deleted. This unique ID cannot be erased or reset by the end user without deleting

Term	Definition
	the app. The application developer can store a different ID by erasing the existing ID [ClientID.resetID(context)] and setting a new ID [ClientID.resetID(NEW_ID)]. • **All other environments (HTML5, Flash, Chromecast):** In other environments, a unique user is identified by local storage or cookie. To generate the GUID, Flash players use the timestamp of when the GUID is generated and append random data to it. The string is then converted to base64. To generate the GUID, HTML5 players use the current time, browser information, and random data and hash it and convert it to base64. Within the same browser on desktop, once a GUID is set by one platform it is used for both platforms for the user. If a user clears their browser cache, that user/device's ID will get regenerated next time they watch video. Incognito modes will track a user for a single session, but once the browser is closed the GUID is erased.
video starts	The number of times users started watching actual video content (non-ad content). Recorded via the playStarted event.

V2 ANALYTICS

Ooyala has the basic analytics feature of retrieving data with the v2 Analytics API.

Discussed here are some uses of this data and using custom analytics.

Converting Analytics JSON to CSV

With the Backlot UI you can download data in comma-separated value (CSV) format. Here is a general approach for doing this programmatically.

All responses by the Analytics API are in the form of JavaScript Object Notation (JSON). Much useful analysis can be done with spreadsheet programs, which can in general read CSV data.

Because the Backlot API is language-neutral, we do not focus on any particular language, relying on JavaScript to illustrate the conversion. The Internet has many free converter tools for many different languages.

Below is a sample JavaScript program (based somewhat on ideas from json.org) that converts the JSON in an array (named `json3`) to CSV. In actual practice, instead of defining an array, you might want to save the JSON responses to files and read the file, or change the script below to read from some other data source. Likewise, this script opens a new window to prompt to download the CSV data; you might want to change this behavior.

```
<script src="scripts/json.js" type="text/javascript"></script>
<script type="text/javascript">
var json3 = { "d": "[{\"Id\":1,\"UserName\":\"Sam Smith\"},{\"Id\":2,
\"UserName\":\"Fred Frankly\"},{\"Id\":1,\"UserName\":\"Zachary Zupers
\"}]" }

DownloadJSON2CSV(json3.d);

function DownloadJSON2CSV(objArray)
{
```


```
  var array = typeof objArray != 'object' ? JSON.parse(objArray) :
objArray;

  var str = '';

  for (var i = 0; i < array.length; i++) {
      var line = '';

      for (var index in array[i]) {
           line += array[i][index] + ',';
      }

      // Here is an example where you would wrap the values in double
quotes
      // for (var index in array[i]) {
      //    line += '"' + array[i][index] + '",';
      // }

      line.slice(0,line.Length-1);

      str += line + '\r\n';
  }
  // Might want to change this output
  window.open( "data:text/csv;charset=utf-8," + escape(str))
}
</script>
```

1. Make the API requests necessary to retrieve the data you want.

   Be sure that you are dealing with data that is amenable to conversion and that you are including similar dimensions in the data, regardless of type.
2. Convert the data to CSV by using the desired converter program, either a variant of the JavaScript shown above or for your own preferred language.

How to Export Your Data With the v2 Analytics API

INTRODUCTION

The following information will guide you on retrieving and saving your analytics v2 data.

As we migrate customers from v2 Analytics to Ooyala IQ (v3 Analytics), you need to know the following:

- All customers will have access to reprocessed data starting from January 1, 2014.
- You will continue to have access to the old v2 Analytics API until March 31, 2016.

This means that if you need access to more than 1 year of historical analytics data, you need to export the data using the v2 APIs while they are still active.

OVERVIEW: THE OOYALA V2 ANALYTICS API

With the Ooyala v2 Analytics API you can easily create a report that will provide you with your analytics data. The results will be in JSON format. (JSON is a lightweight data-interchange format that is easy to read and write.)

To retrieve analytics results from a specific date range you simply need to define the type of result you need with the api call /v2/analytics/reports/, either in a terminal emulator such as Terminal on a Mac or in the Ooyala Scratchpad.

What is an API?

In computer programming. **API (Application Programming Interface)** is the name of a set of routines and protocols for software applications. An **API** expresses a software component in terms of its operations and results.

Where can I locate my API Credentials?

You can locate your API Key and Secret in the Backlot UI. Please use your API v2 credentials located in Backlot under the ACCOUNT>Developers tab.

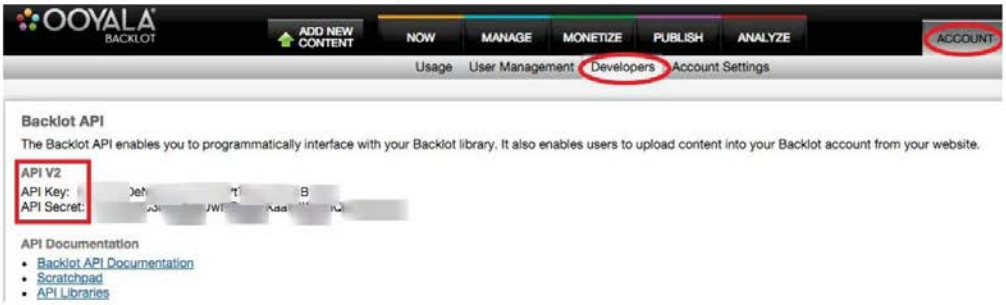


Figure 7: Finding Your API Keys

How Does the API Work?

An Ooyala API call requires 4 basic elements: API Key, API Secret, a Signature and an Expiration time.

API calls are made via *HTTP* methods. The GET API call is used to retrieve data without directly modifying it and allows you to get a typed JSON document response based on the id of the object.

What are the Available Results I can Retrieve with the Ooyala v2 Analytics API?

On the v2 Analytics API there are 4 key qualifiers that you need to identify in order to retrieve your results:

asset_id: This value is referred to by different names depending on where you look for it. In the Backlot API, asset_id is the identifier for a specific asset. asset_id has the same value as the content ID found in the Backlot UI that represents a piece of content. The value is the same for all scenarios. asset_id can be used if you want to retrieve results for a specific asset. Alternatively, you can retrieve results for your account, which would include all of your assets.

date_range: Defines the date range for this report. Analytics is based on dates. You can define the date following the format (YYYY-MM-DD) or you can define a date range with (YYYY-MM-DD...YYYY-MM-DD).

Report Request: Defines the type of report you want to retrieve. Valid values include *performance*, *sharing*, *engagement*, and *delivery*.

Dimension: Dimensions are common criteria that are used to aggregate data, such as the date when the user activity occurred or the country where the users were located. Every Dimension also has "**Drilldowns**", which allow you to filter your results by specific values for each dimension.

Common query string parameters and attributes can be found at *Common Attributes and Query String Parameters*.

HOW TO RETRIEVE YOUR V2 ANALYTICS DATA

What is the API Call Format?

The v2 Analytics API follows a specific order on the body of the API.

For example, the API call: `/v2/analytics/reports/account/performance/[:dimension/:drilldown]/:date_range`

Should appear like the following if you want to get a performance report from 2011-01-01 to 2014-01-01:

```
With Values:
/v2/analytics/reports/account/performance/total/2011-01-01...2014-01-01
API  Type of Report  Report Level  Report Request  Dimension  Date Range
```

Figure 8: API Call For Performance Report

How Can I Use the Scratchpad to Save Reports?

The Scratchpad is a tool created by Ooyala that allows you to make API queries in your browser.

To retrieve an analytics report with the Scratchpad:

1. Go to *https://api.ooyala.com/docs/api_scratchpad?url=*.
2. Select "Your Account" in the Credentials section in the upper right corner of the page.
3. Enter your v2 API credentials (API Key and Secret) in the Credentials section in the upper right corner of the page.
4. In the Query field located on the left side of the page, enter your Analytics query. For example, if you would like to get the performance report from 2011 to 2014, copy and paste this query: `/v2/analytics/reports/account/performance/total/2011-01-01...2014-01-01`
5. Select GET.
6. Click Submit.

   Note: Your response appears in the response field.
7. If you prefer to see your results in a larger browser window, copy the API URL shown in green above the Submit button into your browser. In this case, the URL would be *https://api.ooyala.com/v2/analytics/reports/account/performance/total/2011-01-01...2014-01-01?api_key=yourApiKey&signature=yourSignature&expires=1418771221*.
8. Save the JSON by selecting File > Save As… in your browser. For information on how to convert JSON to CSV, see *Converting Analytics JSON to CSV* on page 128

Additional Query Examples

For more details on how to form queries in Scratchpad and for specific analytics report types, see:

- *Performance* query examples
- *Sharing* query examples
- *Engagement* query examples
- *Delivery* query examples

Note: To retrieve all data for a report type for your account, use the "total" query string parameter. You can find examples using "total" in each of the query example links mentioned above. "total" is used to retrieve all data for that particular report type for your account.

For example, the following query retrieves all performance data for the account over the date range 2011-08-01...2011-08-02.

```
[GET]/v2/analytics/reports/account/performance/total/2011-08-01...2011-08-02
```

How can I Create my own API Report?

You should only create your own API script if you are comfortable with the Ooyala API and have created scripts before or if you have the technical resources available who can modify the pre-made query to retrieve the data for you.

If you check the following snippet from our *sample code* that shows a terminal, you will be able to identify that we send the request of the API call using *cURL*. cURL makes http request where you can modify the parameters and the headers.

```
$apiSecret = "xxxxxx";
$apiKey = "xxxxxx";

$restMethod = 'GET';

$requestPath = '/v2/analytics/reports/account/performance/total/2011-01-01...2014-01-01';

$expDate = time() + 1209600;

$parameters = array( "expires" => $expDate, "api_key" => $apiKey, );

$content = "";

$signature = createSignature($apiSecret, $restMethod, $requestPath,  $parameters, $content);

$url = 'https://api.ooyala.com'.$requestPath.'?&api_key=txxxxxxxxxxx&expires='.$expDate.'&signature='.$signature;

echo $url;

$ch = curl_init($url);
curl_setopt ($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
curl_setopt ($ch, CURLOPT_HTTPHEADER, Array("Content-Type: application/json"));
curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "GET");
curl_setopt($ch, CURLOPT_POSTFIELDS, $content);
```

Your API V2 Credentials

HTTP Method

API Call

Expiration time

Signature

Curl request to the api url

Figure 9: Sample API Report

Note: For more script examples, please check our support site documentation at *Sample Code for Signing Requests* on page 173.

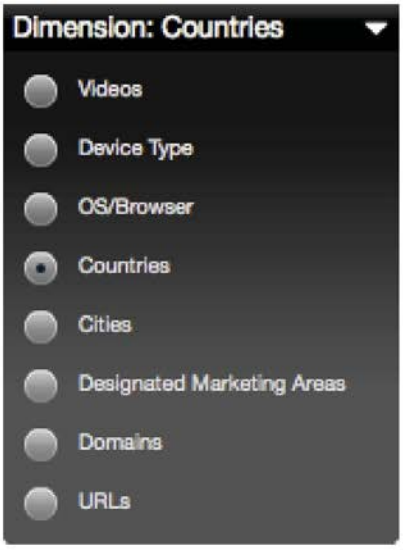
Getting the Data Underneath the Analytics Displays

The Backlot UI displays your analytics data in graphs and charts.

Here some the various graphical displays of analytics data on the **ANALYZE** page on Backlot UI are matched to the API requests that return the data underlying the statistical graphics.

PERFORMANCE: PLAYS BY COUNTRY

In Backlot UI you can several different view of performance of your assets. For instance, with the selection list on the left, you can view performance by country, as shown here.

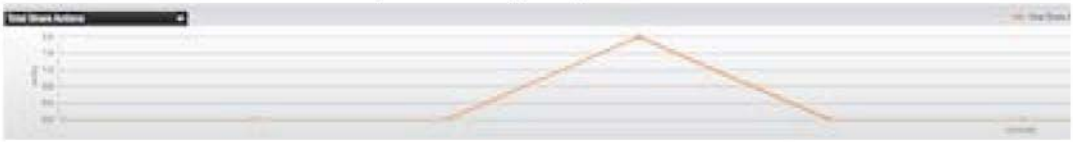


The call to get the underlying data for this display for all videos:

```
[GET]/v2/analytics/reports/account/performance/
countries/2012-01-01...2012-11-11
```

SHARES BY DATE

Similarly, you can see trends in "sharing" (posting or sharing the video on social networks) by looking at the timeline view. The data below represent a single day.



The call to get the underlying data for this display for an entire year:

```
[GET]/v2/analytics/reports/account/sharing/videos/2012-01-01...2012-12-31
```

Device Type Mapping

Within analytics, information is collected about different platforms used to access your content.

Platforms are a combination of device type, operating system, and browser displayed in the following format:

```
device_type:operating_system-browser
```

For example, a desktop computer running Windows and a Chrome browser would be returned as: `desktop:windows-chrome` and an Android tablet would be be returned as `tablet:android-unknown.browser`.

The following table lists supported device types:

Device Type	Backlot Name
Mobile	mobile
Tablet	tablet
Desktop/Laptop	desktop
Set-top	settop
Unknown Device Type	unknown.devicetype

The following table lists supported operating systems:

Operating System	Backlot Name
Mobile	
iOS (iPhone)	iphone
iOS (iPod)	ipod
Android	android
Blackberry	blackberry
PalmOS	palm
Windows Mobile	windowsmob
PlayStation Portable	psp
Other mobile OS	other
Tablet	
iOS (iPad)	iPad
Android	android
Other OS	unknown.os
Desktop	
Windows	windows
Mac	mac
Linux	linux
Unix (BSD)	bsd
Unix (Sun)	sunos
Other OS	unknown.os
Set-Top	
Nintendo Wii	wii
PlayStation	playstation
Boxee	boxee
Roku	roku
GoogleTV	google_tv
Other OS	unknown.os

The following table lists supported browsers:

Browser	Backlot Name
Chrome	chrome
Firefox	firefox
Internet Explorer	ie
Safari	safari
Opera	opera
Other Browser	unknown.browser

Custom Analytics

As users consume your content, Ooyala collects detailed information such as how long they watched, the amount of data delivered, conversion rates, source domains, sharing, and so on. Although this is a substantial amount of analytics, you might have information that is unavailable to Ooyala.

Data that is unavailable to Ooyala might include the age and gender of the users, whether they are free or paying users, language, political leanings, and so on.

Custom analytics enables you to inject custom information that you can use to create custom reports. To use custom analytics, you dynamically specify one or more tags in the client code at time of embed. When a user accesses the asset, the tags are automatically recorded along with the analytics data.

To use custom analytics, it must be enabled for your account. Once enabled, you set it up through the Backlot API, embed client code that records the custom analytics, and view the output through the Backlot API or Backlot UI.

Note: For information about adding custom analytics to your account, contact Sales, your Customer Success Manager, or Technical Support.

Working with Custom Analytics

There are many types of user information that you might want to record, which are not available to Ooyala. This section provides an example of how to configure Backlot to record the gender of each viewer.

To configure custom analytics, there essentially two steps:

1. Creating the custom tags with the Analytics REST API
2. Modifying your player creation code to send the custom tags when a player is created

This example deals with three custom tags. Every tag has an optional display name and can be managed via the APIs. When viewing the custom analytics in the Backlot the following format is presented: [display name] (Tag ID: [tag])

For example:

- standard (Tag ID: standard)
- custom widget (Tag ID: custom)
- toolbar (Tag ID: toolbar)

1. The following example creates a “standard”, “custom” and “toolbar” tags using Ooyala REST API.

```
[PUT]/v2/analytics/tags/standard{
   "display_name":"Standard"
}
[PUT]/v2/analytics/tags/custom{
   "display_name":"Custom"
}
[PUT]/v2/analytics/tags/toolbar{
   "display_name":"Toolbar"
}
```

2. In your JavaScript, assign tag identifiers:

```
var tags = new Array();

/* Assign the correct value depending on the widget type*/

if (widgetType == "standard") tags[0] = "standard";
if (widgetType == "custom") tags[0] = "custom";
if (widgetType == "toolbar") tags[0] = "toolbar";
```


3. Modify your play embeds to send the `tags` array when a player is created. The following is a basic example of a player V3 creation.

```
window.player = OO.Player.create(playerID, embedCode, {
"analytics":{
                        "tags": tags
                }
 });
```

Custom analytics are configured. You can begin viewing the results through the Backlot UI or Backlot API.

Recording Facebook Data within Custom Analytics

If your users log into your site through Facebook Connect, you have access to their basic information (demographics, education, work history, likes or preferences, and more).

The following is an example of the type of information you can obtain from Facebook Connect:

```
https://graph.facebook.com/btaylor
```

which returns a response similar to the following:

```
{
   "id": "220439",
   "name": "Bret Taylor",
   "first_name": "Bret",
   "last_name": "Taylor",
   "link": "http://www.facebook.com/btaylor",
   "username": "btaylor",
   "gender": "male",
   "locale": "en_US"
}
```

Note: For more information about how to use Facebook Connect, refer to the *Facebook Developers* site.

The following example describes how to create record a user's gender based from Facebook data using JavaScript.

1. Create a tag for each piece of information to record. In this example, you create a `male` and `female` tag.

   The following example creates a `male` tag:

   ```
   [PUT] /v2/analytics/tags/male{
      "display_name": "Male"
   }
   ```

   The following example creates a `female` tag:

   ```
   [PUT] /v2/analytics/tags/female{
      "display_name": "Female"
   }
   ```

   If the requests are successful, Backlot returns a 200 response.
2. Initialize FB.api and make sure the user is logged in. For more information, refer to the *Facebook JavaScript SDK*.
3. Embed the client code that records the male or female tag through the `playerAPICallback` call.

The following example code gets the gender of the user by making a call to the Facebook API and records the gender each time the `playerAPICallback` call is made.

```
var tags = new Array();

FB.api( "/me", function(response) {
 tags[0] = response.gender;
});

function playerAPICallback(playerId, eventName, eventParams) {
 if(eventName=="playerEmbedded"){
  document.getElementById(playerId).setModuleParams({
   "analytics":{
    "tags": tags
   }
  });
 }
}
```

4. Embed the client code that embeds the player, specifying the `OnCreate` embedded parameter with the name of your previously defined function `playerAPICallback`.

   The following is a basic example of a player V3 creation. The third argument is a hash of name/value pairs (detailed in the *Player API Reference*).

```
  window.player = OO.Player.create('playerwrapper','w3ZHc0Njr33Tdp-
RRcwfZMjaOrmzOP82', {
    onCreate: window.playerAPICallback,
    .
    .
    .
    }
.
.
.
```

You can begin viewing the results through the Backlot UI or Backlot API.

v2 Analytics API

Based on your account, different analytics packages are available.

The following are the currently offered Ooyala account packages:

- Custom Analytics Package—Custom Analytics
- Default Analytics Package—Totals, Videos, Engagement, Sharing
- Device Types Analytics Package—Device types, Platforms
- Domain Analytics Package—Domains, URLs
- Geo Analytics Package—Countries, Regions, Cities, DMAs

All packages are discussed here.

Please note that all our platform services are currently dependent on Ooyala players being the generator of analytics data.

GENERALIZED SYNTAX

With only one exception, all API requests for Analytics use the `GET` method, because you are only reading data, not writing it. (The one exception is Custom Analytics to create your own custom tags for tracking; see *Custom Analytics: Tags*.)

Thus, directly on the `GET` request, you specify various levels of detail, various types of data, or various constraints. On the route there are either *attributes* or *query string parameters* or both:

```
[GET]/v2/analytics/
reports/attributes_suchas_levels_geo_andsoforth[/
optional_attributes]?query_string_parms
```

As a result, the routes themselves can appear long or verbose but because of their verbosity, the meaning of the routes is clear.

In the syntax for the analytics routes, in the notation `[/optional_qualifiers]` the square brackets indicate that the attrubte is optional; they are not part of the real syntax.

MONETIZATION

Ooyala enables you to monetize your content through advertising and paywalls.

MONETIZATION AND THE API

The Backlot API enables you to associate ad sets with assets.

You can use the Backlot API for the following monetization features:

- Uploading ads
- Associating ad sets with assets

The following monetization features are not currently supported by the Backlot API:

- Configuring ad sources in ad sets
- Configuring paywalls

ADVERTISING

Ooyala supports several types of advertising: in-stream, linear in-stream, and overlay.

Some useful definitions:

In-stream Ads: An ad that is played before, during or after the main video content. Linear video ads (pre-, mid-, and post-roll ads) and non-linear ads (overlay ads) and companion ads are all considered in-stream ads.

- **Linear In-stream Ads**:Linear in-stream ads are video ads that play before the main content has started, during a designed ad spot in the middle of the video or after the main content has completed. Pre-, mid- and post-roll ads are all considered linear in-stream. It's important to note that the main video content is not playing while the video ads are playing.
- **Non-Linear In-stream Ads**: Non-linear in-stream ads run concurrently with the video content so the users see the ad while viewing the main video content. Overlay ads are considered non-linear in-stream ads.
    - **Overlay Ads**:Graphical or text ad that appears over the lower third of the main video content and is displayed while the video is playing.

You can assign linear ads directly to videos or assign videos to an ad set that contains a combination of linear ads and non-linear ads.

Working with Your Own Ad Assets, Ad Sets, and Ad Sources

This material includes how to create ad sets in Backlot and how to integrate with ad sources.

Ads can come from a variety of sources, such as from your own assets on Backlot or from other ad sources. The term *ad source* refers to any system that returns a valid ad response, including ad servers and ad managers, as well as ad networks and exchanges.

You can create as many ad sets as you like, but each player will work with only one ad set. You can assign any number of ad tags and ad sets programmatically on your webpage as long as each of them is assigned to different players on the same webpage. For an ad set based on your own assets (ad source **My Ads**), you can assign as many of your own videos as you like.

The general process for working with ad sets is as follows.

1. If you are working with your own ads, get your video file(s) in order to upload to Backlot.
2. If you are working with an ad source partner, get configuration details from them, such as identifying keys, codes, or passwords. You need these details to configure Backlot.
3. In Backlot, create an ad set for your own material or for each ad source partner, relying on the configuration details you obtained above.
4. Associate the ad set with an asset.
5. At run time, if desired, pass ad-source-specific parameters to the ad source, for targeting or other needs. For further information on ad parameters, see *Ads Embedded Parameter Overview*.

Note: For Ooyala-hosted ads (ad source set to "My Ads"), in-stream and mid-roll ads are not currently supported. Do not select "In-Stream" or "mid-roll" as an ad position for ad sets with Ad Source set to "My Ads."

Uploading a Video Advertisement

Before you can add a video advertisement to a video, you must upload the video advertisement.

To upload a video ad:

1. Log in to the *Backlot UI*.
   The Backlot UI opens.
2. Click **ADD CONTENT** and select **AD**.
3. Select the file to upload and click Open.
   The ad appears on the **MANAGE** page as Backlot begins processing the file. After Backlot completes, it will change the **STATUS** of the video to **LIVE**
4. Click the **Ad Details** tab and specify the **Default Click URL** when the ad is clicked. Additionally, you can specify the **Default Tracking Pixel URL**.

   The click URL is the webpage that users are directed to when they click the ad. The click URL is also used by ad sources to track click-through and the tracking pixel URL is used to track ad impressions.

Ad Sets

An ad set is a group of one or more advertisements (linear and/or non-linear) that you can assign to videos.

Ad sets make it easier to manage, organize, and change advertisements. If you add linear advertisement directly to videos, you have to manually change the ad for each video whenever you update or change the ad. With ad sets, you simply add or remove videos from the ad set. All videos that use the ad set are automatically updated.

The following example has an ad set that contains two alternating pre-roll ads.

Position	Ad	Frequency
Pre-roll	Premium Sofa Manufacturer	Show first ad after 0 videos. Play an ad every 2 videos.
Pre-roll	Premium Shoe Manufacturer	Show first ad after 1 videos. Play an ad every 2 videos.

The following example has an ad set that consists of two pre-roll video ads, two alternating mid-roll video ads that start after 65% of the video has played, three alternating post-roll video ads, and Google Adsense non-linear ads that start 10 seconds into the video.

Position	Ad	Frequency
Pre-roll	Premium Caffeinated Sports Drink's Currently Living Athlete #217	Show first ad after 0 videos. Play an ad every 1 video.
Pre-roll	Premium Shoe Manufacturer	Show first ad after 0 videos. Play an ad every 1 video.
In-stream	Mega Sports Hut	Show first ad after 0 videos. Play an ad every 2 videos. Play ad at 65%
In-stream	Gigantachain Sports Emporium	Show first ad after 1 videos. Play an ad every 2 videos. Play ad at 65%
Post-roll	Granny Sue's Homemade Energy Cookies	Show first ad after 0 videos. Play an ad every 3 videos.
Post-roll	Itchy's Sport Performance Cream	Show first ad after 1 videos. Play an ad every 3 videos.
Post-roll	Fray's Used Hang Gliders	Show first ad after 2 videos. Play an ad every 3 videos.
In-stream	Google Adsense	Show first ad after 0 videos. Play an ad every 1 video. Start after 10 seconds.

Using this example, you can change any of the ad tags at any time and the changes are made for all videos that use the ad set.

Note: If you specify more than one video ad in the same position, Backlot plays all video ads in that position.

Creating an Ad Set

You can create as many ad sets as you like, but you can only assign one ad set to a video.

Note: For Ooyala-hosted ads (ad source set to "My Ads"), in-stream and mid-roll ads are not currently supported. Do not select "In-Stream" or "mid-roll" as an ad position for ad sets with Ad Source set to "My Ads."

To create an ad set:

1. Use the `/v2/ad_sets` route with the details of an ad to include in the POST body.

   The following example creates an ad set and associates the 01M2F0MTr3SWBB9dfAPT5xEsg8Gpy asset from the user's own assets with it.

```
[POST]/v2/ad_sets{
   "name":"My new Ad Set",
   "default":true,
   "ads":[
      {
         "ad_source":"assets",
```


```
            "ad_type":"preroll",
            "embed_code":"01M2F0MTr3SWBB9dfAPT5xEsg8Gpy",
            "plays_before":0,
            "plays_between":1,
            "click_url":"http://www.myclickurl.com",
            "tracking_pixel_urls":[
               "http://www.mytrackingpixel.com"
            ]
         }
   ]
}
```

The Backlot UI responds similarly to the following.

```
{
   "name":"My new Ad Set",
   "id":"77506f70fc1e4b08805687980cc4a24d",
   "ads":[

   ]
}
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

2. Change the name of the ad set, using a PATCH with the `/v2/ad_sets` route and the identifier of the desired ad set.

```
[PATCH]/v2/ad_sets/77506f70fc1e4b08805687980cc4a24d{
   "name":"Ad Set #9",

}
```

The Backlot UI responds similarly to the following.

```
{
   "name":"Ad Set #9",
   "id":"77506f70fc1e4b08805687980cc4a24d",
}
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

Adding a Video Ad to an Ad Set

After you create an ad set, you can add as many videos as you like.

To add an ad to an ad set:

POST to the `/v2/ad_set` route with identifier of an ad set, the `/ads` qualifier, and the details of the video ad in the POST body.

The following example creates an ad set and associates the 01M2F0MTr3SWBB9dfAPT5xEsg8Gpy asset from the user's own assets with it.

```
[POST]/v2/ad_sets/77506f70fc1e4b08805687980cc4a24d/ads{
   "ads":[
      {
         "ad_source":"assets",
         "ad_type":"preroll",
```


```
            "embed_code":"01M2F0MTr3SWBB9dfAPT5xEsg8Gpy",
            "plays_before":0,
            "plays_between":1,
            "click_url":"http://www.myclickurl.com",
            "tracking_pixel_urls":[
               "http://www.mytrackingpixel.com"
            ]
         }
      ]
}
```

The Backlot UI responds similarly to the following.

```
{
   "name":"My new Ad Set",
   "id":"77506f70fc1e4b08805687980cc4a24d",
   "ads":[
      {
         "ad_source":"assets",
         "ad_type":"preroll",
         "embed_code":"01M2F0MTr3SWBB9dfAPT5xEsg8Gpy",
         "plays_before":0,
         "plays_between":1,
         "click_url":"http://www.myclickurl.com",
         "tracking_pixel_urls":[
            "http://www.mytrackingpixel.com"
         ]
      }
   ]
}
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

Creating Ad Sets for Integrating with Ad Sources

Instead of using your own assets, you can work with ad networks.

This section describes a general process for setting up ad sets specific to ad sources, including the following:

1. *Google IMA V3*
2. *FreeWheel*
3. *LiveRail*
4. *Adap.TV*
5. *Tremor Media*

Note: Although you can use the /v2/ad_sets route to create ad sets, including specifying the ad source name, you must use the Backlot UI to edit the configuration settings for the ad source.

Note: The information required differs from ad source to ad source. Always check your ad source's documentation for exact meaning of a field.

1. Make a `POST` request to `/v2/ad_sets`, as shown below.

```
[POST]/v2/ad_sets{
   "name":"My new Ad Set",
   "default":true
}
```


The Backlot API responds with a response similar to the following.

```
{
    "default":"true",
    "id":"ed4a4c6abdfc4ab8aa3e7b8a05f31ec6",
    "ads":[

   ],
    "name":"My new Ad Set"
}
```

2. Associate an ad from your own assets or from any number of ad sources to an ad set.

```
[POST]/v2/ad_sets/ed4a4c6abdfc4ab8aa3e7b8a05f31ec6/ads{
   "ad_source":"assets",
   "ad_type":"preroll",
   "embed_code":"01M2F0MTr3SWBB9dfAPT5xEsg8Gpy-jP",
   "plays_before":0,
   "plays_between":1,
   "click_url":"http://www.myclickurl.com",
   "tracking_pixel_urls":[
      "http://www.mytrackingpixel.com"
   ]
}
```

3. With the Backlot UI complete the necessary fields for the specific ad source, as detailed in the Backlot User Guide.

Insert Tracking Pixel URLs

How to insert tracking pixel URLs to track ad delivery.

Currently Backlot does not have the functionality to track video ads built-in. Generally third party ad platforms are tracking the delivery of particular Ad asset. You need to create your "pixel counter" logic in your server create the pixel and track it.

Note: For additional instructions on how to create a tracking pixel see: *http://www.ehow.com/how_12026748_create-internal-tracking-pixels.html*.

After your tracking pixel URL is complete you can add it to your Backlot Ads.

1. Select your Ad in the Manage Tab.
2. Click on the AD Details.
3. Insert you tracking pixel URL in the "Default Tracking Pixel URL" field.

Deleting an Ad from an Ad Set

You can delete an ad from an ad set at any time.

To delete an ad from an ad set:

Make a DELETE request to the `/v2/ad_sets/ad_set_id/ads/ad_id` route, as in the following example.

```
[DELETE]/v2/ad_sets/ed4a4c6abdfc4ab8aa3e7b8a05f31ec6/
ads/01M2F0MTr3SWBB9dfAPT5xEsg8Gpy-jP
```

Deleting an Ad Set

You can delete an ad set at any time. All settings for the ad set are deleted and cannot be recovered.

To delete an ad set:

Make a `DELETE` request to the `/v2/ad_sets/ad_set_id`, as in the following example.

```
[DELETE]/v2/ad_sets/ed4a4c6abdfc4ab8aa3e7b8a05f31ec6
```

PAYWALL INTEGRATION

Paywalls enable you to monetize your content by requiring payment before viewers can access it. Ooyala supports integration with third-party paywalls.

Integrate with the Tinypass Paywall

This topic provides an overview of the integration of the *Tinypass* paywall with Ooyala.

Note: The Backlot paywall is now EOL and will no longer be supported. Ooyala has partnered with Tinypass to provide a robust paywall solution.

ABOUT THE TINYPASS PAYWALL

Tinypass is an e-commerce platform that allows content creators and media owners to monetize their digital content quickly, powerfully, and effectively. The Tinypass paywall is a turnkey implementation with a simple workflow that provides a frictionless checkout experience and is easy to set up.

INTEGRATION FEATURES

When integrated with Ooyala, the Tinypass paywall:

- Provides an HTML5- and Adobe Flash-capable paywall that works in-browser and enhances multi-device support by allowing for easy viewing in desktop and mobile environments.
- Supports multiple, secure payment methods: major credit cards (American Express, Visa, MasterCard, Discover) and payment partners (PayPal, Amazon Payments, and Dwolla).
- Accepts payments in over 30 currencies, including bitcoin (prices quoted in USD).
- Ensures Level 1 DSS PCI Compliance so that cardholder data is encrypted and maintained on a secure network that is regularly tested and monitored.
- Provides monetization options (time-based, metering, minimum price, and price windowing).
- Allows users to create a Tinypass password after they have completed their purchase.
- Provides the Tinypass REST API, which allows you to expand options for contact access, pricing, upgrades, and payment.
- Provides downloadable data and reports powered by Google Analytics to monitor your transactions, revenue, net income, fees, and conversion rates.

INTEGRATION INSTRUCTIONS

Paywall integration requires a Tinypass account, which is easy to set up and integrate with your new or existing Ooyala account. To set up integration between Ooyala and the Tinypass paywall, refer to the *Ooyala / Tinypass integration instructions* hosted on the Tinypass web site. Once the accounts are connected, it is easy to import video information from the publisher's Backlot account and apply a simple paywall.

INTEGRATION TECHNICAL SUPPORT

For technical support of Ooyala-Tinypass integrations:

Contact	For
Publisher	End users contact the publisher with questions via their Tinypass dashboard.
Tinypass	First-line support regarding paywall, payment, and authentication issues, such as Tinypass account login, viewing initial paywall page, payment verification, and re-accessing videos.
Ooyala Technical Support	Post-transaction issues, including video playback and quality.

Integrate with the Cleeng Paywall

This topic provides an overview of the integration of the *Cleeng* paywall with Ooyala.

Note: The Backlot paywall is now EOL and will no longer be supported. Ooyala has partnered with Cleeng to provide a robust paywall solution.

ABOUT OOYALA-CLEENG INTEGRATION

The Ooyala-Cleeng paywall integration is a turnkey solution that helps you monetize your Live, TVOD, and SVOD content. Integrating Ooyala with the Cleeng paywall enables you to instantly start monetizing your content while providing your viewers with a fast, seamless checkout experience.

INTEGRATION FEATURES

When integrated with Ooyala, the Cleeng paywall provides the following key capabilities:

- Live, TVOD, and SVOD support
- Accepts payments in 12 currencies in over 170 countries
- Social login support
- Encryption and watermarking support
- Standard currencies and automatic currency conversion
- In-depth reporting
- Integration with the Ooyala Player Token
- Monetization options (pay-per-view, rental, bundles and pass, and subscriptions)
- Many payment methods (including Cleeng credits, major credit cards, PayPal, and mobile payments)
- Customization using the Cleeng API

INTEGRATION INSTRUCTIONS

To begin your integration, start by registering a Cleeng account at the *Cleeng registration page*. Once registered, this link will take you to your Cleeng dashboard page.

INTEGRATION TECHNICAL SUPPORT

For technical support of Ooyala-Cleeng integrations:

Contact	For
Publisher	End users contact the publisher with questions via their Cleeng dashboard.
Cleeng	First-line support regarding paywall, payment, and authentication issues, such as Cleeng account login, viewing initial paywall page, payment verification, and re-accessing videos.

Contact	For
Ooyala Technical Support	Post-transaction issues, including video playback and quality.

MONETIZING YOUR OOYALA CONTENT WITH DFP

You can synch your Ooyala-hosted content with DoubleClick for Publishers (DFP) via either the Backlot API or an MRSS feed so you can monetize it with Google IMA.

You can set up your DoubleClick for Publishers (*Google DFP*) account to transfer your Ooyala-hosted content, so you can monetize that content with Google IMA V3.

All work described here (except for setting ad rules, if desired; see below) is done in DFP itself and involves defining a new data source, which can be either via the Backlot API or an MRSS feed.

PREREQUISITES AND YOUR DECISION

The following are necessary:

- A working DFP account, as required by Google.
- For syndicating via the Backlot API, have your Backlot API key and secret handy to configure in DFP. For more details, see *Your API Credentials* on page 9.

Decide which approach you want: pull content via the Backlot API with your key and secret, or use MRSS. After you decide, follow the appropriate steps for the desired option described below.

CREATING A NEW SOURCE IN DFP

To transfer content, you need to define a new source in DFP. In DFP:

1. Click the **Video** tab, then select **Sources**.
2. After your **Content Sources** appear, click **New Source**.
3. Complete the required fields, and select the source **Type**. For MRSS, select the MRSS type. If you want to pull content via the Backlot API, select the Ooyala type.

AD RULES

Note: Cuepoints are no longer supported for Google IMA V3. You will now set ad positions through ad rules which are created in your DFP account.

Ad rules will define when ads are inserted, how long they should run, what ads are run, and the ad source. For matrices describing what ad rule ad positions are supported for different platforms and browsers, see *Google IMA Ads Integration*.

Note: Google IMA's AdRules feature supports ad-breaks without additional code. Do not use AdRules with other types of ads. We highly discourage you from mixing other adTagUrls with an AdRule adTag as it may lead to unexpected behavior. For example, we recommend against using an AdRule adTag along with a mid-roll ad position set via Backlot.

To enable ad rules for an ad:

1. Specify your ad rules in DFP. For information on implementing DFP ad rules, go to *https://support.google.com/dfp_premium/answer/2553686?hl=en*.
2. To enable your DFP ad rules to correctly render for your Google IMA V3 ad with the Backlot UI, click **MONETIZE** > > **Ad Sets**, and set the ad position to "ad rules" for the desired ad. For more information on Google IMA V3 ads in the Backlot UI, see *Ad Set Fields for Google IMA V3*.

OR

To enable your DFP ad rules to correctly render for your Google IMA V3 ad with the Backlot API, set "ad_type": "rules" for the desired ad. For more information see *Ad Sets*.

Note: Settings applied at the page level with the ad tag url will override Backlot settings. However, the position type (ad rule or non ad rule) must match on the page level and in Backlot for ads and ad rules to properly render.

SYNCH VIA BACKLOT API

DFP will read this metadata key and associate it with the cuepoints object on their CMS.

To syndicate content via the Backlot API, in DFP:

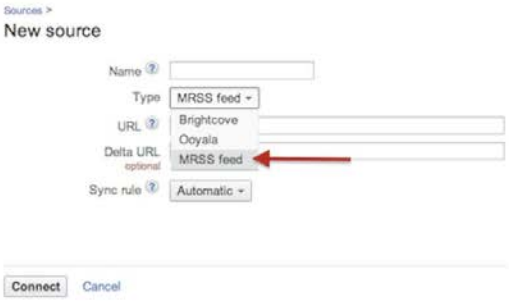
1. From the Sources, select **Ooyala**.
2. In the displayed input fields, enter your API key and secret.

DFP will pull each asset's title, duration, and custom metadata from the source.

SYNCH VIA MRSS FEED

To syndicate content via MRSS, in the source definition in DFP:

1. For **Type**, select MRSS feed.
2. Set the URL pointing to your MRSS feed with the content you wish to ingest. For details about the required MRSS elements, see *https://support.google.com/dfp_premium/answer/2475956?ref_topic=2475994*.



INITIATE THE SYNCH

After you set up your new source, click **Connect**, and DFP will start pulling your content.

METADATA TARGETING

For information on metadata targeting, please refer to Google DFP's developer documentation.

- *Content targeting overview*
- *Manage content metadata*

CONTENT PUBLISHING

Publishing lets you share videos and video metadata with multiple web properties and content networks.

MANUAL PUBLISHING

The most common way to publish content is to copy the HTML embed code to a web page.

You can manually embed all types of assets, including:

- Videos
- Audio assets
- Ads
- Live streams
- Channels
- Channel sets
- YouTube videos
- Remote assets

Manually Embedding an Asset

To make an asset available on a web page, you can manually embed it.

This section provides information on how to create a basic embed snippet. For advanced embed snippets, you can use automatically generated snippets from the Backlot UI or from the syndication feeds. For information on getting snippets from the Backlot UI, refer to the Backlot User Guide. For information on getting snippets from syndication feeds, see *Syndication* on page 150.

To embed an asset:

1. Create an embed snippet that specifies the width, height, and embed code.
   The following is an example of the minimum embed code for the asset with the `Q3bDhiMToizUftO` ID.

```
<script src="http://player.ooyala.com/player.js
?width=640
&height=480
&embedCode=Q3bDhiMToizUftO">
</script>
```

2. Paste the snippet into a web page.
3. Open the web page to verify the embed.

SYNDICATION

Syndication enables you to externally publish your content to third-party sites such as Boxee, YouTube, iTunes, and others.

Backlot makes publishing easy. Whether your video or video channel will be published on 5 websites or 500, the process takes only a few steps.

Syndication lets you:

- Create and dynamically update channel lineups
- Encourage viral video syndication through the Ooyala player
- Distribute videos to YouTube, iTunes and create your own Player MRSS feeds
- Create/participate in syndication networks to increase your audience reach and monetization opportunities
- Syndicate videos and video channels to content affiliates, while maintaining syndication authority over your content

Note: Due to storage restrictions, files larger than 5GB do not appear in syndications. For more details on this limit please contact your Customer Success Manager.

Backlot supports two types of syndication: YouTube syndication and feed-based syndication for all other third parties.

For YouTube syndication, Backlot automatically uploads the source files of videos directly to YouTube, which handles all transcoding and the generation of preview images.

For feed-based syndication, Ooyala hosts Media RSS (MRSS) files that contain information about each video.

Note: The syndication of audio assets is not supported.

Access keys can be used to create private URLs for a syndication's MRSS feed. With this you can give out unique feed URLs to third parties, and if needed, revoke their access to your MRSS feed without having to delete the feed. If you create access keys for your syndications you may need to get and share the access keys with your partners so they are able to use the syndication feed. See *the topics on access keys* to learn how to enable, create, and manage access keys.

The following is an example of an iTunes syndication with a single video:

```
<?xml version="1.0" encoding="UTF-8"?>
<rss xmlns:dcterms="http://purl.org/dc/terms/" version="2.0"
 xmlns:itunes="http://www.itunes.com/dtds/podcast-1.0.dtd"
 xmlns:media="http://search.yahoo.com/mrss/" xmlns:ext="http://ooyala.com/
syndication/ext/">
  <channel>
    <title></title>
    <description></description>
    <link>http://api.ooyala.com/v2/syndications/
d4fe34fe44ea4e648fd07d45094566d9/feed?pcode=Y4cG06BlqYMLIjPGyv1NbacSK-x3</
link>
    <itunes:author></itunes:author>
    <itunes:keywords></itunes:keywords>
    <itunes:subtitle></itunes:subtitle>
    <itunes:summary></itunes:summary>
    <item>
      <title>My Movie</title>
      <description>It's 20% longer than my last one!</description>
      <guid isPermaLink="false">BrdXVjMjrgtupU3M1hBAXx2Y_HpgQ4ho</guid>
      <link></link>
      <pubDate>Sun, 1 May 2011 20:36:40 +0000</pubDate>
      <media:title>My Movie</media:title>
      <media:description>It's 20% longer than my last one!</
media:description>
      <media:restriction type="country" relationship="allow">US EU</
media:restriction>
      <media:restriction type="uri" relationship="allow">http://
mywebsite.com</media:restriction>
      <media:category name="Gripping" id="c42b225963ad481b916d87ec77d7767b"
 full_name="/Gripping" parent_id="" scheme="http://www.ooyala.com"/>
      <media:thumbnail height="75" url="http://ak.c.ooyala.com/
BrdXVjMjrgtupU45uyBAXx2Y_HpgQ4ho/promo136839624" width="100"/>
```

```
      <dcterms:valid>start=2011-03-31T07:00:00+00:00;scheme=W3C-DTF</
dcterms:valid>
      <enclosure type="video/m4v" length="2161327" url="http://
api.ooyala.com/v2/syndications/d4fe34fe44ea4e648fd07d45094566d9/
redirect/7143813?expires=1320085707&amp;pcode=Y4cG06BlqYMLIjPGyv1NbacSK-
x3&amp;signature=jZS%2BSavYkEYDg8tuQfCn%2FzqMlzGW8itvMBdSOH2MXhc"/>
      <itunes:author></itunes:author>
      <itunes:duration>20</itunes:duration>
      <itunes:summary>I used a Steadicam!</itunes:summary>
    </item>
  </channel>
</rss>
```

When you provide this URL to iTunes, it uses this information to automatically ingest your content metadata and make the video available on iTunes.

To let Backlot know which videos to make available to which syndications, you configure it using labels. To do this, you create a syndication and associate it with a label. Then, anytime you associate a video, channel, or channel set with that label, it is automatically added to the syndication.

For example, if you want to syndicate a video podcast to Boxee, YouTube, and iTunes, you might set up three syndications and associate them with the *publish_podcast* label. Then, whenever you add the *publish_podcast* label to an asset, Backlot automatically adds it to Boxee, YouTube, and iTunes.

Creating a Syndication

After creating a syndication, you can associate it with labels; all videos and channels that use the one of the labels will automatically be syndicated using the settings.

To create a syndication:

1. Use the `/v2/syndications` route, specifying the name and type (advanced, boxee, google, iphone, iphone_abr, itunes, mp4, or roku).

   The following example creates the "My iPhone Feed" iPhone feed.

   ```
   [POST]/v2/syndications{
      "name":"My iPhone Feed",
      "type":"iphone"
   }
   ```

   Note: If you are creating a YouTube feed, you must also specify your YouTube `username` and `password`.

   Backlot returns a response similar to the following.

   ```
   {
      "name":"My iPhone Feed",
      "include_encodings":[

      ],
      "destination_url":"",
      "require_access_key":false,
      "title":"",
      "include_labels":[

      ],
      "asset_types":[
         "video",
         "ad"
      ],
   ```


```
   "syndication_url":"http://api.ooyala.com/syndications/6b6e72dd71bd4fa/
feed?pcode=Y4cG06BlqYMLIjPGyv1NbacSK-x3",
   "id":"6b6e72dd71bd4fa",
   "include_all_content":false,
   "type":"iphone",
   "description":""
}
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

2. For the feed to contain videos and other assets, you must add labels. Whenever you add a label to an asset that is associated with a feed, the asset is automatically added to the feed.

   The following example adds the `638aed2a18734375b` and `c42b225963ad481b9` labels to the `6b6e72dd71bd4fa` syndication.

```
[PATCH]/v2/syndications/6b6e72dd71bd4fa{
   "include_labels":[
      "638aed2a18734375b",
      "c42b225963ad481b9"
   ]
}
```

   Backlot returns a response similar to the following.

```
{
   "name":"My iPhone Feed",
   "include_encodings":[

   ],
   "destination_url":"",
   "require_access_key":false,
   "title":"",
   "include_labels":[
      "638aed2a18734375b",
      "c42b225963ad481b9"
   ],
   "asset_types":[
      "video",
      "ad"
   ],
   "syndication_url":"http://api.ooyala.com/syndications/6b6e72dd71bd4fa/
feed?pcode=Y4cG06BlqYMLIjPGyv1NbacSK-x3",
   "id":"6b6e72dd71bd4fa",
   "include_all_content":false,
   "type":"iphone",
   "description":""
}
```

3. To filter the types of encodings provided, you can specify device or container limits.

   The following example restricts the encodings to iPhone and the container to MP4 for the `6b6e72dd71bd4fa` syndication.

```
[PATCH]/v2/syndications/6b6e72dd71bd4fa{
   "include_encodings":[
      {
         "device":"iphone",
         "container":"mp4"
      }
   ]
```


```
}
```

Backlot returns a response similar to the following.

```
{
   "name":"My iPhone Feed",
   "include_encodings":[
      {
         "device":"iphone",
         "container":"mp4"
      }
   ],
   "destination_url":"",
   "require_access_key":false,
   "title":"",
   "include_labels":[
      "638aed2a18734375b",
      "c42b225963ad481b9"
   ],
   "asset_types":[
      "video",
      "ad"
   ],
   "syndication_url":"http://api.ooyala.com/syndications/6b6e72dd71bd4fa/
feed?pcode=Y4cG06BlqYMLIjPGyv1NbacSK-x3",
   "id":"6b6e72dd71bd4fa",
   "include_all_content":false,
   "type":"iphone",
   "description":""
}
```

The syndication is successfully created.

Note: Labels, encodings, and other settings can be specified during the initial creation of the syndication.

Editing a Syndication

You can edit a syndication at any time; any changes are made within minutes.

To edit a syndication:

1. Use the `syndications` route to get information about a syndication.
   The following example performs a get to obtain the current settings for the asset with the `6b6e72dd71bd4fa` ID.

   ```
   [GET]/v2/syndications/6b6e72dd71bd4fa
   ```

   Note: To get information about all syndications, do not specify the ID in the request.

   Backlot returns a response similar to the following.

   ```
   {
      "name":"My iPhone Feed",
      "include_encodings":[

      ],
      "destination_url":"",
      "require_access_key":false,
      "title":"",
      "include_labels":[
         "638aed2a18734375b",
         "c42b225963ad481b9"
   ```


```
   ],
   "asset_types":[
      "video",
      "ad"
   ],
   "syndication_url":"http://api.ooyala.com/syndications/6b6e72dd71bd4fa/
feed?pcode=Y4cG06BlqYMLIjPGyv1NbacSK-x3",
   "id":"6b6e72dd71bd4fa",
   "include_all_content":false,
   "type":"iphone",
   "description":""
}
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

2. Make any changes with a PATCH request.

   The following example changes the name, hosted at URL, and description.

```
[PATCH]/v2/syndications/ZnbWVpMjqToT1W{
   "name":"My Awesome iPhone Feed",
   "description":"My iPhone feed is awesome(2)."
}
```

   Backlot returns a response similar to the following.

```
{
   "name":"My Awesome iPhone Feed",
   "include_encodings":[

   ],
   "destination_url":"",
   "require_access_key":false,
   "title":"",
   "include_labels":[
      "638aed2a18734375b",
      "c42b225963ad481b9"
   ],
   "asset_types":[
      "video",
      "ad"
   ],
   "syndication_url":"http://api.ooyala.com/syndications/6b6e72dd71bd4fa/
feed?pcode=Y4cG06BlqYMLIjPGyv1NbacSK-x3",
   "id":"6b6e72dd71bd4fa",
   "include_all_content":false,
   "type":"iphone",
   "description":"My iPhone feed is awesome(2)."
}
```

   Note:

   Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

The syndication is successfully edited.

Viewing a Syndication

After creating a syndication, you can access it through a web browser, RSS application, or a custom application.

To view the results of a syndication:

1. Get the ID of the syndication.

   The following example gets syndication IDs.

```
[GET]/v2/syndications
```

   Backlot returns a response similar to the following.

```
{
   "items":[
      {
         "name":"my youtube syndication",
         "created_at":"2011-08-10T21:13:29+00:00",
         "require_access_key":false,
         "include_labels":[

         ],
         "should_create_youtube_videos":true,
         "asset_types":[
            "video",
            "ad"
         ],
         "username":"",
         "id":"9a6e1dcba36c452d95",
         "include_all_content":false,
         "type":"youtube",
         "unrestricted_youtube_account":false,
         "should_delete_youtube_videos":true
      },
      {
         "include_encodings":[

         ],
         "name":"my iTunes syndication",
         "created_at":"2011-08-10T20:11:56+00:00",
         "require_access_key":false,
         "category":"",
         "title":"",
         "include_labels":[

         ],
         "author":"",
         "asset_types":[
            "video",
            "ad"
         ],
         "syndication_url":"http://api.ooyala.com/v2/syndications/
b839b27a4e844a5c9/feed?pcode=Y4cG06BlqYMLIjPGyv1NbacSK-x3",
         "subtitle":"",
         "id":"b839b27a4e844a5c9",
         "include_all_content":false,
         "type":"itunes",
         "description":"",
         "keywords":""
      }
```


```
    ]
}
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

2. Open a web browser and paste the URL from one of the `syndication_url` entries.

   Backlot displays a list of assets.

Deleting a Syndication

You can delete a syndication at any time. No new content will be sent to the syndication and any feeds will be deleted.

To delete a syndication:

Use delete with the `syndications` route.

The following example deletes the `ZnbWVpMjqToT1W` syndication.

```
[DELETE]/v2/syndications/ZnbWVpMjqToT1W
```

Backlot returns a 200 response.

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

The syndication is successfully deleted.

Access Key Restrictions

Access keys can be used to create private URLs for a syndication's MRSS feed. With this you can give out unique feed URLs to third parties, and if needed, revoke their access to your MRSS feed without having to delete the feed.

Turn on Access Keys

You can turn on access keys for a syndication's MRSS feed.

To turn on access keys for a syndication:

Use a PATCH request to set "require_access_key" to "true" for the desired syndication.

The following example turns on access keys for the syndication with the `6b6e72dd71bd4fa`syndication ID.

```
[PATCH]/v2/syndications/6b6e72dd71bd4fa{
   "require_access_key":true
}
```

If "require_access_key" is set to true, navigating to the feed's syndication_url will no longer work. To access the feed, an "access_key=..." query string parameter must be added to the syndication_url, e.g.:

```
http://api.ooyala.com/v2/syndications/syndication_id/feed?
pcode=yourPcode&access_key=accessKey
```

Backlot returns a response similar to the following.

```
{
```


```
   "require_access_key":"true",
   "description":"",
   "destination_url":"",
   "type":"iphone",
   "include_labels":[

   ],
   "asset_types":[
      "video",
      "ad"
   ],
   "id":"6b6e72dd71bd4fa",
   "include_encodings":[

   ],
   "name":"iPhone feed",
   "title":"",
   "syndication_url":"http://api.ooyala.com/syndications/6b6e72dd71bd4fa/
feed?pcode=Y4cG06BlqYMLIjPGyv1NbacSK-x3",
   "include_all_content":"false",
   "created_at":"2013-07-08T17:09:02Z"
}
```

Creating an Access Key

You can create an access key for a specific syndication.

To create an access key:

Use the `/v2/syndications/syndication_id/access_keys` route, specifying the name of the access key.

The following example creates the "My Access Key" access key.

```
[POST]/v2/syndications/b839b27a4e844a5c9/access_keys{
   "name":"My Access Key"
}
```

Backlot returns a response similar to the following:

```
{
   "name":"My access key",
   "access_key":"12345678-abcd-abcd-12ab-12345678"
}
```

Viewing Access Keys

You can view the access keys associated with a syndication.

To view the access keys associated with a syndication:

1. Get the ID of the syndication.

   The following example gets syndication IDs.

   ```
   [GET]/v2/syndications
   ```

   Backlot returns a response similar to the following.

   ```
   {
      "items":[
         {
            "name":"my youtube syndication",
            "created_at":"2011-08-10T21:13:29+00:00",
   ```


```
            "require_access_key":false,
            "include_labels":[

            ],
            "should_create_youtube_videos":true,
            "asset_types":[
               "video",
               "ad"
            ],
            "username":"",
            "id":"9a6e1dcba36c452d95",
            "include_all_content":false,
            "type":"youtube",
            "unrestricted_youtube_account":false,
            "should_delete_youtube_videos":true
         },
         {
            "include_encodings":[

            ],
            "name":"my iTunes syndication",
            "created_at":"2011-08-10T20:11:56+00:00",
            "require_access_key":true,
            "category":"",
            "title":"",
            "include_labels":[

            ],
            "author":"",
            "asset_types":[
               "video",
               "ad"
            ],
            "syndication_url":"http://api.ooyala.com/v2/syndications/
b839b27a4e844a5c9/feed?pcode=Y4cG06BlqYMLIjPGyv1NbacSK-x3",
            "subtitle":"",
            "id":"b839b27a4e844a5c9",
            "include_all_content":false,
            "type":"itunes",
            "description":"",
            "keywords":""
         }
      ]
}
```


Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

2. View all access keys associated with the syndication.

   The following example gets syndication IDs.

```
[GET]/v2/syndications/b839b27a4e844a5c9/access_keys
```

Backlot returns a response similar to the following:


```
{
   "name":"My access key",
   "access_key":"12345678-abcd-abcd-12ab-12345678"
}
```


Remove an Access Key

You can remove an access key from a syndication.

To remove an access key from a syndication:

1. Use delete with the `v2/syndications/syndication_id/access_keys/access_key` route.

   The following example deletes the `12345678-abcd-abcd-12ab-12345678` access key from the b839b27a4e844a5c9 syndication.

   ```
   [DELETE]/v2/syndications/b839b27a4e844a5c9/access_keys/12345678-abcd-abcd-12ab-12345678
   ```

2. Backlot returns a 200 response.

The access is successfully removed from the syndication.

Specific Syndications

Live Stream Syndication

A live stream syndication is a variant on a universal syndication.

These are the key details about syndicating a live stream via the Backlot API.

1. Create the syndcation as described in *Creating a Syndication* on page 152, but in the body of the `POST` request set the syndication `type` to `universal` and the `asset_type` property to include the `live_stream` value (and other desired asset types), as shown in the snippet below.

   ```
   .
   .
   .
     "name": "My Live Stream",
     "type": "universal",
   .
   .
   . "asset_types": [
       "live_stream",
       "ad"
     ],
   .
   .
   .
   ```

The syndication is created.

In your Universal Syndication Template Language (USTL) feed template (see *http://support.ooyala.com/users/documentation/reference/fms.html*), you can include properties specific to live streams.

Syndication with YouTube

For YouTube syndication (that is, exporting videos to YouTube), Backlot automatically uploads the source files of videos directly to YouTube, which handles all transcoding and the generation of preview images.

Syndication with YouTube is "one-way": your settings and content on Backlot are pushed to YouTube, but settings and content on YouTube are *not* pulled to Backlot.

You can have only one YouTube syndication for a single YouTube account. For example, if you have YouTube accounts abc@abc.com and xyz@abc.com, abc@abc.com has its YouTube syndication and xyz@abc.com must have a separate YouTube syndication.

Ooyala does not impose a limit on the size and duration of videos files that you syndicate to YouTube. These limits are determined by your relationship with YouTube (your YouTube account settings). Ignore the message in the Backlot UI shown below.

YouTube itself is designed such that you cannot replace the content of a previously uploaded (or syndicated) video. Any new video content is assigned a new identifier (ID) by YouTube. Because of this design, many changes you make to assets or settings on Backlot cause YouTube to create new video identifiers, including the following:

- Changing a video's name
- Changing other details, such as description
- Changing the video's metadata
- Removing labels from the video, either with the Backlot UI or by editing the feed itself to remove the label. The videos formerly associated with that label are deleted from YouTube.
- Privacy settings. If you change privacy settings on YouTube, when Backlot updates YouTube, your former changes on YouTube are lost, including statistics relating to the account.

  Backlot supports only two YouTube privacy settings: public or private. When you create a YouTube syndication, by default your videos are public on YouTube. You can change this default, as detailed in the final step below.

  Changing the privacy setting affects only videos added to the syndication *after the change*. To change the privacy setting on videos previously syndicated to YouTube, change the setting on the individual asset with the **MANAGE** page, **Details** subtab for the specific asset, as shown in the following example.

Handling CAPTCHA Challenges from YouTube

You might been faced with a CAPTCHA challenge if you have tried to create a YouTube syndications several times with a bad password.

Here's how to handle such challenges.

1. YouTube challenges you if your syndication requests have the wrong password.
   The response body when a CAPTCHA challenge is triggered looks similar to the following.

```
{
 "captcha": [
```

```
     {
       "captcha_token": "6vKtzYIq85l-t",
       "captcha_url": "http://www.google.com/accounts/Captcha?ctoken=vd-
jLZqn55i",
       "message": "Youtube returned a CAPTCHA challenge"
        }
            ]
}
```

The value of `captcha_url` is a link to the CAPTCHA image.

You must the `captcha_token` name/value pair in the next request, along with one other field.

2. Make your request again, but include the following two fields in the POST body:
   - The `captcha_token` name/value pair you received in the response to the failed attempt to create the syndication
   - The `captcha_response` property with the word represented by the CAPTCHA image as value

Setting Privacy and Other Considerations for YouTube Syndication

For syndication with YouTube, you can set the default privacy for assets.

The default privacy settings for videos syndicated to YouTube can be set with the `/v2/syndications` route.

1. To see the current privacy settings of the syndication use `[GET] /v2/syndications` and the syndication ID. Look for the `asset_defaults` block in the response.
   The response body looks similar to the following.

```
[GET]/v2/syndications/fbdd2472css9476496cdaa5dc05ce6fb{
   "name":"YouTube",
   "created_at":"2011-06-15T23:05:51Z",
   "require_access_key":false,
   "should_create_youtube_videos":true,
   "include_labels":[
      "64df5faas3fcx27383d80744345ac866"
   ],
   "asset_types":[
      "video",
      "ad"
   ],
   "asset_defaults":{
      "private":true
   },
   "username":"GeorgeBush",
   "id":"fbdd2472css9476496cdaa5dc05ce6fb",
   "include_all_content":false,
   "type":"youtube",
   "unrestricted_youtube_account":false,
   "should_delete_youtube_videos":true
}
```

2. To change the privacy setting, use `PATCH /v2/syndications` with the syndication ID and the `asset_defaults` block with the `private` property. Valid values for this property are true and or false.

```
[PATCH]/v2/syndications/syndication_id{
   "assets_default":{
      "private":false
   }
}
```


PUBLISHING RULES

Publishing rules give you complete control over your content by enabling you to control where and when it can be viewed.

Backlot publishing rules enable you to:

- Allow or restrict website domains that can show your content—you can either specify a list of domains that are allowed to embed your content or specify a list of domains that cannot embed your content.
- Allow or prevent your content from being viewed in specific geographic regions, DMAs, and IP addresses—you can specify a list to countries, DMA regions, and IP addresses to allow or block.
- Detect and restrict anonymous proxy connections.

  For example, a user in country X spoofs his IP address through a VPN/proxy to look like he's in country Y to accesses content only permitted in country Y. You can set up anonymous proxy controls to restrict this user from viewing your content.
- Express embargo rules by restricting your content to specific ranges or recurring intervals—you specify a time that is applied around the world, based on your current time zone.

  For example, if you allow content to be viewed from 1:00 to 4:00 from the New York time zone (GMT -05:00), the content will be accessible around the world (in geographic regions that you allow) from 6:00 to 9:00 GMT.
- Limit your content to specific devices—you can currently specify whether to allow iOS devices to view your content.

You can use any combination of publishing rules to meet your requirements. For example, you might want to only allow your content to be viewed in the United States, between 6:00 PM and 11:00 PM, on non-iOS devices.

Creating Publishing Rules

You can create as many publishing rules as you like and assign them to videos, channels, or channel sets. During playback, the most restrictive rule as applied.

To create a publishing rule:

Use the `/v2/publishing_rule` route.

The following example creates the "My Publishing Rule" publishing rule which restricts viewing to the US and EU, restrict embedding to mywebsite.com, allows the content to be viewed on Mondays and Wednesdays, and allows the content to be played on desktops/notebooks and tablets.

```
[POST]/v2/publishing_rules{
   "name":"My Publishing Rule",
   "geographic_restrictions":{
      "type":"whitelist",
      "locations":[
         "US",
         "EU"
      ]
   },
   "domain_restrictions":{
      "type":"whitelist",
      "domains":[
         "mywebsite.com"
      ]
   },
   "time_restrictions":{
      "type":"recurring",
```


```
      "start_date":"2010-01-01",
      "end_date":null,
      "recurring_days":[
         "MON",
         "WED"
      ],
      "all_day":false,
      "start_time":"20:00:00",
      "end_time":"23:59:00"
   },
   "allowed_devices":[
      "desktop",
      "ipad",
      "android",
      "blackberry"
   ]
}
```

Backlot returns a response similar to the following.

```
{
   "name":"My Awesome Publishing Rule",
   "allowed_devices":[
      "desktop",
      "ipad",
      "android",
      "blackberry"
   ],
   "time_restrictions":{
      "start_date":"2010-01-01",
      "end_time":"23:59:00",
      "recurring_days":[
         "MON",
         "WED"
      ],
      "type":"recurring",
      "end_date":null,
      "all_day":false,
      "start_time":"20:00:00"
   },
   "domain_restrictions":{
      "domains":[
         "mywebsite.com"
      ],
      "type":"whitelist"
   },
   "geographic_restrictions":{
      "type":"whitelist",
      "locations":[
         "US",
         "EU"
      ]
   },
   "id":"5b3ff777724d46f"
}
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

The publishing rule is successfully added.

Editing Publishing Rules

You can edit a publishing rule at any time. Any videos, channels, or channel sets that use the publishing rule are automatically updated.

To edit a publishing rule:

Use PATCH with the `publishing_rule` route.

The following example changes the name of the publishing rule and adds Sundays to the list of allowed days for the `5b3ff777724d46f` publishing rule.

```
[PATCH]/v2/publishing_rules/5b3ff777724d46f{
   "name":"My Super Awesome Publishing Rule",
   "time_restrictions":{
      "type":"recurring",
      "start_date":"2010-01-01",
      "end_date":null,
      "recurring_days":[
         "MON",
         "WED",
         "SUN"
      ],
      "all_day":false,
      "start_time":"20:00:00",
      "end_time":"23:59:00"
   }
}
```

Backlot returns a response similar to the following.

```
{
   "name":"My Super Awesome Publishing Rule",
   "allowed_devices":[
      "desktop",
      "ipad"
   ],
   "time_restrictions":{
      "start_date":"2010-01-01",
      "end_time":"23:59:00",
      "recurring_days":[
         "SUN",
         "MON",
         "WED"
      ],
      "type":"recurring",
      "end_date":null,
      "all_day":false,
      "start_time":"20:00:00"
   },
   "domain_restrictions":{
      "domains":[
         "mywebsite.com"
      ],
      "type":"whitelist"
   },
   "geographic_restrictions":{
      "type":"whitelist",
      "locations":[
         "US",
         "EU"
      ]
```


```
    },
    "id":"5b3ff777724d46f"
}
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

The publishing rule is successfully edited.

Deleting Publishing Rules

You can delete a publishing rule at any time. Any videos, channels, or channel sets that use the publishing rule will automatically have any restrictions removed.

To delete a publishing rule:

Use delete with the `publishing_rule` route.

The following example deletes the `5b3ff777724d46f` publishing rule.

```
[DELETE]/v2/publishing_rules/5b3ff777724d46f
```

Backlot returns a 200 response.

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

The publishing rule is successfully deleted.

LABELS

When added to external publishing targets, labels specify which videos, channels, and channel sets to publish externally.

Additionally, labels are also useful for organizing your video library, searching for videos, and retrieving targeted analytics.

BEST PRACTICES FOR LABELS

The following best practices will ensure that you have optimal performance and speed in Theme Builder.

- Limit the use of excess labels in your Backlot account as a low this will keep load times low in Theme Builder.
- Create broad and categorical labels as labels that apply to only 1 or 2 movies will result in many labels that are of limited usefulness.

Creating Labels

With the Backlot API, you can create top-level and child labels.

To create labels:

1. Create a top-level label.

   The following example creates a "Hobbies" label.

   ```
   [POST]/v2/labels{
   ```


```
   "name":"Hobbies"
}
```

Backlot returns a response similar to the following.

```
{
   "name": "Hobbies",
   "id": "Yh98G7hsaß",
   "full_name": "/Hobbies"
}
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

2. Create a child label.

   The following example creates a "Hockey" label under the "Hobbies" parent label by specifying the ID of Hobbies label.

```
[POST] /v2/labels{
   "parent_id":"Yh98G7hsaß",
   "name":"Hockey"
}
```

Backlot returns a response similar to the following.

```
{
   "name": "Hockey",
   "id": "Yh98G7hsaß",
   "full_name": "/Hobbies/Hockey",
   "parent_id": "Hwe8G7hrt5"
}
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

Deleting Labels

With the Backlot API, you can delete labels at any time.

To delete labels:

1. If you don't have the ID of the label to delete, use GET with the `/v2/labels` route.

   The following is a GET example.

```
[GET]/v2/labels
```

Backlot returns a response similar to the following.

```
[
   {
      "name":"Hobbies",
      "id":"Yh98G7hsaß",
      "full_name":"/Hobbies"
   },
   {
      "name":"Hockey",
      "id":"82d562682",
```


```
        "full_name":"/Hobbies/Hockey",
        "parent_id":"Yh98G7hsaß"
    },
    {
        "name":"Disc Golf",
        "id":"55514522",
        "full_name":"/Sports/Disc Golf",
        "parent_id":"Yh98G7hsaß"
    }
]
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

2. Delete a label.

   The following example deletes the "Disc Golf" label.

```
[DELETE]/v2/labels/55514522
```

   Backlot returns a 200 response code.

   Note:

   Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

Working with Labels

After you associate a video, channel, remote asset, or channel set with a label, Backlot automatically publishes it based on how you configured syndication.

Before you can associate them with assets, you must create the labels. See .

To work with labels for an asset:

1. To associate a label with an asset, use `POST` with the `/v2/assets` route, the asset ID, the `/v2/labels` qualifier, and a label ID. You can also associate multiple labels at the same time.

   The following example adds the Motorcycle Racing label to the `JxbzdkMjqBEsO` asset.

```
[PUT]/v2/assets/JxbzdkMjqBEsO/labels/bace921fdea44cc18a5a273155514522
```

   Backlot returns a response similar to the following.

```
{
    "items":[
        {
            "name":"Sports",
            "id":"814efb109416490a98ee3f4fcd6784cf",
            "full_name":"/Sports",
            "parent_id":null
        },
        {
            "name":"Motorcycle Racing",
            "id":"bace921fdea44cc18a5a273155514522",
            "full_name":"/Sports/Motorcycle Racing",
            "parent_id":"814efb109416490a98ee3f4fcd6784cf"
        }
    ]
}
```


Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

2. View the current labels for an asset with `GET /v2/assets/asset_id/labels`.

   The following example gets the labels for the `JxbzdkMjqBEsO` asset.

```
[GET]/v2/assets/JxbzdkMjqBEsO/labels
```

   Backlot returns a response similar to the following.

```
{
   "items":[
      {
         "name":"Sports",
         "id":"814efb109416490a98ee3f4fcd6784cf",
         "full_name":"/Sports",
         "parent_id":null
      }
   ]
}
```

   Note:

   Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

3. To remove a single label from an asset, use `DELETE` with the `/v2/assets` route, the asset ID, the `/labels` qualifier, and a label ID. To disassociate all labels from the asset, specify `/labels` without a label ID.

   Note: To permanently delete a label and remove it from all assets, see *http://support.ooyala.com/documentation/users/tasks/label_delete.html*.

   The following example removes the Motorcycle Racing label from the `JxbzdkMjqBEsO` asset.

```
[DELETE]/v2/assets/JxbzdkMjqBEsO/labels/814efb109416490a98ee3f4fcd6784cf
```

   Backlot returns a 200 response.

   Note:

   Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

UNIVERSAL SYNDICATION

Universal Syndication enables you to easily create and modify custom feeds, in any format that meets your needs. Once configured, you can immediately deliver video to new partners, devices, and distribution channels such as Roku, Boxee, AppleTV, and so on.

To use Universal Syndication, you create asset templates using the Universal Syndication Template Language (USTL). For example, you can create lists of assets and their metadata in XML, JSON, CSV, or any other text format.

Note: Universal Syndication is included with the Enterprise package. If you would like to try it for 90 days, contact Sales, your Customer Success Manager, or Technical Support.

Creating a Simple Custom XML Syndication

To create a simple custom XML syndication:

1. Create an advanced syndication.

   The following example creates the "My Custom XML Feed" advanced syndication.

```
[POST]/v2/syndications{
   "name":"My Custom XML Feed",
   "type":"advanced"
}
```

   Backlot returns a response similar to the following, which includes the URL where the feed can be accessed.

```
{
   "name":"My Custom XML Feed",
   "include_encodings":[

   ],
   "destination_url":"",
   "require_access_key":false,
   "title":"",
   "include_labels":[

   ],
   "asset_types":[
      "video",
      "ad"
   ],
   "syndication_url":"http://api.ooyala.com/syndications/6b6e72dd71bd4fa/
feed?pcode=Y4cG06BlqYMLIjPGyv1NbacSK-x3",
   "id":"6b6e72dd71bd4fa",
   "include_all_content":false,
   "type":"advanced",
   "description":""
}
```

   Note:

   Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

2. Create a Universal Syndication Template Language (USTL) template.

   The following template creates a simple list of assets, which specifies the name, description, and length of each asset.

```
<feed header>List of My Assets</feed header>
{% for asset in assets %}
  <item>
    <title> {{asset.name}} </title>
    <description> {{asset.description}} </description>
    <length> {{asset.duration}} minutes</length>
  </item>
{% endfor assets %}
```


Note: For more information about USTL, see the Universal Syndication Template Language section of the Backlot API Reference.

3. Attach the Universal Syndication Template Language (USTL) template

   The following example attaches the template to the newly created syndication.

```
[PUT]/v2/syndications/6b6e72dd71bd4fa/template
 <feed header>List of My Assets</feed header>{
   % for asset in assets %
}<item>
 <title>{
   {
      asset.name
   }
}</title>
 <description>{
   {
      asset.description
   }
}</description>
 <length>{
   {
      asset.duration
   }
}minutes</length>
 </item>{
   % endfor assets %
}
```

   Backlot returns a 200 response.

4. For the feed to contain videos and other assets, you must add labels. Whenever you add a label to an asset that is associated with a feed, the asset is automatically added to the feed.

   The following example adds the `638aed2a18734375b` and `c42b225963ad481b9` labels to the `6b6e72dd71bd4fa` syndication.

```
[PATCH]/v2/syndications/6b6e72dd71bd4fa{
   "include_labels":[
      "638aed2a18734375b",
      "c42b225963ad481b9"
   ]
}
```

   Backlot returns a response similar to the following.

```
{
   "name":"My iPhone Feed",
   "include_encodings":[

   ],
   "destination_url":"",
   "require_access_key":false,
   "title":"",
   "include_labels":[
      "638aed2a18734375b",
      "c42b225963ad481b9"
   ],
   "asset_types":[
      "video",
      "ad"
   ],
```


```
   "syndication_url":"http://api.ooyala.com/syndications/6b6e72dd71bd4fa/
feed?pcode=Y4cG06BlqYMLIjPGyv1NbacSK-x3",
   "id":"6b6e72dd71bd4fa",
   "include_all_content":false,
   "type":"iphone",
   "description":""
}
```

Note:

Try out the code samples using your account credentials in the Ooyala Scratchpad. For information about using the Scratchpad, see *The Scratchpad*.

5. To view the results of the feed, open the syndication URL. If you do not have it, you can get it by making a get request against the syndication ID.

   In this example, the `http://api.ooyala.com/syndications/6b6e72dd71bd4fa/feed?pcode=Y4cG06BlqYMLIjPGyv1NbacSK-x3` URL returns results similar to the following:

```
<feed header>List of My Assets</feed header>
<item>
  <title>Presidential Speech #125784</title>
  <video_info>The president said something important today about something
 that happened.</video_info>
</item>
  <title>Presidential Speech #125785</title>
  <video_info>The president said something important today about something
 that should happen.</video_info>
</item>
```


SAMPLE CODE FOR SIGNING REQUESTS

Here is a collection of sample code in various programming languages for signing your requests.

SAMPLE SIGNATURE CODE (RUBY)

This following is Ruby sample code for generating signatures:

```
require "digest/sha2"
require "base64"

# Both of these values should be obtained from the Developers tab in
 Backlot.
API_KEY = ""
SECRET = ""

module OoyalaApi
  def self.generate_signature(secret, http_method, request_path,
 query_string_params, request_body)
    string_to_sign = secret + http_method + request_path
    sorted_query_string = query_string_params.sort { |pair1, pair2| pair1[0]
 <=> pair2[0] }
    string_to_sign += sorted_query_string.map { |key, value|
 "#{key}=#{value}"}.join
    string_to_sign += request_body.to_s
    signature = Base64::encode64(Digest::SHA256.digest(string_to_sign))
[0..42].chomp("=")
    return signature
  end
end

# Example usage of the generate_signature function:
#
# Set `expires` in 1-hour intervals for higher caching performance:
# t = Time.now
# expires = Time.local(t.year, t.mon, t.day, t.hour + 1).to_i
#
# params = { "api_key" => API_KEY, "expires" => expires }
# signature = OoyalaApi.generate_signature(SECRET, "GET", "/v2/players/
HbxJKM", params, nil)
```

SAMPLE SIGNATURE CODE (JAVA)

This following is Java sample code for generating signatures:

```
import java.util.Collections;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.Vector;
```


```
import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.net.URLEncoder;
import java.io.IOException;

import sun.misc.BASE64Encoder;

public class Signature {

    private String concatenateParams(HashMap<String, String> parameters,
 String separator) {
        Vector<String> keys = new Vector<String>(parameters.keySet());
        Collections.sort(keys);

        String string = "";
        for (Enumeration<String> e = keys.elements(); e.hasMoreElements();)
 {
            String key    = (String)e.nextElement();
            String value  = (String)parameters.get(key);
            if (!string.isEmpty())
                string += separator;
            string += key + "=" + value;
        }
        return string;
    }

    public String generateSignature(String secretKey, String HTTPMethod,
 String requestPath, HashMap<String, String> parameters, String
 requestBody) throws NoSuchAlgorithmException, UnsupportedEncodingException
 {
        String stringToSign = secretKey + HTTPMethod + requestPath;
        stringToSign += concatenateParams(parameters, "");
        stringToSign += requestBody;
        MessageDigest digestProvider = MessageDigest.getInstance("SHA-256");
        digestProvider.reset();

        byte[] digest = digestProvider.digest(stringToSign.getBytes());
        BASE64Encoder base64Encoder = new BASE64Encoder();
        String signedInput = base64Encoder.encode(digest);
        return URLEncoder.encode(signedInput.substring(0, 43), "US-ASCII");
    }

}
```

SAMPLE SIGNATURE CODE (PHP)

This following is PHP sample code for generating signatures:

```
class OoyalaAPI{
 /**
  *
  * Generates the url for the request
  * @param string $HTTP_method the HTTP method to be used when generating
 the signature, i.e. GET, POST, etc.
  * @param string $request_path the path to the resource to use for the
 call, i.e. "/v2/players"
```


```
 * @param string $api_key the API key for the account to use for the call.
This key can be found in the developers tab in Backlot
 * @param string $secret_key the Secret key for the account to use for the
call. This key can be found in the developers tab in Backlot
        * @param string $expires UNIX timestamp (seconds since Jan 1st,
1970) indicating when your request should be valid until
 * @param array $parameters extra parameters to be specified in the url,
i.e include => metadata
 * @param string $request_body string containing data in JSON format to
update or create objects
        * @return string containing the URL for the API request to be made
 */
public function generateURL($HTTP_method, $api_key, $secret_key, $expires,
$request_path, $request_body = "", $parameters=array())
{

 $parameters["api_key"] = $api_key;
 $parameters["expires"] = $expires;
 $signature = $this->generateSignature($HTTP_method, $secret_key,
$request_path, $parameters, $request_body);
 $base = "https://api.ooyala.com";
 $url = $base.$request_path."?";
 foreach ($parameters as $key => $value) {
  $url .=  $key . "=" . urlencode($value) . "&";
 }
 $url .= "signature=" . $signature;
 return $url;
}

/**
 *
 * Generates the signature for a request
 * @param $secretKey secret key
 * @param $HTTPMethod GET, POST, PUT, PATCH, DELETE
 * @param $requestPath the path of the object to request
 * @param $parameters array of parameters [key => value]
 * @param $request_body body for the request
 */
private function generateSignature($HTTP_method, $secret_key,
$request_path, $parameters, $request_body = "")
{
 $to_sign = $secret_key . $HTTP_method . $request_path;
 $keys = $this->sortKeys($parameters);
 foreach ($keys as $key) {
  $to_sign .= $key . "=" . $parameters[$key];
 }
 $to_sign .= $request_body;
 $hash = hash("sha256", $to_sign,true);
 $base = base64_encode($hash);
 $base = substr($base,0,43);
 $base = urlencode($base);
 return $base;

}

private function sortKeys($array)
{
 $keys = array();$ind=0;
 foreach ($array as $key => $val) {
  $keys[$ind++]=$key;
 }
 sort($keys);
 return $keys;
}
```


```
}
```

SAMPLE SIGNATURE CODE (C#)

This following is C# sample code for generating signatures:

```
using System;
using System.Collections.Generic;
using System.Security.Cryptography;
using System.Text;
using System.Web;

public class OoyalaAPI
{
 public const String BaseURL = "https://api.ooyala.com";

 /// <summary>
 /// Takes in the necessary parameters to build a V2 signature for the
 Ooyala API
 /// </summary>
 /// <param name="apiKey">
 /// The API key for the Ooyala account to generate the URL for. You can
 find this key in the Developers tab in Backlot
 /// </param>
 /// <param name="secretKey">
 /// The Secret key for the Ooyala account to generate the URL for. You can
 find this key in the Developers tab in Backlot
 /// </param>
 /// <param name="HTTPMethod">
 /// The method to be used for the request. Possible values are: GET, POST,
 PUT, PATCH or DELETE
 /// </param>
 /// <param name="path">
 /// The path to use for the request
 /// </param>
 /// <param name="parameters">
 /// A hash containing the list of parameters that will be included in the
 request.
 /// </param>
 /// <param name="body">
 /// A String containing the JSON representation of the data to be sent on
 the request. If its a GET request, the body parameter will not be used to
 generate the signature.
 /// </param>
 /// <returns>
 /// The URL to be used in the HTTP request.
 /// </returns>
 public String generateURL (String apiKey, String secretKey, String
 HTTPMethod, String path, Dictionary<System.String,System.String>
 parameters, String body)
 {
  var url = BaseURL + path;

  parameters.Add("api_key", apiKey);

  //Sorting the keys
```


```
 var sortedKeys = new String[parameters.Keys.Count];
 parameters.Keys.CopyTo(sortedKeys,0);
 Array.Sort(sortedKeys);

 for (int i = 0; i < sortedKeys.Length; i++) {
  url += (i == 0 ? "?" : "&") + sortedKeys[i] + "=" +
parameters[sortedKeys[i]];
 }

 url += "&signature=" + this.generateRequestSignature(secretKey,
HTTPMethod, path, sortedKeys, parameters, body);

 return url;
}

public String generateRequestSignature(String secretKey, String HTTPMethod,
String path, String[] sortedParameterKeys, Dictionary<String,String>
parameters, String body){
 var StringToSign = secretKey + HTTPMethod + path;

 for (int i = 0; i < sortedParameterKeys.Length; i++) {
  StringToSign += sortedParameterKeys[i] + "=" +
parameters[sortedParameterKeys[i]];
 }

               stringToSign += body;

               var sha256 = new SHA256Managed();
 byte[] digest = sha256.ComputeHash(Encoding.ASCII.GetBytes(StringToSign));
          String signedInput = Convert.ToBase64String(digest);

 //Removing the trailing = signs
 var lastEqualsSignIndex = signedInput.Length - 1;
 while (signedInput[lastEqualsSignIndex] == '=')
 {
  lastEqualsSignIndex--;
 }

 signedInput = signedInput.SubString(0, lastEqualsSignIndex + 1);

          return HttpUtility.UrlEncode(signedInput.SubString(0, 43));
}

}
```

SAMPLE SIGNATURE CODE (PYTHON)

This following is Python sample code for generating signatures:

```
import hashlib
import base64
import urllib

class ooyala_api(object):
```


```
    def generate_signature(self, secret_key, http_method, request_path,
 query_params, request_body=''):
        signature = secret_key + http_method.upper() + request_path
        for key, value in query_params.iteritems():
            signature += key + '=' + value
        signature = base64.b64encode(hashlib.sha256(signature).digest())
[0:43]
        signature = urllib.quote_plus(signature)
        return signature

# Example usage of the generate_signature function:
# Example URL : http://api.ooyala.com/docs/v2/
#new_ooyala_api = ooyala_api()
#query_params = {'api_key': '7ab06', 'expires': '1299991855'}
#print
 new_ooyala_api.generate_signature('329b5b204d0f11e0a2d060334bfffe90ab18xqh5',
 'get', '/v2/players/HbxJKM', query_params)
```

SAMPLE SIGNATURE CODE (OBJECTIVE C)

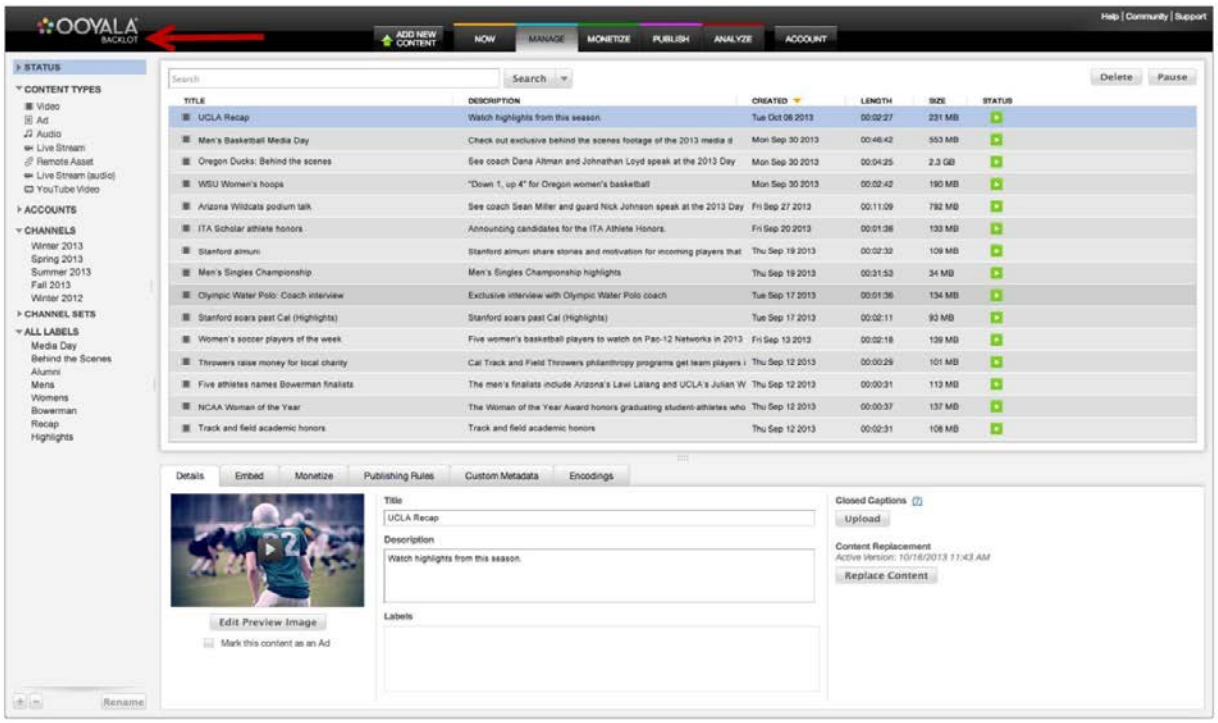
This following is Objective C sample code for generating signatures:

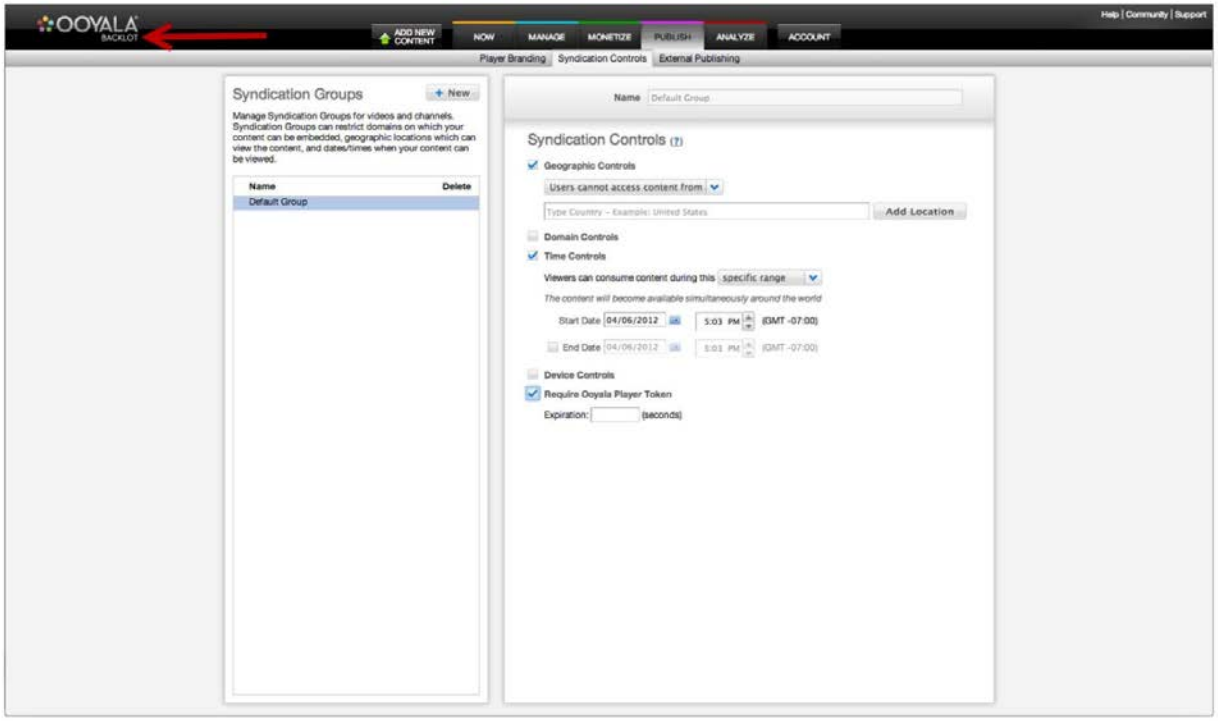
```
#import "OoyalaAPI.h"
#import <CommonCrypto/CommonDigest.h>
#import "GTMBase64.h"
#import "GTMDefines.h"

#define ROUNDING_WINDOW 300

@implementation OoyalaAPI
/// <summary>
/// Takes in the necessary parameters to build a V2 signature for the Ooyala
 API
/// Use generateEncodedSignedURLWithHTTPMethod
/// </summary>
/// <param name="apiKey">
/// The API key for the Ooyala account to generate the URL for. You can find
 this key in the Developers tab in Backlot
/// </param>
/// <param name="secretKey">
/// The Secret key for the Ooyala account to generate the URL for. You can
 find this key in the Developers tab in Backlot
/// </param>
/// <param name="HTTPMethod">
/// The method to be used for the request. Possible values are: GET, POST,
 PUT, PATCH or DELETE
/// </param>
/// <param name="requestPath">
/// The path to use for the request
/// </param>
/// <param name="queryStringParameters">
/// A dictionary containing the list of parameters that will be included in
 the request.
/// </param>
/// <param name="requestBody">
```

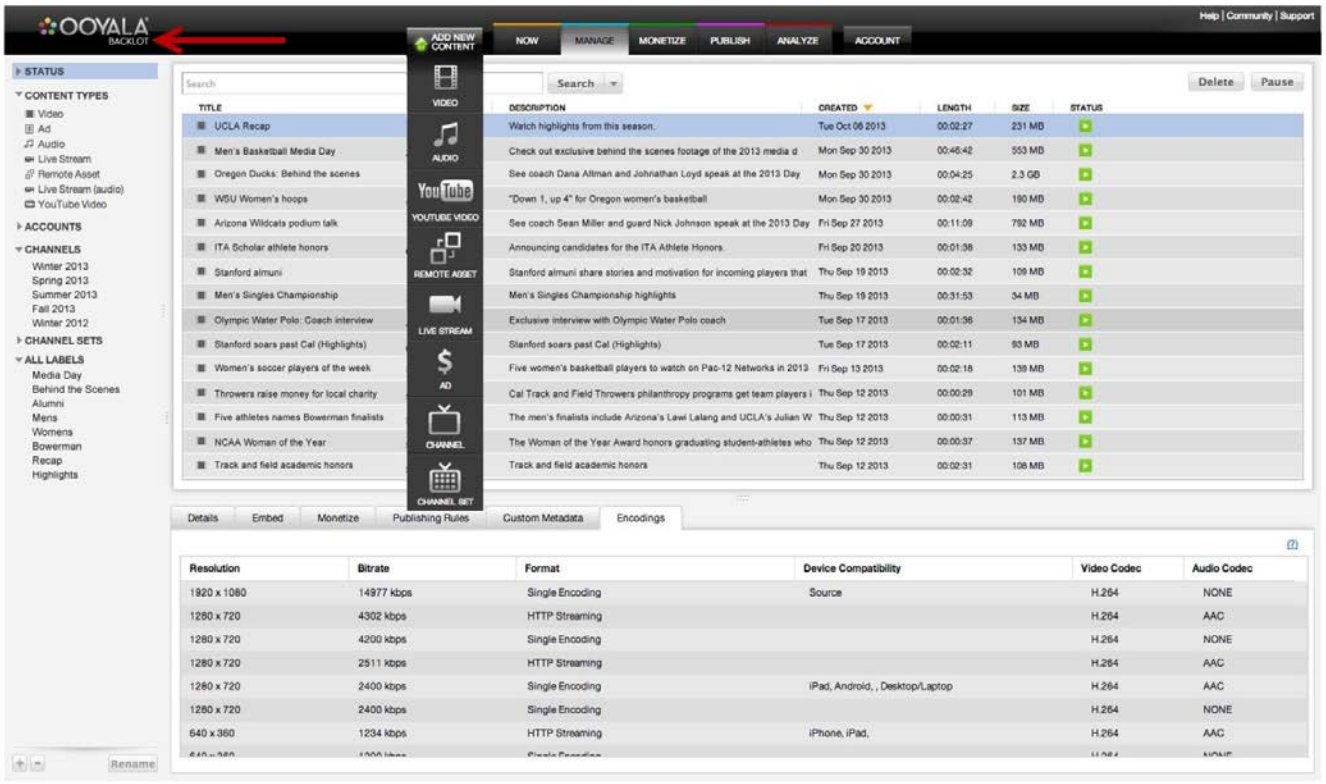

```
/// An NSData containing the JSON representation of the data to be sent
 on the request. If there's no body for the request, just include an empty
 NSData object
/// </param>
/// <returns>
/// The signed URL to be used in the HTTP request.
/// </returns>



+ (NSString *)URLEncodeString:(NSString *)string
{
  CFTypeRef URLEncodedCFType = CFURLCreateStringByAddingPercentEscapes(NULL,
 (CFStringRef)string, NULL, (CFStringRef)@"!*'\"();:@&=+$,/?%#[]% ",
 kCFStringEncodingUTF8);
  NSString *URLEncodedString = [NSString
 stringWithFormat:@"%@",URLEncodedCFType];
  CFRelease(URLEncodedCFType);
  return URLEncodedString;
}

- (NSString *)generateEncodedSignatureWithHTTPMethod:(NSString *)HTTPMethod
 requestPath:(NSString *)requestPath queryStringParameters:(NSDictionary
 *)queryStringParameters secretKey:(NSString *)secretKey apiKey:(NSString
 *)apiKey andRequestBody:(NSData *)requestBody
{

  //
  // Generate stringToSign
  //

  //Concatenate first parameters to stringToSign
  NSString *stringToSign = [NSString stringWithFormat:@"%@%@
%@",secretKey,HTTPMethod,requestPath];

  //Generate mutable dictionary for parameters
  NSMutableDictionary *parametersDictionary = [NSMutableDictionary
 dictionaryWithDictionary:queryStringParameters];

  //Expires
  //Generate and add expires parameter if not already present
  //Default expires time: 5min = 300s
  if(![parametersDictionary objectForKey:@"expires"]){
    NSNumber *expiresWindow = [NSNumber numberWithInt:15];
    NSUInteger timestamp = (long)[[NSDate date] timeIntervalSince1970] +
 [expiresWindow intValue];
    timestamp += [[NSNumber numberWithInt:ROUNDING_WINDOW] intValue] -
 (timestamp % [[NSNumber numberWithInt:ROUNDING_WINDOW] intValue] );
     [parametersDictionary setValue:[NSString stringWithFormat:@"%d",
 timestamp] forKey:@"expires"];
  }

  //Add api_key parameter
  [parametersDictionary setValue:[NSString stringWithFormat:@"%@", apiKey]
 forKey:@"api_key"];

  //Sort parameters and append to stringToSign
  NSArray *keys = [[parametersDictionary allKeys]
 sortedArrayUsingSelector:@selector(localizedCaseInsensitiveCompare:)];
  for (NSUInteger i = 0; i < [keys count]; i++) {
    NSString *key = [keys objectAtIndex:i];
    NSString *value = [parametersDictionary objectForKey:key];
```


```
    stringToSign = [stringToSign stringByAppendingFormat:@"%@=%@", key,
 value];
  }

  //Append Body
  NSString *requestBodyString = [[NSString alloc] initWithData:requestBody
 encoding:NSUTF8StringEncoding];
  stringToSign = [stringToSign
 stringByAppendingFormat:@"%@",requestBodyString];
  [requestBodyString release];

  //
  // Generate signature from stringToSign
  //

  unsigned char hashedChars[32];
  NSUInteger i;

  //Generate SHA-256 in Base64
  CC_SHA256([stringToSign UTF8String], [stringToSign
 lengthOfBytesUsingEncoding:NSUTF8StringEncoding], hashedChars);
  NSData *hashedData = [NSData dataWithBytes:hashedChars length:32];
  Class _gtmBase64 = NSClassFromString(@"GTMBase64");
  if (!_gtmBase64){
    [NSException raise:@"OOMissingLibraryException" format:@"GTMBase64
 is not pressent on the current Target. Add GTMBase64.h, GTMBase64.m and
 GTMDefines.h from the Google Toolbox for Mac (http://code.google.com/p/
google-toolbox-for-mac/)"];
  }
  NSString *signature = [_gtmBase64 stringByEncodingBytes:[hashedData bytes]
 length:32];

  //Truncate signature to 43 characters
  signature = [signature substringToIndex:(NSUInteger)43];

  //Remove from signature trailing = signs
  for (i = [signature length] - 1; [signature characterAtIndex:i] == '='; i
 = [signature length] - 1) {
    signature = [signature substringToIndex:i];
  }

  //URL-encode signature
  return [OoyalaAPI URLEncodeString:signature];
}



- (NSURL *)generateEncodedSignedURLWithHTTPMethod:(NSString *)HTTPMethod
 requestPath:(NSString *)requestPath queryStringParameters:(NSDictionary
 *)queryStringParameters apiKey:(NSString *)apiKey secretKey:(NSString
 *)secretKey andRequestBody:(NSData *)requestBody
{
  //Append first parts of URLString
  NSString *URLString = [NSString stringWithFormat:@"%@%@",@"https://
api.ooyala.com",requestPath];

  /**
   * Process queryStringParameters
   */

  //Generate mutable dictionary for parameters
  NSMutableDictionary *parametersDictionary = [NSMutableDictionary
 dictionaryWithDictionary:queryStringParameters];
```


```
  //Expires
  //Generate and add expires parameter if not already present
  //Default expires time: 5min = 300s
  if(![parametersDictionary objectForKey:@"expires"]){
    NSNumber *expiresWindow = [NSNumber numberWithInt:15];
    NSUInteger timestamp = (long)[[NSDate date] timeIntervalSince1970] +
 [expiresWindow intValue];
    timestamp += [[NSNumber numberWithInt:ROUNDING_WINDOW] intValue] -
 (timestamp % [[NSNumber numberWithInt:ROUNDING_WINDOW] intValue] );
    [parametersDictionary setValue:[NSString stringWithFormat:@"%d",
 timestamp] forKey:@"expires"];
  }

  //Add api_key parameter
  [parametersDictionary setValue:[NSString stringWithFormat:@"%@", apiKey]
 forKey:@"api_key"];

  //Sort parameters and append to URLString
  NSArray *keys = [[parametersDictionary allKeys]
 sortedArrayUsingSelector:@selector(localizedCaseInsensitiveCompare:)];
  for (NSUInteger i = 0; i < [keys count]; i++) {
    NSString *key = [keys objectAtIndex:i];
    NSString *value = [parametersDictionary objectForKey:key];
    NSString *format = (i==0)?@"?%@=%@":@"&%@=%@";
    URLString = [URLString stringByAppendingFormat:format, [OoyalaAPI
 URLEncodeString:key], [OoyalaAPI URLEncodeString:value]];
  }

  //Append the signature
  URLString = [URLString stringByAppendingFormat:@"&signature=%@",[self
 generateEncodedSignatureWithHTTPMethod:HTTPMethod requestPath:requestPath
 queryStringParameters:queryStringParameters secretKey:secretKey
 apiKey:apiKey andRequestBody:requestBody]];

  return [NSURL URLWithString:URLString];
}


@end
```



OOYALA
BACKLOT
ADD NEW CONTENT
NOW
MANAGE
MONETIZE
PUBLISH
ANALYZE
ACCOUNT
Help | Community | Support
STATUS
CONTENT TYPES
Video
Ad
Audio
Live Stream
Remote Asset
Live Stream (audio)
YouTube Video
ACCOUNTS
CHANNELS
Winter 2013
Spring 2013
Summer 2013
Fall 2013
Winter 2012
CHANNEL SETS
ALL LABELS
Media Day
Behind the Scenes
Alumni
Mens
Womens
Bowerman
Recap
Highlights
Rename
Search
Delete
Pause
TITLE
DESCRIPTION
CREATED
LENGTH
SIZE
STATUS
UCLA Recap
Watch highlights from this season
Tue Oct 08 2013
00:02:27
231 MB
Men's Basketball Media Day
Check out exclusive behind the scenes footage of the 2013 media d
Mon Sep 30 2013
00:46:42
553 MB
Oregon Ducks: Behind the scenes
See coach Dana Altman and Johnathan Loyd speak at the 2013 Day
Mon Sep 30 2013
00:04:25
2.3 GB
WSU Women's hoops
"Down 1, up 4" for Oregon women's basketball
Mon Sep 30 2013
00:02:42
190 MB
Arizona Wildcats podium talk
See coach Sean Miller and guard Nick Johnson speak at the 2013 Day
Fri Sep 27 2013
00:11:09
782 MB
ITA Scholar athlete honors
Announcing candidates for the ITA Athlete Honors.
Fri Sep 20 2013
00:01:36
133 MB
Stanford almuni
Stanford almuni share stories and motivation for incoming players that
Thu Sep 19 2013
00:02:32
109 MB
Men's Singles Championship
Men's Singles Championship highlights
Thu Sep 19 2013
00:31:53
34 MB
Olympic Water Polo: Coach interview
Exclusive interview with Olympic Water Polo coach
Tue Sep 17 2013
00:01:36
134 MB
Stanford soars past Cal (Highlights)
Stanford soars past Cal (Highlights)
Tue Sep 17 2013
00:02:11
93 MB
Women's soccer players of the week
Five women's basketball players to watch on Pac-12 Networks in 2013
Fri Sep 13 2013
00:02:18
139 MB
Throwers raise money for local charity
Cal Track and Field Throwers philanthropy programs get team players i
Thu Sep 12 2013
00:00:29
101 MB
Five athletes names Bowerman finalists
The men's finalists include Arizona's Lawi Lalang and UCLA's Julian W
Thu Sep 12 2013
00:00:31
113 MB
NCAA Woman of the Year
The Woman of the Year Award honors graduating student-athletes who
Thu Sep 12 2013
00:00:37
137 MB
Track and field academic honors
Track and field academic honors
Thu Sep 12 2013
00:02:31
108 MB
Details
Embed
Monetize
Publishing Rules
Custom Metadata
Encodings
Edit Preview Image
Mark this content as an Ad
Title
UCLA Recap
Description
Watch highlights from this season.
Labels
Closed Captions
Upload
Content Replacement
Active Version: 10/16/2013 11:43 AM
Replace Content


OOYALA
BACKLOT
ADD NEW CONTENT
NOW
MANAGE
MONETIZE
PUBLISH
ANALYZE
ACCOUNT
Help | Community | Support
Player Branding
Syndication Controls
External Publishing
Syndication Groups
+ New
Manage Syndication Groups for videos and channels. Syndication Groups can restrict domains on which your content can be embedded, geographic locations which can view the content, and dates/times when your content can be viewed.
Name
Delete
Default Group
Name Default Group
Syndication Controls (?)
Geographic Controls
Users cannot access content from
Type Country – Example: United States
Add Location
Domain Controls
Time Controls
Viewers can consume content during this specific range
The content will become available simultaneously around the world
Start Date 04/06/2012
5:03 PM (GMT -07:00)
End Date 04/06/2012
5:03 PM (GMT -07:00)
Device Controls
Require Ooyala Player Token
Expiration: (seconds)


OOYALA
BACKLOT
Help | Community | Support
ADD NEW CONTENT
NOW
MANAGE
MONETIZE
PUBLISH
ANALYZE
ACCOUNT
STATUS
CONTENT TYPES
Video
Ad
Audio
Live Stream
Remote Asset
Live Stream (audio)
YouTube Video
ACCOUNTS
CHANNELS
Winter 2013
Spring 2013
Summer 2013
Fall 2013
Winter 2012
CHANNEL SETS
ALL LABELS
Media Day
Behind the Scenes
Alumni
Mens
Womens
Bowerman
Recap
Highlights
Rename
Search
Delete
Pause
VIDEO
AUDIO
YOUTUBE VIDEO
REMOTE ASSET
LIVE STREAM
AD
CHANNEL
CHANNEL SET
TITLE
DESCRIPTION
CREATED
LENGTH
SIZE
STATUS
UCLA Recap
Watch highlights from this season.
Tue Oct 08 2013
00:02:27
231 MB
Men's Basketball Media Day
Check out exclusive behind the scenes footage of the 2013 media d
Mon Sep 30 2013
00:46:42
553 MB
Oregon Ducks: Behind the scenes
See coach Dana Altman and Johnathan Loyd speak at the 2013 Day
Mon Sep 30 2013
00:04:25
2.3 GB
WSU Women's hoops
"Down 1, up 4" for Oregon women's basketball
Mon Sep 30 2013
00:02:42
190 MB
Arizona Wildcats podium talk
See coach Sean Miller and guard Nick Johnson speak at the 2013 Day
Fri Sep 27 2013
00:11:09
792 MB
ITA Scholar athlete honors
Announcing candidates for the ITA Athlete Honors.
Fri Sep 20 2013
00:01:38
133 MB
Stanford almuni
Stanford almuni share stories and motivation for incoming players that
Thu Sep 19 2013
00:02:32
109 MB
Men's Singles Championship
Men's Singles Championship highlights
Thu Sep 19 2013
00:31:53
34 MB
Olympic Water Polo: Coach interview
Exclusive interview with Olympic Water Polo coach
Tue Sep 17 2013
00:01:36
134 MB
Stanford soars past Cal (Highlights)
Stanford soars past Cal (Highlights)
Tue Sep 17 2013
00:02:11
93 MB
Women's soccer players of the week
Five women's basketball players to watch on Pac-12 Networks in 2013
Fri Sep 13 2013
00:02:18
139 MB
Throwers raise money for local charity
Cal Track and Field Throwers philanthropy programs get team players i
Thu Sep 12 2013
00:00:29
101 MB
Five athletes names Bowerman finalists
The men's finalists include Arizona's Lawi Lalang and UCLA's Julian W
Thu Sep 12 2013
00:00:31
113 MB
NCAA Woman of the Year
The Woman of the Year Award honors graduating student-athletes who
Thu Sep 12 2013
00:00:37
137 MB
Track and field academic honors
Track and field academic honors
Thu Sep 12 2013
00:02:31
108 MB
Details
Embed
Monetize
Publishing Rules
Custom Metadata
Encodings
Resolution
Bitrate
Format
Device Compatibility
Video Codec
Audio Codec
1920 x 1080
14977 kbps
Single Encoding
Source
H.264
NONE
1280 x 720
4302 kbps
HTTP Streaming
H.264
AAC
1280 x 720
4200 kbps
Single Encoding
H.264
NONE
1280 x 720
2511 kbps
HTTP Streaming
H.264
AAC
1280 x 720
2400 kbps
Single Encoding
iPad, Android, , Desktop/Laptop
H.264
AAC
1280 x 720
2400 kbps
Single Encoding
H.264
NONE
640 x 360
1234 kbps
HTTP Streaming
iPhone, iPad,
H.264
AAC