Kollective ECDN 10.4 REST API Developer's Guide

Last published on:06/21/2018

Introduction to the Kollective REST API

This chapter provides prerequisite information and introduces the Kollective REST API and architecture. For a detailed list of specific API resources, see ECDN 10.4 Resource API Requests.

Before You Begin

Before beginning to develop a solution using the Kollective REST API, you should be familiar with web application development and have some knowledge of REST concepts, including how to make HTTP requests, construct URIs to access resources, and read WADL files.
You should also have a basic understanding of the Kollective SD ECDN and the objects used when managing and publishing content in

the Kollective system. Doing so will help you to understand how to work with Kollective objects and design your integration accordingly. This document assumes that your Kollective company is already created and working. Consult your Kollective representative for assistance.

Settings in Company Portal

Using the Kollective REST API requires a Kollective portal account. A portal account enables the configuration of certain application-level settings that are used by some API resources. The portal account is necessary even if you do not intend to use Kollective applications such as MediaCenter or VideoCenter.=

The application-level settings that are configured in MediaCenter or VideoCenter are:

- The groups associated with each authorization role. See Authorization Roles for details.
- To enable the approval feature. In MediaCenter, choose *Manage > Global Configuration* to modify this setting. In VideoCenter, choose *Admin > Configure VideoCenter*.
  If your company already uses MediaCenter or VideoCenter, you are all set. If not, contact your Kollective representative to have a portal account created.

About the Kollective REST API

The Kollective REST API is an HTTP-based interface that attempts to conform to the design principles of REST (Representational State Transfer). It supports the following basic HTTP operations: POST, GET, PUT, and DELETE.

WADL file

The Kollective REST API is defined by a WADL (Web Application Description Language) file. You can obtain the WADL file by entering the URL listed below in your browser.

```
https://hostName/api/v2?_wadl&_type=xml
```

where *hostName* is the hostname of your Kollective company.

You can obtain the WADL for search resources (OpenSearch and Kollective Feeds) by entering this URL:

```
https://hostName/api/search?_wadl&_type=xml
```

URI format

Construct an HTTP URI for accessing an API resource using the following format:

```
https://hostName/api/version/resourceRequest
```

where:

Parameter	Description
`hostName`	The hostname for accessing your Kollective company, such as api.kontiki.com. Contact Kollective Support for verify your hostName.
`version`	The version number of the API. Specify: **v2**
`resourceRequest`	The path to a resource name plus any required parameters. Resource API Requests provides a complete reference for all supported resource requests.

For example, the following HTTP GET request uses the metadata/content/list resource. It retrieves all live events, including their formats, for events titled, ‘Monthly Sales Meeting’.

```
https://api.kontiki.com/api/v2/metadata/content/list;formats=true?
filter=CONTENT_TYPE&value=LIVE_EVENT&q=Monthly%20Sales%20Meeting
```

Data Formats

The Kollective REST API supports XML, JSON, and JSONP data formats. By default, results are returned in XML format.
To receive JSON or JSONP formats in the response, you can specify the format either in the HTTP Accept header or in the _type request attribute.

To get results in	Set the Accept header to	Or add this to the request
JSON	`application/json`	`_type=json`
JSONP	`application/jsonp` and include a non-empty callback query parameter in the URL.	`_type=jsonp`

For PUT or POST requests, where you provide additional data to send with the request, you can specify the format in a Content-Type header.

NOTE: The Open Search API also supports the following data formats (use the key for *_type* parameter and *value* represents the mime type):

```
<entry key="xml" value="application/xml"/>
<entry key="rss" value="application/rss+xml"/>
<entry key="rss-plain" value="application/rss-plain+xml"/>
<entry key="atom" value="application/atom+xml"/>
<entry key="html" value="text/html"/>
<entry key="json" value="application/json"/>
```

See Open Search and Feed Resources for more information.

Example: XML format

This example specifies application/xml to get results in XML.

```
GET https://hostName/api/v2/metadata/content/list;formats=true HTTP/1.1
Accept-Encoding: gzip,deflate
Accept: application/xml
Authorization: Kontiki         cli-Sc9DF9/aDsbNW+9kX8WZiEXF0+KsxZFLiPLnBMb5Q2oEDt/
EoIQr+A==
User-Agent: Jakarta Commons-HttpClient/3.1
Host: hostName
```

Example: PUT request with Content-Type header

This example of a PUT request uses the Content-Type header to specify XML as the format for the request data. In this case, an Accept header is not required.

```
PUT https://hostName/api/v2/metadata/content/5c9dac9d-a090-4fea-ac16-
a73ecc96edd9 HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: application/xml
Authorization: Kontiki         cli-Sc9DF9/aDsbNW+9kX8WZiEXF0+KsxZFLiPLnBMb5Q2oEDt/
EoIQr+A==
User-Agent: Jakarta Commons-HttpClient/3.1
Host: hostName
Content-Length: 104
It includes the following content object with a new title for the video.
<ns:content xmlns:ns="http://api.kontiki.com">
    <title>My video's new title</title>
</ns:content>
```

Example: JSONP Format

This example specifies application/xml and includes a callback parameter to get results in JSONP.

```
GET https://hostName/api/v2/metadata/content/list;formats=true?callback=foo HTTP/1.1
Accept-Encoding: gzip,deflate
Accept: application/json
Authorization: Kontiki         cli-Sc9DF9/aDsbNW+9kX8WZiEXF0+KsxZFLiPLnBMb5Q2oEDt/
EoIQr+A==
User-Agent: Jakarta Commons-HttpClient/3.1
Host: hostName
```

Example: JSON format with _type attribute

This examples uses the _type attribute to specify JSON at the format.

```
https://api.kontiki.com/auth/check?auth=cli-38i8A67GZPmBdjrN8KaGhbnvNQXQYi2Affl9bY8db/
yk3JB/l0Iplg==&_type=json
```

MOIDs

Objects in the Kollective system, including content items, uploaded files, and channels, are identified by statistically unique MOIDs (Media Object Identifiers) that look something like this:

```
1ca2e3c1-50e0-09d8-0eb1-23d528283918
```

Many API resource requests take a MOID as a parameter to identify the content, channel, format or other item you are working with. Responses that include a Kollective object will also include its MOID.
This example is a DELETE request that deletes a content identified by its MOID.

```
https://hostname/api/v2/metadata/content/40fd281c-1b0b-497c-8177-8a1baa63d11f
```

Special characters

The values of some resource parameters (such as usernames, group names, or tags) may contain special characters, including / or %. When constructing the URI for a resource, any parameters containing special characters must be URL encoded with the UTF-8 character set. If the encoded string also contains special characters, it should be double encoded.

Use the following Java function to ensure that the string value is suitably encoded:

```
encodeURIComponent(str)
```

JavaScript interprets the \ character as an escape and give inconsistent results.

Example:

```
encodeURIComponent('a\b')

= a%08
```

The \b is a special character and thus it is encoded. If the literal string is *a\b*, the code should be:

```
encodeURIComponent('a\\b')
= a%5Cb
```

⚠ An error is not thrown when invalid escape sequence is used, the character is simply stripped. Example:

`encodeURIComponent('a\c')`

`= ac`

Handling cross-domain restrictions in JavaScript

Developers working with the Kollective REST API using client-side JavaScript may encounter security restrictions arising from the "same-origin policy" in browsers. These restrictions make it difficult to access resources across different domains. This section describes a way to make API requests using client-side JavaScript that can overcome these restrictions.
The key to overcoming cross-domain restrictions is to make all resource requests (even those with PUT, POST or DELETE) using the HTTP GET method. The Apache CXF framework on which the Kollective REST API is based provides certain attributes that allow overriding the HTTP method of a request.

`_method`	Overrides the HTTP operation method. You can make a non-GET request as a GET by specifying the actual method for the request using _method.
`_ctype`	Specifies the content type of the data sent with the request, either XML or JSON.

For example, the request to delete a comment for a video uses the DELETE method. You can send this as a GET request and override the GET method by specifying the _method=DELETE parameter as follows:

```
GET https://hostname/api/v2/metadata/content/MOID/comment/1?
auth=authTokenKey&_method=DELETE
```

Other HTTP operations, such as PUT or POST, often require additional data or parameters included in the request body.
To support making these requests with a GET operation, the REST API provides the following attributes that let you include the data in the URL:

`_data`	Specifies the Base64-encoded request body data.
`_zdata`	Specifies the Base64-encoded compressed request body data. After compressing the request body, Base64-encode the zip entry portion of the compressed data.

The following example programs a "Create" button to create a content item. It uses the JQuery JavaScript library to construct the URL and call the resource, and JSZip to compress the request body data.

```
<script src="javascripts/jquery-1.6.4.min.js"></script>
<script src="javascripts/jquery.base64.js"></script>
<script src="javascripts/jquery.cookie.js"></script>
<script src="javascripts/jszip.js"></script>
<script src="javascripts/jszip-deflate.js"></script>
<script src="javascripts/json2.js"></script>
<script>
....
$("CreateButton").click(function() {
        // define Content and Format
         var upload = {content:{
                contentType:"CONTENT",
                title:”My Video”,
                description:”This is my video.”,
                format:{
                        uploadMoid:MOID,
                        bitRate:300000,
                        height:360,
                        width:480
                          }
                }
             };
        // Convert to JSON
         var data = JSON.stringify(upload);
        // Compress using JSZip
        var zip = new JSZip();
        zip.file("data", data);
        var zdata = zip.generate({compression:"DEFLATE"});
        $.ajaxSetup({ error:handleAuthError});
        // Call the create content resource and return the MOID
        $.getJSON( "https://hostname/api/v2/metadata/content?_method=POST&_ctype=json&_zdata="+encodeURIComponent(zdata) +
"&auth=tokenKey" + "&callback=?",
          {},
          function(data) {
        $("ContentMoid").val(data.contentResult.content[0].moid);
          });
      });
....
</script>
```

Authentication

The Kollective REST API uses token-based authentication to provide access to API resources. You typically provide the token key in the HTTP Authorization header or as a parameter in the URL or post body.
Kollective supports three types of authentication tokens, each with different levels of access: Service, Client, and Public.

Service tokens

Service tokens are special administrative tokens constructed using an ID that you obtain from Kollective Support. You might use service tokens when setting up your company or performing other administrative tasks.

Service tokens:

- Begin with the **srv-** prefix.
- Do not expire.
- Do not require login; they can be used directly with any API request.
- Can impersonate another user.
- Require SSL when making API requests.

Each service token ID is associated with a user who is a member of a group that has the Administrator role; API requests are made as that user. A service token can also impersonate another user.

You construct a service token as follows:

`srv-base64EncodedString`

where base64EncodedString consists of

`serviceTokenId[#act_as_username]`

`serviceTokenId`	A string that you obtain from Kollective Support that is associated with a user who is a member of a group that has the Administrator role.
`act_as_username`	*Optional*. A username of a user to impersonate. All transactions will be performed under this username. The authentication token will also inherit the role assigned to this user.

For example, if the service token ID you obtained from Kollective Support is:

`b5194edc-07e5-4bd1-ad24-1e8c49e0c738`

You may encode it (using any Base 64 encoder) and construct the following service token:

`srv-YjUxOTRlZGMtMDdlNS00YmQxLWFkMjQtMWU4YzQ5ZTBjNzM4`

To use your service token to impersonate Joe User, you may encode the following string:

`b5194edc-07e5-4bd1-ad24-1e8c49e0c738#joeuser`

To construct this service token:

`srv-YjUxOTRlZGMtMDdlNS00YmQxLWFkMjQtMWU4YzQ5ZTBjNzM4I2pvZXVzZXI=`

You can then use the service token with any resource request. No login is required. This example of a request to retrieve a list of content items includes the service token key in the URL:

```
https://hostname/api/v2/metadata/content/list?auth=srv-
YjUxOTRlZGMtMDdlNS00YmQxLWFkMjQtMWU4YzQ5ZTBjNzM4I2pvZXVzZXI=
```

Client tokens

Client tokens provide authenticated access to users from a specific company. These tokens are generated by the Kollective system and returned as part of the authToken object when you use the auth/login resource. See Log in to Kollective for details on this resource.

Client tokens:

- Begin with the **cli-** prefix.
- Expire in a configurable amount of time; the default is 8 hours.
- Can impersonate another user if the authenticated user has the Administrator role. (See the actAsUser parameter of the auth/login resource.)
- Require SSL when making API requests.

The information you provide to the auth/login resource to obtain the client token can vary depending on the circumstance. In the typical case, you provide a username and password along with the company name or company prefix. The username and password must be encoded using any Base 64 encoder.

For example, to authenticate Joe User from My Company, you may encode the following credentials:

```
joeuser:h32JHz88
```

to yield this encoded string:

am9ldXNlcjpoMzJKSHo4OA==

and construct following auth/login GET request:

```
https://hostname/api/v2/auth/login?
credentials=am9ldXNlcjpoMzJKSHo4OA==&companyName=mycompan
```

If the login is successful, the request returns an **authToken** object containing a token key, a string that is required for all subsequent API requests.

```
<ns2:authToken xmlns:ns2="http://api.kontiki.com">
  <tokenKey>cli-17dfa0c3-a4d7-4614-bca8-1a68e7ad54c2</tokenKey>
  <validityEndDate>2012-01-11T05:56:47.328Z</validityEndDate>
  <userEmail>joeuser@mycompany.com</userEmail>
  <userFirstName>Joe</userFirstName>
  <userLastName>User</userLastName>
  <username>joeuser</username>
  <companyId>2600</companyId>
  <sessionID>btn7eh4fpod2q0l</sessionID>
  <granted-role>ROLE_PORTAL_VIEWER</granted-role>
  <granted-role>ROLE_WIDGET_VIEWER</granted-role>
</ns2:authToken>
```

You can now access any other API resource by specifying the token key along with the resource. You typically provide the token key in the HTTP Authorization header or as a parameter in the URL or post body.
This example of a request to retrieve a list of content items includes the token key in the URL:

```
https://hostname/api/v2/metadata/content/list?
auth=cli-38i8A67GZPmBdjrN8KaGhbnvNQXQYi2Affl9bY8db/yk3JB/l0Iplg==
```

This example shows the same request with the client token key in the Authorization header:

```
POST https://hostname/api/v2/metadata/content/list HTTP/1.1
Accept: application/xml
Content-Type: application/xml
Authorization: Kontiki         cli-38i8A67GZPmBdjrN8KaGhbnvNQXQYi2Affl9bY8db/
```

```
yk3JB/l0Iplg==
Host: hostname
```

Public tokens

Public tokens provide anonymous or guest access. They are useful for accessing publicly available content that has no group restrictions. A user accessing your content with a public token is assigned the Widget Viewer role.

Public tokens:

- Begin with the **pub-** prefix.
- Are not authenticated; they can be used directly with any API request that allows the Widget Viewer role.
- Do not require SSL when making API requests.

A public token key includes the user's email address and Kollective company prefix. Construct a public token as follows:

`pub-base64EncodedString`

where **base64EncodedString** consists of

`emailAddress#CompanyPrefix[#optionalData]`

`emailAddress`	Any email address. It is used only for logging purposes.
`CompanyPrefix`	Your Kollective company prefix as provided by Kollective Support.
`optionalData`	Any optional data you want to include. This may be useful for access logging.

For example, you may encode the following string (using any Base 64 encoder):

`joepublic@anycompany.com#myco`

to construct this public token, use this format:

`pub-am9lcHVibGljQGFueWNvbXBhbnkuY29tI215Y29tcGFueQ==`

Global Login Service

The Global Login Service provides a central web service for authentication across Kollective applications (MediaCenter, Analytics, embed objects, VideoCenter,), as well as implementations using the REST API. The Global Login Service handles authentication from all supported sources of user directories and integrations, including LDAP, single sign-on (SSO), and SAML, as well as the Kollective LoginManager.

To enable centralized authentication, the Global Login Service employs a temporary authentication key called a valet key. After authentication with the GLS, a valet key is generated that can be used for access to other Kollective applications. The valet key is valid for only one minute and only one authentication attempt.

The general procedure for authentication using the Global Login Service using the REST API is as follows:

1. Call the **auth/login/url** resource to get the Global Login Service URL that is returned.
2. Direct the browser to the Global Login Service URL.
3. The user provides credentials and is authenticated.
4. After authentication, the browser is automatically redirected to the callback URL you specified with the auth/login/url resource. The URL includes a **valetKey** parameter with a temporary session ID that is valid for only one minute and one authentication attempt.
5. Get the value of the **valetKey** parameter from the URL and call the auth/login resource, passing the **valetKey** value as the **webSessionId**. The resource returns a client token in the **tokenKey** property of the **authToken** object.
6. Use the **tokenKey** value in the **auth** parameter for authentication in subsequent API calls.

The following code, using the JQuery JavaScript library, provides an example of this procedure.

```
<script src="javascripts/jquery-1.6.4.min.js"></script>
<script src="javascripts/jquery.base64.js"></script>
<script src="javascripts/jquery.cookie.js"></script>
<script src="javascripts/json2.js"></script>
```

```
function loginToCompany() {
  // clear session Id cookie
  $.cookie('sid', '');
  $.ajaxSetup({ error:handleAuthError });
  // replace with your hostname and company prefix
  var api_url = 'hostName';
  var company_prefix = 'companyPrefix';
  var url = document.location.href;
  // call resource to get the GLS URL
  $.getJSON( api_url + "/api/v2/auth/login/url?companyPrefix="+company_prefix+
"&callbackURL=" + encodeURIComponent(url) ,
    {},
    // Pull url from response and redirect there
    function(data) {
    document.location.replace(data.authIdentityUrlInfo.url);
    });
}
var valetKey = 'valetKey=';
var tokenKey;
function findTokenBySessionIdOrValetKey() {
  var sid = $.cookie('sid');
  if (!sid) {
     // parse valetKey from the URL
     var url = document.location.href;
     var i = url.indexOf(valetKey);
     if (i>0) {
        sid = url.substring(i + valetKey.length);
     }
  }
  if (sid) {
     $.ajaxSetup({ error:handleAuthError });
     var api_url = 'hostName';;
     // log in using valetKey as the webSessionId
     $.getJSON( api_url + "/api/v2/auth/login?webSessionId=" + sid ,
       {},
        // retrieve the tokenKey from the response
       function(data) {tokenKey=data.authToken.tokenKey;});
  }
}
```

Authorization Roles

Access to API resources is based on the authorization roles assigned to the user making the request.

When working with Network Publisher content or live events, the Administrator role is required for any resource that creates or modifies data. For MediaCenter content, the required role is noted in the description for each resource in Resource API Requests..

You set up the association between user groups and authorization roles using MediaCenter. In VideoCenter, an administrator chooses **Configuration** > **User Management** to configure the roles and groups.
See the *MediaCenter Administrator's Guide* for details on the authorization roles and capabilities granted to each role.

API Features

For a general description of some REST API features that are implemented in MediaCenter, refer to the MediaCenter help. This topic provides details for using the REST API to implement certain features where applicable.

Date/Time Availability

The Availability settings let video owners, content managers, and administrators designate when a video is available for viewing by others. For details on how the availability settings work, see the MediaCenter help.
Using the REST API, you specify the from and until dates in properties of the content object:

- Set the from date in the **availabilityStartDate** property.
- Set the until date in the **availabilityEndDate** property.

Subtitles

Subtitles (sometimes called captions or closed captions) display a text version of words as they are spoken in the video. For an example of how to implement subtitles, see the MediaCenter Help.
The REST API supports these subtitle file formats:

Extension	File Type
`.SRT`	SubRip
`.VTT`	WebVTT

To work with subtitles using the REST API:

- First use any of the Upload Resources to upload the subtitle file, and obtain the MOID of the upload.
- If the content is not already created, you can specify the subtitle's upload MOID in the subtitlesUploadMoid property of the Content object when you create the content. See Create a new content item.
- If the content already exists, call the subtitles resource to associate the uploaded subtitle file to the content. See Assign a subtitle file to a content item for more information.

Send Error Logs

This call sends logs to Kollective Support, in the same format and fashion as the Send Logs function available in Kollective agents.

```
http://127.0.0.1:31013/api/v1/sendErrorLogs
```

The following JSON reply is returned upon success:

```
{ "method" : "sendErrorLogs", "returnCode" : "200" }
```

Schema Reference

This section provides a reference for the most common data objects used by the REST API.

authToken

Property	Type	Description
`companyId`	long	For Kollective internal use.
`granted-role`	string	One or more authorization roles assigned to this user. See Authorization Roles.
`persistent`	boolean	Indicates whether the authorization token persists after the user closes the browser window. The value comes from the *persistentSession* property of **globalConfig**.
`sessionId`	string	The web session id, available when a user is already logged in via the Global Login Service.
`tokenKey`	string	The authorization token key. You provide this key in subsequent resource requests after authenticating. See Authentication.
`userEmail`	string	The user's email address.
`userFirstName`	string	The user's first name or given name.
`userLastName`	string	The user's last name or surname.
`username`	string	If the user is authenticated, this is the username. For public access, this is an identifier for the user.
`validityEndDate`	dateTime	The date and time when the authentication key will expire, expressed in UTC (Coordinated Universal Time). When persistent is true, this value is set based on the value of the *persistentSessionLimitHours* property of **globalConfig**.

channel

Property	Type	Description
`bannerUrl`	string	*Read-only.* The URL for accessing the banner image displayed for the channel. To add or change the image, see *Add a banner image for a channel*. To remove the image and revert to the default image, see Remove the banner image from a channel.
`createDate`	dateTime	*Read-only.* The date and time when the channel was created, expressed in UTC (Coordinated Universal Time).
`description`	string	*Optional.* Description of the channel's content.
`ext-metadata`	child object	An external metadata item. Properties include: *id* — The name of the metadata field. *delete* — Specify true to delete this item.
`feedPollInterval`	long	The interval in seconds at which to check for new feed content for this channel. Specify a number of seconds from 1800 (30 minutes) to 2592000 (30 days). The default is 43200 (12 hours).
`feeds`	child object	The feed targets for the channel. Properties include: *group* — A group defined in the user directory. *profile* — A client profile identifier, a MOID. *mask* — A range of IP addresses specified in CIDR notation. See Update feed targets for a channel.
`inFavorites`	boolean	*Optional.* Specifies whether the channel is a favorite of the current user.
`isEmbedAllowed`	boolean	*Optional.* Specifies whether the code for sharing of this channel as an embedded widget appears in VideoCenter. This has no effect on operations with the API. Default is true.
`isMediaCenterHome`	boolean	*Optional.* Specifies whether the channel is designated as a possible home channel for MediaCenter.
`isMediaCenterHomeDefault`	boolean	*Optional.* Specifies whether the channel is the home channel for MediaCenter.
`moderated`	boolean	Specifies whether the channel has at least one user assigned with a Channel Moderator role.
`moid`	string	*Read-only.* The Media Object Identifier for the channel. See MOIDs.
`name`	string	*Required.* The name of the channel.
`pendingAssignment`	boolean	When channel moderating is enabled (the contentChannelModerated property of globalConfig is true), this indicates whether content has been added to the channel but not yet approved by a channel moderator. This property is typically used when retrieving a list of channels to which a content item has been added using the channels matrix parameter. (See Get metadata for a content item.)
`permission`	child object	*Read-only.* The permissions associated with the channel, including who the channel owner is, and which groups are allowed to view it. If a channel has multiple permissions, permission is repeated. Properties include: *user* — A user with the permission specified by role. *group* — A group with the permission specified by role. *role* — Either CHANNEL_OWNER, CHANNEL_VIEWER or CHANNEL_CONTRIBUTOR. See *Add groups that can view a channel.*
`related-upload`	upload	*Read-only.* This object contains upload information about the thumbnail image associated with this channel. It is applicable only for thumbnails uploaded through the API (not through VideoCenter) within the last 30 days; uploads expire and are removed after 30 days.
`showName`	boolean	Specifies whether to display the channel name along with the banner image.
`template`	string	The name of a display template associated with the channel. If empty, the Global template is used. See the MediaCenter Administrator's Guide for details on templates.
`thumbnailUrl`	string	*Read-only.* The URL for accessing the thumbnail image representing the channel. To add or change the image, see Add a thumbnail image for a channel. To remove the image and revert to the default image, see Remove the thumbnail image from a channel.

Property	Type	Description
`type`	string	*Read-only.* *EDITORIAL* — All channels created by users have this type. *HIGHEST_RATED* — This built-in channel lists the videos that receive the highest average rating. A video must have ratings from at least five different users to qualify for this channel. *MOST_DISCUSSED* — This built-in channel lists the videos that receive the most comments. A video must have comments from at least five different users to qualify for this channel. *RECENTLY_ADDED* — This built-in channel lists videos in reverse order of when they were published. *USER* — No longer used.
`updateDate`	dateTime	*Read-only.* The date and time when the channel metadata was last modified, expressed in UTC (Coordinated Universal Time).
`watcher`	child object	A user who has added the channel to favorites, or a group that was subscribed to the channel by an administrator. Properties include: *group* — A group subscribed to the channel. See Subscribe groups to a channel. *user* — A user who has added the channel to favorites. See Add a channel to favorites. *mandatory* — Indicates whether the subscription is mandatory.

content

Property	Type	Description
`approved`	boolean	*Read-only.* When the video approval feature is enabled in VideoCenter, this property indicates whether the content has been approved. To approve an item, see Approve a content item.
`archiveMoid`	array of strings	Valid only for live events. The Media Object Identifiers for the archives of the event. See MOIDs.
`archivePending`	boolean	Valid only for live events. When true, creation of a new archive content item for the event is pending.
`availabilityEndDate`	dateTime	Optional. The date and time when the content becomes unavailable for viewing, expressed in UTC (Coordinated Universal Time). See Date/Time Availability for details.
`availabilityEndDateClear`	boolean	Valid only for update. Set to true to clear the value of *availabilityEndDate*.
`availabilityStartDate`	dateTime	*Optional.* The date and time when the content becomes available for viewing, expressed in UTC (Coordinated Universal Time). See Date/Time Availability for details.
`availabilityStartDateClear`	boolean	Valid only for update. Set to true to clear the value of *availabilityStartDate*.
`chapter`	child object	*Optional.* Specifies details for each chapter in a video. For each chapter, properties include: *start* — start time of the chapter, in seconds. *title* — title of the chapter. *description* — description of the chapter contents. *thumbnailUrl* — URL for a thumbnail image representing the chapter. *uploadName* — name of thumbnail file representing the chapter. *uploadMoid* — unique ID returned for thumbnail file from upload call.
`commentsAllowed`	boolean	*Optional.* Specifies whether comments may be added for the content in VideoCenter. This has no effect on operations with the API. The default is false.
`contentStatusType`	string	*Read-only.* A code indicating the status of the content. After a video is published, the file is processed before being made available for viewing. Processing can include transcoding to the proper format, generating thumbnails, scanning for viruses, and other steps.
`contentType`	string	*Required for creation, then read-only.* *CONTENT* — A video or audio content. All VideoCenter content has this type. *CONTENT_LEGACY* — An item published through the Network Publisher application. *LIVE_EVENT* — A content item representing the captured stream of a live broadcast event.
`created`	dateTime	*Read-only.* The date and time when the content was published, expressed in UTC (Coordinated Universal Time).

Property	**Type**	**Description**
creator	string	*Read-only.* The username of the person who published the content.
description	string	*Required.* The description of the content.
dscp	byte	*Optional.* For live events, you can specify a DSCP (differentiated services code point) so that your IT professionals can prioritize the event stream traffic within your network for improved QoS (Quality of Service) during important events.
durationMS	long	*Optional.* The playing time of the content in milliseconds.
emailAllowed	boolean	Optional. Specifies whether sharing the content by email is allowed in VideoCenter. This has no effect on operations with the API. Default is false.
embedAllowed	boolean	*Optional.* Specifies whether the code for embedding the content appears in VideoCenter. This has no effect on operations with the API. Default is true.
encoderConnectionMethod	string	Optional for creation of live events, then read-only. The method by which the captured event stream is transmitted from the encoder to the Kollective grid server. *PUSH* — Encoder pushes to Kollective via Windows Media over HTTP. *WOWZARTMP* — Encoder pushes to Kollective via RTMP with RTMP client playback. *PULL* — Kollective pulls from Encoder via Windows Media over HTTP. See Push or pull distribution for more information.
encoderBackupURL	string	Required if **encoderConnectionMethod** is set to *WOWZARTMP*. The secondary URL of the Kollective server to which the stream will be pushed.
encoderFmsURL	string	Required if **encoderConnectionMethod** is set to *WOWZARTMP*. The primary URL of the Kollective server to which the stream will be pushed.
encoderMoid	string	The MOID of the registered encoder associated with this live event. Required only if the event links to a registered encoder.
encoderPushAuthPassword	string	Required. For push connections (**encoderConnectionMethod** is *PUSH* or *WOWZARTMP*) this is the password the encoder will use when accessing the Kollective grid server. For pull connections, this is not used but a dummy value must still be specified.
encoderPushAuthUserName	string	Required. For push connections (encoderConnectionMethod is *PUSH* or *WOWZARTMP*) this is the username the encoder will use when accessing the Kollective grid server. For pull connections, this is not used but a dummy value must still be specified.
encoderPushHost	string	Required if **encoderConnectionMethod** is set to *PUSH*. The name of the Kollective server to which the stream will be pushed.
encoderStreamName	string	Required if **encoderConnectionMethod** is set to *WOWZARTMP*. The name of the stream to push to the FMS URL. This is formatted as liveEventContentMOID_%b where _%b is a literal value representing the bit rate of the stream. The Adobe Flash Media Live Encoder interprets this value and replaces it with the appropriate bit rate.
encrypted	boolean	*Optional* for creation, then read-only. Indicates whether the content should be encrypted.
eventEnd	dateTime	*Optional.* The date and time when the event is scheduled to end, expressed in UTC (Coordinated Universal Time). The value can be any time after the eventStart date & time. If the actual event continues beyond the eventEnd time, current viewers may continue watching but no new viewers can join the event. Leave blank for a continuous live feed. If an event's end time is variable or unknown, enter what is likely to be the latest end time. If the event ends sooner, be sure to close the event promptly by updating the eventEnd time. For the best user experience and most accurate reporting data, it is preferable to overestimate an event's length and close it manually than to have an event last beyond its scheduled end time.
eventEndClear	boolean	Valid only for update. Set to true to clear the value of eventEnd.
eventNoAuth	boolean	Valid only for live events. When true, the event does not require authentication for publishing from encoders.
eventStart	dateTime	Required for live events. The date and time when the event will start, expressed in UTC (Coordinated Universal Time).
extension	string	The extension of the file associated with this content item.
externalId	string	*Optional*. A value to identify this file in the system from where it originated.
externalPlayerTarget	string	The name of the target window in which to open the player specified by **externalPlayerUrl**. The value is initially NULL; if you set the value, then remove it, the value becomes an empty string. This is provided for third-party integration. Kollective applications currently do not make use of this property.
externalPlayerUrl	string	The URL of an external player for playing the content. The value is initially NULL; if you set the value, then remove it, the value becomes an empty string. This is provided for third-party integration. Kollective applications currently do not make use of this property.

Property	Type	Description
flag	child object	A **flaggedMessage** object representing one flag of either the content or a comment on the content. Properties include: *author* — The user who flags the content or comment. *created* — The date and time the flag was created, expressed in UTC (Coordinated Universal Time). *text* — The text of the flag explaining the objection. *comment* — The comment being flagged. If the flag is for the content, leave this blank. Related resources: Flag a content item, Flag a comment, Remove a flag.
flagged	boolean	Indicates whether the content item has been flagged. Related resources: Flag a content item
icon	string	Full path to one of the possible thumbnail images for the video. When multiple thumbnail images are generated, the icon property repeats. The path to the chosen thumbnail image is stored in the **thumbnailUrl** property.
icon128x72	string	*Read-only.* Full path to the small icon used for display in VideoCenter. Used only when contentType is CONTENT.
icon640x360	string	*Read-only.* Full path to the large icon used for display in VideoCenter. Used only when contentType is CONTENT.
ignoreTranscoding	boolean	Optional for creation, then read-only. Causes the video to be published without transcoding. Applies only to VideoCenter content. Default is false; if set to true, the type property in any associated format objects must be set to a valid value.
internalDescription	string	Optional. This field appears in MediaCenter only if enabled by an administrator. It does not appear on content viewing pages and is visible only to administrators, content managers and contributors. It is searchable from the Search box, but the results page does not show the field.
mediumIcon	string	Read-only. Full path to the medium-sized icon for display in Network Publisher. Used only when contentType is CONTENT_LEGACY or LIVE_EVENT.
modified	dateTime	Read-only. The date and time when the content was last modified, expressed in UTC (Coordinated Universal Time).
modifiedBy	string	The username of the person who last modified the content.
moid	string	Read-only. The Media Object Identifier for the content item. See MOIDs.
rating	ratingInfo	Read-only. These values are maintained automatically when you add ratings. See Add a rating to a content item. *count*—The number of unique ratings. If a person rates more than once, only the latest rating is counted. *average*—The average rating.
ratingsAllowed	boolean	Optional. Specifies whether a rating may be added for the content in VideoCenter. This has no effect on operations done with the API. Default is false.
related-upload	upload	This object contains upload information about a file associated with this content item, including the content file or a subtitle file.
showInGuide	boolean	Deprecated.
stopped	boolean	Valid only for live events. When true, the event has been stopped.
subtitlesStatus	string	When a subtitle file is uploaded with a content, it is processed before becoming ready to use. Possible values: *none* — No subtitle file. *scan* — The subtitle file is being processed. *ready* — Processing is completed; the subtitle file is ready.
subtitlesUploadMoid	string	The MOID of the upload that is the subtitle file for this content.
smallIcon	string	Read-only. Full path to the small-sized icon for display in Network Publisher. Used only when contentType is CONTENT_LEGACY or LIVE_EVENT.
tag	string	Optional for creation, then read-only. A tag associated with the content. If a content has multiple tags, the tag property is repeated. See Add tags to a content item.
tagsAllowed	boolean	Optional. Specifies whether tags may be added for the content in VideoCenter. This has no effect on operations with the API. Default is false.
thumbnailUrl	string	Full path to the chosen thumbnail image for the video.
template	string	The name of a display template associated with the channel. If empty, the Global template is used. See the MediaCenter Administrator's Guide for details on templates.
title	string	Required. The title of the content.
transcoderEnd	long	The end point in seconds of the transcoding of the content.

Property	Type	Description
`transcoderProfileCode`	string	A code indicating the profile to use when transcoding the video for playback in VideoCenter or the Kollective client. See Get all transcoder profile codes.
`transcoderStart`	long	The starting point in seconds of the transcoding of the content.
`transcoderVideoFilterCode`	string	A code indicating the video filter to use when transcoding the video. See Get all transcoder video filters.
`transcodeStatus`	string	The transcodeJobInfo object details the status of a transcoding.
`urn`	string	Read-only. The uniform resource name for accessing the content.
`unlisted`	boolean	Indicates whether the content should be excluded from display in MediaCenter. When true, the content does not appear in search results and channel listings. The content can still be accessed if a user already has a link, or has downloaded it, but new users cannot discover it through MediaCenter.
`views`	long	The number of times the content was played. This value may not includes plays occurring within the last 10 minutes.
`channel`	child object	Objects representing each of the channels where the content is found.
`format`	child object	Objects representing each of the formats associated with the content.
`permission`	child object	Read-only. The permissions associated with the content, including who the content owner is, and which groups are allowed to view it. If a content has multiple permissions, permission is repeated. Properties include: *user* — The user who created the content. *group* — A group with permission to view the content. *role* — Either CONTENT_OWNER or CONTENT_VIEWER. See Assign groups permitted to view a content item.
`ext-metadata-list`	child object	Read-only. Object containing one or more external metadata fields associated with the content. Properties include: *id* — The name of the metadata field. *delete* — Specify true to delete this item.

`related-link-list`	child object	*Read-only.* Object containing one or more related links associated with the content. Specify the list as follows:
		`<ns:related-link-list xmlns:ns="http://api.kontiki.com">`
		  `<related-link text="linkText">`
		    `linkURL`
		  `</related-link>`
		`</ns:related-link-list>`

`comment`	child object	Object containing a comment associated with the content. See Add a comment to a content item. Properties include: *author* — Read-only. Username of the person who wrote the comment. *created* — Read-only. The date and time when the comment was last modified, expressed in UTC (Coordinated Universal Time). *text* — The text of the comment.

encoder

Property	Type	Description
`moid`	string	Read-only. The MOID of this encoder definition. See MOIDs.
`encoderId`	string	An identifier for the encoder. Specify any unique string.
`serviceProvider`	string	The manufacturer of the encoder, such as Adobe or Telestream, or the name of the provider of the stream source.
`serviceName`	string	The model or version number of the encoder, or the service name given by the stream source provider.
`serviceId`	string	The serial number of the encoder, or the identifier given by the stream source provider.

Property	Type	Description
`streamCount`	long	The number of streams this encoder will send. At the encoder, you can configure each stream with a different combination of bit rate and resolution.
`authRequired`	boolean	Specify true to provide authentication credentials for accessing the Kollective server. Specify false to allow the encoder to access the Kollective server without providing credentials.
`authUsername`	string	Required when authRequired is true. The username that the encoder will use to access the Kollective grid server.
`authPassword`	string	Required when authRequired is true. The password must be at least 10 character long and have one or more of each of following categories: uppercase letter, lowercase letter, number, and non-alphanumeric character. Use only characters from the 7-bit ASCII character set.
`encoderFmsURL`	string	Read-only. The URL of the Kollective server to which the stream will be pushed.
`encoderBackupURL`	string	Read-only. The secondary URL of the Kollective server to which the stream will be pushed.
`activeLiveEvent`	child object	One or more content objects representing live events linked to this encoder.

format

The minimum required properties for a format differ depending on the type of content it represents. For details see *Add a format to a content item*.

Property	Type	Description
`bitRate`	long	The bit rate in kbps at which a video or live event stream is encoded. For non-video or live content, set this value to -1.
`contentMoid`	string	Read-only. The MOID of the content item associated with this format. See MOIDs.
`contentType`	string	The MIME type of the file. This is set automatically based on the file extension. If the item was published and transcoded by VideoCenter (that is, the associated Content object has a *contentType* of CONTENT and *ignoreTranscoding* is false), this property is read-only. Otherwise, it is editable.
`created`	dateTime	Read-only. The date and time when the format was created, expressed in UTC (Coordinated Universal Time).
`encodingInfo`	string	*Optional.* The type of video encoding. Options are: *SMOOTH* - Microsoft Smooth streaming *DASH* - MPEG DASH *HLS4* - Apple HLS v4, used for Azure integration. *HLS3/HLS* - Apple HLS v3, used for Azure integration.
`fileChecksumMD5`	string	Optional, then read-only after READY status. The MD5 checksum value for the file.
`fileNameHint`	string	Read-only. The user-friendly display name for the file, used by the Kollective client.
`fileSize`	long	Optional, then read-only after READY status. The file size.
`height`	int	Optional. The height in pixels of a video file.
`modified`	dateTime	Read-only. The date and time when the format was last modified, expressed in UTC (Coordinated Universal Time).
`moid`	string	Read-only. The MOID of the format. See MOIDs.
`originalStreamsInfo`	string	Information about the stream as detected from the file metadata.
`publishingPointUrl`	string	Read-only. The URL where the live event stream is sent from the encoder for push distribution.
`redundantUrl`	string	Optional. The secondary URL for accessing the live event stream when the primary url is unavailable.
`streamName`	string	Optional. The stream name for a live event format.

Property	Type	Description
`type`		See Create a new content item for requirements. *ORIGINAL* — Format for the original uploaded content. *PRIME* — For video content that has been trimmed. *NETPUB* — For content published through Network Publisher. *LEGACY* — For playback in the Kontiki Media Manager client. *IOS* — For playback in an iOS device. A content may have multiple IOS type formats with different bit rates and resolutions for each device. *NON_IOS_MOBILE* — For playback in a mobile device other than iOS. *PROGRESSIVE* — For playback on a Windows phone. *CLIENTLESS* — For playback from Wowza Media Server. *EXTERNAL*
`uploadMoid`	string	The MOID of the uploaded file associated with this format. See MOIDs.
`url`	string	The primary URL for accessing the live event stream. See also redundantUrl.
`width`	int	Optional. The width in pixels of a video file.

health

There are two available calls, **status** and **license:**

`GET  /api/v2/health/status`

Returned structure contains the following information:

state - (**ok, warning, failure, panic**)

node - a list of degraded or interrupted services (**service type, host, problem code, last time of success report**)

`GET  /api/v2/health/license/{key}`

Returned structure contains the following information:

state - (ok, warning, failure, panic)

license - license key

The **health** check returns the following JSON:

```
{"status" : "ok" , "stats" : "Summary: items=4 cached=4 bytes=0 scoreMin=1.000000000e+36
scoreCutoff=-1.000000000 scoreMax=1.000000000e+36 Lowest cached: 2aa034f4-e1bc-4308-
a668-78085859d66c:u9YC+ifUl5zo
ST_OK filled:8897288/-1 connections:2 starttime:2016-02-01 14:10:40.082 score:
1.000000000e+36 requests:1.000000000 "}
```

Only the **status** key of the above JSON is consumed. The other keys are meant for troubleshooting purposes only.

panel

Property	Type	Description
`isFilteredByChannel`	boolean	If true, only items that belong to the channel appear on the panel.
`layout`	string	A code representing the layout of items in the panel. Possible codes: `D1S1, D1Q4, Q4S1, Q4Q4, D1S1, S1Q1, S1Q4, S1S1, M1`
`moid`	string	The MOID of the panel.
`name`	string	The display name of the panel.
`order`	long	Number indicating the order in which the panel appears on a page that has multiple panels.

Property	Type	Description
parentChannelMoid	string	The MOID of the channel, if it is a media page panel.
parentChannelName	string	The name of the channel, if it is a media page panel.
parentContentMoid	string	The MOID of the content item, if it is a content page panel.
parentContentName	string	The name of the content item, if it is a content page panel.
query		A search term to filter the items.
sortField		The field on which panel items are sorted. Specify *DURATION, NAME*, or *LAST_MODIFIED_DATE*
sortOrder		Specify ASC for ascending or DESC for descending.
type		The panel type. One of the following: *MEMBERSHIP (channel content), SEARCH, HIGHEST_RATED, MOST_DISCUSSED, RECENTLY_ADDED, FAVORITES_CONTENT, SHARED_CONTENT, SUBSCRIBED_CHANNEL, ARCHIVED_EVENTS, CUSTOM*
context	child object	
item	child object	An object representing one item on the panel.

playbackVideo

Property	Type	Description
autoPlay	string	Indicates whether the video should begin playing immediately, or wait for the user to start play
azure		*Optional.* Used to add a live event optimized for Azure cloud and return playback info in one call.
clientFileName	string	The filename of the content. This is used for building a URL for playing content from the browser using the Kollective client.
clientHost	string	The hostname of the Kollective client. This is used for building a URL for playing content from the browser using the Kollective client.
clientPortHttp	string	The HTTP port used by the Kollective client. Default is 31013. This is used for building a URL for playing content from the browser using the Kollective client.
clientPortRtmp	string	The RTMP port used by the Kollective client. Default is 31014. This is used for building a URL for playing content from the browser using the Kollective client.
mediaHost	string	The hostname of the Kollective server to access for playback without using a Kollective client.
mediaPort	string	The port of the Kollective server to access for playback without using a Kollective client.
mediaKey	string	An authentication ticket valid for a certain time period to access the Kollective server for playback without using a Kollective client.
description	string	The description of the video that was played.
encrypted	boolean	Indicates whether the file is encrypted on disk.
fileUrl	string	The encoded path and filename of the video.
formatAnchor	string	An anchor key for accessing the format file associated with the content. This is used primarily for non-video content. See Get a file using an anchor key.
formatMoid	string	The MOID of the format for the video that was played. See MOIDs.
formatMoidForClient	string	The MOID of the format associated with the content. See MOIDs.
guid	string	The GUID (Globally Unique Identifier) for this play instance.
hasSubtitle	boolean	When true, the video has subtitles
liveCdnEnabled	boolean	When true, the live content can be played without a Kollective client using the EdgeCast CDN.

Property	Type	Description
`liveCdnHlsPlaybackUrl`	string	The URL for initiating clientless playback of a live event when no Kollective client is detected.
`mediaCdnUrl`	string	The full URL for playing the video clientlessly from EdgeCast. The URL includes an encrypted token containing an expiration time equal to the current time plus twice the length of video (minimum 30 minutes).
`moid`	string	The MOID of the content item for the video that was played. See MOIDs.
`realmId`	string	The MOID of the client realm for the company network.
`realmTicket`	string	The client realm ticket for the current authenticated user. The realm ticket is an authentication entity that identifies the group membership for the user.
`subtitleAnchor`	string	An anchor key for accessing the subtitle file associated with the content. See Get a file using an anchor key.
`subtitleLangCode`	string	A standard two-letter ISO language code representing the language of the subtitles.
`title`	string	The title of the video that was played.
`urn`	string	The URN for the video. A URN is a statistically unique identifier for a video consisting of a namespace following by a MOID.

reportPlay

Property	Type	Description
`browser`	string	The browser used when playing the content
`browserVersion`	string	The version of the browser used when playing the content
`bufferTime`	int	Amount of time in milliseconds spent buffering before the video begins playing. See also *stallTime*.
`contentMoid`	string	The MOID of the content item for the video that was played. See MOIDs.
`endTime`	long	The date and time when the user stopped viewing the video, expressed in UTC (Coordinated Universal Time).
`eventList`	string	For Kollective internal use.
`formatMoid`	string	The MOID of the format for the video that was played. See MOIDs.
`inProgress`	boolean	Only for live events. Indicates whether the live event is currently being viewed.
`machineName`	string	The machine name of the device where the content is played.
`nodeId`	string	The machine name of the device where the content is played.
`os`	string	The node ID of the device where the content is played.
`osVersion`	string	The operating system on the device used when playing the content.
`percentage`	int	The percent of the video by play time that was actually viewed. For non-live video content.
`playContext`	string	Identifies the context in which the content was accessed. For embed objects, this comes from the context parameter in the embed code.
`playDuration`	int	Amount of time in milliseconds the video was viewed. This can be greater than the video length if it was viewed multiple times.
`playGuid`	string	The GUID (Globally Unique Identifier) for this play instance.
`playType`	string	Specifies the type of external live event. LIVE - playing as a live stream VOD - playing as recorded Video on Demand content
`prepareTime`	int	Amount of time in milliseconds from when the player is instantiated to the start of play. This includes the *bufferTime*.
`segmentList`	string	For Kollective internal use.
`stallCount`	int	Number of times the play was stalled during viewing.
`stallTime`	int	Amount of time spent buffering while the video is being played. See also *bufferTime*.

Property	Type	Description
`startTime`	long	The date and time when the user first began viewing the video, expressed in UTC (Coordinated Universal Time).

transcoderProfile

Property	Type	Description
`bitrate`	long	The bitrate at which the video is encoded.
`code`	string	A code representing the profile. Use this code is the transcoderProfileCode property of the content object.
`description`	string	Description of the profile.
`height`	int	The video height in pixels.
`label`	string	The name of the profile as it appears in VideoCenter.
`name`	string	This name is not used.
`selected`	boolean	Indicates whether this is the default profile on the Create Content page in MediaCenter.
`type`	string	One of the following: flash, nonios, ios
`width`	int	The video width in pixels.

upload

Property	Type	Description
`attemptCount`	long	*Read-only.* Number of attempts to retrieve the file from the specified URL. If an attempt fails, Kollective will retry the upload up to three times. While the upload is being retried, the status remains as UPLOADING.
`contentType`	string	*Optional.* The MIME type of the content being uploaded.
`errorCode`	string	*Read-only.* Possible values are: *TRANSFER_ERROR* — An error occurred while the file was being uploaded via pull. *AUTH_ERROR* — The authentication of a pull upload failed. *FILE_SIZE_MISMATCH* — The file sizes of the original file and the uploaded file do not match. *FILE_MD5_MISMATCH* — The checksum values for the original file and the uploaded file do not match. *SCAN_ERROR* — An error occurred during the virus scan. *SCAN_FAILURE* — The file failed the virus scan. *OK* — The file successfully uploaded.
`expirationTime`	dateTime	*Read-only.* The date and time when the uploaded file will expire, expressed in UTC (Coordinated Universal Time). An uploaded file is normally published by creating a content item and associating the file with it. After the expirationTime is reached, the upload is deleted whether or not it has been published.
`extension`	string	*Optional.* The filename extension.
`fileChecksumMD5`	string	*Optional.* The MD5 checksum value for the file, used to check whether the file was corrupted during the upload process.
`fileSize`	long	*Optional.* The initial file size. The uploaded file is checked against this size.
`moid`	string	*Read-only.* The MOID assigned to the uploaded file. The MOID is available once the status in INITIATED. See MOIDs.
`originalFilename`	string	*Optional.* The file name of the original file.
`partsCount`	int	Optional for multi-part uploads, then read-only. For larger files that are uploaded in parts, this is the total number of parts in the uploaded file. You can specify a value when you initiate a multi-part upload, or let the API increment it for you as you add the parts.
`sourceCredentials`	string	Optional. A base64-encoded string of the credentials (username:password) for accessing a file from the sourceURL, typically via FTP or SFTP. See Upload a file using pull.
`sourceURL`	string	Optional. The URL from which the file was uploaded, for files uploaded using pull. Special characters, including spaces, in the URL must be encoded. See Upload a file using pull.

Property	Type	Description
`status`	string	*Read-only.* *INITIATED* — An upload resource was called and the upload object was created; the uploading has not yet begun. *UPLOADING* — The file is in the process of uploading. The upload object is now read-only. *UPLOADED* — The file has completed uploading but has not been scanned. *SCANNING* — The file has completed uploading and is being scanned. *FAILED* — The upload failed. Check *errorCode* and *statusLog* for the details on the failure. SUCCEEDED — The file has completed uploading and scanning.
`statusLog`	string	*Read-only.* A log of messages for all operations, warnings, or errors during upload process.
`uploadType`	string	*Read-only*. The API assigns the type when the upload is used. ORIGINAL — A content file to be published. *ICON* — An image file to be used as an icon or thumbnail for a content. *BANNER* — An image file to be used as a banner for a channel. *SUBTITLE* — A subtitle file. See Subtitles for details.
`username`	string	*Read only.* The Kollective username of the person who uploaded the file.

Creating Live Events

This chapter introduces Kollective Live Webcasting and describes how to create live events.

About Kollective Live Webcasting

Kollective Live Webcasting lets you distribute a captured stream of a live event to your workforce using your Kollective network. Event broadcasts are delivered using Kollective's peer-to-peer technology and viewed through a Kollective client.

Capturing a live event requires operation of video equipment and a media encoder that can stream the live content. Kollective does not provide these services. Once you have set up the capture and encoding of an event, Kollective Live Webcasting lets you distribute the stream to users in your Kollective network.

First, you need to define the live event by creating a content item. See Defining a live event for complete details.

Users begin viewing an event the same way they access other published content, by invoking the URN found in the urn property of the content object.

The live stream received by Kollective clients is buffered for approximately 30 seconds before it is played. This improves the uninterrupted viewing experience and enables peering among the clients.

Push or pull distribution

Kollective supports both push and pull connection methods for live event streams. With push distribution, the encoder initiates the connection with a Kollective grid server. With pull distribution, a Kollective grid server initiates the connection with the encoder.
You specify the connection method in the **encoderConnectionMethod** property of the content object. You can choose from these push options:

- **PUSH**— Encoder pushes to Kollective via Windows Media over HTTP. Any encoder may be used.
- **WOWZARTMP** — Encoder pushes to Kollective via RTMP with RTMP client playback. Requires the Adobe Flash encoder. Not supported for Kollective 7.5 or older clients.

There is one pull option, **PULL**, where Kollective pulls from the encoder via Windows Media over HTTP.
In most cases, push distribution is preferable since it avoids having to make your encoder accessible to the Kollective grid server. You might choose pull distribution if you already have an encoder set up that is accessible over the Internet.

Multiple bit rates

Kollective supports multiple bit rates for a live event by allowing multiple streams to be delivered to the Kollective grid server, each with a different bit rate. A single multiple bit rate stream is not supported.
You configure a live event for multiple bit rates by creating a format object for each bit rate. Each format specifies the bit rate in kbps plus the height and width dimensions of the video.

The bit rate at which you encode a live stream can affect the viewing experience for the end user. In general, higher bit rates provide better quality, but they also result in larger stream sizes that require greater available bandwidth to deliver properly.
If all viewers of your event will be accessing it from connections with similar bandwidths (such as a T1 connection at the office), you can encode the stream at one bit rate.
But if some of your viewers will be accessing the event from high bandwidth connections (T1) and others from lower speeds (such as DSL or dial-up), you should consider encoding the stream at more than one bit rate.

Low-bandwidth connections If some of your viewers have low-bandwidth connections, be sure to encode a stream at a rate low enough to accommodate the slowest connection. If a client accesses a live event where the lowest bit rate available is higher than the client's bandwidth, the player will still try to play the event but will most likely fail.

Multiple formats for pull distribution

A format for pull distribution consists of a primary URL for accessing the event stream (url property), an optional secondary URL for when the primary URL is not available (redundantUrl property), and the bit rate. For a given format, both the primary and secondary URLs must be encoded with the same bit rate. To support another bit rate, you define a new format accessible from a different HTTP URL.
When you define multiple formats for pull distribution:

- Viewers of the event all access the stream from the same Kollective URL.
- Each Kollective client selects the event source with the bit rate that is most appropriate for its available bandwidth.
- The client first attempts to access the primary URL for the chosen source. If it is not available, the secondary URL is used.

Multiple encoder streams for push distribution

For push distribution, Kollective provides two destinations for the encoder stream: an FMS URL (**encoderFmsURL**) and an optional Backup URL (**encoderBackupURL**). The Backup URL provides redundancy for the client's connection to the Kollective server. If you direct an encoder stream to the Backup URL, it should be sending the same content as the stream being sent to the FMS URL. Do not use the Backup URL for sending different content.

Source Redundancy and Archiving Source redundancy from the Backup URL does not protect a live event archive. If a video source fails, the backup stream takes over and keeps the event running, but recording of the archive will stop.

- Point both encoders to the same FMS URL.
- Start the encoder with the content you want to begin streaming.
- To switch content, start the second encoder; it will supplant the first encoder's stream but will not disconnect it from the publishing point.
- To switch back to the first encoder stream, you must stop and restart the first encoder.

Supported encoders

Kollective Webcaster is certified with the following encoders:

- Adobe Flash Media Live Encoder 3.2
- Telestream Wirecast 4.1 or higher
- Teradek Cube 255
- Matrox Monarch HD

Other encoders may also work. Contact Kollective Support for details on whether your specific encoder is known to work.

Know your encoder Before attempting a live event broadcast, you should be familiar with the features of your encoder, including starting and stopping streams, providing a backup stream in case the first one fails, and configuring for multiple bit rates.

For any of the encoders, the captured content must be streamed using Windows Media HTTP Streaming Protocol.

Adobe Flash Media Live Encoder 3.2

When setting up your live stream using the Adobe Flash Media Live encoder, on the Encoding Options tabbed panel:

- For the video **Format**, select *H.264*.
- Set the key frame interval to **2** seconds.
- The key frame interval must be a divisor of the chunk size. Kollective's media server delivers the stream with a chunk size of 6 seconds. So, the key frame interval can be 1, 2, 3, or 6 seconds. To reach users who do not have a Kollective client installed (clientless live streaming) the value must be less than 6. For optimal performance, Kollective recommends a value of 2 seconds.

In the Output options section:

- For **FMS URL**, enter the FMS URL as found on the Sources tab of the Edit Content page in MediaCenter.
- For **Backup URL**, enter the Backup URL as found on the Sources tab of the Edit Content page in MediaCenter.
- For **Stream**, enter the Stream name as found on the Sources tab of the Edit Content page in MediaCenter.

After clicking Connect, you enter the username and password for accessing the event on the Kollective server at the FMS URL. Enter the values exactly as they appear on the Sources tab of the Edit Content page in MediaCenter. If you also entered a Backup URL, you are prompted for the username and password again; enter the same values.

Important For a given stream, if you entered zero (the default) for the bit rate in MediaCenter, the bit rate is automatically configured. However, if you chose to enter bit rates for each stream in MediaCenter, then the video bit rate plus the audio bit rate in the encoder must match the MediaCenter bit rate. For instance, if you entered 500 as the bit rate in MediaCenter, you might set the video bit rate in the encoder to 450 and the audio bit rate to 50. If you increase the video bit rate to, say, 460, you must reduce the audio bit rate to 40 so the total still equals 500.

Clientless support

To send live event streams to users who do not have a Kollective client installed, you must install the MainConcept AAC Encoder plug-in for the Adobe Flash Media Live Encoder. Without this plug-in, the audio portion of the stream may not work.

Telestream Wirecast 4.2

When setting up your live stream for push distribution using the Telestream Wirecast encoder, on the Broadcast Settings panel:

- For **Encoder Preset**, select a Flash preset with a bit rate matching the bit rate you entered on the Edit Live Event page in Network Publisher.
- For **Destination**, select RTMP Flash Server.
- For **Address**, enter the FMS URL exactly as it appears on the Edit Live Event page in Network Publisher.
- For **Stream**, enter the Stream Name from the Edit Live Event page, remove the %b from the end of the stream name, and add the bit rate.

  For instance, if the stream name is

  ```
  cbe64059-4638-cdc8-d76b-7e0a080d91ff_%b
  ```

  and the bit rate is 500, enter this for Stream:

  ```
  cbe64059-4638-cdc8-d76b-7e0a080d91ff_500
  ```

The **%b** in the stream name is a value representing the bit rate of the stream. The Adobe Flash Media Live Encoder interprets this value and replaces it with the appropriate bit rate.

The Telestream Wirecast encoder does not interpret the **%b**, so you must replace it with the actual bit rate.

Click **Set Credentials** and enter the username and password from the Edit Live Event page.

For further details on Webcaster encoders, see the *Kollective Webcaster User Guide*.

Defining a live event

You create and manage live events using API resources in the same way you work with content or subscriptions.
To define a live event, you create a content object with the following settings:

- Set the **contentType** property to LIVE_EVENT.
- Set the start and end time for the event with the **eventStart** and **eventEnd** properties.
- Set the encoder connection method (push or pull) with the **encoderConnectionMethod** property.
- If you chose one of the push options for the connection method, you specify the credentials that the encoder will use when accessing the Kollective grid server in the **encoderPushAuthUserName** and **encoderPushAuthPassword** properties.
- Define a child format object for each of the streams to be broadcast. The format includes properties that define the bit rate, and the height and width dimensions of the video.
  In addition, if you chose pull for the connection method, you specify the URL(s) for accessing the stream in the **url** and **redundantUrl** properties of the format object.

As with other content items, you can also set properties to define other metadata, such as a description, custom icon, and the groups that are allowed to access the event.
Here is a sample content object in XML that defines a live event for push distribution with two formats.

```
<ns:content xmlns:ns="http://api.kontiki.com">
      <contentType>LIVE_EVENT</contentType>
      <title>Company Meeting</title>
      <description>Our CEO addresses the troops.</description>
    <eventStart>2012-01-17T09:00:00-08:00</eventStart>
      <encoderConnectionMethod>PUSH</encoderConnectionMethod>
    <encoderPushAuthUserName>joeuser</encoderPushAuthUserName>
      <encoderPushAuthPassword>K12z00@jfX</encoderPushAuthPassword>
<format>
      <bitRate>300</bitRate>
      <height>360</height>
      <width>480</width>
</format>
      <format>
      <bitRate>700</bitRate>
      <height>360</height>
      <width>480</width>
</format>
</ns:content>
```

You can send this content object in the following metadata/content request:

```
POST https://hostName/api/v2/metadata/content HTTP/1.1
```

For more details on the metadata/content request, see Create a new content item.

Video Upload Guidelines

To provide a quality experience for viewers of your video, it is important to start with a high quality input file. By following the guidelines listed below, you can ensure that the resulting transcoded video has the best quality possible.

Supported file extensions	wmv, f4v, avi, mpg, mpeg, mp4, mov, 3gp, m4v, mv4, asf, wm, flv, mp1, mp2, mp3, wma, wav, aiff, aac
Bit rate	550kbps minimum, 1.5 to 2 Mbps recommended
Aspect ratio	16:9 widescreen (to avoid letterboxing)
Resolution	640 x 360 minimum, 1280 x 720 recommended
Frames per second	24 minimum

Supported codecs

The video file types that Kollective supports can use a variety of different codecs. Most popular codecs are supported; however some codecs may be incompatible with the transcoding technology used by Kollective.
If you encounter a transcoding error when uploading a video, you may contact Kollective Support for a list of the latest compatible codecs.

Transcoding specifications

After transcoding is completed, videos have the following specifications:

Transcode Profile	H.264 (MPEG-4) video with MP4 file extension
Frames per second	30
Aspect ratio	16:9 widescreen

The resolution and bitrate vary depending on the chosen transcode profile.

Transcode Profile	Resolution	Bitrate
Low	640 x 360	350kbps
Standard	640 x 360	550kbps
High	640 x 360	800kbps
HD Standard	1280 x 720	800kbps
HD High	1280 x 720	1200kbps

Resolution and aspect ratio considerations

As part of the transcoding process, videos are compressed and scaled. The resulting resolution (either 640 by 360, or 1280 by 720) depends on the transcode profile you selected when uploading the video.

All profiles result in a widescreen 16:9 aspect ratio. Uploaded videos that have an aspect ratio other than 16:9 are "letterboxed" with black bars inserted on the sides, or top and bottom, of the video image to fill each frame.

Scaling a low-resolution video can magnify its imperfections resulting in excess pixilation or blurry images. Videos with resolutions smaller than the transcode profile resolution are stretched, usually with a visible loss of quality.

Videos with resolutions higher than the transcode profile resolution are scaled down, minimizing any imperfections in the original video. Consequently, it is best to upload a higher resolution video when possible.

HTTP Status Codes

The following table lists the most common HTTP response status codes returned by the REST API.

⚠ The status codes indicate the status of the request at the HTTP level. They may not represent the success or failure of the request as intended.
For instance, if you supply incorrect criteria in a search request, the result will not contain the data you were searching for but the HTTP response code may still be 200 (success) if the request is formatted properly.

Code	Response Text	HTTP Operation	Description
`200`	`OK`	GET, PUT, DELETE	No error, operation successful.
`201`	`Created`	POST	Successful creation of a resource.
`204`	`No Content`	GET, PUT, DELETE	The request was processed successfully, but no response body is needed.
`400`	`Bad Request`	GET, POST, PUT, DELETE	Malformed syntax or a bad query.
`401`	`Unauthorized`	GET, POST, PUT, DELETE	Action requires user authentication.
`403`	`Forbidden`	GET, POST, PUT, DELETE	Authentication failure or invalid Application ID.
`404`	`Not Found`	GET, POST, PUT, DELETE	Resource not found.
`405`	`Not Allowed`	GET, POST, PUT, DELETE	Method not allowed on resource.
`406`	`Not Acceptable`	GET	Requested representation not available for the resource.
`408`	`Request Timeout`	GET, POST	Request has timed out.
`409`	`Resource Conflict`	PUT, DELETE	State of the resource doesn't permit request.

Code	Response Text	HTTP Operation	Description
411	Length Required	POST, PUT	The server needs to know the size of the entity body and it should be specified in the *Content-Length* header.
412	Precondition failed	GET	Operation not completed because preconditions were not met.
413	Request Entity Too Large	POST, PUT	The representation was too large for the server to handle.
414	Request URI too long	POST, PUT	The URI has more than 2000 characters.
415	Unsupported Type	POST, PUT	Representation not supported for the resource.
416	Requested Range Not Satisfiable	GET	Requested range not satisfiable.
500	Server Error	GET, POST, PUT	Internal server error.
501	Not Implemented	POST, PUT, DELETE	Requested HTTP operation not supported.
502	Bad Gateway	GET, POST, PUT, DELETE	Backend service failure (data store failure)

ECDN 10.4 Resource API Requests