Clipgenerator® SDK

Trivid GmbH

January 29, 2013

Contents

1 Overview

The Clipgenerator Service allows users to create appealing video clips by combining text, animations and music. The Clipgenerator API opens the Clipgenerator service to developer who can now leverage integration of video clips with their own services.

This document is organized as follows. This chapter discusses some basic clipgenerator practices that apply in general to the Clipgenerator Service and serves as an overview about how the service works. The second chapter jumps right into details of creating clips using the Clipgenerator SDK libraries. Details about the xml format used for the clip's configuration is discussed chapter three and the REST API in chapter four allows fans of other programming languages to make use of the Clipgenerator API.

1.1 Accessing the Clipgenerator API

In order access the Clipgenerator API either using the supplied SDK libraries or directly using HTTP/REST interfaces an **API-ID** and an **API-SECRET** is required.

In order to obtain these credentials please contact our support at service@clipgeneratr.com.

1.2 Core API

The HTTP/REST API is the core of the Clipgenerator service. Most higher level services at Clipgenerator are built on top of the HTTP/REST interface. Compared to the other higher level Clipgenerator APIs such as *SimpleFTP* or *SimpleImmo* the direct HTTP API offers the widest control possible. This includes but is not limited to user management, provisioning and the management of the video clips.

The supplied SDK libraries are wrappers around the core HTTP/REST API and are intended to support developers while creating video clips and intreating the Clipgenerator services.

1.3 Basic functionality

The Clipgenerator API provides the following core functionality

1.3.1 User management

The Clipgenerator API is able to manage an unlimited amount of users.

1.3.2 Package management

Packages are configured on our backend per application and define features users are allowed to use. For instance, it is possible to distinguish between *basic* users who are only allowed to manage 3 clips at the same time and use have acces to 20 songs. While *premium* users are allowed to choose from a list of 100 songs and manage up to 20 video clips. The packaging is up to the applications needs, and can be configured individually.

1.3.3 Music management

The available music to a user is managed in packages by the package management system. Additional to this, the API allows listing and previewing the configured songs. Covers and metadata for sorting is provided in multiple languages.

1.3.4 Video Designs

The Clipgenerator video designs describe how images will be animated in the final clip. As music, video designs are managed in packages and can be listed on a per user manner. Metadata and previews are supplied by the API.

1.3.5 Upload Pictures

A clip is composed by uploading pictures to the API. They are stored on a per user manner and can only be used in clips by the same user.

1.3.6 Upload Logo

In addition to pictures a clip can be configured to contain one logo shown in one of the corners of the clip.

1.3.7 Uploading custom Music

Custom music can be used inside the clip by uploading custom mp3 files via the API. The music is stored on a per user manner and can only be used in clips by the same user.

1.3.8 Clip management

Clips are managed in slots by the Clipgenerator service. Depending on the configured package a user is allowed to save a predefined number of clips. Clips can be deleted, updated and viewed and embedded. The representation of a clip is saved in an XML and created by the SDK libraries. See the Clip XML chapter for a detailed description of the clip configuration.

1.3.9 Create downloads

At clipgenerator clips are usually saved and served in a dynamic manner. This allows the clips to be updated easily at a later stage. Customers who want to create a static clip (e.g. an MP4 file) for further processing can create a downloadable version of the clip.

2 Clipgenerator PHP SDK

2.1 Creating simple clips

In order to create a simple clip using the Clipgenerator PHP SDK the following steps must be executed:

1. Create and configure Clipgenerator Client with your access credentials
2. Create the clip xml that describes the clip's configuration.
3. Save the clip using the Clipgenerator Client

The `ClipgeneratorClient` PHP Class wraps the Clipgenerator API using `curl` as http client. It directly communicates with the HTTP API and does not require any other dependencies than curl.

In addition to the `ClipgeneratorClient` the Clipgenerator PHP SDK contains a collection of other Classes that are intended to support the developer while creating the clip configuration xml. The following code snippet shows the first steps how to create a first rudimentary clip. More examples can be found in the examples directory.

```
// create clipgenerator client, configure language and credentials
$cg = new ClipgeneratorClient(API_ID, API_SECRET, API_USER_ID, 'en');

// create new video clip using php sdk library
$title = "My First Video";
$video = new Video($title);

// create basic configuration
```

```
$video->designId = ”ce_evolution”;
$video->songId = ”Wasser_Clipgenerator_51a_3_258.mp3”;

// add first frame to video clip

// upload first picture to backend
$pic1 = $cg->uploadPicture(’images/00-Fashion.jpg’);

// make sure upload was successful
assert(’$pic1 !== false’);

// create picture element with corresponding picture id at the backend
$picture1 = new Picture($pic1[’id’]);

// create frame element, that holds the picture
$frame1 = new Frame($picture1);

// add the frame to video clip
$video->addFrame($frame1);

// add second frame to video clip
$pic2 = $cg->uploadPicture(’images/01-Fashion.jpg’);
assert(’$pic2 !== false’);
$picture2 = new Picture($pic2[’id’]);
$frame2 = new Frame($picture2);
$video->addFrame($frame2);

// add third frame to video clip
$pic3 = $cg->uploadPicture(’images/02-Fashion.jpg’);
assert(’$pic3 !== false’);
$picture3 = new Picture($pic3[’id’]);
$frame3 = new Frame($picture3);
$video->addFrame($frame3);

// get the created video xml configuration
$videoXml = $video->asXml();

// save video clip as new clip (by not providing an video id)
```

```
$videoId = $cg->saveVideo($videoXml);
```

The following sections will discuss of each PHP SDK Class in detail.

2.1.1 ClipgeneratorClient

The `ClipgeneratorClient` client is responsible for the communication with the Clipgenerator API server. It provides wrappers for some curl calls that direct calls directly to the Clipgenerator HTTP/REST API.

Creating a new ClipgeneratorClient instance

```
require('ClipgeneratorClient.php');

$cg = new ClipgeneratorClient(
   API_ID,
   API_SECRET,
   API_USER_ID,
   LANGUAGE,
);
```

Parameters:

API_ID : Your API-ID credentials (mandatory)

API_SECRET : The API-SECRET you received with the API-ID (mandatory)

API_USER_ID : User-id that should be used while retrieving user based information. This parameter is optionally when managing users. For all other calls this parameter is mandatory.

Retrieving users

```
require('ClipgeneratorClient.php');
$cg = new ClipgeneratorClient(API_ID, API_SECRET, API_USER_ID, 'en');
$user = $cg->getUser();
print_r($user);
```

Creating users

Deleting users

Assigning Packages

Retrieving videos

Saving Videos

Retrieving music

Uploading music

Retrieving video designs

Uploading Pictures

Uploading Logos

3 Clipgenerator Clip XML

The configuration of a dynamic clip at Clipgenerator is stored as an XML file. Using the API the configuration can be retrieved and saved. Provided SDK libraries usually serve as generators and readers. Before a clip is saved they are usually serialized into the this Clipgenerator Clip XML format.

Although the SDK libraries provide an easy way to create and modify clips, creating the XML instead of using the libraries may be more convenient. Especially when combined with common template engines.

3.1 Top level structure

The Clip XML is build on the following five main node: `video`, `timeline`, `logo` and `fonts`.

```
<?xml version="1.0" encoding="UTF-8"?>
<webcart version="2.0">
    <video> … </video>
    <timeline> … </timeline>
    <logo> … </logo>
    <fonts> … </fonts>
</webcart>
```

The following chapters will discuss the single node elements, their child nodes and attributes in detail.

3.2 Video Element

The video element contains the video id, video title and other video metadata.

```
<video>
    <id>SOME-VIDEO-ID</id>
    <builder>clipgenerator.api</builder>
    <title>First video</title>
    <producer>Acme Inc.</producer>
    <description>This is a first video.</description>
    <keywords>key1,key2</keywords>
    <copyright>2013 Clipgenerator</copyright>
    <content>Contains pictures of Acme Inc.</content>
    <targetAudience></targetAudience>
</video>
```

Parameters

`video/id` Unique video id generated by the Clipgenerator API. If omitted while saved, the a new video id will be assigned automatically. When supplied during saving the target video will be overwritten.

`video/title` String. Short title of the created clip.

`video/producer` String. Denotes the producer of this clip.

`video/description` XML escaped String. May contain HTML encoded characters. Contains a short description of the clip and is often used along with the title to configure the metadata for SEO purposes in downloadable clips and embedded clips.

`video/keywords` Comma-separated strings. Used in SEO relevant tasks and should fit the content of the clip.

`video/copyright` String. Holds additional copyright information of the clip, it's audio and images.

`video/content` String. Holds a short description of the contents of the clip. Can be used for SEO purposes later.

`video/builder` String. Identifies the generator the Clip XML was created with. Currently the API rewrites this string to `clipgenerator.api`.

3.3 Timeline Element

The timeline element is holds the storyboard of the clip. It configures some parameters related to the storyboard in the `config` element. Allows global configuration of lower thirds which will be shown during the whole clip and describes each frame with its configuration separately.

```
<timeline>
    <config>
    <lowerThirds/>
    <frames>
        <frame … />
    </frames>
</timeline>
```

3.3.1 Timeline Config Element

The config node of the timeline element holds the clip's configuration regarding the song, video design (aka variation) format and behavior when the clip starts or ends.

```
<config>
   <song id="Edison_Clipgenerator_63b_16_425.mp3">http:…Vd.mp3</song>
   <variation id="cg_RotatingBack_Clipgenerator">http:…tor.xml</variation>
    <ratio>16x9</ratio>
    <format name="360p"/>
    <startPage>
        <enabled>true</enabled>
        <duration>5</duration>
        <showLowerThirds>true</showLowerThirds>
    </startPage>
    <endPage>
        <enabled>true</enabled>
        <duration>5</duration>
        <onEndAction name="showOutro" parameter=""/>
        <showLowerThirds>true</showLowerThirds>
    </endPage>
    <clipMaxLenght>0</clipMaxLenght>
</config>
```

Parameters

`config/song` String. Will be automatically filled by the Clipgenerator API with the url of the song configured by the song id parameter. Can be ignored / left empty during the creation.

`config/song/@id` String. Denotes the song that will be used while clip is playing. Optionally. If omitted while saving a silent clip will be created.

`config/variation` String. Will be automatically filled by the Clipgenerator API with the url holding the animation that will be used during the playback of the clip. Can be ignored / left empty during the creation.

`config/variation/@id` String. Denotes the animation (aka variation or video design) that will be used during the playback of the clip. Images will be fed to the variation and animated. Mandatory.

`config/ratio` String. Defines the default ratio the clip was created for. Must be either `16x9` or `4x3`. Mandatory.

`config/format/@name` String. Defines the default format the clip was created for. Must be either one of `240p`, `360`, `480`, `720p`. Mandatory.

`config/startPage/enabled` String. Must be `true` or `false`. If set to `true` the first frame will not be animated but shown as a static picture. This option is often used to create an intro with an static image such as a company logo. Optional.

`config/startPage/duration` Integer. Denotes the time in seconds the first picture is shown if `config/startPage/enabled` is enabled. Otherwise this value is ignored. Optional.

`config/startPage/showLowerThirds` Boolean. Must be either `true` or `false`. Denotes whether lower third elements are shown during the start page or not.

`config/endPage/enabled` String. Must be either `true` or `false`. If set to `true` the last frame will not be animated but shown as a static picture. This option is often used to create an outtro with an static image such as a company logo. Optional.

`config/endPage/duration` Integer. Denotes the time in seconds the last picture is shown. Optional.

`config/endPage/onEndAction/@name` String. If configured to `showOutro` the clip will not jump to the first frame after finishing playing.

`config/endPage/showLowerThirds` Boolean. Must be either `true` or `false`. Denotes whether lower third elements are shown during the end page or not.

4 Clipgenerator HTTP/REST API

4.1 Retrieving users

The following call retrieves detailed information about a user with the user name API_USER.

```
curl http://api-v2.clipgenerator.com/getUser \
-d apiId=$API_ID \
-d apiSecret=$API_SECRET \
-d userId=$API_USER
-d lang=en
```

Parameters:

- `apiId` & `apiSecret`: Your HTTP API credentials. Mandatory.
- `userId` : Username that identifies the user on clipgenerator. Mandatory.
- `lang` : One of [en, de, es, pl, fr]. Optional. Default is en.

Result

A successful request returns the following JSON structure:

```
{
”result” : {
    ”id” : ”sergej”,
    ”createdAt” : ”2012-10-10 16:16:31”,
    ”lastName” : ”Mueller”,
    ”firstName” : ”Sergej”,
    ”vatId” : ””,
    ”usedVideoSlots” : 15,
    ”freeVideoSlots” : 485,
    ”street” : ”Adlerstr. 54”,
    ”company” : ”Trivid GmbH”,
    ”city” : ””,
    ”language” : ”en”,
    ”availableVideoSlots” : 500,
```

```
    "zipCode" : "76137",
    "country" : "",
    "email" : "simpleftp@sergejmueller.com",
    "isActive" : true
},
"code" : 200,
"message" : "OK"
}
```

4.2 Creating users

Users can be created by using the following request. The NEW_API_USER_ID variable must contain a unique user id which should identify the corresponding user uniquely. Most of the data attributes are optional. Mandatory attributes include `email` and `userId`.

```
curl http://api-v2.clipgenerator.com/createUser \
-d apiId=$API_ID \
-d apiSecret=$API_SECRET \
-d userId=$NEW_API_USER_ID \
-d lang=en \
-d company="Trivid GmbH" \
-d firstName="Sergej" \
-d lastName="Mueller" \
-d street="Adlerstr. 54" \
-d zipCode="76137" \
-d password="$NEW_API_USER_PW" \
-d isActive="true" \
-d email="random-user3@sergejmueller.com"
```

Parameters:

- `apiId` & `apiSecret`: Your HTTP API credentials. Mandatory.

- `userId` : User id that will uniquely identify the user on clipgenerator. Mandatory.
- `lang` : One of [en, de, es, pl, fr]. Optional. Default is en.
- `email`: Email address of the new user. Mandatory.
- `password`: Password for the new user. Mandatory. Currently ignored during api calls since only apiId based authentication is used.
- `company`: Company name of the new user. Optional.
- `firstName`: Given name of the new user. Optional.
- `lastName`: Surname of the new user. Optional.
- `street`: Street / address block of the new user. Optional.
- `zipCode`: Zip code of the new user. Optional.
- `isActive`: Defines whether user is active or inactive at the after creation. Optional. Default value is true.

Result

A successful request returns the following JSON structure:

```
{
    ”result” : null,
    ”code” : 200,
    ”message” : ”User $NEW_API_USER_ID successfully created.”
}
```

4.3 Deleting users

Users can be removed using the following request

```
curl http://api-v2.clipgenerator.com/deleteUser \
-d apiId=$API_ID \
-d apiSecret=$API_SECRET \
-d userId=$API_USER_ID \
-d lang=en
```

Parameters:

- `apiId` & `apiSecret`: Your HTTP API credentials. Mandatory.
- `userId` : User id of the user to be deleted. Mandatory.
- `lang` : One of [en, de, es, pl, fr]. Optional. Default is en.

Result

A successful request returns the following JSON structure:

```
{
    ”result” : null,
    ”code” : 200,
    ”message” : ”User with id $API_USER_ID successfully deleted.”
}
```

4.4 Loadig video xml configuration

Clips at clipgenerator are defined by an clip XML file. The following request retrieves a previously saved clip and it's XML.

```
curl http://api-v2.clipgenerator.com/loadVideo \
-d apiId=$API_ID \
-d apiSecret=$API_SECRET \
-d userId=$API_USER \
-d lang=en \
-d videoId=”4o0HTN7K9wj3vFtHtAUp8K36BN9Z28fh”
```

Parameters:

- `apiId` & `apiSecret`: Your HTTP API credentials. Mandatory.
- `userId` : Username that identifies the owner of the clip. Mandatory.
- `lang` : One of [en, de, es, pl, fr]. Optional. Default is en.
- `videoId` : Clipgenerator video id. Determines which clip's xml to retrieve. Mandatory.

Result

A successful request returns the following JSON structure:

```
{
    ”result” : {
        ”id” : ”4o0HTN7K9wj3vFtHtAUp8K36BN9Z28fh”,
        ”video” : ”<?xml ... [omitted for illustration reasons]” },
    ”code” : 200,
    ”message” : ”OK”
}
```

4.5 Saving videos

A clip is always saved by saving it's clip xml configuration. For this purpose the whole clip configuration must be supplied. Depending on intent a clip can be saved as a new clip or updated by overwriting an existing clip.

Update existing clip: in order to update an existing clip, the clip must be retrieved using the `loadVideo` api call first. Modified as desired. And then saved with the `saveVideo` call. Using the video id parameter in the clip xml configuration the backend will automatically detect that this clip should be updated and thus will overwrite the existing clip configuration xml with the newer one. The following request saves a sample clip xml and overwrites the existing clip configuration:

```
#!/bin/bash
clipxml=”‘cat sample-clip.xml‘”
curl http://api-v2.clipgenerator.com/saveVideo \
-d apiId=$API_ID \
-d apiSecret=$API_SECRET \
-d userId=$API_USER \
-d lang=en \
-d video=”$clipxml”
```

Parameters:

- `apiId` & `apiSecret`: Your HTTP API credentials. Mandatory.

- `userId` : User id that identifies the clip's owner. Mandatory.
- `lang` : One of [en, de, es, pl, fr]. Optional. Default is en.
- `video` : Full contents of the clip xml supplied as a string parameter.

Result

A successful request returns the following JSON structure:

```
{
    ”result” : {
        ”id” : ”2Rj3m6evkYeDN66NbwO3TCYgAC4c8GKA”,
        ”video” : ”<?xml ... [omitted for illustration reasons]” },
    ”code” : 200,
    ”message” : ”OK”
}
```

Creating new video clips: in order to create a new clip or save an existing one as a new one the `video id` field inside the clip's xml configuration must be set to an empty string. The api backend will detect the missing video id and thus save the clip to a new video slot. The following request fills a new video slot with the corresponding video xml and returns the new video id.

```
#!/bin/bash
clipxml=”‘cat sample-clip-as-new.xml‘”
curl http://api-v2.clipgenerator.com/saveVideo \
-d apiId=$API_ID \
-d apiSecret=$API_SECRET \
-d userId=$API_USER \
-d lang=en \
-d videoId=”” \
-d video=”$clipxml”
```

Parameters:

- `apiId` & `apiSecret`: Your HTTP API credentials. Mandatory.
- `userId` : User id that identifies owner of the clip. Mandatory.
- `lang` : One of [en, de, es, pl, fr]. Optional. Default is en.

- `video` : Full contents of the clip xml supplied as a string parameter.

Result

A successful request returns the following JSON structure including a new clip id for the saved clip:

```
{
    ”result” : {
        ”id” : ”2Rj3m6evkYeDN66NbwO3TCYgAC4c8GKA”,
        ”video” : ”<?xml ... [omitted for illustration reasons]” },
    ”code” : 200,
    ”message” : ”OK”
}
```

4.6 Retrieving music

Each user is associated with at least one music package. The music packages determine which songs a user is allowed to use while creating clips. The song information is checked on the the `saveVideo` api call and whenever the video is fetched by the clipgenerator player using the `getVideoXml` api call. The following request retrieves the complete music list of a user, including meta data, preview urls, cover urls and song categories. This structure is usually used directly by an HTML5 client to present the information to it's users in adequate way.

```
curl http://api-v2.clipgenerator.com/getMusic \
-d apiId=$API_ID \
-d apiSecret=$API_SECRET \
-d userId=$API_USER \
-d lang=en
```

Parameters:

- `apiId` & `apiSecret`: Your HTTP API credentials. Mandatory.
- `userId` : User id. Will determine who's song to fetch. Mandatory.
- `lang` : One of [en, de, es, pl, fr]. Optional. Default is en.

Result

A successful request returns the following JSON structure. All required data for the representation the song list is included in this structure. For a detailed description please see Appendix A.


```
{
”code”: 200,
”message”: ”OK”,
”result”: {
    ”packages”: [{ … ommited … }],
    ”genres”: [{ … ommited ... }],
    ”moods”: [{ … ommited … }],
    ”instruments”: [{ … ommited … }],
    ”speeds”: [{ … ommited … }],
    ”themes”: [{ … ommited …}],
    ”categoryThemes”: [{ … ommited … }],
    ”songs”: [{
        ”label”: ”BV”,
        ”artist”: ””,
        ”length”: ”268”,
        ”title”: ”Fashion Groove”,
        ”description”: ”cool catwalk tune based on electro beats”,
     ”packages”: [”1”, ”2”, ”3”, ”4”, ”5”, ”10”, ”11”, ”20”, ”30”, ”40”],
        ”genres”: [”3”, ”15”, ”92”, ”100”, ”103”],
        ”moods”: [”4”, ”6”, ”9”],
        ”instruments”: [],
        ”speeds”: [”1”],
        ”themes”: [”1”, ”8”, ”9”, ”10”, ”132”, ”141”, ”142”],
        ”categoryGenres”: [”5”, ”7”, ”8”],
        ”categoryThemes”: [”2”, ”6”, ”9”],
        ”previewUrl”: ”http..._preview.mp3”,
        ”url”: ”http:...5mA1G9.mp3”,
        ”cover”: ”http:...G9_cover.jpg”,
        ”id”: ”Fashion_Groove_Clipgenerator_54b_29.mp3”,
        ”designId”: ”cg_Rotating1_Clipgenerator”,
        ”hasCover”: true,
        ”isInstrumental”: true
```

```
    }, { … ommited … }]
}
}
```

4.7 Retrieving video designs

Users are configured to use only a subset of video designs. The list can be retrieved using the following request:

```
curl http://api-v2.clipgenerator.com/getDesigns \
-d apiId=$API_ID \
-d apiSecret=$API_SECRET \
-d userId=$API_USER \
-d lang=en
```

Parameters:

- `apiId` & `apiSecret`: Your HTTP API credentials. Mandatory.
- `userId` : User id. Will determine who's song to fetch. Mandatory.
- `lang` : One of [en, de, es, pl, fr]. Optional. Default is en.

Result

A successful request returns the following JSON structure including the title, speed category, complexity and preview url.

```
{
    ”code”: 200,
    ”message”: ”OK”,
    ”result”: [{
        ”id”: ”ce_taketwo”,
        ”title”: ”Take Two”,
        ”speed”: ”slow”,
        ”complexity”: ”medium”,
        ”previewUrl”: ”http:…ce_taketwo.mp4”
        },
```

```
        {
        "id": "ce_blackwhite",
        "title": "BlackWhite",
        "speed": "fast",
        "complexity": "high",
        "previewUrl": "http:…ce_blackwhite.mp4"
        }, … ommited …
    ]
}
```

4.8 Uploading pictures

In order to create valid clip xml configurations, the xml must contain a list of frames configured with pictures. Before pictures can be used in a clip configuration xml, they must be uploaded separately to the Clipgenerator backend. The following request uploads a picture of the corresponding user and returns the required picture id that can be used clip's xml configuration.

```
curl http://api-v2.clipgenerator.com/uploadPicture \
-F apiId=$API_ID \
-F apiSecret=$API_SECRET \
-F userId=$API_USER \
-F file=@./sample-picture.jpg
```

Parameters:

- `apiId` & `apiSecret`: Your HTTP API credentials. Mandatory.
- `userId` : User id. Will determine the owner of the uploaded image. Mandatory.
- `file` : Image file that should be uploaded. Name of the parameter is ignored. Mandatory.

Result

A successful request returns the following JSON structure which includes the important picture id. This id can now be used inside of the users clip xmls. Additionally

to the picture id the call also returns a list of urls to the different sizes used by the clipgenerator's player. The `thumbnailUrl` for instance can be very handy when it comes showing thumbnails inside an UI interface after user has uploaded a file.

```
{
    ”code”: 200,
    ”message”: ”OK”,
    ”result”: {
        ”id”: 205392,
        ”url”: ”http:...86c.jpg”,
        ”thumbnailUrl”: ”http:...86c_thumb.jpg”,
        ”url_240p”: ”http:…86c_240p.jpg”,
        ”url_360p”: ”http:…86c_360p.jpg”,
        ”url_480p”: ”http:…86c_480p.jpg”,
        ”url_720p”: ”http:…86c_720p.jpg”
    }
}
```

4.9 Creating downloadable clips

Clipgenerator's core expertise is the creation of dynamic clips that can be easily modified any time. Nevertheless customers who want to create downloadable video files (e.g MP4 files) can do so by executing the following request. The request schedules the creation of a downloadable version of the supplied clip via the `videoId` parameter. The download can be retrieved later using the `url` attributed in the `getVideos` api call.

```
curl http://api-v2.clipgenerator.com/scheduleDownloadJob \
-d apiId=$API_ID \
-d apiSecret=$API_SECRET \
-d userId=$API_USER \
-d lang=en \
-d videoId=”2Rj3m6evkYeDN66NbwO3TCYgAC4c8GKA” \
-d resolution=”480x360” \
-d format=”mp4”
```

Parameters:

- `apiId` & `apiSecret`: Your HTTP API credentials. Mandatory.
- `userId` : User id. Owner of the clip.
- `resolution` : Determines the resolution of the downloadable clip. Must be one of '480x360', '640x360', '640x480', '853x480', '960x720', '1280x720', '320x240', '320x180', '400x300', '400x226', '480x270', '426x240', '768x576'. Mandatory.
- `format` : Determines the video format used to create the download. Must be one of 'wmv', 'mov', 'avi', 'flv', 'mp4', 'ogv'. Mandatory.

Result

A successful request returns the following JSON structure. Depending on the resolution and format, creating a downloadable video may take up to 20 minutes. Therefore this call returns immediately in an asynchronous manner. As soon as the downloadable version is created, it will be available via the `url` attribute in the video list returned by the `getVideos` api call.

```
{ "code":200, "message": "OK", "result":null }
```

4.10 Activating user packages

Every user is associated a set of packages. Packages determine which music the user is allowed to use and which features. The `getMusic` call for example returns the music available to the user depending on his music package configuration. The following request activates a music package for the configured user:

```
curl http://api-v2.clipgenerator.com/activatePackage \
-d apiId=$API_ID \
-d apiSecret=$API_SECRET \
-d userId=$API_USER \
-d lang=en \
-d packageId="$MUSIC-PACKAGE-ID" \
-d packageType="music"
```

Parameters:

- `apiId` & `apiSecret`: Your HTTP API credentials. Mandatory.

- `userId` : User id of user the package should be assigned to.
- `packageId` : the id of the package to activate. Package ids are customer specific and usually configured when a new apiId is created in the clipgenerator backend. For more information please contact the Clipgenerator service.
- `packageType` : currently supported package types: “music” and “editor”.

Result

A successful request returns the following JSON structure.

```
{
    ”code”:200,
    ”message”: ”Music package X successfully activated.”,
    ”result”:null
}
```

4.11 Retrieving video list

In order to retrieve a list of all videos a user owns the following request must be executed:

```
curl http://api-v2.clipgenerator.com/getVideos \
-d apiId=$API_ID \
-d apiSecret=$API_SECRET \
-d userId=$API_USER \
-d lang=en
```

Parameters:

- `apiId` & `apiSecret`: Your HTTP API credentials. Mandatory.
- `userId` : User id. Owner of the clip.

Result

A successful request returns the following JSON structure.

```
{
    ”code”: 200,
    ”message”: ”OK”,
    ”result”: [{
        ”id”: ”QeiypbYvmI6L2Ei251ZbBeIMm9I5wDN7”,
        ”songId”: ”Wasser_Clipgenerator_51a_3_258.mp3”,
        ”designId”: ”ce_evolution”,
        ”title”: ”My First Video”,
        ”dateCreated”: ”2013-01-28 09:50:20”,
        ”ratio”: ”4x3”,
        ”downloadUrl”: ””,
        ”thumbnailUrl”: ”http:…98_thumb.jpg”,
        ”frameCount”: ”3”
    }, … {…} ]
}
```