fv Solution API

2017.07.31

Table of Content

fvLIBRARY API

This document is written about fvLIBRARY API (called “API”).
API is mainly consisted by 3 types of below.

1. Media registration API
2. fv creation request API
3. Media management API

Each types are delivered as REST API of WEB.

Definition

Video
the data before upload/registration.
Media
the data including meta-data after upload/registration.
Tile
the divided media into some part after upload/registration.
fabric video (fv)
the combined data (H.264/AVC) by some tiles.

Basic Limitation of API

Creation of Tile

Tile Creation API creates tiles specified by integer seconds(1-900 seconds) when registering a video.
However, tile of level 1 is always created, no matter what your level selection.
Specification of basic tiles are as follows.

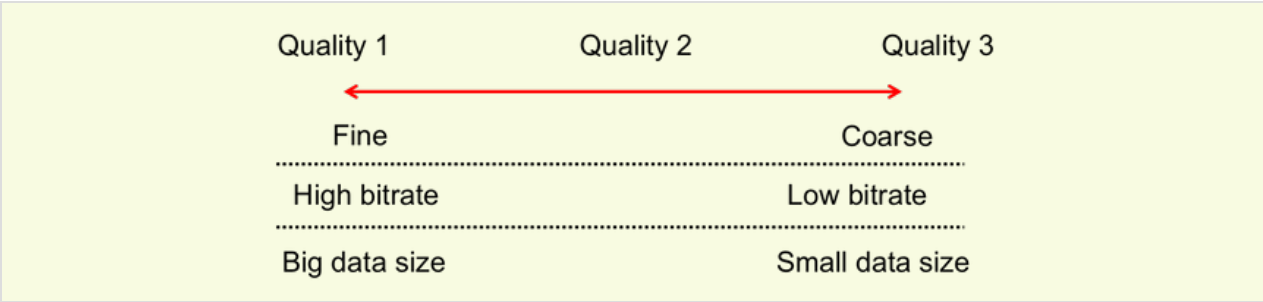
Tile Type	Resolution [px]			
	Level 1	Level 2	Level 3	Level 4
HD	256 x 144	512 x 288	768 x 432	1024 x 576
SD	128 x 96	256 x 192	384 x 288	512 x 384
R	96 x 64	192 x 128	288 x 192	384 x 256
S	64 x 64	128 x 128	192 x 192	256 x 256

If tile’s time duration that you specified is longer than the registered media’s duration, the duration is cut down to the registered media’s duration.
Similarly, if zero or no value has been specified for tile’s duration, tile’s duration is set to the media’s duration.
The media’s duration of less than one second will be rounded up and the time of the fraction will be filled with the last frame of the media.

> **Note**
> The left, right, top, and bottom edge part (each 2px) of the combined video for each level tile is filled with black line, thus “Resolution” above specification table includes the black line.

About Tile quality

Tile have the concept of Quality. User can set three level quality that is described between 1~3 quality like below.



Multiple quality can be set in media upload phase, also single quality can be set in fv generation and fv request. By doing so, user can

select quality according to services or network bandwidth.

Media Data

Gnzo holds the created tiles and the meta-data of media only.

Meta Data	input character qualification
media_id	half-width digit
caption	half-width alphanumeric、full-width character、symbol　(1 - 120 characters)
tag	half-width alphanumeric、underline (1 - 120 characters) multiple input is available by inserting comma.
description	half-width alphanumeric、full-width character、symbol　(1 - 1000 characters)
create_date	yyyy-mm-dd hh:mm:ss
update_date	yyyy-mm-dd hh:mm:ss
status	0,1

Maximum size of fabric video

If tile type is same, tile can be combined each other, which creates fabric video (called fv), but the number of tiles to be combined are limited, described as below table.

tle type	max size	
	column × row [number of tiles]	resolution [px]
HD	7x7	1792x1008
SD	11x11	1408x1056
R	15x15	1440x960
S	16x16	1024x1024

About API authentication

The API server authenticates client which calls API.
There are two authentication method which are Referrer+Key and Basic authentication.
Please note that the api for (D) marked row line would be deprecated in the future.

API		Referrer + Key	Basic
Media registration API	POST /api/1/videos	×	○
fv cretion request API	GET /api/1/videos/fv	○	○
	GET /api/1/videos/fv/timeline	○	○
	GET /api/1/videos/fv/random	○	○
	GET /api/1/videos/fv/search	○	○
	GET /api/1/admin/fv	×	○
Media management API	GET /api/1/videos/media	○	○
	GET /api/1/videos/{media_id}/info (D)	×	○
	GET /api/1/videos/info	×	○
	POST /api/1/videos/{media_id}/info	×	○
	POST /api/1/videos/{media_id}/activate (D)	×	○
	POST /api/1/videos/activate	×	○
	POST /api/1/videos/{media_id}/deactivate (D)	×	○
	POST /api/1/videos/deactivate	×	○

	DELETE /api/1/videos/{media_id} (D)	×	○
	POST /api/1/videos/delete	×	○
	GET /api/1/videos/{media_id}/tile-status	×	○
	GET /api/1/videos/tile	×	○
	GET /api/1/videos/search	×	○
	GET /api/1/admin/bill_plan	×	○
	POST /api/1/videos/{media_id}/cancel	×	○

If ○ symbol is marked on both (Referrer+Key and Basic), first API server will perform Basic authentication. After that, if it fails, API server will perform Referrer + Key(or Key only) authentication.
The basic authentication requires access over HTTPS. If the client try to basic authentication with no HTTPS, the 403 Error will occur.

File Format for Media Registration

The file format and Video/Audio Codec for Media Registration are described.

File Format	Type	Codec
MPG	Video	MPEG1
		MPEG2
	Audio	MP1
		MP2
		MP3
MOV	Video	MPEG1
		MPEG2
		MPEG4
		MPEG4 AVC/H.264
		H.263
		MOTION JPEG
	Audio	MP3
		AAC
		APPLE LOSSLESS
		AIFF
MP4	Video	MPEG1
		MPEG2
		MPEG4
		MPEG4 AVC/H.264
	Audio	MP1
		MP2
		MP3
		AAC
		AC3
		APPLE LOSSLESS
WebM	Video	VP8
	Audio	VORBIS

AVI	Video	MPEG1
		MPEG2
		MPEG4
		MPEG4 AVC/H.264
		H.263
		H.261
		VP8
		MS-MPEG4
		VC1
		THEORA
		MOTION JPEG
	Audio	MP3 LPCM
		AAC
		AC3
		LPCM
		FLAC
FLV	Video	MPEG4 AVC/H.264
	Audio	VP6
		MP3
		AAC
		PCM
F4V	Video	MPEG4 AVC/H.264
	Audio	MP3
		AAC
OGG	Video	THEORA
	Audio	VORBIS
		FLAC
MKV	Video	MPEG1
		MPEG2
		MPEG4
		MPEG4 AVC/H.264
		VP8
		WMV
		VC1
		THEORA
		MOTION JPEG
	Audio	MP2
		MP3
		AAC
		AC3
		WMA

		VORBIS
		FLAC
ASF	Video	WMV
		VC1
		MS-MPEG4
	Audio	WMA
		LPCM
		MP3

Media registration API

Video registration

This API registers a video to API server.After the video registration, tile creation process starts automatically. さらに特定のパラメータを追加することで、自動タグ付けやnsfw(not safe for work)に 該当する動画の確率を返すことが可能になります。また、動画中から音声（日本語のみ）を取得 及び解析し、文字列を返すことも可能です。

```
POST /api/1/videos
```

Note

About media activation：Media status will not be active after running this API yet. You need to activate the media to render it as fv. Please performs media activation API after tile creation is finished. [Appendix] 音声認識パラメータを付けて動画登録した場合は、動画から音声が抽出・解析が行われ、その中で名詞をデータベースに登録します。 登録された音声タグは[音声タグ取得API]により取得することが出来ます。

Parameters

- **output_type**
  - Response Type
    - x: Output in XML format
    - blank: Output in JSON format
- **upfile** [required]
  - Upload file is Up to 200MB
- **tile_type** [required]
  - Tile Type. you can specify only one type.
  - hd 16:9 / sd 4:3 / r 3:2 / s 1:1
- **duration**
  - Duration of the tile for creation (1-900 seconds)
    Please specify the time duration within the registration video's time duration.The duration will be the time duration of posting video automatically if you specify a value of time duration which exceeds it.[Appendix]
- **levels**
  - This parameter specifies the level of the tile to be created. Multiple input is available by inserting comma. if this part is not specified, it creates level1 tiles only.
    Level 1 tiles are automatically created in the video registration process.
    ex)
    If level = 2 is specified: tile level 1 and 2 are created.
    If level = 3,4 is specified: tile level of 1, 3, and 4 are created.
- **tile_qualitys**
  - This parameter specifies the quality of the tile to be created. Multiple input is available by inserting comma. If this parameter is not specified, it returns error.
    ex)
    If tile_qualitys=2 is specified, tile with quality=2 is created.

If tile_qualitys=1,2,3 are specified, tile with quality=1,2,3 is created.

- **audio_recog** [オプション][本機能を利用するためには，別途お問い合わせ下さい．]
  - 登録された動画から音声情報を抽出・解析し、AudioTagとして登録するパラメータです。
    指定する場合は、”audio_recog=1”を指定して下さい。
    （但し本パラメータを指定すると動画登録プロセス処理速度が数秒遅くなります。）

- **autotag** [オプション][本機能を利用するためには，別途お問い合わせ下さい．]
  - 登録された動画に対して画像・動画解析を行い、自動タグ付けを行うパラメータです。
    指定する場合は”autotag=1”を指定して下さい。
    (但し本パラメータを指定すると動画登録プロセス処理速度が数秒遅くなります。)

- **nsfw** [オプション][本機能を利用するためには，別途お問い合わせ下さい．]
  - 登録された動画に対して画像・動画解析を行い、該当メディアがNot-Safe-For-Workである確率（百分率）の
    計算を行うパラメータです。
    指定する場合は”nsfw=1”を指定して下さい。
    (但し本パラメータを指定すると動画登録プロセス処理速度が数秒遅くなります。)

- **thumbnail_position**
  - you can specify up to 10 thumbnail positions (which is needed to specify by frame) you want to make thumbnail image.
    Specifies the frame based on the frame rate of the API Reference video registration.
    Default thumbnail image (0 frame) is generated even if any frame are specified.
    ex)
    thumbnail_position = 5 ->
    Two thumbnails are generated (0 and 5 frames of the video).

- **crop_position**
  - When input video resolution is larger than output one, this parameter is required.
    1 : Center
    2 : Top-left
    3 : Top-right
    4 : Bottom-left
    5 : Bottom-right

- **padding_position**
  - When input video resolution is smaller than output one, this parameter is required.
    1 : Center
    2 : Top-left
    3 : Top-right
    4 : Bottom-left
    5 : Bottom-right

- **crop_inheriting**
  - The cropping / padding policy between multiple levels.
    0 (On) : Applying for same cropping / padding area
    1 (Off) : Optimizing cropping area each levels.

- **crop_policy_ob**
  - The policy for cropping / padding when output resolution is bigger than input one
    0 : [Cover] The resizing process with keeping aspect ratio.
    1 : [Contain] The resizing process with keeping aspect ratio.
    2 : [Fill] The resizing process without keeping aspect ratio.
    3 : [Dot by dot] The easy padding

- **crop_policy_ib**
  - The policy for cropping / padding when input resolution is bigger than output one.
    0 : [Cover] The resizing process with keeping aspect ratio.
    1 : [Contain] The resizing process with keeping aspect ratio.
    2 : [Fill] The resizing process without keeping aspect ratio.
    3 : [Dot by dot] The easy cropping

Sample Curl Command

- **Basic認証を使用した方法**

```
curl --basic -u {your account}:{your password} \
  "https://example.com/api/1/videos?tile_type=hd&duration=10&levels=1,2&tile_qualitys=2&autotag=1&nsfw=1&audio_recog=1" \
  -X POST -F "upfile=@{upload_file_path}"
```

- **AccessToken認証を使用した方法**

```
curl "https://example.com/api/1/videos?tile_type=hd&duration=10&levels=1,2&tile_qualitys=2&autotag=1&nsfw=1&audio_recog=1" \
  -H "Authorization: Bearer {your access_token}" \
  -X POST -F "upfile=@{upload_file_path}"
```

Response

- **status**
  - Result of Process
    success: sucess
    other: failure
- **media_id**
  - Media ID registered
- **autotags**
  - 動画登録オプションにより”autotag=1”が指定された場合に、本パラメータが返却されます。
    配列であり，自動メディア解析及び付与されたタグが挿入されています。
    タグの数はメディアに応じて変わります。
- **nsfw**
  - 動画登録オプションにより”nsfw=1”が指定された場合に、本パラメータが返却されます。
    該当メディアがNot-Safe-For-Workである確率を百分率で返します。
- **audio_recognition**
  - 動画登録オプションにより”audio_recog=1”が指定され、かつ正常に分析が成功した場合に、
    本パラメータが返却されます。
    該当メディアからの音声を抽出し、その解析結果を文字列として返却致します。

Response Example - JSON

```
{
  "media_id":474,
  "status":"success",
  "autotags":["technology","screen"],
  "nsfw":"0.0019919699989259",
  "audio_recognition":["愛する奥さんへ","いつも朝早く起きておにぎりを握ってくれていつもありがとうあなたのおにぎりのおかげでいつも仕
}
```

Response Example - XML

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
  <media_id>475</media_id>
  <status>success</status>
  <autotags>
      <entry>technology</entry>
      <entry>screen</entry>
  </autotags>
  <nsfw>0.0019919699989259</nsfw>
```

fv creation request API

Acquisition of specified order based fv

This API generate a fv in specified tile order.
The tile to be displayed using this API are tiles which is activated with media activation API.

```
GET /api/1/videos/fv
```

Note

There are two ways to specify the order of tile that make up the fv.
Required parameters change as follows according to each way. It is not possible to mix medias which have different tile type and different tile quality.

1. The way to create a fv with comma-separated media ID and level [Appendix]
   Required parameters: media_ids and (levels or level)
2. The way to create a fv with a map of JSON format [Appendix]
   Required parameters: map_data

Parameters

- **key** [required]
  - Client authorization key
- **output_type**
  - Response type
    - x: Output in XML format
    - blank: Output in JSON format
- **duration**
  - Time of fv to generate (1～900seconds)
    An integer [Appendix]
- **loop**
  - parameter for determining the length of fv [Appendix]
- **row** [required]
  - Number of rows of fv to generate
- **col** [required]
  - Number of columns of fv to generate
- **media_ids**
  - media ID that constitutes the fv
- **levels**
  - Specifies the level of each media ID specified in media_ids
- **level**
  - Specifies a level collectively for each media ID specified in media_ids
- **map_data**
  - map data in JSON format
- **sound_ids**
  - The media ID to add the sound of media ID to the fv. [Appendix]
- **tile_type**
  - Tile type
    hd 16:9 / sd 4:3 / r 3:2 / s 1:1
    For fv creation with map_data, if media ID is not specified in the map_data, tile_type can be retrieved. Thus tile_type is required in this case.
- **tile_quality**
  - Tile Quality
    1 :High / 2 :Middle / 3 :Low

For fv creation with map_data, if media ID is not specified in the map_data, tile_quality is needed. Thus tile_quality is required in this case.

- **start_time**
  - specifies the start time of each media that is specified in the media_ids. [Appendix]
- **media_duration**
  - specifies the time of each media that is specified in the media_ids. [Appendix]

Response

- **status**
  - result of process
    success: success
    other: failure
- **video_url**
  - fv URL
- **thumbnail_url**
  - fv thumbnail URL
- **cahched_data**
  - This is the flag whether fabric video's mp4 file already have been genearted or not. If fabric video's mp4 file alread have been generated, this flag should specify "1", if generating process is running, this flag should specify "0".
- **black_edge**
  - 生成されたfvのmp4ファイルの周り（上と左右）に16pixの黒色の帯が入っている場合には、"1"が返り、16pixの黒色の帯が入っていないならば、"0"が入る。
    黒帯を入れるか否かはfvLIBRARY環境作成時にのみ決定可能であり、途中からの変更は出来ないので、留意して下さい。（詳細はお問い合わせ下さい。）
    本パラメータは基本的にバージョン互換性を持たせるためのパラメータであり、基本的にはblack_edgeパラメータには"0"が入る。
- **map_data**
  - map data

Request Example

Media_ids and levels

```
GET
http://example.com/api/1/videos/fv?
key=xxxx&output_type=x&duration=60&row=1&col=2&
media_ids=59,58&levels=1,1&sound_ids=87
```

Map data

```
GET
http://example.com/api/1/videos/fv?
key=xxxx&output_type=x&duration=60&row=1&col=2&
map_data=[[ { "id":"59", "lev":"1", "rx":"0", "ry":"0" },
{ "id":"58", "lev":"1", "rx":"0", "ry":"0" } ]]
&sound_ids=87
```

Sample Curl Command

- **Basic認証を使用した方法(media_idsとlevelsで指定)**

```
curl --basic -u {your account}:{your password} "https://example.com/api/1/videos/fv?duration=10&row=2&col=2&tile_type=hd&tile_quality=2&
```

- **AccessToken認証を使用した方法(media_idsとlevelsで指定)**

```
curl "https://example.com/api/1/videos/fv?duration=10&row=2&col=2&tile_type=hd&tile_quality=2&media_ids=472,473,474,475&level=1" \
-H "Authorization: Bearer {your access_token}"
```

Response Example - JSON


```
{
   "status":"success",
   "video_url":"http:\/\/example.com\/fv\/gnzo-xxxxxxxxxxxx.mp4",
   "thumbnail_url":"http:\/\/example.com\/fvt\/gnzo-xxxxxxxxxx.png",
   "cached_data":"1",
   "black_edge":"0",
   "map_data":[
      [
         {
         "id":"59",
         "lev":"1",
         "all-lev":[
            "1",
            "2"
         ],
         "audio":"1",
         "rx":"0",
         "ry":"0",
         "caption":"caption test",
         "description":"Description test"
        },
        {
         "id":"58",
         "lev":"1",
         "all-lev":[
            "1",
            "2"
         ],
         "audio":"1",
         "rx":"0",
         "ry":"0",
         "caption":"",
         "description":""
        }
      ]
   ]
}
```


Response Example - XML


```
<?xml version="1.0" encoding="UTF-8"?>
<response>
<status>success</status>
<has_next>1</has_next>
<video_url>http://example.com/fv/gnzo-xxxxxxxxxx.mp4</video_url>
<thumbnail_url>http://example.com/fvt/gnzo-xxxxxxxxxxxxx.png</thumbnail_url>
<cached_data>1</cached_data>
<black_edge>0</bloack_edge>
<map_data>
<entry>
   <entry>
   <id>59</id>
   <lev>1</lev>
```

```
    <all-leve>
      <entry>1</entry>
      <entry>2</entry>
    </all-leve>
    <audio>1</audio>
    <rx>0</rx>
    <ry>0</ry>
    <caption>test caption</caption>
    <description>test description</description>
    </entry>
    <entry>
      <id>58</id>
      <lev>1</lev>
      <all-leve>
        <entry>1</entry>
        <entry>2</entry>
      </all-leve>
      <audio>1</audio>
      <rx>0</rx>
      <ry>0</ry>
      <caption />
      <description />
    </entry>
    </entry>
<entry>
    <entry>
      <id>57</id>
      <lev>1</lev>
      <all-leve>
        <entry>1</entry>
        <entry>2</entry>
      </all-leve>
      <audio>1</audio>
      <rx>0</rx>
      <ry>0</ry>
      <caption />
      <description />
    </entry>
    <entry>
      <id>52</id>
      <lev>1</lev>
      <all-leve>
        <entry>1</entry>
        <entry>2</entry>
      </all-leve>
      <audio>1</audio>
      <rx>0</rx>
      <ry>0</ry>
      <caption />
      <description />
    </entry>
</entry>
</map_data>
</response>
```

Acquisition of timeline order based fv

This API generate a fv in descending order of registration time.
Tile configuration of fv is with only level 1 and same tile type.

```
GET /api/1/videos/fv/timeline
```

Parameters

- **key** [required]
  - Client authorization key
- **output_type**
  - Response type
    - x: Output in XML format
    - blank: Output in JSON format
- **duration**
  - Time of fv to generate (1〜900seconds)
    An integer [Appendix]
- **loop**
  - parameter for determining the length of fv [Appendix]
- **row** [required]
  - Number of rows of fv to generate
- **col** [required]
  - Number of columns of fv to generate
- **tile_type** [required]
  - Tile type
    hd 16:9 / sd 4:3 / r 3:2 / s 1:1
- **tile_quality** [required]
  - Tile Quality. Single Tile Quality can be specified.
    1 :High / 2 :Middle / 3 :Low
- **sound_ids**
  - The media ID to add the sound into the fv. [Appendix]
- **page**
  - Page which will be rendered. Integer greater than 1. [Appendix]
- **keywords** [オプション]
  - 検索キーワード（実際には中括弧{}は入れません。）
    - #{string} : タグ検索（完全一致検索）
      - 文字列キーワードの直前に#をつけて下さい。
    - {string} （OR検索指定）または {“string”} （AND検索指定）: キャプション，デスクリプション検索（部分一致AND/OR検索）
    - &{integer} : ID検索（部分一致検索）
  - 但し，#(シャープ)は%23とURLエンコードされている必要があり、&(アンパサンド)は%26とURLエンコードされている必要があることに注意して下さい。
  - またスペースは%20です。
  - ルールや指定方法詳細については，[詳細]を参照

Response

- **status**
  - result of process
    success: success
    other: failure
- **has_next**
  - existence on the next page
    true: exist
    false: not exist
- **video_url**

  - fv URL

- **thumbnail_url**
  - fv thumbnail URL

- **cahched_data**
  - This is the flag whether fabric video's mp4 file already have been genearted or not. If fabric video's mp4 file alread have been generated, this flag should specify "1", if generating process is running, this flag should specify "0".

- **black_edge**
  - 生成されたfvのmp4ファイルの周り（上と左右）に16pixの黒色の帯が入っている場合には、"1"が返り、16pixの黒色の帯が入っていないならば、"0"が入る。
    黒帯を入れるか否かはfvLIBRARY環境作成時にのみ決定可能であり、途中からの変更は出来ないので、留意して下さい。（詳細はお問い合わせ下さい。）
    本パラメータは基本的にバージョン互換性を持たせるためのパラメータであり、基本的にはblack_edgeパラメータには"0"が入る。

- **map_data**
  - map data

Request Example

```
GET
http://example.com/api/1/videos/fv/timeline?
key=xxx&duration=6&row=2&col=2&tile_type=s&tile_quality=2
```

Sample Curl Command

```
curl  "https://example.com/api/1/videos/fv/timeline?key={your api key}&duration=10&row=2&col=2&tile_type=hd&tile_quality=2&keywords=%23h
```

Response Example - JSON


```
{
   "status":"success",
   "has_next":true,
   "video_url":"http:\/\/example.com\/fv\/gnzo-xxxxxxxxxx.mp4",
   "thumbnail_url":"http:\/\/example.com\/fvt\/gnzo-xxxxxxxxxx.png",
   "cached_data":"1",
   "black_edge":"0",
   "map_data":[
      [
         {
            "id":"59",
            "lev":"1",
            "all-lev":[
               "1",
               "2"
            ],
            "audio":"1",
            "rx":"0",
            "ry":"0",
            "caption":"caption test",
            "description":"description test"
         },
         {
            "id":"58",
            "lev":"1",
            "all-lev":[
               "1",
               "2"
```

```
            ],
            "audio":"1",
            "rx":"0",
            "ry":"0",
            "caption":"",
            "description":""
         }
      ],
      [
         {
            "id":"57",
            "lev":"1",
            "all-lev":[
              "1",
              "2"
            ],
            "audio":"1",
            "rx":"0",
            "ry":"0",
            "caption":"",
            "description":""
         },
         {
            "id":"52",
            "lev":"1",
            "all-lev":[
              "1",
              "2"
            ],
            "audio":"1",
            "rx":"0",
            "ry":"0",
            "caption":"",
            "description":""
         }
      ]
   ]
}
```


Response Example - XML


```
<?xml version="1.0" encoding="UTF-8"?>
<response>
<status>success</status>
<has_next>1</has_next>
<video_url>http://example.com/fv/gnzo-xxxxxxxxxx.mp4</video_url>
<thumbnail_url>http://example.com/fvt/gnzo-xxxxxxxxxxxxxx.png</thumbnail_url>
<cached_data>1</cached_data>
<black_edge>0</bloack_edge>
<map_data>
<entry>
   <entry>
   <id>59</id>
   <lev>1</lev>
   <all-leve>
      <entry>1</entry>
      <entry>2</entry>
   </all-leve>
   <audio>1</audio>
```

```
    <rx>0</rx>
    <ry>0</ry>
    <caption>test caption</caption>
    <description>test description</description>
    </entry>
    <entry>
    <id>58</id>
    <lev>1</lev>
    <all-leve>
      <entry>1</entry>
      <entry>2</entry>
    </all-leve>
    <audio>1</audio>
    <rx>0</rx>
    <ry>0</ry>
    <caption />
    <description />
    </entry>
</entry>
<entry>
    <entry>
    <id>57</id>
    <lev>1</lev>
    <all-leve>
      <entry>1</entry>
      <entry>2</entry>
    </all-leve>
    <audio>1</audio>
    <rx>0</rx>
    <ry>0</ry>
    <caption />
    <description />
    </entry>
    <entry>
    <id>52</id>
    <lev>1</lev>
    <all-leve>
      <entry>1</entry>
      <entry>2</entry>
    </all-leve>
    <audio>1</audio>
    <rx>0</rx>
    <ry>0</ry>
    <caption />
    <description />
    </entry>
</entry>
</map_data>
</response>
```

Acquisition of random order based fv

This API generates a fv in random order.
Tile configuration of fv is with only level 1 and same tile type.

```
GET /api/1/videos/fv/random
```

Parameters

- **key** [required]
  - Client authorization key
- **output_type**
  - Response type
    - x: Output in XML format
    - blank: Output in JSON format
- **duration**
  - Time of fv to generate (1〜900seconds)
    An integer [Appendix]
- **loop**
  - parameter for determining the length of fv [Appendix]
- **row** [required]
  - Number of rows of fv to generate
- **col** [required]
  - Number of columns of fv to generate
- **tile_type** [required]
  - Tile type
    hd 16:9 / sd 4:3 / r 3:2 / s 1:1
- **tile_quality** [required]
  - Tile Quality. Single Tile Quality can be specified.
    1 :High / 2 :Middle / 3 :Low
- **sound_ids**
  - The media ID to add the sound into the fv. [Appendix]
- **keywords** [オプション]
  - 検索キーワード（実際には中括弧{}は入れません。）
    - #{string} : タグ検索（完全一致検索）
      - 文字列キーワードの直前に#をつけて下さい。
    - {string} （OR検索指定）または {“string”} （AND検索指定）: キャプション，デスクリプション検索（部分一致AND/OR検索）
    - &{integer} : ID検索（部分一致検索）
  - 但し，#(シャープ)は%23とURLエンコードされている必要があり、&(アンパサンド)は%26とURLエンコードされている必要があることに注意して下さい。
  - またスペースは%20です。
  - ルールや指定方法詳細については，[詳細]を参照

Response

- **status**
  - result of process
    success: success
    other: failure
- **video_url**
  - fv URL
- **thumbnail_url**
  - fv thumbnail URL
- **cahched_data**
  - This is the flag whether fabric video’s mp4 file already have been genearted or not. If fabric video’s mp4 file alread have been generated, this flag should specify “1”, if generating process is running, this flag should specify “0”.
- **black_edge**
  - 生成されたfvのmp4ファイルの周り（上と左右）に16pixの黒色の帯が入っている場合には、”1”が返り、16pixの黒色の帯が入っていないならば、”0”が入る。
    黒帯を入れるか否かはfvLIBRARY環境作成時にのみ決定可能であり、途中からの変更は出来ないので、留意して下さい。（詳細はお問

い合わせ下さい。）
本パラメータは基本的にバージョン互換性を持たせるためのパラメータであり、基本的にはblack_edgeパラメータには”0”が入る。

- **map_data**
  - map data

Request Example

```
GET
http://example.com/api/1/videos/fv/random?
key=xxxxx&duration=6&row=2&col=2&tile_type=s&tile_quality=2
```

Sample Curl Command

```
curl  "https://example.com/api/1/videos/fv/random?key={your api key}&duration=10&row=2&col=2&tile_type=hd&tile_quality=2&keywords=%23h
```

Response Example - JSON


```
{
   "status":"success",
   "video_url":"http:\/\/example.com\/fv\/gnzo-xxxxxxxxxxx.mp4",
   "thumbnail_url":"http:\/\/example.com\/fvt\/gnzo-xxxxxxxxxxxx.png",
   "cached_data":"1",
   "black_edge":"0",
   "map_data":[
      [
         {
            "id":"9",
            "lev":"1",
            "all-lev":[
              "1",
              "2"
            ],
            "audio":"1",
            "rx":"0",
            "ry":"0",
            "caption":"test caption ",
            "description":"test description"
         },
         {
            "id":"8",
            "lev":"1",
            "all-lev":[
              "1",
              "2"
            ],
            "audio":"1",
            "rx":"0",
            "ry":"0",
            "caption":"test caption ",
            "description":"test description"
         }
      ],
      [
         {
            "id":"10",
            "lev":"1",
            "all-lev":[
```

```
            "1",
            "2"
          ],
          "audio":"1",
          "rx":"0",
          "ry":"0",
          "caption":"test caption",
          "description":"test description"
        },
        {
          "id":"39",
          "lev":"1",
          "all-lev":[
            "1",
            "2"
          ],
          "audio":"1",
          "rx":"0",
          "ry":"0",
          "caption":"",
          "description":""
        }
      ]
    ]
}
```


Response Example - XML


```
<?xml version="1.0" encoding="UTF-8"?>
<response>
<status>success</status>
<video_url>http://example.com/fv/gnzo-xxxxxxxxxx.mp4</video_url>
<thumbnail_url>http://example.com/fvt/gnzo-xxxxxxxxxxxx.png</thumbnail_url>
<cached_data>1</cached_data>
<black_edge>0</black_edge>
<map_data>
  <entry>
    <entry>
      <id>32</id>
      <lev>1</lev>
      <all-leve>
        <entry>1</entry>
        <entry>2</entry>
      </all-leve>
      <audio>1</audio>
      <rx>0</rx>
      <ry>0</ry>
      <caption />
      <description />
    </entry>
    <entry>
      <id>3</id>
      <lev>1</lev>
      <all-leve>
        <entry>1</entry>
        <entry>2</entry>
      </all-leve>
      <audio>1</audio>
      <rx>0</rx>
```

```
        <ry>0</ry>
        <caption>test caption</caption>
        <description>test description</description>
      </entry>
    </entry>
    <entry>
      <entry>
        <id>14</id>
        <lev>1</lev>
        <all-leve>
          <entry>1</entry>
          <entry>2</entry>
        </all-leve>
        <audio>1</audio>
        <rx>0</rx>
        <ry>0</ry>
        <caption>test caption</caption>
        <description>test description</description>
      </entry>
      <entry>
        <id>19</id>
        <lev>1</lev>
        <all-leve>
          <entry>1</entry>
          <entry>2</entry>
        </all-leve>
        <audio>1</audio>
        <rx>0</rx>
        <ry>0</ry>
        <caption />
        <description />
      </entry>
    </entry>
</map_data>
</response>
```

Acquisition of keyword search based fv

This API generates a fv based on the keywords search results.
Tag will be searched with perfect matching and caption will be searched with partial match. Tile configuration of fv is with only level 1 and same tile type.

```
GET /api/1/videos/fv/search
```

Parameters

- **key** [required]
  - Client authorization key
- **output_type**
  - Response type
    - x: Output in XML format
    - blank: Output in JSON format
- **duration**
  - Time of fv to generate (1～900seconds)
    An integer [Appendix]
- **loop**
  - parameter for determining the length of fv [Appendix]

- **row** [required]
  - Number of rows of fv to generate
- **col** [required]
  - Number of columns of fv to generate
- **tile_type** [required]
  - Tile type
    hd 16:9 / sd 4:3 / r 3:2 / s 1:1
- **tile_quality** [required]
  - Tile Quality. Single Tile Quality can be specified.
    1 :High / 2 :Middle / 3 :Low
- **sound_ids**
  - The media ID to add the sound into the fv. [Appendix]
- **page**
  - Page which will be rendered. Integer greater than 1. [Appendix]
- **keywords** [required]
  - search keyword
    # (characters): tag search (Perfect match)
    (characters): caption and description search (Partial match)
- **keywords** [required]
  - search keyword (middle-bracket{} is not required in practice.)
    - #{string} : Tag search (Exact match)
      - Append # just before keyword string.
    - {string} (OR search) / {“string”} (AND search) : search string in caption/description (AND/OR partial search)
    - &{integer} : ID search (Partial search)
  - Provided that, # should be URL encoded to %23, and &, ampersand, should be URL encoded to %26
  - In addition, Space should be URL encoded to %20.
  - Please refer to [Detail] for search rule detail.

Response

- **status**
  - result of process
    success: success
    other: failure
- **has_next**
  - existence on the next page
    true: exist
    false: not exist
- **video_url**
  - fv URL
- **thumbnail_url**
  - fv thumbnail URL
  - This is the flag whether fabric video’s mp4 file already have been genearted or not. If fabric video’s mp4 file alread have been generated, this flag should specify “1”, if generating process is running, this flag should specify “0”.
- **cahched_data**
  - This is the flag whether fabric video’s mp4 file already have been genearted or not. If fabric video’s mp4 file alread have been generated, this flag should specify “1”, if generating process is running, this flag should specify “0”.
- **black_edge**
  - 生成されたfvのmp4ファイルの周り（上と左右）に16pixの黒色の帯が入っている場合には、”1”が返り、16pixの黒色の帯が入っていないならば、”0”が入る。
    黒帯を入れるか否かはfvLIBRARY環境作成時にのみ決定可能であり、途中からの変更は出来ないので、留意して下さい。（詳細はお問

い合わせ下さい。）
本パラメータは基本的にバージョン互換性を持たせるためのパラメータであり、基本的にはblack_edgeパラメータには”0”が入る。

- **map_data**
  - map data

Request Example

```
GET
http://example.com/api/1/videos/fv/search?
key=xxxxxx&duration=30&row=2&col=2&tile_type=hd&tile_quality=2&
sound_ids=171&page=1&keywords=Movie
```

Sample Curl Command

```
curl  "https://example.com/api/1/videos/fv/search?key={your api key}&duration=10&row=2&col=2&tile_type=hd&tile_quality=2&keywords=%23hc
```

Response Example - JSON


```
{
   "status":"success",
   "has_next":true,
   "video_url":"http:\/\/example.com\/fv\/gnzo-xxxxxxxxxx.mp4",
   "thumbnail_url":"http:\/\/example.com\/fvt\/gnzo-xxxxxxxxxxxxxx.png",
   "cached_data":"1",
   "black_edge":"0",
   "map_data":[
      [
         {
         "id":"17",
         "lev":"1",
         "all-lev":[
             "1",
             "2"
         ],
         "audio":"1",
         "rx":"0",
         "ry":"0",
         "caption":" Test Movie 1",
         "description":" Description 17"
         "map_id":null
         },
         {
         "id":"16",
         "lev":"1",
         "all-lev":[
             "1",
             "2"
         ],
         "audio":"1",
         "rx":"0",
         "ry":"0",
         "caption":" Test Movie 2",
         "description":" Description 16"
         "map_id":null
         }
      ],
      [
```

```
        {
        "id":"15",
        "lev":"1",
        "all-lev":[
            "1",
            "2"
        ],
        "audio":"1",
        "rx":"0",
        "ry":"0",
        "caption":" Test Movie 3",
        "description":" Description 15"
        "map_id":null
        },
        {
        "id":"14",
        "lev":"1",
        "all-lev":[
            "1",
            "2"
        ],
        "audio":"1",
        "rx":"0",
        "ry":"0",
        "caption":" Test Movie 4",
        "description":" Description 14"
        "map_id":null
        }
    ]
  ]
}
```


Response Example - XML


```
<?xml version="1.0" encoding="UTF-8"?>
<response>
<status>success</status>
<has_next>1</has_next>
<video_url>http://example.com/fv/gnzo-xxxxxxxxxxxxx.mp4</video_url>
<thumbnail_url>http://example.com/fvt/gnzo-xxxxxxxxxxxxxxxx.png</thumbnail_url>
<cached_data>1</cached_data>
<black_edge>0</bloack_edge>
<map_data>
  <entry>
    <entry>
      <id>17</id>
      <lev>1</lev>
      <all-leve>
        <entry>1</entry>
        <entry>2</entry>
      </all-leve>
      <audio>1</audio>
      <rx>0</rx>
      <ry>0</ry>
      <caption>Test Movie 1</caption>
      <description>Description 17</description>
      </map_id>
    </entry>
    <entry>
```

```
        <id>16</id>
        <lev>1</lev>
        <all-leve>
          <entry>1</entry>
          <entry>2</entry>
        </all-leve>
        <audio>1</audio>
        <rx>0</rx>
        <ry>0</ry>
        <caption> Test Movie 2</caption>
        <description> Description 16</description>
        </map_id>
      </entry>
    </entry>
    <entry>
      <entry>
        <id>15</id>
        <lev>1</lev>
        <all-leve>
          <entry>1</entry>
          <entry>2</entry>
        </all-leve>
        <audio>1</audio>
        <rx>0</rx>
        <ry>0</ry>
        <caption> Test Movie 3</caption>
        <description> Description 15</description>
        </map_id>
      </entry>
      <entry>
        <id>14</id>
        <lev>1</lev>
        <all-leve>
          <entry>1</entry>
          <entry>2</entry>
        </all-leve>
        <audio>1</audio>
        <rx>0</rx>
        <ry>0</ry>
        <caption> Test Movie 4</caption>
        <description> Description 14</description>
        </map_id>
      </entry>
    </entry>
</map_data>
</response>
```

Acquisition of specified order based fv for admin

This API generate a fv in specified order.
This API is for administrators and the target media are included inactive media unlike API: Acquisition of specified order based fv.
Before media activation, you can check the registered media data.

```
GET /api/1/admin/fv
```

Note

There are two ways to specify the order of media that make up the fv. Required parameters change as follows according to each way. It is not possible to mix medias which have different tile type and different tile quality.

1. The way to create a fv with comma-separated media ID and level [Appendix]
   Required parameters: media_ids and (levels or level)
2. The way to create a fv with a map of JSON format [Appendix]
   Required parameters: map_data

Parameters

- **output_type**
  - Response type
    - x: Output in XML format
    - blank: Output in JSON format
- **duration**
  - Time of fv to generate (1～900seconds)
    An integer [Appendix]
- **loop**
  - parameter for determining the length of fv [Appendix]
- **row** [required]
  - Number of rows of fv to generate
- **col** [required]
  - Number of columns of fv to generate
- **media_ids**
  - media ID that constitutes the fv
- **levels**
  - Specifies the level of each media ID specified in media_ids
- **level**
  - Specifies a level collectively for each media ID specified in media_ids
- **map_data**
  - map data in JSON format
- **sound_ids**
  - The media ID to add the sound of media ID to the fv. [Appendix]
- **tile_type**
  - Tile type
    hd 16:9 / sd 4:3 / r 3:2 / s 1:1
    For fv creation with map_data, if media ID is not specified in the map_data, tile_type can be retrieved. Thus tile_type is required in this case.
- **tile_quality**
  - Tile Quality
    1 :High / 2 :Middle / 3 :Low
    For fv creation with map_data, if media ID is not specified in the map_data, tile_quality is needed. Thus tile_quality is required in this case.
- **start_time**
  - specifies the start time of each media that is specified in the media_ids. [Appendix]
- **media_duration**
  - specifies the time of each media that is specified in the media_ids. [Appendix]

Response

- **status**
  - result of process
    success: success
    other: failure
- **video_url**

  - fv URL

- **thumbnail_url**
  - fv thumbnail URL

- **cahched_data**
  - This is the flag whether fabric video's mp4 file already have been genearted or not. If fabric video's mp4 file alread have been generated, this flag should specify "1", if generating process is running, this flag should specify "0".

- **black_edge**
  - 生成されたfvのmp4ファイルの周り（上と左右）に16pixの黒色の帯が入っている場合には、"1"が返り、16pixの黒色の帯が入っていないならば、"0"が入る。
    黒帯を入れるか否かはfvLIBRARY環境作成時にのみ決定可能であり、途中からの変更は出来ないので、留意して下さい。（詳細はお問い合わせ下さい。）
    本パラメータは基本的にバージョン互換性を持たせるためのパラメータであり、基本的にはblack_edgeパラメータには"0"が入る。

- **map_data**
  - map data

Request Example

media_ids and levels

```
GET
https://example.com/api/1/admin/fv?
output_type=x&duration=60&row=1&col=2&
media_ids=59,58&levels=1,1&sound_ids=87
```

MAP

```
GET
http://example.com/api/1/admin/fv?
key=xxxx&output_type=x&duration=60&row=1&col=2&
map_data=[[ { "id":"59", "lev":"1", "rx":"0", "ry":"0" },
{ "id":"58", "lev":"1", "rx":"0", "ry":"0" } ]]&sound_ids=87
```

Sample Curl Command

```
curl --basic -u {your account}:{your password} "https://example.com/api/1/admin/fv?duration=60&row=1&col=2&tile_quality=2&media_ids=58
```

Response Example - JSON


```
{
   "status":"success",
   "video_url":"http:\/\/example.com\/fv\/gnzo-xxxxxxxxxx.mp4",
   "thumbnail_url":"http:\/\/example.com\/fvt\/gnzo-xxxxxxxxxxxxx.png",
   "cached_data":"1",
   "black_edge":"0",
   "map_data":[
      [
         {
            "id":"59",
            "lev":"1",
            "all-lev":[
              "1",
              "2"
            ],
            "audio":"1",
            "rx":"0",
```

```
            "ry":"0",
            "caption":"test caption",
            "description":"test description"
          },
          {
            "id":"58",
            "lev":"1",
            "all-lev":[
              "1",
              "2"
            ],
            "audio":"1",
            "rx":"0",
            "ry":"0",
            "caption":"",
            "description":""
          }
        ]
    ]
}
```


Response Example - XML


```
<?xml version="1.0" encoding="UTF-8"?>
<response>
<status>success</status>
<video_url>http://example.com/fv/gnzo-xxxxxxxxxxxx.mp4</video_url>
<thumbnail_url>http://example.com/fvt/gnzo-xxxxxxxxxxxxxxx.png</thumbnail_url>
<cached_data>1</cached_data>
<black_edge>0</bloack_edge>
<map_data>
  <entry>
    <entry>
      <id>59</id>
      <lev>1</lev>
      <all-leve>
        <entry>1</entry>
        <entry>2</entry>
      </all-leve>
      <audio>1</audio>
      <rx>0</rx>
      <ry>0</ry>
      <caption>test caption</caption>
      <description>test description</description>
    </entry>
    <entry>
      <id>58</id>
      <lev>1</lev>
      <all-leve>
        <entry>1</entry>
        <entry>2</entry>
      </all-leve>
      <audio>1</audio>
      <rx>0</rx>
      <ry>0</ry>
      <caption  />
      <description  />
    </entry>
  </entry>
```

```
</map_data>
</response>
```

Media management API

Acquisition of media information list

This API returns a media information list.

```
GET /api/1/videos/media
```

> **Note**
>
> This API's behavior depends on the authentication.
> If the digest authentication is passed, this API returns a list of media that has "not deleted media".
> If the Referrer + Kye authentication is passed, this API returns a list of media that has "activated media".

Parameters

- **output_type**
  - Response type
    - x: Output in XML format
    - blank: Output in JSON format
- **page** [required]
  - Page which will be rendered. Integer greater than 1.[Appendix]
- **num** [required]
  - Number per page [Appendix]
    Ascending order if num is positive value
    Descending order if num is negative value
- **tile_type**
  - Tile type to get medias
    All tile type is applied if tile_type is not set.
    hd 16:9 / sd 4:3 / r 3:2 / s 1:1
- **tile_quality**
  - Tile quality to get medias
    All tile quality is applied if tile_type is not set.
    There are three optional tile quality. 1 / 2 / 3
- **level**
  - Level to get medias
    All level is applied if level is not set.
    There are four optional level 1 / 2 / 3 / 4.
- **media_id**
  - Media id to get medias (Partial match)
    Medias for the specified integer number containing media ids will be target.

Response

- **status**
  - Result of process
    success: success
    other: failure
- **has_next**
  - existence on the next page
    true: exist
    false: not exist

- **total**
  - total media number
- **info**
  - media information
    - **id**
      - media id
    - **caption**
      - caption
    - **description**
      - description
    - **tags**
      - tag
    - **thumbnail_url**
      - URL for fv thumbnail that is generated
      - File name of the thumbnail will be “(MediaID)_(length of thumbnail)_(frame_position).jpg”. Video frame rate is fixed automatically to 30fps when you register video, so the frame position of video you have and the frame position of media may be different.
    - **sound_url**
      - URL for fv sound that is generated
    - **tile_info**
      - tile information
        - **version**
          - version of fv encoder
        - **type**
          - tile type
        - **fps**
          - fps
        - **frames**
          - total frame of tile
        - **level**
          - tile level (array style)
        - **tile_quality**
          - tile quality (array style)
        - **create_date**
          - creation date of tile
        - **update_date**
          - updated date of tile
        - **status**
          - tile status
    - **create_date**
      - creation date of media
    - **update_date**
      - updated date of media
    - **status**
      - media status
        0: invalid
        1: valid

Request Example

```
GET
https://example.com/api/1/videos/media?page=30&num=2
```

Sample Curl Command

- **Basic認証を使用した方法**

```
curl --basic -u {your account}:{your password} "https://example.com/api/1/videos/media?page=1&num=-10"
```

- **AccessToken認証を使用した方法**

```
curl "https://example.com/api/1/videos/media?page=1&num=-10" \
-H "Authorization: Bearer {your access_token}"
```

Response Example - JSON


```
{
  "status":"success",
  "has_next":true,
  "total":59,
  "info":[
    {
      "id":"2",
      "caption":"test caption",
      "tags":[
      "tag1",
      "tag2"
      ],
      "thumbnail_url":[
        "http:\/\/example.com\/r\/p\/images\/tt\/2_144_0.jpg"
      ],
      "sound_url":[
        "http:\/\/example.com\/r\/p\/audios\/ta\/2_1.m4a"
      ],
      "tile_info":{
        "version":"3",
        "type":"hd144",
        "fps":30,
        "frames":420,
        "level":[
          1
        ],
        "tile_quality":[
          1,
          2
        ],
        "create_date":"2013-01-11 12:15:41",
        "update_date":"2013-01-11 12:15:44",
        "status":1
      },
      "create_date":"2013-01-11 12:15:41",
      "update_date":"2013-01-30 13:49:26",
      "status":"1"
      },
      {
        "id":"3",
        "caption":"caption test",
```

```
            "tags":[
            "test tag1",
            "test tag2"
            ],
            "thumbnail_url":[
            "http:\/\/example.com\/r\/p\/images\/tt\/3_144_0.jpg"
            ],
            "sound_url":[
            "http:\/\/example.com\/r\/p\/audios\/ta\/3_1.m4a"
            ],
            "tile_info":{
            "version":"3",
            "type":"hd144",
            "fps":30,
            "frames":360,
            "level":[
              1
            ],
            "tile_quality":[
             2
            ],
            "create_date":"2013-01-11 12:16:12",
            "update_date":"2013-01-11 12:16:16",
            "status":1
          },
          "create_date":"2013-01-11 12:16:12",
          "update_date":"2013-01-11 12:16:15",
          "status":"1"
        }
    ]
}
```


Response Example - XML


```
<?xml version="1.0" encoding="UTF-8"?>
<response>
<status>success</status>
<has_next>1</has_next>
<total>59</total>
<info>
  <entry>
    <id>2</id>
    <caption>test caption</caption>
    <tags>
      <entry>tag1</entry>
      <entry>tag2</entry>
    </tags>
    <thumbnail_url>
      <entry>http://example.com/r/p/images/tt/2_144_0.jpg</entry>
    </thumbnail_url>
    <sound_url>
      <entry>http://example.com/r/p/audios/ta/2_1.m4a</entry>
    </sound_url>
    <tile_info>
      <version>3</version>
      <type>hd144</type>
      <fps>30</fps>
      <frames>420</frames>
      <level>
```

```
          <entry>1</entry>
        </level>
        <tile_quality>
          <entry>1</entry>
          <entry>2</entry>
        </tile_quality>
        <create_date>2013-01-11 12:15:41</create_date>
        <update_date>2013-01-11 12:15:44</update_date>
        <status>1</status>
      </tile_info>
      <create_date>2013-01-11 12:15:41</create_date>
      <update_date>2013-01-30 13:49:26</update_date>
      <status>1</status>
    </entry>
    <entry>
      <id>3</id>
      <caption>test caption</caption>
      <tags>
        <entry>test tag1</entry>
        <entry>test tag2</entry>
      </tags>
      <thumbnail_url>
        <entry>http://example.com/r/p/images/tt/3_144_0.jpg</entry>
      </thumbnail_url>
      <sound_url>
        <entry>http://example.com/r/p/audios/ta/3_1.m4a</entry>
      </sound_url>
      <tile_info>
        <version>3</version>
        <type>hd144</type>
        <fps>30</fps>
        <frames>360</frames>
        <level>
          <entry>1</entry>
        </level>
        <tile_quality>
          <entry>1</entry>
        </tile_quality>
        <create_date>2013-01-11 12:16:12</create_date>
        <update_date>2013-01-11 12:16:16</update_date>
        <status>1</status>
      </tile_info>
      <create_date>2013-01-11 12:16:12</create_date>
      <update_date>2013-01-11 12:16:15</update_date>
      <status>1</status>
    </entry>
  </info>
  </response>
```

Acquisition of media metadata

This API returns a media information

```
GET /api/1/videos/{media_id}/info
```

Parameters

- **output_type**
  - Response type

- x: Output in XML format
- blank: Output in JSON format

Response

- **status**
  - Result of process
    success: success
    other: failure
- **info**
  - media information
    - **id**
      - media id
    - **caption**
      - caption
    - **description**
      - description
    - **tags**
      - tag
    - **thumbnail_url**
      - fv thumbnail URL
    - **sound_url**
      - fv sound file URL
    - **tile_info**
      - tile information
        - **version**
          - version of fv encoder
        - **type**
          - tile type
        - **fps**
          - fps
        - **frames**
          - total frame of tile
        - **level**
          - tile level
        - **tile_quality**
          - tile quality
        - **create_date**
          - creation date of tile
        - **update_date**
          - updated date of tile
        - **status**
          - tile status
    - **create_date**
      - creation date of media
    - **update_date**
      - updated date of media
    - **status**
      - media status

0: invalid
1: valid

Request Example

```
GET
https://example.com/api/1/videos/22/info
```

Response Example - JSON


```
{
    "status":"success",
    "info":{
        "id":"22",
        "caption":"test caption",
        "description":"test description",
        "tags":[
            "a"
        ],
        "thumbnail_url":[
            "http:\/\/example.com\/r\/p\/images\/tt\/22_144_0.jpg"
        ],
        "sound_url":[
            "http:\/\/example.com\/r\/p\/audios\/ta\/22_1.m4a"
        ],
        "tile_info":{
            "version":"3",
            "type":"hd144",
            "fps":30,
            "frames":600,
            "level":[
              1
            ],
            "tile_quality":[
              1
            ],
            "create_date":"2013-01-21 13:51:35",
            "update_date":"2013-01-21 13:51:50",
            "status":1
        },
        "create_date":"2013-01-21 13:51:10",
        "update_date":"2013-02-01 14:15:08",
        "status":"1"
    }
}
```


Response Example - XML


```
<?xml version="1.0" encoding="UTF-8"?>
<response>
<status>success</status>
<info>
    <id>22</id>
    <caption>test caption</caption>
    <description>test description</description>
    <tags>
        <entry>a</entry>
    </tags>
```

```
    <thumbnail_url>
      <entry>http://example.com/r/p/images/tt/22_144_0.jpg</entry>
    </thumbnail_url>
    <sound_url>
      <entry>http://example.com/r/p/audios/ta/22_1.m4a</entry>
    </sound_url>
    <tile_info>
      <version>3</version>
      <type>hd144</type>
      <fps>30</fps>
      <frames>600</frames>
      <level>
        <entry>1</entry>
      </level>
      <tile_quality>
        <entry>1</entry>
      </tile_quality>
      <create_date>2013-01-21 13:51:35</create_date>
      <update_date>2013-01-21 13:51:50</update_date>
      <status>1</status>
    </tile_info>
    <create_date>2013-01-21 13:51:10</create_date>
    <update_date>2013-02-01 14:15:08</update_date>
    <status>1</status>
  </info>
</response>
```

複数メディアのメタデータの取得

複数のメディアのメタデータを取得します。

```
GET /api/1/videos/info
```

Parameters

- **output_type**
  - Response 形式
    - x : XML 形式で出力
    - 指定無し : JSON 形式で出力
- **media_ids** [必須]
  - 対象となるメディアID列をカンマ区切りで指定して下さい。
    対象メディアIDが一つの時でも対応しております。
    ID間に空白等をいれないで下さい。メディア指定フォーマットエラーとなります。

Response

- **status**
  - 処理結果
    success : 成功
    それ以外 : 失敗
- **info**
  - メディア情報
    - **id**
      - メディアID
    - **caption**
      - キャプション
    - **description**

    - ディスクリプション
- **tags**
    - タグ
- **thumbnail_url**
    - サムネイルのURL
- **sound_url**
    - 音声データへのURL
- **tile_info**
    - タイル情報
        - **version**
            - fvエンコーダのバージョン
        - **type**
            - タイル種別
        - **fps**
            - fps
        - **frames**
            - 総フレーム数
        - **level**
            - タイルレベル
        - **tile_quality**
            - タイル品質
        - **create_date**
            - 作成日時
        - **update_date**
            - 更新日時
        - **status**
            - タイルのステータス
- **create_date**
    - 作成日時
- **update_date**
    - 更新日時
- **status**
    - メディアのステータス
      0 : 無効
      1 : 有効

Request Example

```
GET
https://example.com/api/1/videos/info?media_ids=386,387
```

Sample Curl Command

- **Basic認証を使用した方法**

```
curl --basic -u {your account}:{your password} \
"https://example.com/api/1/videos/info?media_ids=386,387" \
-X GET
```

- **AccessToken認証を使用した方法**

```
curl "https://example.com/api/1/videos/info?media_ids=386,387" \
-H "Authorization: Bearer {your access_token}" \
-X GET
```

Response Example - JSON


```
{
  "status":"success",
  "info":[
      {
        "id":"386",
        "caption":"test caption",
        "description":"test description",
        "tags":[
          "water",
          "travel",
          "no_person"
        ],
        "thumbnail_url":[
          "http:\/\/example.com\/r\/p\/images\/tt\/386_144_0.jpg",
          "http:\/\/example.com\/r\/p\/images\/tt\/386_288_0.jpg"
        ],
        "sound_url":[
          "http:\/\/example.com\/r\/p\/audios\/ta\/386_1.m4a"
        ],
        "tile_info":{
          "version":"3",
          "type":"hd144",
          "fps":30,
          "frames":600,
          "level":[
           1,
           2
          ],
          "tile_quality":[
           2
          ],
          "create_date":"2016-08-04 10:57:53",
          "update_date":"2016-08-04 10:58:04",
          "status":1
        },
        "create_date":"2016-08-04 10:57:48",
        "update_date":"2016-08-04 10:57:48",
        "status":"1"
      },
      {
        "id":"387",
        "caption":"test caption",
        "description":"test description",
        "tags":[
          "water",
          "travel",
          "no_person"
        ],
        "thumbnail_url":[
          "http:\/\/example.com\/r\/p\/images\/tt\/387_144_0.jpg",
          "http:\/\/example.com\/r\/p\/images\/tt\/387_288_0.jpg"
        ],
        "sound_url":[
          "http:\/\/example.com\/r\/p\/audios\/ta\/387_1.m4a"
        ],
```

```
        "tile_info":{
          "version":"3",
          "type":"hd144",
          "fps":30,
          "frames":600,
          "level":[
           1,
           2
          ],
          "tile_quality":[
           2
          ],
          "create_date":"2016-08-04 10:57:53",
          "update_date":"2016-08-04 10:58:04",
          "status":1
        },
        "create_date":"2016-08-04 10:57:48",
        "update_date":"2016-08-04 10:57:48",
        "status":"1"
      }
}
```

Response Example - XML

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
<status>success</status>
<info>
    <entry>
        <id>387</id>
        <caption />
        <description />
        <tags>
            <entry>water</entry>
            <entry>travel</entry>
            <entry>no_person</entry>
        </tags>
        <thumbnail_url>
            <entry>http://example.com/r/p/images/tt/387_144_0.jpg</entry>
            <entry>http://example.com/r/p/images/tt/387_288_0.jpg</entry>
        </thumbnail_url>
        <sound_url />
        <tile_info>
            <version>3</version>
            <type>hd144</type>
            <fps>30</fps>
            <frames>270</frames>
            <level>
                <entry>1</entry>
                <entry>2</entry>
            </level>
            <tile_quality>
                <entry>2</entry>
            </tile_quality>
            <create_date>2016-08-04 10:57:53</create_date>
            <update_date>2016-08-04 10:58:04</update_date>
            <status>1</status>
        </tile_info>
        <create_date>2016-08-04 10:57:48</create_date>
        <update_date>2016-08-04 10:57:48</update_date>
```

```
            <status>0</status>
        </entry>
        <entry>
            <id>386</id>
            <caption />
            <description />
            <tags>
                <entry>water</entry>
                <entry>travel</entry>
                <entry>no_person</entry>
            </tags>
            <thumbnail_url>
                <entry>http://example.com/r/p/images/tt/386_144_0.jpg</entry>
                <entry>http://example.com/r/p/images/tt/386_288_0.jpg</entry>
            </thumbnail_url>
            <sound_url />
            <tile_info>
                <version>3</version>
                <type>hd144</type>
                <fps>30</fps>
                <frames>270</frames>
                <level>
                    <entry>1</entry>
                    <entry>2</entry>
                </level>
                <tile_quality>
                    <entry>2</entry>
                </tile_quality>
                <create_date>2016-08-04 10:57:28</create_date>
                <update_date>2016-08-04 10:57:40</update_date>
                <status>1</status>
            </tile_info>
            <create_date>2016-08-04 10:57:24</create_date>
            <update_date>2016-08-04 10:57:24</update_date>
            <status>0</status>
        </entry>
    </info>
    </response>
```

Update of media metadata

This API updates the metadata of the media.

```
POST /api/1/videos/{media_id}/info
```

Parametres

- **output_type**
  - Response Type
    - x: Output in XML format
    - blank: Output in JSON format
- **caption**
  - caption
- **description**
  - description
- **tags**
  - tag

Response

- **status**
  - Result of Process
    success: sucess
    other: failure
- **caption**
  - updated caption
- **description**
  - updated description
- **tags**
  - updated tags

Response Example - JSON

```
{
   "status":"success",
   "caption":"test caption",
   "description":"description test",
   "tags":[
      "tag test1",
      "tag test2",
   ]
}
```

Response Example - XML

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
<status>success</status>
<caption>test caption</caption>
<description>description test</description>
<tags>
   <entry>tag test1</entry>
   <entry>tag test2</entry>
</tags>
</response>
```

Media activation

This API activates the media.
The media which is activated by this API will appear in fv.

```
POST /api/1/videos/{media_id}/activate
```

Parametres

- **output_type**
  - Response Type
    - x: Output in XML format
    - blank: Output in JSON format

Response

- **status**
  - Result of Process

success: sucess
other: failure

Response Example - JSON

```
{
  "status": "success"
}
```

Response Example - XML

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
  <status>success</status>
</response>
```

複数メディアの有効化

複数のメディアを有効にします。
本 API によって有効化したメディアは、fv 生成時の対象になり、fv に表示されるようになります。

```
POST /api/1/videos/activate
```

Parametres

- **output_type**
  - Response 形式
    - x : XML 形式で出力
    - 指定無し : JSON 形式で出力
- **media_ids** [必須]
  - 対象となるメディアID列をカンマ区切りで指定して下さい.
    対象メディアIDが一つの時でも対応しております。
    ID間に空白等をいれないで下さい. メディア指定フォーマットエラーとなります。

Response

- **status**
  - 処理結果
    success : 成功
    それ以外 : 失敗

Sample Curl Command

- **Basic認証を使用した方法**

```
curl --basic -u {your account}:{your password} \
-d "media_ids=386,387" \
"https://example.com/api/1/videos/activate" \
-X POST
```

- **AccessToken認証を使用した方法**

```
curl "https://example.com/api/1/videos/activate" \
-H "Authorization: Bearer {your access_token}" \
-d "media_ids=386,387" \
-X POST
```

Response Example - JSON

```
{
  "status": "success"
}
```

Response Example - XML

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
  <status>success</status>
</response>
```

Media deactivation

This API deactivate the media.
The media which is deactivated by this API will not display in fv.

```
POST /api/1/videos/{media_id}/deactivate
```

Parametres

- **output_type**
  - Response Type
    - x: Output in XML format
    - blank: Output in JSON format

Response

- **status**
  - Result of Process
    success: sucess
    other: failure

Response Example - JSON

```
{
  "status": "success"
}
```

Response Example - XML

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
  <status>success</status>
</response>
```

複数メディアの無効化

複数のメディアを無効にします。
本 API によって無効化したメディアは、fv の生成時の対象に含まれなくなり、fv に表示されなくなります。

```
POST /api/1/videos/deactivate
```

Parametres

- **output_type**
  - Response 形式
    - x : XML 形式で出力
    - 指定無し : JSON 形式で出力
- **media_ids** [必須]
  - 対象となるメディアID列をカンマ区切りで指定して下さい.
    対象メディアIDが一つの時でも対応しております。
    ID間に空白等をいれないで下さい. メディア指定フォーマットエラーとなります。

Response

- **status**
  - 処理結果
    success : 成功
    それ以外 : 失敗

Sample Curl Command

- **Basic認証を使用した方法**

```
curl --basic -u {your account}:{your password} \
-d "media_ids=386,387" \
"https://example.com/api/1/videos/deactivate" \
-X POST
```

- **AccessToken認証を使用した方法**

```
curl "https://example.com/api/1/videos/deactivate" \
-H "Authorization: Bearer {your access_token}" \
-d "media_ids=386,387" \
-X POST
```

Response Example - JSON

```
{
  "status": "success"
}
```

Response Example - XML

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
  <status>success</status>
</response>
```

Media deletion

This API delete the media physically.
The media file(tile) will be deleted physically from API server by this API.

```
DELETE /api/1/videos/{media_id}
```

Parametres

- **output_type**
  - Response Type

- x: Output in XML format
- blank: Output in JSON format

Response

- **status**
  - Result of Process
    success: sucess
    other: failure

Response Example - JSON

```
{
  "status":  "success"
}
```

Response Example - XML

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
  <status>success</status>
</response>
```

複数メディアの削除

複数のメディアを削除します。
本 API によりサーバ上からメディアファイル（タイル）を物理的に削除します。
またメディアがエンコード中であった場合、エンコードをキャンセルし、生成中のファイルを削除致します。
HTTP MethodはDELETEではなく、POSTであることに注意して下さい。

```
POST /api/1/videos/delete
```

Parametres

- **output_type**
  - Response 形式
    - x : XML 形式で出力
    - 指定無し : JSON 形式で出力
- **media_ids** [必須]
  - 対象となるメディアID列をカンマ区切りで指定して下さい。
    対象メディアIDが一つの時でも対応しております。
    ID間に空白等をいれないで下さい。メディア指定フォーマットエラーとなります。

Response

- **status**
  - API要求自体の処理結果
    success : 成功
    それ以外 : 失敗
- **result**
  - 各々メディアの削除結果
    対象のメディアID
    - 各々の処理結果
      success : 成功
      それ以外 : 失敗

Sample Curl Command

- **Basic認証を使用した方法**

```
curl --basic -u {your account}:{your password} \
-d "media_ids=386,387" "https://example.com/api/1/videos/delete" \
-X POST
```

- **AccessToken認証を使用した方法**

```
curl "https://example.com/api/1/videos/delete" \
-H "Authorization: Bearer {your access_token}" \
-d "media_ids=386,387" \
-X POST
```

Response Example - JSON

```
{
  "status": "success",
  "result":
  {
    "386": "success",
    "387": "success"
  }
}
```

Response Example - XML

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
  <status>success</status>
  <result>
      <entry>success</entry>
      <entry>success</entry>
  </result>
</response>
```

Acquisition of tile generation status

This API returns tile creation status.

```
GET /api/1/videos/{media_id}/tile-status
```

Parametres

- **output_type**
  - Response Type
    - x: Output in XML format
    - blank: Output in JSON format

Response

- **status**
  - Result of Process
    success: creation completion
- **tile_status**
  - tile status
    1: tile creation is success

3: tile is not created yet
4: tile creation is failed

Request Example

```
GET
https//example.com/api/1/videos/153/tile-status
```

Response Example - JSON

```
{
   "status":  "success",
   "tile_status":1
}
```

Response Example - XML

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
   <status>success</status>
   <tile_status>1</tile_status>
</response>
```

Acquisition of tile information list

This API returns a tile information list.

```
GET /api/1/videos/tile
```

Parameters

- **output_type**
  - Response type
    - x: Output in XML format
    - blank: Output in JSON format
- **page** [required]
  - Page which will be rendered. Integer greater than 1. [Appendix]
- **num** [required]
  - Number per page [Appendix]
- **tile_type**
  - Tile type to get tile information
    All tile type is applied if tile_type is not set.
    hd 16:9 / sd 4:3 / r 3:2 / s 1:1
- **level**
  - Level to get tile information
    All level(1-4) is applied if level is not set.
    1 / 2 / 3 / 4
- **tile_quality**
  - Tile Quality to get tile information
    All tile_quality(1~3) are applied if tile_quality is not set.
    1 / 2 / 3

Response

- **status**

- result of process
  success: success
  other: failure

- **has_next**
  - existence on the next page
    true : exist
    false : not exist

- **total**
  - total media number

- **info**
  - tile information
    - **media_id**
      - media id
    - **version**
      - version of fv encoder
    - **type**
      - tile type
    - **fps**
      - fps
    - **frames**
      - total frame of tile
    - **level**
      - tile level
    - **tile_quality**
      - Tile Quality
    - **create_date**
      - creation date of tile
    - **update_date**
      - updated date of tile
    - **status**
      - tile status
        0 : invalid
        1 : tile is created
        3 : tile is not created yet
        4 : tile creation is failed

Request Example

```
GET
https://example.com/api/1/videos/tile?page=1&num=3
```

Response Example - JSON

```
{
"status":"success",
"has_next":true,
"total":58,
"info":[
            {
               "media_id":10,
               "version":"3",
               "type":"s64",
```

```
            "fps":30,
            "frames":150,
            "level":1,
            "tile_quality":2,
            "create_date":"2012-10-01 16:57:09",
            "update_date":"2012-10-01 16:57:14",
            "status":1
        },
        {
            "media_id":11,
            "version":"3",
            "type":"s64",
            "fps":30,
            "frames":150,
            "level":2,
            "tile_quality":2,
            "create_date":"2012-10-01 16:57:09",
            "update_date":"2012-10-01 16:57:14",
            "status":1
        },
        {
            "media_id":12,
            "version":"3",
            "type":"s64",
            "fps":30,
            "frames":150,
            "level":3,
            "tile_quality":2,
            "create_date":"2012-10-01 16:57:09",
            "update_date":"2012-10-01 16:57:14",
            "status":1
        }
    ]
}
```


Response Example - XML


```
<?xml version="1.0" encoding="UTF-8"?>
<response>
  <status>success</status>
  <has_next>1</has_next>
  <total>58</total>
  <info>
    <entry>
      <media_id>10</media_id>
      <version>3</version>
      <type>s64</type>
      <fps>30</fps>
      <frames>150</frames>
      <level>1</level>
      <tile_quality>2</tile_quality>
      <create_date>2012-10-01 16:57:09</create_date>
      <update_date>2012-10-01 16:57:14</update_date>
      <status>1</status>
    </entry>
    <entry>
      <media_id>11</media_id>
      <version>3</version>
      <type>s64</type>
```

```
        <fps>30</fps>
        <frames>150</frames>
        <level>2</level>
        <tile_quality>1</tile_quality>
        <create_date>2012-10-01 16:57:09</create_date>
        <update_date>2012-10-01 16:57:14</update_date>
        <status>1</status>
      </entry>
      <entry>
        <media_id>12</media_id>
        <version>3</version>
        <type>s64</type>
        <fps>30</fps>
        <frames>150</frames>
        <level>3</level>
        <tile_quality>3</tile_quality>
        <create_date>2012-10-01 16:57:09</create_date>
        <update_date>2012-10-01 16:57:14</update_date>
        <status>1</status>
      </entry>
    </info>
</response>
```

Acquisition of keyword search result

This API returns keywords search result.
Tag will be searched with perfect matching and caption will be searched with partial match.

```
GET /api/1/videos/search
```

> **Note**
>
> This API's behavior depends on the authentication.
> If the digest authentication is passed, this API returns a list of keyword result that has "not deleted media".
> If the Referrer + Key authentication is passed, this API returns a list of keyword result that has "activated media".

Parameters

- **output_type**
  - Response type
    - x: Output in XML format
    - blank: Output in JSON format
- **page** [required]
  - Page which will be rendered. Integer greater than 1.[Appendix]
- **num** [required]
  - Number per page [Appendix]
- **keywords** [required]
  - search keyword
    #(characters): tag search (Perfect match)
    (characters): caption and description search (Partial match)
    &(integer): (Partial match)

Response

- **status**
  - Result of process
    success: success
    other: failure

- **has_next**
  - existence on the next page
    true: exist
    false: not exist
- **total**
  - total media number
- **info**
  - media information
    - **id**
      - media id
    - **caption**
      - caption
    - **description**
      - description
    - **tags**
      - tag
    - **thumbnail_url**
      - fv thumbnail URL
    - **sound_url**
      - fv sound URL
    - **tile_info**
      - tile information
        - **version**
          - version of fv encoder
        - **type**
          - tile type
        - **fps**
          - fps
        - **frames**
          - total frame of tile
        - **level**
          - tile level
        - **tile_quality**
          - tile quality
        - **create_date**
          - creation date of tile
        - **update_date**
          - updated date of tile
        - **status**
          - tile status
    - **create_date**
      - creation date of media
    - **update_date**
      - updated date of media
    - **status**
      - media status
        0: invalid
        1: valid

Request Example

```
GET
https://example.com/api/1/videos/search?page=1&num=2&keywords=abc
```

Response Example - JSON


```
{
  "status":"success",
  "has_next":true,
  "total":8,
  "info":[
    {
      "id":17,
      "caption":"test caption",
      "tags":[
        "tag",
        "abc",
        "test"
      ],
      "thumbnail_url":[
        "http:\/\/example.com\/r\/p\/images\/tt\/17_144_0.jpg",
        "http:\/\/example.com\/r\/p\/images\/tt\/17_288_0.jpg",
        "http:\/\/example.com\/r\/p\/images\/tt\/17_432_0.jpg",
        "http:\/\/example.com\/r\/p\/images\/tt\/17_576_0.jpg"
      ],
      "sound_url":[
        "http:\/\/example.com\/r\/p\/audios\/ta\/17_1.m4a"
      ],
      "tile_info":{
        "version":"3",
        "type":"hd144",
        "fps":30,
        "frames":4500,
        "level":[
          1,
          2,
          3,
          4
        ],
        "tile_quality":[1,2],
        "create_date":"2013-01-11 12:25:17",
        "update_date":"2013-01-11 12:38:43",
        "status":1
      },
      "create_date":"2013-01-11 12:25:17",
      "update_date":"2013-01-11 15:48:15",
      "status":1
    },
    {
      "id":16,
      "caption":"test caption",
      "tags":[
        "tag",
        "abc",
        "test"
      ],
      "thumbnail_url":[
```

```
            "http:\/\/example.com\/r\/p\/images\/tt\/16_144_0.jpg",
            "http:\/\/example.com\/r\/p\/images\/tt\/16_288_0.jpg",
            "http:\/\/example.com\/r\/p\/images\/tt\/16_432_0.jpg",
            "http:\/\/example.com\/r\/p\/images\/tt\/16_576_0.jpg"
        ],
        "sound_url":[
            "http:\/\/example.com\/r\/p\/audios\/ta\/16_1.m4a"
        ],
        "tile_info":{
            "version":"3",
            "type":"hd144",
            "fps":30,
            "frames":3090,
        "level":[
            1,
            2,
            3,
            4
        ],
        "tile_quality":[1,3],
        "create_date":"2013-01-11 12:24:53",
        "update_date":"2013-01-11 12:35:40",
        "status":1
    },
    "create_date":"2013-01-11 12:24:53",
    "update_date":"2013-01-11 12:25:01",
    "status":1
    }
  ]
}
```


Response Example - XML


```
<?xml version="1.0" encoding="UTF-8"?>
<response>
<status>success</status>
<has_next>1</has_next>
<total>8</total>
<info>
  <entry>
    <id>17</id>
    <caption>test caption</caption>
    <tags>
      <entry>tag</entry>
      <entry>abc</entry>
      <entry>test</entry>
    </tags>
    <thumbnail_url>
      <entry>http://example.com/r/p/images/tt/17_144_0.jpg</entry>
      <entry>http://example.com/r/p/images/tt/17_288_0.jpg</entry>
      <entry>http://example.com/r/p/images/tt/17_432_0.jpg</entry>
      <entry>http://example.com/r/p/images/tt/17_576_0.jpg</entry>
    </thumbnail_url>
    <sound_url>
      <entry>http://example.com/r/p/audios/ta/17_1.m4a</entry>
    </sound_url>
    <tile_info>
      <version>3</version>
      <type>hd144</type>
```

```
      <fps>30</fps>
      <frames>4500</frames>
      <level>
        <entry>1</entry>
        <entry>2</entry>
        <entry>3</entry>
        <entry>4</entry>
      </level>
      <tile_quality>
       <entry>1</entry>
       <entry>2</entry>
      </tile_quality>
      <create_date>2013-01-11 12:25:17</create_date>
      <update_date>2013-01-11 12:38:43</update_date>
      <status>1</status>
    </tile_info>
    <create_date>2013-01-11 12:25:17</create_date>
    <update_date>2013-01-11 15:48:15</update_date>
    <status>1</status>
  </entry>
  <entry>
    <id>16</id>
    <caption>test caption</caption>
    <tags>
      <entry>tag</entry>
      <entry>abc</entry>
      <entry>test</entry>
    </tags>
    <thumbnail_url>
      <entry>http://example.com/r/p/images/tt/16_144_0.jpg</entry>
      <entry>http://example.com/r/p/images/tt/16_288_0.jpg</entry>
      <entry>http://example.com/r/p/images/tt/16_432_0.jpg</entry>
      <entry>http://example.com/r/p/images/tt/16_576_0.jpg</entry>
    </thumbnail_url>
    <sound_url>
      <entry>http://example.com/r/p/audios/ta/16_1.m4a</entry>
    </sound_url>
    <tile_info>
      <version>3</version>
      <type>hd144</type>
      <fps>30</fps>
      <frames>3090</frames>
      <level>
        <entry>1</entry>
        <entry>2</entry>
        <entry>3</entry>
        <entry>4</entry>
      </level>
      <tile_quality>
        <entry>1</entry>
        <entry>3</entry>
      </tile_quality>
      <create_date>2013-01-11 12:24:53</create_date>
      <update_date>2013-01-11 12:35:40</update_date>
      <status>1</status>
    </tile_info>
    <create_date>2013-01-11 12:24:53</create_date>
    <update_date>2013-01-11 12:25:01</update_date>
    <status>1</status>
  </entry>
```

```
</info>
</response>
```

Acquisition of contract information

This API returns current contract information.

```
GET /api/1/admin/bill_plan
```

Parametres

- **output_type**
  - Response type
    - x: Output in XML format
    - blank: Output in JSON format

Response

- **status**
  - Result of process
    success: success
    other: failure
- **info**
  - contract information
    - **plan_name**
      - Plan name you have contract
    - **monthly_fee**
      - onthly payments (yen)
    - **disk_usage**
      - Disk space currently in use (Byte)
    - **max_disk_usage**
      - Maximum available disk space (Byte)
    - **upload_max_size**
      - 一度に投稿可能な最大ファイルサイズ （ Byte ）

Request Example

```
GET
https://example.com/api/1/admin/bill_plan?output_type=x
```

Sample Curl Command

- **Basic認証を使用した方法**

```
curl --basic -u {your account}:{your password} "https://example.com/api/1/admin/bill_plan"
```

- **AccessToken認証を使用した方法**

```
curl "https://example.com/api/1/admin/bill_plan" \
-H "Authorization: Bearer {your access_token}"
```

Response Example - JSON

```
{
  "status":"success",
```

```
  "info":{
    "plan_name":"Test Plan",
    "monthly_fee":250000,
    "disk_usage":3904360448,
    "max_disk_usage":8200000000,
    "upload_max_size":10000000
  }
}
```

Response Example - XML

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
<status>success</status>
<info>
  <plan_name>test plan</plan_name>
  <monthly_fee>250000</monthly_fee>
  <disk_usage>3904360448</disk_usage>
  <max_disk_usage>8200000000</max_disk_usage>
  <upload_max_size>10000000</upload_max_size>
</info>
</response>
```

Cancel of media registration

This API cancel the process of media creation.
When you run the API, all data, such as tile data that has been generated during the creation process of the target media will be deleted.

```
POST /api/1/videos/{media_id}/cancel
```

Parametres

- **output_type**
  - Response type
    - x: Output in XML format
    - blank: Output in JSON format

Response

- **status**
  - Result of process
    success: success
    other: failure

Request Example

```
POST
https://example.com/api/1/videos/249/cancel
```

Response Example - JSON

```
{
  "status":  "success"
}
```

Response Example - XML

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
  <status>success</status>
</response>
```

映像から抽出された音声タグの取得（日本語のみ対応）

本APIは指定されたパラメータにより取得出来る情報が異なります。 どのオプションパラメータも指定しない場合は、音声解析済みの登録ビデオのタグ統計が取得されます。 メディアIDを指定すると、該当のメディアに付けられた音声タグを返却致します。 JSONにより音声タグを指定して、該当音声タグが付けられたメディアIDを返却することも出来ます。

```
GET /api/1/videos/audiotag
```

Parameters

- **output_type** [オプション]
  - Response 形式
    - x : XML 形式で出力
    - 指定無し : JSON 形式で出力
- **media_ids** [オプション]
  - 本パラメータにより指定するメディアID（複数指定可）に付けられた音声タグをJSONにより返却します。
- **audio_tag** [オプション]
  - JSONにより特定の音声タグを指定し、該当するメディアIDを返却します。

Response

- **status**
  - 処理結果
    success : 成功
    それ以外 : 失敗
- **media_ids**
  - 該当するメディアID（配列）
- **statistics**
  - 音声タグの統計情報
    - **allnumtags**
      - 全音声タグ個数
    - **audio_tags**
      - 音声タグ情報（配列）
        - **audio_tag**
          - 音声タグ
        - **numtimes**
          - 該当音声タグが付けられているビデオの数
- **audio_tag_information**
  - 音声タグとメディアID組（配列）
    - **media_id**
      - メディアID
    - **audio_tag**
      - 該当メディアIDに付けられた音声タグ

Request Example 1

```
GET
https://example.com/api/1/videos/audiotag
```

Sample Curl Command 1

- **Basic認証を使用した方法**

```
curl --basic -u {your account}:{your password} "https://example.com/api/1/videos/audiotag"
```

- **AccessToken認証を使用した方法(media_idsとlevelsで指定)**

```
curl "https://example.com/api/1/videos/audiotag" \
-H "Authorization: Bearer {your access_token}"
```

Response Example 1 - JSON

```
{
"status":"success",
"statistics":
    {
    "allnumtags":9,
    "audio_tags":[
        {"audio_tag":"奥さん", "numtimes":1},
        {"audio_tag":"おにぎり","numtimes":1},
        {"audio_tag":"おかげ","numtimes":1},
        {"audio_tag":"おじいちゃん","numtimes":1},
        {"audio_tag":"92年間","numtimes":1},
        {"audio_tag":"幼稚園","numtimes":1},
        {"audio_tag":"90歳","numtimes":1},
        {"audio_tag":"米","numtimes":1},
        {"audio_tag":"野菜","numtimes":1}
        ]
    }
}
```

Response Example - XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<response>
<status>success</status>
<statistics>
  <allnumtags>9</allnumtags>
  <audio_tags>
    <audio_tag>奥さん</audio_tag>
    <numtimes>1</numtimes>
  </audio_tags>
  <audio_tags>
    <audio_tag>おにぎり</audio_tag>
    <numtimes>1</numtimes>
  </audio_tags>
  <audio_tags>
    <audio_tag>おかげ</audio_tag>
    <numtimes>1</numtimes>
  </audio_tags>
  <audio_tags>
    <audio_tag>おじいちゃん</audio_tag>
    <numtimes>1</numtimes>
  </audio_tags>
  <audio_tags>
    <audio_tag>92年間</audio_tag>
    <numtimes>1</numtimes>
```

```
    </audio_tags>
    <audio_tags>
      <audio_tag>幼稚園</audio_tag>
      <numtimes>1</numtimes>
    </audio_tags>
    <audio_tags>
      <audio_tag>90歳</audio_tag>
      <numtimes>1</numtimes>
    </audio_tags>
    <audio_tags>
      <audio_tag>米</audio_tag>
      <numtimes>1</numtimes>
    </audio_tags>
    <audio_tags>
      <audio_tag>野菜</audio_tag>
      <numtimes>1</numtimes>
    </audio_tags>
  </statistics>
  </response>
```

Request Example 2

```
GET
https://example.com/api/1/videos/audiotag?media_ids=580,581
```

Sample Curl Command 2

- **Basic認証を使用した方法**

```
curl --basic -u {your account}:{your password} "https://example.com/api/1/videos/audiotag?media_ids=580,581"
```

Response Example 2 - JSON

```
{
"status":"success",
"audio_tag_information":[
   {"media_id":580,
   "audio_tag":[
         "奥さん","おにぎり","おかげ"
        ]
   },
   {"media_id":581,
   "audio_tag":[
      "おじいちゃん","92年間","幼稚園","90歳","米","野菜"
     ]
   }
   ]
}
```

Request Example 3

```
GET
https://example.com/api/1/videos/audiotag?audio_tag={"tag":"世界"}
```

Sample Curl Command 3

- **Basic認証を使用した方法**

```
curl --basic -u {your account}:{your password} "https://example.com/api/1/videos/audiotag?audio_tag=\{\"tag\":\"世界\"\}" -X GET
```

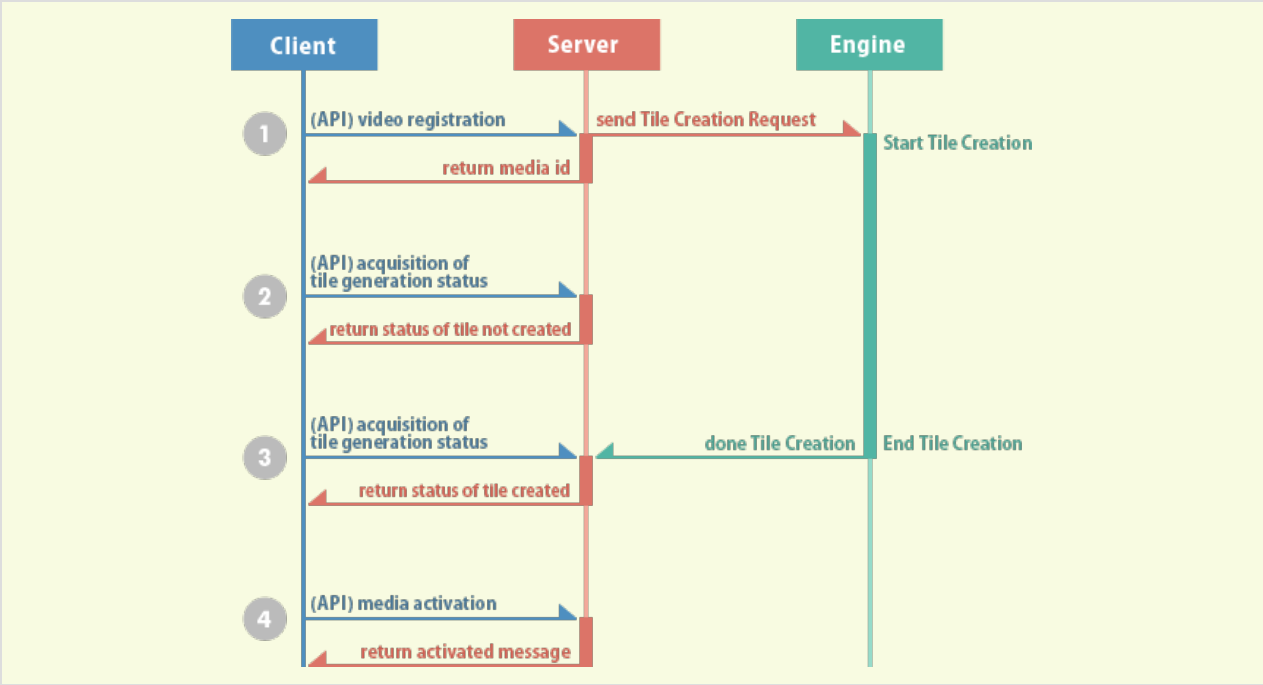
Response Example 3 - JSON

```
{
"status":"success",
"media_ids":[
   "570","571"
   ]
}
```

Appendix

Procedure from fv registration to fv acquisition

Procedure from fv registration to fv acquisition



The tile is inactive at the stage of video registration, so the tile is not available to use for fv.
The tile will be ready to use in fv after running media activation using proper API .(4 in above figure)

About uploaded video size and output fv size on video registration phase

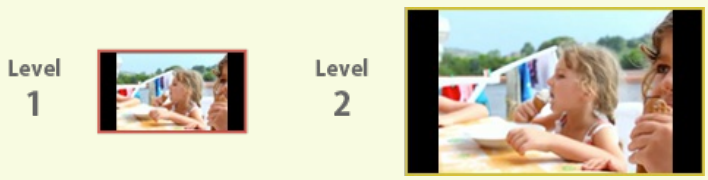
In the video registration phase, specifications of video size and output fv video size is as below. The media's aspect ratio will be same and the media will not be clipped after conversion.

If the uploaded video size is smaller than output fv size,

If the size of the input video is smaller than the output size, the difference of size will be filled with black color.

If uploaded video size is larger than otuput fv size,

If the size of the input video is larger than the output size, input video is cut to output video size and the difference of size with black color.

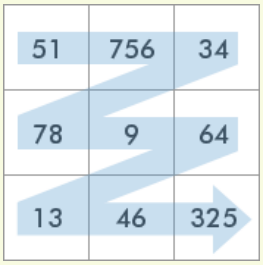


The way to create a fv with media ID and level

The way to create a fv with comma-separated media ID and level.

If you specify comma-separated media ids, each media will appear side by side in sequence from upper left to lower right.

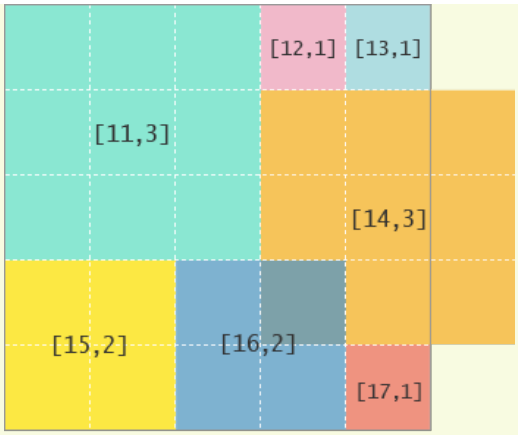
```
media_id: 51,756,34,78,9,64,13,46,325
```



Example of the case where various levels are specified.

```
media_id: 11,12,13,14,15,16,17,18,19
level   :  3, 1, 1, 3, 2, 2, 1, 1, 1
```

As shown in the below figure, fv is filled by some tiles in order from the upper left to bottom right, such as fitting the tile level of the specified media.

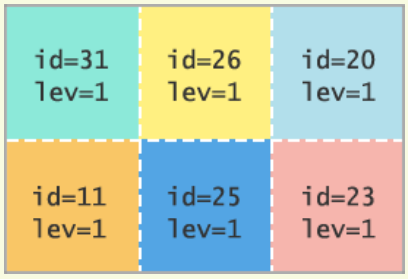


Level1 5×5 matrix
[media_id , level]

- The medias (ID=18,19) are not used in this fv, because it already is exceeding amount of fv size.
- The part of media (id14) which run over fv size is not included in fv.
- If there is overlap part like media id 14 and 16 in the above figure, the media that is specified later will be on.
- If the specified media is not enough, so if the tile is lacking for fv size, black colored tile (media id = 0) is embedded into fv. However you can not purposely specify tile which have media id=0.

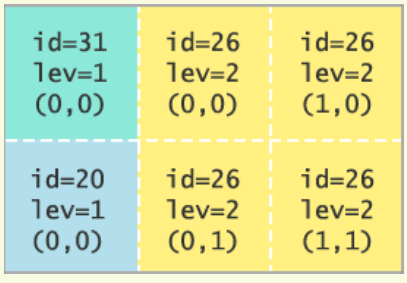
The way to create a fv with map of JSON format

[Example 1] Map data in JSON format : Same levels case




```
map_data=
[
  [
    { "id":"31", "lev":"1", "rx":"0", "ry":"0" },
    { "id":"26", "lev":"1", "rx":"0", "ry":"0" },
    { "id":"20", "lev":"1", "rx":"0", "ry":"0" }
  ],
  [
    { "id":"11", "lev":"1", "rx":"0", "ry":"0" },
    { "id":"25", "lev":"1", "rx":"0", "ry":"0" },
    { "id":"23", "lev":"1", "rx":"0", "ry":"0" }
  ]
]
```

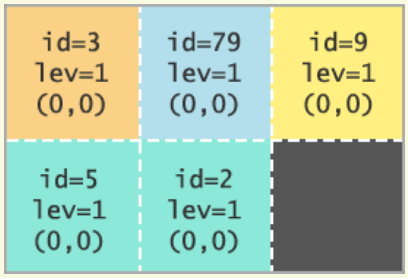
[Example 2] Map data in JSON format : Different levels case



```
map_data=
[
  [
    { "id":"31", "lev":"1", "rx":"0", "ry":"0" },
    { "id":"26", "lev":"2", "rx":"0", "ry":"0" },
    { "id":"26", "lev":"2", "rx":"1", "ry":"0" }
  ],
  [
    { "id":"20", "lev":"1", "rx":"0", "ry":"0" },
    { "id":"26", "lev":"2", "rx":"0", "ry":"0" },
    { "id":"26", "lev":"2", "rx":"1", "ry":"0" }
  ],
]
```

For the way to create a fv with map, it will cause an error if the tile is not enough to the size of the fv.

[Example 3] Map data in JSON : Tag



When id 2 and id 5 are associated with green tag and id 9 is associated with yellow.

```
map_data=
[
  [
    {"search":{"keyword":"testx","default":{ "id":"3", "lev":"1", "rx":"0", "ry":"0"}}},
    {"id":"79", "lev":"1", "rx":"0", "ry":"0"},
    {"search":{"keyword":"yellow"}}
  ],
  [
    {"search":{"keyword":"green","default":{"id":"99", "lev":"1", "rx":"0", "ry":"0"}}},
    {"search":{"keyword":"green","default":{"id":"78", "lev":"1", "rx":"0", "ry":"0"}}},
    {"search":{"keyword":"yellow"}}
  ]
]
```

when you generate a fv by using the above map

For the first column and the first row, testx tag does not exist, thus default tile id 3 is set.
For second column and the first row, becuase the id 79 is specified, id 79 is shown.
For third column and the first row, becuase the yello tag is specified, id 9 is shown.
The green tag is specified on the first and second row in second column, thus id 5 is displayed in first row and id 2 is displayed in second row. (The media will be displayed with newest order.)
Because the third column and the second row, the yellow tag is specified, but there is not any id for the tag. Thus the black media is set.

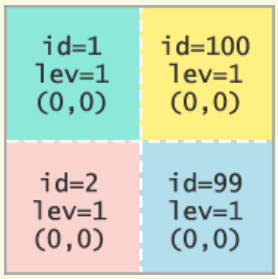
[Example 4] Map data in JSON : using Timeline

If there are 100 medias (media id 1 - 100 exists)

```
map_data=
[
  [
    {"id":"1", "lev":"1", "rx":"0", "ry":"0"},
    {"timeline":"true"}
```

```
  ],
  [
    { "id":"2", "lev":"1", "rx":"0", "ry":"0"},
    {"timeline":"true"}
  ]
]
```

For the above map, the larger number of media id is newest.
Thus the fv is generated as below.



[Example 5] Map data in JSON format : For using start position

When considering time position and duration with map, you can set the following JSON format.
{ “id”:”300”, “lev”:”1”, “rx”:”0”, “ry”:”0”, “st”:”6”,”dur”:”6”}
The st is the start time position of the media.
The dur is the playing time duration from the start time position of the media.

```
map_data=
[
  [
    { "id":"1", "lev":"1", "rx":"0", "ry":"0"},
    { "id":"2", "lev":"1", "rx":"0", "ry":"0", "st":"6","dur":"4"}
  ],
  [
    { "id":"3", "lev":"1", "rx":"0", "ry":"0", "st":"6","dur":"6"},
    { "id":"4", "lev":"1", "rx":"0", "ry":"0", "st":"6","dur":"8"}
  ]
]
```

For above example, “st” and “dur” is not set for id 1, thus start time position is set to 0, time duration is set to original media duration.
Because id 2 has “st” and “dur”, this media is played from 6 seconds point to 10 seconds point (for 4 seconds).

And, the media id 3 and id 4 is played from 6 seconds point for each 6 and 8 seconds.

If the time duration of fv is longer than the dur, the media is played repeatedly from the “st” time position for “dur” time duration.

page and num (number of items per page) parameters

You can limit the number of items in a response when you retrieve media information with API.

[Example]

- the number of output items: 600
- page: 3
- num: 250

Media information that is displayed in the response is from 501 items.

fv audio setting

You can include sound by specifying a media id to sound_ids part. When media specified by sound_ids does not have audio data, fv generation will be failed.

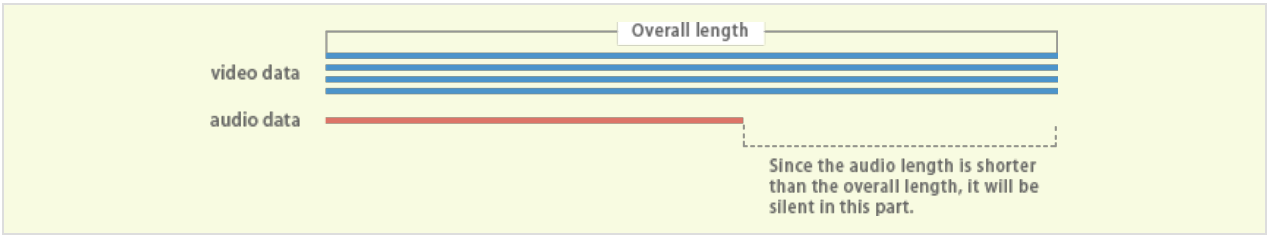
The length of the audio is up to the length specified in fv time duration. If the audio time duration that you specified by sound_ids is longer than the fv time duration, the excess will be deleted.

There are two audio repetition behaviors. First one is the case where the specified sound_ids is not included in media ids which constitute the fv. Second one is the case of vice versa.

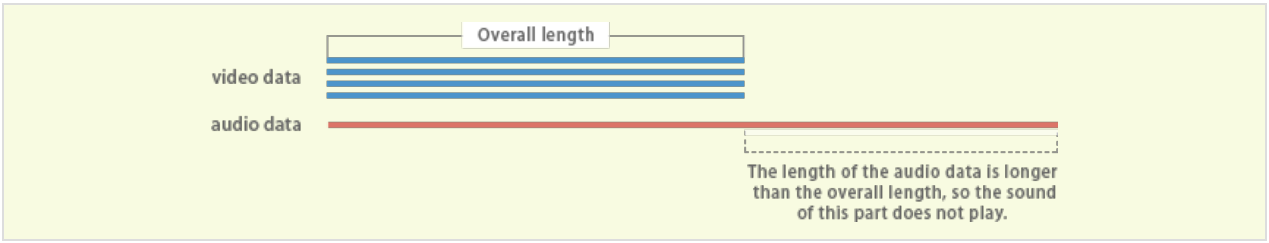
1. The case where the specified sound_ids is not included in media ids which constitute the fv.

In this case, if the fv time duration is larger than audio time duration, audio will not be repeated. If audio time duration is larger than fv time duration, the excess will be deleted.

Case: Creating 2×2 fv and audio data is shorter than the fv



Case: 2×2 fv and audio data is longer than the fv

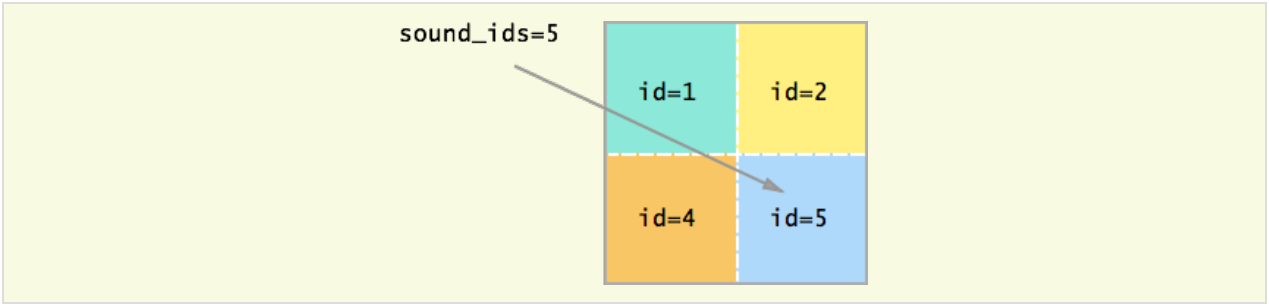


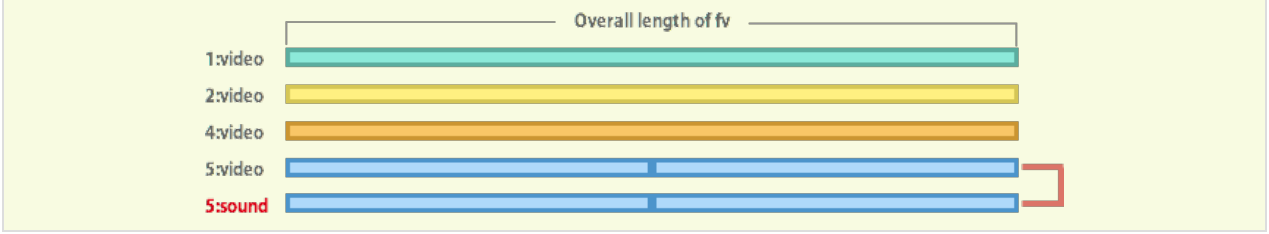
2. The case where the specified sound_ids is not selected in media ids which constitute the fv.

In this case, audio repetition behavior conforms a start_time and media_duration of media, which equals to sound_ids.

the case of 2x2 fv composed of level 1's tiles.

If media ids which constitute fv are 1, 2, 4, and 5, and sound_ids is specified by 5, audio repetition behavior is as following figure.

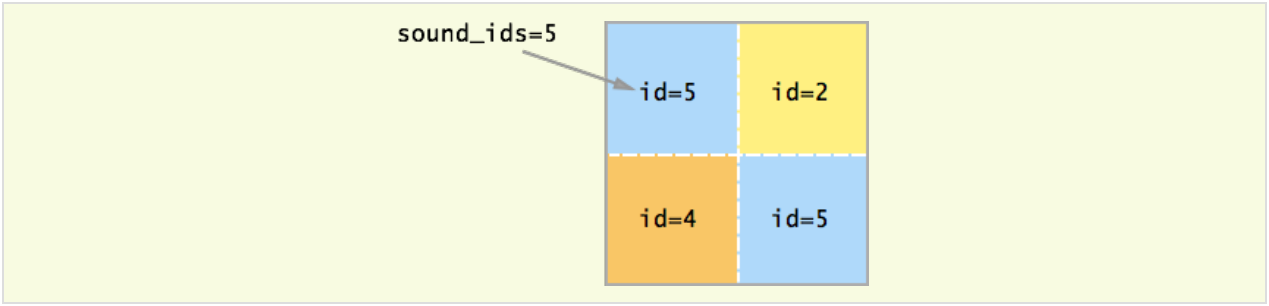


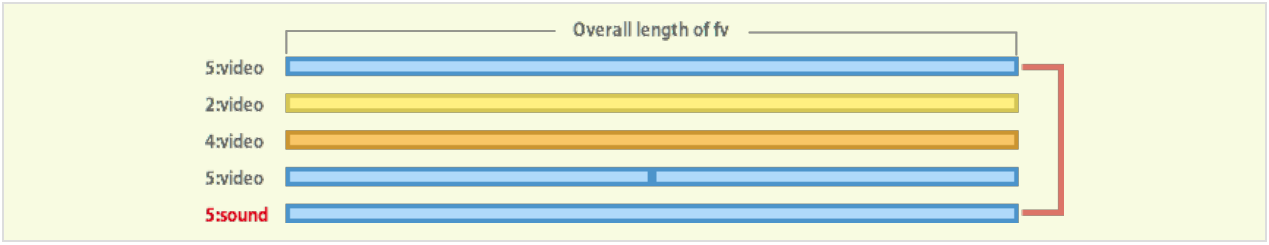


The case of 2x2 fv composed of level 1's tiles with same media id.

When media ids that constitute fv are 5, 2, 4, and 5, and sound ids is 5, audio repetition behavior conforms a start_time and media_duration

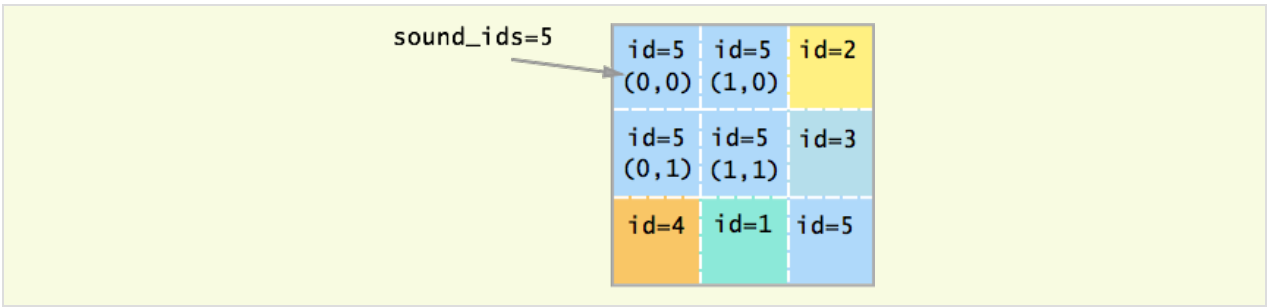
of top-left media as following figure.

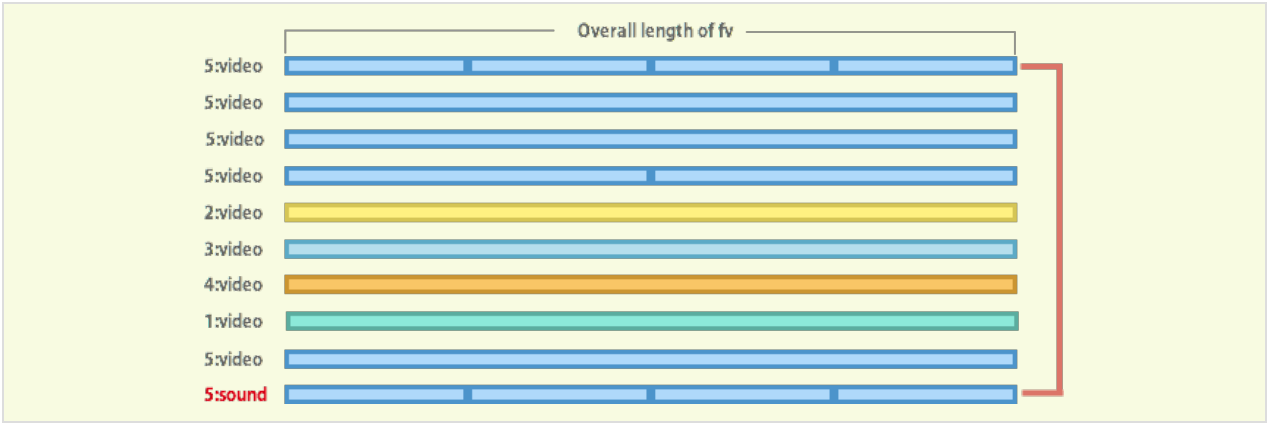




The case of 3x3 fv contains level 2's tiles as following figure.

When fv contains level 2's tiles and sound_ids is specified by 5, sound repetition behavior same as level 2's top-left tile. You require attention to generate fv when you want to specify the different repetition behavior [Appendix] between same levels by using map as following figure. The below figure represent the case at where four-times repetition for top-left in level 2's tiles and tow-times repetition for bottom-right in level 2's tiles are set. In this case, sound repetition behavior conforms top-left in level 2's tiles.



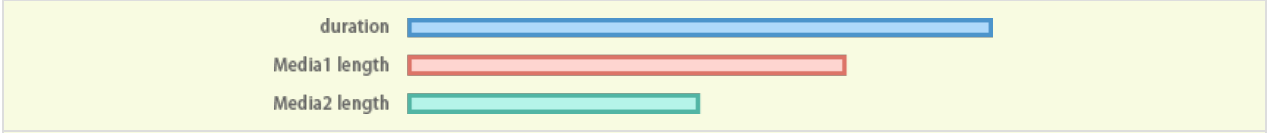


About fv time duratioin

fv time duration is determined by the loop and time duration parameter.
The default value of the loop parameter is 1.fv time duration is categorized to the below 3 patterns.

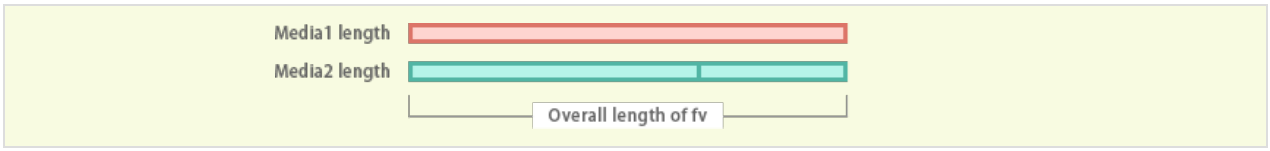
[Example] fv which have tile structure (1 × 2)

And, if the fv time duration, Media 1's time duration, and Media 2's time duration are as follows,



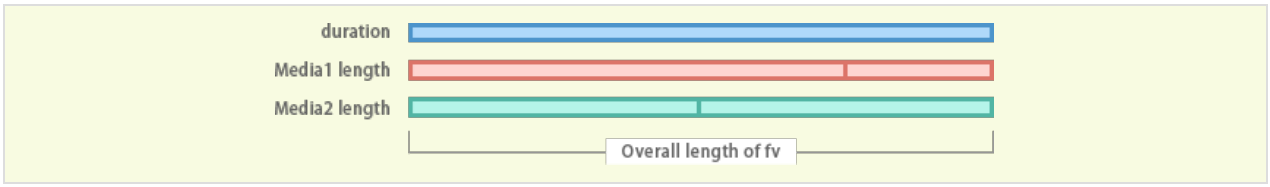
1. No setting of duration

The maximum time duration in medias included in fv will be set as fv time duration.
Medias that is less than longest time duration are played repeatedly.



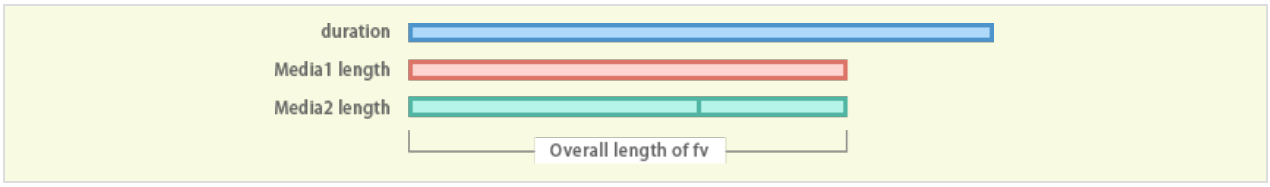
2. Duration parameter is set and loop parameter is set to 1

fv time duration is set to duration parameter.
Medias that is less than the specified duration are played repeatedly.



3. Duration parameter is set and loop parameter is set to 0. (same as pattern 1)

The maximum time duration in medias included in fv will be set as fv time duration.
Medias that is less than longest time duration are played repeatedly.



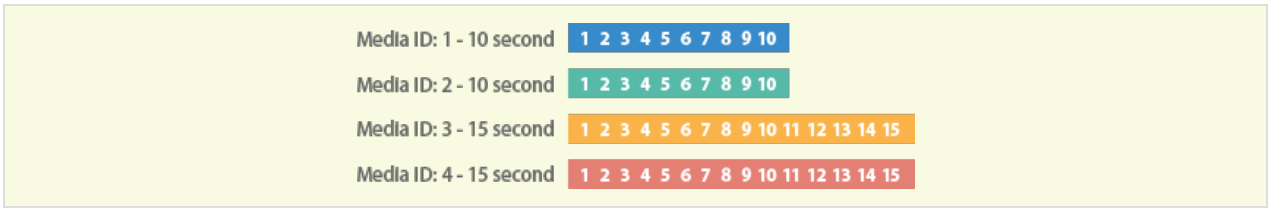
Example of start_time and media_duration parameter

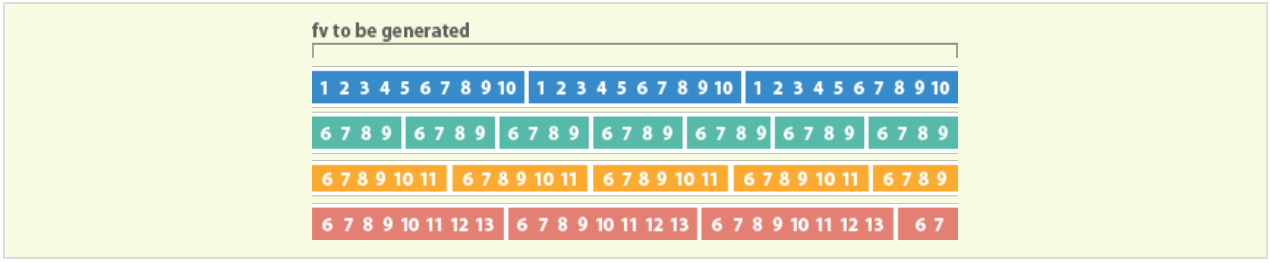
By the specifying the media_duration and start_time in case of fv creatioin with specified order, you can set the start time point and play time duration of each medias.

In the case of below map data

```
[ [
        { "id":"1", "lev":"1", "rx":"0", "ry":"0"},
        { "id":"2", "lev":"1", "rx":"0", "ry":"0", "st":"6","dur":"4"}
   ],
   [
        { "id":"3", "lev":"1", "rx":"0", "ry":"0", "st":"6","dur":"6"},
        { "id":"4", "lev":"1", "rx":"0", "ry":"0", "st":"6","dur":"8"}
] ]
```

If the fv time duration is set to 30 seconds, the fv which will be generated is represented as below image.





Because fv time duration is set to 30 seconds and each media time duration is less than 30 seconds, each media is played repeatedly from the start_time for media_duration between fv time duration.

Cropping and padding parameter in media registration phase

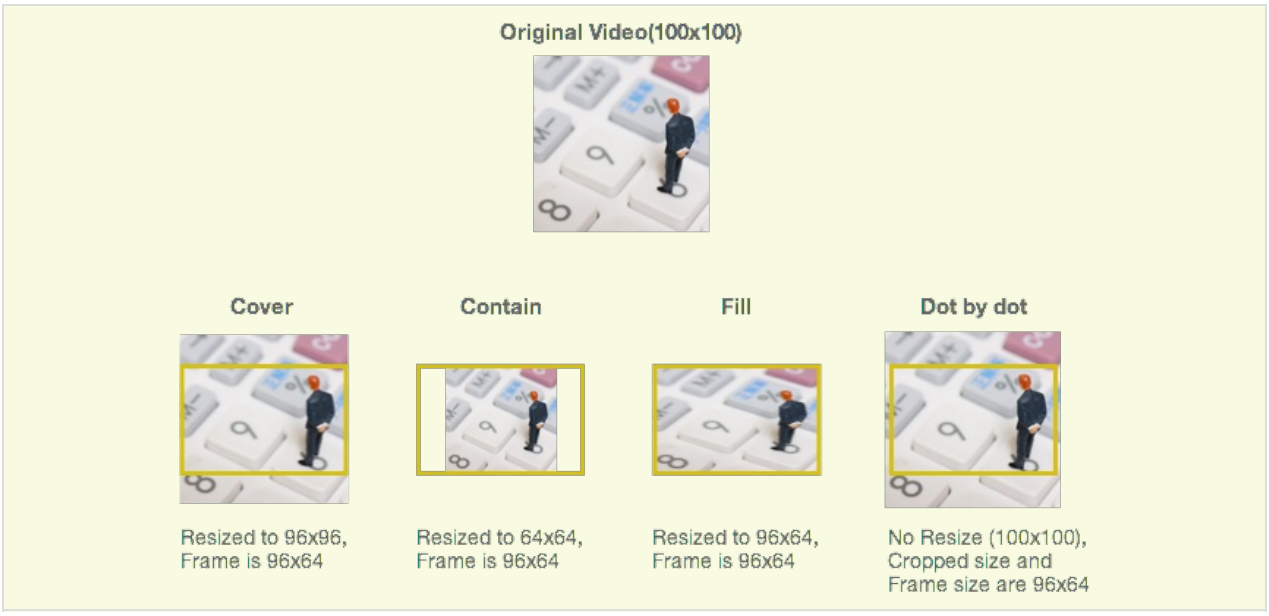
You can arrange the rendering area and position in media registration phase by specifying parameters for cropping and padding crop_position, padding_popsition, crop_inheriting, crop_policy_ob, and crop_policy_ib. For the way to use these parameters, please refer to “Video registration ”.

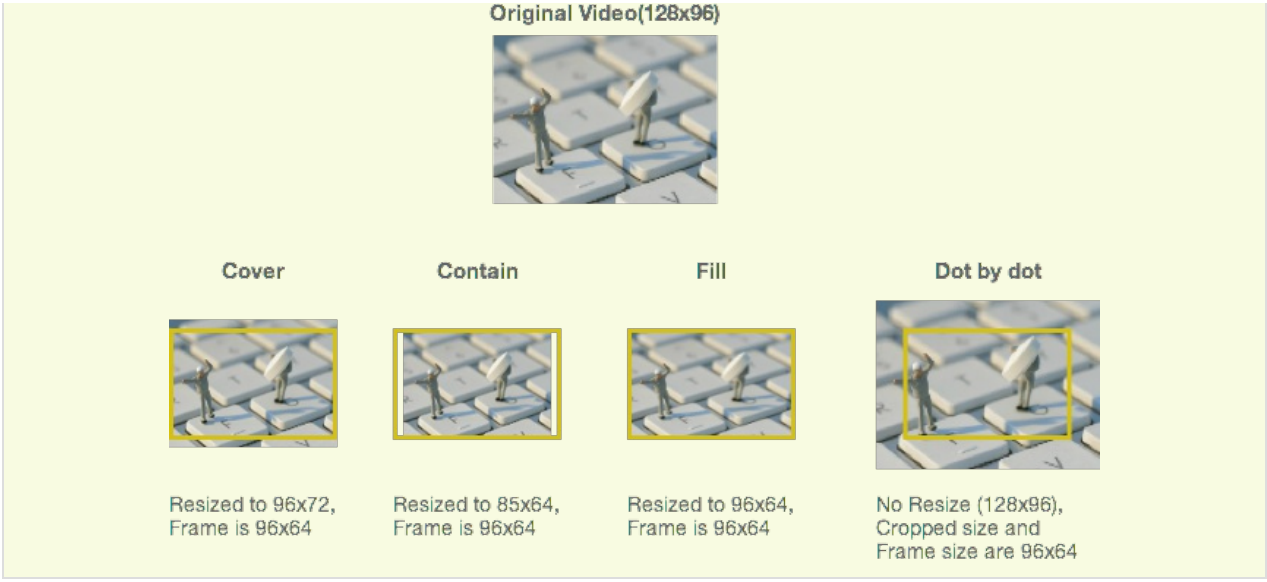
The detail contents of policy for crop/padding and rendering area.

The parameters crop_policy_ob and crop_policy_ib are used for padding and cropping gap between the frame size of input original media and the frame size of output tile.
Following figure describes the difference between original video and output tile file under the following conditions:
Conditions

Type	Attribute	Value
Input	Original media size	1. 100x100 (aspect ratio 1:1) 2. 128x96 (aspect ratio 4:3)
Output	TileType	R (aspect ratio 3:2)
	Level	1 (resolution is 96x64)
Policy	crop_position (ex. location of focus)	1: [Center]
	padding_position (ex. location of efficient region on output)	1: [Center]
	crop_inheriting (ex. Whether cropping / padding result for first output tile is reused for other level output tile by simply resizing first output or not)	1: [Optimizing cropping area each levels]

- Orange frame represents output tile size and outside of red-dot frame is not rendered on tile.



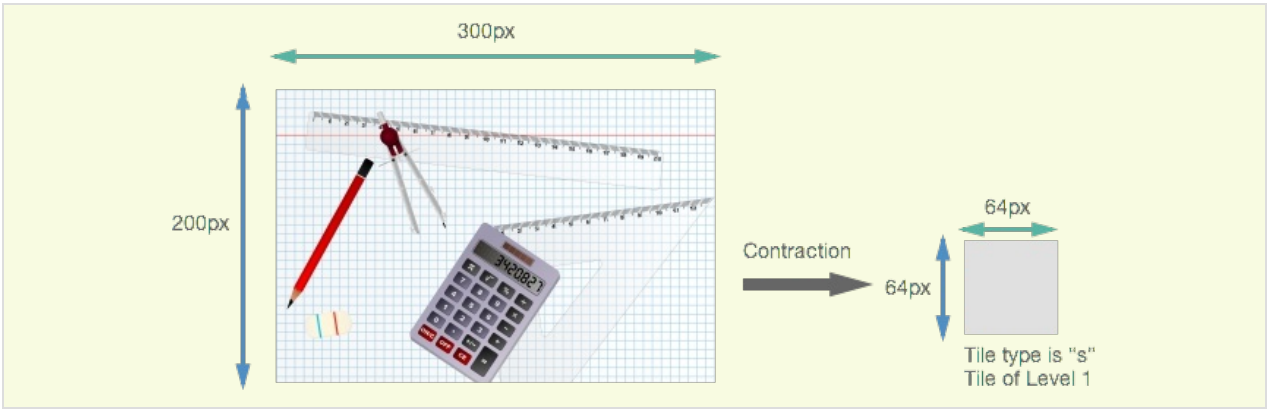


The difference of Cover and Contain

In this part, the difference of the case of Cover and Contain as crop policy is described by using following figure. In both cases original videos are resized keeping aspect ratio.

In Cover policy, original media is resized to completely cover output region (no vacant region). Among width and height of resized original media, the one fits with the output region and the other protrudes from output region and cropped.

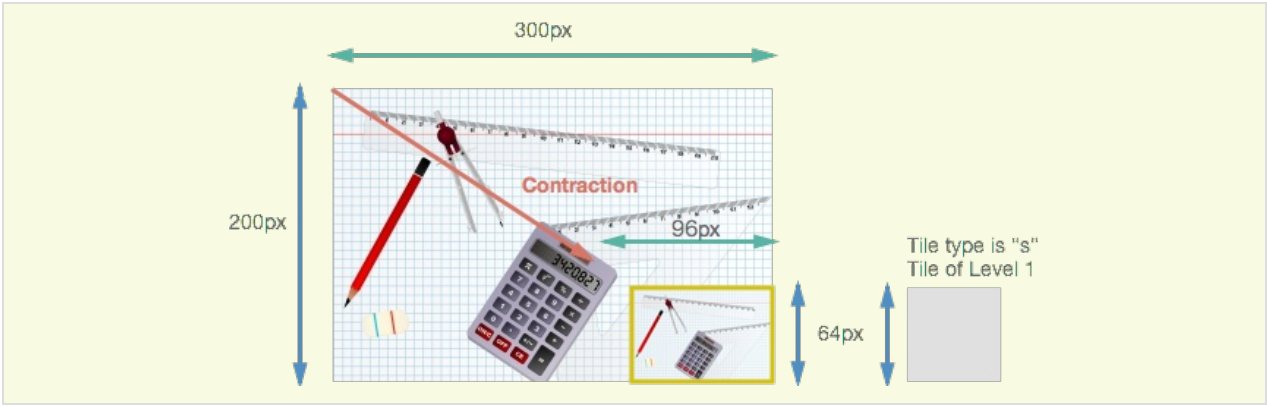
In Contain policy, original media is resized to be completely contained within the output region. Among width and height of resized original media, the one fits with the output region and the other is smaller than output region, and vacant region around that side is filled with black pixels.



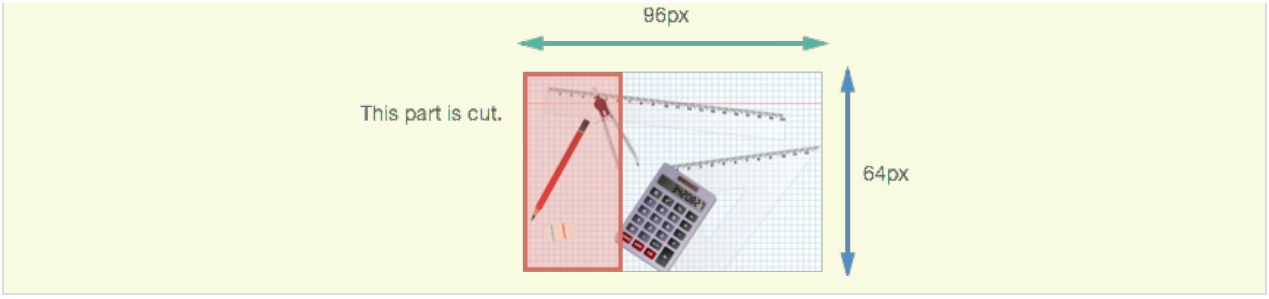
The Cover policy case (top-right/bottom-right alignment)

1. original media is resized to completely cover output region.

In this example, the height fits with the output region and the width protrudes from output region.



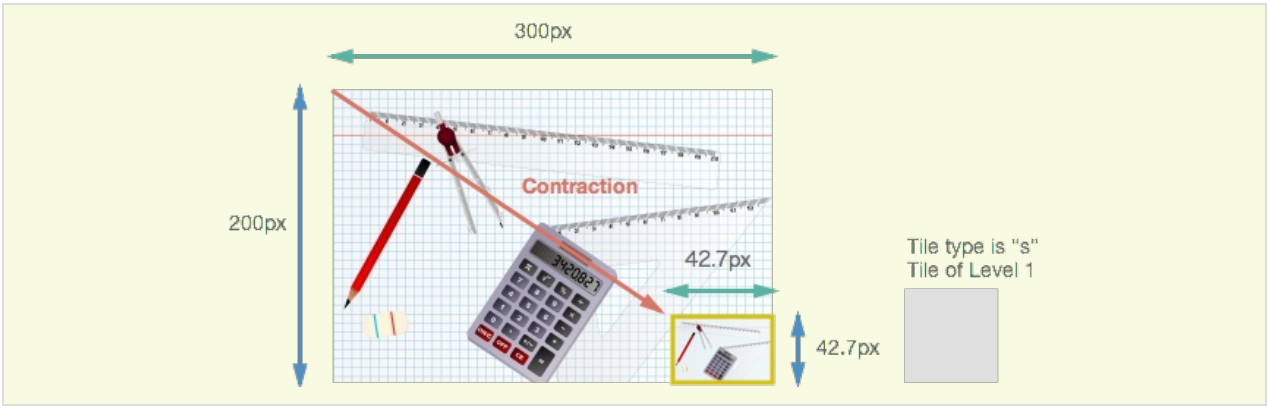
2. The part protrudes in target tile size is cropped.



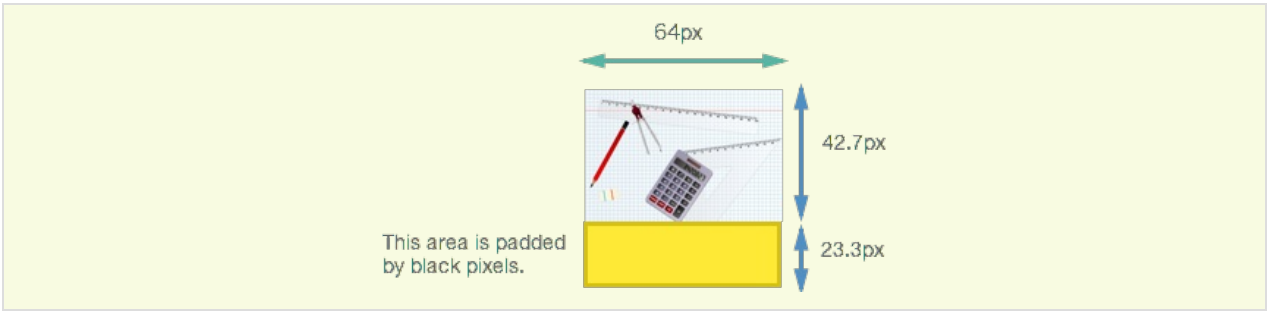
The Contain policy case (bottom-left/bottom-right alignment)

1. original media is resized to be completely included output region.

In this example, the width fits with the output region and the height become shorter than output region.



2. The part that does not reach to target tile size is padded by black pixels.



[Example 1] The case where the policy parameters about cropping and padding are not specified.

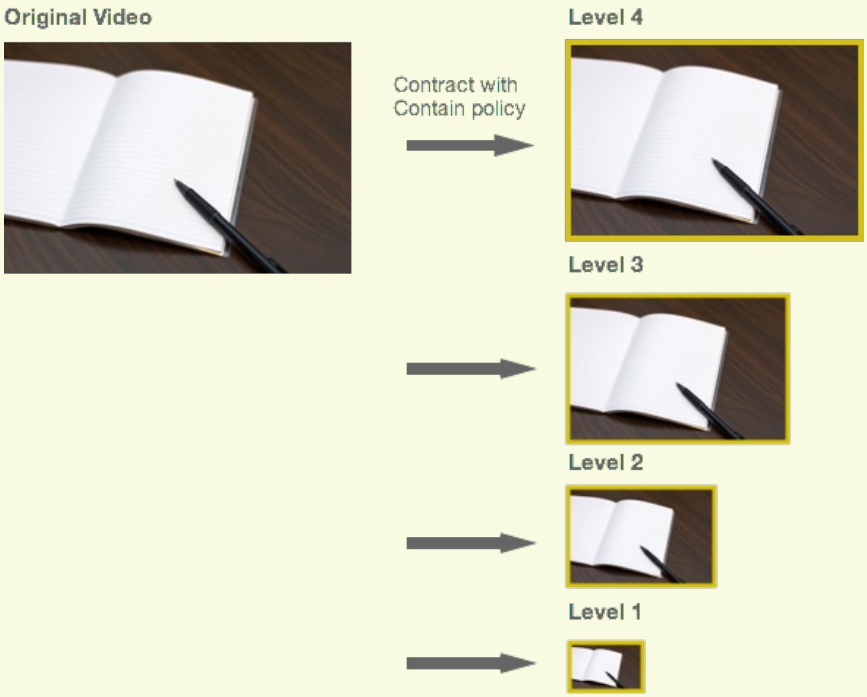
The conditions for this example are as follows.

Conditions

Type	Attribute	Value
Input	Original media size	1920x1080 (aspect ratio 16:9)
Output	TileType	HD (aspect ratio 16:9)
	Level	4,3,2,1 (resolutions are 1024x576, 768x432, 512x288, and 256x144 for each level)
Policy	crop_position (ex. location of focus)	Not specified, default value 1: [Center] is used
	padding_position (ex. location of efficient region on output)	Not specified, default value 1: [Center] is used
	crop_inheriting (ex. Whether cropping / padding result for first output tile is reused for other level output tile by simply resizing first output or not)	Not specified, default value 1: [Optimizing cropping area each levels.] is used
	crop_policy_ob (ex. The policy for cropping / padding when output resolution is bigger than output one)	Not specified, default value 1: [Contain] is used
	crop_policy_ib (ex. The policy for cropping/padding when	Not specified, default value 1: [Contain] is used

	input resolution is bigger than output one)

When tile_type is HD, original media size is resized to 1024x576, 768x432, 512x288, and 256x144, for each level. When the default policy parameter is used, crop_policy_ib is set to 1, which is the Contain policy. In this case, aspect ratio is same between original media and output tile size. Therefore, original media size is contracted with keeping aspect ratio without cropping or padding.



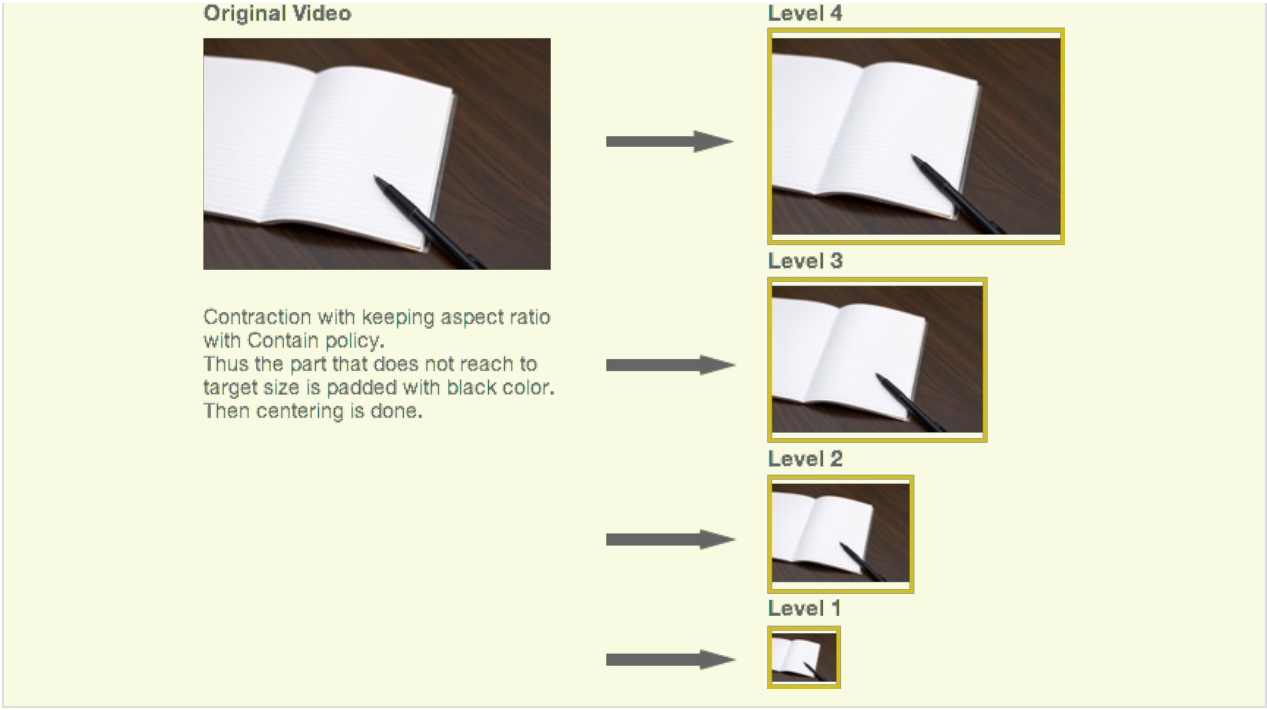
[Example 2] The case where the policy parameters about cropping and padding are not specified.

This example is same as example 1 except tile_type is SD. The conditions for this example are as follows.

Conditions

Type	Attribute	Value
Input	Original media size	1920x1080 (aspect ratio 16:9)
Output	TileType	SD (aspect ratio 4:3)
	Level	4,3,2,1 (resolutions are 512x384, 384x288, 256x192, 128x96 for each level)
Policy	crop_position (ex. location of focus)	Not specified, default value 1: [Center] is used
	padding_position (ex. location of efficient region on output)	Not specified, default value 1: [Center] is used
	crop_inheriting (ex. Whether cropping / padding result for first output tile is reused for other level output tile by simply resizing first output or not)	Not specified, default value 1: [Optimizing cropping area each levels.] is used
	crop_policy_ob (ex. The policy for cropping / padding when output resolution is bigger than output one)	Not specified, default value 1: [Contain] is used
	crop_policy_ib (ex. The policy for cropping/padding when input resolution is bigger than output one)	Not specified, default value 1: [Contain] is used

In this example, for all level (4, 3, 2, 1) crop_policy_ib:1[contain] is applied because original media size is smaller than output size (1920x1080 > 512x384, 384x288, 256x192, 128x96). Original media is resized so that resized media is completely contained within the output region. Among width and height of resized original media, the one fits with the output region and the other is smaller than output region, and vacant region around that side is filled with black pixels. According to option padding_position: 1 [center], the region around original videos are simply padded so that original video position is center on output. And according to policy crop_inheriting: 1, first output is not re-used but padding region is decided individually for each level and original video region sizes.



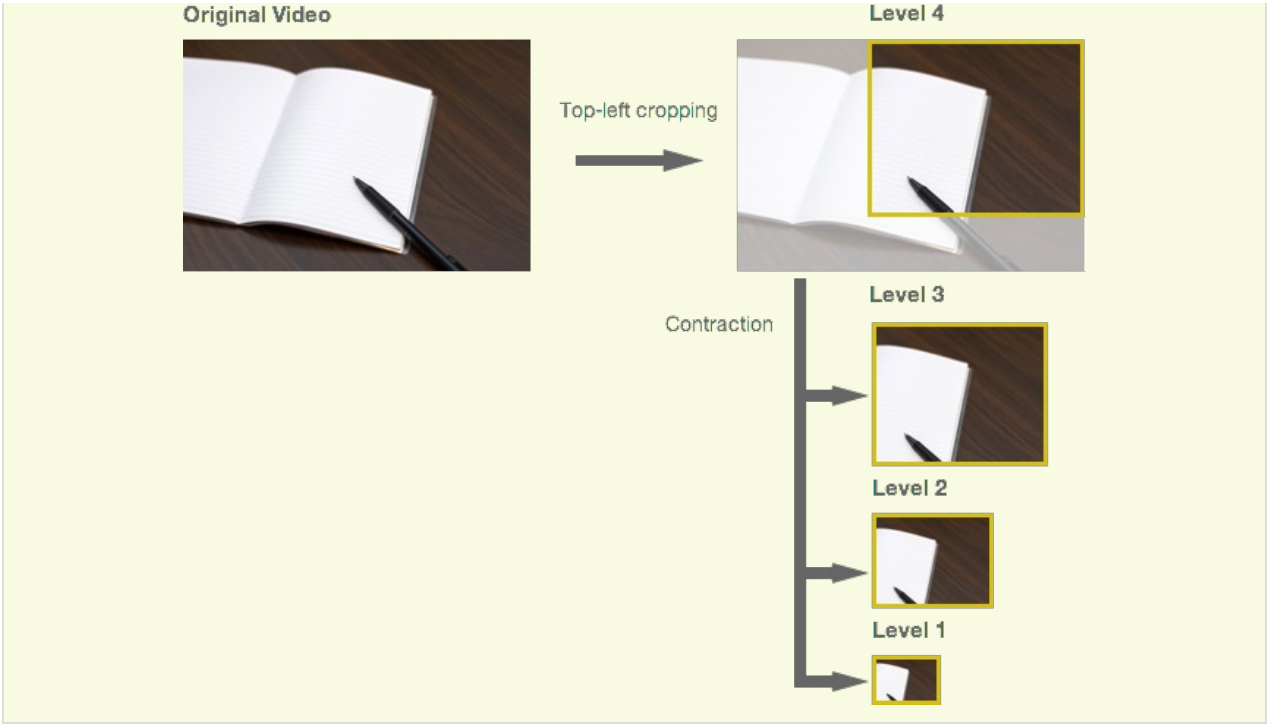
[Example 3] The case crop_policy is designated explicitly.

The conditions for this example are as follows.

Conditions

Type	Attribute	Value
Input	Original media size	1920x1080 (aspect ratio 16:9)
Output	TileType	R (aspect ratio 3:2)
	Level	4,3,2,1 (resolutions are 384x256, 288x192, 192x128, 96x64 for each level)
Policy	crop_position (ex. location of focus)	3: [top-right]
	padding_position (ex. location of efficient region on output)	2: [top-left]
	crop_inheriting (ex. Whether cropping / padding result for first output tile is reused for other level output tile by simply resizing first output or not)	0: [Applying for same cropping / padding area]
	crop_policy_ob (ex. The policy for cropping / padding when output resolution is bigger than output one)	3: [dot by dot] (simple padding)
	crop_policy_ib (ex. The policy for cropping/padding when input resolution is bigger than output one)	3: [dot by dot] (simple cropping)

In this case, as crop_policy_ib is 3, “dot by dot”, expansion or contraction is not applied and simply cropped according to the output tile resolution. In addition, because crop_position is 3, top-left part in original media is cropped. Also according to crop_inheriting is 0 [Applying for same cropping / padding area], the cropped picture in max level is reused for the creation of other level by simply resizing it. Thus the valid contents of all levels are same with first output even its region size is different.



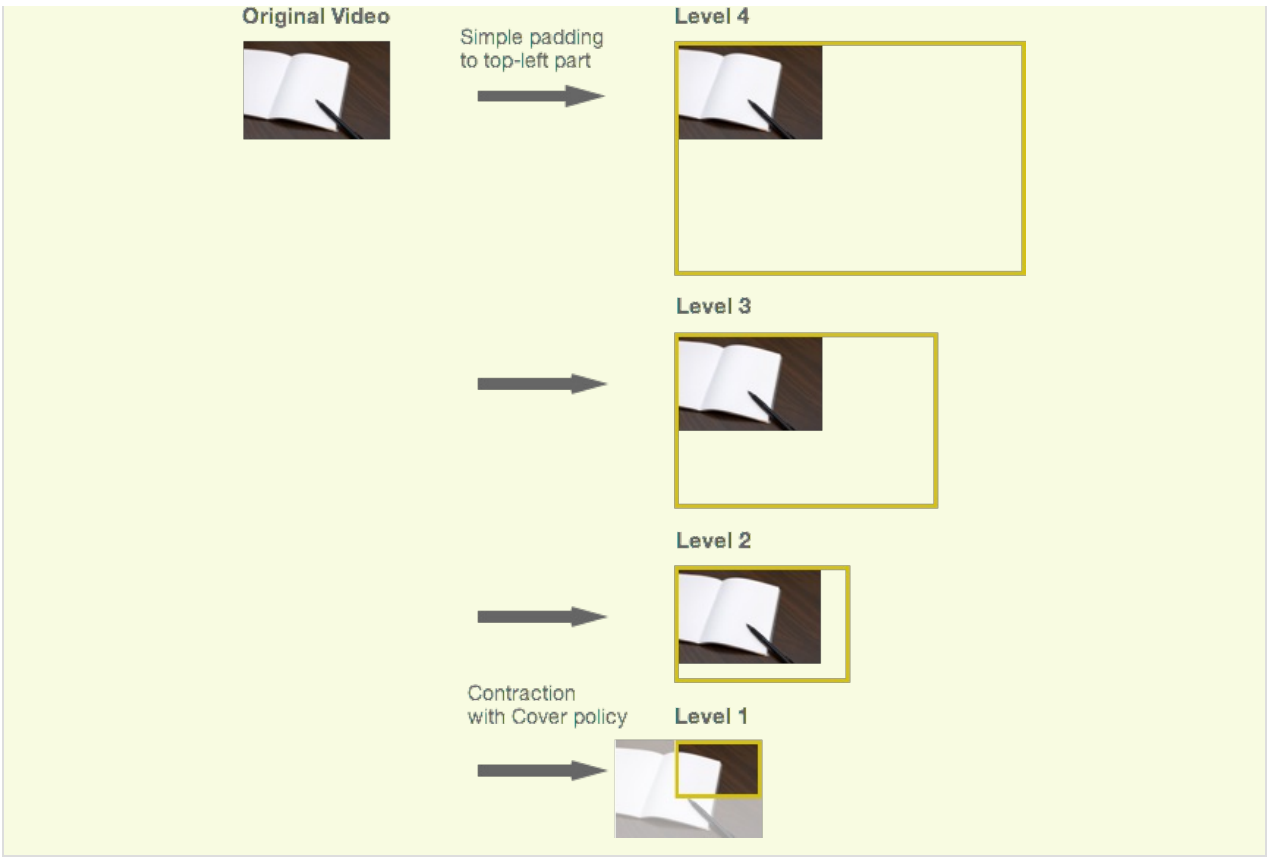
[Example 4] The case crop_policy is designated explicitly.

The conditions for this example are as follows.

Conditions

Type	Attribute	Value
Input	Original media size	128x64 (aspect ratio 2:1)
Output	TileType	R (aspect ratio 3:2)
	Level	4,3,2,1 (resolutions are 384x256, 288x192, 192x128, 96x64 for each level)
Policy	crop_position (ex. location of focus)	3: [top-right]
	padding_position (ex. location of efficient region on output)	2: [top-left]
	crop_inheriting (ex. Whether cropping / padding result for first output tile is reused for other level output tile by simply resizing first output or not)	1: [optimizing cropping area each levels]
	crop_policy_ob (ex. The policy for cropping / padding when output resolution is bigger than output one)	3: [dot by dot] (simple padding)
	crop_policy_ib (ex. The policy for cropping/padding when input resolution is bigger than output one)	0: [cover] (Resize without aspect ratio change and with shorter side of input be same size of output region)

In this example, for level 4, 3, and 2, crop_policy_ob is applied because original media size is smaller than output size (128x96 < 384x256, 288x192, 192x128) and according to option padding_position: 2 [top-left], the region around original videos are simply padded so that original video position top-left on output. And according to policy crop_inheriting: 1, first output is not re-used but padding region is decided individually for each level and original video region sizes and contents on outputs are same between level 4, 3, and 2. After then, for level 1, crop_policy_ib:0 [cover] is applied because input size is larger than output size (128x96 > 96x64) and original media is resized so that it completely cover output region (no vacant region). Among width and height of resized original media, the one fits with the output region and the other protrudes from output region and cropped. Cropped region is left side of output according to the policy crop_position: 3 [top-right].



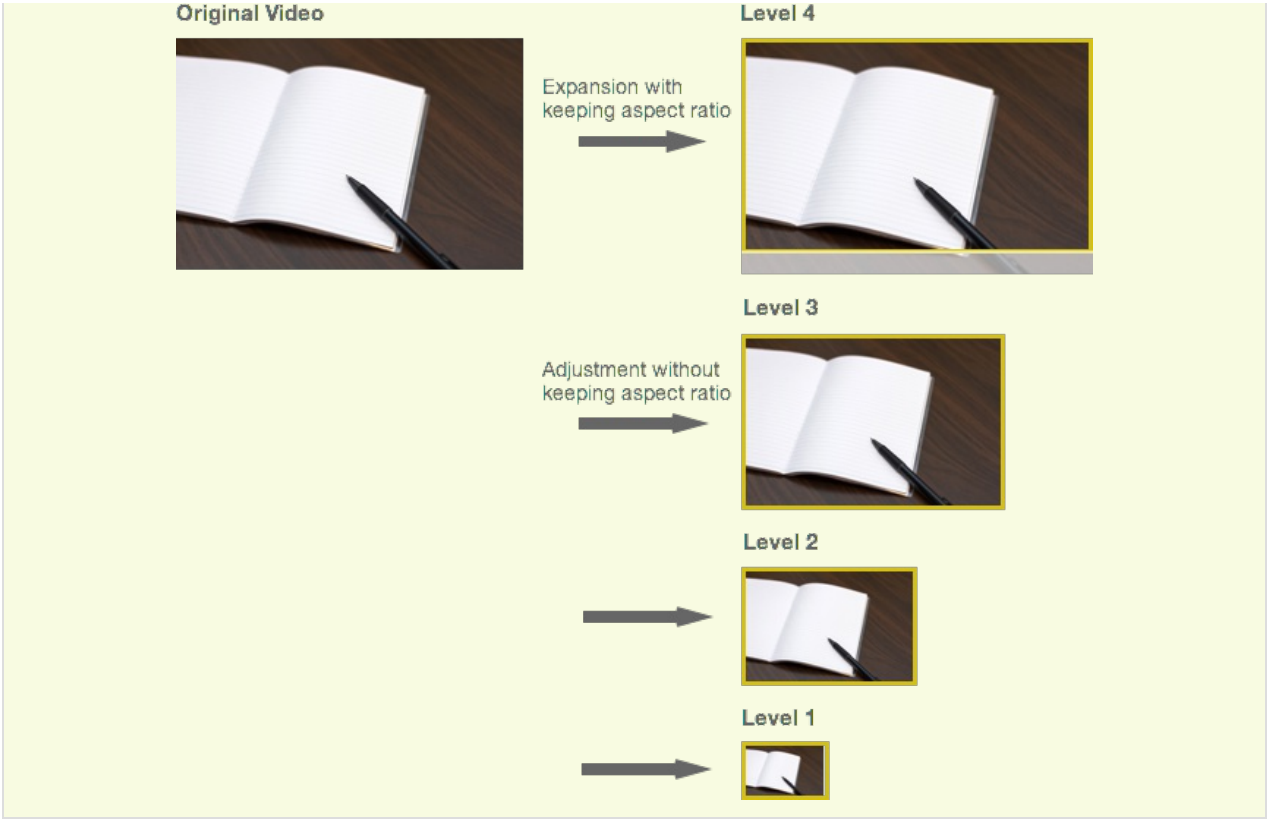
[Example 5] The case crop_policy is designated explicitly.

The conditions for this example are as follows.

Conditions

Type	Attribute	Value
Input	Original media size	320x240 (aspect ratio 4:3)
Output	TileType	R (aspect ratio 3:2)
	Level	4,3,2,1 (resolutions are 384x256, 288x192, 192x128, 96x64 for each level)
Policy	crop_position (ex. location of focus)	3: [top-right]
	padding_position (ex. location of efficient region on output)	2: [top-left]
	crop_inheriting (ex. Whether cropping / padding result for first output tile is reused for other level output tile by simply resizing first output or not)	1: [optimizing cropping area each levels]
	crop_policy_ob (ex. The policy for cropping / padding when output resolution is bigger than output one)	0: [cover] (Resize without aspect ratio change and with longer side of input be same size of output region)
	crop_policy_ib (ex. The policy for cropping/padding when input resolution is bigger than output one)	2: [fill] (Resizing to full output region with aspect ratio change)

In this example, for level 4, crop_policy_ob [0: cover] is applied because original media size is smaller than output size (320x240 < 384x256) so original media is resized so that it completely cover output region (no vacant region). Among width and height of resized original media, the one fits with the output region and the other protrudes from output region and cropped. According to option crop position: 3 [top-right], the region cropped is left part of output. After then, for level 3, 2, and 1, crop_policy_ib: 2 [fill] is applied because input size is larger than output size (320x240 > 288x192, 192x128, 96x64) and original video is resized to fulfill output tile size with its aspect ratio changed.



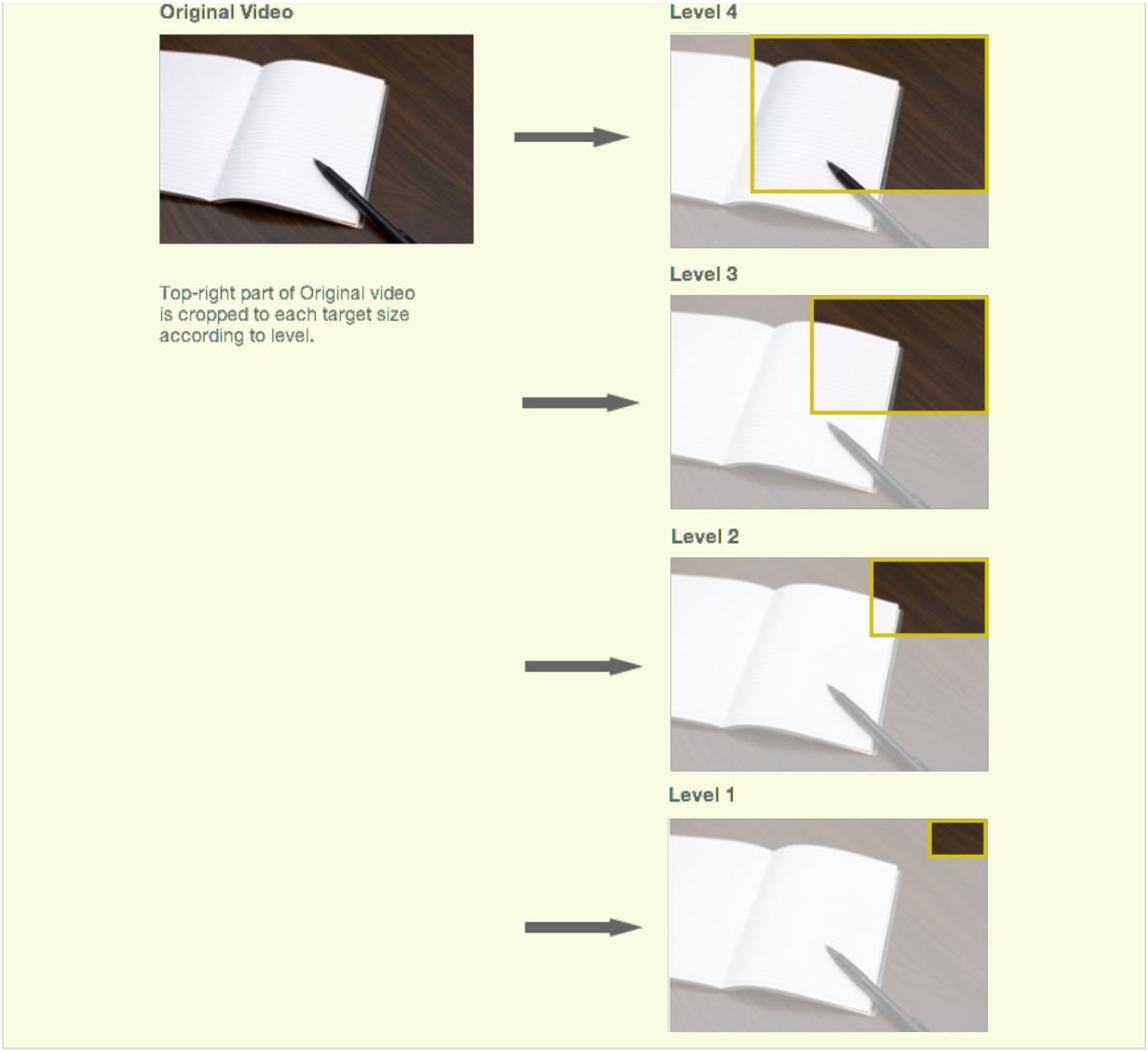
[Example 6] The case crop_policy is designated explicitly and same as example 3 except crop_inheriting parameter is 1.

The conditions for this example are as follows.

Conditions

Type	Attribute	Value
Input	Original media size	1920x1080 (aspect ratio 16:9)
Output	TileType	R (aspect ratio 3:2)
	Level	4,3,2,1 (resolutions are 384x256, 288x192, 192x128, 96x64 for each level)
Policy	crop_position (ex. location of focus)	3: [top-right]
	padding_position (ex. location of efficient region on output)	2: [top-left]
	crop_inheriting (ex. Whether cropping / padding result for first output tile is reused for other level output tile by simply resizing first output or not)	1: [optimizing cropping area each levels]
	crop_policy_ob (ex. The policy for cropping / padding when output resolution is bigger than output one)	3: [dot by dot] (simple padding)
	crop_policy_ib (ex. The policy for cropping/padding when input resolution is bigger than output one)	3: [dot by dot] (simple cropping)

In this case, because crop_policy_ib is 3, “dot by dot”, expansion or contraction is not applied. In addition, as crop_position is 3 [top-right], outside of top-right region in original media is cropped. In example 3, according to crop_inheriting is 0 [Applying for same cropping area], the cropped picture in max level (level 4) is reused for other level’s outputs. But, in this example, as crop_inheriting is 1 [Optimizing cropping area each levels], tile of level 1, 2, and 3 is created individually from original media so the valid region of all level’s outputs are different among each output tiles.



These contents are a thing of July 31, 2017.
These contents may be changed in the future to improve our service.