fv ソリューション API

2017.07.31

目次

fvLIBRARY API

ここでは、fvLIBRARY API (以降 API) について記述しています。
API は、主に次の 3 タイプのもので構成されます。

1. 動画登録 API
2. fv生成要求 API
3. メディア管理 API

各タイプはそれぞれ Web用API (REST API) として提供されます。
なお、本ドキュメントはライト・スタンダードプラン向けの内容で記述されています。プランによるAPI仕様の違いについては、プランによるAPI仕様相違点 を参照してください。

API基本制限

タイルの生成

API では、動画を登録すると 1 秒〜900 秒（15分）までの整数秒で、指定したタイルを生成します。ただし、どのレベルを指定してもレベル 1 のタイルは常に生成されます。それぞれの基本タイルの仕様は以下の通りです。

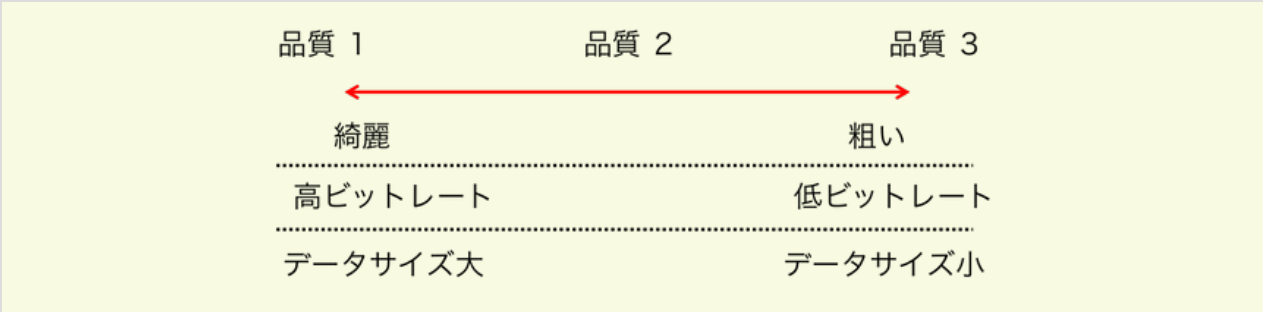
タイル種別	解像度[px]			
	レベル１	レベル２	レベル３	レベル４
HD	256 x 144	512 x 288	768 x 432	1024 x 576
SD	128 x 96	256 x 192	384 x 288	512 x 384
R	96 x 64	192 x 128	288 x 192	384 x 256
S	64 x 64	128 x 128	192 x 192	256 x 256

もし、指定した秒数が登録動画より長い場合は、登録動画の秒数でタイルが生成されます。指定がない場合や 0 が指定されていた場合も同様に、登録動画の秒数でタイルが生成されます。
1 秒以下は繰り上げられ、端数分の時間は最後のフレームで埋められます。

> ノート
>
> デフォルトでタイルの結合時には、上下左右に 2px 分の黒い枠が表示され、実際に表示されるのは上記サイズの縦横の長さから -4px したものになります。 本値は0pxにすることも可能ですので、詳細はお問い合わせ下さい。

タイルの品質について

タイルには、品質を設定することが出来ます。選択出来る品質は３段階あり、以下のような関係が有ります。



品質はタイルを生成する段階に複数指定することが出来、fvを生成する段階で単一の値を設定することが出来ます。 そのようにすることで、サービスや帯域に合わせたfvを要求することが出来るようになります。

メディアのデータ

Gnzoでは生成したタイル、メディアのメタデータを保持しています。

メタデータ	文字制限
media_id	半角数字
caption	半角英数、全角、記号（1〜120字）
tag	半角英数、アンダーライン（1〜120字）

	カンマを入れて複数入力が可能
description	半角英数、全角、記号（1〜1000字）
create_date	yyyy-mm-dd hh:mm:ss
update_date	yyyy-mm-dd hh:mm:ss
status	0,1

fabric videoの最大サイズ

fabric video（以下、「fv」）はタイル種別が同じものであれば自由に結合できますが、生成できる最大のサイズ（タイルの個数）に制限を設けています。

タイル種別	**最大サイズ**	
	縦×横 [個数]	**解像度 [px]**
HD	7x7	1792x1008
SD	11x11	1408x1056
R	15x15	1440x960
S	16x16	1024x1024

APIの認証について

各 API を呼出す際に、サーバー側で認証を行っています。
認証方法は、Referrer+Key、Basic 認証（ユーザ名、パスワード）, GIAS(Gnzo Integrated Authentication System)を利用したAccessTokenの 3 通りがあります。
なお、（D）マークがついたAPIは将来廃止される可能性がありますので、ご注意下さい。

API		**Referrer + Key 認証**	**Basic 認証**	**AccessToken**
動画登録 API	POST /api/1/videos	×	○	○
fv生成要求 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	×	○	○
メディア管理 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	×	○	○

全てのAPIで、まず有効なAccessTokenが指定されているかをチェックし、指定されていなければBasic認証、Basic認証情報が無ければ、最後にReferrer+Key（もしくはKeyのみ） での認証を行います。AccessTokenでの認証、Basic認証はHTTPSでのアクセスが必要です。HTTPSではない方法で、AccessToken認証またはBasic認証を行おうとすると、403 Errorが発生します。 Referrer + Keyによる認証では、誰でもKey情報を知ることが可能であるため、使用可能なAPIを制限しております。 AccessTokenによる認証はここ（）を参照して下さい。（またはお問い合わせ下さい。）

動画投稿のファイルフォーマット

動画登録の入力となるファイルのフォーマットとビデオ及びオーディオの符号化形式（コーデック）
について記載してます。

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

動画登録API

動画の登録

サーバに動画を登録します。動画の登録が完了すると、タイルの生成が自動的に開始されます。 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
```

ノート

メディアの有効化について：本 API を実行してもメディアは有効になりません。登録した動画（メディア）を fv に表示させるためには、メディアの有効化が必要になります。タイルの生成が完了してから、メディアの有効化 API によりメディアの有効化を行います。[詳細] 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 形式
    - x : XML 形式で出力
    - 指定無し : JSON 形式で出力
- **upfile** [必須]
  - アップロードするファイル（バイナリ）200MBまで。
- **tile_type** [必須]
  - タイル種別。1 種類のみ指定可能。
  - hd 16:9 / sd 4:3 / r 3:2 / s 1:1
- **duration**
  - 生成するタイルの時間（1〜900秒）。整数で指定
    時間は登録動画の時間以内で指定してください。登録動画の時間を超えた値を指定すると自動的に duration は登録動画の時間までとなります。
- **levels**
  - 生成するタイルのレベルを指定。カンマで複数指定が可能。指定なしの場合、レベル1 のタイルを生成。
    レベル1のタイルは動画登録時に自動的に生成されます。
    ex)
    level=2 と指定：レベル1,2 のタイルが生成
    level=3,4 と指定：レベル1,3,4 のタイルが生成
- **tile_qualitys** [必須]
  - 生成するタイルの品質を指定。カンマで複数指定可能。指定なしの場合、エラーを返す。
    ex)
    tile_qualitys=2と指定：品質2のタイルが生成されます。
    tile_qualitys=1,2,3と指定：品質1,2,3のタイルが生成
- **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**
  - 登録動画のフレームレートに基づいたフレームを指定。10個まで指定可能。
    サムネイルはデフォルトでは 0フレーム目（最初のフレーム）から自動的に jpg ファイルを生成します。thumbnail_position でフレーム指定をしていても、デフォルトのサムネイル（ 0フレーム目 ）は生成されます。
    ex)
    thumbnail_position=5 と指定: 0,5フレームのサムネイルが生成（2枚生成）

- **crop_position**
  - 入力サイズが、出力サイズより大きい場合に使用されるパラメータ
    1 : 中央
    2 : 左上
    3 : 右上
    4 : 左下
    5 : 右下

- **padding_position**
  - 出力サイズが、入力サイズより大きい場合に使用されるパラメータ
    1 : 中央
    2 : 左上
    3 : 右上
    4 : 左下
    5 : 右下

- **crop_inheriting**
  - 複数の出力サイズに対して、タイルを同時生成する場合のポリシーを選択します。
    0 (On) : 有効表示範囲をレベル間で統一
    1 (Off) : 有効表示範囲をレベル毎に最適化

- **crop_policy_ob**
  - 出力サイズが、入力サイズより大きい場合の伸縮ポリシー
    0 : [Cover] アスペクト比を保ってリサイズ（リサイズ後登録動画 ⊃ 出力領域）
    1 : [Contain] アスペクト比を保ってリサイズ（リサイズ後登録動画 ⊂ 出力領域）
    2 : [Fill] アスペクト比を崩してリサイズ
    3 : [Dot by dot] 単純なパディング

- **crop_policy_ib**
  - 入力サイズが、出力サイズより大きい場合の伸縮ポリシー
    0 : [Cover] アスペクト比を保ってリサイズ（リサイズ後登録動画 ⊃ 出力領域）
    1 : [Contain] アスペクト比を保ってリサイズ（リサイズ後登録動画 ⊂ 出力領域）
    2 : [Fill] アスペクト比を崩してリサイズ
    3 : [Dot by dot] 単純なクロップ

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**
  - 処理結果
    success : 成功
    それ以外 : 失敗
- **media_id**
  - 登録されたメディアID
- **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生成要求API

指定順の fv の取得

指定順の fv を取得します。
この API で表示対象になるタイルは、メディアの有効化API でアクティベートされたメディアが対象になります。

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

ノート

fv を構成するメディアの並び順を指定する方法は 2 種類あります。
各指定方法によって必須パラメータは下記のように変わります。タイル種別及びタイル品質が異なるメディアID を混在して指定することはできません。
1. media_id と level を紐づけてそれぞれカンマ区切りで指定する方法 [詳細]
   → 必須パラメータ：media_ids, levels または level
2. JSON 形式のマップを指定する方法 [詳細]
   → 必須パラメータ：map_data

Parameters

- **key** [必須]
  - クライアント認証キー
- **output_type**
  - Response 形式
    - x : XML 形式で出力
    - 指定無し : JSON 形式で出力
- **duration**
  - 生成する fv の時間（1〜900秒）
    整数で指定。[詳細]
- **loop**
  - fv の長さを決定するパラメータ [詳細]
- **row** [必須]
  - 生成する fv の行数
- **col** [必須]
  - 生成する fv の列数
- **media_ids**
  - fv を構成するメディアID
- **levels**
  - media_ids で指定した各メディアID の level を指定。
- **level**
  - media_ids で指定した各メディアID の level を一括指定。
- **map_data**
  - JSON 形式のマップデータ
- **sound_ids**
  - fv に埋め込む音声をメディアID で指定。[詳細]
- **tile_type**
  - タイル種別
    hd 16:9 / sd 4:3 / r 3:2 / s 1:1
    本パラメータが指定されない場合で、かつ指定するmedia_idの中に異なるタイルタイプを持つタイルが存在する場合はエラーが返ります。
    ※ tile_type は、map_data 指定での fv 生成時で、すべて search 指定かつ default でのメディアID指定がない場合必要になります。
- **tile_quality**
  - タイル品質
    1 高 / 2 中 / 3 低
  - 本パラメータが指定されない場合には、指定するmedia_idの中で最初に指定されるmedia_idの品質が適用され、そのタイル品質で生成されていないタイルは、黒色のタイルで埋められます。
    ※ tile_quality は、map_data 指定での fv 生成時で、すべて search 指定かつ default でのメディアID指定がない場合必要になります。
- **start_time**
  - media_ids で指定した各メディアの開始時間を指定 [詳細]
- **media_duration**
  - media_ids で指定した各メディアの時間を指定 [詳細]

Response

- **status**
  - 処理結果
    success : 成功
    それ以外 : 失敗
- **cached_data**
  - fvのmp4ファイルが既に作成されているかどうかを示すフラグ。既に作成されているならば”1”が入り、本リクエストをで作成されたmp4ファイルならば”0”が返る。
- **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。
- **video_url**
  - 生成された fv のURL
- **thumbnail_url**
  - 生成された fv のサムネイルのURL
- **map_data**
  - マップデータ

Request Example

media_ids と levels で指定

```
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指定

```
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-xxxxxxxxxxxxx.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>
      <rx>0</rx>
      <ry>0</ry>
      <caption />
      <description />
    </entry>
    </entry>
  <entry>
    <entry>
      <id>57</id>
      <lev>1</lev>
      <rx>0</rx>
      <ry>0</ry>
      <caption />
      <description />
    </entry>
    <entry>
      <id>52</id>
      <lev>1</lev>
      <rx>0</rx>
      <ry>0</ry>
      <caption />
      <description />
    </entry>
  </entry>
  </map_data>
  </response>
```

タイムライン順のfvの取得

投稿時間の降順で fv を生成します。
fv のタイル構成は、指定したタイル種別のレベル 1 のみを使用した構成となります。

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

Parameters

- **key** [必須]
  - クライアント認証キー
- **output_type**
  - Response 形式
    - x : XML 形式で出力
    - 指定無し : JSON 形式で出力
- **duration**
  - 生成する fv の時間（1〜900秒）。整数で指定。[詳細]
- **loop**
  - fv の長さを決定するパラメータ [詳細]
- **row** [必須]
  - 生成する fv の行数
- **col** [必須]
  - 生成する fv の列数
- **tile_type** [必須]

  - タイル種別。1 種類のみ指定可能。
    hd 16:9 / sd 4:3 / r 3:2 / s 1:1

- **tile_quality** [必須]
  - タイル品質。1 種類のみ指定可能。
    1 :高 / 2 :中 / 3 :低

- **sound_ids**
  - fv に埋め込む音声をメディアID で指定。[詳細]

- **page**
  - 表示するページ位置。1 以上の整数。[詳細]

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

- **has_next**
  - 次ページの有無
    true : 有
    false : 無

- **video_url**
  - 生成された fv のURL

- **thumbnail_url**
  - 生成された fv のサムネイルのURL

- **cached_data**
  - fvのmp4ファイルが既に作成されているかどうかを示すフラグ。既に作成されているならば”1”が入り、本リクエストをで作成されたmp4ファイルならば”0”が返る。

- **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**
  - マップデータ

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>
```

ランダム順のfvの取得

ランダム順で fv を生成します。
fv のタイル構成は、指定したタイル種別のレベル 1 のみを使用した構成となります。

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

Parameters

- **key** [必須]
  - クライアント認証キー
- **output_type**
  - Response 形式
    - x : XML 形式で出力
    - 指定無し : JSON 形式で出力
- **duration**
  - 生成する fv の時間（1〜900秒）。整数で指定。[詳細]
- **loop**
  - fv の長さを決定するパラメータ [詳細]
- **row** [必須]
  - 生成する fv の行数
- **col** [必須]
  - 生成する fv の列数
- **tile_type** [必須]
  - タイル種別。1 種類のみ指定可能。
    hd 16:9 / sd 4:3 / r 3:2 / s 1:1
- **tile_quality** [必須]
  - タイル品質。1 種類のみ指定可能。
    1 :高 / 2 :中 / 3 :低

- **sound_ids**
  - fv に埋め込む音声をメディアID で指定。[詳細]

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

- **video_url**
  - 生成された fv のURL

- **thumbnail_url**
  - 生成された fv のサムネイルのURL

- **cached_data**
  - fvのmp4ファイルが既に作成されているかどうかを示すフラグ。既に作成されているならば”1”が入り、本リクエストをで作成されたmp4ファイルならば”0”が返る。

- **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**
  - マップデータ

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>
```

キーワード検索結果のfvの取得

入力されたキーワードの検索結果で fv を生成します。
タグは完全一致、キャプションは部分一致で検索されます。fv のタイル構成は、指定したタイル種別のレベル1 のみを使用した構成となります。

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

Parameters

- **key** [必須]
  - クライアント認証キー
- **output_type**
  - Response 形式
    - x : XML 形式で出力
    - 指定無し : JSON 形式で出力
- **duration**
  - 生成する fv の時間（1〜900秒）。整数で指定。[詳細]
- **loop**
  - fv の長さを決定するパラメータ [詳細]
- **row** [必須]
  - 生成する fv の行数
- **col** [必須]
  - 生成する fv の列数
- **tile_type** [必須]
  - タイル種別。1 種類のみ指定可能。
    hd 16:9 / sd 4:3 / r 3:2 / s 1:1
- **tile_quality** [必須]
  - タイル品質。1 種類のみ指定可能。
    1 :高 / 2 :中 / 3 :低
- **sound_ids**
  - fv に埋め込む音声をメディアID で指定。[詳細]
- **page**
  - 表示するページ位置。1 以上の整数。[詳細]
- **keywords** [必須]
  - 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**
  - 処理結果
    success : 成功
    それ以外 : 失敗
- **has_next**
  - 次ページの有無
    true : 有
    false : 無
- **video_url**
  - 生成された fv のURL
- **thumbnail_url**
  - 生成された fv のサムネイルのURL
- **map_data**
  - マップデータ
- **cached_data**
  - fvのmp4ファイルが既に作成されているかどうかを示すフラグ．既に作成されているならば”1”が入り，作成中ならば”0”が入る.
- **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。

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-xxxxxxxxxxxxx.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>
```

管理者用指定順のfvの取得

指定順の fv を取得します。
管理者用の API で、表示対象になるのは、指定順の fv の取得 の API と異なりアクティベートされていないメディアも含まれます。公開前に投稿した動画の状態を確認することができます。

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

> **ノート**
>
> fv を構成するメディアの並び順を指定する方法は 2 種類あります。
> 各指定方法によって必須パラメータは下記のように変わります。タイル種別やタイル品質が異なるメディアID を混在して指定することはできません。
> 1. media_id と level を紐づけてそれぞれカンマ区切りで指定する方法 [詳細]
> → 必須パラメータ：media_ids, levels または level
> 2. JSON 形式のマップを指定する方法 [詳細]
> → 必須パラメータ：map_data

Parameters

- **output_type**
  - Response 形式
    - x : XML 形式で出力
    - 指定無し : JSON 形式で出力
- **duration**
  - 生成する fv の時間（1〜900秒）
    整数で指定。[詳細]
- **loop**
  - fv の長さを決定するパラメータ [詳細]
- **row** [必須]
  - 生成する fv の行数
- **col** [必須]
  - 生成する fv の列数
- **media_ids**
  - fv を構成するメディアID
- **levels**
  - media_ids で指定した各メディアID の level を指定。
- **level**
  - media_ids で指定した各メディアID の level を一括指定。
- **map_data**
  - JSON 形式のマップデータ

- **sound_ids**
  - fv に埋め込む音声をメディアID で指定。[詳細]

- **tile_type**
  - タイル種別
    hd 16:9 / sd 4:3 / r 3:2 / s 1:1
  - 本パラメータが指定されない場合で、かつ指定するmedia_idの中に異なるタイルタイプを持つタイルが存在する場合はエラーが返ります。
  - ※ tile_type は、map_data 指定での fv 生成時で、すべて search 指定かつ default でのメディアID指定がない場合必要になります。

- **tile_quality**
  - タイル品質
    1 高 / 2 中 / 3 低
  - 本パラメータが指定されない場合には、指定するmedia_idの中で最初に指定されるmedia_idの品質が適用され、そのタイル品質で生成されていないタイルは、黒色のタイルで埋められます。
  - ※ tile_quality は、map_data 指定での fv 生成時で、すべて search 指定かつ default でのメディアID指定がない場合必要になります。

- **start_time**
  - media_ids で指定した各メディアの開始時間を指定 [詳細]

- **media_duration**
  - media_ids で指定した各メディアの時間を指定 [詳細]

Response

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

- **video_url**
  - 生成された fv のURL

- **thumbnail_url**
  - 生成された fv のサムネイルのURL

- **cached_data**
  - fvのmp4ファイルが既に作成されているかどうかを示すフラグ。既に作成されているならば”1”が入り、本リクエストをで作成されたmp4ファイルならば”0”が返る。

- **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**
  - マップデータ

Request Example

media_ids と levels で指定

```
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指定

```
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>
```

メディア管理API

メディア情報一覧の取得

メディア情報一覧を取得します。

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

> **ノート**
>
> この API は、認証によって挙動が異なります。
> Digest 認証に成功すると、パラメータに合致し、“削除されていない”メディアの一覧を返します。
> Referer+Key での認証が成功すると、パラメータに合致し、ステータスが“有効になっている”メディアの一覧を返します。

Parameters

- **output_type**
  - Response 形式
    - x : XML 形式で出力
    - 指定無し : JSON 形式で出力
- **page** [必須]
  - 表示するページ位置。1 以上の整数。[詳細]
- **num** [必須]
  - 1 ページあたりの件数。[詳細]
    正の値を入力すると、昇順で出力
    負の値を入力すると、降順で出力
- **tile_type**

- 一覧取得するタイル種別
  tile_type の指定がない場合は全タイル種別が適用
  hd 16:9 / sd 4:3 / r 3:2 / s 1:1

- **tile_quality**
  - 一覧取得するタイル品質
    tile_quality の指定がない場合は全タイル品質が適用
    品質は 1 / 2 / 3 から選択可能

- **level**
  - 一覧取得するタイルのレベル
    level の指定がない場合は全レベルのタイルが適用
    レベルは 1 / 2 / 3 / 4 から選択可能

- **media_id**
  - 一覧取得するタイルのID（部分一致）
    指定された数字をメディアIDに含むメディアがを対象とする.

Response

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

- **has_next**
  - 次ページの有無
    true : 有
    false : 無

- **total**
  - 総メディア数

- **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
      - サムネイルURLのルールは{メディアID}_{サムネイルの高さ}_{指定されたフレーム位置}.jpgとなる。また、本fvLIBRARY APIを使用して投稿されたメディアは全て30FPSのタイルに変換されます。そのため投稿したオリジナルメディアのフレーム位置とfabric videoであるmp4のフレーム位置は異なる可能性が有ります。
    - **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(配列)
        - **tile_quality**
            - タイル品質(配列)
        - **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/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>
```

メディアのメタデータの取得

メディアのメタデータを取得します。

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

Parameters

- **output_type**
  - Response 形式
    - x : XML 形式で出力
    - 指定無し : JSON 形式で出力

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/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>
```

メディアのメタデータの更新

メディアのメタデータを更新します。

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

Parametres

- **output_type**
  - Response 形式
    - x : XML 形式で出力
    - 指定無し : JSON 形式で出力
- **caption**
  - キャプション
- **description**
  - ディスクリプション
- **tags**
  - タグ

Response

- **status**
  - 処理結果
    success : 成功
    それ以外 : 失敗
- **caption**
  - 登録されたキャプション
- **description**
  - 登録されたディスクリプション
- **tags**
  - 登録されたタグ

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>
```

メディアの有効化

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/{media_id}/activate
```

Parametres

- **output_type**
  - Response 形式
    - x : XML 形式で出力
    - 指定無し : JSON 形式で出力

Response

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

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>
```

メディアの無効化

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/{media_id}/deactivate
```

Parametres

- **output_type**
  - Response 形式
    - x : XML 形式で出力
    - 指定無し : JSON 形式で出力

Response

- **status**
  - 処理結果
    success : 成功

それ以外 : 失敗

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>
```

メディアの削除

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。

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

Parametres

- **output_type**
  - Response 形式
    - x : XML 形式で出力
    - 指定無し : JSON 形式で出力

Response

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

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>
```

タイル生成状態の取得

タイルの生成状態を取得します。

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

Parametres

- **output_type**
  - Response 形式
    - x : XML 形式で出力
    - 指定無し : JSON 形式で出力

Response

- **status**
  - 処理結果
    success : 成功
- **tile_status**
  - タイルステータス
    1 : タイル生成済
    3 : タイル未生成
    4 : タイル生成失敗

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>
```

タイル情報一覧の取得

タイル情報の一覧を取得します。

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

Parameters

- **output_type**
  - Response 形式
    - x : XML 形式で出力
    - 指定無し : JSON 形式で出力
- **page** [必須]
  - 表示するページ位置。1 以上の整数。[詳細]
- **num** [必須]
  - 1 ページあたりの件数。[詳細]
- **tile_type**
  - 一覧取得するタイル種別
    tile_type の指定がない場合は全タイル種別が適用
    hd 16:9 / sd 4:3 / r 3:2 / s 1:1
- **level**
  - 一覧取得するタイルレベル。
    level の指定がない場合は全レベル（1〜4）が適用
    1 / 2 / 3 / 4
- **tile_quality**
  - 一覧取得するタイル品質。
    tile_qualityの指定がない場合は，全品質（1〜3）が適用
    1 / 2 / 3

Response

- **status**
  - 処理結果
    success : 成功
    それ以外 : 失敗
- **has_next**
  - 次ページの有無
    true : 有
    false : 無
- **total**
  - 総メディア数
- **info**
  - タイル情報
    - **media_id**
      - メディアID
    - **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 （配列）
    - **tile_quality**
      - タイル品質 （配列）
    - **create_date**
      - 作成日時

- **update_date**
  - 更新日時

- **status**
  - タイルのステータス
    0：無効
    1：有効
    3：タイル未生成状態
    4：タイル生成失敗

Request Example

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

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?page=1&num=10&tile_quality=2&level=2&tile_type=hd"
```

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

```
curl "https://example.com/api/1/videos/tile?page=1&num=10&tile_quality=2&level=2&tile_type=hd" \
-H "Authorization: Bearer {your access_token}"
```

Response Example - JSON


```
{
"status":"success",
"has_next":true,
"total":58,
"info":[
        {
            "media_id":10,
            "version":"3",
            "type":"s64",
            "fps":30,
            "frames":150,
            "level":[
                1,2
            ],
            "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":[
                1,2
            ],
```

```
                "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":[
                    1,3
                ],
                "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>
              <entry>1</entry>
            </level>
            <tile_quality>
                <entry>2</entry>
            </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>
              <entry>1</entry>
              <entry>2</entry>
```

```
        </level>
        <tile_quality>
          <entry>2</entry>
        </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>
          <entry>1</entry>
          <entry>3</entry>
        </level>
        <tile_quality>
          <entry>2</entry>
        </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>
```

キーワード検索の結果の取得

キーワード検索の結果を取得します。
タグは完全一致、キャプション/ディスクリプションは部分一致で検索されます。
またメディアIDによる検索も可能です。（部分一致）

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

> **ノート**
>
> このAPIは、認証の仕方によって、結果が異なります。
> Referrer + Key での認証の場合は、メディアがアクティブなもののみ、
> Digest 認証の場合は、削除されていないメディアが、キーワード検索の結果として返されます。

検索のルール

1. キャプション・ディスクリプションは部分一致検索

- **複数キーワードの設定について**
  - 複数の文字列を指定する場合、AND検索とOR検索が可能です。キーワード間はスペースで区切ってご指定下さい。
  - AND検索の場合は、複数の指定された対象文字列を全て含むメディアが返されます。
    - AND検索対象とする文字列は、ダブルクォーテーション”“で括って下さい。
  - OR検索の場合は、複数の指定された対象文字列の内、一つでも含むメディアが返されます。

2. タグ検索は完全一致検索

- タグ[abc]とタグ[abcd]は全く別のタグとして扱われます。

3. メディアID検索は部分一致検索

- メディアID検索は部分一致検索です。メディアID[5]を指定すると、メディアID[5, 15, 25, ,,, 51,,]など5を含む全てのメディアIDのメディアが返されます。

4. タグ検索やキャプション／ディスクリプション検索の同時指定

- タグ検索とキーワード／ディスクリプション検索は同時に指定することが可能です。
  メディアID検索は他のキーワードと同時に指定することができません。（想定している挙動を保証致しません。）
  指定するキーワードと返却されるメディアについての関係例は以下になります。

例

指定するキーワード	指定方法	返却される対象メディア
タグabcと文字列”def”	#abc “def”	タグabcを持ち,かつキャプション／ディスクリプションにdefを含むメディア
文字列”abc”と文字列”def”	“abc” “def”	キャプション／ディスクリプションにabcとdefの両方の文字列を含むメディア
文字列abcと文字列def	abc def	キャプション／ディスクリプションにabcまたはdefのどちらかの文字列を含むメディア
文字列”abc”と文字列def	“abc” def	キャプション／ディスクリプションにabcの文字列を含むメディア

Parameters

- **output_type**
  - Response 形式
    - x : XML 形式で出力
    - 指定無し : JSON 形式で出力
- **page** [必須]
  - 表示するページ位置。1 以上の整数。[詳細]
- **num** [必須]
  - 1 ページあたりの件数。[詳細]
- **keywords** [必須]
  - 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。
- **tile_type**
  - 一覧取得するタイル種別
    tile_type の指定がない場合は全タイル種別が適用
    hd 16:9 / sd 4:3 / r 3:2 / s 1:1
- **tile_quality**
  - 一覧取得するタイル品質
    tile_quality の指定がない場合は全タイル品質が適用
    品質は 1 / 2 / 3 から選択可能
- **level**
  - 一覧取得するタイルのレベル
    level の指定がない場合は全レベルのタイルが適用
    レベルは 1 / 2 / 3 / 4 から選択可能

Response

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

- 次ページの有無
  true : 有
  false : 無

- **total**
  - 総メディア数

- **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
      - サムネイルURLのルールは{メディアID}_{サムネイルの高さ}_{指定されたフレーム位置}.jpgとなる。また、本fvLIBRARY APIを使用して投稿されたメディアは全て30FPSのタイルに変換されます。そのため投稿したオリジナルメディアのフレーム位置とfabric videoであるmp4のフレーム位置は異なる可能性が有ります。
    - **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(配列)
        - **tile_quality**
          - タイル品質(配列)
        - **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/search?page=1&num=2&keywords=abc
```

Sample Curl Command

- **Basic認証を使用した方法(キャプション・ディスクリプション検索)**
  - キャプション・ディスクリプションから”abc”と”def”という文字列をOR検索
    結果はabcまたはdefという文字列のどちらかがキャプション・ディスクリプションに含まれる
    メディアが返される。

```
curl --basic -u {your account}:{your password} "https://example.com/api/1/videos/search?page=1&num=2&keywords=abc%20def"
```

  - キャプション・ディスクリプションから”abc”と”def”いう文字列をAND検索
    結果はabc及びdefという文字列の両方がキャプション・ディスクリプションに含まれる
    メディアが返される。

```
curl --basic -u {your account}:{your password} 'https://example.com/api/1/videos/search?page=1&num=2&keywords="abc"%20"def"'
```

  - タグabcとキャプション・ディスクリプションから”def”という文字列をAND検索
    結果はタグabcか付けられているメディアでかつdefという文字列がキャプション・ディスクリプションに含まれる
    メディアが返される。

```
curl --basic -u {your account}:{your password} 'https://example.com/api/1/videos/search?page=1&num=2&keywords=%23abc%20"def"'
```

- **AccessToken認証を使用した方法(タグ検索)**

```
curl --basic -u {your account}:{your password} "https://example.com/api/1/videos/search?page=1&num=2&keywords=%23abc" \
-H "Authorization: Bearer {your access_token}"
```

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>
```

契約情報の取得

現在契約中のプランの情報を取得します。

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

Parametres

- **output_type**
  - Response 形式
    - x : XML 形式で出力
    - 指定無し : JSON 形式で出力

Response

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

- **info**
  - 契約情報
    - **plan_name**
      - 契約しているプラン名称
    - **monthly_fee**
      - 毎月の支払額 (円)

- **disk_usage**
  - 現在使用中のディスク容量 (Byte)
- **max_disk_usage**
  - 最大使用可能ディスク容量 (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>
```

メディア登録処理のキャンセル

メディアの登録処理を途中で終了させます。
このAPIを実行すると、登録時に生成していたタイルデータなどのすべてのデータが削除されます。

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

Parametres

- **output_type**
  - Response 形式
    - x : XML 形式で出力
    - 指定無し : JSON 形式で出力

Response

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

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"
   ]
}
```


参考情報

プランによるAPI仕様相違点

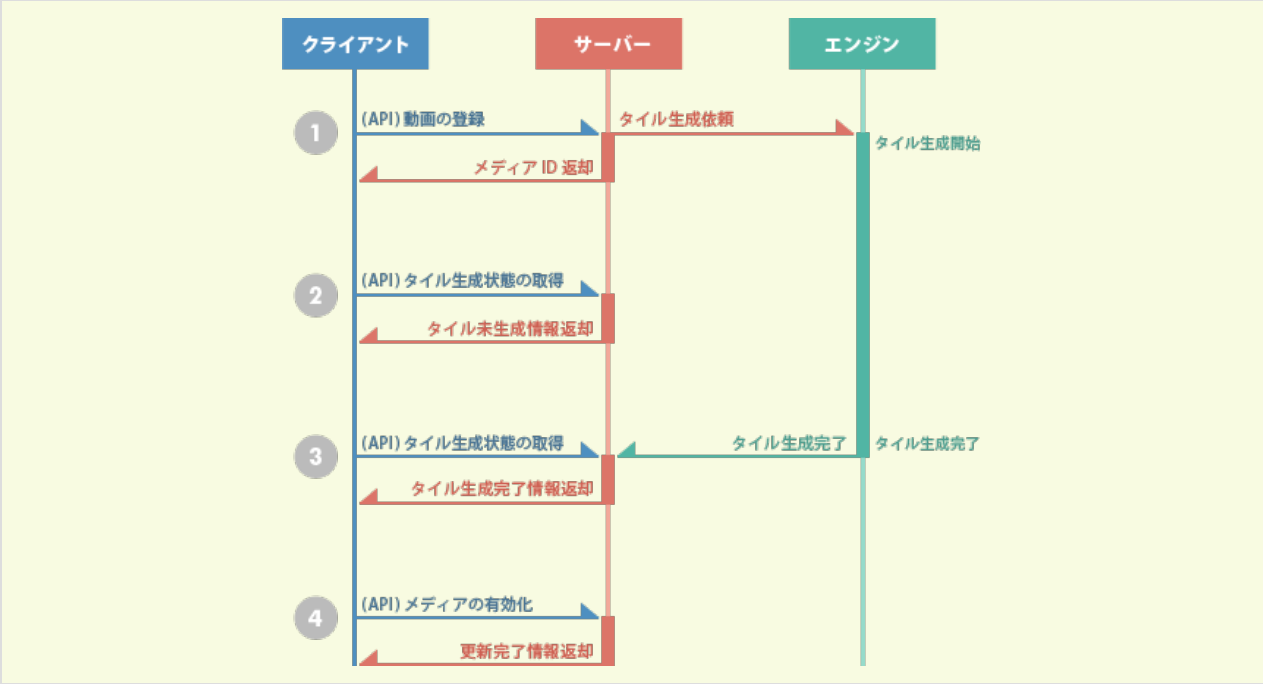
契約するプランにより、APIの仕様に相違があります。

相違点は下表の通りです。 基本的にはアップロードサイズ制限しか違いを記載出来ませんが、 fvLIBRARY稼働サーバのスペックが変わるため、パフォーマンスに 影響致します。 具体的には、投稿時のエンコードスピードが影響しますが、fvの 生成時間は大差ございません。

対象	プラン	
	ミニ	ライト
生成タイルの最大時間長（秒）	300	900
生成および指定可能レベル	1,2,3,4	1,2,3,4
生成fvの最大時間長（秒）	300	900
アップロードファイルサイズ制限(MB)	100	200

動画登録〜fvまでの使用手順

動画の登録から fv で使用するための手順



1 の動画の登録の段階ではタイルが fv で使える状態にはなっていません。
4 の メディアの有効化 API によって初めてタイルがアクティブになり、fv 生成で使える状態になります。

動画登録時の動画サイズと出力サイズ

動画登録時の動画サイズと出力サイズに対する仕様は以下のようになっています。アスペクト比が変更されたり、入力動画が切り取られたりすることはありません。

サイズが 入力動画 < 出力動画 の場合

入力動画のサイズが出力サイズより小さい場合は、入力動画を引き延ばすことなく出力サイズとの差分を黒色で塗りつぶします。

サイズが 入力動画 > 出力動画 の場合

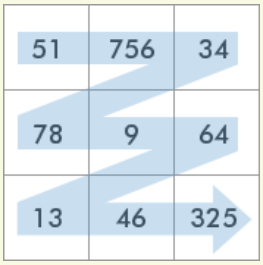
入力動画のサイズが出力サイズより大きい場合は、入力動画が出力動画に入りきるよう縮小して、出力サイズとの差分を黒色で塗りつぶします。

media_idとlevelの指定方法

media_idとlevelを紐づけてそれぞれカンマ区切りで指定する方法

メディアID を指定した場合、メディアID の指定をした場合、左上から右下に向かって順に並んでいくことになります。

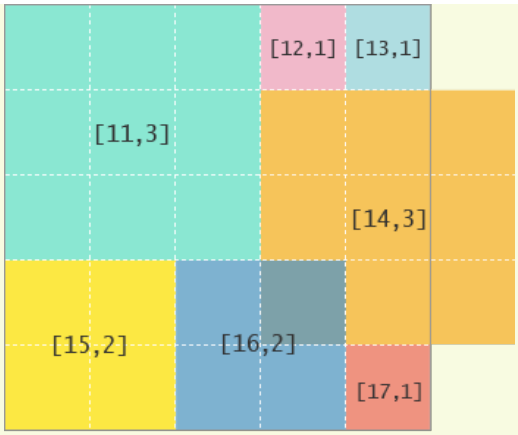
```
media_id: 51,756,34,78,9,64,13,46,325
```



複数レベルを使用したfvの場合

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

下図のように、左上から順番に、fvサイズ（枠）に対して指定されたメディアのタイル（パズルのピース）をはめ込んでいくような形になります。

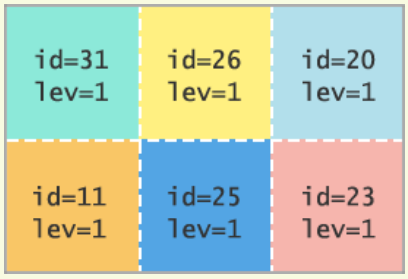


Level1の 5×5 マトリックス
[media_id , level]

- fvのサイズを超えた分のメディア（ID=18,19）は使用されません。
- メディア14のように枠内に入り切らない部分は表示されません。
- メディア14と16のように重なる部分がある場合は、後に指定したメディアが上になります。
- メディアの指定数が少なく、fv のサイズに対してタイルが足りない（fvがタイルで埋まらない）場合、黒いタイル（メディアID=0 ）が自動的に埋め込まれます。ただし、メディアID=0 のタイルを API で指定することはできません。

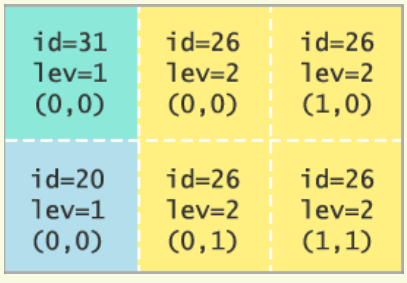
JSON形式のマップ指定方法

[例1] JSON形式のマップデータ：レベルが同じ場合




```
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" }
  ]
]
```

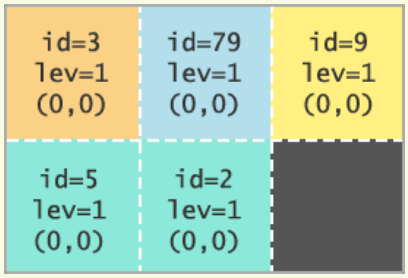
[例2] JSON形式のマップデータ：レベルが異なる場合



```
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" }
  ],
]
```

マップで指定する場合、fvのサイズに対してタイルが足りない場合はエラーとなります。

[例3] JSON形式のマップデータ：タグでの指定



greenタグと紐づいているIDは [2,5] 、yellowタグと紐づいているIDは [9] の場合

```
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"}}
  ]
]
```

以上のマップを元に fv を生成すると、

- 1行目の1列目は、testxというタグが存在しないので、default で指定されている id が [3] のものが表示されます。
- 1行目の2列目は、id 指定なので、その id が [79] のものを表示します。
- 1行目の3列目は、yellow のタグ指定なので、id が [9] のものを表示します。
- 2行目の1列目、2列目は共に green タグを指定したものが存在するので、1列目に id が [5]、2列目に id が [2] のものが表示されます。（最新のものから順番に表示されていきます。）
- 2行目の3列目は、yellow タグが指定されていますが、表示するものがないので、黒い動画が表示されます。

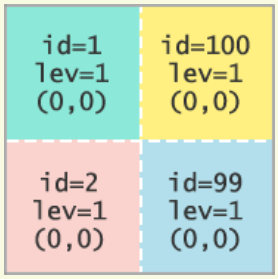
[例4] JSON形式のマップデータ：タイムラインでの指定

メディア ID が 100 まで存在している場合

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

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

上記のマップの場合、メディア ID の数が大きいほど最新のものになるので、 生成される fv の構成は以下となります。



[例5] JSON形式のマップデータ：fv 上での再生箇所の指定

マップ指定で時間軸結合を行う場合は、
{ “id”:”300”, “lev”:”1”, “rx”:”0”, “ry”:”0”, “st”:”6”,”dur”:”6”},
のような形で、st と dur の指定をし結合を行います。

- st には、そのメディアの開始位置
- dur には、そのメディアの開始位置からの再生時間

の指定をします。

```
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"}
  ]
]
```

この指定の場合は、st と dur の指定がされていない ID:1 は既存の再生の仕方と同様で,メディアID:2 は、そのメディアの6秒目から4秒間動画が再生されます。ID:3,4 は、そのメディアの6秒目から、6秒間と8秒間動画が再生されます。

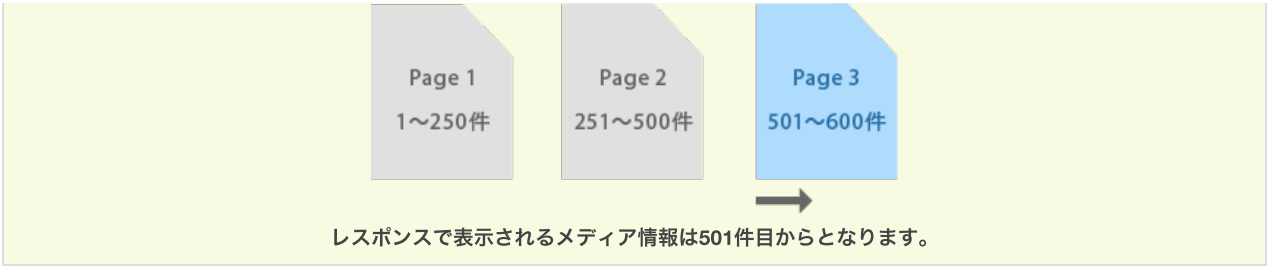
fv 全体の長さが、dur で指定した長さ以上の場合は、st で指定した地点から dur で指定した秒数の間の動画を繰り返し再生します。

page と num について

page（ページ）とnum（ 1ページあたりの件数）を使うと、多数のメディア情報などを取得する場合に、一度に表示する件数を制御することができます。

[例]

- 出力件数: 600件
- page: 3
- num: 250



fvへの音声指定

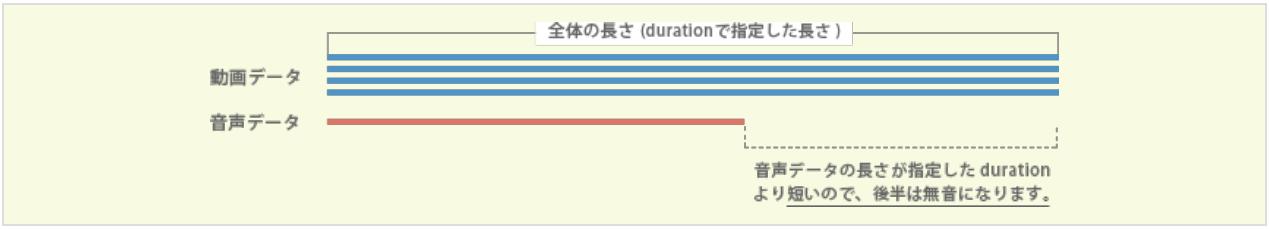
sound_idsにメディアIDを指定することで、そのメディアIDの音声をfvに含めることができます。音声データが存在しないメディアIDを指定することはできません。

合成できる音声の長さは、durationで指定した長さまでとなります。sound_idsで指定した音声ファイルの時間がdurationより長い場合、超過分は削除されます。逆に、音声ファイルよりdurationの方が長い場合、音声ファイルは繰り返されません。

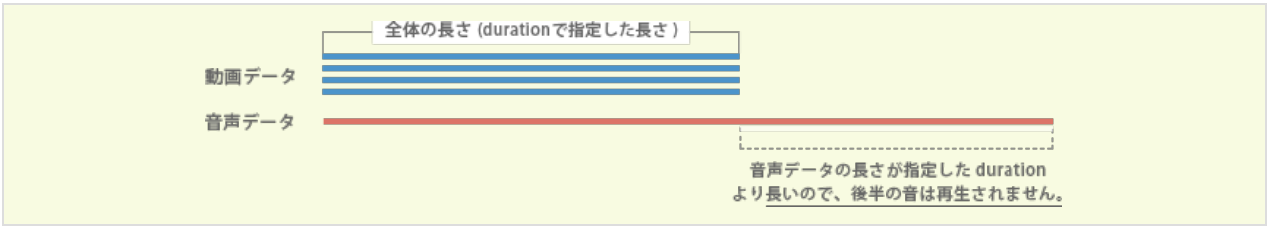
① 指定したsound_idsに該当するmedia_idがfvに含まれていない場合

この場合、音声ファイルよりdurationの方が長い場合、音声ファイルは繰り返されません。

2×2のfvを生成、音声データが短い場合



2×2のfvを生成、音声データが長い場合

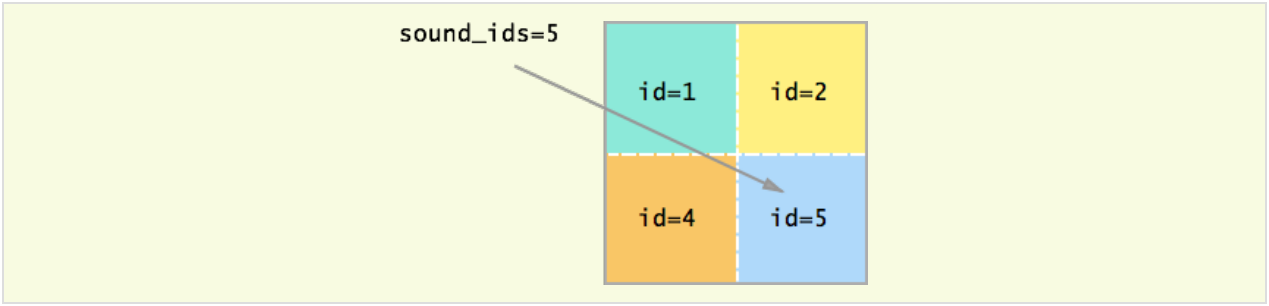


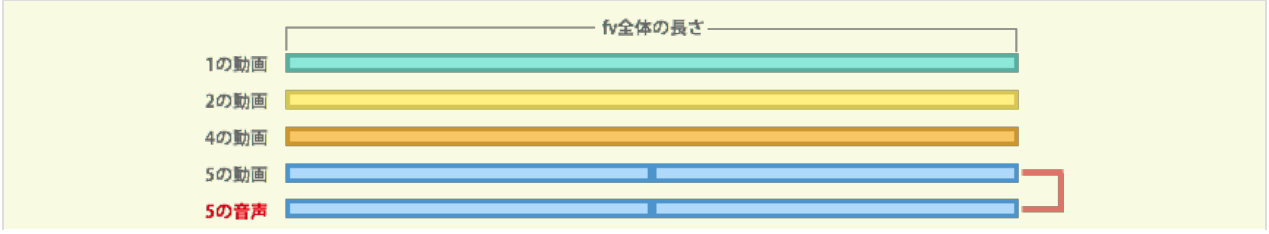
② 指定したsound_idsに該当するmedia_idがfvに含まれている場合

この場合、sound_idsに合致するタイルの、再生開始位置、再生時間に従って再生を行います。fvのdurationより短い場合は、ループします。

2 x 2のfv、レベル 1 のみの場合

fvに含まれるmedia_idが、1,2,4,5で、sound_idsに5が指定されている場合は以下の図のように音声が、media_idが５に設定されている再生方法に従って再生されます。

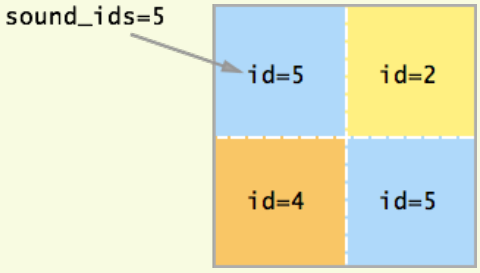


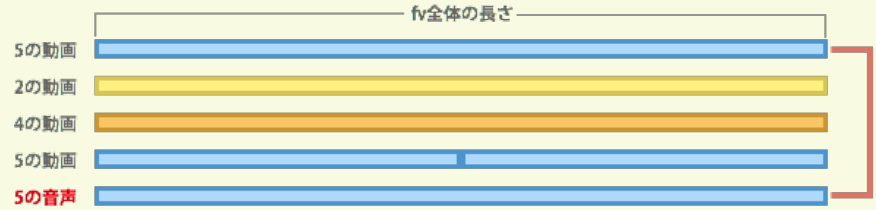


fv上での各メディアの再生パターン

2 x 2のfv、レベル１のみ、同じmedia_idが含まれる場合

fvに含まれるmedia_idが、5,2,4,5で、sound_idsに5が指定されている場合は、音声は、最も上で最も左に出現するmedia_idの再生方法に従います。この場合は以下のようになります。





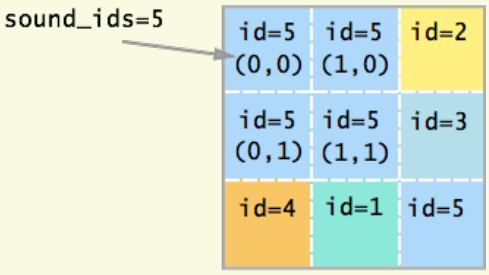
fv上での各メディアの再生パターン

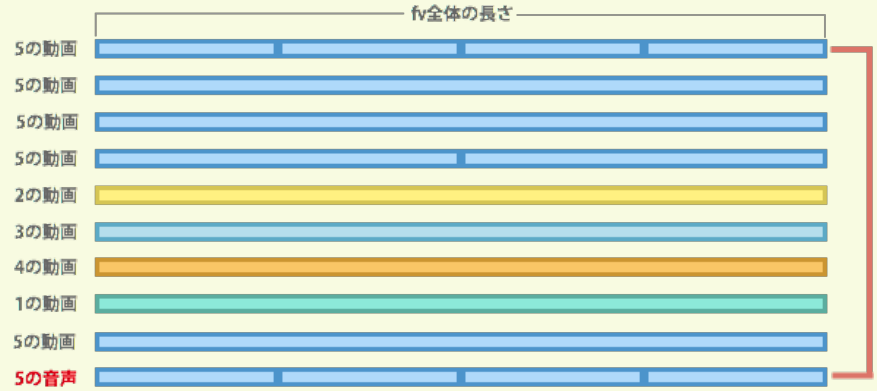
3 x 3のfv、レベル2のタイルが含まれる場合

レベル２の指定が行われていて、そのタイルのmedia_idを、sound_idsとして指定している場合、音声の再生方法は、必ずレベル２の一番左上のものになります。

マップの指定をしてfvを生成する場合にレベル２のタイル毎に再生方法を異ならせる場合[詳細]に注意する必要が有ります。

以下の図は、media_idが５のレベル２のタイルで、左上を４回ループするように、右下を2回ループするように指定しています。この場合、音声データは左上のタイルの再生方法に合わせた音声になります。





fv上での各メディアの再生パターン

durationの決定方法

fv生成時のdurationの決定の仕方について、生成されるfvの長さはdurationとloopパラメーターで決定されます。 loopパラメータはデフォルト

値 1 が設定されます。fvの長さは以下の３通りになります。

例) 1×2 の fv を生成する場合

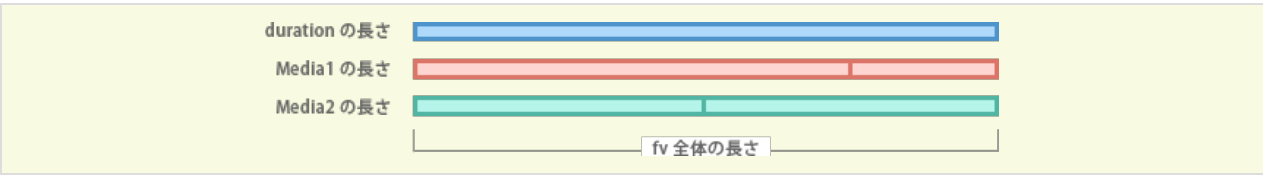
1×2 の fv を生成、duration, Media1, Media2 の各長さが以下だった場合

1. duration の設定なし

fvを構成するメディアの中の最大長のメディアの時間が、生成されるfvの長さになります。
一番長いメディアの時間以下のものは、繰り返し再生されます。

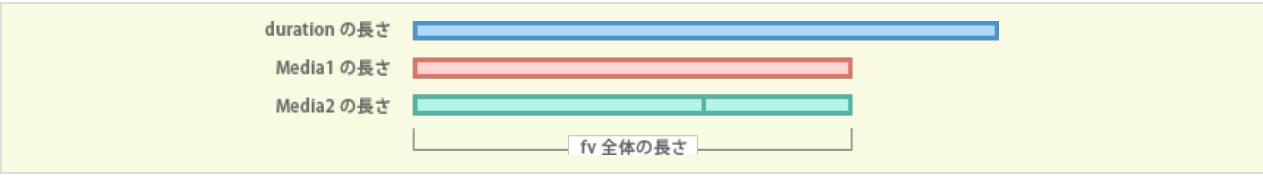
2. duration の設定あり、loop に 1 を設定

指定した duration が、生成されるfvの長さになります。
指定した duration 以下の長さメディアは、繰り返し再生されます。



3. duration の設定あり、loop に 0 を設定

fv を構成するメディアの中の最大長のメディアの時間が、生成される fv の長さになります。
一番長いメディアの時間以下のものは、繰り返し再生されます。



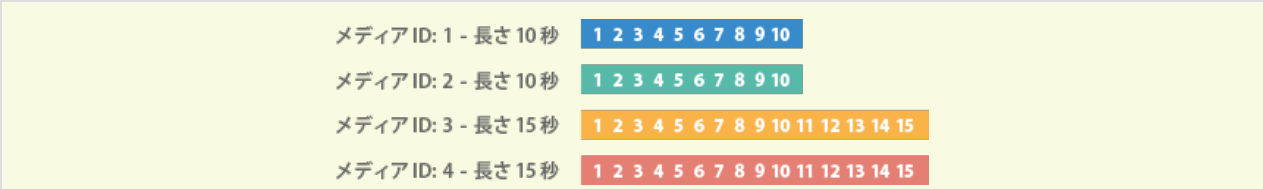
fv 生成時の start_time と media_duration の影響について

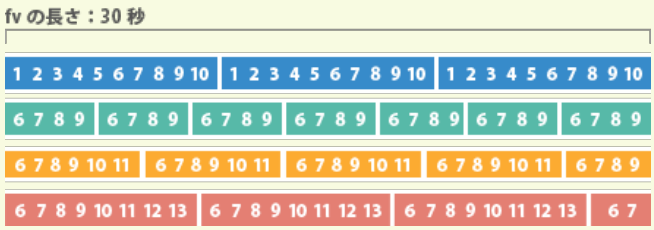
指定順の fv の取得の API で start_time と media_duration の指定をすることによって、個々のメディアの再生開始時間、再生時間を指定することが出来ます。

以下のマップデータの場合に、

```
[ [
        { "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"}
] ]
```

fv の duration に 30 秒が指定されている場合、fv の内容は、下記の画像の様な形になります。





生成される fv

全体の長さが 30 秒で、各 fv 上の長さが 30 秒に満たないので、start_time とmedia_duration で指定した箇所が、繰り返し再生されます。

タイル生成時の位置調整パラメータ

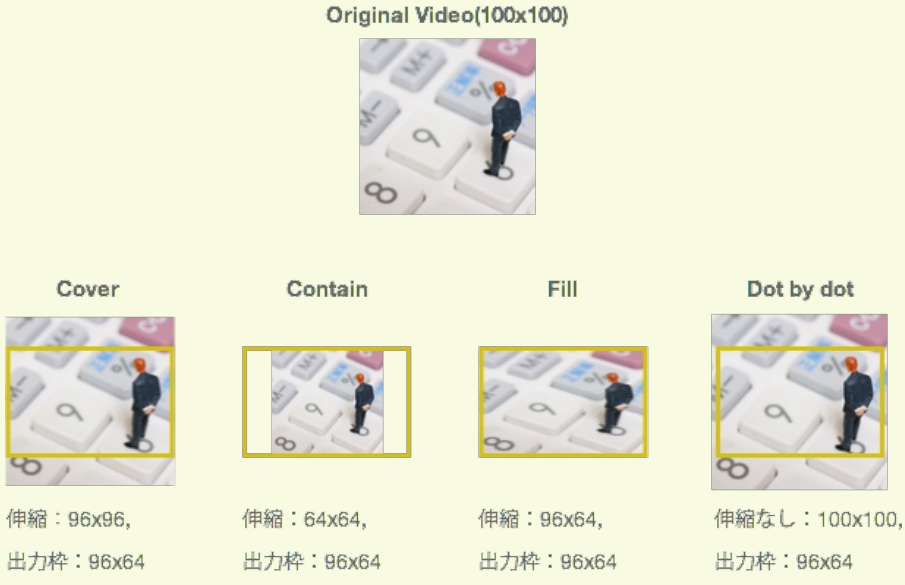
動画の登録時にcrop_position, padding_position, crop_inheriting, crop_policy_ob, crop_policy_ibの指定を行うことで、登録する動画をタイルに変換する際に表示する位置の調整を行うことが出来ます。各パラメータについては、動画の登録 の箇所を参照してください。

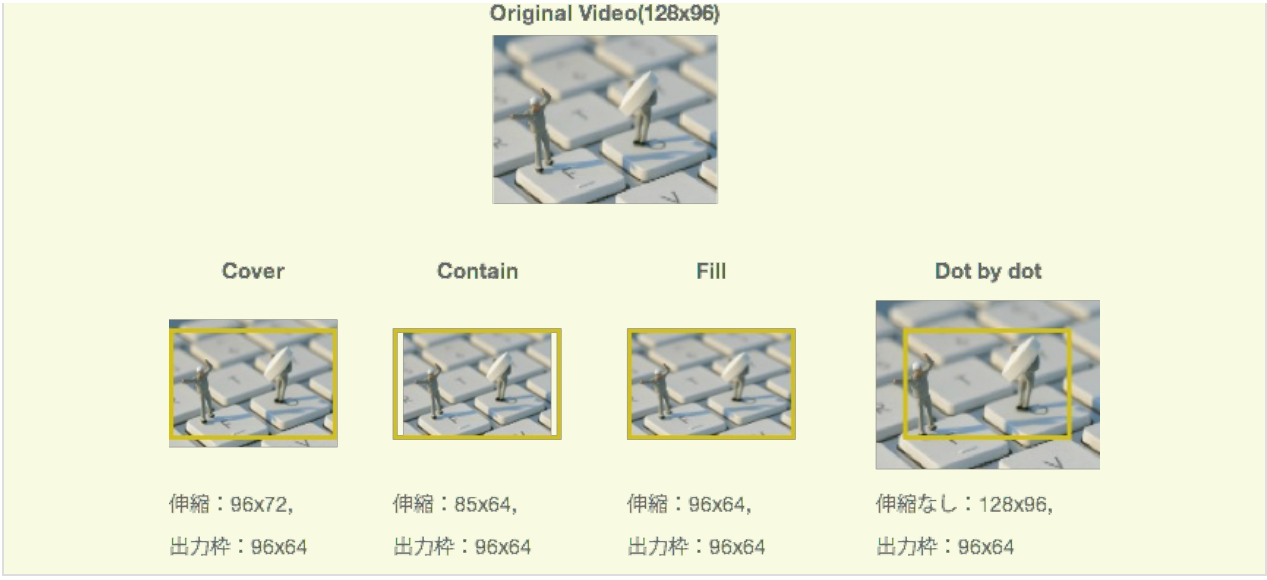
ポリシーの違いによる表示の違い

crop_policy_obとcrop_policy_ibは、登録動画と出力領域であるタイルのサイズを比較し、そのサイズの差分をどのように埋めるかの指定を行います。タイル生成はレベル毎に、大きなレベルから小さいレベルへ順に行われますが、この時、各レベル毎の出力サイズと登録動画のサイズを比較し、その大小関係によって、crop_policy_obとcrop_policy_ibのいずれかを適用して伸縮処理を行います。

以下に、例として、下記の条件で動画を登録した場合の表示の違いを記載します。
※ 図の中のオレンジの枠がタイルのサイズで、枠内以外は表示されません。
条件

種類	属性	値
入力	登録動画サイズ	1. 100x100 (アスペクト比 1:1) 2. 128x96 (アスペクト比 4:3)
出力	TileType Level	R (アスペクト比 3:2) 1 (96x64)
Policy	crop_position	1: [中央]
	padding_position	1: [中央]
	crop_inheriting (複数の出力サイズに対して、 タイルを同時生成する場合のポリシー)	1: [有効表示範囲をレベル毎に最適化]



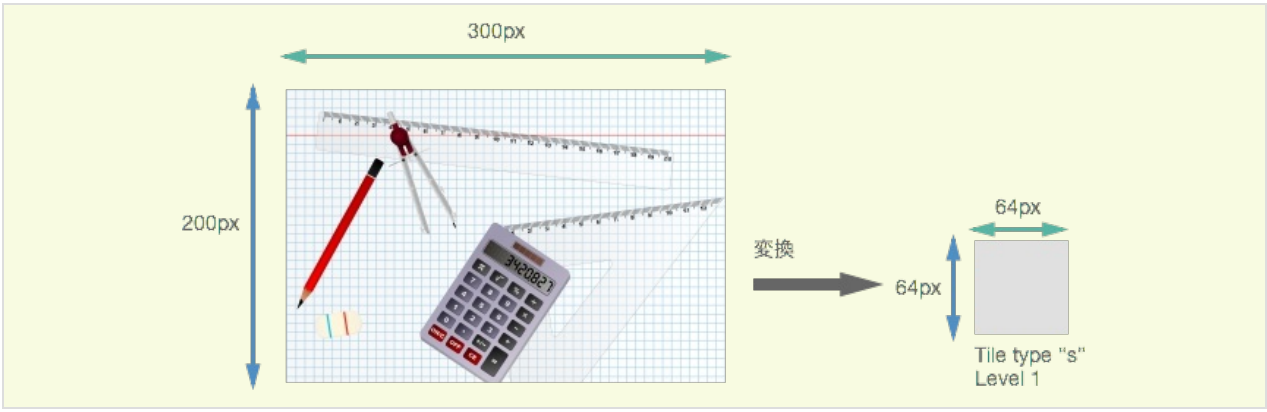


CoverとContainの違いについて

クロップする際のポリシーで、CoverとContainが指定されている場合の違いについて図を使用して説明します。両方ともアスペクト比を保ったままリサイズをしますが、

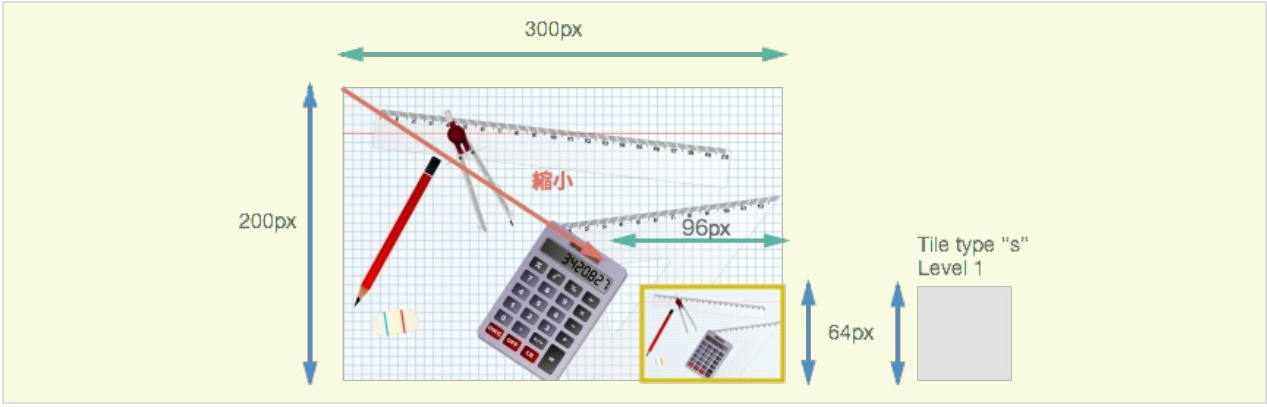
Coverの方は、登録メディアが出力領域を完全に覆うようにリサイズし、はみ出した部分を切り捨てます。

Containの場合は、登録メディア全体が出力領域に収まるようにリサイズします。登録動画と出力領域のアスペクト比が異なる場合、出力領域の余白が発生しますが、該当領域は、黒の画素で埋められます。

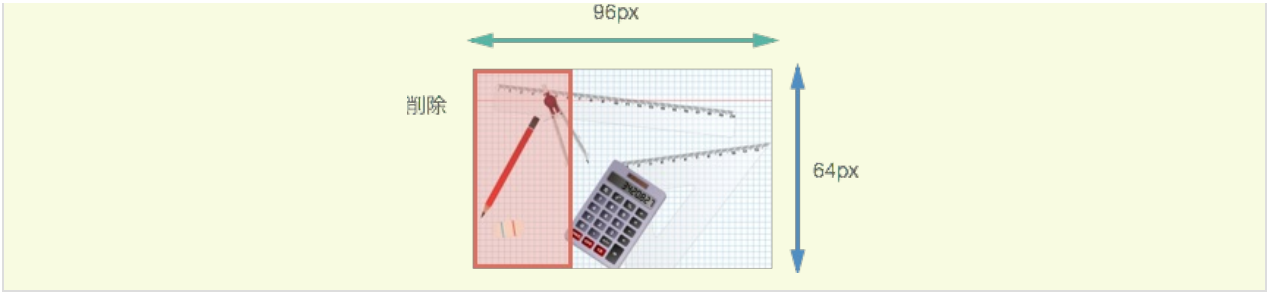


Coverの場合(右上寄せ)

1. 短い方の１辺が同じ長さになるまで縮小

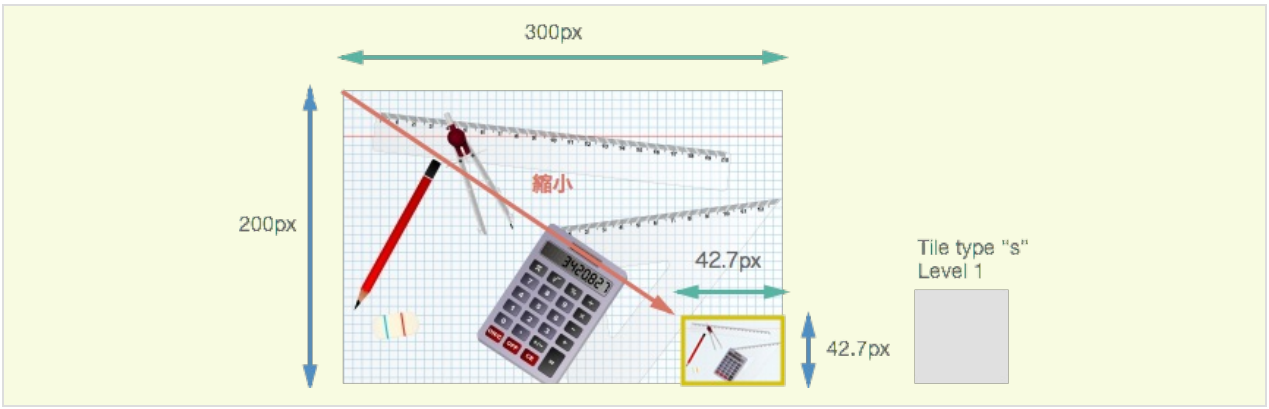


2. 出力タイルより大きい部分を削除する

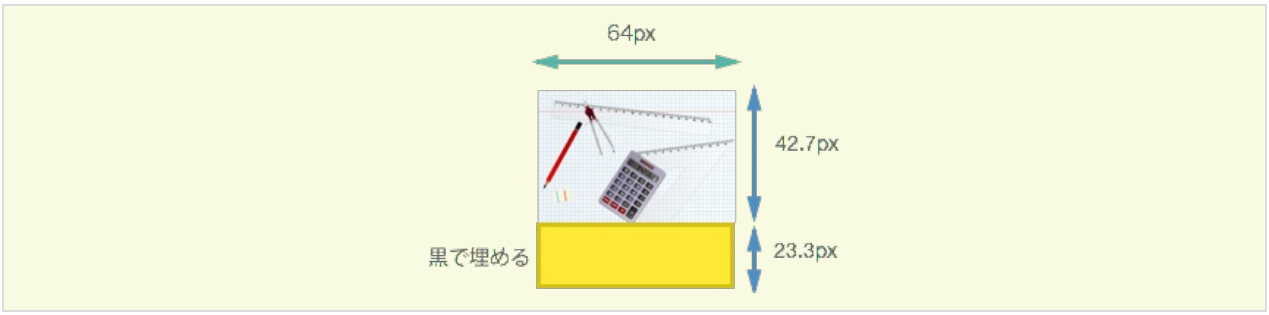


Containの場合(右上寄せ)

1. 長い方の 1 辺が同じ長さになるまで縮小



2. 出力タイルの空きエリアを黒で埋める



[例 1] パラメータを指定しない場合

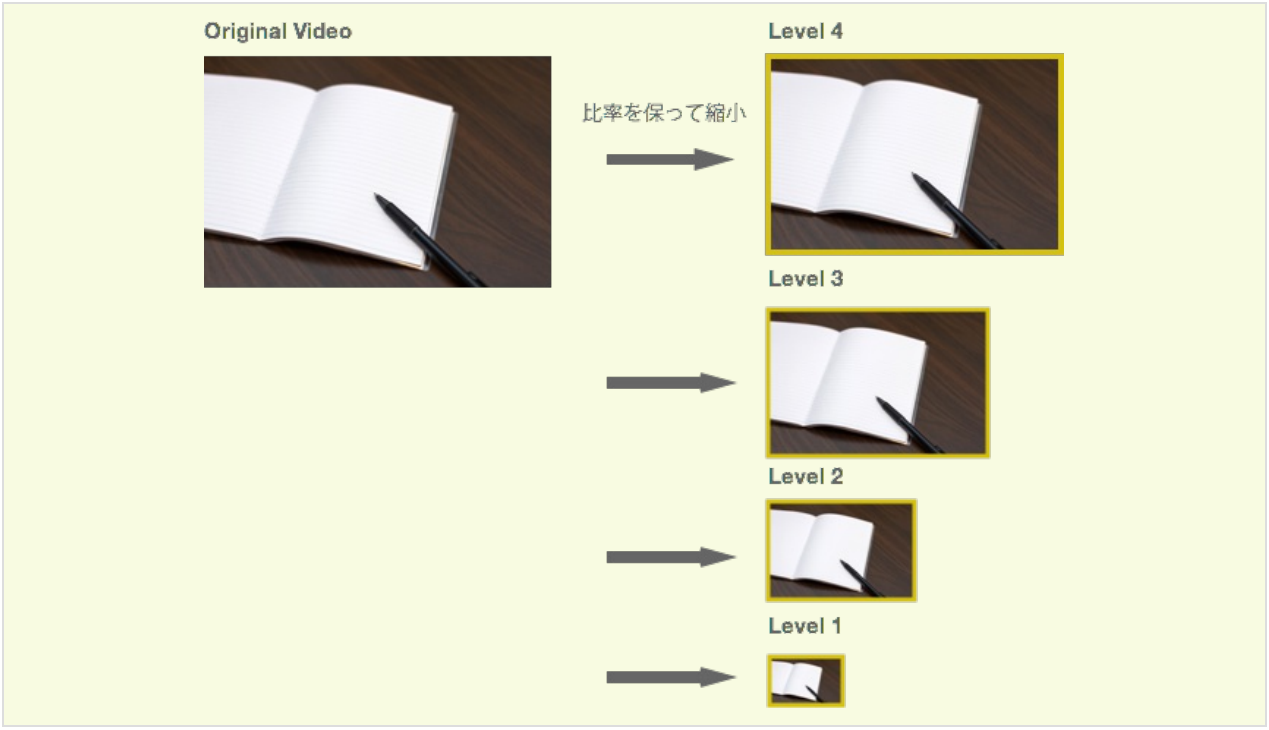
(タイル種別:HD レベル:1-4　登録動画サイズ:1920x1080)
パラメータを指定しないと、デフォルトの値である 1 が上記パラメータに設定されます。
条件

種別	属性	値
入力	登録動画サイズ	1920x1080 (アスペクト比 16:9)
出力	TileType	HD (アスペクト比 16:9)
	Level	4,3,2,1 (解像度1024x576, 768x432, 512x288, 256x144)
Policy	crop_position	指定無の場合、デフォルト値1: [中央]
	padding_position	指定無の場合、デフォルト値1: [中央]
	crop_inheriting （複数の出力サイズに対して、タイル を同時生成する場合のポリシー）	指定無の場合、デフォルト値1: [有効表示範囲をレベル毎に最適化]
	crop_policy_ob （出力サイズが、入力サイズより 大きい場合の伸縮ポリシー）	指定無の場合、デフォルト値1: [Contain] アスペクト比を保ってリサイズ （リサイズ後登録動画 ⊂ 出力領域）
	crop_policy_ib （入力サイズが、出力サイズより 大きい場合の伸縮ポリシー）	指定無の場合、デフォルト値1: [Contain] アスペクト比を保ってリサイズ （リサイズ後登録動画 ⊂ 出力領域）

- 1024 x 576 (HD Level 4) ＜ 1920 x 1080 (登録動画サイズ)

- 768 x 432 (HD Level 3) ＜ 1920 x 1080
- 512 x 288 (HD Level 2) ＜ 1920 x 1080
- 256 x 144 (HD Level 1) ＜ 1920 x 1080

この場合、レベル4,3,2,1のタイルを順に生成し、登録動画は全てのレベルの出力サイズより大きいので、全てのレベルのタイル生成でcrop_policy_ibが使用されます。この場合、crop_policy_ibは1 (contain)なので、アスペクト比を保って、登録動画が出力領域が完全に包含されるようにリサイズされますが、登録動画と出力領域のアスペクト比が同じであるため、出力領域に空白は発生しません。



[例 2] パラメータを指定しない場合

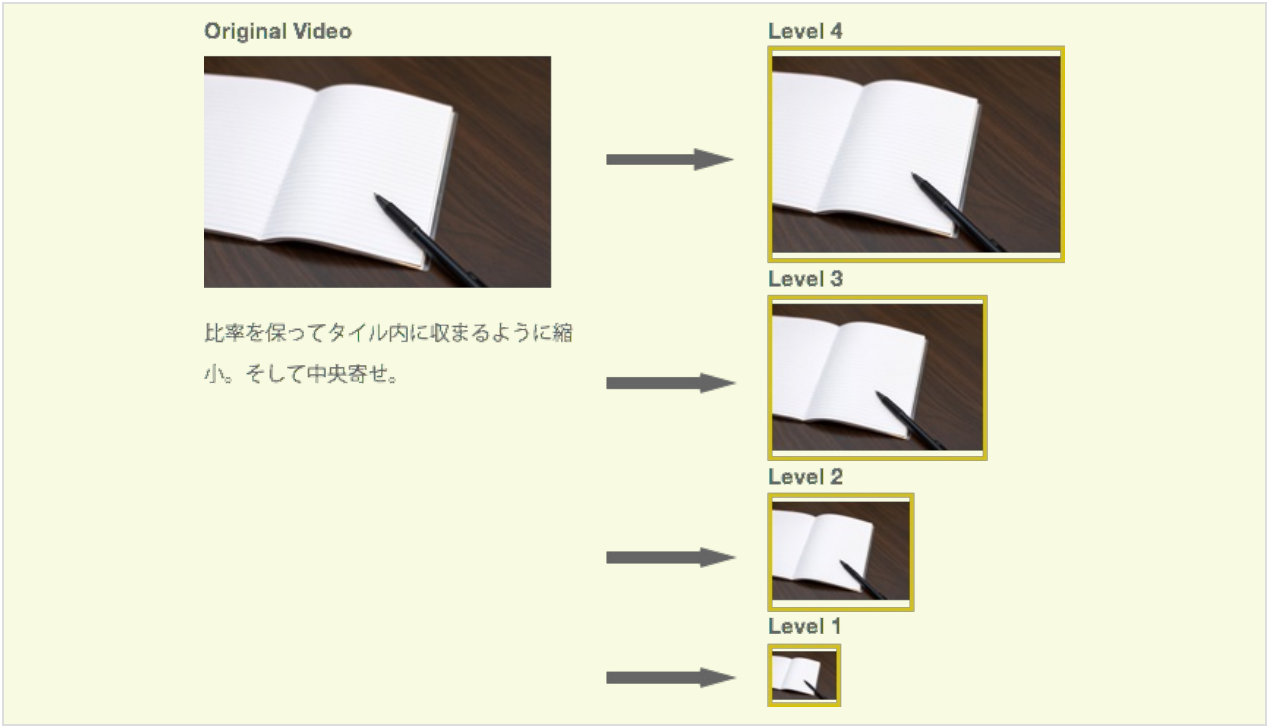
(タイル種別:SD レベル:1-4　登録動画サイズ:1920x1080)
タイル種別がSDである以外は例 1 と同様
条件

種別	属性	値
入力	登録動画サイズ	1920x1080 (アスペクト比 16:9)
出力	TileType	SD (アスペクト比 4:3)
	Level	4,3,2,1 (解像度512x384, 384x288, 256x192, 128x96)
Policy	crop_position	指定無の場合、デフォルト値1: [中央]
	padding_position	指定無の場合、デフォルト値1: [中央]
	crop_inheriting （複数の出力サイズに対して、タイル を同時生成する場合のポリシー）	指定無の場合、デフォルト値1: [有効表示範囲をレベル毎に最適化]
	crop_policy_ob （出力サイズが、入力サイズより 大きい場合の伸縮ポリシー）	指定無の場合、デフォルト値1: [Contain] アスペクト比を保ってリサイズ （リサイズ後登録動画 ⊂ 出力領域）
	crop_policy_ib （入力サイズが、出力サイズより 大きい場合の伸縮ポリシー）	指定無の場合、デフォルト値1: [Contain] アスペクト比を保ってリサイズ （リサイズ後登録動画 ⊂ 出力領域）

- 512 x 384 (SD Level 4) ＜ 1920 x 1080 (登録動画サイズ)
- 384 x 288 (SD Level 3) ＜ 1920 x 1080
- 256 x 192 (SD Level 2) ＜ 1920 x 1080
- 128 x 96 (SD Level 1) ＜ 1920 x 1080

この場合、レベル4,3,2,1のタイルを順に生成し、登録動画が全てのレベルのサイズより大きいので、crop_policy_ibが全てのレベルのタイルで使用されます。crop_policy_ibは 1 (contain)なので、アスペクト比を保って、登録動画が出力領域が完全に包含されるようにリサイズされます。この例では、登録動画と出力領域のアスペクト比が異なるため、出力領域に一部空白が生じ、黒画素で埋められます。黒画素と有効画

像領域の配置について、crop_positionが 1 (中央)なので、以下のように、有効画像領域が中央に来るように配置されます。



[例 3] crop_position:3, padding_position2, crop_inheriting:0, crop_policy_ob:3, crop_policy_ib:3の場合

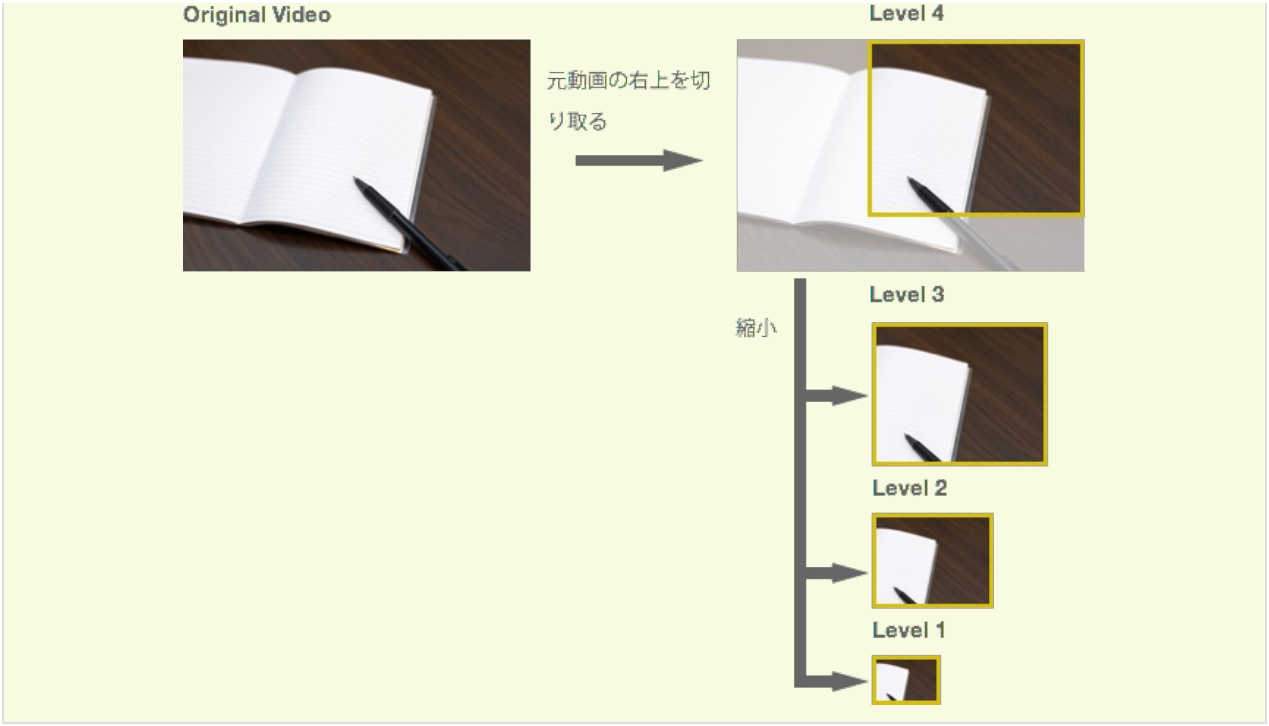
(タイル種別:r レベル:1-4　登録動画サイズ:1920x1080)

条件

種別	属性	値
入力	登録動画サイズ	1920x1080 (アスペクト比 16:9)
出力	TileType	R (アスペクト比 3:2)
	Level	4,3,2,1 (解像度384x256, 288x192, 192x128, 96x64)
Policy	crop_position	3: [右上]
	padding_position	2: [左上]
	crop_inheriting （複数の出力サイズに対して、タイル を同時生成する場合のポリシー）	0: [有効表示範囲をレベル間で統一]
	crop_policy_ob （出力サイズが、入力サイズより 大きい場合の伸縮ポリシー）	3: [Dot by dot] 単純なパディング
	crop_policy_ib （入力サイズが、出力サイズより 大きい場合の伸縮ポリシー）	3: [Dot by dot] 単純なクロップ

- 384 x 256 (R Level 4) ＜ 1920 x 1080 (登録動画サイズ)
- 288 x 192 (R Level 3) ＜ 1920 x 1080
- 192 x 128 (R Level 2) ＜ 1920 x 1080
- 96 x 64 (R Level 1) ＜ 1920 x 1080

この場合、レベル4,3,2,1のタイルを順に生成し、登録動画が全てのレベルのサイズより大きいので、crop_policy_ib:3（Dot by dot）が使用され、また、crop_inheriting:0（有効表示範囲をレベル間で統一）なので、最初に生成するレベル４のタイルをレベル毎の出力サイズに順次縮小する形で各レベルのタイルを生成します。各レベルで、画像サイズは異なりますが、画像内容は同じになります。最初のタイル生成時、crop_position:3 (右上) の指定に従って、登録動画の右上部分を切り取ります。



[例 4] crop_position:3, padding_position2, crop_inheriting:1, crop_policy_ob:3, crop_policy_ib:0の場合

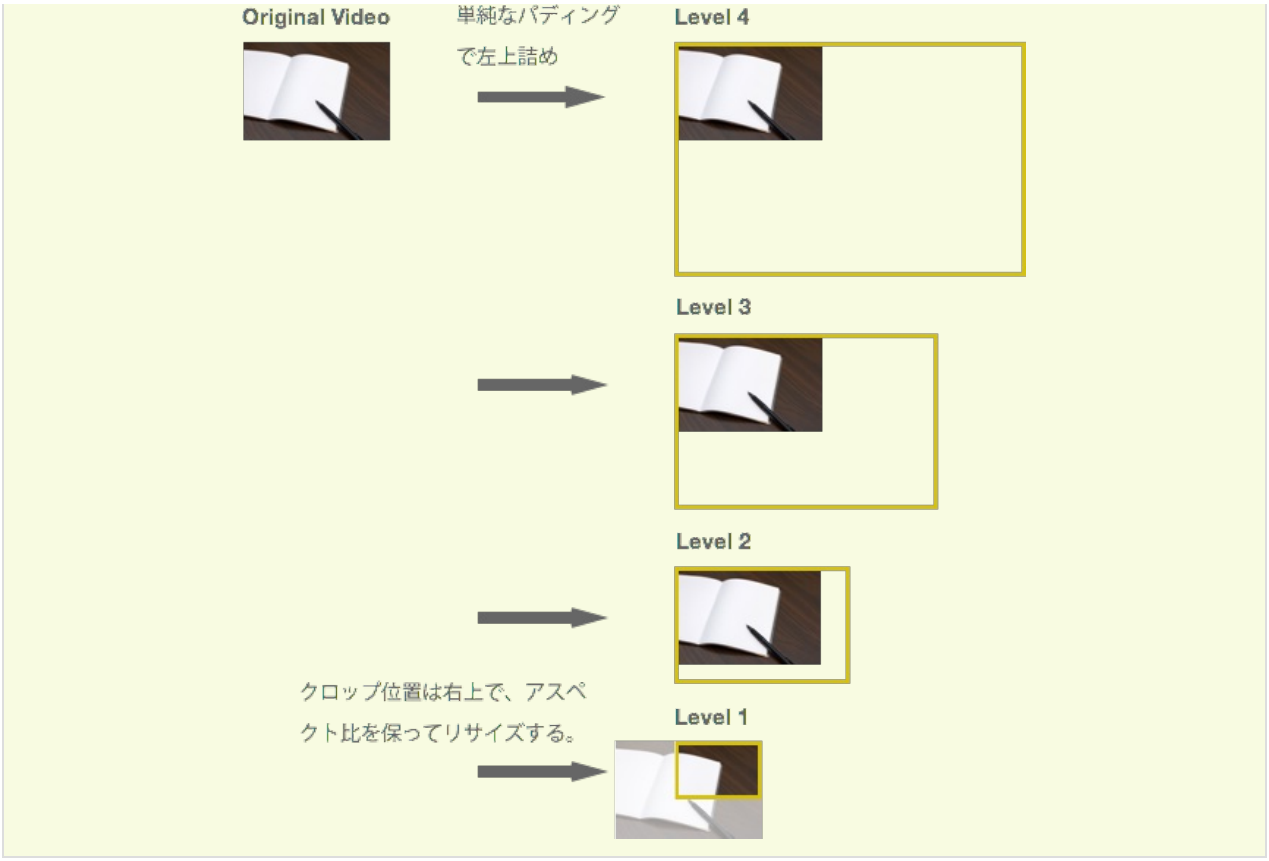
(タイル種別:r レベル:1-4　登録動画サイズ:128 x 64)

条件

種別	属性	値
入力	登録動画サイズ	128x64 (アスペクト比 2:1)
出力	TileType	R (アスペクト比 3:2)
	Level	4,3,2,1 (解像度384x256, 288x192, 192x128, 96x64)
Policy	crop_position	3: [右上]
	padding_position	2: [左上]
	crop_inheriting （複数の出力サイズに対して、タイル を同時生成する場合のポリシー）	1: [有効表示範囲をレベル毎に最適化]
	crop_policy_ob （出力サイズが、入力サイズより 大きい場合の伸縮ポリシー）	3: [Dot by dot] 単純なパディング
	crop_policy_ib （入力サイズが、出力サイズより 大きい場合の伸縮ポリシー）	0: [Cover] アスペクト比を保ってリサイズ （リサイズ後登録動画 ⊃ 出力領域）

- 384 x 256 (R Level 4) ＞ 128 x 64 (登録動画サイズ)
- 288 x 192 (R Level 3) ＞ 128 x 64
- 192 x 128 (R Level 2) ＞ 128 x 64
- 96 x 64 (R Level 1) ＜ 128 x 64

この場合、レベル4,3,2,1のタイルを順に生成し、レベル4,3,2のタイルに対しては、入力となる登録動画が出力サイズより小さいため、crop_policy_ob: 3（Dot by dot）が適用されます。登録動画は拡大縮小されず、padding_position:2（左上）の指定に従って、有出力領域の左上に配置され、空白領域は黒画素で埋められます。レベル 1 のタイルに対しては、入力となる登録動画が出力領域より大きいため、crop_policy_ib:0（Cover）が適用されます。この場合、アスペクト比を保ちながら登録動画が出力領域を完全に包含するように拡大します。入りきらない部分は捨てられます。



[例 5] crop_position:3, padding_position2, crop_inheriting:1, crop_policy_ob:0, crop_policy_ib:2の場合

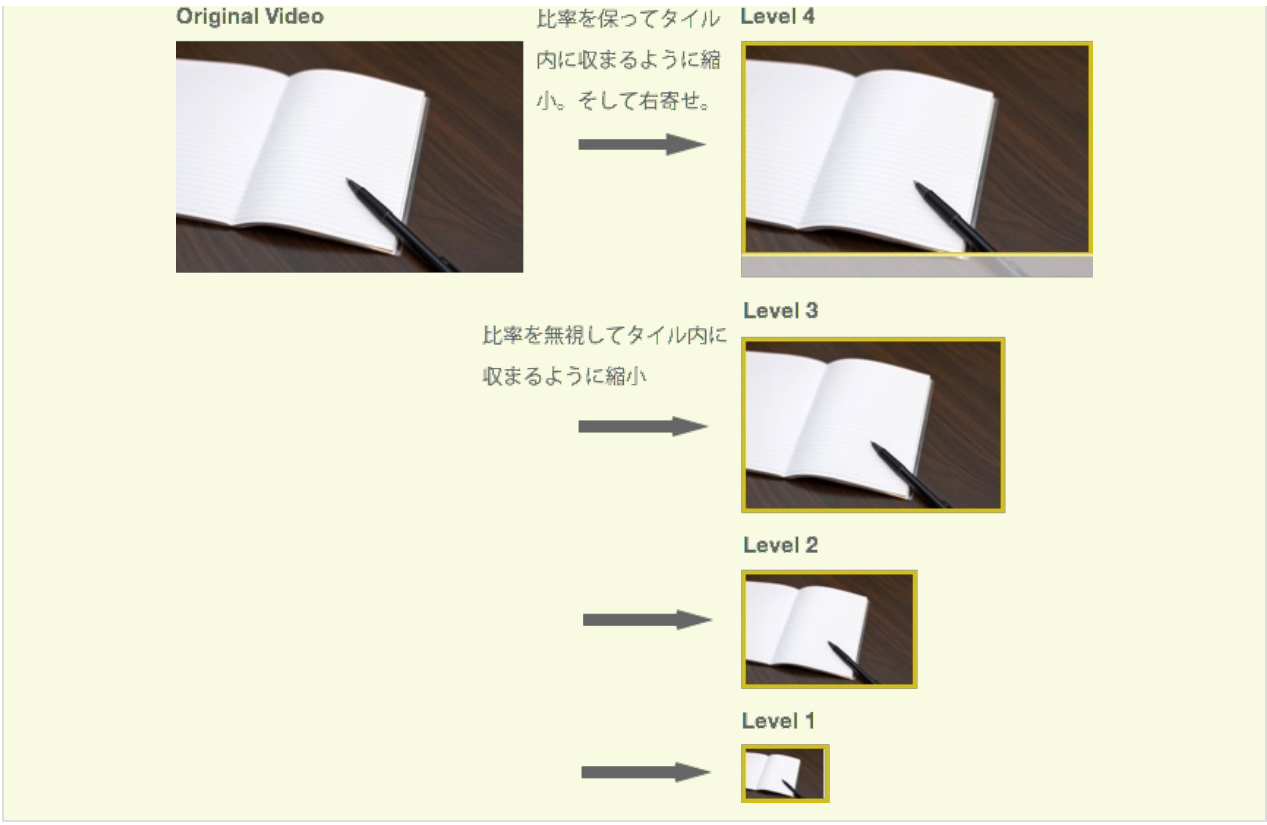
(タイル種別:r レベル:1-4　登録動画サイズ:320 x 240)

条件

種別	属性	値
入力	登録動画サイズ	320x240 (アスペクト比 4:3)
出力	TileType	R (アスペクト比 3:2)
	Level	4,3,2,1 (解像度384x256, 288x192, 192x128, 96x64)
Policy	crop_position	3: [右上]
	padding_position	2: [左上]
	crop_inheriting （複数の出力サイズに対して、タイル を同時生成する場合のポリシー）	1: [有効表示範囲をレベル毎に最適化]
	crop_policy_ob （出力サイズが、入力サイズより 大きい場合の伸縮ポリシー）	0: [Cover] アスペクト比を保ってリサイズ （リサイズ後登録動画 っ 出力領域）
	crop_policy_ib （入力サイズが、出力サイズより 大きい場合の伸縮ポリシー）	2: [fill] アスペクト比を崩してリサイズ

- 384 x 256 (R Level 4) ＞ 320 x 240 (登録動画サイズ)
- 288 x 192 (R Level 3) ＜ 320 x 240
- 192 x 128 (R Level 2) ＜ 320 x 240
- 96 x 64 (R Level 1) ＜ 320 x 240

この場合、レベル4,3,2,1のタイルを順に生成し、レベル４の場合のみ、登録動画のサイズが出力領域より小さいので、crop_policy_ob:0（Cover）が適用されます。この場合、この場合、アスペクト比を保ちながら登録動画が出力領域を完全に包含するように拡大します。入りきらない部分は捨てられます。crop_position:3（右上）に従って、出力領域の左側が捨てられます。 レベル3,2,1の場合、登録動画のサイズが出力動画より大きいので、crop_policy_ib:2（Fill）が適用されます。この場合、登録動画は元のアスペクト比を保たず、出力領域とぴったり合うように伸縮されます。



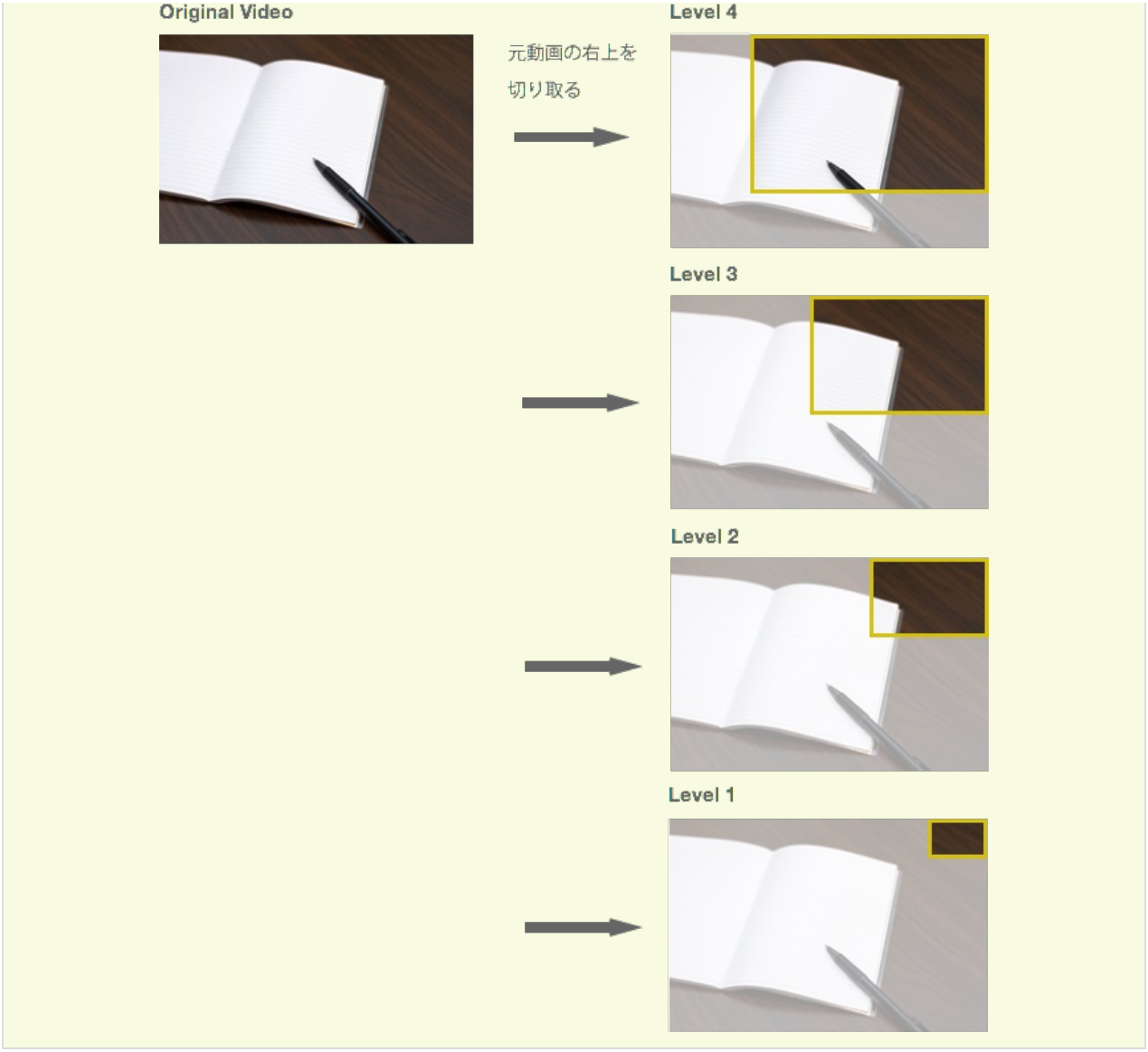
[例 6]

例３のcrop_inheritingを1にした場合の例になります。
(タイル種別:r レベル:1-4　登録動画サイズ:1920x1080)
条件

種別	属性	値
入力	登録動画サイズ	1920x1080 (アスペクト比 16:9)
出力	TileType	R (アスペクト比 3:2)
	Level	4,3,2,1 (解像度384x256, 288x192, 192x128, 96x64)
Policy	crop_position	3: [右上]
	padding_position	2: [左上]
	crop_inheriting （複数の出力サイズに対して、タイル を同時生成する場合のポリシー）	1: [有効表示範囲をレベル毎に最適化]
	crop_policy_ob （出力サイズが、入力サイズより 大きい場合の伸縮ポリシー）	3: [Dot by dot] 単純なパディング
	crop_policy_ib （入力サイズが、出力サイズより 大きい場合の伸縮ポリシー）	3: [Dot by dot] 単純なクロップ

- 384 x 256 (R Level 4) ＜ 1920 x 1080 (登録動画サイズ)
- 288 x 192 (R Level 3) ＜ 1920 x 1080
- 192 x 128 (R Level 2) ＜ 1920 x 1080
- 96 x 64 (R Level 1) ＜ 1920 x 1080

この場合、レベル4,3,2,1のタイルを順に生成し、登録動画が全てのレベルの出力サイズより大きいので、crop_policy_ib:3（Dot by dot）が使用され、出力サイズに合わせて登録動画を切り取ります。この時、crop_position:3（右上）に従って、登録動画の右上を切り取ります。また、この時、crop_inheriting:1（有効表示範囲をレベル毎に最適化）に従って、各レベル毎の、出力領域のサイズに合わせて、切り取る範囲を個別に決定します。このため、例３では、各レベルの画像内容が同じでしたが、この例では、各レベルの画像内容が変化します。



内容は 2017 年 07 月 31 日 時点のものです。今後サービスの向上などの理由により、内容が変更される場合があります。