Media Syndication API Documentation

VUALTO

Mar 15, 2021

Contents

Here you will find technical documenation for the Media Syndication API for publishing both Live and VOD content to supported media destinations - such as Facebook or YouTube.

If you have any questions or need assistance in using this platform, please view our Support page

CHAPTER 1

Contents:

1.1 DEVELOPER DOCUMENTATION

The Media Syndication API provides integration with different media services (i.e. Facebook, MPX, YouTube, etc.) using two VUALTO concepts; Destinations and Publications.

Destinations represent the target location, such as a Facebook page, and store information related to accessing these targets (i.e. page ID, access tokens, etc.).

Publications represent the content being sent to a Destination, and can contain various forms of metadata depending on the support at the other end. They also contain 'state' information depending on what is going on via this API. Every publication must have a Destination, and deleting the Destination will delete any associated publications. Publications are split into Live and VOD through different API routes as the workflows are vastly different.

To get more information about each component, view the links below:

1.1.1 DESTINATIONS

Destinations represent a location where content should be pushed to- we refer to this as a Target and it is enumerated consistently throughout the system.

Destinations generally contain information relating to API access, and identifying information for where on the Target platform the content is being pushed to. For example, a Facebook destination will have a `ResourceId` which relates to the Facebook page or timeline where the content will exist.

Custom Destinations are an exception in that they do not always determine a consistent location. It is possible for two publications to share a Custom destination, but push to completely separate platforms.

Destination Object

A Destination will contain the following:

```
{
    "id": "2fd62eed-24ea-4feb-a409-a1fe1704cc86",
    "name": "Example Destination",
    "target": 5,
    "configJson": "{...}",
    "client": "vualto-example",
    "active": false,
    "link": "...",
    "destinationProperties": []
}
```

The `configJson` property will vary between Target types and is used by the system to manage API access. Please refer to the *below* for details on required data.

Destination Metadata

There are a set of permitted values querying the API for Metadata. See *query Target metadata* and *query Destination metadata*.

Metadata:

- YouTubeRegions
- YouTubeVideoCategories
- YouTubePrivacyStatus
- YouTubeLicense

Some data requires or supports extra data sent in the request, if providing the DestinationId this can be pre-populated from the existing data. This is as follows:

```
    YouTubeVideoCategories - requires `?youTubeRegions=X` to be specified, unless␣
↪providing DestinationId.
```

Supported Actions

GET `/destination/{id}`

Get a Destination by its ID.

Response: `200 OK` and JSON containing *Destination object*.

GET `/destination/`

Gets all Destinations (for the given Client).

Response: `200 OK` and a JSON array containing multiple *Destination objects*.

GET `/destination/metadata`

Return the available metadata values for a given Target and metadata field.

Query Parameters:

```
    metadata: {string} - Required. Set it to be the name of the metadata field.

    target: {string} - Required. Set it to be the string value of your Target (i.e.
↪'Facebook').

    noCache: {bool} - Optional.
```

Refer to *the above* for permitted values.

Response: `200 OK` and the following JSON structure:

```
{
    "name": "metadata",
    "target": 0,
    "Data": {
        "Key": "Value"
    }
}
```

GET `/destination/metadata/{id}`

Return the available metadata values for a given Destination and metadata field. This differs to the above as some metadata is dependent on the configured Destination (such as YouTube getting Categories based on the region).

Query Parameters:

```
    metadata: {string} - Required. Set it to be the name of the metadata field (see␣
↪below).

    target: {string} - Required. Set it to be the string value of your Target (i.e.
↪'Facebook').

    noCache: {bool} - Optional.
```

Refer to *the above* for permitted values.

Response: `200 OK` and the following JSON structure:

```
{
    "name": "metadata",
    "target": 0,
    "Data": {
        "Key": "Value"
    }
}
```

POST /destination/

Add a Destination.

Request Body:

```
{
    "name": "Example Destination",
    "target": "Custom",
    "configJson": "{...}"
}
```

The `target` property expects a string representation of the Target, rather than the enumerated version.

The `configJson` property will vary between Target types. Please refer to the *below* for details on required data.

Response: `201 Created` and JSON of the *Destination object*.

PUT /destination/{id}

Updates the specified Destination.

```
{
    "name": "New Name",
    "target": "Facebook",
    "configJson": "{...}",
    "active": true
}
```

Response: `200 OK` and JSON containing *Destination object*.

PUT /destination/state/{id}

Updates the `active` property for the specified Destination.

Query Parameters:

```
    active: {bool} - Required. bool value to set the active property to.
```

Response: `200 OK`

PUT /destination/activate/{id}

Queries the activation process for the specified Destination.

For detailed information, refer to activation processes.

Response: `200 OK` and the following JSON (details as example):

```
{
    "actionTitle": "Destination does not require action.",
    "actionDescription": "This is an example message. Destination is Active.",
    "actionRequired": false
}
```

DEL /destination/{id}

Deletes the specified Destination.

Response: `200 OK`

DEL /destination/cache

Deletes the Destination(s) cache.

Query Parameters:

```
destinationId: {GUID} - Optional. GUID of the Destination to restrict cache deletion␣
↪to a single Destination.
```

Response: `200 OK`

1.1.2 DESTINATION ACTIVATION

Every Destination requires activation before it can be used, and each Target type will require a different workflow to activate. This process is determined by the Target, and can require re-activation during the lifespan of the Destination.

Activation Response

```
{
    "actionTitle": "Destination does not require action.",
    "actionDescription": "This is an example message. Destination is Active.",
    "actionRequired": false
}
```

Destination JSON

Each Target requires/supports different data being stored in the Destination.

Facebook:

```
{
    "fb_resourceId": "234510293992583",
    "fb_encoderBrand": "encoder-brand",
    "fb_encoderModel": "encoder-model",
    "fb_encoderVersion": "encoder-version"
}
```

If intended to create Live Publications, you will need to provide Facebook with the details of your encoder. However, there are no checks in place on this information and you can submit any non-blank value for these properties.

Additionally, you may need to send Facebook a `resourceId` which will need to be the Page ID you wish to connect. If this is supposed to be a user's timeline, you can leave this blank and it will be picked up during the activation process.

Currently, uploading VOD to a user's timeline has been removed from Facebook. However, you can stream Live to a user's timeline.

YouTube:

If you intend to create VOD Publications, you will need to provide YouTube with the region of your account. Otherwise, no details are required.

```
{
    "yt_region": "..."
}
```

Custom:

Custom Destinations do not require any details, but can be configured to have static live stream details, which Publications can inherit from.

```
{
    "cstm_streamUrl": "...",
    "cstm_streamName": "...",
    "cstm_backupStreamUrl": "...",
    "cstm_backupStreamName": "..."
}
```

Twitter:

There are no required or optional properties for connecting a Twitter account.

Activation Processes

Custom

There are no necessary steps to activate a Custom Destination. They are active by default.

Facebook

1. Create the Destination, optionally providing a `resourceId` in the *Destination JSON*.
2. Call GET /destination/activate
3. Go to https://www.facebook.com/device.
4. Enter the code provided in Step 2 and authenticate with Facebook.
5. Repeat step 2.

This should result in a confirmation that the Destination is now active.

YouTube

1. Create the Destination, optionally providing a Region in the *Destination JSON*.
2. Call GET /destination/activate
3. Go to the provided URL in the response and authenticate with Google.
4. Repeat step 2.

This should result in a confirmation that the Destination is now active.

Twitter

1. Create the Destination.
2. Call GET /destination/activate
3. Go to the provided URL in the response and authenticate with Twitter.
4. Repeat step 2.

This should result in a confirmation that the Destination is now active.

1.1.3 DESTINATION PROPERTIES

Destination Properties are additional custom metadata that can exist on a Destination. They provide a method of storing any custom information which is not suitable on the Destination itself.

Destination Property Object

A Destination Property will contain the following:

```
{
    "id": "1c7743ac-34a7-4868-aa04-7557facc4ad0",
    "destinationId": "12e3c83a-c829-47ee-b77c-44806f12576a",
    "category": "my-category",
    "key": "my-key",
    "value": "my-value",
```

(continues on next page)

(continued from previous page)

```
    "client": "vualto"
}
```

The Category, Key, and Value properties are always string data, but can be anything at all. This means you can break down your data into separate properties or store them as a serialized string if you prefer.

Additionally, only a single Category/Key combination can exist for the given Destination. This means if you try to Add a Destination Property which already exists, it will instead perform an Update.

Supported Actions

GET `/destinationproperty/id/{id}`

Get a Destination Property by Id.

Response: `200 OK` and JSON containing the *Destination Property object*.

GET `/destinationproperty/{destinationId}`

Get a Destination Property by DestinationId, Category, and Key.

Query Parameters:

```
    Category: {string} - Required. The Category value to refine to.

    Key: {string} - Required. The Key value to refine to.
```

Response: `200 OK` and JSON containing the *Destination Property object*.

GET `/destinationproperty/all/{destinationId}`

Get all DestinationProperties for a given DestinationId.

Query Parameters:

```
    Category: {string} - Optional. The Category value to refine to.

    Key: {string} - Optional. The Key value to refine to.

    Value: {string} - Optional. The Value to refine to.
```

Response: `200 OK` and JSON containing multiple *Destination Property objects*.

POST `/destinationproperty/`

Add or Update a Destination Property.

```
{
    "destinationId": "",
    "category": "",
    "key": "",
    "value": "",
    "id": null
}
```

By specifying a value for `Id` this will perform an Update, else it will perform an Add.

Response: `200 OK` and JSON containing the *Destination Property object*.

DEL `/destinationproperty/id/{id}`

Delete a Destination Property by Id.

Response: `200 OK`

DEL `/destinationproperty/{destinationId}`

Delete Destination Properties for a given DestinationId, Category, and Key.

Query Parameters:

```
    Category: {string} - Required. The Category value to refine to.

    Key: {string} - Required. The Key value to refine to.
```

Response: `200 OK`

DEL `/destinationproperty/all/{destinationId}`

Delete all Destination Properties for a given DestinationId, optionally filtered by Category, Key, and Value.

Query Parameters:

```
    Category: {string} - Optional. The Category value to refine to.

    Key: {string} - Optional. The Key value to refine to.

    Value: {string} - Optional. The Value to refine to.
```

Response: `200 OK`

1.1.4 LIVE PUBLICATIONS

Publications represent *some* content at a given Destination. In the case of Live Publications, this is a live stream in the past, present, or future.

Live publications contain metadata information, which can be replicated on the desired Destination, as well as the stream information which can be consumed by an external service to stream to. They also contain some basic state information.

Live Publication Object

A Live Publication is defined as the following structure:

```
{
    "id": "16e4d793-0484-4923-a4d8-3e3a0604d725",
    "target": 0,
    "destinationId": "8b8d0b78-60f0-4257-b2e7-13874ecd28f6",
    "destinationName": "Example Destination",
    "transactionId": "bfe6e52b-a5c5-47f6-b77d-be0d80ff46cf",
    "contentId": "my-content-id",
    "dateCreated": "2018-10-05T12:28:14Z",
    "dateCompleted": "",
    "dateStarted": "2018-10-05T12:29:39Z",
    "publicationState": 3,
    "publicationStateString": "Completed",
    "details": {},
    "callbacks": [
        {
            "id": "4bf271e2-1544-4343-ad43-15b6d6d2cdd8",
            "publicationId": "16e4d793-0484-4923-a4d8-3e3a0604d725",
            "url": "...",
            "lastTriggered": "2018-10-05T12:29:40Z"
        }
    ]
}
```

The `details` property differs per Target and is here.

The property `callbackUrls` inside of `details` will always be a string array of the URL properties from the `callbacks` property from the *Publication* itself.

VOD Publications match Live Publications in structure, except for the `details` property. This means you can work with both generically.

Live Publication States

A Live Publication can go through the following states:

- 1 = Scheduled - the Publication exists in the API but not on the Destination.
- 2 = InProgress - the Publication is currently started but not yet complete.
- 3 = Completed - the Publication has been completed successfully.

- 4 = FailedDestination - the Publication’s Destination is not active.
- 6 = FailedPublication - the Publication is lacking ingestionDetails or isUnavailable is true.

Supported Actions

Almost all Live Publication actions require both the Id of the Publication and the Id of the Destination in order to properly identify the Publication for the action.

GET `/livepublications/`

Get multiple Live Publications and refine by other parameters.

Query Parameters:

```
   targetFilter: {string} - Optional. Specify the Target (i.e. 'Facebook') to filter␣
↪to.

   contentIdFilter: {string} - Optional. Specify a value to filter the ContentId to␣
↪include (i.e. any publication with a content ID including the specified string).

   queryTarget: {bool} - Optional. Should the Target be queried for the additional␣
↪details?

   noCache: {bool} - Optional. Set to true to bypass any existing cache.

   orderBy: {PublicationDateField} - Optional. Set to the Date field you wish to␣
↪order by (Created, Started, Completed)

   from: {string} - Optional. Set to the ISO8601 DateTime string to search From.

   to: {string} - Optional. Set to the ISO8601 DateTime string to search To.

   pageNumber: {int} - Optional. Set to the Page Number to search by.

   pageSize: {int} - Optional. Set to the Page Size to search by.
```

Response: `200 OK` and a paginated set of *Publications*.

GET `/livepublications/transactionId/{transactionId}`

Get multiple Live Publications by TransactionId and refine by other parameters.

Query Parameters:

```
   transactionId: {GUID} - Required. Required to identify the publication(s).

   targetFilter: {string} - Optional. Specify the Target (i.e. 'Facebook') to filter␣
↪to.

   queryTarget: {bool} - Optional. Should the Target be queried for the additional␣
↪details?
```

(continues on next page)

(continued from previous page)

```
    noCache: {bool} - Optional. Set to true to bypass any existing cache.
```

Response: `200 OK` and an array of *Publications*.

GET `/livepublications/contentId/`

Get a single publication by ContentId and DestinationId and refine by other parameters.

Query Parameters:

```
    contentId: {string} - Required to identify the publication.

    destinationId: {GUID} - Required to identify the publication.

    queryTarget: {bool} - Optional. Should the Target be queried for the additional␣
↪details?

    noCache: {bool} - Optional. Set to true to bypass any existing cache.
```

Response: `200 OK` and a *Publication*.

GET `/livepublications/{id}`

Get a single publication by Id and DestinationId.

Query Parameters:

```
    destinationId: {GUID} - Required to identify the publication.

    queryTarget: {bool} - Optional. Should the Target be queried for the additional␣
↪details?

    noCache: {bool} - Optional. Set to true to bypass any existing cache.
```

Response: `200 OK` and a *Publication*.

GET `/livepublications/stats/{id}`

Get the viewing statistics for a single publication.

Query Parameters:

```
    noCache: {bool} - Optional. Set to true to bypass any existing cache.
```

Response: `200 OK` and the relevant JSON from below:

Facebook:

```
{
    "target": 0,
    "details": {
        "liveViewsMax": 0,
        "liveViews": 0,
        "likeCount": 0,
        "loveCount": 0,
        "hahaCount": 0,
        "wowCount": 0,
        "sadCount": 0,
        "angryCount": 0
    }
}
```

YouTube:

```
{
    "target": 2,
    "details": {
        "concurrentViewers": 0,
        "viewCount": 0,
        "likeCount": 0,
        "dislikeCount": 0,
        "favouriteCount": 0,
        "commentCount": 0
    }
}
```

POST `/livepublications/`

Create multiple Live Publications, using the specified Destination & PublicationJson.

Request Body:

```
{
    "publications": [
        {
            "contentId": "example-contentId",
            "destinationId": "72e196a4-e3a3-4ad6-b15c-2c04f9ee3d9c",
            "publicationJson": "{...}"
        }
    ]
}
```

It is important to ensure that the JSON format is correct for the Destination Target - visible here.

All publications that get created in the same request will have the same `transactionId` value.

Response: `200 OK` and the JSON containing the *Publication*

PUT `/livepublications/{id}`

Update a Live Publication.

Request Body:

```
{
    "destinationId": "8b8d0b78-60f0-4257-b2e7-13874ecd28f6",
    "publicationJson": "{...}"
}
```

See *PublicationJson* for details on what can be submitted. It is important to ensure that the JSON format is correct for the Destination Target.

Response: `200 OK` and the JSON containing the *Publication*

PUT `/livepublications/recreate/`

Recreate a single publication by Id and DestinationId.

Recreate is the process of re-setting a Publication and acquiring fresh streaming URLs. Useful in the case of unexpected errors without having to re-submit the metadata.

Response: `200 OK` and the *Publication*.

PUT `/livepublications/start/`

Start a single publication by Id and DestinationId.

Support varies by Target type. However, the system will always record this action as the `DateStarted` time.

Response: `200 OK` and an operation response:

```
{
    "description": "The Publication has been unpublished.",
    "isSuccessful": true
}
```

PUT `/livepublications/stop/`

Stop a single publication by Id and DestinationId.

Support varies by Target type. However, the system will always record this action as the `DateCompleted` time.

Response: `200 OK` and an operation response:

```
{
    "description": "The Publication has been unpublished.",
    "isSuccessful": true
}
```

PUT `/livepublications/publish/`

Publish a single publication by Id and DestinationId.

Support varies by Target type.

Response: `200 OK` and an operation response:

```
{
    "description": "The Publication has been published.",
    "isSuccessful": true
}
```

PUT `/livepublications/unpublish/`

Unpublish a single publication by Id and DestinationId.

Support varies by Target type.

Response: `200 OK` and an operation response:

```
{
    "description": "The Publication has been unpublished.",
    "isSuccessful": true
}
```

DEL `/livepublications/{id}`

Delete a single publication by Id and DestinationId.

Response: `200 OK`

DEL `/livepublications/all/{contentId}`

Delete multiple Live Publications by ContentId.

Response: `200 OK`

DEL `/livepublications/cache/`

Delete all Live Publication cache.

Response: `200 OK`

1.1.5 LIVE PUBLICATION DETAILS

The `details` property exists for each Live Publication and differs per Target.

For Live, the `details` property also contains the streaming URLs as an array called `ingestionDetails`. Typically there are 1 or 2 of these, with one of these marked as Primary. You can view the structure *below*

- Add objects can be used to represent the initial data sent into the system when Creating.
- Update objects can be used to represent the updated data sent into the system when Updating.
- View objects are returned inside of the PublicationModel, most of these values will be null unless `queryTarget` is set to `True`.

Shared

Shared properties can exist on the details object. These can be identified by the lack of a prefix (i.e. `fb_`, `yt_`, etc.).

This is useful for working with properties that do not need to respect the Target, such as `embedHtml`, `link` or `ingestionDetails`. Shared properties are not required to exist on every Publication type however - such as Custom lacking the `embedHtml` and `link` properties.

IngestionDetails

All RTMP URLs are returned inside a shared object inside of the `details` object. They are returned in the following structure:

```
{
    "rtmpServerUrl": "...",
    "rtmpStreamName": "...",
    "rtmpApplicationName": "...",
    "isPrimary": true,
    "isSsl": false
}
```

Custom

Add Details

```
{
    "cstm_description": "Custom Description",
    "cstm_streamUrl": "http://server-to-send-to.com",
    "cstm_streamName": "streamKeyGoesHere",
    "cstm_backupStreamUrl": null,
    "cstm_backupStreamName": null,
}
```

Required: `cstm_streamUrl`, `cstm_streamName`

These are not required if the Custom Destination has URLs configured for this Publication to inherit from.

Update Details

```
{
    "cstm_description": "Custom Description",
    "callbackUrls": [
        "http://my-callback-url.com/Publication/Update"
    ]
}
```

View Details

```
{
    "cstm_description": "Custom Description",
    "ingestionDetails": []
}
```

Facebook

Add Details

```
{
    "fb_title": "Post Title",
    "fb_description": "Post Description",
    "fb_saveVod": true,
    "fb_startTime": "2019-05-19T12:45:00Z",
    "fb_endTime": "2019-05-19T12:55:00Z",
    "fb_contentTags": [
        "my-tag"
    ],
    "callbackUrls": [
        "http://my-callback-url.com/Publication/Update"
    ]
}
```

Required: `fb_title`

Update Details

```
{
    "fb_title": "Post Title",
    "fb_description": "Post Description",
    "fb_saveVod": true,
    "fb_startTime": "2019-05-19T12:45:00Z",
    "fb_endTime": "2019-05-19T12:55:00Z",
    "fb_contentTags": [
        "my-tag"
    ],
    "callbackUrls": [
        "http://my-callback-url.com/Publication/Update"
```

(continues on next page)

(continued from previous page)

```
    ]
}
```

View Details

```
{
    "fb_title": "Post Title",
    "fb_description": "Post Description",
    "fb_saveVod": true,
    "fb_startTime": "2019-05-19T12:45:00Z",
    "fb_endTime": "2019-05-19T12:55:00Z",
    "fb_contentTags": [
        "my-tag"
    ],
    "fb_resourceId": "2147581300458",
    "fb_status": "LIVE",
    "embedHtml": "",
    "ingestionDetails": []
}
```

YouTube

Add Details

```
{
    "yt_title": "YouTube Video",
    "yt_description": "YouTube Description",
    "yt_startTime": "2019-05-19T12:45:00Z",
    "yt_endTime": "2019-05-19T12:55:00Z",
    "yt_privacyStatus": "unlisted",
    "yt_ingestionType": "rtmp",
    "callbackUrls": [
        "http://my-callback-url.com/Publication/Update"
    ]
}
```

Required: `yt_title`, `yt_startTime`, `yt_privacyStatus`

Permitted values for privacy status: `public`, `private`, `unlisted`

Permitted values for ingestion type: `rtmp`, `dash`

Update Details

```
{
    "yt_title": "YouTube Video",
    "yt_description": "YouTube Description",
    "yt_startTime": "2019-05-19T12:45:00Z",
    "yt_endTime": "2019-05-19T12:55:00Z",
```

(continues on next page)

(continued from previous page)

```
    "yt_privacyStatus": "unlisted",
    "yt_ingestionType": "rtmp",
    "callbackUrls": [
        "http://my-callback-url.com/Publication/Update"
    ]
}
```

View Details

```
{
    "yt_title": "YouTube Video",
    "yt_description": "YouTube Description",
    "yt_startTime": "2019-05-19T12:45:00Z",
    "yt_endTime": "2019-05-19T12:55:00Z",
    "yt_privacyStatus": "unlisted",
    "yt_ingestionType": "rtmp",
    "yt_broadcastId": "29834148",
    "yt_liveStreamId": "655612",
    "ingestionDetails": []
}
```

1.1.6 MEDIA ASSETS

Media Assets are entities which represent the content required for a VOD Publication, and effectively define the state of the relevant MP4.

Media Asset Object

```
{
    "id": "1ab537cc-e787-46d9-a221-025a8554c79d",
    "publicationId": "6444cb7c-567a-4d8d-bbcf-a4bec5b197d5",
    "destinationId": "133a6e5c-dfb4-459b-b1ba-0ce99d0c47fc",
    "contentUrl": "https://d13hfsir6wgktj.cloudfront.net/vuworkflow/content/vod/
↪432db26a-1c72-4784-bbb3-9f9d3ea70c71/432db26a-1c72-4784-bbb3-9f9d3ea70c71.mp4",
    "client": "vualto-example",
    "assetState": "Complete",
    "delayedState": null,
    "delayedUntil": null,
    "contentUrlId": "8d504555-544c-28c5-7b51-066e1cbbbcd5",
    "retryCount": 0,
    "dateCreated": "2019-05-20T08:42:22Z",
    "dateUpdated": "2019-05-20T08:43:15Z",
    "dateCompleted": "2019-05-20T08:43:15Z",
    "fileSizeInBytes": 14615254,
    "contentDuration": "00:00:33.7600000"
}
```

Initially, MediaAssets start with the minimal information, but will be populated as the asset progress through it's state workflow.

Media Asset States

Any MediaAsset will go through a series of states as it processes. The list of states are:

```
    PendingCapture     = 0   : This asset is awaiting capture externally.
    Capturing          = 1   : This asset is undergoing capture externally.
    CaptureError       = 2   : This asset has failed to capture externally. This will␣
↪be retried.
    PendingDownload    = 3   : This asset is pending download.
    Downloading        = 4   : This asset is currently downloading.
    DownloadError      = 5   : This asset has failed to download. This will be retried.
    PendingUpload      = 6   : This asset is pending upload.
    Uploading          = 7   : This asset is currently uploading.
    UploadError        = 8   : This asset has failed to upload. This will be retried.
    Complete           = 9   : This asset has completed the process of uploading to␣
↪the Target.
    Cancelled          = 10  : This asset is not suitable for the Target therefore was␣
↪cancelled.
    Delayed            = 11  : This asset is delayed to run at a later time.
```

Supported Actions

GET /mediaassets/

Get all MediaAssets.

Query Parameters:

```
  contentUrlFilter: {string} - Optional. Search for MediaAssets for a given␣
↪contentUrl (or part of).

  contentUrlIdFilter: {guid} - Optional. Search for MediaAssets for a given␣
↪contentUrlId.

  assetStates: {string} - Optional. Search for MediaAssets whose state are in the␣
↪given comma separated list of states.

  noCache: {bool} - Optional. Set to true to bypass any existing cache.
```

Response: `200 OK` and a JSON array containing multiple *MediaAsset objects*.

GET /mediaassets/publication/

Get MediaAsset for the given Publication.

Query Parameters:

```
  publicationId: {guid} - Required. The PublicationId to search for.

  destinationId: {guid} - Required. The DestinationId to search for.

  noCache: {bool} - Optional. Set to true to bypass any existing cache.
```

Response: `200 OK` and a *MediaAsset object*.

PUT `/mediaassets/state`

Update the state for a given MediaAsset.

Request Body:

```
{
  "Id": "c68bec3c-c334-47c1-97a0-6be93c3523fd",
  "AssetState": "PendingDownload",
  "ForceChange": false
}
```

ForceChange is an optional property and will default to False. Passing True will force the state to change, even if it was in-progress or Cancelled.

Response: `200 OK` and a *MediaAsset object*.

PUT `/mediaassets/state/content/`

Update the state for all MediaAssets for a given ContentUrlId and existing states.

Request Body:

```
{
  "contentUrlId": "c68bec3c-c334-47c1-97a0-6be93c3523fe",
  "newState": "PendingDownload",
  "applicableStates": [
    "DownloadError"
  ]
}
```

You must provide a set of states to change from, else the call could affect too many items.

Response: `200 OK` and a JSON array containing multiple *MediaAsset objects*.

PUT `/mediaassets/state/content/validate`

Update the state for all MediaAssets for a given ContentUrlId and existing states, performing validation on the file size & length.

Request Body:

```
{
  "contentUrlId": "c68bec3c-c334-47c1-97a0-6be93c3523fe",
  "newState": "PendingDownload",
  "applicableStates": [
    "DownloadError"
```

(continues on next page)

(continued from previous page)

```
  ],
  "contentLength": "00:12:55",
  "fileSizeInBytes": 1000
}
```

You must provide a set of states to change from, else the call could affect too many items.

If contentLength or fileSizeInBytes is not provided, then this will not be checked, and may error when attempting Upload.

Response: `200 OK` and a JSON array containing multiple *MediaAsset objects*.

DEL `/mediaassets/cache/`

Clear the MediaAsset cache.

Response: `200 OK`

1.1.7 VOD PUBLICATIONS

Publications represent *some* content at a given Destination. In the case of VOD Publications, this is a video file either uploaded or to be uploaded.

VOD Publications contain metadata for the video (and associated Destination object). They also contain state information about the uploading process.

VOD Publications are backed by Media Assets, which represent the MP4 content itself.

VOD Publication Object

A VOD Publication is defined as the following structure:

```
{
    "id": "16e4d793-0484-4923-a4d8-3e3a0604d725",
    "target": 0,
    "destinationId": "8b8d0b78-60f0-4257-b2e7-13874ecd28f6",
    "destinationName": "Example Destination",
    "transactionId": "bfe6e52b-a5c5-47f6-b77d-be0d80ff46cf",
    "contentId": "my-content-id",
    "dateCreated": "2018-10-05T12:28:14Z",
    "dateCompleted": "",
    "dateStarted": "2018-10-05T12:29:39Z",
    "publicationState": 3,
    "publicationStateString": "Completed",
    "details": {},
    "callbacks": [
        {
            "id": "4bf271e2-1544-4343-ad43-15b6d6d2cdd8",
            "publicationId": "16e4d793-0484-4923-a4d8-3e3a0604d725",
            "url": "",
```

(continues on next page)

(continued from previous page)

```
            "lastTriggered": "2018-10-05T12:29:40Z"
        }
    ]
}
```

The `details` property differs per Target and is detailed here.

Live Publications match VOD Publications in structure, except for the `details` property. This means you can work with both generically.

VOD Publication States

A VOD Publication can go through the following states:

- 0 = Processing - the Publication is undergoing additional processing (i.e. MP4 download & upload).
- 1 = Scheduled - the Publication is awaiting further action (Uploading/Publishing).
- 3 = Completed - the Publication has been uploaded successfully.
- 4 = FailedDestination - the Publication's Destination is not active.
- 5 = FailedMediaAsset - the Publication has failed to download and/or upload to the Target.
- 6 = FailedPublication - the Publication is lacking details from the Target or IsUnavailable is true.

In addition to these states, it is recommended to check the `isUnavailable` property within the details object. This property is `True` if the object is not found on the Target platform.

Supported Actions

Almost all VOD Publication actions require both the Id of the Publication and the Id of the Destination in order to properly identify the Publication for the action.

GET `/vodpublications/`

Get multiple VOD Publications and refine by other parameters.

Query Parameters:

```
    targetFilter: {string} - Optional. Specify the Target (i.e. 'Facebook') to filter␣
↪to.

    contentIdFilter: {string} - Optional. Specify a value to filter the ContentId to␣
↪include (i.e. any publication with a content ID including the specified string).

    queryTarget: {bool} - Optional. Should the Target be queried for the additional␣
↪details?

    noCache: {bool} - Optional. Set to true to bypass any existing cache.

    orderBy: {PublicationDateField} - Optional. Set to the Date field you wish to␣
↪order by (Created, Started, Completed)
```

(continues on next page)

(continued from previous page)

```
    from: {string} - Optional. Set to the ISO8601 DateTime string to search From.

    to: {string} - Optional. Set to the ISO8601 DateTime string to search To.

    pageNumber: {int} - Optional. Set to the Page Number to search by.

    pageSize: {int} - Optional. Set to the Page Size to search by.
```

Response: `200 OK` and a paginated set of *Publications*.

GET `/vodpublications/transactionId/{transactionId}`

Get multiple VOD Publications by TransactionId and refine by other parameters.

Query Parameters:

```
    transactionId: {GUID} - Required. Required to identify the publication(s).

    targetFilter: {string} - Optional. Specify the Target (i.e. 'Facebook') to filter␣
↪to.

    queryTarget: {bool} - Optional. Should the Target be queried for the additional␣
↪details?

    noCache: {bool} - Optional. Set to true to bypass any existing cache.
```

Response: `200 OK` and an array of *Publications*.

GET `/vodpublications/contentId/`

Get a single publication by ContentId and DestinationId and refine by other parameters.

Query Parameters:

```
    contentId: {string} - Required to identify the publication.

    destinationId: {GUID} - Required to identify the publication.

    queryTarget: {bool} - Optional. Should the Target be queried for the additional␣
↪details?

    noCache: {bool} - Optional. Set to true to bypass any existing cache.
```

Response: `200 OK` and a *Publication*.

GET /vodpublications/{id}

Get a single publication by Id and DestinationId.

Query Parameters:

```
    destinationId: {GUID} - Required to identify the publication.

    queryTarget: {bool} - Optional. Should the Target be queried for the additional␣
↪details?

    noCache: {bool} - Optional. Set to true to bypass any existing cache.
```

Response: `200 OK` and a *Publication*.

POST /vodpublications/

Create multiple VOD Publications, using the specified Destination & PublicationJson.

Request Body:

```
{
    "publications": [
        {
            "contentId": "example-contentId",
            "destinationId": "72e196a4-e3a3-4ad6-b15c-2c04f9ee3d9c",
            "publicationJson": "{...}"
        }
    ]
}
```

It is important to ensure that the JSON format is correct for the Destination Target - visible here.

All publications that get created in the same request will have the same `transactionId` value.

Response: `200 OK` and the JSON containing the *Publication*

PUT /vodpublications/{id}

Update a VOD Publication.

Request Body:

```
{
    "destinationId": "8b8d0b78-60f0-4257-b2e7-13874ecd28f6",
    "publicationJson": "{...}"
}
```

See VOD Publication Details for more information on what can be submitted. It is important to ensure that the JSON format is correct for the Destination Target.

Response: `200 OK` and the JSON containing the *Publication*

PUT /vodpublications/publish/

Publish a single publication by Id and DestinationId.

Support varies by Target type.

Response: `200 OK` and an operation response:

```
{
    "description": "The Publication has been published.",
    "isSuccessful": true
}
```

PUT /vodpublications/unpublish/

Unpublish a single publication by Id and DestinationId.

Support varies by Target type.

Response: `200 OK` and an operation response:

```
{
    "description": "The Publication has been unpublished.",
    "isSuccessful": true
}
```

DEL /vodpublications/{id}

Delete a single publication by Id and DestinationId.

Response: `200 OK`

DEL /vodpublications/all/{contentId}

Delete multiple VOD Publications by ContentId.

Response: `200 OK`

1.1.8 VOD PUBLICATION DETAILS

The `details` property exists for each Vod Publication and differs per Target.

- Add objects can be used to represent the initial data sent into the system when Creating.
- Update objects can be used to represent the updated data sent into the system when Updating.
- View objects are returned inside of the PublicationModel, most of these values will be null unless `queryTarget` is set to `True`.

Shared

Shared properties can exist on the details object. These can be identified by the lack of a prefix (i.e. `fb_`, `yt_`, etc.).

This is useful for working with properties that do not need to respect the Target, such as `embedHtml` or `link`. Shared properties are not required to exist on every Publication type.

The `isUnavailable` property signals if the MediaSyndication platform could not find the VOD on the social media site. This will usually be because it is not yet uploaded, but can be that the video is deleted from the platform itself.

Facebook

Add Details

```
{
    "fb_title": "Post Title",
    "fb_description": "Post Description",
    "fb_noStory": true,
    "fb_embeddable": true,
    "fb_unpublishedContentType": null,
    "publishAt": null,
    "callbackUrls": [
        "http://my-callback-url.com/Publication/Update"
    ]
}
```

Permitted values for `fb_unpublishedContentType`: refer to Facebook's video documentation

PublishAt should be an ISO8601 date time string. If not provided then the video will Publish as soon as possible.

If PublishAt is within 10 minutes of the completed upload, then this value will be pushed back. This is a requirement on Facebook's side.

Required: none.

Update Details

```
{
    "fb_contentCategory": "",
    "fb_description": "",
    "fb_expireAt": "2019-05-19T17:00:00Z",
    "fb_expirationType": "expire_only",
    "fb_expireNow": false,
    "fb_embeddable": false,
    "fb_name": "videoName",
    "publishAt": null,
    "callbackUrls": [
        "http://my-callback-url.com/Publication/Update"
    ]
}
```

Permitted values for expiration type: `expire_and_delete`, `expire_only`

Permitted values for `fb_contentCategory`: please refer to Facebook's video documentation

View Details

```
{
    "fb_title": "Post Title",
    "fb_description": "Post Description",
    "fb_embeddable": true,
    "fb_noStory": true,
    "fb_videoId": "2147581300458",
    "publishAt": null,
    "published": true,
    "link": "...",
    "embedHtml": "...",
    "isUnavailable": false
}
```

YouTube

Add Details

```
{
    "yt_title": "YouTube Video",
    "yt_description": "YouTube Description",
    "yt_tags": [
        "tag1"
    ],
    "yt_categoryId": 1,
    "yt_defaultAudioLanguage": "en",
    "yt_defaultLanguage": "en",
    "yt_embeddable": true,
    "yt_license": "YouTube",
    "yt_privacyStatus": "Unlisted",
    "yt_publicStats": false,
    "yt_recordingDate": "2019-05-19T12:00:00Z",
    "publishAt": null,
    "callbackUrls": [
        "http://my-callback-url.com/Publication/Update"
    ]
}
```

Permitted values for privacy status: `Public`, `Private`, `Unlisted`

Permitted values for license: `CreativeCommon`, `YouTube`

Permitted values for `yt_categoryId` can be obtained from the metadata API.

PublishAt should be an ISO8601 date time string. If not provided then the video will Publish as soon as possible.

Update Details

```
{
    "yt_title": "YouTube Video",
    "yt_description": "YouTube Description",
```

(continues on next page)

(continued from previous page)

```
    "yt_tags": [
        "tag1"
    ],
    "yt_categoryId": 1,
    "yt_defaultAudioLanguage": "en",
    "yt_defaultLanguage": "en",
    "yt_embeddable": true,
    "yt_license": "YouTube",
    "yt_privacyStatus": "Unlisted",
    "yt_publicStats": false,
    "yt_recordingDate": "2019-05-19T12:00:00Z",
    "publishAt": null,
    "callbackUrls": [
        "http://my-callback-url.com/Publication/Update"
    ]
}
```

View Details

```
{
    "yt_title": "YouTube Video",
    "yt_description": "YouTube Description",
    "yt_tags": [
        "tag1"
    ],
    "yt_categoryId": 1,
    "yt_defaultAudioLanguage": "en",
    "yt_defaultLanguage": "en",
    "yt_embeddable": true,
    "yt_license": "YouTube",
    "yt_privacyStatus": "Unlisted",
    "yt_publicStats": false,
    "yt_recordingDate": "2019-05-19T12:00:00Z",
    "callbackUrls": [
        "http://my-callback-url.com/Publication/Update"
    ],
    "yt_videoId": "16985713",
    "publishAt": null,
    "published": true,
    "link": "...",
    "embedHtml": "...",
    "isUnavailable": false
}
```

Twitter

Add Details

```
{
    "tw_status": "Example Tweet content to be Published",
```

(continues on next page)

(continued from previous page)

```
    "publishAt": null,
    "callbackUrls": [
        "http://my-callback-url.com/Publication/Update"
    ]
}
```

Required: `tw_status`

PublishAt should be an ISO8601 date time string. If not provided then the video will Publish as soon as possible.

Update Details

```
{
    "tw_status": "Example Tweet content to be Edited",
    "publishAt": null,
    "callbackUrls": [
        "http://my-callback-url.com/Publication/Update"
    ]
}
```

Required: `tw_status`

If the provided `tw_status` matches the existing text, no update will be performed.

View Details

```
{
    "tw_title": "Example Tweet content here.",
    "tw_mediaId": "...",
    "tw_tweetId": "...",
    "publishAt": null,
    "published": true,
    "link": "...",
    "embedHtml": "...",
    "isUnavailable": false
}
```

1.1.9 CACHE MANAGEMENT

These endpoints allow for clients to clear their own cache within the application.

Supported Actions

DEL `/cache/`

Here.

Query Parameters:

```
  client {string} - Optional. Clears the specified client. Only applicable to␣
↪administrators.

  destinationId: {Guid} - Optional. Clears the cache related to a specified␣
↪Destination Id.

  livePublicationId: {Guid} - Optional. Clears the cache related to a specified Live␣
↪Publication Id.

  vodPublicationId: {Guid} - Optional. Clears the cache related to a specified Vod␣
↪Publication Id.

  clearAssetCache: {bool} - Optional. Set to True if you wish to clear the MediaAsset␣
↪cache.
```

Response: `200 OK` or `403 Forbidden` if attempting to clear another client's cache.

DEL `/cache/all/`

Clear all cache for the specified client.

Query Parameters:

```
  client {string} - Optional. Clears the specified client. Only applicable to␣
↪administrators.
```

Response: `200 OK` or `403 Forbidden` if attempting to clear another client's cache.

1.1.10 GLOBAL REQUIREMENTS

Authorisation: Every call to the Media Syndication API requires the following authorisation headers:

```
Client:     client-name-here
API-KEY:    xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxx
```

If you fail to provide a matching client and API key, you will recieve a *401 Unauthorised*.

If you have access to multiple clients, you must query these individually - you cannot specify multiple clients.

For administrators, using the central client & authentication key will return results from across **all** clients.

Route Prefix:

Every API route has the prefix */api/*. This means that a call documented as *GET /destination/* will require a GET request to *http://server.url.com/api/destination*. Failure to account for this will result in a *404 Not Found*.

Example CURL Request:

```
curl -X GET \
http://local.mediasyndication.vuworkflow.vualto.com/api/destination/ \
-H 'api-key: 00000000-0000-0000-0000-000000000000' \
-H 'cache-control: no-cache' \
-H 'client: vualto-example'
```

—

1.1.11 OTHER INFORMATION

Examples in this documentation are accurate where possible, but a number of the response messages have been made more generic and should not be expected to be verbatim.

Target Enumeration

The Media Syndication API uses 'Targets' to refer to the various Media platforms that could be pushed to. This is enumerated within the API to the following:

- Facebook = 0
- Twitter = 1
- YouTube = 2
- MPX = 3
- Custom = 4

Custom is a unique type, which can be used flexibly. Please refer to the relevant documentation for either Live or VOD publications.

MPX is reserved for future use and is not supported at this time.

Paginated Responses

For any response that returns a "paginated set of {objects}" it will follow this structure:

```
{
    "results": [],
    "totalCount": 56,
    "pageSize": 10,
    "pageNumber": 2
}
```

Supported Syndications

Live Syndication:

- Facebook
- YouTube
- Custom

VOD Syndication:

- Facebook
- YouTube
- Twitter

1.2 SUPPORT

1.2.1 HOW TO CONTACT US

For all non-critical support requests please email support@vualto.com. If you are a registered user you can contact support via the help centre.

For all critical issues please contact support via one of the numbers below.

- +44 (0)800 0314391
- +44 (0)1752 916051
- (800) 857 1808 (toll-free US number)