1 Campaign Items API Reference

The following operations are available via the API:

1. Fetch a List of Items for a specific Campaign - Fetch all of the Items associated with a certain Campaign.
2. Fetch a Single Item from a Campaign - Fetch a single Item associated with a certain Campaign.
3. Create an Item - Create a new Item in a specific Campaign.
4. Update an Item - Update an existing Item.
5. Delete (Stop) an Item - Move an existing Item to a 'STOPPED' status.
6. Fetch Children Items of an RSS Item
7. Fetch a specific Child Item of an RSS Item
8. Update a child of an RSS Item
9. Batch create campaign items with predefined thumbnails and titles (*skips* Taboola crawling).
10. Upload an image
11. Getty API's: Get taxonomies, Upload image, Get image

1.1 Cheat Sheet

This is merely a quick reference of the available end-points. For further information please continue to the sections below.

Description	Method	Path (prefix: `/backstage/api/1.0/[account-id]/campaigns`)
Fetch all Items for Campaign More info	GET	/[campaign-id]/items/
Fetch a specific Item from Campaign More info	GET	/[campaign-id]/items/[item-id]/
Add an Item to a Campaign More info	POST	/[campaign-id]/items/
Update an Item More info	POST, PUT	/[campaign-id]/items/[item-id]/

Delete an Item from a Campaign More info	DELETE	/[campaign-id]/items/[item-id]/
Fetch children of RSS Item More info	GET	/[campaign-id]/items/[item-id]/children/
Fetch a specific child of RSS Item More info	GET	/[campaign-id]/items/[item-id]/children/[child-id]/
Update a child of RSS Item More info	POST, PUT	/[campaign-id]/items/[item-id]/children/[child-id]/
Upload new Image and return the uploaded image url.	POST	/operations/upload-image
Batch create campaign items. More info	POST	/[campaign-id]/items/mass

1.2 Campaign Item Resource

1.2.1 Fields List

Required fields are fields that must be sent to the server when creating a new resource.
Read-only fields are fields that should never be sent to the server, and will appear only when *fetching* a resource.
Final fields are set once, when creating the resource, and become *read-only* afterwards.

Name	Modifier & Defaults	Type	Description
id	Read-only	String	Unique numeric ID of an Item (returned as a string)
campaign_id	Read-only	String	Unique numeric ID of the Item's parent Campaign resource (returned as a string)
type	Read-only	String Most updated possible values are accessible via	Automatically inferred from the content of the url

		dictionary. Here's the current ones: ITEM RSS	
url	Required	String Valid URL string Requires certain permissions Max length: 2000	A valid URL referencing the Item
thumbnail_url	N/A	String Valid URL string Cannot be modified while status is CRAWLING Max length: 1000	URL for the item's thumbnail image Automatically inferred from the content of the url
title	N/A	String Cannot be modified while status is CRAWLING	Item's title Automatically inferred from the content of the url
approval_state	Default: PENDING OR APPROVED (for users with the appropriate permissions)	String Requires certain permissions to modify. Cannot be modified while status is CRAWLING Most updated possible values are accessible via dictionary. Here's the current ones: APPROVED REJECTED PENDING	This field determines whether the Item is approved to be served. The Item can only be served if its approval_state is APPROVED. This field affects the status of the Item.
is_active	Default: true	Boolean Can be modified only when the Item has status=RUNNING or status=PAUSED	Determines whether the Item is active (served to widgets) or not. This field affects the status of the Item.
status	Read-only	String Most updated possible values are accessible via dictionary. Here's the current ones: RUNNING	An item is served as a recommendation only if its approval_state is APPROVED and its status is RUNNING.

		CRAWLING CRAWLING_ERROR NEED_TO_EDIT PAUSED STOPPED PENDING_APPROVAL REJECTED	

Trying to set field values which do not adhere to their restrictions, will result in a `400 Bad Request` error response.

1.3 Different Item Types

The `type` field defines the type of Item at hand. It is a *read-only* field, and its value is automatically determined from the URL supplied when creating the Item.
There are two possible types of items:

- `ITEM` - A simple Item, referencing a specific URL of a web page (e.g. link to an article).
- `RSS` - An item with a URL referencing an RSS feed (we support mRSS format). This Item will have Children Items - an Item for each URL in its feed.

The handling of both these types is similar, with minor differences in the request URL. The following sections will describe the handling of an Item of type `ITEM`. An additional section, "Handling RSS Items", concerns the differences and explains how to handle Items of type `RSS` and their children-items.

RSS Items should reference a URL which contains a feed in mRSS format. Other formats will not be recognised as RSS feeds.

1.4 Handling Item Statuses

1.4.1 CRAWLING

As long as an Item has the `CRAWLING` status, it is in *read-only* state - no field can be modified. It needs to be polled repeatedly until its status changes. Read the "Creating" section for further details and a concrete example.

> Items with `status=CRAWLING` are not served in Taboola Widgets.

1.4.2 CRAWLING_ERROR

The `CRAWLING_ERROR` status means there was some error while crawling - probably due to invalid or wrong URL. To solve this, try updating the Item's URL.

> Items with `status=CRAWLING_ERROR` are not served in Taboola Widgets.

1.4.3 NEED_TO_EDIT

The `NEED_TO_EDIT` status means the Item has been crawled successfully, but some fields could not be parsed from the item's content - either `thumbnail_url` or `title` (or both). To solve this, try updating these fields which have a value of `NULL`.

> Items with `status=NEED_TO_EDIT` are not served in Taboola Widgets.

1.5 Reading

Reading can be performed in two ways:

1. Fetch a list of Items of a specific Campaign
2. Fetch a single Item of a specific Campaign

1.5.1 Fetching a List

1.5.1.1 Request

To retrieve a list of Items for a specific Campaign, send an HTTP *GET* request in the following form:

```
GET /backstage/api/1.0/[account-id]/campaigns/[campaign-id]/items/
Host: https://backstage.taboola.com
Authorization: Bearer [access-token]
```

1.5.1.2 Example Response:

```
200 OK
```

```
{
    "results":[
        {
            "id"             : "1",
            "campaign_id"    : "124",
            "type"           : "ITEM",
            "url"            : "http://news.example.com/demo_article.html",
            "thumbnail_url"  : "http://cdn.example.com/demo_image.jpg",
            "title"          : "Demo Article",
            "approval_state" : "APPROVED",
            "is_active"      : true,
            "status"         : "RUNNING"
        }
    ]
}
```

1.5.2 Fetching a Specific Instance

1.5.2.1 Request

To retrieve a specific Item of a specific Campaign, send an HTTP *GET* request in the following form:

```
GET /backstage/api/1.0/[account-id]/campaigns/[campaign-id]/items/[item-id]/
Host: https://backstage.taboola.com
Authorization: Bearer [access-token]
```

The response will include a valid JSON object in its body representing the Campaign Item.

1.5.2.2 Example Response:

```
200 OK
```

```
{
    "id"             : "1",
    "campaign_id"    : "124",
    "type"           : "ITEM",
    "url"            : "http://news.example.com/demo_article.html",
    "thumbnail_url"  : "http://cdn.example.com/demo_image.jpg",
    "title"          : "Demo Article",
    "approval_state" : "APPROVED",
    "is_active"      : true,
    "status"         : "RUNNING"
}
```

1.6 Creating

When creating a new Item, *the only field acceptable* is the `url` field. After sending the appropriate request, a new Item resource is created with a status of `CRAWLING`. As long as an Item has the `CRAWLING` status, it is in *read-only* state - no field can be modified.
For the client-app to know when the Item has stopped crawling, and is available for edit, it needs to poll the server every couple of seconds (10 seconds is a good number to choose), requesting the Item each time, and checking whether its status has changed.
Once the Item has its status changed from `CRAWLING`, it becomes available for edit.

To create an item, send an HTTP *POST* request in the following form:

```
POST /backstage/api/1.0/[account-id]/campaigns/[campaign-id]/items/
Host: https://backstage.taboola.com
Authorization: Bearer [access-token]
```

```
{
    "url": [url]
}
```

1.6.1 Example: Creating an Item

1.6.1.1 Request

```
POST /backstage/api/1.0/taboola-demo/campaigns/124/items/
Host: https://backstage.taboola.com
Authorization: Bearer [access-token]
```

```
{
    "url": "http://news.example.com/demo_article.html"
}
```

Only the `"url"` field is acceptable when *creating* an Item.

1.6.1.2 Response:

```
200 OK
```

```
{
    "id"             : "1",
    "campaign_id"    : "124",
    "type"           : "ITEM",
    "url"            : "http://news.example.com/demo_article.html",
    "thumbnail_url"  : null,
    "title"          : null,
    "approval_state" : "PENDING",
    "is_active"      : true,
    "status"         : "CRAWLING"
}
```

The new Item returns with its status set to `"CRAWLING"`. This means it is in *read-only* state - no field can be modified.

After creating the Item, we need to poll the server every couple of seconds to see whether the Item's status has changed.

1.6.1.3 jQuery Example: Creating an Item and Polling its Status

create-item.js

```
var baseUrl = '/backstage/api/1.0/' + partner + '/campaigns/' + campaignId + '/items/';

$.post({
  url: baseUrl,
  headers: {
    Authorization: 'Bearer [access-token]'
  },
  contentType: 'application/json',
  dataType: 'json',
  data: JSON.stringify({
    url: 'http://news.example.com/demo_article.html'
  })
}).done(function (item) {
  if (item.status === 'CRAWLING') pollItem(item.id);
});

function pollItem (itemId) {
  var url = baseUrl + itemId;

  setTimeout(function () {
    $.getJSON(url).done(function (item) {
      if (item.status === 'CRAWLING') pollItem(itemId);
      else console.debug('Item is available!', item.title, item.thumbnail_url);
    }
  }, 10 * 1000);
}
```

Eventually, you will get a response containing the *crawled* item:

```
200 OK
```

```
{
    "id"             : "2",
    "campaign_id"    : "124",
    "type"           : "ITEM",
    "url"            : "http://news.example.com/demo_article.html",
    "thumbnail_url"  : "http://cdn.example.com/demo_image.jpg",
    "title"          : "Demo Article",
    "approval_state" : "APPROVED",
    "is_active"      : true,
    "status"         : "RUNNING"
}
```

1.7 Updating

As long as an Item has its status set to `"CRAWLING"` it is in *read-only* state - no field can be modified.

After successfully creating an Item (*POST*ing it to the server, then polling it until its status changes to `RUNNING`), you can send update requests.

To update an Item, send an HTTP *POST* request in the following form:

```
POST /backstage/api/1.0/[account-id]/campaigns/[campaign-id]/[item-id]/
Host: https://backstage.taboola.com
Authorization: Bearer [access-token]
```

When updating resources, it is possible to use either the `POST` method, or the `PUT` method. The API will respect both, but the documentation only uses the `POST` approach in examples.

An update request may include only a subset of the resource's fields. Included fields with `NULL` value will be treated as if they are missing from the request.
Fields which are missing or set to null in the request will not be modified.

1.7.1 Example: Update a Single Field

```
POST /backstage/api/1.0/taboola-demo/campaigns/124/items/2/
Host: https://backstage.taboola.com
Authorization: Bearer [access-token]
```

```
{
     "title": "Updated Title"
}
```

1.7.1.1 Response

```
200 OK
```

```
{
    "id"             : "2",
    "campaign_id"    : "124",
    "type"           : "ITEM",
    "url"            : "http://news.example.com/demo_article.html",
    "thumbnail_url"  : "http://cdn.example.com/demo_image.jpg",
    "title"          : "Updated Title",
    "approval_state" : "APPROVED",
    "is_active"      : true,
    "status"         : "RUNNING"
}
```

1.8 Deleting

Deleting Items per se is not possible in Taboola's system. Instead, the Item status is set to `STOPPED`, it will stop appearing in the Campaign's items list and will not be served as an ad.

Once an Item is deleted, it will stop appearing in the list of Items of the Campaign. Trying to fetch/update the Item will result in a `404 Not Found` error response.

To "delete" an Item from a specific Campaign, send an HTTP *DELETE* request in the following form:

```
DELETE /backstage/api/1.0/[account-id]/campaigns/[campaign-id]/items/[item-id]/
Host: https://backstage.taboola.com
Authorization: Bearer [access-token]
```

In response, the "deleted" item will be returned, and its status will be set to `STOPPED`.

1.8.1 Example: Delete an Item From Campaign

```
DELETE /backstage/api/1.0/taboola-demo/campaigns/124/items/2/
Host: https://backstage.taboola.com
Authorization: Bearer [access-token]
```

1.8.1.1 Response

```
200 OK
```

```
{
    "id"             : "2",
    "campaign_id"    : "124",
    "type"           : "ITEM",
    "url"            : "http://news.example.com/demo_article.html",
    "thumbnail_url"  : "http://cdn.example.com/demo_image.jpg",
    "title"          : "Updated Title",
    "approval_state" : "APPROVED",
    "is_active"      : true,
    "status"         : "STOPPED"
}
```

The Item's data is preserved, but its status is changed to `STOPPED`, and it will no longer appear in the list of Items for this Campaign.

1.9 Pausing / Unpausing Items

The Item's `status` field is *read-only*, and thus cannot be modified via an update request. Instead, one must change the Item's `is_active` field. The `is_active` field has two possible values: `true` and `false`. When an Item has both `is_active=true` and `status=RUNNING`, it is considered "active". If the Item has `is_active=false` its `status` will become `PAUSED`.

When an Item is `PAUSED`, it is not served in Taboola widgets.

1.9.1 Pause

In order to *pause* an Item, send an *update* request in the following form:

```
POST /backstage/api/1.0/[account-id]/campaigns/[campaign-id]/items/[item-id]/
Host: https://backstage.taboola.com
Authorization: Bearer [access-token]
```

```
{
    "is_active" : false
}
```

1.9.1.1 Example Response

```
200 OK
```

```
{
    "id"             : "3",
    "campaign_id"    : "124",
    "type"           : "ITEM",
    "url"            : "http://news.example.com/demo_article.html",
    "thumbnail_url"  : "http://cdn.example.com/demo_image.jpg",
    "title"          : "Updated Title",
    "approval_state" : "APPROVED",
    "is_active"      : false,
    "status"         : "PAUSED"
}
```

1.9.2 Unpause

In order to *unpause* an Item, send an HTTP *POST* request in the following form:

```
POST /backstage/api/1.0/[account-id]/campaigns/[campaign-id]/items/[item-id]/
Host: https://backstage.taboola.com
Authorization: Bearer [access-token]
```

```
{
    "is_active" : true
}
```

1.9.2.1 Example Response

```
200 OK
```

```
{
    "id"             : "3",
    "campaign_id"    : "124",
    "type"           : "ITEM",
    "url"            : "http://news.example.com/demo_article.html",
    "thumbnail_url"  : "http://cdn.example.com/demo_image.jpg",
    "title"          : "Updated Title",
    "approval_state" : "APPROVED",
    "is_active"      : true,
    "status"         : "RUNNING"
}
```

1.10 Handling RSS Items

A convenient way of adding multiple Items at once, is using an RSS feed. When adding an Item with an RSS feed URL, all the items in that feed are crawled and added as Children Items of the RSS Item. The Children Items of an RSS Item will be served like the regular Items of the Campaign.

The type of the Item resource is determined according to the crawled URL. If the supplied URL contains an RSS feed, the created Item will have a type of `RSS`.

1.10.1 Differences

RSS Items have several unique features that regular Items don't have:

- RSS Items have a list of associated *Children Items*, similar to how Campaigns have a list of Items.
- RSS Items have a `CRAWLING` status as long as their children are `CRAWLING`.
- You cannot *add* items to an RSS item, its children are fetched automatically.
- You can *update* a specific Child Itemq of an RSS Item.
- You can not *delete* a specific Child Item from an RSS Item.
- You can *pause/unpause* a specific Child Item of an RSS Item.

1.10.2 RSS Creation Flow

Creating an RSS Item is identical to creating a regular Item, with the difference that the URL of the Item must point to a valid RSS feed. The identification of an RSS feed is performed by Taboola's system automatically.

The status of an RSS Item will be `CRAWLING`, as long as its children are `CRAWLING`.

1.10.3 Fetching a List of RSS Children

Once an RSS Item has `status=RUNNING`, you can fetch a list of its Children Items.

To fetch a list of Children Items for a specific RSS Item, send an HTTP *GET* request in the following form:

```
GET
/backstage/api/1.0/[account-id]/campaigns/[campaign-id]/items/[item-id]/children/
Host: https://backstage.taboola.com
Authorization: Bearer [access-token]
```

The mechanism is similar to fetching a list of Items for a specific Campaign.

1.10.4 Fetching a Specific Child of an RSS Item

Once an RSS Item has `status=RUNNING`, you can fetch a list of its Children Items.

To fetch a list of Children Items for a specific RSS Item, send an HTTP *GET* request in the following form:

```
GET /backstage/api/1.0
      /[account-id]/campaigns/[campaign-id]/items/[item-id]/children/[child-item-i
      d]/
Host: https://backstage.taboola.com
Authorization: Bearer [access-token]
```

The mechanism is similar to fetching a specific Item for a specific Campaign.

1.10.5 Updating Items of an RSS

To *update* one of the Children Items of a specific RSS Item, send an HTTP *POST* request in the following form:

```
POST /backstage/api/1.0/
      [account-id]/campaigns/[campaign-id]/items/[item-id]/children/[child-id]/
Host: https://backstage.taboola.com
Authorization: Bearer [access-token]
```

The mechanism is similar to updating a regular item.

1.10.6 Pause/unpause Items of an RSS

To *pause* one of the Children Items of a specific RSS Item, send an HTTP *POST* request in the following form:

```
POST /backstage/api/1.0/
       [account-id]/campaigns/[campaign-id]/items/[item-id]/children/[child-id]/
Host: https://backstage.taboola.com
Authorization: Bearer [access-token]
```

```
{
    "is_active" : false
}
```

To *unpause* one of the Children Items of a specific RSS Item, send an HTTP *POST* request in the following form:

```
POST /backstage/api/1.0/
       [account-id]/campaigns/[campaign-id]/items/[item-id]/children/[child-id]/
Host: https://backstage.taboola.com
Authorization: Bearer [access-token]
```

```
{
    "is_active" : true
}
```

The mechanism is similar to pausing/unpausing a regular item.

1.11 Creating a Batch of Items

When creating a new item, *the only acceptable field* is the `url` field, which allows Taboola to crawl the item and extract the relevant metadata out of it. This method allows inserting a batch of items, each item characterized by its URL, title and thumbnail. This assumes Taboola crawling was performed externally, and forces these values into the DB.

Please note that this method also allows creating multiple items in a single hit (batch operation), in which case you'll need to send a collection of items with all of their details.

Please note that this is a sensitive process, as it bypasses the crawling mechanism of Taboola, and should only be used if all 3 values are acceptable by Taboola.

To create an item, send an HTTP *POST* request in the following format:

```
POST /backstage/api/1.0/[account-id]/campaigns/[campaign-id]/items/mass
Host: https://backstage.taboola.com
```

```
Authorization: Bearer [access-token]
```

```
{

    "collection": [
        {
            "url"            : [url],
            "thumbnail_url"  : [thumbnail url],
            "title"          : [title]
        }
    ]
}
```

1.11.1 Example: Creating an Item

1.11.1.1 Request

```
POST /backstage/api/1.0/taboola-demo/campaigns/124/items/mass
Host: https://backstage.taboola.com
Authorization: Bearer [access-token]
```

```
{
    "collection": [
        {
            "url"            : "http://news.example.com/demo_article.html",
            "thumbnail_url"  : "http://news.example.com/demo_article_thumbnail.jpg",
            "title"          : "Demo Article"
        }
    ]
}
```

1.11.1.2 Example Response:

```
200 OK
```

```
{
    "results":[
        {
            "id"             : "1",
            "campaign_id"    : "124",
            "type"           : "ITEM",
            "url"            : "http://news.example.com/demo_article.html",
            "thumbnail_url"  : "http://cdn.example.com/demo_image.jpg",
            "title"          : "Demo Article",
            "approval_state" : "APPROVED",
            "is_active"      : true,
            "status"         : "RUNNING"
        }
    ]
}
```

1.11.1 Example Request

```
POST /backstage/api/1.0/operations/upload-image
Content-Type: multipart/form-data; boundary="xxx"
Host: https://backstage.taboola.com
Authorization: Bearer [access-token]

--xxx
Content-Type: image/jpeg
name="file"; filename="original_file_name.jpg"

...image content...
```

- With Curl:

```
curl -H "Authorization: Bearer YOUR_TOKEN" -F "file=@FILE_PATH"
http://backstage.taboola.com/backstage/api/1.0/operations/upload-image
```

The method will return the uploaded image url.

1.11.2 Example Response

```
200 OK
```


```
{
    "name": "image_url",
    "value":
"https://s3.amazonaws.com/qa-upload-image/libtrc/static/thumbnails/689e01799d881dbbb7004ad4ea6cd496.jpg"
}
```


1.12 Uploading an Image

To *upload* an image to Taboola, send an HTTP *POST* request in the following form
With a file multipart containing the uploaded image:

```
POST /backstage/api/1.0/operations/upload-image
Content-Type: multipart/form-data
Host: https://backstage.taboola.com
Authorization: Bearer [access-token]
```

1.13 Get taxonomies

Get taxonomies of a given url.

1.12.1 Field List

End point fields

Name	Modifier & Defaults	Type	Description
url	Required	String Valid URL string Max length: 2000	A valid URL to the page to create taxonomies from

1.12.2 get taxonomies

To *get taxonomies*, send an HTTP *GET* request in the following form:

```
GET /backstage/api/1.0/operations/thirdparty/taxonomies?url=[url]
Host: https://backstage.taboola.com
Authorization: Bearer [access-token]
```

1.12.2.1 Example Request

```
GET
/backstage/api/1.0/operations/thirdparty/taxonomies?url=https://www.msn.com/en-xl/northamerica/northamerica-top-stories/trump-suggests-he-raised-biden-with-ukraines-president/ar-AAHGCG4?li=BBKxOg5
Host: https://backstage.taboola.com
Authorization: Bearer [access-token]
```

The method will return the taxonomies.

1.12.2.2 Example Response

```
200 OK
```

```
{
    "results": [
        {
            "name": "law, govt and politics"
```

```
        },
        {
            "name": "Joe Biden"
        },
        {
            "name": "presidential elections"
        }
    ],
    "metadata": {
        "total": null,
        "count": null,
        "static_fields": [],
        "static_total_fields": []
    }
}
```

1.14 Get image

Get images in accordance with the taxonomies.

1.14.1 Field List

End point fields

Name	Modifier & Defaults	Type	Description
taxonomies	Required	String[]	An array of taxonomies to get images from

1.14.2 get image

To *get images*, send an HTTP *GET* request in the following form:

```
GET /backstage/api/1.0/operations/thirdparty/image?taxonomies=[“t1”, “t2”, “t3”]
Host: https://backstage.taboola.com
Authorization: Bearer [access-token]
```

1.14.2.1 Example Request

```
GET /backstage/api/1.0/operations/thirdparty/image?taxonomies=[“Joe
Biden”,”presidential elections”]
Host: https://backstage.taboola.com
Authorization: Bearer [access-token]
```

The method will return the Geographic Info.

1.14.2.2 Example Response

```
200 OK
```

```
{
    "results": [
        {
            "source": "GETTY_IMAGES",
            "url":
"https://media.gettyimages.com/photos/vice-presidential-seal-and-empty-podium-picture-id7091
13893?b=1&k=6&m=709113893&s=170x170&h=Lk89iED4atdfS6U-Zmm3IGJh9u-Lr2XlL7pYdWGnoeI=",
            "id": "709113893",
            "cdn_url": null
        },
        {
            "source": "GETTY_IMAGES",
            "url":
"https://media.gettyimages.com/photos/vice-presidential-seal-and-empty-podium-picture-id7091
13889?b=1&k=6&m=709113889&s=170x170&h=TnKp6Wktm7ev9G3N6JQM5QaJPJxtPAp99gTrAjTAsMI=",
            "id": "709113889",
            "cdn_url": null
        }
    ],
    "metadata": {
        "total": null,
        "count": null,
        "static_fields": [],
        "static_total_fields": []
    }
}
```


1.15 upload thirdparty image

Upload a third party image.

1.15.1 Field List

End point fields

Name	Modifier & Defaults	Type	Description
image	Required	APIImage	An image to upload

1.15.2 upload image

To *upload* a third-party image to Taboola, send an HTTP *POST* request:

```
POST /backstage/api/1.0/operations/thirdparty/image
Host: https://backstage.taboola.com
Authorization: Bearer [access-token]
```

1.15.2.1 Example Request: Only Required Fields

```
POST /backstage/api/1.0/operations/thirdparty/image
Host: https://backstage.taboola.com
```

```
Authorization: Bearer [access-token]
```

```
{
    "source": "GETTY_IMAGES",
    "url":
"https://media.gettyimages.com/photos/vice-presidential-seal-and-empty-podium-picture-id7091
13893?b=1&k=6&m=709113893&s=170x170&h=Lk89iED4atdfS6U-Zmm3IGJh9u-Lr2XlL7pYdWGnoeI=",
     "id": "709113893",
     "cdn_url": null
}
```

The method will return the uploaded image.

1.15.2.2 Example Response

```
200 OK
```

```
{
    "source": "GETTY_IMAGES",
    "url":
"https://media.gettyimages.com/photos/vice-presidential-seal-and-empty-podium-picture-id7091
13893?b=1&k=6&m=709113893&s=170x170&h=Lk89iED4atdfS6U-Zmm3IGJh9u-Lr2XlL7pYdWGnoeI=",
    "id": "709113893",
    "cdn_url":
"http://cdn.taboola.com/libtrc/static/thumbnails/GETTY_IMAGES/CMF/709113893__XezrUa1B.jpg"
}
```