API Documentation

Rev 1/5/17

Contents:

Integration Considerations **1**
- Data Mapping 1
- Middleware 2
- Synchronization 2
- Step 1: Determine the Business Requirements of your Integration 3
- Step 2: Decide What Data to Synchronize 3
- Step 3: Create the Synchronization Script 3

Getting Started with the API **5**
- 360Alumni provides the following REST endpoints and methods: 5
- The API URLs 5
- Get Your API Key 5
- Requesting Data 5

Filtering Responses **12**
- Filtering Campaign Results 12
- Filtering Donation Results 15
- Filtering User Results 18

Creating and Updating Users **21**
- Create a New User with POST /api/users/ 21
- Get an Array of User Records with GET /api/users/ 22
- Get a Specific User's Record with GET /api/users/{id} 23
- Update a User Record with PUT /api/users/{id} 27

Integration Considerations

Data Mapping

You'll need to create a plan for how to keep your data models synchronized. Data must be pulled from 360Alumni's API rather than pushed via an event-driven integration model. So your integration needs to periodically scan the data and provide a change log to act upon.
Once you have decided upon your synchronization model, you need to identify which fields you are going to keep synchronized between the two systems. Use the User Object reference page to help with your mapping. You can also download a copy of the following spreadsheet to use for your mapping:

1. Open the 360Alumni User Data Mapping Template.
2. Select **File > Download as >**, from the open template.
3. Select your preferred format to save the spreadsheet:
    - Excel
    - OpenDocument
    - Comma separated values (CSV)
    - Tab separated values (TSV)
4. The template is sorted by **JSON** field column. You might find it useful to do your mapping by resorting your spreadsheet by the **Preferred Order** column.

Middleware

Another consideration is the middleware that provides the "bridge" between 360Alumni and your database. Middleware provides enforcement of business logic such as conversion of the data format or structures. Another example of middleware tasks is adding a layer of protection so bad data or glitches can not create chaos in the destination system. An example of middleware is Zapier, which is a middleware "clearinghouse" - offering prepackaged middleware "scripts" for thousands of software applications. 360Alumni has a privately available "Zap" which can be used by our clients who run Salesforce.

Synchronization

One method of integrating your application with 360Alumni is via a polling and synchronization method. This involves checking the database periodically for changes and then updating records accordingly. A polling and sync integration works by running a script that checks for updates and then will update a target based on changes. You can make these integrations two-way by simply applying the business logic you create to both checks.

The following integration example uses a CRM to show how a sync integration can work. Awesome University has a CRM that they use to keep track of their alumni and ensure they can keep track of their donations. They have recently become a partner with 360Alumni and would like to ensure that their alumni database in their CRM and 360Alumni remains in sync. In order to accomplish this they will be setting up a sync with 360Alumni using the "Users" endpoint of the API.

- Step 1: Determine the Business Requirements of your Integration
- Step 2: Decide What Data to Synchronize
- Step 3: Create the Synchronization Script

Step 1: Determine the Business Requirements of your Integration

For Awesome University's integration they have determined they want to perform a two way sync with 360Alumni and the contacts in their CRM system. They determined this by finding out the business requirements by talking with stakeholders in their organization. This involved a short series of meetings where they discussed what data they wanted to keep in sync with 360Alumni. They decided upon a two way sync because it would allow them to maintain their alumni directory both organically and still be able to manage it through their CRM.

Step 2: Decide What Data to Synchronize

Awesome University had to decide what parts of their contact data they wanted to keep in sync, so they referred to 360Alumni's field list and settled upon the first name, last name, email, and phone number fields only. They did this because they had only maintained these pieces of contact information in their CRM so they wanted to get them updated as users provided updates. They could have included additional fields but chose a lean approach for phase 1 of their integration.

Step 3: Create the Synchronization Script

Awesome University maintains a couple of in house servers so they decided their script will be written in node.js and run once a day at 1 AM on one of their linux environments. Their script will simply look at records that were changed in each system and then update or create records on each of the respective environments. This also allows Awesome University to put in any transformations they need.

Getting Started with the API

360Alumni provides the following REST endpoints and methods:

- /api/campaigns/
- GET method
- /api/campaigns/{id}
- GET method
- /api/donations/
- GET method
- /api/users/
- GET and POST methods
- /api/users/{id}
- GET and PUT methods

The API URLs

Use one of the following URLs to open the API reference.

Test system	https://demo.s.360alumni.com/api/documentation
Your system	https://<<your-system>>.360alumni.com/api/documentation

Get Your API Key

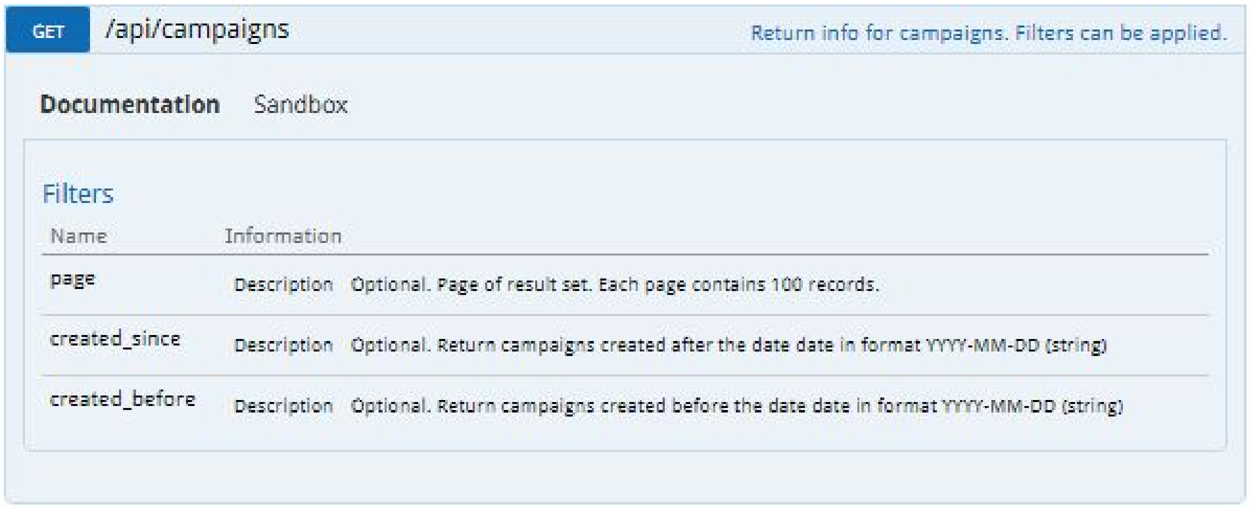
Before you can use the API you'll need to get your API key, which you use to authenticate with the API. Your account manager at 360Alumni will coordinate with our Client Integrations Team to create an API key for your organization. Once you have the key, connecting to the 360Alumni API is easy. For each API request, simply include your API key as the value in an HTTP header named **x-api-key.** See the examples for details.

Requesting Data

The following steps walk you through the basics to get started using the API.

1. Click an API method to display it's reference documentation. For example, click **GET /api/campaigns/**.

An API Reference Page



GET /api/campaigns Return info for campaigns. Filters can be applied.

Filters

Name	Information	
page	Description	Optional. Page of result set. Each page contains 100 records.
created_since	Description	Optional. Return campaigns created after the date date in format YYYY-MM-DD (string)
created_before	Description	Optional. Return campaigns created before the date date in format YYYY-MM-DD (string)

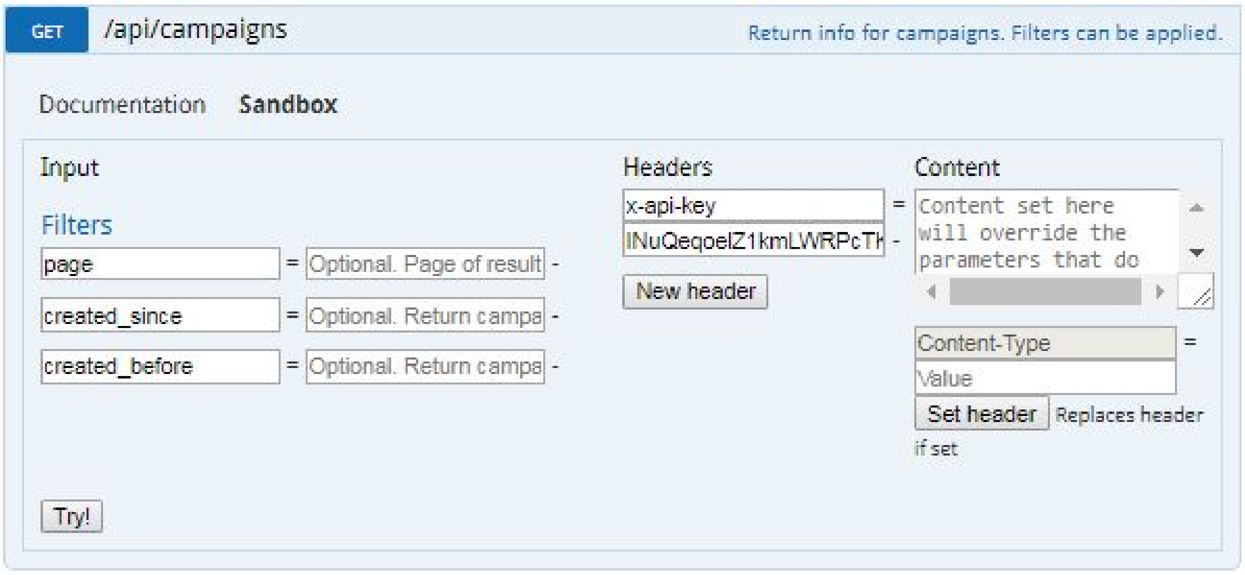
Note, for a **GET /api/campaigns/** request the **page** parameter indicates which set of results to return. The default **page** value is 1 representing the first 100 results.

A page represents up to 100 results returned in a JSON array.

2. Click **Sandbox** to display a form for entering values and trying the request.

3. Enter your API key in an HTTP request header.
    1. Use the header name **x-api-key**.
    2. Use the value obtained from your 360Alumni representative.

An API Sandbox Test Form



GET /api/campaigns

Return info for campaigns. Filters can be applied.

Documentation **Sandbox**

Input

Filters

page	=	Optional. Page of result
created_since	=	Optional. Return campa
created_before	=	Optional. Return campa

Headers

x-api-key = INuQeqoelZ1kmLWRPcTk

Content

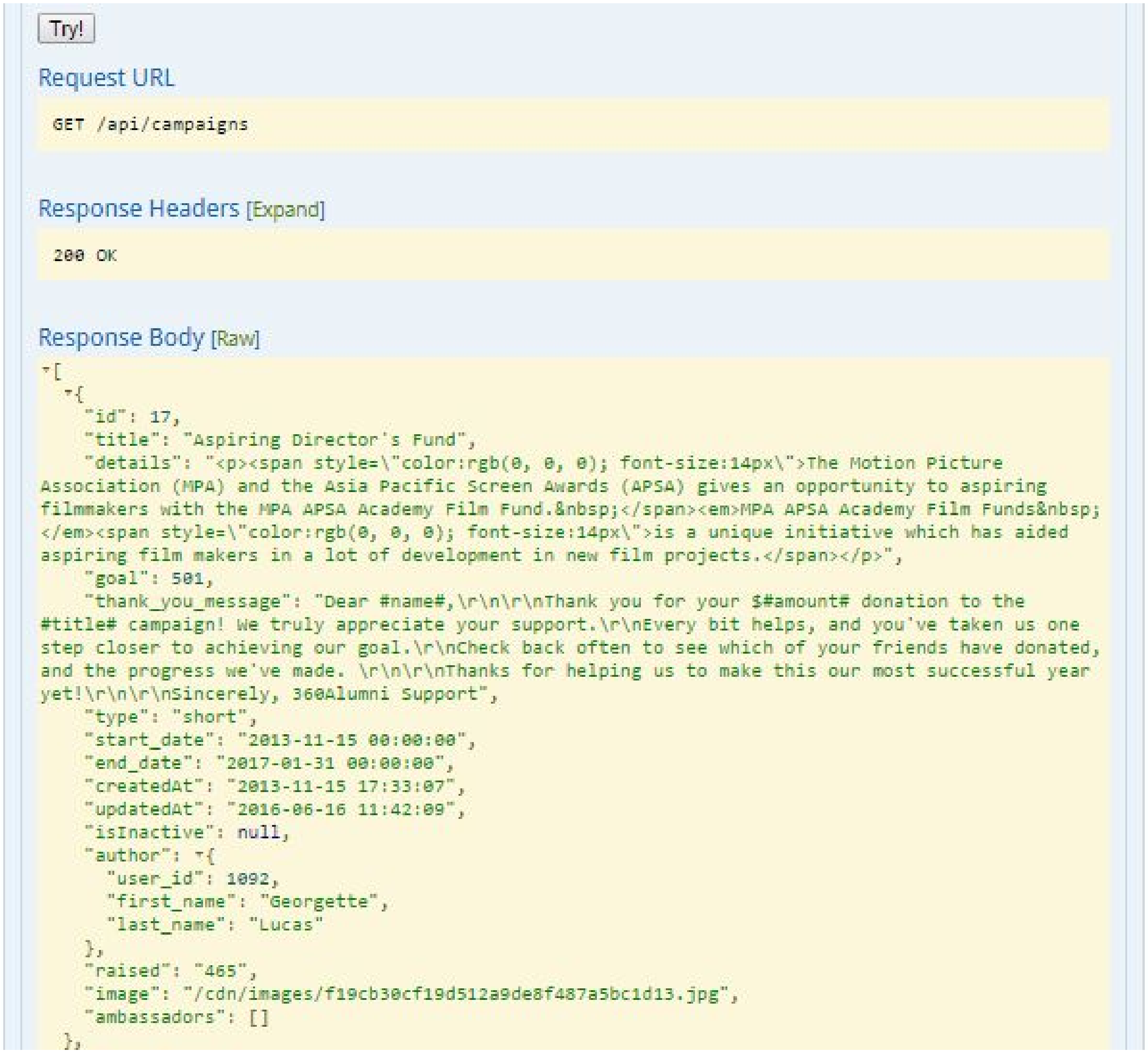
```
Content set here
will override the
parameters that do
```

Content-Type = Value

Set header Replaces header if set

4. Click **Try!** to execute the API request.

5. The display shows results such as the following:



Try!

Request URL

```
GET /api/campaigns
```

Response Headers [Expand]

```
200 OK
```

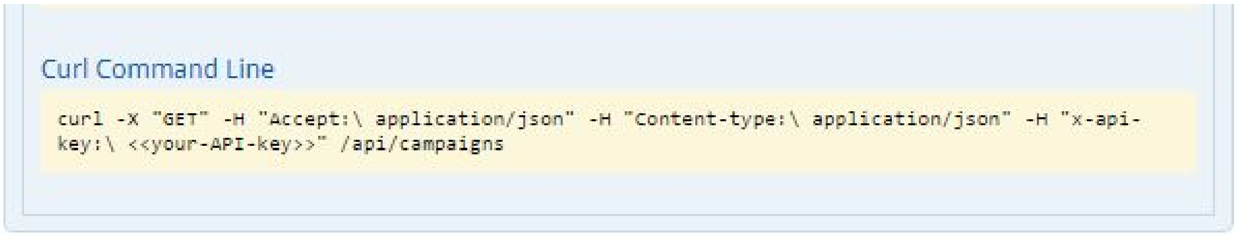
Response Body [Raw]

```
[
  {
    "id": 17,
    "title": "Aspiring Director's Fund",
    "details": "<p><span style=\"color:rgb(0, 0, 0); font-size:14px\">The Motion Picture
Association (MPA) and the Asia Pacific Screen Awards (APSA) gives an opportunity to aspiring
filmmakers with the MPA APSA Academy Film Fund.&nbsp;</span><em>MPA APSA Academy Film Funds&nbsp;
</em><span style=\"color:rgb(0, 0, 0); font-size:14px\">is a unique initiative which has aided
aspiring film makers in a lot of development in new film projects.</span></p>",
    "goal": 501,
    "thank_you_message": "Dear #name#,\r\n\r\nThank you for your $#amount# donation to the
#title# campaign! We truly appreciate your support.\r\nEvery bit helps, and you've taken us one
step closer to achieving our goal.\r\nCheck back often to see which of your friends have donated,
and the progress we've made. \r\n\r\nThanks for helping us to make this our most successful year
yet!\r\n\r\nSincerely, 360Alumni Support",
    "type": "short",
    "start_date": "2013-11-15 00:00:00",
    "end_date": "2017-01-31 00:00:00",
    "createdAt": "2013-11-15 17:33:07",
    "updatedAt": "2016-06-16 11:42:09",
    "isInactive": null,
    "author": {
      "user_id": 1092,
      "first_name": "Georgette",
      "last_name": "Lucas"
    },
    "raised": "465",
    "image": "/cdn/images/f19cb30cf19d512a9de8f487a5bc1d13.jpg",
    "ambassadors": []
  },
```

.

.

.



Curl Command Line

```
curl -X "GET" -H "Accept:\ application/json" -H "Content-type:\ application/json" -H "x-api-
key:\ <<your-API-key>>" /api/campaigns
```

Note the following about the results:

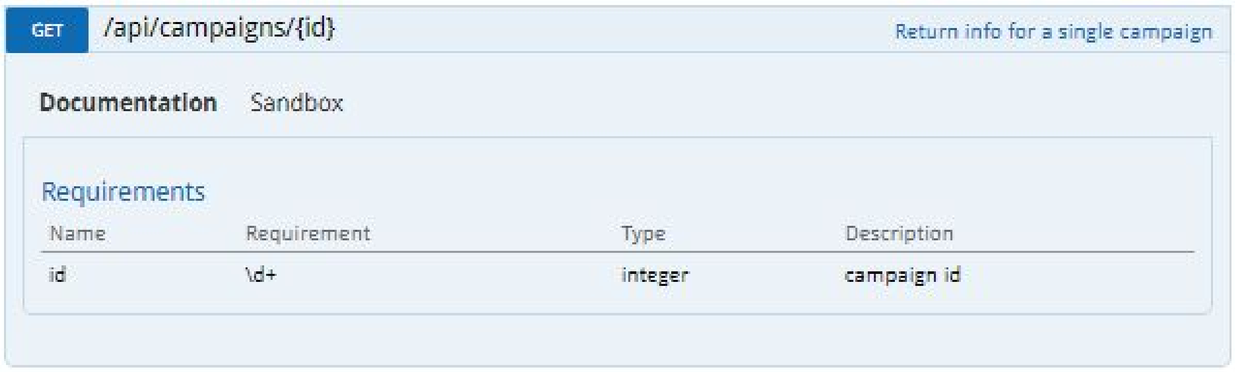
- **Request URL:** Shows what executed.
- **Response Headers:** Contains messages about the success or failure of the request.
- **Response Body:** Contains a JSON array of the results, in this case an array of Campaign Objects. Other API requests use the Donation Object or the User Object.
- **Curl Command Line:** Contains a form of the request that you can copy and paste to execute at the command line or in scripts.

```
curl -X "GET" -H "Accept:\ application/json" -H "Content-type:\ application/json" -H
"x-api-key:\ <<your-API-key>>" /api/campaigns
```

6. Each campaign object in the returned array is identified with an **id** value. Note the first object in the array above contains the id value 17.

7. To get a single campaign object, click **GET /api/campaigns/{id}**.
This API request requires an integer **id** parameter that represents the compaign identifier.

The Reference Page for GET /api/campaigns/{id}



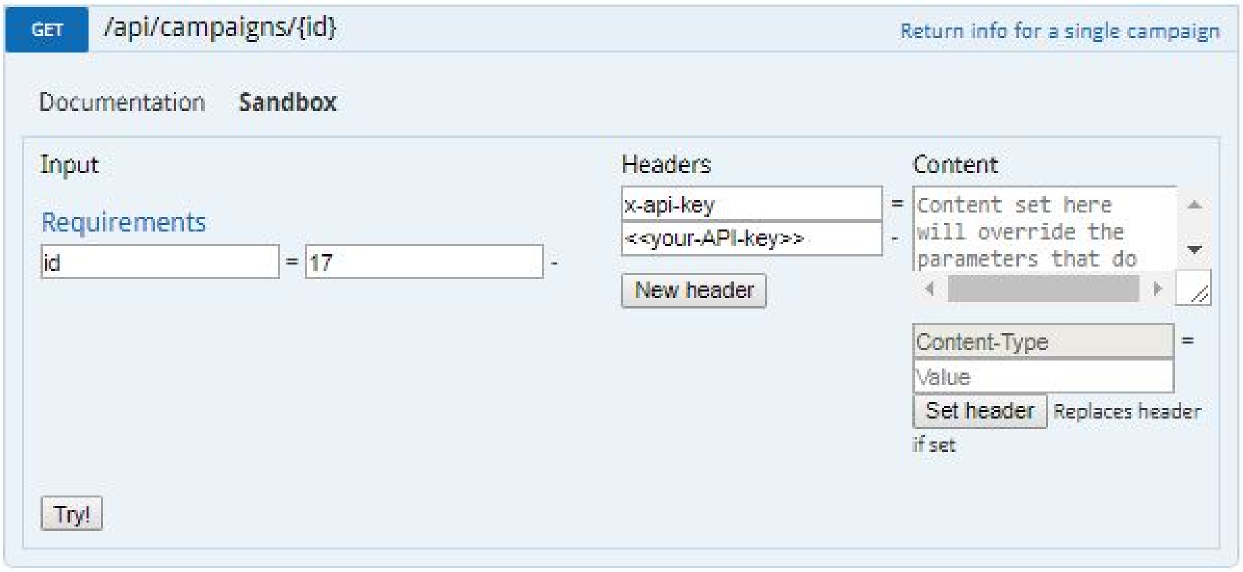
Name	Requirement	Type	Description
id	\d+	integer	campaign id

8. Click **Sandbox** and enter the following to execute the API request:

A. Enter your API key in an HTTP request header.
B. Enter an **id** parameter value representing the campaign you want to get.

C. Click **Try!** to execute the API request.

The Sandbox Form for a GET /api/campaigns/{id} Request

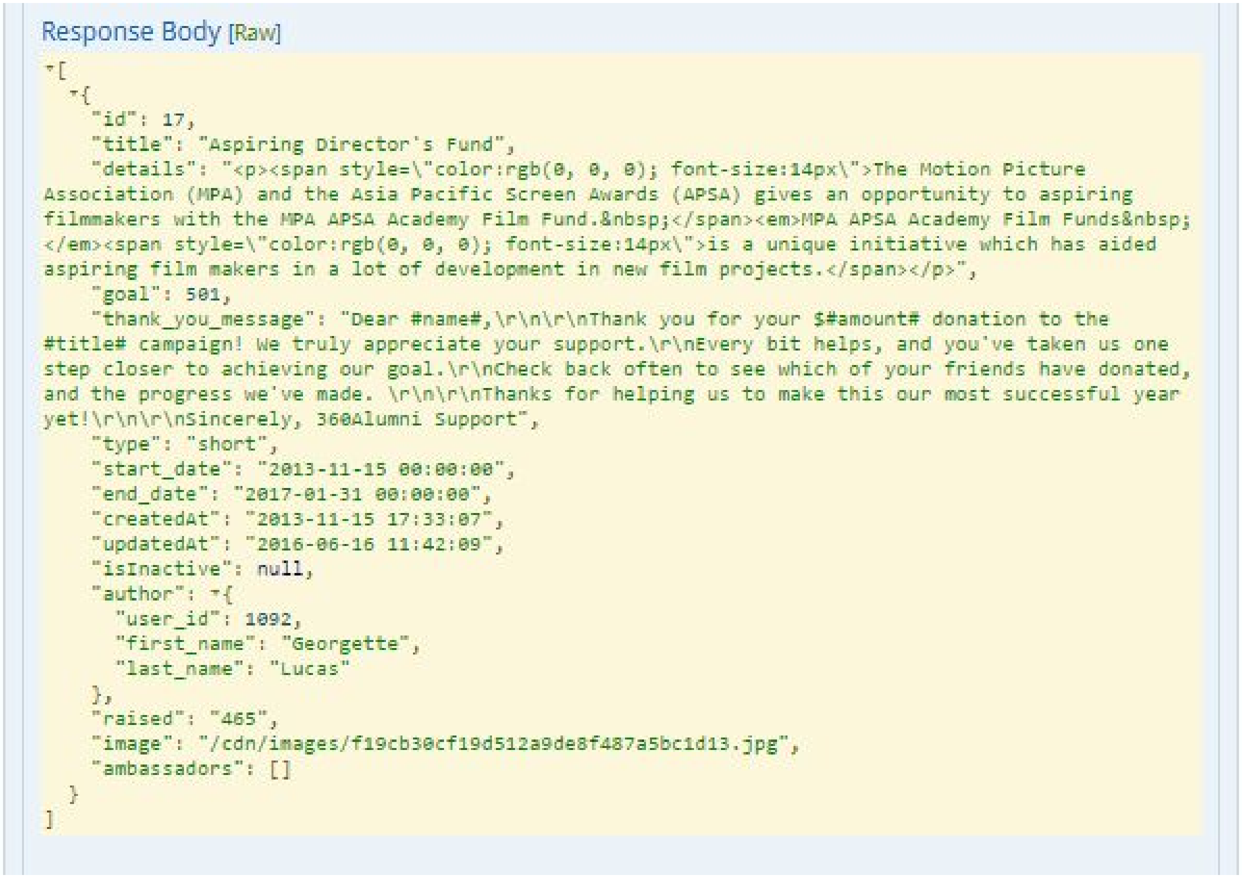


id	=	17
x-api-key	=	<<your-API-key>>
Content-Type	=	Value

Content set here will override the parameters that do

9. The **Response Body** shows the results in a formatted JSON Campaign Object.

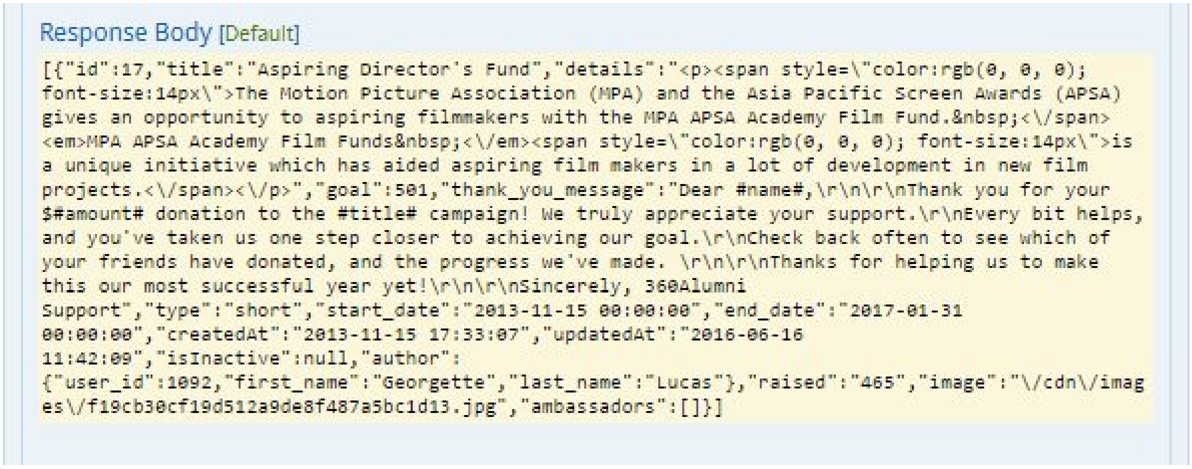
The Response Body, Formatted


Response Body [Raw]

```
[
  {
    "id": 17,
    "title": "Aspiring Director's Fund",
    "details": "<p><span style=\"color:rgb(0, 0, 0); font-size:14px\">The Motion Picture
Association (MPA) and the Asia Pacific Screen Awards (APSA) gives an opportunity to aspiring
filmmakers with the MPA APSA Academy Film Fund.&nbsp;</span><em>MPA APSA Academy Film Funds&nbsp;
</em><span style=\"color:rgb(0, 0, 0); font-size:14px\">is a unique initiative which has aided
aspiring film makers in a lot of development in new film projects.</span></p>",
    "goal": 501,
    "thank_you_message": "Dear #name#,\r\n\r\nThank you for your $#amount# donation to the
#title# campaign! We truly appreciate your support.\r\nEvery bit helps, and you've taken us one
step closer to achieving our goal.\r\nCheck back often to see which of your friends have donated,
and the progress we've made. \r\n\r\nThanks for helping us to make this our most successful year
yet!\r\n\r\nSincerely, 360Alumni Support",
    "type": "short",
    "start_date": "2013-11-15 00:00:00",
    "end_date": "2017-01-31 00:00:00",
    "createdAt": "2013-11-15 17:33:07",
    "updatedAt": "2016-06-16 11:42:09",
    "isInactive": null,
    "author": {
      "user_id": 1092,
      "first_name": "Georgette",
      "last_name": "Lucas"
    },
    "raised": "465",
    "image": "/cdn/images/f19cb30cf19d512a9de8f487a5bc1d13.jpg",
    "ambassadors": []
  }
]
```

Click **Raw** to show the JSON object without formatting, suitable for copying and using in code.

The Response Body, Unformatted


Response Body [Default]

```
[{"id":17,"title":"Aspiring Director's Fund","details":"<p><span style=\"color:rgb(0, 0, 0);
font-size:14px\">The Motion Picture Association (MPA) and the Asia Pacific Screen Awards (APSA)
gives an opportunity to aspiring filmmakers with the MPA APSA Academy Film Fund.&nbsp;<\/span>
<em>MPA APSA Academy Film Funds&nbsp;<\/em><span style=\"color:rgb(0, 0, 0); font-size:14px\">is
a unique initiative which has aided aspiring film makers in a lot of development in new film
projects.<\/span><\/p>","goal":501,"thank_you_message":"Dear #name#,\r\n\r\nThank you for your
$#amount# donation to the #title# campaign! We truly appreciate your support.\r\nEvery bit helps,
and you've taken us one step closer to achieving our goal.\r\nCheck back often to see which of
your friends have donated, and the progress we've made. \r\n\r\nThanks for helping us to make
this our most successful year yet!\r\n\r\nSincerely, 360Alumni
Support","type":"short","start_date":"2013-11-15 00:00:00","end_date":"2017-01-31
00:00:00","createdAt":"2013-11-15 17:33:07","updatedAt":"2016-06-16
11:42:09","isInactive":null,"author":
{"user_id":1092,"first_name":"Georgette","last_name":"Lucas"},"raised":"465","image":"\/cdn\/imag
es\/f19cb30cf19d512a9de8f487a5bc1d13.jpg","ambassadors":[]}]
```

Filtering Responses

Some GET methods provide optional filter parameters you can include to narrow API results based on various criteria. The following sections show how to use some filter parameters to get data on campaigns, donations, and users. This page contains the following topics:

- Filtering Campaign Results
- Filtering Donation Results
- Filtering User Results

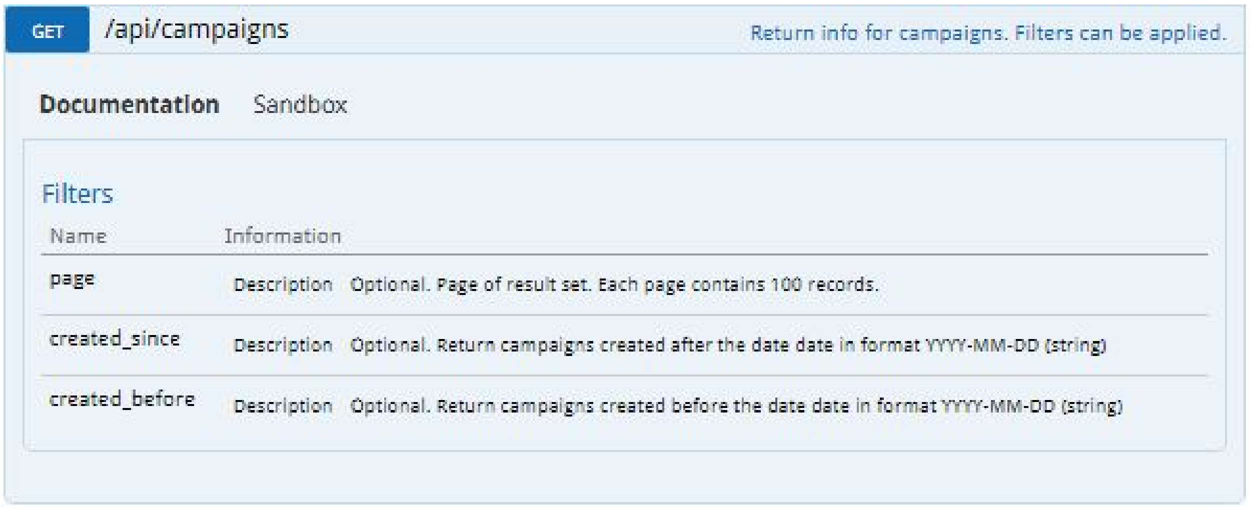
Use one of the following URLs to open the API reference.

Test system	https://demo.s.360alumni.com/api/documentation
Your system	https://<<your-system>>.360alumni.com/api/documentation

Filtering Campaign Results

The following steps walk you through using a filter parameter to get data on campaigns.

1. Click the **GET /api/campaigns/** method.
   Optional filter parameters include **created_since** and **created_before.** Use these parameters to narrow the campaigns returned to within a specific time frame.



GET /api/campaigns Return info for campaigns. Filters can be applied.

Documentation Sandbox

Filters

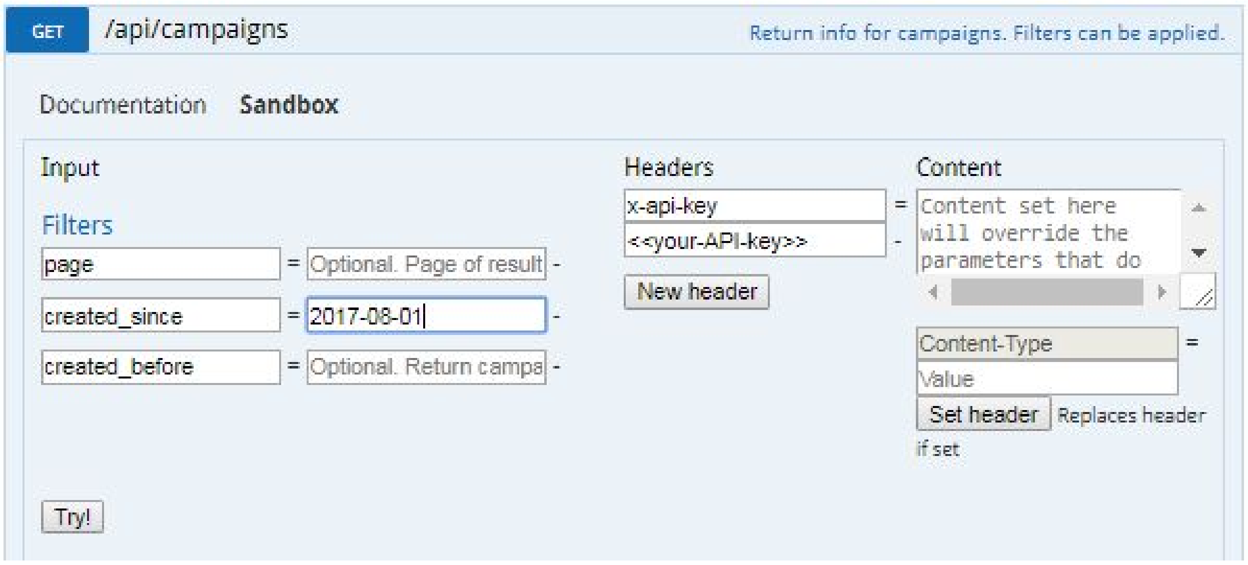
Name	Information	
page	Description	Optional. Page of result set. Each page contains 100 records.
created_since	Description	Optional. Return campaigns created after the date date in format YYYY-MM-DD (string)
created_before	Description	Optional. Return campaigns created before the date date in format YYYY-MM-DD (string)

2. Click **Sandbox** to display the form for entering values and trying the request. For this example we'll narrow the results to those campaigns created since August 1, 2017.

A. Enter your API key in an HTTP request header.

B. Enter the value **2017-08-01** for the **created_since** parameter.

C. Click **Try!** to execute the API request.



GET /api/campaigns Return info for campaigns. Filters can be applied.

Documentation Sandbox

Input

Filters

page	=	Optional. Page of result
created_since	=	2017-08-01
created_before	=	Optional. Return campa

Headers

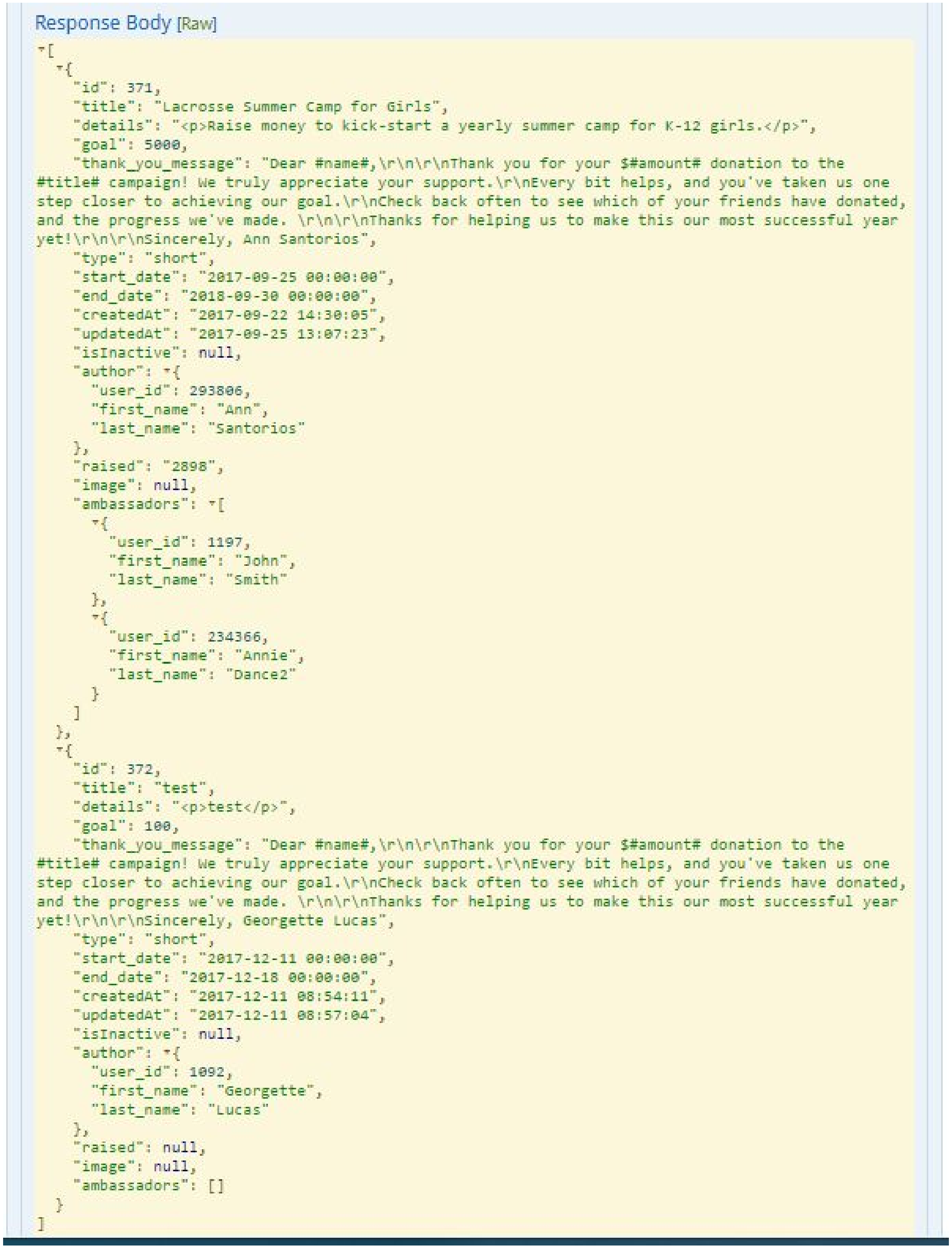
x-api-key = <<your-API-key>>

Content

Content set here will override the parameters that do

Content-Type = Value

3. The **Response Body** shows the JSON results.
Note the values for the **createdAt** field are more recent than the value input for **created_since.**


Response Body [Raw]

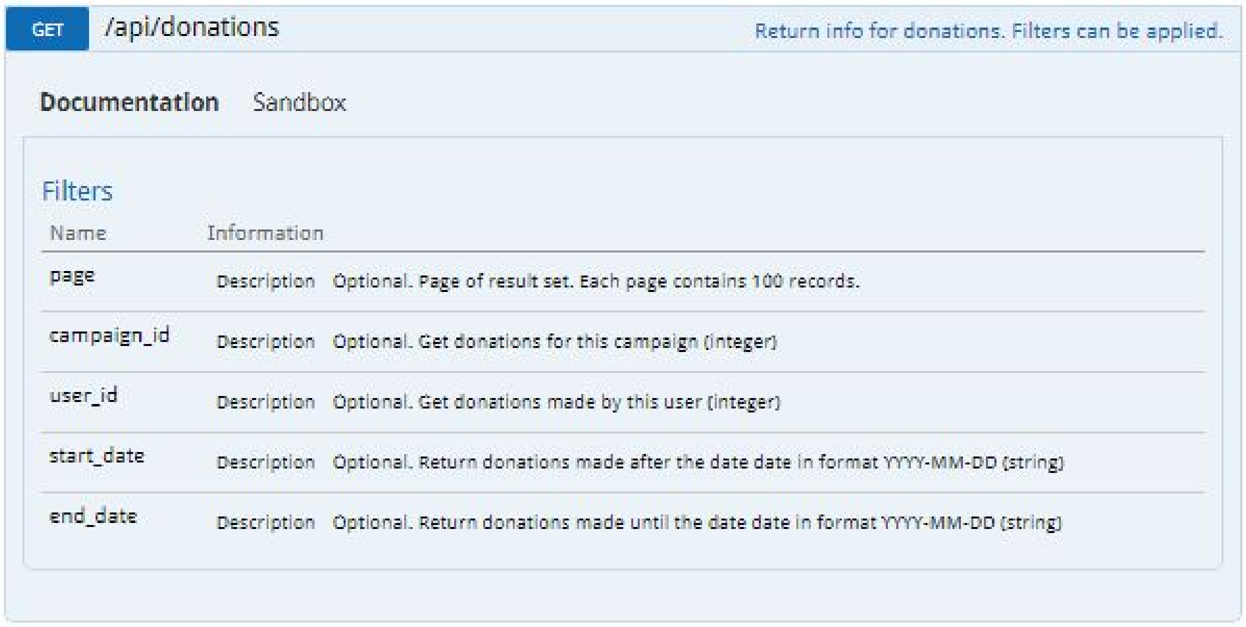
```
[
  {
    "id": 371,
    "title": "Lacrosse Summer Camp for Girls",
    "details": "<p>Raise money to kick-start a yearly summer camp for K-12 girls.</p>",
    "goal": 5000,
    "thank_you_message": "Dear #name#,\r\n\r\nThank you for your $#amount# donation to the
#title# campaign! We truly appreciate your support.\r\nEvery bit helps, and you've taken us one
step closer to achieving our goal.\r\nCheck back often to see which of your friends have donated,
and the progress we've made. \r\n\r\nThanks for helping us to make this our most successful year
yet!\r\n\r\nSincerely, Ann Santorios",
    "type": "short",
    "start_date": "2017-09-25 00:00:00",
    "end_date": "2018-09-30 00:00:00",
    "createdAt": "2017-09-22 14:30:05",
    "updatedAt": "2017-09-25 13:07:23",
    "isInactive": null,
    "author": {
      "user_id": 293806,
      "first_name": "Ann",
      "last_name": "Santorios"
    },
    "raised": "2898",
    "image": null,
    "ambassadors": [
      {
        "user_id": 1197,
        "first_name": "John",
        "last_name": "Smith"
      },
      {
        "user_id": 234366,
        "first_name": "Annie",
        "last_name": "Dance2"
      }
    ]
  },
  {
    "id": 372,
    "title": "test",
    "details": "<p>test</p>",
    "goal": 100,
    "thank_you_message": "Dear #name#,\r\n\r\nThank you for your $#amount# donation to the
#title# campaign! We truly appreciate your support.\r\nEvery bit helps, and you've taken us one
step closer to achieving our goal.\r\nCheck back often to see which of your friends have donated,
and the progress we've made. \r\n\r\nThanks for helping us to make this our most successful year
yet!\r\n\r\nSincerely, Georgette Lucas",
    "type": "short",
    "start_date": "2017-12-11 00:00:00",
    "end_date": "2017-12-18 00:00:00",
    "createdAt": "2017-12-11 08:54:11",
    "updatedAt": "2017-12-11 08:57:04",
    "isInactive": null,
    "author": {
      "user_id": 1092,
      "first_name": "Georgette",
      "last_name": "Lucas"
    },
    "raised": null,
    "image": null,
    "ambassadors": []
  }
]
```

Filtering Donation Results

The following steps walk you through using filter parameters to get data on donations.

1. Click the **GET /api/donations/** method. Note the following:

Optional parameters to filter the results include **campaign_id, user_id, start_date,** and **end_date**.



GET /api/donations Return info for donations. Filters can be applied.

Documentation Sandbox

Filters

Name	Information	
page	Description	Optional. Page of result set. Each page contains 100 records.
campaign_id	Description	Optional. Get donations for this campaign (integer)
user_id	Description	Optional. Get donations made by this user (integer)
start_date	Description	Optional. Return donations made after the date date in format YYYY-MM-DD (string)
end_date	Description	Optional. Return donations made until the date date in format YYYY-MM-DD (string)

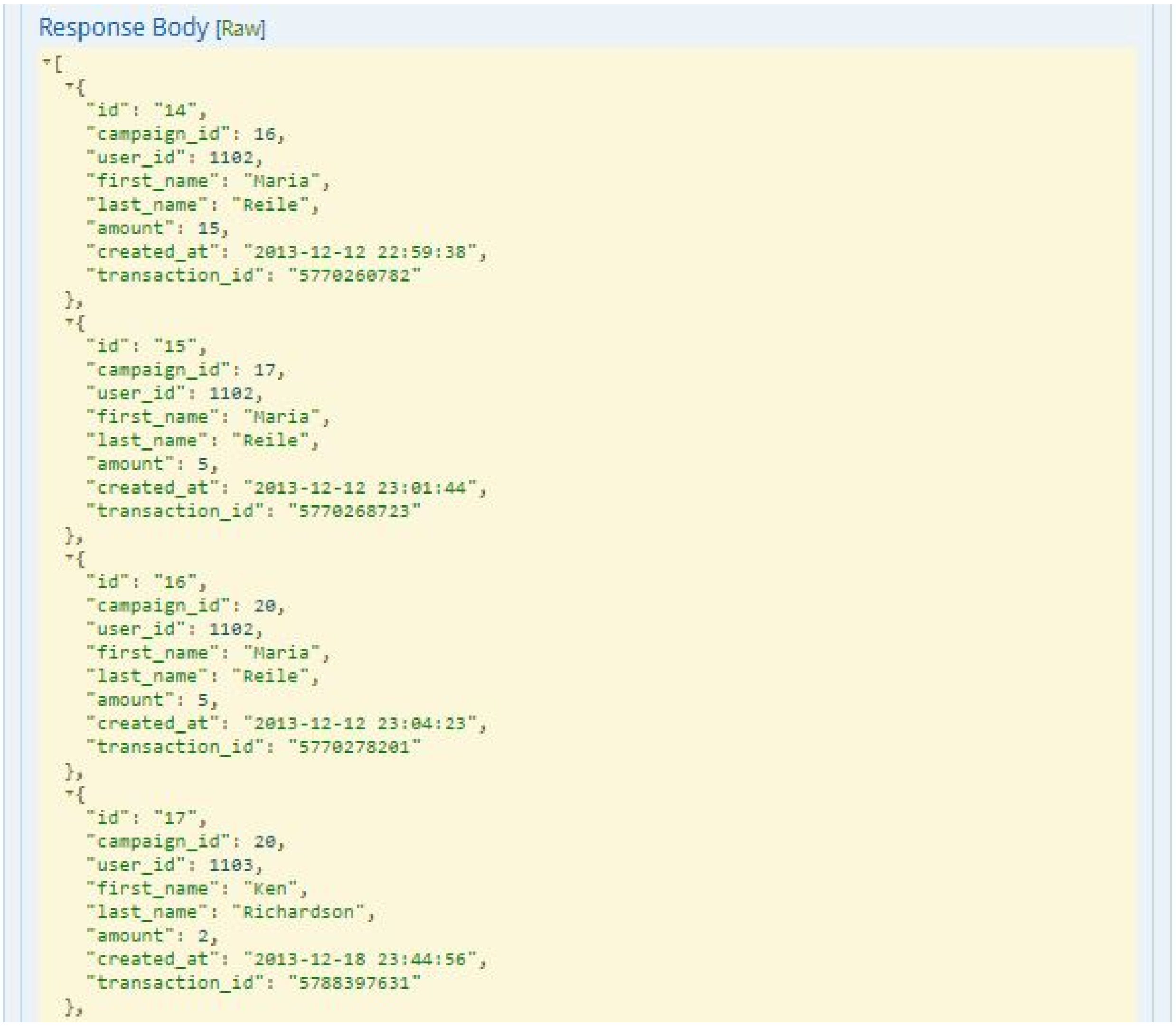
2. Click **Sandbox** to display the form for entering values and trying requests.

3. First we'll request all the donations.
   A. Enter your API key in an HTTP request header.
   B. Click **Try!** to execute the API request.

4. The **Response Body** contains a JSON array of Donation Objects that includes all donations. The following image shows a few of these results. Note the following:

   - Donation object fields correspond to the filter parameters **campaign_id** and **user_id** for **GET /api/donations/** requests.

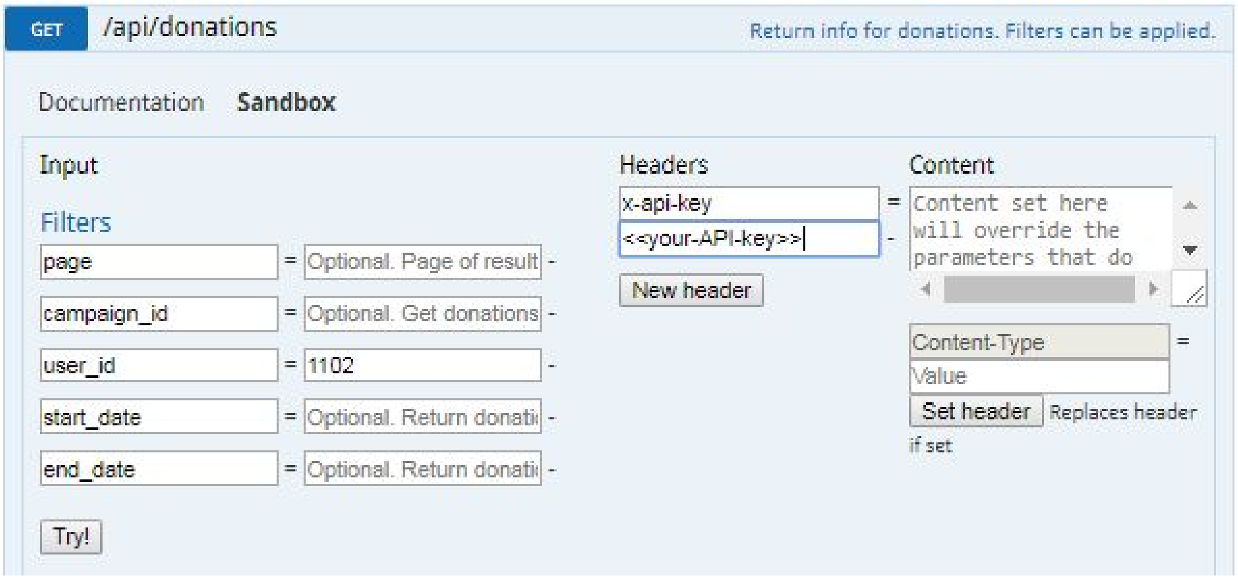
- The **created_at** field determines the results of a **GET /api/donations/** request if you use the **start_date**or **end_date** filter parameters.



```
[
  {
    "id": "14",
    "campaign_id": 16,
    "user_id": 1102,
    "first_name": "Maria",
    "last_name": "Reile",
    "amount": 15,
    "created_at": "2013-12-12 22:59:38",
    "transaction_id": "5770260782"
  },
  {
    "id": "15",
    "campaign_id": 17,
    "user_id": 1102,
    "first_name": "Maria",
    "last_name": "Reile",
    "amount": 5,
    "created_at": "2013-12-12 23:01:44",
    "transaction_id": "5770268723"
  },
  {
    "id": "16",
    "campaign_id": 20,
    "user_id": 1102,
    "first_name": "Maria",
    "last_name": "Reile",
    "amount": 5,
    "created_at": "2013-12-12 23:04:23",
    "transaction_id": "5770278201"
  },
  {
    "id": "17",
    "campaign_id": 20,
    "user_id": 1103,
    "first_name": "Ken",
    "last_name": "Richardson",
    "amount": 2,
    "created_at": "2013-12-18 23:44:56",
    "transaction_id": "5788397631"
  },
```

5. The next request filters donations by **user_id.**

   A. Enter 1102 for the **user_id** value.
   B. Click **Try!** to execute the API request.

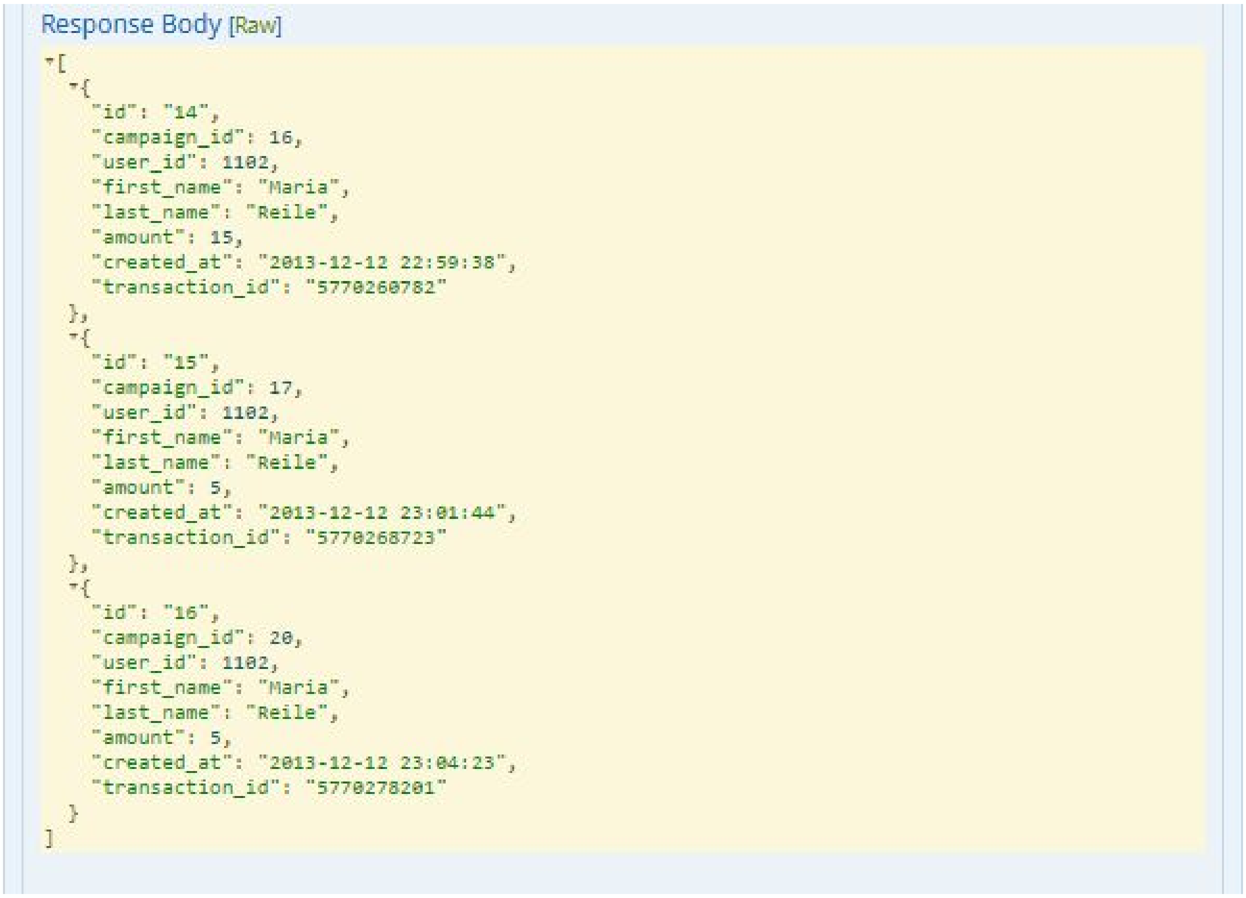


page	=	Optional. Page of result
campaign_id	=	Optional. Get donations
user_id	=	1102
start_date	=	Optional. Return donati
end_date	=	Optional. Return donati

x-api-key	=	<<your-API-key>>
Content-Type	=	Value

```
Content set here
will override the
parameters that do
```

6. Note the results contain a few donation objects.

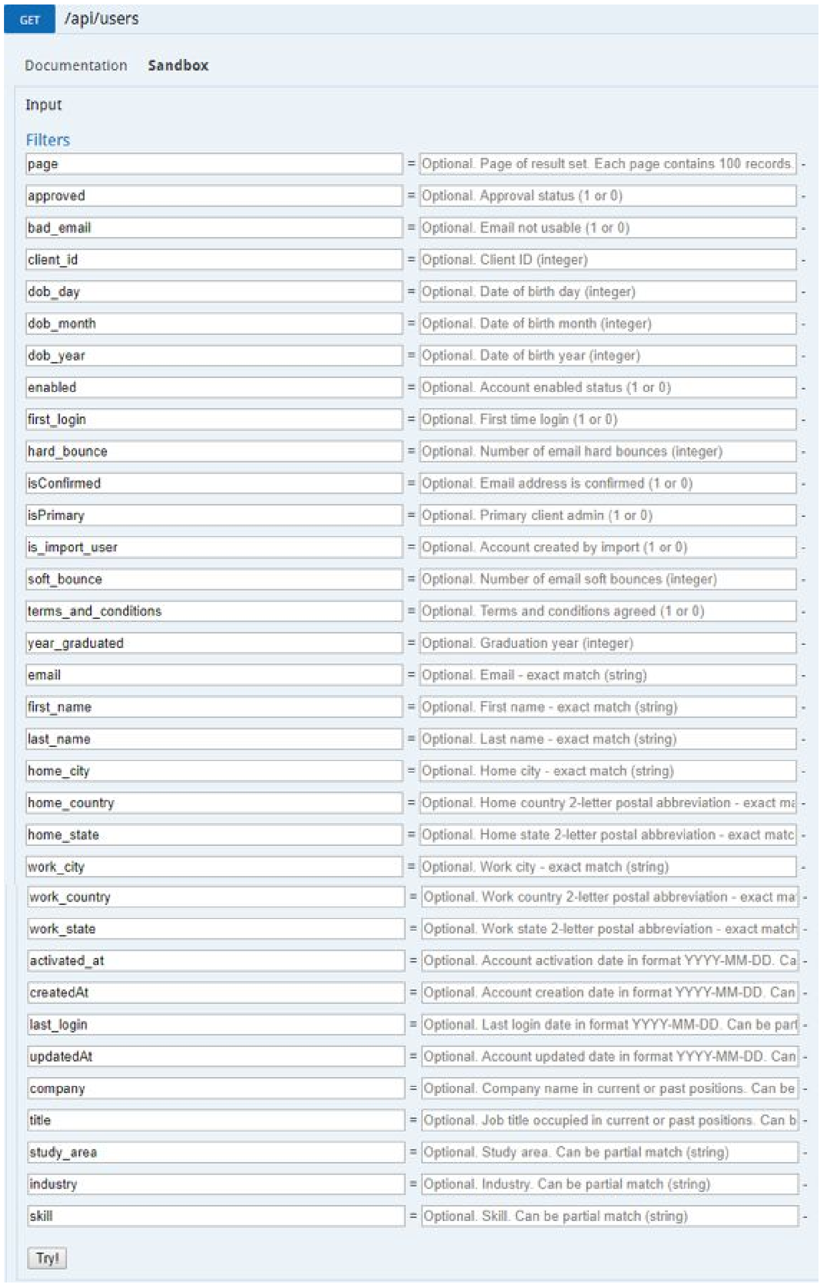


```
[
  {
    "id": "14",
    "campaign_id": 16,
    "user_id": 1102,
    "first_name": "Maria",
    "last_name": "Reile",
    "amount": 15,
    "created_at": "2013-12-12 22:59:38",
    "transaction_id": "5770260782"
  },
  {
    "id": "15",
    "campaign_id": 17,
    "user_id": 1102,
    "first_name": "Maria",
    "last_name": "Reile",
    "amount": 5,
    "created_at": "2013-12-12 23:01:44",
    "transaction_id": "5770268723"
  },
  {
    "id": "16",
    "campaign_id": 20,
    "user_id": 1102,
    "first_name": "Maria",
    "last_name": "Reile",
    "amount": 5,
    "created_at": "2013-12-12 23:04:23",
    "transaction_id": "5770278201"
  }
]
```

Filtering User Results

The following steps walk you through using filter parameters to get data on users.

1. Click the **GET /api/users/** method.
2. Click **Sandbox** to display the form for entering values and trying requests.
3. Note there are a great number of optional parameters available to filter the results.



GET /api/users

Documentation Sandbox

Input

Filters

page	=	Optional. Page of result set. Each page contains 100 records.	-
approved	=	Optional. Approval status (1 or 0)	-
bad_email	=	Optional. Email not usable (1 or 0)	-
client_id	=	Optional. Client ID (integer)	-
dob_day	=	Optional. Date of birth day (integer)	-
dob_month	=	Optional. Date of birth month (integer)	-
dob_year	=	Optional. Date of birth year (integer)	-
enabled	=	Optional. Account enabled status (1 or 0)	-
first_login	=	Optional. First time login (1 or 0)	-
hard_bounce	=	Optional. Number of email hard bounces (integer)	-
isConfirmed	=	Optional. Email address is confirmed (1 or 0)	-
isPrimary	=	Optional. Primary client admin (1 or 0)	-
is_import_user	=	Optional. Account created by import (1 or 0)	-
soft_bounce	=	Optional. Number of email soft bounces (integer)	-
terms_and_conditions	=	Optional. Terms and conditions agreed (1 or 0)	-
year_graduated	=	Optional. Graduation year (integer)	-
email	=	Optional. Email - exact match (string)	-
first_name	=	Optional. First name - exact match (string)	-
last_name	=	Optional. Last name - exact match (string)	-
home_city	=	Optional. Home city - exact match (string)	-
home_country	=	Optional. Home country 2-letter postal abbreviation - exact ma	-
home_state	=	Optional. Home state 2-letter postal abbreviation - exact matc	-
work_city	=	Optional. Work city - exact match (string)	-
work_country	=	Optional. Work country 2-letter postal abbreviation - exact ma	-
work_state	=	Optional. Work state 2-letter postal abbreviation - exact match	-
activated_at	=	Optional. Account activation date in format YYYY-MM-DD. Ca	-
createdAt	=	Optional. Account creation date in format YYYY-MM-DD. Can	-
last_login	=	Optional. Last login date in format YYYY-MM-DD. Can be part	-
updatedAt	=	Optional. Account updated date in format YYYY-MM-DD. Can	-
company	=	Optional. Company name in current or past positions. Can be	-
title	=	Optional. Job title occupied in current or past positions. Can b	-
study_area	=	Optional. Study area. Can be partial match (string)	-
industry	=	Optional. Industry. Can be partial match (string)	-
skill	=	Optional. Skill. Can be partial match (string)	-

Try!

4. First we'll request all users that are not yet approved.
   A. Enter your API key in an HTTP request header.
   B. Enter the value **0** for the **approved** field. A **1** represents approved and a **0** represents not approved.
   C. Click **Try!** to execute the API request.

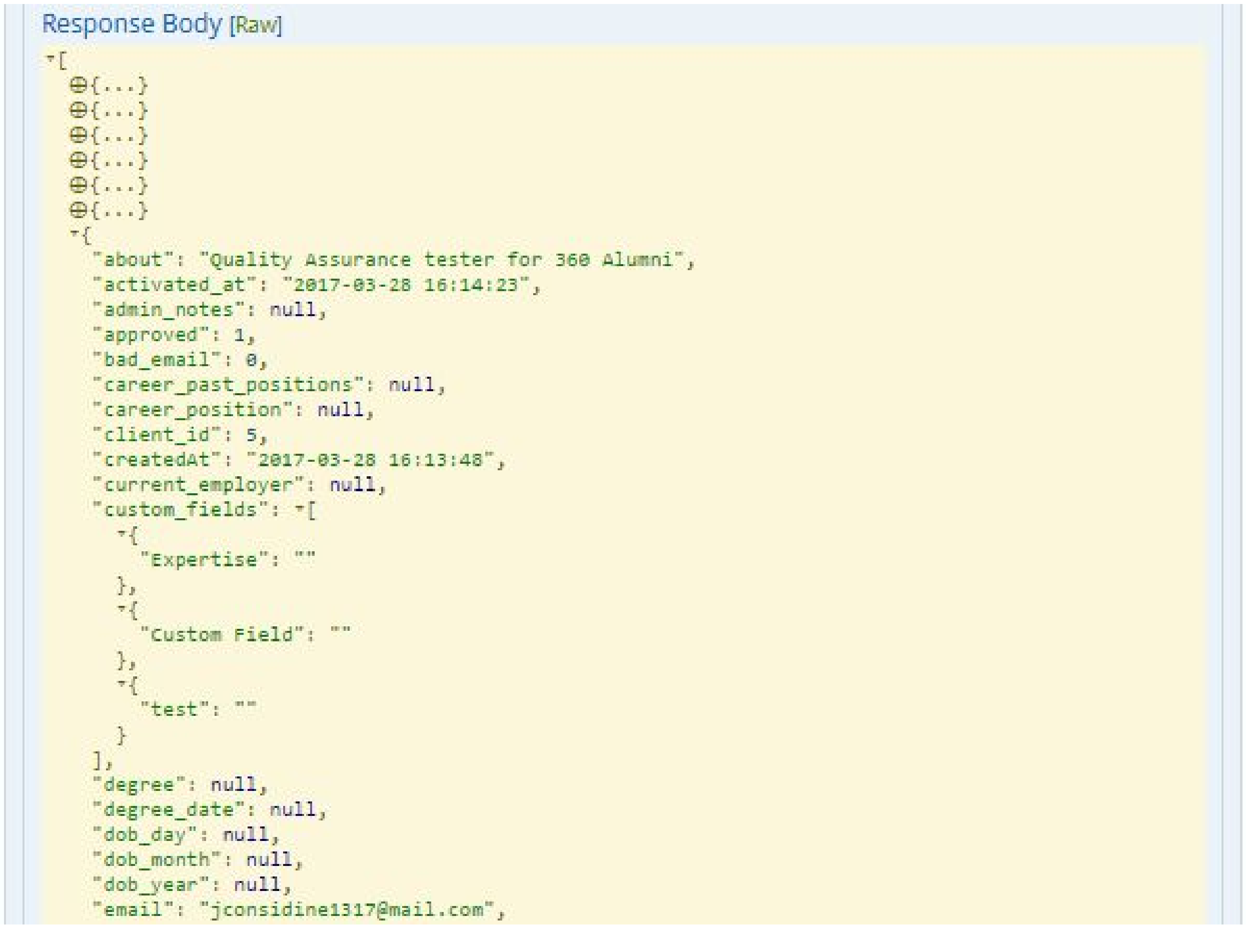
5. The **Response Body** contains a JSON array of user objects where the **approved** field value is 0.

6. The next request filters for all **approved** users that are not **enabled.**

   A. Enter the value **1** for the **approved** field.
   B. Enter the value **0** for the **enabled** field. A **1** represents enabled accounts and a **0** represents accounts that are not enabled.
   C. Click **Try!** to execute the API request.

Note the results contain fewer user records.

8. Click a down arrow in the **Response Body** to collapse the view of an object's fields.



```
[
  ⊕{...}
  ⊕{...}
  ⊕{...}
  ⊕{...}
  ⊕{...}
  ⊕{...}
  {
    "about": "Quality Assurance tester for 360 Alumni",
    "activated_at": "2017-03-28 16:14:23",
    "admin_notes": null,
    "approved": 1,
    "bad_email": 0,
    "career_past_positions": null,
    "career_position": null,
    "client_id": 5,
    "createdAt": "2017-03-28 16:13:48",
    "current_employer": null,
    "custom_fields": [
      {
        "Expertise": ""
      },
      {
        "Custom Field": ""
      },
      {
        "test": ""
      }
    ],
    "degree": null,
    "degree_date": null,
    "dob_day": null,
    "dob_month": null,
    "dob_year": null,
    "email": "jconsidine1317@mail.com",
```

Creating and Updating Users

You can create new users in the system and give them the appropriate access. This is because you have access to more than just the information provided by the 360Alumni GUI and you can perform complex operations such as approving users when they are created.

1. Create a New User with POST /api/users/
2. Get an Array of User Records with GET /api/users/
3. Get a Specific User's Record with GET /api/users/{id}
4. Update a User Record with PUT /api/users/{id}

Use one of the following URLs to open the API reference.

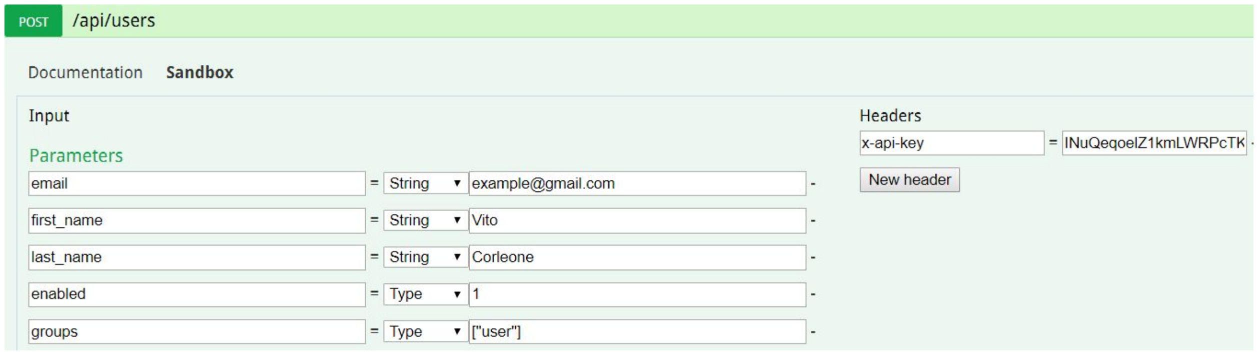
Test system	https://demo.s.360alumni.com/api/documentation
Your system	https://your-system.360alumni.com/api/documentation

Create a New User with POST /api/users/

Follow these steps to create a user.

1. Click the **POST /api/users/** method.
   Required parameters to create a new user include email, first_name and last_name. These parameters as well as all the optional parameters are described in the User Object reference.
2. Click **Sandbox** to display the form for entering values and trying the request.
   a. Enter your API key in an HTTP request header.
   b. Enter a string email address for the email parameter.
   c. Enter a name for the first_name parameter.
   d. Enter a name for the last_name parameter.
   e. Enter 0 | 1 for the enabled parameter.
   f. Enter an array ["admin" | "user"] for the groups parameter.
   g. Click Try! to execute the API request.

The POST /api/users/ Request


POST /api/users

Documentation Sandbox

Input

Parameters

email	=	String	example@gmail.com
first_name	=	String	Vito
last_name	=	String	Corleone
enabled	=	Type	1
groups	=	Type	["user"]

Headers

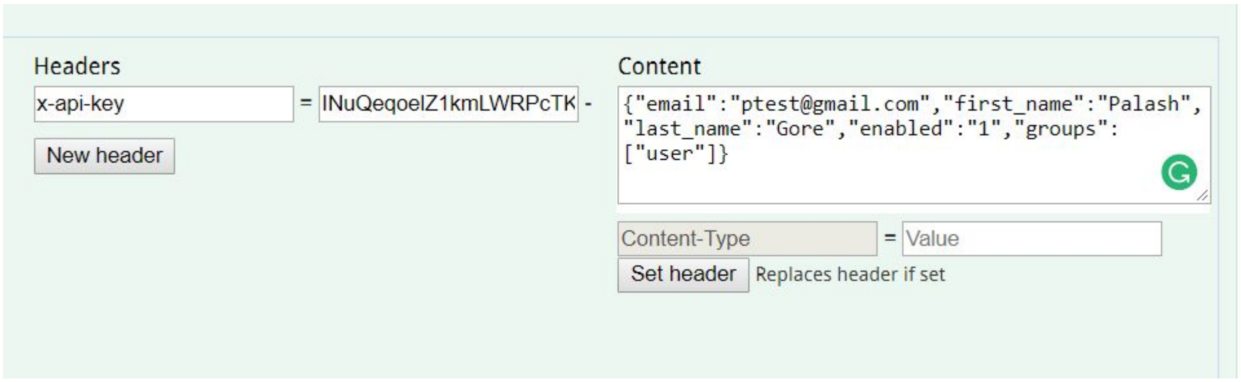
x-api-key = INuQeqoelZ1kmLWRPcTK

New header

3. The **Response Body** shows the JSON results.

Note that the **user_id** field uniquely identifies a user. Make a note of this value to use for identifying the user in later examples.

Note: If the response header is **"400 Bad Request"** try sending the request as a JSON through the Content Textarea


Headers

x-api-key = INuQeqoelZ1kmLWRPcTK

New header

Content

```
{"email":"ptest@gmail.com","first_name":"Palash","last_name":"Gore","enabled":"1","groups":["user"]}
```

Content-Type = Value

Set header Replaces header if set

Get an Array of User Records with GET /api/users/

1. Click the **GET /api/users/** method.
   Note that the **page** parameter is required and the default value is **1**.
2. Click **Sandbox** to display the form for entering values and trying requests.
3. To request all user records:
   a. Enter your API key in an HTTP request header.
   b. Click **Try!** to execute the API request.
4. The **Response Body** contains a JSON array of User Objects.

5. Note that the **user_id** field uniqluy identifes each user. Save one of the **user_id** values to use in later examples.

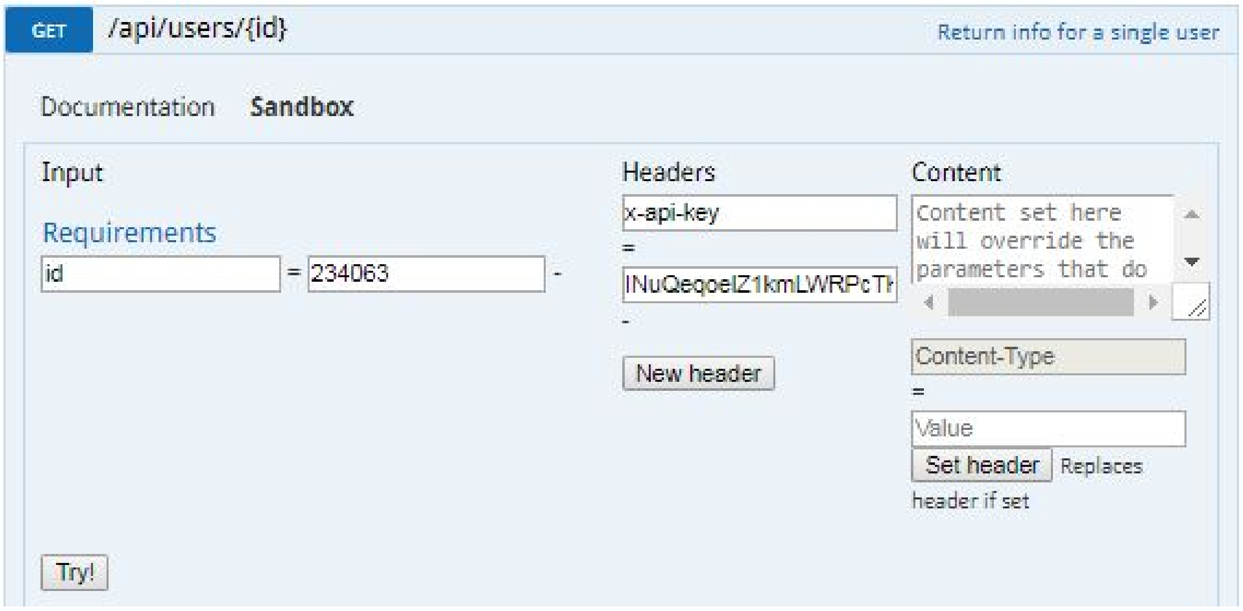
Examples of user_id fields in the Response

```
[
  ...
  {
    ...
    "updatedAt": "2016-04-27 19:20:25",
    "user_id": 1196,
    "user_timezone": null,
    ...
  },
  {
    ...
    "updatedAt": "2016-08-09 18:28:03",
    "user_id": 234063,
    "user_timezone": null,
    ...
  }
  {
    "about": null,
    "activated_at": null,
    "admin_notes": null,
    "approved": 1,
    ...
]
```

Get a Specific User's Record with GET /api/users/{id}

1. Click the **GET /api/users/{id}** method.
   The single required parameter is the **user_id.**
2. Click **Sandbox** to display the form for entering a user ID and trying the request.
   a. Enter your API key in an HTTP request header.
   b. Enter a value for the **id** parameter.
   c. Click **Try!** to execute the API request.

The GET /api/users/{id} Request



id = 234063

x-api-key = INuQeqoeIZ1kmLWRPcTł

Content set here will override the parameters that do

Content-Type = Value

3. The single User Object is returned. Copy and save the JSON object from the Response Body to use in the next example.

A User Object

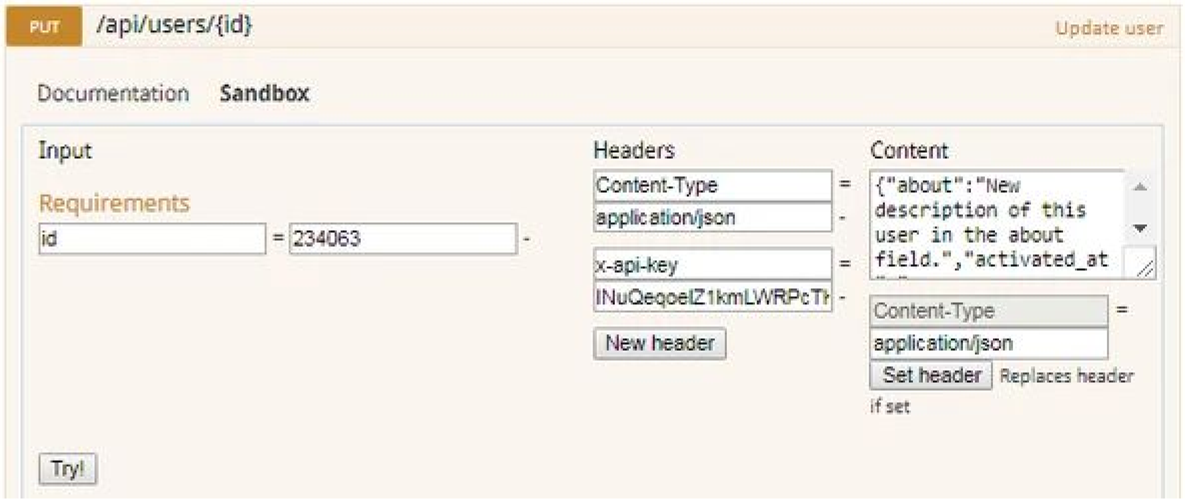
[
  {
    "about": "New description of this user in the about field.",
    "activated_at": "2016-06-22 01:07:19",
    "admin_notes": null,
    "approved": 1,
    "bad_email": 0,
    "client_id": 5,

```
"createdAt": "2016-06-22 01:07:19",
"custom_fields": [],
"dob_day": null,
"dob_month": null,
"dob_year": null,
"email": null,
"emma_member_id": null,
"enabled": 0,
"facebook_link": null,
"first_login": null,
"first_name": "Christina",
"gender": null,
"groups": [
  "user"
],
"hard_bounce": 0,
"home_address1": null,
"home_address2": null,
"home_city": null,
"home_country": null,
"home_geohash": null,
"home_geohash_l1": null,
"home_geohash_l2": null,
"home_geohash_l3": null,
"home_geohash_l4": null,
"home_geohash_l5": null,
"home_geohash_l6": null,
"home_lat": null,
"home_lng": null,
"home_phone": null,
"home_state": null,
"home_zipcode": null,
"image": null,
"inst_id": null,
"instagram_link": null,
"isConfirmed": 0,
"isPrimary": null,
"is_import_user": 0,
"last_activity_at": "2016-06-22 01:07:26",
"last_login": "2016-06-22 01:07:19",
"last_name": "attendee",
"linkedin_link": null,
"maiden_name": null,
"middle_name": null,
"mobile_phone": null,
"name_suffix": null,
"name_title": null,
"nickname": null,
"preferred_email": null,
"preferred_phone": null,
"salt": "",
"secondary_email": [],
"soft_bounce": 0,
"tags": [],
"terms_and_conditions": 1,
"twitter_link": null,
"unsubscribe": 0,
"updatedAt": "2016-08-09 18:28:03",
"user_career_info": [],
"user_id": 234063,
"user_industry": [],
"user_language": [],
"user_skill": [],
"user_study_area": [],
"user_timezone": null,
"username": null,
"website": null,
"work_address1": null,
"work_address2": null,
"work_city": null,
"work_country": null,
"work_geohash": null,
"work_geohash_l1": null,
"work_geohash_l2": null,
"work_geohash_l3": null,
"work_geohash_l4": null,
"work_geohash_l5": null,
"work_geohash_l6": null,
    "work_lat": null,
    "work_lng": null,
    "work_phone": null,
    "work_state": null,
    "work_zipcode": null,
    "year_graduated": null
  }
]
```

Update a User Record with PUT /api/users/{id}

1. Copy the JSON user object from the response in the previous example into an editor and change a few field values.
2. Click the **PUT /api/users/{id}** method. The required parameter is the **user_id.**
3. Click **Sandbox** to display the form for entering information and trying the request.
   a. A JSON user object is required for the content. Copy your changed JSON user object into the **Content**field on the form.
   b. Set the content type.
   c. Enter the **Content-Type** as **application/json** and click **Set header.**
   d. Enter your API key in another HTTP request header.
   e. Click **New header** and enter **x-api-key** and your API key value.
   f. Enter a value for the **id** parameter.
   g. Click **Try!** to execute the API request.

The PUT /api/users/{id} Request



id = 234063

Content-Type = application/json

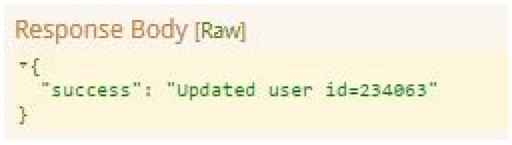
x-api-key = INuQeqoelZ1kmLWRPcTI

```
{"about":"New
description of this
user in the about
field.","activated_at
```

Content-Type = application/json

If the request is successful the **Response Body** indicates the user record was updated.

Response body



```
{
  "success": "Updated user id=234063"
}
```