
eventbrite-sdk-python Documentation

Release 3.3.4

Eventbrite

December 18, 2016

1	eventbrite-sdk-python	3
1.1	Installation from PyPI	3
1.2	Usage	3
1.3	Usage with Frameworks	4
1.4	Versioning	4
2	Installation	5
3	Usage	7
3.1	Example: Get User Info	7
3.2	Example: Pretty print an object	7
4	Cookbook	9
4.1	Get a List of My Draft/Unpublished Events	9
5	Contributing	11
5.1	Types of Contributions	11
5.2	Get Started!	12
5.3	Adding Environment Variables	12
5.4	Pull Request Guidelines	13
5.5	Tips	13
6	Credits	15
6.1	Development Leads	15
6.2	Contributors	15
7	History	17
8	3.3.4 (2016-05-05)	19
9	3.3.3 (2015-08-24)	21
10	3.3.2 (2015-08-17)	23
11	3.2.1 (2015-08-10)	25
12	3.2.0 (2015-07-07)	27
13	3.1.0 (2015-05-11)	29

14	3.0.5 (2015-04-24)	31
15	3.0.4 (2015-03-12)	33
16	3.0.3 (2015-03-02)	35
17	3.0.2 (2015-01-30)	37
18	3.0.1 (2015-01-30)	39
19	3.0.0 (2015-01-28)	41
20	3.0.0-alpha (2014-12-05)	43
21	Indices and tables	45

Contents:

eventbrite-sdk-python

- Official [Eventbrite SDK for Python](#)
- Free software: Apache 2 license
- Full Documentation: <http://eventbrite-sdk-python.readthedocs.org/>
- API Reference: <https://developer.eventbrite.com/docs/>

1.1 Installation from PyPI

```
$ pip install eventbrite
```

If you need to, you can also use *easy_install*:

```
$ easy_install eventbrite
```

1.2 Usage

The Eventbrite Python SDK makes it trivial to interact with the Eventbrite API:

```
>>> from eventbrite import Eventbrite
>>> eventbrite = Eventbrite('my-oauth-token')
>>> user = eventbrite.get_user() # Not passing an argument returns yourself
>>> user['id']
1234567890
>>> user['name']
Daniel Roy Greenfeld
```

You can also specify API endpoints manually:

```
>>> user = eventbrite.get('/users/me')
>>> user['id']
1234567890
>>> user['name']
Daniel Roy Greenfeld
```

[Expansions](#) can be included in a returned GET resource by simply adding the `expand` keyword to the calling method:

```
>>> event = eventbrite.get_event('my-event-id')
>>> 'ticket_classes' in evbobject
False
>>> event = eventbrite.get_event('my-event-id', expand='ticket_classes')
>>> 'ticket_classes' in evbobject
True
```

1.3 Usage with Frameworks

When using Flask, you can convert incoming webhook requests into Eventbrite API objects using the `webhook_to_object()` method:

```
@app.route('/webhook', methods=['POST'])
def webhook():

    # Use the API client to convert from a webhook to an API object
    api_object = eventbrite.webhook_to_object(request)

    # Process the API object
    if api_object.type == 'User':
        do_user_process(api_object)

    if api_object.type == 'Event':
        do_event_process(api_object)

    return ""
```

1.4 Versioning

Because this client interacts with Eventbrite’s third API (a.k.a. APIv3), we are tying our release numbers against it in a modified-semantic system:

- 3.x.x where ‘3’ matches the API version. This will not change until Eventbrite releases a new API version.
- x.0.x where ‘0’ is increased any time there is a significant change to the API that possibly breaks backwards compatibility
- x.x.1 where ‘1’ is increased on any release that does not break backwards compatibility (small, new features, enhancements, bugfixes)

Installation

At the command line:

```
$ easy_install eventbrite
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv eventbrite  
$ pip install eventbrite
```

Usage

To use eventbrite-sdk-python in a project:

```
from eventbrite import Eventbrite
eventbrite = Eventbrite('my-oauth-token')
```

3.1 Example: Get User Info

The following code gets our user object and prints our *id* and *name*.

```
user = eventbrite.get_user() # Not passing an argument returns yourself
print(user['id'])
print(user['name'])
```

This is what gets printed out:

```
1234567890
Daniel Roy Greenfeld
```

3.2 Example: Pretty print an object

Eventbrite objects are dictionaries with extra attributes. Our favorite is *pretty*, which formats their data more legibly:

```
>>> user = eventbrite.get_user() # Not passing an argument returns yourself
>>> print(user.pretty)
{'emails': [{u'email': u'danny@eventbrite.com',
              u'primary': True,
              u'verified': True}],
 u'first_name': u'Daniel',
 u'id': u'1234567890',
 u'last_name': u'Greenfeld',
 u'name': u'Daniel Greenfeld'}
```


4.1 Get a List of My Draft/Unpublished Events

```
from eventbrite import Eventbrite
eventbrite = Eventbrite(MY_OAUTH_TOKEN)

# Get my own User ID
my_id = eventbrite.get_user()['id']

# Get a raw list of events (includes pagination details)
events = eventbrite.event_search(**{'user.id': my_id})

# List the events in draft status
[x for x in events['events'] if x['status'] == 'draft']
```

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given. You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at <https://github.com/eventbrite/eventbrite-sdk-python/issues>.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “feature” is open to whoever wants to implement it.

5.1.4 Write Documentation

eventbrite-sdk-python could always use more documentation, whether as part of the official eventbrite-sdk-python docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at <https://github.com/eventbrite/eventbrite-sdk-python/issues>.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *eventbrite* for local development.

1. Fork the *eventbrite* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/eventbrite.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv eventbrite
$ cd eventbrite/
$ python setup.py develop
$ pip install -r requirements.txt
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 eventbrite tests
$ py.test tests
$ tox
```

To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Adding Environment Variables

In order to run the full test suite, you will need your `USER_ID` and `OAUTH` token from Eventbrite added as environment variables.

In your `.bash_profile` add:

```
# Eventbrite envariables variables
EVENTBRITE_USER_ID=XXXXXXX
EVENTBRITE_OAUTH_TOKEN=XXXXXXX
```

5.4 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, 3.3, and 3.4, and for PyPy. Check https://travis-ci.org/eventbrite/eventbrite-sdk-python/pull_requests and make sure that the tests pass for all supported Python versions.

5.5 Tips

5.5.1 Running a subset of tests

```
$ python -m unittest tests.test_eventbrite
```

5.5.2 Checking test coverage

```
$ make coverage
```

5.5.3 Running integration tests

In order to expedite development, by default these do not run.

1. Get an Eventbrite OAUTH token.
2. Via the Eventbrite website, create an event. Get the Event ID
3. Add those values as environment variables

```
$ export EVENTBRITE_EVENT_ID=XXXXXXXXXX
$ export EVENTBRITE_OAUTH_TOKEN=XXXXXXXXXXXX
```

4. Run the test suite:

```
make test
```

Credits

6.1 Development Leads

- Daniel Greenfeld <danny@eventbrite.com>
- Bartek Ogryczak <bartek@eventbrite.com>

6.2 Contributors

- Piotr Banaszkiewicz <piotr@banaszkiewicz.org>
- Ryan Bagwell <ryan@ryanbagwell.com>
- Bill So <bill@eventbrite.com>
- Jon Kaczynski <jon@generalassemb.ly>
- Dan Moore <dan@moore.cx>

History

3.3.4 (2016-05-05)

- Added new organizers endpoint (thanks tp @mgrdcm)
 - GET /organizers/:id/events/

3.3.3 (2015-08-24)

- Added 3 new user endpoints, thanks to @jon-ga (#29)
 - GET /users/:id/events/
 - GET /users/:id/venues/
 - GET /users/:id/organizers/

3.3.2 (2015-08-17)

- Removed type mapping as it added unnecessary complexity preventing easy management of paginated responses.

3.2.1 (2015-08-10)

- Enabled webhooks
- Fixed ticket definitions in Event creation test
- Set input variable using input argument thanks to Bill So (#27).

3.2.0 (2015-07-07)

- Added new publish and unpublish methods thanks to Ryan Bagwell.
- Eventbrite client now accepts an `eventbrite_api_url` argument.

3.1.0 (2015-05-11)

- Added control over expansion of response. Documentation at <http://www.eventbrite.com/developer/v3/reference/expansions/>

3.0.5 (2015-04-24)

- Removed 'content-type' header from all GET requests. Thank you @xxv for identifying the problem and contributing code.

3.0.4 (2015-03-12)

- Resolved the search result response problem where filtering did not work.

3.0.3 (2015-03-02)

- Fixed import issue with `__version__`. Thank you @meshy and @longjos for identifying the problem.

3.0.2 (2015-01-30)

- Event creation now working.
- Added feature allowing the use of Eventbrite API url at test servers. Should expedite development of tricky post actions.

3.0.1 (2015-01-30)

- Added reverse mapping for `get_event_ticket_class()` method.
- Added `events` mapping to provide GET access to the Event endpoint.
- Removed several deprecated JSON mappings.

3.0.0 (2015-01-28)

- Initial release of 3.0.0 client

3.0.0-alpha (2014-12-05)

- Inception

Indices and tables

- `genindex`
- `modindex`
- `search`