Superscription Documentation

Release 0.1.2

Shrikant Joshi

March 31, 2014

Contents

Contents:

CHAPTER 1

Superscription

Superscriptions: A (super-)thin Python2.7 wrapper around the Superfeedr PubSubHubbub API.

It uses the excellent `requests` module by Kenneth Reitz for interacting with the Superfeedr API endpoints. As a result, all responses are available (as object attributes on the `superscription` object itself) in their entirety, should you choose to introspect and/or work with them further.

- Free software: BSD license
- Documentation: http://superscription.rtfd.org.

1.1 Features

1. The various `hub.modes` offered by Superfeedr are made available as straightforward & intuitive methods.
2. An (almost) one-to-one mapping of allowed parameters to keyword-arguments for these methods.
3. Each response is available in its entirety for inspection if needed, thanks to Kenneth Reitz's excellent `requests` module.

1.2 Basic Usage

Basic usage is pretty straightforward and looks like this:

```
>>> from superscription import Superscription
>>> ss = Superscription("Marvin", "PainInMyDiodes")
```

More examples and details available in the documentation!

Please feel free to add to/modify the documentation and the project by forking and submitting a PR.

CHAPTER 2

Installation

At the command line:

```
$ easy_install superscription
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv superscription
$ pip install superscription
```

CHAPTER 3

Usage

3.1 Basic Usage

Basic usage is pretty straightforward and looks like this:

```
>>> from superscription import Superscription
>>> ss = Superscription("Marvin", "PainInMyDiodes")
```

You can also, pass in a token, instead of a password. In fact, it is recommended!

```
>>> ss = Superscription("Marvin", token="0123456789abcdef0123456789abcdef")
```

Don't worry, not passing either the password or the token will raise an `AttributeError`!

```
>>> ss = Superscription("ArthurDent")
Traceback (most recent call last):
[...]
AttributeError: You must initialize the object with either a password or a token! We recommend the to
```

3.2 Modes? Methods!

Each Superfeedr `hub.mode` (`subscribe`, `unsubscribe`, `list`, `display`) is available as an object method of the Superscription class. All parameters accepted by Superfeedr can be declared as keyword arguments.

> NOTE: The order of the parameters is the same as specified in the Superfeedr PuSH Webhooks documentation, in case you want to pass them as `args` rather than `kwargs`. Using keyword arguments is always recommended, though. :)

3.3 Paramters? Kwargs!

Any optional parameters allowed by the Superfeedr PuSH API can simply be passed as keyword arguments for each of the corresponding method calls. To construct the keyword argument corresponding to a parameter, replace the dot (.) in the parameter name, with an underscore (_).

For instance, `hub.secret` corresponds to `hub_secret`:

```
>>> ss = Superscription("demo", token="demo")
>>> result = ss.subscribe(hub_topic='http://push-pub.appspot.com/feed', hub_callback="http://my.callb
```

```
>>> print result
True
```

Parameters without a '.' in their name correspond to keyword arguments with the same name. for instance, `page` can be passed as `page` itself:

```
>>> result = ss.list(hub_callback="http://my.callback.tld/callback/", page="1")
>>> print result
True
>>> ss.response.status_code
202
```

...except for the parameter `format` which corrresponds to the keyword argument `fmt`. This has been done deliberately, to avoid any potential confusion/problems with the `str.format()` method/call.

3.4 Available methods

All methods return a boolean `True` or `False` depending on whether the subscription was successfully performed using Superfeedr or ran into problems.

Potential problems have been documented (to some extent) in the section Errors & Warnings.

3.4.1 Subscribe

The `.subscribe()` method takes TWO mandatory arguments, `hub_topic` & `hub_callback`:

```
>>> result = ss.subscribe('http://push-pub.appspot.com/feed', "http://my.domain.tld/callback/")
>>> print result
True
>>> ss.response.status_code
202
```

3.4.2 Unsubscribe

The `.unsubscribe()` method takes ONE mandatory argument, `hub_topic`:

```
>>> result = ss.unsubscribe(hub_topic='http://push-pub.appspot.com/feed')
>>> print result
True
>>> ss.response.status_code
202
```

If you have multiple subscriptions for the same `hub_topic` but each one is associated with a different `hub_callback`, you will have to call the method each time for each (of the) callback(s) for which you want to unsubscribe.

3.4.3 List

The `.list()` method takes ONE mandatory argument, `hub_callback`:

```
>>> result = ss.list(hub_callback='http://my.domain.tld/callback')
>>> print result
True
```

```
>>> ss.response.status_code
200
>>> ss.response.json()
[{u'subscription': {u'feed': {u'url': u'http://push-pub.appspot.com/feed', u'title': u'Publisher exa
```

3.4.4 Retrieve

The `.retrieve()` method takes ONE mandatory argument, `hub_topic`:

```
>>> result = ss.retrieve(hub_topic='http://push-pub.appspot,com/feed')
>>> print result
True
>>> ss.response.status_code
200
>>> data = ss.response.json()
>>> data.keys()
[u'status', u'items', u'title']
```

3.5 Responses

The response from Superfeedr is the standard `Response` object returned by the 'requests module and is saved as an attribute on the `superscription` object itself:

```
>>> result = ss.subscribe('http://push-pub.appspot.com/feed', "http://my.domain.tld/callback/")
>>> print ss.response
<Response [204]>
```

Do note that the 'response' attribute is available on `ss` & not `result`.

```
>>> type(ss.response)
requests.models.Response
>>> ss.response.status_code
204
>>> ss.response.content
''
>>> ss.response.text
u''
```

3.6 Errors and Warnings

If the Superfeedr request runs into problems, or is generally unsuccessful for some reason, the execution is stopped by raising the corresponding error.

Arguments with `None` values are discarded and a `RuntimeWarning` is displayed before the API endpoint request is made:

```
>>> result = ss.subscribe("http://push-pub.appspot.com/feed", "http://my.domain.tld/callback", hub_se
[...]
RuntimeWarning: Extra arguments passed in function call: hub_secret, hub_verify
```

Not passing in the mandatory arguments raises the standard `TypeError`:

```
>>> result = ss.subscribe('http://push-pub.appspot.com/feed')
[...]
TypeError: subscribe() takes at least 3 arguments (2 given)
```

URLs for `hub_topic` and `hub_callback` must be fully-qualified URIs, with a minimum of `scheme` and `hostname`, else you'll get an `AttributeError` on the first of them to be caught:

```
>>> ss.subscribe("http://google", "my.domain.tld/callback")
[...]
AttributeError: http://google - URL is not fully-qualified!
```

CHAPTER 4

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:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/shrikant-joshi/superscription/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.

4.1.2 Fix Bugs

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

4.1.3 Implement Features

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

4.1.4 Write Documentation

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

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/shrikant-joshi/superscription/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 :)

4.2 Get Started!

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

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

   ```
   $ git clone git@github.com:your_name_here/superscription.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 superscription
   $ cd superscription/
   $ python setup.py develop
   ```

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 superscription tests
   $ python setup.py test
   $ 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.

4.3 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, and 3.3, and for PyPy. Check https://travis-ci.org/shrikant-joshi/superscription/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

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

CHAPTER 5

Credits

5.1 Development Lead

- Shrikant Joshi <shrikant.j@gmail.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.1.0 (2014-03-22)

- First release on PyPI.

CHAPTER 7

Indices and tables

- *genindex*
- *modindex*
- *search*