BloomSky API Documentation

Release 0.3.2+0.g05bd45f.dirty

Dave Forgac

Dec 28, 2017

Contents

Contents:

CHAPTER 1

BloomSky API

A simple Python client for the BloomSky API.

Note: Neither this package nor its maintainer are affiliated with BloomSky.

For more information about the BloomSky device and its API, see: http://weatherlution.com/bloomsky-api/

1.1 Prerequisites

- Python (2.7, 3.3, 3.4, 3.5)
- BloomSky API key (get it here: https://dashboard.bloomsky.com/)

1.2 Getting Started

1.2.1 Installation

```
pip install BloomSky-API
```

To install optional command-line interface (requires click):

```
pip install BloomSky-API[cli]
```

1.2.2 Usage

You can either store the API key in an environment variable named *BLOOMSKY_API_KEY* or provide it as an argument when creating the client.

Stored in environment variable:

```
import bloomsky_api
client = bloomsky_api.BloomSkyAPIClient()
data = client.get_data()
```

Provided via argument:

```
import bloomsky_api
client = bloomsky_api.BloomSkyAPIClient(api_key='Your-real-API-key-goes-here')
data = client.get_data()
```

1.2.3 Command Line Interface

If you install the optional command-line interface, the *bloomsky-api* command will be available. Usage:

```
Usage: bloomsky-api [OPTIONS]

  Retrieve data from the BloomSky API and output it as JSON.

Options:
  --api-key TEXT          BloomSky API key (can be set via env var
                          BLOOMSKY_API_KEY).
  --api-url TEXT          Override BloomSky API endpoint URL.
  --json-indent INTEGER   Number of spaces to indent nested JSON levels.
  -i, --intl-units        Use SI units instead of the default US.
  --raw                   Return raw response instead of remapped keys.
  --help                  Show this message and exit.
```

1.2.4 Data

The returned data contains all of the information from the API response but with more Pythonic names and data types.

CHAPTER 2

Installation

Install using pip:

```
pip install BloomSky-API
```

If you'd like to use the optional command-line client, install the *cli* "extras" using this command:

```
pip install BloomSky-API[cli]
```

CHAPTER 3

Usage

You can either store the API key in an environment variable named *BLOOMSKY_API_KEY* or provide it as an argument when creating the client.

Stored in environment variable:

```
import bloomsky_api
client = bloomsky_api.BloomSkyAPIClient()
data = client.get_data()
```

Provided via argument:

```
import bloomsky_api
client = bloomsky_api.BloomSkyAPIClient(api_key='Your-real-API-key-goes-here')
data = client.get_data()
```

CHAPTER 4

Contributing

Contributions of all types are welcome!

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

If you think something's not working correctly, report it here: https://github.com/tylerdave/bloomsky-api/issues.

If you are reporting a bug, please include:

- Detailed steps to reproduce the bug.
- Any details about your local setup that might be helpful in troubleshooting.

4.1.2 Fix Bugs

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

4.1.3 Implement Features

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

4.1.4 Write Documentation

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

4.1.5 New Features & Feedback

The best way to send feedback is to file an issue at https://github.com/tylerdave/bloomsky-api/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 *BloomSky-API* for local development.

1. Fork the *bloomsky-api* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/bloomsky-api.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 bloomsky-api
$ cd bloomsky-api/
$ pip install -e .[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 bloomsky_api 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 2.7, 3.3, 3.4, 3.5 and for PyPy. Check https://travis-ci.org/tylerdave/bloomsky-api/pull_requests and make sure that the tests pass for all supported Python versions.

CHAPTER 5

Credits

5.1 Maintainer

- Dave Forgac <tylerdave@tylerdave.com>

5.2 Contributors

None yet. Why not be the first? See: CONTRIBUTING.rst

CHAPTER 6

History

- 0.3.0 - 2017-01-07
  - Include device_type in response
- 0.2.0 - 2016-08-22
  - Add CLI
- 0.1.0 - 2016-08-21
  - Convert timestamp to ISO timestamp accounting for offset
- 0.0.1 - 2016-08-20
  - Initial release!

CHAPTER 7

Indices and tables

- genindex
- modindex
- search