Verifone Documentation

Release 0.1.24

Jaana Sarajärvi

Jan 07, 2022

Contents:

CHAPTER 1

Verifone

Python package for Verifone

- Free software: MIT license
- Documentation: https://verifone.readthedocs.io.

1.1 Features

- Payment with Verifone
- Get available payment methods
- Refund payments

1.2 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install Verifone, run this command in your terminal:

```
$ pip install verifone
```

This is the preferred method to install Verifone, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for Verifone can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/vilkasgroup/Verifone
```

Or download the tarball:

```
$ curl  -OL https://github.com/vilkasgroup/Verifone/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

To use Verifone in a project:

```
import verifone
```

3.1 Using Verifone class

```
from verifone import verifone

# Create object
verifone_client = Verifone('Agreement_code', 'Private_key', 'Verifone_public_key',
↪'software_name, 'version')

# Test agreement code and keys. Check value in i-f-1-1_availability element (if 0␣
↪then no access to server).
response = verifone_client.is_available()

# If method needs a parameters, define them in dictionary
params = {
    's-f-1-30_buyer-first-name': 'Test',
    's-f-1-30_buyer-last-name': 'Tester',
    's-f-1-100_buyer-email-address': 'tester@tester.email,
    's-t-1-30_buyer-phone-number': '123456789',
    's-t-1-255_buyer-external-id': '123456,
}

# Call method with parameters
response = verifone_client.list_saved_payment_methods(params)
```

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/vilkasgroup/verifone/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" and "help wanted" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

4.1.4 Write Documentation

Verifone could always use more documentation, whether as part of the official Verifone 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/vilkasgroup/verifone/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 *verifone* for local development.

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

```
$ git clone git@github.com:your_name_here/verifone.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 verifone
$ cd verifone/
$ 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 verifone tests
$ python setup.py test or 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 and 3.6, and for PyPy. Check https://travis-ci.org/vilkasgroup/Verifone/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_verifone
```

4.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER 5

Credits

5.1 Development Lead

- Jaana Sarajärvi <jaana.sarajarvi@vilkas.fi>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.1.24 (2020-06-29)

- Upadated version of pycryptodome.
- Updated version of requests.
- Updated Pipfile.lock.

6.2 0.1.23 (2020-02-21)

- Updated version of requests.

6.3 0.1.22 (2019-11-07)

- Fields ‘s-t-1-40_shop-receipt__phase’ and ‘s-t-1-40_submit’ are now excluded from the signature verification process.

6.4 0.1.21 (2019-09-11)

- Added ‘l-t-1-20_saved-payment-method-id’ to ‘generate_payment_data’ function.

6.5 0.1.20 (2019-05-27)

- Added checking that product name can not be longer than 30 character.

6.6 0.1.19 (2019-05-21)

- Updated version of requests.

6.7 0.1.18 (2019-04-26)

- Check is endpoint available. If not, use other endpoint.

6.8 0.1.17 (2019-04-24)

- Modified logging.

6.9 0.1.16 (2019-04-23)

- Updated Pipfile.lock.

6.10 0.1.15 (2019-04-15)

- Updated Jinja2 version in Pipfile.

6.11 0.1.14 (2019-04-05)

- Updated Pipfile.lock and requirements.

6.12 0.1.13 (2019-04-01)

- Updated Pipfile.lock and dev requirements.

6.13 0.1.12 (2019-03-26)

- Updated Pipfile.lock and requirements.

6.14 0.1.11 (2019-02-28)

- Modified Verifone class. New parameter “return_error_dict” was added.
- Updated test cases.

6.15 0.1.10 (2019-02-28)

- Modified method "send_request". It does not throw anymore error if 's-f-1-30_error-message' is returned.
- Updated Pipfile.lock.

6.16 0.1.9 (2019-02-20)

- Updated Pipfile.lock.

6.17 0.1.8 (2019-01-07)

- Updated Pipfiles and requirement files. There was security issue in PyYAML module.

6.18 0.1.7 (2018-12-17)

- Length of 'dynamic_feedback' parameter in method 'generate_payment_data'.

6.19 0.1.6 (2018-12-17)

- Added 'dynamic_feedback' to method 'generate_payment_data'.

6.20 0.1.5 (2018-12-14)

- Changes in pycountry module which needed some changes also to Verifone's country and currency methods.

6.21 0.1.4 (2018-11-12)

- Fixed length of extra data fields. External id can be 255 characters and note 36 characters.

6.22 0.1.3 (2018-10-26)

- Remove s-t-1-40_shop-order__phase from signature verification.

6.23 0.1.2 (2018-09-12)

- Changes in generate_payment_data feature.

6.24 0.1.1 (2018-09-06)

- Added post urls for hosted pages.

6.25 0.1.0 (2018-08-31)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search