ePages provisioning Documentation

Release 1.0.2

Tatu Wikman

Jan 04, 2021

Contents

Contents:

CHAPTER 1

ePages provisioning

Python library for calling ePages provisioning services

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

1.1 Features

- ePages SimpleProvisioningService for easy shop creation, modifying and deletion

1.2 Credits

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

[~ Dependencies scanned by PyUp.io ~]

CHAPTER 2

Installation

2.1 Stable release

To install ePages provisioning, run this command in your terminal:

```
$ pip install epages_provisioning
```

This is the preferred method to install ePages provisioning, 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 ePages provisioning can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/tswfi/epages_provisioning
```

Or download the tarball:

```
$ curl  -OL https://github.com/tswfi/epages_provisioning/tarball/master
```

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

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

CHAPTER 3

Usage

There are three different ways to use this module.

- **SimpleProvisioningService**
  - Simple soap service for handling the shop data
  - does not support Attributes but can handle most of the required stuff
- **ShopConfigService**
  - bit more complete service that can add extra attributes to the shop
- **Shop**
  - pythonic way of calling ShopConfigService

3.1 Shop

Shop uses the ShopConfigService as a "transport" layer.

3.1.1 Create new shop

```
from epages_provisioning.provisioning import ShopConfigService
from epages_provisioning.shop import Shop
sc = ShopConfigService(
    server = "example.com",
    provider = "Distributor",
    username = "admin",
    password = "admin",
)

# this wont create the shop yet
shop = Shop(Alias="MyTestShop", provisioning=sc)
```

(continues on next page)

(continued from previous page)

```
# shop type is mandatory
shop.ShopType="MinDemo"
# create the shop
shop.create()
# access the shop attributes, and change them as required
shop.IsTrialShop = False
# apply the changes to the server
shop.apply()
# get one attribute from shop (particulary useful if you can also extend
# the ePages end, for example add attribute SSO_URL to shop ;)
shop.get_shop_attribute('CreationDate')
# or set a attribute
shop.set_shop_attribute('GrantServiceAccessUntil', '2100-01-01')
```

3.1.2 Mark the shop for deletion

Mark the shop for delete, ePages will periodically check for shops that have been marked for deletion for long enough and will delete them

```
# this will call apply automatically
# WARNING: this will also do a refresh from the server, you should call
# apply before to prevent losing information in our shop object
shop.mark_for_delete()
# and reverse delete
shop.mark_for_delete(mark=False)
```

3.1.3 Reset merchants password

Only the super merchant password can be resetted.

```
# this will call apply automatically
# WARNING: this will also do a refresh from the server, you should call
# apply before to prevent losing information in our shop object
shop.reset_merchant_pass(newpass="hunter2")
```

3.1.4 Rename shop

This will change the shops alias and thus all the url structures, not really recommended for a live shop.

```
# this will call apply automatically
# WARNING: this will also do a refresh from the server, you should call
# apply before to prevent losing information in our shop object
shop.rename("MyOtherTestShop")
```

3.1.5 Delete shop

Totally remove the shop

```
shop.delete(shopref=True)
```

3.2 SimpleProvisioningService

3.2.1 Create new shop

```
from epages_provisioning import provisioning
sp = provisioning.SimpleProvisioningService(
    server = "example.com",
    provider = "Distributor",
    username = "admin",
    password = "admin",
)
shop = sp.get_createshop_obj(
    {
        'Alias': 'TestShop1',
        'ShopType': 'MinDemo',
    }
)
sp.create(shop)
```

3.2.2 Get shop info

```
from epages_provisioning import provisioning
sp = provisioning.SimpleProvisioningService(
    server = "example.com",
    provider = "Distributor",
    username = "admin",
    password = "admin",
)
shop = sp.get_shopref_obj(
    {
        'Alias': 'TestShop1',
    }
)
shopinfo = sp.get_info(shop)
```

3.3 ShopConfigService

3.3.1 Create new shop

```
from epages_provisioning import provisioning
sc = provisioning.ShopConfigService(
    server = "example.com",
    provider = "Distributor",
    username = "admin",
    password = "admin",
)
shop = sc.get_createshop_obj(
    {
        'Alias': 'TestShop1',
        'ShopType': 'MinDemo',
    }
```

(continues on next page)

(continued from previous page)

```
)
sc.create(shop)
```

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/tswfi/epages_provisioning/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

ePages provisioning could always use more documentation, whether as part of the official ePages provisioning 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/tswfi/epages_provisioning/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 *epages_provisioning* for local development.

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

```
$ git clone git@github.com:your_name_here/epages_provisioning.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 epages_provisioning
$ cd epages_provisioning/
$ 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 epages_provisioning 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. and the tests should pass ;)
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.

4.4 Tips

To run a subset of tests:

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

4.5 Releasing new version

When you are ready to release a new version follow these steps:

1. Merge all changes that should be included in the new release to master. And checkout master.
2. Update HISTORY.rst with the new version number and changes. And commit your changes to master.
3. run:

   ```
   $ bumpversion patch|minor|major
   ```

4. push to master with tags to trigger deploy:

   ```
   $ git push --tags
   $ git push
   ```

This will build the tag and when it is successfull will also deploy to pypi and testpypi

CHAPTER 5

Credits

5.1 Development Lead

- Tatu Wikman <tatu.wikman@gmail.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 1.0.2 (2021-01-04)

- remove last remrants of travis from docs
- fix readme in built package
- retest deploy process

6.2 1.0.1 (2021-01-04)

- drop travis and switch to github actions for pypi publish
- fix pypi and docs build

6.3 1.0.0 (2021-01-04)

- drop support for python 2.7
- update zeep to 4.0.0

6.4 0.5.0 (2018-10-02)

- added two new methods get_shop_attribute and set_shop_attribute to shop class
- fixed some documentation typos
- fixed some code comments
- updated dev requirements via pyup

6.5 0.4.0 (2018-09-10)

- update zeep to 3.1.0
- added coveralls to testsuite
- ShopAddress attributes to shop class
- update to ShopConfigService12, getAllInfo is now fixed

6.6 0.3.0 (2017-12-22)

- added “shop” class which is a pythonic wrapper over the shopconfigservice

6.7 0.2.1 (2017-12-19)

- fixed AddtionalAttributes and SecondaryDomains
- restructuring

6.8 0.2.0 (2017-12-08)

- First “working” release

6.9 0.1.2 through 0.1.7 (2017-12-08)

- Travis deployment tests

6.10 0.1.0 (2017-12-05)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search