jamf Pro API Documentation

Release 0.1.2

Ryan Meyers

Feb 06, 2018

Contents

Contents:

CHAPTER 1

jamf Pro API

Provide API calls for jamf Pro (formerly JSS)

- Free software: MIT license
- Documentation: https://jamf-pro-api.readthedocs.io.

1.1 Features

- TODO

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 jamf Pro API, run this command in your terminal:

```
$ pip install jssapi
```

This is the preferred method to install jamf Pro API, 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 jamf Pro API can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/sreyemnayr/jamf_pro_api
```

Or download the tarball:

```
$ curl  -OL https://github.com/sreyemnayr/jamf_pro_api/tarball/master
```

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

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

CHAPTER 3

Usage

To use jamf Pro API in a project:

```
from jssapi import JSSApi

"""Initialize the api with your credentials"""
api = JSSApi(url=JSS_URL, user=JSS_USER, pwd=JSS_PASS, dbhost=JSS_DB_HOST,
db=JSS_DB_DB, dbuser=JSS_DB_USER, dbpasswd=JSS_DB_PASS)

"""Get all mobile devices and print their names"""
devices = api.get(method='mobiledevices')

for device in devices:
    print(device['name'])

"""Get device with ID #1 and print its name"""

device = api.get(method='mobiledevices/id/1')
print(device['location']['real_name'])


"""Set Wallpaper for device with ID #1"""
import base64

with open('temp.png',"rb") as image_file:
    encoded_string = base64.b64encode(image_file.read()).decode()
body = str("<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"?><mobile_
↪device_command><command>Wallpaper</command><wallpaper_setting>1</wallpaper_setting>
↪<wallpaper_content>") + \
       str(encoded_string) + \
       str("</wallpaper_content><mobile_devices><mobile_device><id>%s</id></mobile_
↪device></mobile_devices></mobile_device_command>") % str(iid)
api.post(method='mobiledevicecommands/command/Wallpaper',body=body)
```

CHAPTER 4

jssapi

4.1 jssapi package

4.1.1 Submodules

4.1.2 jssapi.cli module

Console script for jssapi.

4.1.3 jssapi.cph module

class jssapi.cph.**ConfigProfileHelper**(*name*, *description*, *organization*)
Bases: object

add_payload(*payload_type*, *content*)

add_restrictions_payload(*restrictions*)

generate_profile(*escape=False*)

payloads = []

profile_description = ''

profile_name = ''

profile_organization = ''

profile_uuid = '7eb5576e-ffd6-4ff7-b81f-e0fded0003ad'

4.1.4 jssapi.decorators module

class jssapi.decorators.**alias**(**aliases*)
Bases: object

Alias class that can be used as a decorator for making methods callable through other names (or "aliases"). Note: This decorator must be used inside an @aliased -decorated class. For example, if you want to make the method shout() be also callable as yell() and scream(), you can use alias like this:

@alias('yell', 'scream') def shout(message):

....

jssapi.decorators.**aliased**(*aliased_class*)

4.1.5 jssapi.jssapi module

Main JSSApi module.

class jssapi.jssapi.**JSSApi**(*url=''*, *head={'Accept': 'application/json'}*, *user=''*, *pwd=''*, *dbhost=''*, *db=''*, *dbuser=''*, *dbpasswd=''*)

Bases: object

change(*method='mobiledevices'*, *body=''*)
Interact with PUT methods of JSS API

create(*method='mobiledevices'*, *body=''*)
Interact with POST methods of JSS API

delete(*method='mobiledevices'*)
Interact with DELETE methods of JSS API

edit(*method='mobiledevices'*, *body=''*)
Interact with PUT methods of JSS API

find(*method='mobiledevices'*)
Interact with GET methods of JSS API

get(*method='mobiledevices'*)
Interact with GET methods of JSS API

insert(*method='mobiledevices'*, *body=''*)
Interact with POST methods of JSS API

modify(*method='mobiledevices'*, *body=''*)
Interact with PUT methods of JSS API

new(*method='mobiledevices'*, *body=''*)
Interact with POST methods of JSS API

post(*method='mobiledevices'*, *body=''*)
Interact with POST methods of JSS API

put(*method='mobiledevices'*, *body=''*)
Interact with PUT methods of JSS API

remove(*method='mobiledevices'*)
Interact with DELETE methods of JSS API

retrieve(*method='mobiledevices'*)
Interact with GET methods of JSS API

set_auth(*user=''*, *pwd=''*)
Provide login credentials

set_db(*host=None*, *db=''*, *user=''*, *passwd=''*)
Provide DB credentials if you want to directly interact with MySQL DB

update(*method='mobiledevices'*, *body=''*)
Interact with PUT methods of JSS API

4.1.6 Module contents

CHAPTER 5

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/sreyemnayr/jamf_pro_api/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” and “help wanted” is open to whoever wants to implement it.

5.1.3 Implement Features

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

5.1.4 Write Documentation

jamf Pro API could always use more documentation, whether as part of the official jamf Pro API 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/sreyemnayr/jamf_pro_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 :)

5.2 Get Started!

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

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

```
$ git clone git@github.com:your_name_here/jamf_pro_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 jamf_pro_api
$ cd jamf_pro_api/
$ 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 jamf_pro_api 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.

5.3 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/sreyemnayr/jamf_pro_api/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ py.test tests.test_jamf_pro_api
```

CHAPTER 6

Credits

6.1 Development Lead

- Ryan Meyers <ryanmeyersweb@gmail.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

History

7.1 0.1.0 (2018-02-02)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

j

Index