
pyyaledoorman

Espen Fjellvær Olsen

Jun 28, 2021

CONTENTS

1	Features	3
2	Requirements	5
3	Installation	7
4	Usage	9
5	Contributing	11
6	License	13
7	Issues	15
8	Credits	17
	Python Module Index	25
	Index	27

A Python library for interfacing with the Yale Doorman Home API.

FEATURES

- Read whether lock is unlocked, locked, or door open
- Locking / Unlocking
- Enabling / Disabling auto lock
- Reading Volume level, Language, Auto lock disabled/enabled

REQUIREMENTS

- Yale Doorman home user account.

INSTALLATION

You can install *Pyyaledoorman* via [pip](#) from [PyPI](#):

```
$ pip install pyyaledoorman
```


USAGE

Please see [Usage](#) for details.

```
import pyyaledoorman
import aiohttp
import asyncio
import pyyaledoorman

async def main():
    async with aiohttp.ClientSession() as session:
        client = pyyaledoorman.Client(
            "username",
            "password",
            session=session,
        )
        assert await client.login() == True
        await client.update_devices()

        for device in client.devices:
            print(device.name)
            await device.disable_autolock()
            await device.unlock(pincodes="123456")

loop = asyncio.get_event_loop()
loop.run_until_complete(main())
```


CONTRIBUTING

Contributions are very welcome. To learn more, see the [Contributor Guide](#).

LICENSE

Distributed under the terms of the [MIT license](#), *Pyyledoorman* is free and open source software.

ISSUES

If you encounter any problems, please [file an issue](#) along with a detailed description.

This project was generated from [@cjolowicz's Hypermodern Python Cookiecutter](#) template.

8.1 Reference

- `pyyaledoorman.Client`
- `pyyaledoorman.Device`

8.1.1 pyyaledoorman.Client

Yale API Client. Used to log in to the API and instantiate Devices.

exception `pyyaledoorman.client.AuthenticationError(*args)`
Exception for authentication errors.

Parameters `args` (*Any*) –

Return type `None`

class `pyyaledoorman.client.Client` (`username`, `password`, `initial_token='VnVWWDZyVjIXSUNzVHJhcUVpdVNCUHBwZ3ZPakxUeXNsRU1LUH'`, `session=None`)

Yale Doorman client.

Parameters

- **username** – Username for logging in to the Yale API.
- **password** – Password for logging in to the Yale API.
- **initial_token** – Initial token for logging in to the Yale API.
- **session** (*optional*) – aiohttp ClientSession to use.

property `devices`

Return list of Devices.

async `login()`

Log in to the Yale API.

Returns `True` if successfully logged in.

Return type `bool`

Raises *AuthenticationError* – Supplied username, password or initial token is wrong.

property login_ts

Return the login timestamp.

property refresh_token

Return the refresh token.

property session

Return the aiohttp session.

property token

Return the access token.

property token_expires_in

Return seconds from login whence the access token expires.

async update_devices()

Update the device states.

Return type None

8.1.2 pyyaledoorman.Device

A Yale Doorman Device instance.

class pyyaledoorman.device.**Device** (*client, device_config*)

Used to instantiate a Yale Doorman device.

property address

Return the device address, most times synonymous with *device_id*.

property area

Return the device area.

property autolock_enabled

Return whether autolock is enabled.

Returns *True* if autolock is enabled, otherwise *False*

Return type bool

property device_id

Return the device ID.

async disable_autolock()

Disable autolocking of the lock.

Returns *True* if successful, *False* otherwise.

Return type bool

async enable_autolock()

Enable autolocking of the lock.

Returns *True* if successful, *False* otherwise.

Return type bool

async get_deviceconfig()

Fetch the device configuration.

Returns The raw API response.

Return type Dict[str, str]

property is_locked

Return *True* if the lock is locked. Otherwise *False*.

property is_open

Return *True* if the door is open. Otherwise *False*.

property language

Return audio language.

Returns *LANG_EN, LANG_DA, LANG_NO, LANG_SE, LANG_FI, LANG_RU, LANG_TR*.

Return type The configured language of the device. Possible value are

async lock ()

Lock the lock and update the internal state to *YALE_LOCK_STATE_LOCKED*.

Returns True if locking was successful, False otherwise.

Return type bool

property name

Return the device name.

parse_config (device_config)

Parse API responses and sets *Device* configuration.

Parameters **device_config** (*Dict[str, str]*) –

Return type None

async set_deviceconfig (config_idx, value)

Set device configuration.

Parameters

- **config_idx** (*str*) – index of the configuration option to change.
- **value** (*str*) – new value to write.

Returns True if the device config was updated successfully, False otherwise.

Return type bool

property state

Return device state.

For locks the state is usually one of: * *device_status.lock* * *device_status.unlock*

property type

Return device type.

Only *device_type.door_lock* is supported at the time being

async unlock (pincode)

Unlocks the lock. Takes *pincode* as a required parameter to unlock.

Parameters **pincode** (*str*) – a valid Yale Doorman pincode

Returns True if unlock successful, False otherwise.

Return type bool

async update_state ()

Update the *Device* status from the API.

Return type None

property volume_level

Return volume level.

Returns *VOLUME_HIGH, VOLUME_LOW, VOLUME_OFF*

Return type The configured volume level. Possible values are

8.2 Usage

Example usage of the library. Please see the [Reference](#) for details.

```
import pyyaledoorman
import aiohttp
import asyncio
import pyyaledoorman

async def main():
    async with aiohttp.ClientSession() as session:
        client = pyyaledoorman.Client(
            "username",
            "password",
            session=session,
        )
        assert await client.login() == True
        await client.update_devices()

        for device in client.devices:
            print(device.name)
            await device.disable_autolock()
            await device.unlock(pincodes="123456")

loop = asyncio.get_event_loop()
loop.run_until_complete(main())
```

8.3 Contributor Guide

Thank you for your interest in improving this project. This project is open-source under the [MIT license](#) and welcomes contributions in the form of bug reports, feature requests, and pull requests.

Here is a list of important resources for contributors:

- [Source Code](#)
- [Documentation](#)
- [Issue Tracker](#)
- [`Code of Conduct`_](#)

8.3.1 How to report a bug

Report bugs on the [Issue Tracker](#).

When filing an issue, make sure to answer these questions:

- Which operating system and Python version are you using?
- Which version of this project are you using?
- What did you do?
- What did you expect to see?
- What did you see instead?

The best way to get your bug fixed is to provide a test case, and/or steps to reproduce the issue.

8.3.2 How to request a feature

Request features on the [Issue Tracker](#).

8.3.3 How to set up your development environment

You need Python 3.6+ and the following tools:

- [Poetry](#)
- [Nox](#)
- [nox-poetry](#)

Install the package with development requirements:

```
$ poetry install
```

You can now run an interactive Python session, or the command-line interface:

```
$ poetry run python
$ poetry run pyyaledoorman
```

8.3.4 How to test the project

Run the full test suite:

```
$ nox
```

List the available Nox sessions:

```
$ nox --list-sessions
```

You can also run a specific Nox session. For example, invoke the unit test suite like this:

```
$ nox --session=tests
```

Unit tests are located in the `tests` directory, and are written using the [pytest](#) testing framework.

8.3.5 How to submit changes

Open a [pull request](#) to submit changes to this project.

Your pull request needs to meet the following guidelines for acceptance:

- The Nox test suite must pass without errors and warnings.
- Include unit tests. This project maintains 100% code coverage.
- If your changes add functionality, update the documentation accordingly.

Feel free to submit early, though—we can always iterate on this.

To run linting and code formatting checks before committing your change, you can install pre-commit as a Git hook by running the following command:

```
$ nox --session=pre-commit -- install
```

It is recommended to open an issue before starting work on anything. This will allow a chance to talk it over with the owners and validate your approach.

8.4 Contributor Covenant Code of Conduct

8.4.1 Our Pledge

We as members, contributors, and leaders pledge to make participation in our community a harassment-free experience for everyone, regardless of age, body size, visible or invisible disability, ethnicity, sex characteristics, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, religion, or sexual identity and orientation.

We pledge to act and interact in ways that contribute to an open, welcoming, diverse, inclusive, and healthy community.

8.4.2 Our Standards

Examples of behavior that contributes to a positive environment for our community include:

- Demonstrating empathy and kindness toward other people
- Being respectful of differing opinions, viewpoints, and experiences
- Giving and gracefully accepting constructive feedback
- Accepting responsibility and apologizing to those affected by our mistakes, and learning from the experience
- Focusing on what is best not just for us as individuals, but for the overall community

Examples of unacceptable behavior include:

- The use of sexualized language or imagery, and sexual attention or advances of any kind
- Trolling, insulting or derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or email address, without their explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

8.4.3 Enforcement Responsibilities

Community leaders are responsible for clarifying and enforcing our standards of acceptable behavior and will take appropriate and fair corrective action in response to any behavior that they deem inappropriate, threatening, offensive, or harmful.

Community leaders have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, and will communicate reasons for moderation decisions when appropriate.

8.4.4 Scope

This Code of Conduct applies within all community spaces, and also applies when an individual is officially representing the community in public spaces. Examples of representing our community include using an official e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event.

8.4.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported to the community leaders responsible for enforcement at espen@mrfjo.org. All complaints will be reviewed and investigated promptly and fairly.

All community leaders are obligated to respect the privacy and security of the reporter of any incident.

8.4.6 Enforcement Guidelines

Community leaders will follow these Community Impact Guidelines in determining the consequences for any action they deem in violation of this Code of Conduct:

1. Correction

Community Impact: Use of inappropriate language or other behavior deemed unprofessional or unwelcome in the community.

Consequence: A private, written warning from community leaders, providing clarity around the nature of the violation and an explanation of why the behavior was inappropriate. A public apology may be requested.

2. Warning

Community Impact: A violation through a single incident or series of actions.

Consequence: A warning with consequences for continued behavior. No interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, for a specified period of time. This includes avoiding interactions in community spaces as well as external channels like social media. Violating these terms may lead to a temporary or permanent ban.

3. Temporary Ban

Community Impact: A serious violation of community standards, including sustained inappropriate behavior.

Consequence: A temporary ban from any sort of interaction or public communication with the community for a specified period of time. No public or private interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, is allowed during this period. Violating these terms may lead to a permanent ban.

4. Permanent Ban

Community Impact: Demonstrating a pattern of violation of community standards, including sustained inappropriate behavior, harassment of an individual, or aggression toward or disparagement of classes of individuals.

Consequence: A permanent ban from any sort of public interaction within the community.

8.4.7 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 2.0, available at https://www.contributor-covenant.org/version/2/0/code_of_conduct.html.

Community Impact Guidelines were inspired by Mozilla's code of conduct enforcement ladder.

For answers to common questions about this code of conduct, see the FAQ at <https://www.contributor-covenant.org/faq>. Translations are available at <https://www.contributor-covenant.org/translations>.

8.5 MIT License

Copyright © 2021 Espen Fjellvær Olsen

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

The software is provided “as is”, without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and noninfringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with the software or the use or other dealings in the software.

PYTHON MODULE INDEX

p

`pyyaledoorman.client`, [17](#)

`pyyaledoorman.device`, [18](#)

INDEX

A

`address()` (*pyyaledoorman.device.Device* property), 18
`area()` (*pyyaledoorman.device.Device* property), 18
`AuthenticationError`, 17
`autolock_enabled()` (*pyyaledoorman.device.Device* property), 18

C

`Client` (class in *pyyaledoorman.client*), 17

D

`Device` (class in *pyyaledoorman.device*), 18
`device_id()` (*pyyaledoorman.device.Device* property), 18
`devices()` (*pyyaledoorman.client.Client* property), 17
`disable_autolock()` (*pyyaledoorman.device.Device* method), 18

E

`enable_autolock()` (*pyyaledoorman.device.Device* method), 18

G

`get_deviceconfig()` (*pyyaledoorman.device.Device* method), 18

I

`is_locked()` (*pyyaledoorman.device.Device* property), 19
`is_open()` (*pyyaledoorman.device.Device* property), 19

L

`language()` (*pyyaledoorman.device.Device* property), 19
`lock()` (*pyyaledoorman.device.Device* method), 19
`login()` (*pyyaledoorman.client.Client* method), 17
`login_ts()` (*pyyaledoorman.client.Client* property), 18

M

module
 pyyaledoorman.client, 17
 pyyaledoorman.device, 18

N

`name()` (*pyyaledoorman.device.Device* property), 19

P

`parse_config()` (*pyyaledoorman.device.Device* method), 19
pyyaledoorman.client
 module, 17
pyyaledoorman.device
 module, 18

R

`refresh_token()` (*pyyaledoorman.client.Client* property), 18

S

`session()` (*pyyaledoorman.client.Client* property), 18
`set_deviceconfig()` (*pyyaledoorman.device.Device* method), 19
`state()` (*pyyaledoorman.device.Device* property), 19

T

`token()` (*pyyaledoorman.client.Client* property), 18
`token_expires_in()` (*pyyaledoorman.client.Client* property), 18
`type()` (*pyyaledoorman.device.Device* property), 19

U

`unlock()` (*pyyaledoorman.device.Device* method), 19
`update_devices()` (*pyyaledoorman.client.Client* method), 18
`update_state()` (*pyyaledoorman.device.Device* method), 19

V

`volume_level()` (*pyyaledoorman.device.Device* property), 19