python-ring-doorbell

Release 0.8.6

Marcelo Moreira de Mello

Jan 30, 2024

CONTENTS

CHAPTER

ONE

PYTHON RING DOOR BELL

Python Ring Door Bell is a library written for Python 3.8+ that exposes the Ring.com devices as Python objects.

There is also a command line interface that is work in progress. Contributors welcome.

Currently Ring.com does not provide an official API. The results of this project are merely from reverse engineering.

Documentation: http://python-ring-doorbell.readthedocs.io/

1.1 Installation

```
# Installing from PyPi
$ pip install ring_doorbell

# Installing latest development
$ pip install \
    git+https://github.com/tchellomello/python-ring-doorbell@master
```

1.1.1 Event Listener

If you want the ring api to listen for push events from ring.com for dings and motion you will need to install with the *listen* extra:

```
$ pip install ring_doorbell[listen]
```

The api will then start listening for push events after you have first called *update_dings()* or *update_data()* but only if there is a running asyncio event loop (which there will be if using the CLI)

1.2 Using the CLI

The CLI is work in progress and currently has the following commands:

1. Show your devices:

```
$ ring-doorbell
```

   Or:

```
$ ring-doorbell show
```

2. List your device names (with device kind):

```
$ ring-doorbell list
```

3. Either count or download your vidoes or both:

```
$ ring-doorbell videos --count --download-all
```

4. Enable disable motion detection:

```
$ ring-doorbell motion-detection --device-name "DEVICENAME" --on
$ ring-doorbell motion-detection --device-name "DEVICENAME" --off
```

5. Listen for push notifications like the ones sent to your phone:

```
$ ring-doorbell listen
```

6. List your ring groups:

```
$ ring-doorbell groups
```

7. Show your ding history:

```
$ ring-doorbell history --device-name "Front Door"
```

8. Show your currently active dings:

```
$ ring-doorbell dings
```

9. Query a ring api url directly:

```
$ ring-doorbell raw-query --url /clients_api/dings/active
```

10. Run `ring-doorbell --help` or `ring-doorbell videos --help` for full options

1.3 Initializing your Ring object

```
import json
import getpass
from pathlib import Path
from pprint import pprint

from ring_doorbell import Ring, Auth
from oauthlib.oauth2 import MissingTokenError


cache_file = Path("test_token.cache")


def token_updated(token):
    cache_file.write_text(json.dumps(token))


def otp_callback():
    auth_code = input("2FA code: ")
    return auth_code


def main():
    if cache_file.is_file():
        auth = Auth("MyProject/1.0", json.loads(cache_file.read_text()), token_updated)
    else:
        username = input("Username: ")
        password = getpass.getpass("Password: ")
        auth = Auth("MyProject/1.0", None, token_updated)
        try:
            auth.fetch_token(username, password)
        except MissingTokenError:
            auth.fetch_token(username, password, otp_callback())

    ring = Ring(auth)
    ring.update_data()

    devices = ring.devices()
    pprint(devices)


if __name__ == "__main__":
    main()
```

1.4 Listing devices linked to your account

```
# All devices
devices = ring.devices()
{'chimes': [<RingChime: Downstairs>],
'doorbots': [<RingDoorBell: Front Door>]}

# All doorbells
doorbells = devices['doorbots']
[<RingDoorBell: Front Door>]

# All chimes
chimes = devices['chimes']
[<RingChime: Downstairs>]

# All stickup cams
stickup_cams = devices['stickup_cams']
[<RingStickUpCam: Driveway>]
```

1.5 Playing with the attributes and functions

```
devices = ring.devices()
for dev in list(devices['stickup_cams'] + devices['chimes'] + devices['doorbots']):
    dev.update_health_data()
    print('Address:    %s' % dev.address)
    print('Family:     %s' % dev.family)
    print('ID:         %s' % dev.id)
    print('Name:       %s' % dev.name)
    print('Timezone:   %s' % dev.timezone)
    print('Wifi Name:  %s' % dev.wifi_name)
    print('Wifi RSSI:  %s' % dev.wifi_signal_strength)

    # setting dev volume
    print('Volume:     %s' % dev.volume)
    dev.volume = 5
    print('Volume:     %s' % dev.volume)

    # play dev test shound
    if dev.family == 'chimes':
        dev.test_sound(kind = 'ding')
        dev.test_sound(kind = 'motion')

    # turn on lights on floodlight cam
    if dev.family == 'stickup_cams' and dev.lights:
        dev.lights = 'on'
```

1.6 Showing door bell events

```
devices = ring.devices()
for doorbell in devices['doorbots']:

    # listing the last 15 events of any kind
    for event in doorbell.history(limit=15):
        print('ID:       %s' % event['id'])
        print('Kind:     %s' % event['kind'])
        print('Answered: %s' % event['answered'])
        print('When:     %s' % event['created_at'])
        print('--' * 50)

    # get a event list only the triggered by motion
    events = doorbell.history(kind='motion')
```

1.7 Downloading the last video triggered by a ding or motion event

```
devices = ring.devices()
doorbell = devices['doorbots'][0]
doorbell.recording_download(
    doorbell.history(limit=100, kind='ding')[0]['id'],
                     filename='last_ding.mp4',
                     override=True)
```

1.8 Displaying the last video capture URL

```
print(doorbell.recording_url(doorbell.last_recording_id))
'https://ring-transcoded-videos.s3.amazonaws.com/99999999.mp4?X-Amz-Expires=3600&X-Amz-
↪Date=20170313T232537Z&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=TOKEN_SECRET/
↪us-east-1/s3/aws4_request&X-Amz-SignedHeaders=host&X-Amz-Signature=secret'
```

1.9 Controlling a Light Group

```
groups = ring.groups()
group = groups['the-group-you-want']

print(group.lights)
# Prints True if lights are on, False if off

# Turn on lights indefinitely
group.lights = True

# Turn off lights
group.lights = False
```

(continues on next page)

(continued from previous page)

```
# Turn on lights for 30 seconds
group.lights = (True, 30)
```

1.10 How to contribute

See our Contributing Page.

1.11 Credits && Thanks

- This project was inspired and based on https://github.com/jeroenmoors/php-ring-api. Many thanks @jeroenmoors.
- A guy named MadBagger at Prism19 for his initial research (http://www.prism19.com/doorbot/second-pass-and-comm-reversing/)
- The creators of mitmproxy (https://mitmproxy.org/) great http and https traffic inspector
- @mfussenegger for his post on mitmproxy and virtualbox https://zignar.net/2015/12/31/sniffing-vbox-traffic-mitmproxy/
- To the project http://www.android-x86.org/ which allowed me to install Android on KVM.
- Many thanks to Carles Pina I Estany <carles@pina.cat> for creating the python-ring-doorbell Debian Package (https://tracker.debian.org/pkg/python-ring-doorbell).

1.11.1 Contributing

Contributions are welcome and very appreciated!! Keep in mind that every little contribution helps, don't matter what.

Types of Contributions

Report Bugs

Report bugs at https://github.com/tchellomello/python-ring-doorbell/issues

If you are reporting a bug, please include:

- Ring product and firmware version
- Steps to reproduce the issue
- Anything you judge interesting for the troubleshooting

Fix Bugs

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

Implement Features

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

Documentation

Documentation is always good. So please feel free to add any documentation you think will help our users.

Request Features

File an issue at https://github.com/tchellomello/python-ring-doorbell/issues.

Get Started!

Ready to contribute? Here's how to set up *python-ring-doorbell* for local development.

1. Fork the *python-ring-doorbell* repo on GitHub.
2. Clone your fork locally:

```
$ cd YOURDIRECTORYFORTHECODE
$ git clone git@github.com:YOUR_GITHUB_USERNAME/python-ring-doorbell.git
```

3. We are using poetry for dependency management.

   If you dont have poetry installed you can install it with:

```
$ curl -sSL https://install.python-poetry.org | python3 -
```

   This installs Poetry in a virtual environment to isolate it from the rest of your system. Then to install *python-ring-doorbell*:

```
$ poetry install
```

   Poetry will create a virtual environment for you and install all the requirements

   If you want to be able to build the docs (not necessary unless you are working on the doc generation):

```
$ poetry install --extras docs
```

4. Create a branch for local development:

```
$ git checkout -b NAME-OF-YOUR-BUGFIX-OR-FEATURE
```

   Now you can make your changes locally.

5. We are using tox for testing and linting:

```
$ poetry run tox -r
```

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.

Thank you!!

Additional Notes

Poetry

Dependencies

Poetry is very useful at managing virtual environments and ensuring that dependencies all match up for you. It manages this with the use of the *poetry.lock* file which contains all the exact versions to be installed. This means that if you add any dependecies you should do it via:

```
$ poetry add pypi_project_name
```

rather than pip. This will update *pyproject.toml* and *poetry.lock* accordingly. If you install something in the virtual environment directly via pip you will need to run:

```
$ poetry lock --no-update
```

to resync the lock file but without updating all the other requirements to latest versions. To uninstall a dependency:

```
$ poetry remove pypi_project_name
```

finally if you want to add a dependency for development only:

```
$ poetry add --group dev pypi_project_name
```

Environments

Poetry creates a virtual environment for the project and you can activate the virtual environment with:

```
$ poetry shell
```

To exit the shell type `exit` rather than deactivate. However you don't **need** to activate the virtual environment and you can run any command without activating it by:

```
$ poetry run SOME_COMMAND
```

It is possible to manage all this from within a virtual environment you create yourself but that requires installing poetry into the same virtual environment and this can potentially cause poetry to uninstall some of its own dependencies in certain situations. Hence the recommendation to install poetry into a seperate virtual environment of its via the install script above or pipx.

See poetry documentation for more info

Documentation

To build the docs install with the docs extra:

```
$ poetry install --extras docs
```

Then generate a Github access token (no permissions are needed) and export it as follows:

```
$ export CHANGELOG_GITHUB_TOKEN="«your-40-digit-github-token»"
```

Then build:

```
$ make -C html
```

You can add the token to your shell profile to avoid having to export it each time. (e.g., .env, ~/.bash_profile, ~/.bashrc, ~/.zshrc, etc)

1.11.2 Changelog

0.8.6

Released on 2024-01-25 - GitHub - PyPI

0.8.5

Released on 2023-12-21 - GitHub - PyPI

0.8.4

Released on 2023-12-12 - GitHub - PyPI

0.8.3

Released on 2023-11-27 - GitHub - PyPI

0.8.2

Released on 2023-11-24 - GitHub - PyPI

0.8.1

Released on 2023-11-15 - GitHub - PyPI

0.8.0

Released on 2023-11-08 - GitHub - PyPI

0.7.7

Released on 2023-10-31 - GitHub - PyPI

0.7.6

Released on 2023-10-25 - GitHub - PyPI

0.7.5

Released on 2023-10-25 - GitHub - PyPI

0.7.4

Released on 2023-09-27 - GitHub - PyPI

0.7.3

Released on 2023-09-11 - GitHub - PyPI

0.7.2

Released on 2021-12-18 - GitHub - PyPI

0.7.1

Released on 2021-08-26 - GitHub - PyPI

0.7.0

Released on 2021-02-05 - GitHub - PyPI

0.6.2

Released on 2020-11-21 - GitHub - PyPI

0.6.1

Released on 2020-09-28 - GitHub - PyPI

0.6.0

Released on 2020-01-14 - GitHub - PyPI

0.5.0

Released on 2020-01-12 - GitHub - PyPI

0.4.0

Released on 2020-01-11 - GitHub - PyPI

Version 0.2.9

Released on 2020-01-03 - GitHub - PyPI

Version 0.2.8

Released on 2019-12-27 - GitHub - PyPI

Version 0.2.6

Released on 2019-12-27 - GitHub - PyPI

Version 0.2.5

Released on 2019-12-20 - GitHub - PyPI

Version 0.2.3

Released on 2019-03-05 - GitHub - PyPI

Version 0.2.2

Released on 2018-10-29 - GitHub - PyPI

Version 0.2.1

Released on 2018-06-15 - GitHub - PyPI

Version 0.2.0

Released on 2018-05-16 - GitHub - PyPI

Version 0.1.9

Released on 2017-11-29 - GitHub - PyPI

Version 0.1.8

Released on 2017-11-22 - GitHub - PyPI

Version 0.1.7

Released on 2017-11-14 - GitHub - PyPI

Version 0.1.6

Released on 2017-10-19 - GitHub - PyPI

Version 0.1.5

Released on 2017-10-17 - GitHub - PyPI

v0.1.4

Released on 2017-04-30 - GitHub - PyPI

v0.1.3

Released on 2017-03-31 - GitHub - PyPI

v0.1.2

Released on 2017-03-20 - GitHub - PyPI

0.1.1

Released on 2017-03-09 - GitHub - PyPI

0.1.0

Released on 2017-02-25 - GitHub - PyPI

0.0.4

Released on 2017-02-15 - GitHub - PyPI

0.0.3

Released on 2017-02-15 - GitHub - PyPI

0.0.2

Released on 2017-02-15 - GitHub - PyPI

0.0.1

Released on 2017-02-15 - GitHub - PyPI