toonapilib Documentation

Release 4.1.0

Costas Tyfoxylos

Feb 24, 2020

Contents

Contents:

CHAPTER 1

toonapilib

A library to interact with eneco's "Toon" smart meter via their official api https://api.toon.eu

- Documentation: https://toonapilib.readthedocs.org/en/latest

1.1 Development Workflow

The workflow supports the following steps

- lint
- test
- build
- document
- upload
- graph

These actions are supported out of the box by the corresponding scripts under _CI/scripts directory with sane defaults based on best practices. Sourcing setup_aliases.ps1 for windows powershell or setup_aliases.sh in bash on Mac or Linux will provide with handy aliases for the shell of all those commands prepended with an underscore.

The bootstrap script creates a .venv directory inside the project directory hosting the virtual environment. It uses pipenv for that. It is called by all other scripts before they do anything. So one could simple start by calling _lint and that would set up everything before it tried to actually lint the project

Once the code is ready to be delivered the _tag script should be called accepting one of three arguments, patch, minor, major following the semantic versioning scheme. So for the initial delivery one would call

```
$ _tag –minor
```

which would bump the version of the project to 0.1.0 tag it in git and do a push and also ask for the change and automagically update HISTORY.rst with the version and the change provided.

So the full workflow after git is initialized is:

- repeat as necessary (of course it could be test - code - lint :)) * code * lint * test
- commit and push
- develop more through the code-lint-test cycle
- tag (with the appropriate argument)
- build
- upload (if you want to host your package in pypi)
- document (of course this could be run at any point)

1.2 Important Information

This template is based on pipenv. In order to be compatible with requirements.txt so the actual created package can be used by any part of the existing python ecosystem some hacks were needed. So when building a package out of this **do not** simple call

```
$ python setup.py sdist bdist_egg
```

as this will produce an unusable artifact with files missing. Instead use the provided build and upload scripts that create all the necessary files in the artifact.

1.3 Project Features

- Reads values for gas, electric, temperature.
- Identifies connected hue lights and fibaro smartplugs
- Can read and set temperature and thermostat state
- Can turn lights or plugs on, off or toggle their state
- Can get consumption values from fibaro plugs
- More . . .

CHAPTER 2

Installation

At the command line:

```
$ pip install toonapilib
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv toonapilib
$ pip install toonapilib
```

Or, if you are using pipenv:

```
$ pipenv install toonapilib
```

CHAPTER 3

Usage

To develop on toonapilib:

```
# The following commands require pipenv as a dependency

# To lint the project
_CI/scripts/lint.py

# To execute the testing
_CI/scripts/test.py

# To create a graph of the package and dependency tree
_CI/scripts/graph.py

# To build a package of the project under the directory "dist/"
_CI/scripts/build.py

# To see the package version
_CI/scipts/tag.py

# To bump semantic versioning [--major|--minor|--patch]
_CI/scipts/tag.py --major|--minor|--patch

# To upload the project to a pypi repo if user and password are properly provided
_CI/scripts/upload.py

# To build the documentation of the project
_CI/scripts/document.py
```

To use toonapilib in a project:

```
from toonapilib import Toon

token = '1234-abcdefg-9876654'

toon = Toon(token)
```

Print information about the agreement. Attributes are self explanatory.

```
print(toon.agreement.id)
print(toon.agreement.checksum)
print(toon.agreement.display_common_name)
print(toon.agreement.display_hardware_version)
print(toon.agreement.display_software_version)
print(toon.agreement.heating_type)
print(toon.agreement.solar)
print(toon.agreement.toonly)
```

Print information about the gas. Values are cached internally for 30 seconds so as to not overwhelm the api. After the 30 seconds any access to any of the attributes will refresh the information through a new call to the api.

```
print(toon.gas.average_daily)
print(toon.gas.average)
print(toon.gas.daily_cost)
print(toon.gas.daily_usage)
print(toon.gas.is_smart)
print(toon.gas.meter_reading)
print(toon.gas.value)
```

Print information about the electricity. Values are cached internally for 30 seconds so as to not overwhelm the api. After the 30 seconds any access to any of the attributes will refresh the information through a new call to the api.

```
print(toon.power.average_daily)
print(toon.power.average)
print(toon.power.daily_cost)
print(toon.power.daily_usage)
print(toon.power.is_smart)
print(toon.power.meter_reading)
print(toon.power.meter_reading_low)
print(toon.power.daily_usage_low)
print(toon.power.value)
```

Print information about the solar power production. Values are cached internally for 30 seconds so as to not overwhelm the api. After the 30 seconds any access to any of the attributes will refresh the information through a new call to the api.

```
print(toon.solar.maximum)
print(toon.solar.produced)
print(toon.solar.average_produced)
print(toon.solar.meter_reading_low_produced)
print(toon.solar.meter_reading_produced)
print(toon.solar.daily_cost_produced)
print(toon.solar.value)
```

Print information about connected hue lights.

```
# loop over all the lights
for light in toon.lights:
    print(light.is_connected)
    print(light.device_uuid)
    print(light.rgb_color)
    print(light.name)
    print(light.current_state)
    print(light.device_type)
    print(light.in_switch_all_group)
```

(continues on next page)

(continued from previous page)

```
    print(light.in_switch_schedule)
    print(light.is_locked)
    print(light.zwave_index)
    print(light.zwave_uuid)

# or get a light by assigned name
light = toon.get_light_by_name('Kitchen Ceiling')

# print current status
print(light.status)

# checking whether the light can be toggled. For that to be able to
# happen the light needs to be connected and not locked.
# this state is checked internally from all the methods trying to toggle
# the switch state of the light
print(light.can_toggle)

# lights can be turned on, off or toggled
light.turn_on()
light.turn_off()
light.toggle()
```

Print information about connected fibaro smart plugs.

```
# get first smartplug
plug = toon.smartplugs[0]

# or get smartplug by assigned name
plug = toon.get_smartplug_by_name('Dryer')

# print all the information about the plug
print(plug.current_usage)
print(plug.current_state)
print(plug.average_usage)
print(plug.daily_usage)
print(plug.device_uuid)
print(plug.is_connected)
print(plug.name)
print(plug.network_health_state)
print(plug.device_type)
print(plug.in_switch_all_group)
print(plug.in_switch_schedule)
print(plug.is_locked)
print(plug.usage_capable)
print(plug.zwave_index)
print(plug.zwave_uuid)
print(plug.flow_graph_uuid)
print(plug.quantity_graph_uuid)


# print current status
print(plug.status)

# checking whether the plug can be toggled. For that to be able to
# happen the plug needs to be connected and not locked.
# this state is checked internally from all the methods trying to toggle
# the switch state of the plug
```

(continues on next page)

(continued from previous page)

```
print(plug.can_toggle)

# plugs can be turned on, off or toggled
plug.turn_on()
plug.turn_off()
plug.toggle()
```

Print information about connected smokedetectors.

```
# loop over all the smokedetectors
for smokedetector in toon.smokedetectors:
    print(smokedetector.device_uuid)
    print(smokedetector.name)
    print(smokedetector.last_connected_change)
    print(smokedetector.is_connected)
    print(smokedetector.battery_level)
    print(smokedetector.device_type)


# or get a smokedetector by assigned name
smokedetector = toon.get_smokedetector_by_name('Kitchen')
```

Get the current temperature

```
# show the current temperature
print(toon.temperature)
```

Work with thermostat states

```
# show the information about the current state
print(toon.thermostat_state.name)
print(toon.thermostat_state.id)
print(toon.thermostat_state.temperature)
print(toon.thermostat_state.dhw)

# set the current state by using a name out of ['comfort', 'home', 'sleep', away]
toon.thermostat_state = 'comfort' # Case does not matter. The actual
                                  # values can be overwritten on the
                                  # configuration.py dictionary.
```

Check out all the thermostat states configured

```
for state in toon.thermostat_states:
    print(state.name)
    print(state.id)
    print(state.temperature)
    print(state.dhw)
```

Work with the thermostat

```
# show current value of thermostat
print(toon.thermostat)

# manually assign temperature to thermostat. This will override the thermostat state
toon.thermostat = 20
```

Exposing flow rrd metrics for for a requested time period

```
# Print default time period flow for power and gas
# from and to arguments can be anything that dateparser can understand. https://
↪dateparser.readthedocs.io/en/latest/
print(toon.data.flow.get_power_time_window('2 months ago', '3 days ago'))
print(toon.data.flow.get_gas_time_window('22 nov 2018', '1 jan 2019'))
```

Exposing graph rrd metrics for a requested time period

```
# Print default time period graph for power, gas and district_heat
# from and to arguments can be anything that dateparser can understand. https://
↪dateparser.readthedocs.io/en/latest/
print(toon.data.graph.get_power_time_window('2 months ago', '3 days ago', 'weeks'))
print(toon.data.graph.get_gas_time_window('22 nov 2018', '1 jan 2019', 'days'))
print(toon.data.graph.get_district_heat_time_window('2 years ago', 'today', 'months'))
```

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

4.1 Submit Feedback

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.

4.1.1 Get Started!

Ready to contribute? Here's how to set up *toonapilib* for local development. Using of pipenv is highly recommended.

1. Clone your fork locally:

```
$ git clone https://github.com/costastf/toonapilib
```

2. Install your local copy into a virtualenv. Assuming you have pipenv installed, this is how you set up your clone for local development:

```
$ cd toonapilib/
$ pipenv install --ignore-pipfile
```

3. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally. Do your development while using the CI capabilities and making sure the code passes lint, test, build and document stages.

4. Commit your changes and push your branch to the server:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

5. Submit a merge request

CHAPTER 5

toonapilib

5.1 toonapilib package

5.1.1 Submodules

5.1.2 toonapilib.configuration module

A place to store the configuration.

5.1.3 toonapilib.helpers module

All helper objects will live here.

class toonapilib.helpers.**Agreement**(*id*, *checksum*, *heating_type*, *display_common_name*, *display_hardware_version*, *display_software_version*, *solar*, *toonly*)

Bases: tuple

checksum
Alias for field number 1

display_common_name
Alias for field number 3

display_hardware_version
Alias for field number 4

display_software_version
Alias for field number 5

heating_type
Alias for field number 2

id
Alias for field number 0

solar
Alias for field number 6

toonly
Alias for field number 7

class toonapilib.helpers.**Data**(*toon_instance*)
Data object exposing flow and graph attributes.

class Flow(*toon_instance*)
Bases: *toonapilib.helpers.TimeWindowRetriever*

The object that exposes the flow information of categories in toon.

The information is rrd metrics

get_gas_time_window(*from_datetime*, *to_datetime*)
Retrieves the gas flow for the provided time window.
Parameters
- **from_datetime** (*str*) – A string representing a date that dateparser can understand
- **to_datetime** (*str*) – A string representing a date that dateparser can understand

Returns rrd response if returned

get_power_time_window(*from_datetime*, *to_datetime*)
Retrieves the power flow for the provided time window.
Parameters
- **from_datetime** (*str*) – A string representing a date that dateparser can understand
- **to_datetime** (*str*) – A string representing a date that dateparser can understand

Returns rrd response if returned

class Graph(*toon_instance*)
Bases: *toonapilib.helpers.TimeWindowRetriever*

The object that exposes the graph information of categories in toon.

The information is rrd metrics and the object dynamically handles the accessing of attributes matching with the corresponding api endpoint if they are know, raises an exception if not.

get_district_heat_time_window(*from_datetime*, *to_datetime*, *interval='hours'*)
Retrieves the district heat graph for the provided time window.
Parameters
- **from_datetime** (*str*) – A string representing a date that dateparser can understand
- **to_datetime** (*str*) – A string representing a date that dateparser can understand
- **interval** (*str*) – A string representing the interval, one of ['hours', 'days', 'weeks', 'months', 'years']

Returns rrd response if returned

get_gas_time_window(*from_datetime*, *to_datetime*, *interval='hours'*)
Retrieves the gas graph for the provided time window.
Parameters
- **from_datetime** (*str*) – A string representing a date that dateparser can understand
- **to_datetime** (*str*) – A string representing a date that dateparser can understand
- **interval** (*str*) – A string representing the interval, one of ['hours', 'days', 'weeks', 'months', 'years']

Returns rrd response if returned

get_power_time_window(*from_datetime*, *to_datetime*, *interval='hours'*)
Retrieves the power graph for the provided time window.
Parameters
- **from_datetime** (*str*) – A string representing a date that dateparser can understand

- **to_datetime** (*str*) – A string representing a date that dateparser can understand
- **interval** (*str*) – A string representing the interval, one of ['hours', 'days', 'weeks', 'months', 'years']

Returns rrd response if returned

class toonapilib.helpers.**Light**(*toon_instance*, *name*)
Bases: *toonapilib.helpers.Switch*

Object modeling the hue light bulbs that toon can interact with.

It inherits from switch which is the common interface with the hue lamps to turn on, off or toggle

rgb_color
The rgb color value of the light.

class toonapilib.helpers.**Low**(*meter_reading_low*, *daily_usage_low*)
Bases: tuple

daily_usage_low
Alias for field number 1

meter_reading_low
Alias for field number 0

class toonapilib.helpers.**PowerUsage**(*average_daily*, *average*, *daily_cost*, *daily_usage*, *is_smart*, *meter_reading*, *value*, *meter_reading_low*, *daily_usage_low*)
Bases: tuple

average
Alias for field number 1

average_daily
Alias for field number 0

daily_cost
Alias for field number 2

daily_usage
Alias for field number 3

daily_usage_low
Alias for field number 8

is_smart
Alias for field number 4

meter_reading
Alias for field number 5

meter_reading_low
Alias for field number 7

value
Alias for field number 6

class toonapilib.helpers.**SmartPlug**(*toon_instance*, *name*)
Bases: *toonapilib.helpers.Switch*

Object modeling the fibaro smart plugs the toon can interact with.

It inherits from switch which is the common interface with the hue lamps to turn on, off or toggle

average_usage
The average power usage.

current_usage
The current power usage.

daily_usage
The daily power usage.

flow_graph_uuid
The uuid of the flow graph.

network_health_state
The state of the network health.

quantity_graph_uuid
The uuid of the quantity graph.

usage_capable
Boolean about the capability of the device to report power usage.

class toonapilib.helpers.**SmokeDetector**(*device_uuid*, *name*, *last_connected_change*, *is_connected*, *battery_level*, *device_type*)

Bases: tuple

battery_level
Alias for field number 4

device_type
Alias for field number 5

device_uuid
Alias for field number 0

is_connected
Alias for field number 3

last_connected_change
Alias for field number 2

name
Alias for field number 1

class toonapilib.helpers.**Solar**(*maximum*, *produced*, *value*, *average_produced*, *meter_reading_low_produced*, *meter_reading_produced*, *daily_cost_produced*)

Bases: tuple

average_produced
Alias for field number 3

daily_cost_produced
Alias for field number 6

maximum
Alias for field number 0

meter_reading_low_produced
Alias for field number 4

meter_reading_produced
Alias for field number 5

produced
Alias for field number 1

value
Alias for field number 2

class toonapilib.helpers.**Switch**(*toon_instance*, *name*)
Core object to implement the turning on, off or toggle.

Both hue lamps and fibaro plugs have a switch component that is shared. This implements that usage.

can_toggle
Boolean about the capability of the device to toggle state.

current_state
The device's current state.

device_type
The type of the device.

device_uuid
The uuid of the device.

in_switch_all_group
Boolean about whether the device is in a switch group.

in_switch_schedule
Boolean about whether the device is in a switch schedule.

is_connected
Boolean about the connection status of the device.

is_locked
Boolean about the lock state of the object.

name
The name of the device.

status
Returns the status of the device in a human friendly way.

toggle()
Toggles the status of the device.

turn_off()
Turns the device off.

turn_on()
Turns the device on.

zwave_index
The zwave index of the device.

zwave_uuid
The zwave uuid.

class toonapilib.helpers.**ThermostatInfo**(*active_state*, *boiler_connected*, *burner_info*, *current_displayed_temperature*, *current_modulation_level*, *current_set_point*, *error_found*, *have_ot_boiler*, *next_program*, *next_set_point*, *next_state*, *next_time*, *ot_communication_error*, *program_state*, *real_set_point*)
Bases: tuple

active_state
Alias for field number 0

boiler_connected
Alias for field number 1

burner_info
Alias for field number 2

current_displayed_temperature
Alias for field number 3

current_modulation_level
Alias for field number 4

current_set_point
Alias for field number 5

error_found
Alias for field number 6

have_ot_boiler
Alias for field number 7

next_program
Alias for field number 8

next_set_point
Alias for field number 9

next_state
Alias for field number 10

next_time
Alias for field number 11

ot_communication_error
Alias for field number 12

program_state
Alias for field number 13

real_set_point
Alias for field number 14

class toonapilib.helpers.**ThermostatState**(*name*, *id*, *temperature*, *dhw*)
Bases: tuple

dhw
Alias for field number 3

id
Alias for field number 1

name
Alias for field number 0

temperature
Alias for field number 2

class toonapilib.helpers.**TimeWindowRetriever**(*toon_instance*)
Object able to retrieve windows of time from endpoints.

class toonapilib.helpers.**Usage**(*average_daily*, *average*, *daily_cost*, *daily_usage*, *is_smart*, *meter_reading*, *value*)
Bases: tuple

average
Alias for field number 1

average_daily
Alias for field number 0

daily_cost
Alias for field number 2

daily_usage
Alias for field number 3

is_smart
Alias for field number 4

meter_reading
Alias for field number 5

value
Alias for field number 6

class toonapilib.helpers.**User**(*client_id*, *client_secret*, *username*, *password*)
Bases: tuple

client_id
Alias for field number 0

client_secret
Alias for field number 1

password
Alias for field number 3

username
Alias for field number 2

5.1.4 toonapilib.toonapilib module

Main code for toonapilib.

class toonapilib.toonapilib.**Toon**(*authentication_token*, *tenant_id='eneco'*, *display_common_name=None*)
Model of the toon smart meter from eneco.

burner_on
Boolean value of the state of the burner.

burner_state
The state the burner is in.

display_names
The ids of all the agreements.

Returns A list of the agreement ids.

Return type list

gas
A gas object modeled as a named tuple.

Type return

get_light_by_name(*name*)
Retrieves a light object by its name.

Parameters **name** – The name of the light to return

Returns A light object

get_smartplug_by_name(*name*)
Retrieves a smartplug object by its name.

Parameters **name** – The name of the smartplug to return

Returns A smartplug object

get_smokedetector_by_name(*name*)
Retrieves a smokedetector object by its name.

Parameters **name** – The name of the smokedetector to return

Returns A smokedetector object

get_thermostat_state_by_id(*id_*)
Retrieves a thermostat state object by its id.

Parameters **id** – The id of the thermostat state

Returns The thermostat state object

get_thermostat_state_by_name(*name*)
Retrieves a thermostat state object by its assigned name.

Parameters **name** – The name of the thermostat state

Returns The thermostat state object

lights
A list of light objects.

Type return

power
A power object modeled as a named tuple.

Type return

program_state
The active program state of the thermostat.

Returns the program state

smartplugs
A list of smartplug objects.

Type return

smokedetectors
A list of smokedetector objects modeled as named tuples.

Type return

solar
A solar object modeled as a named tuple.

Type return

status
The status of toon, cached for 300 seconds.

temperature
The current actual temperature as perceived by toon.

Returns A float of the current temperature

thermostat
The current setting of the thermostat as temperature.

Returns A float of the current setting of the temperature of the

thermostat

thermostat_info
A thermostatinfo object modeled as a named tuple.

Type return

thermostat_state
The state of the thermostat programming.

Returns A thermostat state object of the current setting

thermostat_states
The thermostat states of toon, cached for 1 hour.

5.1.5 toonapilib.toonapilibexceptions module

Custom exception code for toonapilib.

exception toonapilib.toonapilibexceptions.**AgreementsRetrievalError**
Bases: exceptions.Exception

Could not retrieve agreements.

exception toonapilib.toonapilibexceptions.**IncompleteStatus**
Bases: exceptions.Exception

The status received is missing vital information and is unusable.

exception toonapilib.toonapilibexceptions.**InvalidAuthenticationToken**
Bases: exceptions.Exception

The authentication token provided was not accepted as valid.

exception toonapilib.toonapilibexceptions.**InvalidDisplayName**
Bases: exceptions.Exception

The display name provided was not accepted as valid.

exception toonapilib.toonapilibexceptions.**InvalidProgramState**
Bases: exceptions.Exception

The state provided to the program is not a valid one.

exception toonapilib.toonapilibexceptions.**InvalidThermostatState**
Bases: exceptions.Exception

The state provided to the thermostat is not a valid one.

5.1.6 Module contents

toonapilib package.

Import all parts from toonapilib here

CHAPTER 6

Credits

6.1 Development Lead

- Costas Tyfoxylos <costas.tyf@gmail.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

History

CHAPTER 8

0.0.1 (09-12-2017)

- First code creation

CHAPTER 9

3.0.2 (16-02-2019)

- Ported to the latest template. Fixed an issue with the monkey patched requests get method assuming no other process running. Refactored some code to 3.7 specific.

CHAPTER 10

3.0.3 (16-02-2019)

- Small template cleanup

CHAPTER 11

3.0.4 (16-02-2019)

- fixed float representation for temperature

CHAPTER 12

3.0.5 (23-02-2019)

- Tyring to fix library playing well with synology under Home Assistant

CHAPTER 13

3.0.6 (23-02-2019)

- Fixed dumb bug

CHAPTER 14

3.0.7 (23-02-2019)

- re implemented named tuples for python 3.5 and fixed newly introduced bug with token expiry optimization.

CHAPTER 15

3.0.8 (24-02-2019)

- reverted dataclasses to namedtules for 3.5 compatibility

CHAPTER 16

3.0.9 (24-02-2019)

- removed unneeded dependency

CHAPTER 17

3.0.10 (26-02-2019)

- Disregards program if set to bypass race condition is setting the temperature while program is active

CHAPTER 18

3.0.11 (04-03-2019)

- Fixed bug with thermostat state being unsettable with the contribution of John Van De Vrugt https://github.com/JohnvandeVrugt.

CHAPTER 19

3.1.0 (04-03-2019)

- Implemented data object under toon that exposes flow and graph rrd data for power and gas.

CHAPTER 20

3.2.0 (05-03-2019)

- Added capabilities to enable/disable thermostat program with the contribution of John Van De Vrugt https://github.com/JohnvandeVrugt.

CHAPTER 21

3.2.1 (06-03-2019)

- Fixed missplaced files in the root of the virtual environment

CHAPTER 22

3.2.2 (18-03-2019)

- Changed caching from 30 seconds to 300 seconds due to rate limiting

CHAPTER 23

3.2.3 (11-04-2019)

- Updating headers according to the upcoming change from Quby

CHAPTER 24

3.2.4 (10-06-2019)

- Accepted fix from Reinder Reinders (“reinder83”) for thermostat states new api endpoint that sometimes is missing from the status response.

CHAPTER 25

3.2.5 (20-10-2019)

- Removed monkey patching of requests and implemented explicit handling of re authentication.
- Updated tempalate and bumped dependencies.
- Linted.

CHAPTER 26

4.0.0 (30-11-2019)

- Implemented new token authentication and removed all references to the old authentication method which will not be supported after 01/12/19. Added backoff for some methods.

CHAPTER 27

4.1.0 (30-11-2019)

- Exposed display names attribute since it is used in Home Assistant internally.

CHAPTER 28

Indices and tables

- genindex
- modindex
- search

Python Module Index

t

Index

Q

R

S

T

U

V

Z