
pyecodevices-rt2 Documentation

Release 1.3.4

Pierre COURBIN

Oct 10, 2022

CONTENTS:

1	pycodelices-rt2 - Python GCE Ecodevices RT2	1
1.1	Features	1
1.2	Credits	3
2	Installation	5
2.1	Stable release	5
2.2	From sources	5
3	Usage	7
3.1	EcoDevicesRT2	7
3.2	Using cached values	7
3.3	Advanced/API usage	8
3.4	Counter	9
3.5	DigitalInput	9
3.6	EnOcean Switch or Sensor	10
3.7	Post and Sub-Post	10
3.8	Relay	11
3.9	SupplierIndex	11
3.10	Toroid	12
3.11	VirtualOutput	12
3.12	X4FP (Heaters)	12
3.13	XTHL	13
4	pycodelices_rt2	15
4.1	pycodelices_rt2 package	15
5	Contributing	23
5.1	Types of Contributions	23
5.2	Get Started!	24
5.3	Pull Request Guidelines	25
5.4	Tips	25
5.5	Deploying	25
6	Credits	27
6.1	Development Lead	27
6.2	Contributors	27
7	History	29
7.1	1.3.1 (2022-08-07)	29
7.2	1.3.0 (2022-08-07)	29
7.3	1.2.1 (2021-05-15)	29

7.4	1.2.0 (2021-05-14)	29
7.5	1.1.0 (2021-04-17)	29
7.6	1.0.1 (2021-04-12)	30
7.7	1.0.0 (2021-04-08)	30
8	Indices and tables	31
	Python Module Index	33
	Index	35

PYECODEVICES-RT2 - PYTHON GCE ECODEVICES RT2

Get information from [GCE Ecodevices RT2](#).

This work is originally developed for use with [Home Assistant](#) and the *custom component* `ecodevices_rt2`.

- Free software: MIT license
- Documentation: <https://pyecodevices-rt2.readthedocs.io>.

1.1 Features

- Connect to the API (see [GCE Ecodevices RT2 API](#) (or [PDF](#))) and get any value:

```
# Example with indexes
from pyecodevices_rt2 import EcoDevicesRT2
ecodevices = EcoDevicesRT2('192.168.0.20', '80', "mysuperapikey")
ecodevices.get('Index', 'All') # Get all indexes as JSON
ecodevices.get('Index', 'All', 'Index_TI1') # Get specific value
```

- Define a simple object such as *Counter*, *DigitalInput*, *EnOcean Switch or Sensor*, *Post and Sub-Post*, *Relay*, *SupplierIndex*, *Toroid*, *VirtualOutput*, *X4FP (Heaters)*, *XTHL*:

```
# Example with a Relay
from pyecodevices_rt2 import EcoDevicesRT2, Relay
ecodevices = EcoDevicesRT2('192.168.0.20', '80', "mysuperapikey")
# Relay number 1
test = Relay(ecodevices, 1)
print("Current status: %r" % test.status)
```

(continues on next page)

(continued from previous page)

```
test.off() # Change relay to off
test.on() # Change relay to on
test.toggle() # Invert relay status
test.status = True # Change relay to on
```

- Play with cached variables. You can defined a maximum value (in milliseconds) during which you consider an API value do not need to be updated:

```
from pycodelices_rt2 import EcoDevicesRT2

# Create the ecodelices object with a default "cached" value of 1s
ecodelices = EcoDevicesRT2('192.168.0.20', '80', "mysuperapikey", cached_ms=1000)

print("# All Indexes")
print(ecodelices.get('Index', 'All')) # Call the API
print(ecodelices.get('Index', 'All')) # Do not call the API since the last value was
↳retrieved less than 1s (1000ms) ago
print(ecodelices.get('Index', 'All', cached_ms=0)) # Force to call the API even if the
↳last value was retrieved less than 1s (1000ms) ago

# For each property in other objects, you can call "get_PROPERTY(cached_ms=XX)"
# Example with Counter 1:
test = Counter(ecodelices, 1)
print("Current value: %d" % test.value) # Call the API
print("Current price: %d" % test.price) # Call the API
print("Current value: %d" % test.value) # Do not call the API since the last value was
↳retrieved less than 1s (1000ms) ago
print("Current price: %d" % test.price) # Do not call the API since the last value was
↳retrieved less than 1s (1000ms) ago
print("Current value: %d" % test.get_value()) # Do not call the API since the last value
↳was retrieved less than 1s (1000ms) ago
print("Current price: %d" % test.get_price()) # Do not call the API since the last value
↳was retrieved less than 1s (1000ms) ago
print("Current value: %d" % test.get_value(cached_ms=0)) # Force to call the API even if
↳the last value was retrieved less than 1s (1000ms) ago
print("Current price: %d" % test.get_price(cached_ms=0)) # Force to call the API even if
↳the last value was retrieved less than 1s (1000ms) ago
print("Current value: %d" % test.get_value(cached_ms=2000)) # Do not call the API if the
↳last value was retrieved less than 2s (2000ms) ago
print("Current price: %d" % test.get_price(cached_ms=2000)) # Do not call the API if the
↳last value was retrieved less than 2s (2000ms) ago
```

1.2 Credits

This work is inspired by the work of [Aohzan](#).

This package was created with [Cookiecutter](#) and the [audreyr/cookiecutter-pypackage](#) project template.

INSTALLATION

2.1 Stable release

To install pycodevices-rt2, run this command in your terminal:

```
$ pip install pycodevices-rt2
```

This is the preferred method to install pycodevices-rt2, 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 pycodevices-rt2 can be downloaded from the [Github repo](#).

You can either clone the public repository:

```
$ git clone git://github.com/pcourbin/pycodevices_rt2
```

Or download the [tarball](#):

```
$ curl -OJL https://github.com/pcourbin/pycodevices_rt2/tarball/master
```

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

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


3.1 EcoDevicesRT2

For the API and parameters, see [GCE Ecodevices RT2 API](#) (or [PDF](#)).

Parameters:

```
- `host`: ip or hostname
- `port`: (default: 80)
- `apikey`: if authentication enabled on Ecodevices RT2
- `timeout`: (default: 3)
- `cached_ms`: (default: 0), maximum delay in milliseconds during which we consider the
↪ value do not need to be updated using the API.
```

Properties:

```
- `host`: the host
- `apikey`: the apikey
- `apiurl`: the default apiurl
- `cached_ms`: the value of the maximum cached value in milliseconds
```

Methods:

```
- `ping`: return true if the Ecodevices answer
- `get`: return json or part of json.
```

3.2 Using cached values

You can defined a maximum value (in milliseconds) during which you consider an API value do not need to be updated:

```
from pyecodevices_rt2 import EcoDevicesRT2

# Create the ecodevices object with a default "cached" of 1s
ecodevices = EcoDevicesRT2('192.168.0.20', '80', "mysuperapikey", cached_ms=1000)

print("# All Indexes")
print(ecodevices.get('Index', 'All')) # Call the API
print(ecodevices.get('Index', 'All')) # Do not call the API since the last value was
↪ retrieved less than 1s (1000ms) ago
```

(continues on next page)

(continued from previous page)

```

print(ecodevices.get('Index','All',cached_ms=0)) # Force to call the API even if the
↳ last value was retrieved less than 1s (1000ms) ago

# For each property in other objects, you can call "get_PROPERTY(cached_ms=XX)"
# Example with Counter 1:
test = Counter(ecodevices, 1)
print("Current value: %d" % test.value) # Call the API
print("Current price: %d" % test.price) # Call the API
print("Current value: %d" % test.value) # Do not call the API since the last value was
↳ retrieved less than 1s (1000ms) ago
print("Current price: %d" % test.price) # Do not call the API since the last value was
↳ retrieved less than 1s (1000ms) ago
print("Current value: %d" % test.get_value()) # Do not call the API since the last value
↳ was retrieved less than 1s (1000ms) ago
print("Current price: %d" % test.get_price()) # Do not call the API since the last value
↳ was retrieved less than 1s (1000ms) ago
print("Current value: %d" % test.get_value(cached_ms=0)) # Force to call the API even if
↳ the last value was retrieved less than 1s (1000ms) ago
print("Current price: %d" % test.get_price(cached_ms=0)) # Force to call the API even if
↳ the last value was retrieved less than 1s (1000ms) ago
print("Current value: %d" % test.get_value(cached_ms=2000)) # Do not call the API if the
↳ last value was retrieved less than 2s (2000ms) ago
print("Current price: %d" % test.get_price(cached_ms=2000)) # Do not call the API if the
↳ last value was retrieved less than 2s (2000ms) ago

# Force to get all values for all requests defined in ecodevices._cached (see consts.RT2_
↳ API_GET_LINK_CACHED to default requests cached)
ecodevices.get_all_cached()
print("Current value: %d" % test.value) # Do not call the API since the last value may
↳ be retrieved less than 1s (1000ms) ago

```

3.3 Advanced/API usage

To use pycodelices-rt2 in a project, you can directly use parameters from the GCE Ecodevices RT2 API (or PDF):

```

from pycodelices_rt2 import EcoDevicesRT2

ecodevices = EcoDevicesRT2('192.168.0.20', '80', "mysuperapikey")

print("# ping")
print(ecodevices.ping())
print("# Default API call")
print(ecodevices.apiurl)

# Indexes
print("# All Indexes")
print(ecodevices.get('Index','All'))
print("# Only Index 'Index_TI1'")
print(ecodevices.get('Index','All','Index_TI1'))

```

(continues on next page)

(continued from previous page)

```

# Powers
print("# Actual power on 'POSTE5'")
print(ecodevices.get('Get', 'P', 'INSTANT_POSTE5'))

# EnOcean
print("# All EnOcean")
print(ecodevices.get('Get', 'XENO'))
print("# Set EnOcean1 and get status")
print(ecodevices.get('SetEnoPC', '1', 'status'))
print("# Clear EnOcean2 and get status")
print(ecodevices.get('ClearEnoPC', '2', 'status'))

# Heaters / FP Modules
print("# Current state of all zones")
print(ecodevices.get('Get', 'FP'))
print("# Current state of First Zone of First FP module")
print(ecodevices.get('Get', 'FP', 'FP1 Zone 1'))
print("# Force First Zone of First FP module to be on 'Confort' mode and get status")
print(ecodevices.get('SetFP01', '0', 'status'))

```

3.4 Counter

You can define a Counter (see from the [GCE Ecodevices RT2 API \(or PDF\)](#)):

```

from pyecodevices_rt2 import EcoDevicesRT2, Counter

ecodevices = EcoDevicesRT2('192.168.0.20', '80', "mysuperapikey")
print("# ping")
print(ecodevices.ping())

# Counter number 1
test = Counter(ecodevices, 1)
print("Current value: %d" % test.value)
print("Current price: %d" % test.price)

test.value = 20 # Change the value of the counter to 20
test.add(5) # Add 5 to the counter
test.substrat(10) # Subtract 10 to the counter

```

3.5 DigitalInput

You can define a DigitalInput (see from the [GCE Ecodevices RT2 API \(or PDF\)](#)):

```

from pyecodevices_rt2 import EcoDevicesRT2, DigitalInput

ecodevices = EcoDevicesRT2('192.168.0.20', '80', "mysuperapikey")
print("# ping")
print(ecodevices.ping())

```

(continues on next page)

(continued from previous page)

```
# DigitalInput number 1
test = DigitalInput(ecodevices, 1)
print("Current status: %r" % test.status)
```

3.6 EnOcean Switch or Sensor

You can define a EnOcean Switch or Sensor (see from the [GCE Ecodevices RT2 API \(or PDF\)](#)):

```
from pycodelices_rt2 import EcoDevicesRT2, EnOceanSensor, EnOceanSwitch

ecodevices = EcoDevicesRT2('192.168.0.20', '80', "mysuperapikey")
print("# ping")
print(ecodevices.ping())

# EnOceanSensor number 1
test = EnOceanSensor(ecodevices, 1)
print("Current value: %f" % test.value)

# EnOceanSwitch number 1
test = EnOceanSwitch(ecodevices, 1)
print("Current status: %r" % test.status)
test.off() # Change switch to off
test.on() # Change switch to on
test.toggle() # Invert switch status
test.status = True # Change switch to on
```

3.7 Post and Sub-Post

You can define a Post and Sub-post (see from the [GCE Ecodevices RT2 API \(or PDF\)](#)):

```
from pycodelices_rt2 import EcoDevicesRT2, Post

ecodevices = EcoDevicesRT2('192.168.0.20', '80', "mysuperapikey")
print("# ping")
print(ecodevices.ping())

# Post number 1
test = Post(ecodevices, 1)
print("Index: %f" % test.index)
print("Price: %f" % test.price)
print("Index of the day: %f" % test.index_day)
print("Price of the day: %f" % test.price_day)
print("Instant power: %f" % test.instant)

# Sub-post number 2 of Post 1
test = Post(ecodevices, 1, 2)
print("Index: %f" % test.index)
```

(continues on next page)

(continued from previous page)

```
print("Price: %f" % test.price)
print("Index of the day: %f" % test.index_day)
print("Price of the day: %f" % test.price_day)
print("Instant power: %f" % test.instant)
```

3.8 Relay

You can define a Relay (see from the [GCE Ecodevices RT2 API](#) (or [PDF](#))):

```
from pyecodevices_rt2 import EcoDevicesRT2, Relay

ecodevices = EcoDevicesRT2('192.168.0.20', '80', "mysuperapikey")
print("# ping")
print(ecodevices.ping())

# Relay number 1
test = Relay(ecodevices, 1)
print("Current status: %r" % test.status)
test.off() # Change relay to off
test.on() # Change relay to on
test.toggle() # Invert relay status
test.status = True # Change relay to on
```

3.9 SupplierIndex

You can define a SupplierIndex (see from the [GCE Ecodevices RT2 API](#) (or [PDF](#))):

```
from pyecodevices_rt2 import EcoDevicesRT2, SupplierIndex

ecodevices = EcoDevicesRT2('192.168.0.20', '80', "mysuperapikey")
print("# ping")
print(ecodevices.ping())

# SupplierIndex number 1
test = SupplierIndex(ecodevices, 1)
print("Index: %f" % test.value)
print("Price: %f" % test.price)
```

3.10 Toroid

You can define a Toroid (see from the [GCE Ecodevices RT2 API](#) (or [PDF](#))):

```
from pycodevices_rt2 import EcoDevicesRT2, Toroid

ecodevices = EcoDevicesRT2('192.168.0.20', '80', "mysuperapikey")
print("# ping")
print(ecodevices.ping())

# Toroid number 1
test = Toroid(ecodevices, 1)
print("Value: %f" % test.value)
print("Price: %f" % test.price)
```

3.11 VirtualOutput

You can define a VirtualOutput (see from the [GCE Ecodevices RT2 API](#) (or [PDF](#))):

```
from pycodevices_rt2 import EcoDevicesRT2, VirtualOutput

ecodevices = EcoDevicesRT2('192.168.0.20', '80', "mysuperapikey")
print("# ping")
print(ecodevices.ping())

# VirtualOutput number 1
test = VirtualOutput(ecodevices, 1)
print("Current status: %r" % test.status)
test.off() # Change virtualoutput to off
test.on() # Change virtualoutput to on
test.toggle() # Invert virtualoutput status
test.status = True # Change virtualoutput to on
```

3.12 X4FP (Heaters)

You can define a X4FP (see from the [GCE Ecodevices RT2 API](#) (or [PDF](#))):

```
from pycodevices_rt2 import EcoDevicesRT2, X4FP

ecodevices = EcoDevicesRT2('192.168.0.20', '80', "mysuperapikey")
print("# ping")
print(ecodevices.ping())

# X4FP of Module 1, Zone 2
test = X4FP(ecodevices, 1, 2)
print("Current mode: %d" % test.mode)
test.mode = 1 # Change mode to `Eco`
```


Table 1: List of Heater/X4FP mode values

Mode	State (EN)	Etat (FR)
-1	<i>UNKNOWN</i> (or module not present)	<i>UNKNOWN</i> (ou module non présent)
0	<i>Confort</i>	<i>Confort</i>
1	<i>Eco</i>	<i>Eco</i>
2	<i>Frost free</i>	<i>Hors Gel</i>
3	<i>Stop</i>	<i>Arret</i>
4	<i>Confort -1</i>	<i>Confort -1</i>
5	<i>Confort -2</i>	<i>Confort -2</i>

3.13 XTHL

You can define a XTHL (see from the [GCE Ecodevices RT2 API](#) (or [PDF](#))):

```
from pyecodevices_rt2 import EcoDevicesRT2, XTHL

ecodevices = EcoDevicesRT2('192.168.0.20', '80', "mysuperapikey")
print("# ping")
print(ecodevices.ping())

# XTHL number 1
test = XTHL(ecodevices, 1)
print("Temperature: %f" % test.temperature)
print("Humidity: %f" % test.humidity)
print("Luminosity: %f" % test.luminosity)
```


PYECODEVICES_RT2

4.1 pyecodevices_rt2 package

4.1.1 Submodules

4.1.2 pyecodevices_rt2.abstractsensor module

class pyecodevices_rt2.abstractsensor.**AbstractSensor**(ecort2: [EcoDevicesRT2](#), id: int, get_link: str, get_entry: str)

Bases: object

Class representing an AbstractSensor

get_value(cached_ms: Optional[int] = None) → float

Return the current AbstractSensor status.

property value: float

4.1.3 pyecodevices_rt2.abstractswitch module

class pyecodevices_rt2.abstractswitch.**AbstractSwitch**(ecort2: [EcoDevicesRT2](#), id: int, get_link: str, get_entry: str, on_link: str, off_link: str, toggle_link: str)

Bases: object

Class representing an AbstractSwitch

get_status(cached_ms: Optional[int] = None) → bool

Return the current AbstractSwitch status.

off() → bool

Turn off a AbstractSwitch.

on() → bool

Turn on a the AbstractSwitch.

property status: bool

toggle() → bool

Toggle a AbstractSwitch.

4.1.4 pycodevices_rt2.cli module

4.1.5 pycodevices_rt2.const module

4.1.6 pycodevices_rt2.counter module

class pycodevices_rt2.counter.**Counter**(ecort2: [EcoDevicesRT2](#), id: int)

Bases: object

Class representing the Counter

add(value: int) → bool

Add a value to the current Counter value.

get_price(cached_ms: Optional[int] = None) → float

Return the price of counter.

get_value(cached_ms: Optional[int] = None) → int

Return the current Counter value.

property price: float

substrat(value: int) → bool

Subtract a value to the current Counter value.

property value: int

4.1.7 pycodevices_rt2.digitalinput module

class pycodevices_rt2.digitalinput.**DigitalInput**(ecort2: [EcoDevicesRT2](#), id: int)

Bases: object

Class representing the DigitalInput

get_status(cached_ms: Optional[int] = None) → bool

Return the current DigitalInput status.

property status: bool

4.1.8 pycodevices_rt2.ecodevices_rt2 module

class pycodevices_rt2.ecodevices_rt2.**EcoDevicesRT2**(host: str, port: int = 80, apikey: str = "", timeout: int = 10, cached_ms: int = 0)

Bases: object

Class representing the Ecodevices RT2 and its API

property apikey

Return the apikey.

property apiurl

Return the default apiurl.

property cached_ms

Return the maximum cached value in milliseconds.

get(*command*, *command_value*=None, *command_entry*=None, *cached_ms*: Optional[int] = None)

Get value from api : http://{host}:{port}/api/xdevices.json?key={apikey}&{command}={command_value}, then get value {command_entry} in JSON response.

get_all_cached()

property **host**

Return the hostname.

ping() → bool

4.1.9 pycodelices_rt2.enocean module

class pycodelices_rt2.enocean.**EnOceanSensor**(*ecort2*: EcoDevicesRT2, *id*: int)

Bases: *AbstractSensor*

Class representing the EnOceanSensor

class pycodelices_rt2.enocean.**EnOceanSwitch**(*ecort2*: EcoDevicesRT2, *id*: int)

Bases: *AbstractSwitch*

Class representing the EnOceanSwitch

4.1.10 pycodelices_rt2.exceptions module

Exceptions for Ecodevices RT2.

exception pycodelices_rt2.exceptions.**EcoDevicesRT2ConnectError**

Bases: Exception

Exception to indicate an error in connection.

exception pycodelices_rt2.exceptions.**EcoDevicesRT2RequestError**

Bases: Exception

Exception to indicate an error with an API request.

4.1.11 pycodelices_rt2.post module

class pycodelices_rt2.post.**Post**(*ecort2*: EcoDevicesRT2, *id_post*: int, *id_subpost*: Optional[int] = None)

Bases: object

Class representing the Post or Sub-Post

get_index(*cached_ms*: Optional[int] = None) → float

Return the index of post/subpost.

get_index_day(*cached_ms*: Optional[int] = None) → float

Return the index of the current day of post/subpost.

get_instant(*cached_ms*: Optional[int] = None) → float

Return the instant power of post/subpost.

get_price(*cached_ms*: Optional[int] = None) → float

Return the price of post/subpost.

get_price_day(*cached_ms: Optional[int] = None*) → float

Return the price of the current day of post/subpost.

property index: float

property index_day: float

property instant: float

property price: float

property price_day: float

4.1.12 pyecodevices_rt2.relay module

class pyecodevices_rt2.relay.**Relay**(*ecort2: EcoDevicesRT2, id: int*)

Bases: [*AbstractSwitch*](#)

Class representing the Relay

4.1.13 pyecodevices_rt2.supplierindex module

class pyecodevices_rt2.supplierindex.**SupplierIndex**(*ecort2: EcoDevicesRT2, id: int*)

Bases: [*AbstractSensor*](#)

Class representing the SupplierIndex

get_price(*cached_ms: Optional[int] = None*) → float

Return the price of supplier index.

property price: float

4.1.14 pyecodevices_rt2.toroid module

class pyecodevices_rt2.toroid.**Toroid**(*ecort2: EcoDevicesRT2, id: int*)

Bases: object

Class representing the Toroid

get_price(*cached_ms: Optional[int] = None*) → float

Return the price of toroid.

get_value(*cached_ms: Optional[int] = None*) → float

Return the index of toroid.

property price: float

property value: float

4.1.15 pycodevices_rt2.virtualoutput module

class pycodevices_rt2.virtualoutput.**VirtualOutput**(ecort2: EcoDevicesRT2, id: int)
Bases: *AbstractSwitch*
Class representing the VirtualOutput

4.1.16 pycodevices_rt2.x4fp module

class pycodevices_rt2.x4fp.**X4FP**(ecort2: EcoDevicesRT2, module_id: int, zone_id: int)
Bases: object
Class representing the X4FP
get_mode(cached_ms: Optional[int] = None) → int
Return the current X4FP mode.
property mode: int

4.1.17 pycodevices_rt2.xthl module

class pycodevices_rt2.xthl.**XTHL**(ecort2: EcoDevicesRT2, id: int)
Bases: object
Class representing the XTHL
get_humidity(cached_ms: Optional[int] = None) → bool
Return the current XTHL humidity.
get_luminosity(cached_ms: Optional[int] = None) → bool
Return the current XTHL luminosity.
get_temperature(cached_ms: Optional[int] = None) → bool
Return the current XTHL temperature.
property humidity: bool
property luminosity: bool
property temperature: bool

4.1.18 Module contents

Get information from GCE Ecodevices RT2.

class pycodevices_rt2.**AbstractSensor**(ecort2: EcoDevicesRT2, id: int, get_link: str, get_entry: str)
Bases: object
Class representing an AbstractSensor
get_value(cached_ms: Optional[int] = None) → float
Return the current AbstractSensor status.
property value: float

```
class pycodevices_rt2.AbstractSwitch(ecort2: EcoDevicesRT2, id: int, get_link: str, get_entry: str,  
                                     on_link: str, off_link: str, toggle_link: str)
```

Bases: object

Class representing an AbstractSwitch

```
get_status(cached_ms: Optional[int] = None) → bool
```

Return the current AbstractSwitch status.

```
off() → bool
```

Turn off a AbstractSwitch.

```
on() → bool
```

Turn on a the AbstractSwitch.

```
property status: bool
```

```
toggle() → bool
```

Toggle a AbstractSwitch.

```
class pycodevices_rt2.Counter(ecort2: EcoDevicesRT2, id: int)
```

Bases: object

Class representing the Counter

```
add(value: int) → bool
```

Add a value to the current Counter value.

```
get_price(cached_ms: Optional[int] = None) → float
```

Return the price of counter.

```
get_value(cached_ms: Optional[int] = None) → int
```

Return the current Counter value.

```
property price: float
```

```
substrat(value: int) → bool
```

Substract a value to the current Counter value.

```
property value: int
```

```
class pycodevices_rt2.DigitalInput(ecort2: EcoDevicesRT2, id: int)
```

Bases: object

Class representing the DigitalInput

```
get_status(cached_ms: Optional[int] = None) → bool
```

Return the current DigitalInput status.

```
property status: bool
```

```
class pycodevices_rt2.EcoDevicesRT2(host: str, port: int = 80, apikey: str = "", timeout: int = 10,  
                                     cached_ms: int = 0)
```

Bases: object

Class representing the Ecodevices RT2 and its API

```
property apikey
```

Return the apikey.

property apiurl

Return the default apiurl.

property cached_ms

Return the maximum cached value in milliseconds.

get(*command*, *command_value*=None, *command_entry*=None, *cached_ms*: *Optional[int]* = None)

Get value from api : http://{host}:{port}/api/xdevices.json?key={apikey}&{command}={command_value}, then get value {command_entry} in JSON response.

get_all_cached()**property host**

Return the hostname.

ping() → bool

class pycodelices_rt2.**EnOceanSensor**(*ecort2*: [EcoDevicesRT2](#), *id*: *int*)

Bases: [AbstractSensor](#)

Class representing the EnOceanSensor

class pycodelices_rt2.**EnOceanSwitch**(*ecort2*: [EcoDevicesRT2](#), *id*: *int*)

Bases: [AbstractSwitch](#)

Class representing the EnOceanSwitch

class pycodelices_rt2.**Post**(*ecort2*: [EcoDevicesRT2](#), *id_post*: *int*, *id_subpost*: *Optional[int]* = None)

Bases: object

Class representing the Post or Sub-Post

get_index(*cached_ms*: *Optional[int]* = None) → float

Return the index of post/subpost.

get_index_day(*cached_ms*: *Optional[int]* = None) → float

Return the index of the current day of post/subpost.

get_instant(*cached_ms*: *Optional[int]* = None) → float

Return the instant power of post/subpost.

get_price(*cached_ms*: *Optional[int]* = None) → float

Return the price of post/subpost.

get_price_day(*cached_ms*: *Optional[int]* = None) → float

Return the price of the current day of post/subpost.

property index: float

property index_day: float

property instant: float

property price: float

property price_day: float

class pycodelices_rt2.**Relay**(*ecort2*: [EcoDevicesRT2](#), *id*: *int*)

Bases: [AbstractSwitch](#)

Class representing the Relay

class pyecodevices_rt2.**SupplierIndex**(ecort2: [EcoDevicesRT2](#), id: int)

Bases: [AbstractSensor](#)

Class representing the SupplierIndex

get_price(cached_ms: Optional[int] = None) → float

Return the price of supplier index.

property price: float

class pyecodevices_rt2.**Toroid**(ecort2: [EcoDevicesRT2](#), id: int)

Bases: object

Class representing the Toroid

get_price(cached_ms: Optional[int] = None) → float

Return the price of toroid.

get_value(cached_ms: Optional[int] = None) → float

Return the index of toroid.

property price: float

property value: float

class pyecodevices_rt2.**VirtualOutput**(ecort2: [EcoDevicesRT2](#), id: int)

Bases: [AbstractSwitch](#)

Class representing the VirtualOutput

class pyecodevices_rt2.**X4FP**(ecort2: [EcoDevicesRT2](#), module_id: int, zone_id: int)

Bases: object

Class representing the X4FP

get_mode(cached_ms: Optional[int] = None) → int

Return the current X4FP mode.

property mode: int

class pyecodevices_rt2.**XTHL**(ecort2: [EcoDevicesRT2](#), id: int)

Bases: object

Class representing the XTHL

get_humidity(cached_ms: Optional[int] = None) → bool

Return the current XTHL humidity.

get_luminosity(cached_ms: Optional[int] = None) → bool

Return the current XTHL luminosity.

get_temperature(cached_ms: Optional[int] = None) → bool

Return the current XTHL temperature.

property humidity: bool

property luminosity: bool

property temperature: bool

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/pcourbin/pyecodevices_rt2/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

pyecodevices-rt2 could always use more documentation, whether as part of the official pyecodevices-rt2 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/pcourbin/pycodevices_rt2/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 *pycodevices_rt2* for local development.

1. Fork the *pycodevices_rt2* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/pycodevices_rt2.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 pycodevices_rt2
$ cd pycodevices_rt2/
$ 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 pycodevices_rt2 tests
$ python setup.py test or pytest
$ 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 3.5, 3.6, 3.7 and 3.8, and for PyPy. Check https://travis-ci.com/pcourbin/pycodelices_rt2/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ pytest tests.test_pycodelices_rt2
```

5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CREDITS

6.1 Development Lead

- Pierre COURBIN <pierre.courbin@gmail.com>

6.2 Contributors

None yet. Why not be the first?

HISTORY

7.1 1.3.1 (2022-08-07)

- Add option with `cached_ms<0` to force use the cache, or return None

7.2 1.3.0 (2022-08-07)

- Update Toroid API, using new EcoRT2 version 3.00.02

7.3 1.2.1 (2021-05-15)

- Add “get_all_cached” function to call all resquests to get a cached value.

7.4 1.2.0 (2021-05-14)

- Add “cached” possibilities to reduce the number of call to the API.
- The cached possibilities can be defined directly to the `ecodevices_rt2` object (applicable to each call), or to a specific call on a property.

7.5 1.1.0 (2021-04-17)

- Add classes such as *Counter*, *DigitalInput*, *EnOcean Switch or Sensor*, *Post and Sub-Post*, *Relay*, *SupplierIndex*, *Toroid*, *VirtualOutput*, *X4FP (Heaters)*, *XTHL* for ease of use
- Add tests to cover majority of code
- Add full examples in documentation

7.6 1.0.1 (2021-04-12)

- Update package with [Cookiecutter](#) and the [audreyr/cookiecutter-pypackage](#) project template.

7.7 1.0.0 (2021-04-08)

- First release on PyPI.

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

p

- `pyecodevices_rt2`, [19](#)
- `pyecodevices_rt2.abstractsensor`, [15](#)
- `pyecodevices_rt2.abstractswitch`, [15](#)
- `pyecodevices_rt2.const`, [16](#)
- `pyecodevices_rt2.counter`, [16](#)
- `pyecodevices_rt2.digitalinput`, [16](#)
- `pyecodevices_rt2.ecodevices_rt2`, [16](#)
- `pyecodevices_rt2.enocean`, [17](#)
- `pyecodevices_rt2.exceptions`, [17](#)
- `pyecodevices_rt2.post`, [17](#)
- `pyecodevices_rt2.relay`, [18](#)
- `pyecodevices_rt2.supplierindex`, [18](#)
- `pyecodevices_rt2.toroid`, [18](#)
- `pyecodevices_rt2.virtualoutput`, [19](#)
- `pyecodevices_rt2.x4fp`, [19](#)
- `pyecodevices_rt2.xthl`, [19](#)

A

AbstractSensor (class in *pycodevices_rt2*), 19
AbstractSensor (class in *pycodevices_rt2.abstractsensor*), 15
AbstractSwitch (class in *pycodevices_rt2*), 19
AbstractSwitch (class in *pycodevices_rt2.abstractswitch*), 15
add() (*pycodevices_rt2.Counter* method), 20
add() (*pycodevices_rt2.counter.Counter* method), 16
apikey (*pycodevices_rt2.ecodevices_rt2.EcoDevicesRT2* property), 16
apikey (*pycodevices_rt2.EcoDevicesRT2* property), 20
apiurl (*pycodevices_rt2.ecodevices_rt2.EcoDevicesRT2* property), 16
apiurl (*pycodevices_rt2.EcoDevicesRT2* property), 20

C

cached_ms (*pycodevices_rt2.ecodevices_rt2.EcoDevicesRT2* property), 16
cached_ms (*pycodevices_rt2.EcoDevicesRT2* property), 21
Counter (class in *pycodevices_rt2*), 20
Counter (class in *pycodevices_rt2.counter*), 16

D

DigitalInput (class in *pycodevices_rt2*), 20
DigitalInput (class in *pycodevices_rt2.digitalinput*), 16

E

EcoDevicesRT2 (class in *pycodevices_rt2*), 20
EcoDevicesRT2 (class in *pycodevices_rt2.ecodevices_rt2*), 16
EcoDevicesRT2ConnectError, 17
EcoDevicesRT2RequestError, 17
EnOceanSensor (class in *pycodevices_rt2*), 21
EnOceanSensor (class in *pycodevices_rt2.enocean*), 17
EnOceanSwitch (class in *pycodevices_rt2*), 21
EnOceanSwitch (class in *pycodevices_rt2.enocean*), 17

G

get() (*pycodevices_rt2.ecodevices_rt2.EcoDevicesRT2*

method), 16
get() (*pycodevices_rt2.EcoDevicesRT2* method), 21
get_all_cached() (*pycodevices_rt2.ecodevices_rt2.EcoDevicesRT2* method), 17
get_all_cached() (*pycodevices_rt2.EcoDevicesRT2* method), 21
get_humidity() (*pycodevices_rt2.XTHL* method), 22
get_humidity() (*pycodevices_rt2.xthl.XTHL* method), 19
get_index() (*pycodevices_rt2.Post* method), 21
get_index() (*pycodevices_rt2.post.Post* method), 17
get_index_day() (*pycodevices_rt2.Post* method), 21
get_index_day() (*pycodevices_rt2.post.Post* method), 17
get_instant() (*pycodevices_rt2.Post* method), 21
get_instant() (*pycodevices_rt2.post.Post* method), 17
get_luminosity() (*pycodevices_rt2.XTHL* method), 22
get_luminosity() (*pycodevices_rt2.xthl.XTHL* method), 19
get_mode() (*pycodevices_rt2.X4FP* method), 22
get_mode() (*pycodevices_rt2.x4fp.X4FP* method), 19
get_price() (*pycodevices_rt2.Counter* method), 20
get_price() (*pycodevices_rt2.counter.Counter* method), 16
get_price() (*pycodevices_rt2.Post* method), 21
get_price() (*pycodevices_rt2.post.Post* method), 17
get_price() (*pycodevices_rt2.SupplierIndex* method), 22
get_price() (*pycodevices_rt2.supplierindex.SupplierIndex* method), 18
get_price() (*pycodevices_rt2.Toroid* method), 22
get_price() (*pycodevices_rt2.toroid.Toroid* method), 18
get_price_day() (*pycodevices_rt2.Post* method), 21
get_price_day() (*pycodevices_rt2.post.Post* method), 17
get_status() (*pycodevices_rt2.AbstractSwitch* method), 20

`get_status()` (*pycodevices_rt2.abstractswitch.AbstractSwitch method*), 15

`get_status()` (*pycodevices_rt2.DigitalInput method*), 20

`get_status()` (*pycodevices_rt2.digitalinput.DigitalInput method*), 16

`get_temperature()` (*pycodevices_rt2.XTHL method*), 22

`get_temperature()` (*pycodevices_rt2.xthl.XTHL method*), 19

`get_value()` (*pycodevices_rt2.AbstractSensor method*), 19

`get_value()` (*pycodevices_rt2.abstractsensor.AbstractSensor method*), 15

`get_value()` (*pycodevices_rt2.Counter method*), 20

`get_value()` (*pycodevices_rt2.counter.Counter method*), 16

`get_value()` (*pycodevices_rt2.Toroid method*), 22

`get_value()` (*pycodevices_rt2.toroid.Toroid method*), 18

H

`host` (*pycodevices_rt2.ecodevices_rt2.EcoDevicesRT2 property*), 17

`host` (*pycodevices_rt2.EcoDevicesRT2 property*), 21

`humidity` (*pycodevices_rt2.XTHL property*), 22

`humidity` (*pycodevices_rt2.xthl.XTHL property*), 19

I

`index` (*pycodevices_rt2.Post property*), 21

`index` (*pycodevices_rt2.post.Post property*), 18

`index_day` (*pycodevices_rt2.Post property*), 21

`index_day` (*pycodevices_rt2.post.Post property*), 18

`instant` (*pycodevices_rt2.Post property*), 21

`instant` (*pycodevices_rt2.post.Post property*), 18

L

`luminosity` (*pycodevices_rt2.XTHL property*), 22

`luminosity` (*pycodevices_rt2.xthl.XTHL property*), 19

M

`mode` (*pycodevices_rt2.X4FP property*), 22

`mode` (*pycodevices_rt2.x4fp.X4FP property*), 19

`module`

- `pycodevices_rt2`, 19
- `pycodevices_rt2.abstractsensor`, 15
- `pycodevices_rt2.abstractswitch`, 15
- `pycodevices_rt2.const`, 16
- `pycodevices_rt2.counter`, 16
- `pycodevices_rt2.digitalinput`, 16

- `pycodevices_rt2.ecodevices_rt2`, 16
- `pycodevices_rt2.enocean`, 17
- `pycodevices_rt2.exceptions`, 17
- `pycodevices_rt2.post`, 17
- `pycodevices_rt2.relay`, 18
- `pycodevices_rt2.supplierindex`, 18
- `pycodevices_rt2.toroid`, 18
- `pycodevices_rt2.virtualoutput`, 19
- `pycodevices_rt2.x4fp`, 19
- `pycodevices_rt2.xthl`, 19

O

`off()` (*pycodevices_rt2.AbstractSwitch method*), 20

`off()` (*pycodevices_rt2.abstractswitch.AbstractSwitch method*), 15

`on()` (*pycodevices_rt2.AbstractSwitch method*), 20

`on()` (*pycodevices_rt2.abstractswitch.AbstractSwitch method*), 15

P

`ping()` (*pycodevices_rt2.ecodevices_rt2.EcoDevicesRT2 method*), 17

`ping()` (*pycodevices_rt2.EcoDevicesRT2 method*), 21

`Post` (*class in pycodevices_rt2*), 21

`Post` (*class in pycodevices_rt2.post*), 17

`price` (*pycodevices_rt2.Counter property*), 20

`price` (*pycodevices_rt2.counter.Counter property*), 16

`price` (*pycodevices_rt2.Post property*), 21

`price` (*pycodevices_rt2.post.Post property*), 18

`price` (*pycodevices_rt2.SupplierIndex property*), 22

`price` (*pycodevices_rt2.supplierindex.SupplierIndex property*), 18

`price` (*pycodevices_rt2.Toroid property*), 22

`price` (*pycodevices_rt2.toroid.Toroid property*), 18

`price_day` (*pycodevices_rt2.Post property*), 21

`price_day` (*pycodevices_rt2.post.Post property*), 18

`pycodevices_rt2`

- `module`, 19

`pycodevices_rt2.abstractsensor`

- `module`, 15

`pycodevices_rt2.abstractswitch`

- `module`, 15

`pycodevices_rt2.const`

- `module`, 16

`pycodevices_rt2.counter`

- `module`, 16

`pycodevices_rt2.digitalinput`

- `module`, 16

`pycodevices_rt2.ecodevices_rt2`

- `module`, 16

`pycodevices_rt2.enocean`

- `module`, 17

`pycodevices_rt2.exceptions`

- `module`, 17

pycodevices_rt2.post
 module, 17
 pycodevices_rt2.relay
 module, 18
 pycodevices_rt2.supplierindex
 module, 18
 pycodevices_rt2.toroid
 module, 18
 pycodevices_rt2.virtualoutput
 module, 19
 pycodevices_rt2.x4fp
 module, 19
 pycodevices_rt2.xthl
 module, 19

R

Relay (class in pycodevices_rt2), 21
 Relay (class in pycodevices_rt2.relay), 18

S

status (pycodevices_rt2.AbstractSwitch property), 20
 status (pycodevices_rt2.abstractswitch.AbstractSwitch
 property), 15
 status (pycodevices_rt2.DigitalInput property), 20
 status (pycodevices_rt2.digitalinput.DigitalInput
 property), 16
 substrat() (pycodevices_rt2.Counter method), 20
 substrat() (pycodevices_rt2.counter.Counter
 method), 16
 SupplierIndex (class in pycodevices_rt2), 21
 SupplierIndex (class in pycode-
 vices_rt2.supplierindex), 18

T

temperature (pycodevices_rt2.XTHL property), 22
 temperature (pycodevices_rt2.xthl.XTHL property),
 19
 toggle() (pycodevices_rt2.AbstractSwitch method), 20
 toggle() (pycodevices_rt2.abstractswitch.AbstractSwitch
 method), 15
 Toroid (class in pycodevices_rt2), 22
 Toroid (class in pycodevices_rt2.toroid), 18

V

value (pycodevices_rt2.AbstractSensor property), 19
 value (pycodevices_rt2.abstractsensor.AbstractSensor
 property), 15
 value (pycodevices_rt2.Counter property), 20
 value (pycodevices_rt2.counter.Counter property), 16
 value (pycodevices_rt2.Toroid property), 22
 value (pycodevices_rt2.toroid.Toroid property), 18
 VirtualOutput (class in pycodevices_rt2), 22
 VirtualOutput (class in pycode-
 vices_rt2.virtualoutput), 19

X

X4FP (class in pycodevices_rt2), 22
 X4FP (class in pycodevices_rt2.x4fp), 19
 XTHL (class in pycodevices_rt2), 22
 XTHL (class in pycodevices_rt2.xthl), 19