

---

# **python-cozify Documentation**

***Release v0.2.13***

**Juho-Pekka Kuitunen**

**Mar 24, 2018**



---

## Contents:

---

<b>1</b>	<b>Low-level Cloud API calls</b>	<b>3</b>
<b>2</b>	<b>Raised Exceptions</b>	<b>5</b>
<b>3</b>	<b>High-level Hub functions</b>	<b>7</b>
<b>4</b>	<b>Low-level Hub API calls</b>	<b>11</b>
<b>5</b>	<b>python-cozify</b>	<b>15</b>
5.1	Installation . . . . .	15
5.2	Basic usage . . . . .	15
5.3	On Capabilities . . . . .	16
5.4	Keeping authentication valid . . . . .	16
5.5	Working Remotely . . . . .	17
5.6	Using Multiple Hubs . . . . .	17
5.7	Encoding Pitfalls . . . . .	17
5.8	Sample projects . . . . .	17
5.9	Development . . . . .	18
	<b>Python Module Index</b>	<b>19</b>



Module for handling Cozify Cloud highlevel operations.

`cozify.cloud.authenticate` (*trustCloud=True, trustHub=True, remote=False, autoremove=True*)  
Authenticate with the Cozify Cloud and Hub.

Interactive only when absolutely needed, mostly on the first run. By default authentication is run selectively only for the portions needed. Hub authentication lives in the Cloud module since the authentication is obtained from the cloud.

**Authentication is a multistep process:**

- trigger sending OTP to email address
- perform email login with OTP to acquire cloud token
- acquire hub information and authenticate with hub with cloud token
- store hub token for further use

**Parameters**

- **trustCloud** (*bool*) – Trust current stored state of cloud auth. Default True.
- **trustHub** (*bool*) – Trust current stored state of hub auth. Default True.
- **remote** (*bool*) – Treat a hub as being outside the LAN, i.e. calls will be routed via the Cozify Cloud remote call system. Defaults to False.
- **autoremove** (*bool*) – Autodetect hub LAN presence and flip to remote mode if needed. Defaults to True.

**Returns** True on authentication success. Failure will result in an exception.

**Return type** bool

`cozify.cloud.email` (*new\_email=None*)  
Get currently used cloud account email or set a new one.

**Returns** Cloud user account email address.

**Return type** str

`cozify.cloud.ping` (*autorefresh=True, expiry=None*)  
Test cloud token validity. On success will also trigger a refresh if it's needed by the current key expiry.

**Parameters**

- **refresh** (*bool*) – Wether to perform a autorefresh check after a successful ping. Defaults to True.
- **expiry** (*datetime.timedelta*) – timedelta object for duration how often cloud\_token will be auto-refreshed when cloud.ping() is called. If not set, cloud.refresh() defaults are used.

**Returns** validity of stored token.

**Return type** bool

`cozify.cloud.refresh` (*force=False, expiry=datetime.timedelta(1)*)  
Renew current cloud token and store new token in state.

This call will only succeed if the current cloud token is still valid. A new refreshed token is requested from the API only if sufficient time has passed since the previous refresh.

**Parameters**

- **force** (*bool*) – Set to True to always perform a refresh regardless of time passed since previous refresh.
- **expiry** (*datetime.timedelta*) – timedelta object for duration of refresh expiry. Defaults to one day.

**Returns** Success of refresh attempt, True also when expiry wasn't over yet even though no refresh was performed.

**Return type** bool

`cozify.cloud.resetState()`

Reset stored cloud state.

Any further authentication flow will start from a clean slate. Hub state is left intact.

`cozify.cloud.token(new_token=None)`

Get currently used cloud\_token or set a new one.

**Returns** Cloud remote authentication token.

**Return type** str

# CHAPTER 1

---

## Low-level Cloud API calls

---

Module for handling Cozify Cloud API 1:1 functions

`cozify.cloud_api.cloudBase`

*str* – API endpoint including version

`cozify.cloud_api.emaillogin(email, otp)`

Raw Cloud API call, request cloud token with email address & OTP.

**Parameters**

- **email** (*str*) – Email address connected to Cozify account.
- **otp** (*int*) – One time passcode.

**Returns** cloud token

**Return type** *str*

`cozify.cloud_api.hubkeys(cloud_token)`

1:1 implementation of user/hubkeys

**Parameters** **cloud\_token** (*str*) –

**Returns** Map of hub\_id: hub\_token pairs.

**Return type** *dict*

`cozify.cloud_api.lan_ip()`

1:1 implementation of hub/lan\_ip

This call will fail with an `APIError` if the requesting source address is not the same as that of the hub, i.e. if they're not in the same NAT network. The above is based on observation and may only be partially true.

**Returns** List of Hub ip addresses.

**Return type** *list*

`cozify.cloud_api.refreshsession(cloud_token)`

1:1 implementation of user/refreshsession

**Parameters** **cloud\_token** (*str*) –

**Returns** New cloud remote authentication token. Not automatically stored into state.

**Return type** str

`cozify.cloud_api.remote` (*cloud\_token*, *hub\_token*, *apicall*, *payload=None*, *\*\*kwargs*)  
1:1 implementation of 'hub/remote'

**Parameters**

- **cloud\_token** (*str*) – Cloud remote authentication token.
- **hub\_token** (*str*) – Hub authentication token.
- **apicall** (*str*) – Full API call that would normally go directly to hub, e.g. `/cc/1.6/hub/colors`
- **payload** (*str*) – json string to use as payload, changes method to PUT.

**Returns** Requests response object.

**Return type** requests.response

`cozify.cloud_api.requestlogin` (*email*)  
Raw Cloud API call, request OTP to be sent to account email address.

**Parameters** **email** (*str*) – Email address connected to Cozify account.



---

### Raised Exceptions

---

**exception** `cozify.Error.APIError` (*status\_code*, *message*)

Error raised for non-200 API return codes

**Parameters**

- **status\_code** (*int*) – HTTP status code returned by the API
- **message** (*str*) – Potential error message returned by the API

**status\_code**

*int* – HTTP status code returned by the API

**message**

*str* – Potential error message returned by the API

**exception** `cozify.Error.AuthenticationError` (*message*)

Error raised for nonrecoverable authentication failures.

**Parameters** **message** (*str*) – Human readable error description

**message**

*str* – Human readable error description



---

## High-level Hub functions

---

Module for handling highlevel Cozify Hub operations.

`cozify.hub.capability`

*capability* – Enum of known device capabilities. Alphabetically sorted, numeric value not guaranteed to stay constant between versions if new capabilities are added.

`cozify.hub.autoremove (hub_id, new_state=None)`

Get autoremove status of matching `hub_id` or set a new value for it.

**Parameters** `hub_id (str)` – Id of hub to query. The id is a string of hexadecimal sections used internally to represent a hub.

**Returns** True for a hub with autoremove enabled.

**Return type** bool

`class cozify.hub.capability`

An enumeration.

`cozify.hub.default ()`

Return id of default Hub.

If default hub isn't known an `AttributeError` will be raised.

`cozify.hub.device_off (device_id, **kwargs)`

Turn off a device that is capable of turning off. Eligibility is determined by the capability `ON_OFF`.

**Parameters** `device_id (str)` – ID of the device to operate on.

`cozify.hub.device_on (device_id, **kwargs)`

Turn on a device that is capable of turning on. Eligibility is determined by the capability `ON_OFF`.

**Parameters** `device_id (str)` – ID of the device to operate on.

`cozify.hub.device_toggle (device_id, **kwargs)`

Toggle power state of any device capable of it such as lamps. Eligibility is determined by the capability `ON_OFF`.

**Parameters**

- **device\_id** (*str*) – ID of the device to toggle.
- **\*\*hub\_id** (*str*) – optional id of hub to operate on. A specified hub\_id takes presedence over a hub\_name or default Hub.
- **\*\*hub\_name** (*str*) – optional name of hub to operate on.
- **\*\*remote** (*bool*) – Remote or local query.

`cozify.hub.devices(*, capabilities=None, and_filter=False, **kwargs)`

Get up to date full devices data set as a dict. Optionally can be filtered to only include certain devices.

#### Parameters

- **capabilities** (`cozify.hub.capability`) – Single or list of `cozify.hub.capability` types to filter by, for example: [ `cozify.hub.capability.TEMPERATURE`, `cozify.hub.capability.HUMIDITY` ]. Defaults to no filtering.
- **and\_filter** (*bool*) – Multi-filter by AND instead of default OR. Defaults to False.
- **\*\*hub\_name** (*str*) – optional name of hub to query. Will get converted to hubId for use.
- **\*\*hub\_id** (*str*) – optional id of hub to query. A specified hub\_id takes presedence over a hub\_name or default Hub. Providing incorrect hub\_id's will create cruft in your state but it won't hurt anything beyond failing the current operation.
- **\*\*remote** (*bool*) – Remote or local query.
- **\*\*hubId** (*str*) – Deprecated. Compatibility keyword for hub\_id, to be removed in v0.3
- **\*\*hubName** (*str*) – Deprecated. Compatibility keyword for hub\_name, to be removed in v0.3

**Returns** full live device state as returned by the API

**Return type** dict

`cozify.hub.getDefaultHub()`

Deprecated, use default(). Return id of default Hub.

`cozify.hub.getDevices(**kwargs)`

Deprecated, will be removed in v0.3. Get up to date full devices data set as a dict.

#### Parameters

- **\*\*hub\_name** (*str*) – optional name of hub to query. Will get converted to hubId for use.
- **\*\*hub\_id** (*str*) – optional id of hub to query. A specified hub\_id takes presedence over a hub\_name or default Hub. Providing incorrect hub\_id's will create cruft in your state but it won't hurt anything beyond failing the current operation.
- **\*\*remote** (*bool*) – Remote or local query.
- **\*\*hubId** (*str*) – Deprecated. Compatibility keyword for hub\_id, to be removed in v0.3
- **\*\*hubName** (*str*) – Deprecated. Compatibility keyword for hub\_name, to be removed in v0.3

**Returns** full live device state as returned by the API

**Return type** dict

`cozify.hub.getHubId(hub_name)`

Deprecated, use hub\_id(). Return id of hub by it's name.

#### Parameters

- **hub\_name** (*str*) – Name of hub to query. The name is given when registering a hub to an account.
- **str** – hub\_id on success, raises an attributeerror on failure.

**Returns** Hub id or raises

**Return type** str

`cozify.hub.host(hub_id)`

Get hostname of matching hub\_id

**Parameters** **hub\_id** (*str*) – Id of hub to query. The id is a string of hexadecimal sections used internally to represent a hub.

**Returns** ip address of matching hub. Be aware that this may be empty if the hub is only known remotely and will still give you an ip address even if the hub is currently remote and an ip address was previously locally known.

**Return type** str

`cozify.hub.hub_id(hub_name)`

Get hub id by it's name.

**Parameters** **hub\_name** (*str*) – Name of hub to query. The name is given when registering a hub to an account.

**Returns** hub\_id on success, raises an attributeerror on failure.

**Return type** str

`cozify.hub.light_brightness(device_id, brightness, transition=0, **kwargs)`

Set brightness of a light.

**Parameters**

- **device\_id** (*str*) – ID of the device to operate on.
- **brightness** (*float*) – Brightness in the range of [0, 1]. If outside the range an AttributeError is raised.
- **transition** (*int*) – Transition length in milliseconds. Defaults to instant.

`cozify.hub.light_color(device_id, hue, saturation=1.0, transition=0, **kwargs)`

Set color (hue & saturation) of a light.

**Parameters**

- **device\_id** (*str*) – ID of the device to operate on.
- **hue** (*float*) – Hue in the range of [0, Pi\*2]. If outside the range an AttributeError is raised.
- **saturation** (*float*) – Saturation in the range of [0, 1]. If outside the range an AttributeError is raised. Defaults to 1.0 (full saturation.)
- **transition** (*int*) – Transition length in milliseconds. Defaults to instant.

`cozify.hub.light_temperature(device_id, temperature=2700, transition=0, **kwargs)`

Set temperature of a light.

**Parameters**

- **device\_id** (*str*) – ID of the device to operate on.
- **temperature** (*float*) – Temperature in Kelvins. If outside the operating range of the device the extreme value is used. Defaults to 2700K.

- **transition** (*int*) – Transition length in milliseconds. Defaults to instant.

`cozify.hub.name (hub_id)`

Get hub name by it's id.

**Parameters** **hub\_id** (*str*) – Id of hub to query. The id is a string of hexadecimal sections used internally to represent a hub.

**Returns** Hub name or None if the hub wasn't found.

**Return type** `str`

`cozify.hub.ping (**kwargs)`

Perform a cheap API call to trigger any potential APIError and return boolean for success/failure. For optional kwargs see `cozify.hub_api.get()`

**Parameters**

- **\*\*hub\_id** (*str*) – Hub to ping or default if neither id or name set.
- **\*\*hub\_name** (*str*) – Hub to ping by name.

**Returns** True for a valid and working hub authentication state.

**Return type** `bool`

`cozify.hub.remote (hub_id, new_state=None)`

Get remote status of matching `hub_id` or set a new value for it.

**Parameters** **hub\_id** (*str*) – Id of hub to query. The id is a string of hexadecimal sections used internally to represent a hub.

**Returns** True for a hub considered remote.

**Return type** `bool`

`cozify.hub.token (hub_id, new_token=None)`

Get `hub_token` of matching `hub_id` or set a new value for it.

**Parameters** **hub\_id** (*str*) – Id of hub to query. The id is a string of hexadecimal sections used internally to represent a hub.

**Returns** Hub authentication token.

**Return type** `str`

`cozify.hub.tz (**kwargs)`

Get timezone of given hub or default hub if no id is specified. For more optional kwargs see `cozify.hub_api.get()`

Args: **\*\*hub\_id**(*str*): Hub to query, by default the default hub is used.

**Returns** Timezone of the hub, for example: 'Europe/Helsinki'

**Return type** `str`

---

## Low-level Hub API calls

---

Module for all Cozify Hub API 1:1 calls

`cozify.hub_api.apiPath`

*str* – Hub API endpoint path including version. Things may suddenly stop working if a software update increases the API version on the Hub. Incrementing this value until things work will get you by until a new version is published.

`cozify.hub_api.devices(**kwargs)`

1:1 implementation of /devices API call. For remaining kwargs see `cozify.hub_api.get()`

**Parameters** **\*\*mock\_devices** (*dict*) – If defined, returned as-is as if that were the result we received.

**Returns** Full live device state as returned by the API

**Return type** *dict*

`cozify.hub_api.devices_command(command, **kwargs)`

1:1 implementation of /devices/command. For kwargs see `cozify.hub_api.put()`

**Parameters** **command** (*dict*) – dictionary of type *DeviceData* containing the changes wanted. Will be converted to json.

**Returns** What ever the API replied or raises an *APIError* on failure.

**Return type** *str*

`cozify.hub_api.devices_command_generic(*, device_id, command=None, request_type, **kwargs)`

Command helper for CMD type of actions. No checks are made whether the device supports the command or not. For kwargs see `cozify.hub_api.put()`

**Parameters**

- **device\_id** (*str*) – ID of the device to operate on.
- **request\_type** (*str*) – Type of CMD to run, e.g. *CMD\_DEVICE\_OFF*

- **command** (*dict*) – Optional dictionary to override command sent. Defaults to None which is interpreted as { device\_id, type }

**Returns** What ever the API replied or raises an APIError on failure.

**Return type** str

`cozify.hub_api.devices_command_off(device_id, **kwargs)`

Command helper for CMD\_DEVICE\_OFF.

**Parameters** **device\_id** (*str*) – ID of the device to operate on.

**Returns** What ever the API replied or raises an APIException on failure.

**Return type** str

`cozify.hub_api.devices_command_on(device_id, **kwargs)`

Command helper for CMD\_DEVICE\_ON.

**Parameters** **device\_id** (*str*) – ID of the device to operate on.

**Returns** What ever the API replied or raises an APIError on failure.

**Return type** str

`cozify.hub_api.devices_command_state(*, device_id, state, **kwargs)`

Command helper for CMD type of actions. No checks are made whether the device supports the command or not. For kwargs see `cozify.hub_api.put()`

**Parameters**

- **device\_id** (*str*) – ID of the device to operate on.
- **state** (*dict*) – New state dictionary containing changes.

**Returns** What ever the API replied or raises an APIError on failure.

**Return type** str

`cozify.hub_api.get(call, hub_token_header=True, base='/cc/1.8', **kwargs)`

GET method for calling hub API.

**Parameters**

- **call** (*str*) – API path to call after apiPath, needs to include leading /.
- **hub\_token\_header** (*bool*) – Set to False to omit hub\_token usage in call headers.
- **base** (*str*) – Base path to call from API instead of global apiPath. Defaults to apiPath.
- **\*\*host** (*str*) – ip address or hostname of hub.
- **\*\*hub\_token** (*str*) – Hub authentication token.
- **\*\*remote** (*bool*) – If call is to be local or remote (bounced via cloud).
- **\*\*cloud\_token** (*str*) – Cloud authentication token. Only needed if remote = True.

`cozify.hub_api.hub(**kwargs)`

1:1 implementation of /hub API call. For kwargs see `cozify.hub_api.get()`

**Returns** Hub state dict.

**Return type** dict

`cozify.hub_api.put(call, payload, hub_token_header=True, base='/cc/1.8', **kwargs)`

PUT method for calling hub API. For rest of kwargs parameters see `get()`

**Parameters**



- **call** (*str*) – API path to call after apiPath, needs to include leading /.
- **payload** (*str*) – json string to push out as the payload.
- **hub\_token\_header** (*bool*) – Set to False to omit hub\_token usage in call headers.
- **base** (*str*) – Base path to call from API instead of global apiPath. Defaults to apiPath.

`cozify.hub_api.tz(**kwargs)`

1:1 implementation of /hub/tz API call. For kwargs see `cozify.hub_api.get()`

**Returns** Timezone of the hub, for example: 'Europe/Helsinki'

**Return type** str



Unofficial Python3 API bindings for the (unpublished) Cozify API. Includes high-level helpers for easier use of the APIs, for example an automatic authentication flow, and low-level 1:1 API functions.

## 5.1 Installation

The recommended way is to install from PyPi:

```
sudo -H pip3 install cozify
```

or clone the master branch of this repo (master stays at current release) and:

```
sudo python3 setup.py install
```

To develop python-cozify clone the devel branch and submit pull requests against the devel branch. New releases are cut from the devel branch as needed.

## 5.2 Basic usage

These are merely some simple examples, for the full documentation see: <http://python-cozify.readthedocs.io/en/latest/>

### 5.2.1 read devices by capability, print temperature data

```
from cozify import hub
devices = hub.devices(capabilities=hub.capability.TEMPERATURE)
for id, dev in devices.items():
    print('{0}: {1}C'.format(dev['name'], dev['state']['temperature']))
```

## 5.2.2 only authenticate

```
from cozify import cloud
cloud.authenticate()
# authenticate() is interactive and usually triggered automatically
# authentication data is stored in ~/.config/python-cozify/python-cozify.cfg
```

## 5.2.3 authenticate with a non-default state storage

```
from cozify import cloud, config
config.setStatePath('/tmp/testing-state.cfg')
cloud.authenticate()
# authentication and other useful data is now stored in the defined location instead
# of ~/.config/python-cozify/python-cozify.cfg
# you could also use the environment variable XDG_CONFIG_HOME to override where
# config files are stored
```

## 5.3 On Capabilities

The most practical way to “find” devices for operating on is currently to filter the devices list by their capabilities. The most up to date list of recognized capabilities can be seen at [cozify/hub.py](#)

If the capability you need is not yet supported, open a bug to get it added. One way to compare your live hub device’s capabilities to those implemented is running the `util/capabilities_list.py` tool. It will list implemented and gathered capabilities from your live environment. To get all of your previously unknown capabilities implemented, just copy-paste the full output of the utility into a new bug.

In short capabilities are tags assigned to devices by Cozify that mostly guarantee the data related to that capability will be in the same format and structure. For example the capabilities based example code in this document filters all the devices that claim to support temperature and reads their name and temperature state. Multiple capabilities can be given in a filter by providing a list of capabilities. By default any capability in the list can match (OR filter) but it can be flipped to AND mode where every capability must be present on a device for it to qualify. For example, if you only want multi-sensors that support both temperature and humidity monitoring you could define a filter as:

```
devices = hub.devices(capabilities=[ hub.capability.TEMPERATURE, hub.capability.
# HUMIDITY ], and_filter=True)
```

## 5.4 Keeping authentication valid

If the cloud token expires, the only option to get a new one is an interactive prompt for an OTP. Since most applications will want to avoid that as much as possible there are a few tips to keep a valid token alive. At the time of writing tokens are valid for 28 days during which they can be seamlessly refreshed.

In most cases it isn’t necessary to directly call `cloud.refresh()` if you’re already using `cloud.ping()` to test token validity. `cloud.ping()` will also perform a refresh check after a successful ping unless explicitly told not to do so.

To refresh a token you can call as often as you want:

```
cloud.refresh()
```

By default keys older than a day will be re-requested and otherwise no refresh is performed. The refresh can be forced:

```
cloud.refresh(force=True)
```

And the expiry duration can be altered (also when calling `cloud.ping()`):

```
cloud.refresh(expiry=datetime.timedelta(days=20))
# or
cloud.ping(autorefresh=True, expiry=datetime.timedelta(days=20))
```

## 5.5 Working Remotely

By default queries to the hub are attempted via local LAN. Also by default “remoteness” autodetection is on and thus if it is determined during `cloud.authentication()` or a `hub.ping()` call that you seem to not be in the same network, the state is flipped. Both the remote state and autodetection can be overridden in most if not all functions by the boolean keyword arguments ‘remote’ and ‘autoremove’. They can also be queried or permanently changed by the `hub.remote()` and `hub.autoremove()` functions.

## 5.6 Using Multiple Hubs

Everything has been designed to support multiple hubs registered to the same Cozify Cloud account. All hub operations can be targeted by setting the keyword argument ‘hub\_id’ or ‘hub\_name’. The developers do not as of yet have access to multiple hubs so proper testing of multi functionality has not been performed. If you run into trouble, please open bugs so things can be improved.

The remote state of hubs is kept separately so there should be no issues calling your home hub locally but operating on a summer cottage hub remotely at the same time.

## 5.7 Encoding Pitfalls

The hub provides data encoded as a utf-8 json string. Python-cozify transforms this into a Python dictionary where string values are kept as unicode strings. Normally this isn’t an issue, as long as your system supports utf-8. If not, you will run into trouble printing for example device names with non-ascii characters:

```
UnicodeEncodeError: 'ascii' codec can't encode character 'xe4' in position 34: ordinal not in range(128)
```

The solution is to change your system locale to support utf-8. How this is done is however system dependant. As a first test try temporarily overriding your locale:

```
LC_ALL='en_US.utf8' python3 program.py
```

## 5.8 Sample projects

- [github.com/Artanicus/cozify-temp](https://github.com/Artanicus/cozify-temp) - Store Multisensor data into InfluxDB
- Take a look at the `util/` directory for some crude small tools using the library that have been useful during development.
- File an issue to get your project added here

## 5.9 Development

To develop python-cozify clone the devel branch and submit pull requests against the devel branch. New releases are cut from the devel branch as needed.

### 5.9.1 Tests

pytest is used for unit tests. Test coverage is still quite spotty and under active development. Certain tests are marked as “live” tests and require an active authentication state and a real hub to query against. Live tests are non-destructive.

During development you can run the test suite right from the source directory:

```
pytest -v cozify/  
# or include the live tests as well:  
pytest -v cozify/ --live
```

To run the test suite on an already installed python-cozify:

```
pytest -v --pyargs cozify  
# or including live tests:  
pytest -v --pyargs cozify --live
```

### 5.9.2 Roadmap, aka. Current Limitations

- Authentication flow has been improved quite a bit but it would benefit a lot from real-world feedback.
- For now there are only read calls. Next up is implementing ~all hub calls at the raw level and then wrapping them for ease of use. If there’s something you want to use sooner than later file an issue so it can get prioritized!
- Device model is non-existent and the old implementations are bad and deprecated. Active work ongoing to filter by capability at a low level first, then perhaps a more object oriented model on top of that.

### C

`cozify.cloud`, ??  
`cozify.cloud_api`, 3  
`cozify.hub`, 7  
`cozify.hub_api`, 11





## A

APIError, 5  
apiPath (in module cozify.hub\_api), 11  
authenticate() (in module cozify.cloud), 1  
AuthenticationError, 5  
autoremove() (in module cozify.hub), 7

## C

capability (class in cozify.hub), 7  
capability (in module cozify.hub), 7  
cloudBase (in module cozify.cloud\_api), 3  
cozify.cloud (module), 1  
cozify.cloud\_api (module), 3  
cozify.hub (module), 7  
cozify.hub\_api (module), 11

## D

default() (in module cozify.hub), 7  
device\_off() (in module cozify.hub), 7  
device\_on() (in module cozify.hub), 7  
device\_toggle() (in module cozify.hub), 7  
devices() (in module cozify.hub), 8  
devices() (in module cozify.hub\_api), 11  
devices\_command() (in module cozify.hub\_api), 11  
devices\_command\_generic() (in module cozify.hub\_api), 11  
devices\_command\_off() (in module cozify.hub\_api), 12  
devices\_command\_on() (in module cozify.hub\_api), 12  
devices\_command\_state() (in module cozify.hub\_api), 12

## E

email() (in module cozify.cloud), 1  
emaillogin() (in module cozify.cloud\_api), 3

## G

get() (in module cozify.hub\_api), 12  
getDefaultHub() (in module cozify.hub), 8  
getDevices() (in module cozify.hub), 8  
getHubId() (in module cozify.hub), 8

## H

host() (in module cozify.hub), 9  
hub() (in module cozify.hub\_api), 12  
hub\_id() (in module cozify.hub), 9  
hubkeys() (in module cozify.cloud\_api), 3

## L

lan\_ip() (in module cozify.cloud\_api), 3  
light\_brightness() (in module cozify.hub), 9  
light\_color() (in module cozify.hub), 9  
light\_temperature() (in module cozify.hub), 9

## M

message (cozify.Error.APIError attribute), 5  
message (cozify.Error.AuthenticationError attribute), 5

## N

name() (in module cozify.hub), 10

## P

ping() (in module cozify.cloud), 1  
ping() (in module cozify.hub), 10  
put() (in module cozify.hub\_api), 12

## R

refresh() (in module cozify.cloud), 1  
refreshsession() (in module cozify.cloud\_api), 3  
remote() (in module cozify.cloud\_api), 4  
remote() (in module cozify.hub), 10  
requestlogin() (in module cozify.cloud\_api), 4  
resetState() (in module cozify.cloud), 2

## S

status\_code (cozify.Error.APIError attribute), 5

## T

token() (in module cozify.cloud), 2  
token() (in module cozify.hub), 10  
tz() (in module cozify.hub), 10  
tz() (in module cozify.hub\_api), 13