python-cozify Documentation

Release v0.2.34

Juho-Pekka Kuitunen

Jul 03, 2022

CONTENTS:

This page contains auto-generated API reference documentation[1].

[1] Created with sphinx-autoapi

CHAPTER

ONE

COZIFY

1.1 Subpackages

1.1.1 cozify.test

Submodules

cozify.test.debug

Set high log level

Module Contents

cozify.test.debug.**urllib3_logger**

cozify.test.fixtures

Module Contents

Classes

Tmp_hub	Creates a temporary hub section (with test data) in a tmp_cloud

Functions

vcr_config()	
tmp_cloud()	
live_cloud()	
blank_cloud()	
mock_server()	
tmp_hub(tmp_cloud)	
live_hub()	
offline_device()	
online_device()	
real_test_devices()	
real_test_scenes()	
_h6_dict(d)	

cozify.test.fixtures.**vcr_config**()

cozify.test.fixtures.**tmp_cloud**()

cozify.test.fixtures.**live_cloud**()

cozify.test.fixtures.**blank_cloud**()

cozify.test.fixtures.**mock_server**()

cozify.test.fixtures.**tmp_hub**(*tmp_cloud*)

cozify.test.fixtures.**live_hub**()

cozify.test.fixtures.**offline_device**()

cozify.test.fixtures.**online_device**()

cozify.test.fixtures.**real_test_devices**()

cozify.test.fixtures.**real_test_scenes**()

class cozify.test.fixtures.**Tmp_hub**(*tmp_cloud*)

Creates a temporary hub section (with test data) in a tmp_cloud

__enter__(*self*)

__exit__(*self*, *exc_type*, *exc_value*, *traceback*)

devices(*self*)

states(*self*)

cozify.test.fixtures.**_h6_dict**(*d*)

cozify.test.fixtures_devices

Module Contents

cozify.test.fixtures_devices.**lamp_ikea**

cozify.test.fixtures_devices.**lamp_osram**

cozify.test.fixtures_devices.**strip_osram**

cozify.test.fixtures_devices.**twilight_nexa**

cozify.test.fixtures_devices.**plafond_osram**

cozify.test.fixtures_devices.**state_clean**

cozify.test.fixtures_devices.**devices**

cozify.test.fixtures_devices.**device_ids**

cozify.test.fixtures_devices.**states**

1.2 Submodules

1.2.1 cozify.Error

Module Contents

exception cozify.Error.**APIError**(*status_code*, *message*)

Bases: Exception

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

HTTP status code returned by the API

Type
int

message

Potential error message returned by the API

Type
str

__str__(*self*)

Return str(self).

exception cozify.Error.**ConnectionError**(*message*)

Bases: *ConnectionError*

Error raised for connection level failures, such as a lost internet connection.

Parameters
message (*str*) – Potential error message returned by the requests library

message

Potential error message returned by the requests library

Type
str

__str__(*self*)

Return str(self).

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

Bases: Exception

Error raised for nonrecoverable authentication failures.

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

message

Human readable error description

Type
str

__str__(*self*)

Return str(self).

1.2.2 cozify.cloud

Module for handling Cozify Cloud highlevel operations.

Module Contents

Functions

authenticate(trustCloud=True, trustHub=True, remote=False, autoremote=True)	Authenticate with the Cozify Cloud and Hub.
resetState()	Reset stored cloud state.
update_hubs()	Fetch fresh hub ip addresses & names and update our local state.
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.
refresh(force=False, expiry=datetime.timedelta(days=1))	Renew current cloud token and store new token in state.
_need_refresh(force, expiry)	Evaluate if refresh timer is already over or if forcing is valid.
_need_cloud_token(trust=True)	Validate current remote token and decide if we'll request it during authentication.
_need_hub_token(trust=True)	Validate current hub token and decide if we'll request it during authentication.
_getotp()	
_getEmail()	
_getAttr(attr)	Get cloud state attributes by attr name
_setAttr(attr, value, commit=True)	Set cloud state attributes by attr name
_isAttr(attr)	Check validity of attribute by attr name.
token(new_token=None)	Get currently used cloud_token or set a new one.
email(new_email=None)	Get currently used cloud account email or set a new one.

cozify.cloud.**authenticate**(*trustCloud=True*, *trustHub=True*, *remote=False*, *autoremote=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.
- **autoremote** (*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.**resetState**()

Reset stored cloud state.

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

cozify.cloud.**update_hubs**()

Fetch fresh hub ip addresses & names and update our local state.

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(days=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.**_need_refresh**(*force*, *expiry*)

Evaluate if refresh timer is already over or if forcing is valid.

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.

Returns
True if refresh should be done according to forcing and expiry.

Return type
bool

cozify.cloud.**_need_cloud_token**(*trust=True*)

Validate current remote token and decide if we'll request it during authentication.

Parameters
trust (*bool*) – Set to False to always decide to renew. Defaults to True.

Returns
True to indicate a need to request token.

Return type
bool

cozify.cloud.**_need_hub_token**(*trust=True*)

Validate current hub token and decide if we'll request it during authentication.

Parameters
trust (*bool*) – Set to False to always decide to renew. Defaults to True.

Returns
True to indicate a need to request token.

Return type
bool

cozify.cloud.**_getotp**()

cozify.cloud.**_getEmail**()

cozify.cloud.**_getAttr**(*attr*)

Get cloud state attributes by attr name

Parameters
attr (*str*) – Name of cloud state attribute to retrieve

Returns
Value of attribute or exception on failure

Return type
str

cozify.cloud.**_setAttr**(*attr*, *value*, *commit=True*)

Set cloud state attributes by attr name

Parameters

- **attr** (*str*) – Name of cloud state attribute to overwrite. Attribute will be created if it doesn't exist.
- **value** (*str*) – Value to store
- **commit** (*bool*) – True to commit state after set. Defaults to True.

cozify.cloud.**_isAttr**(*attr*)

Check validity of attribute by attr name.

Returns
True if attribute exists

Return type
bool

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

Get currently used cloud_token or set a new one.

Returns
Cloud remote authentication token.

Return type
str

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

1.2.3 cozify.cloud_api

Module for handling Cozify Cloud API 1:1 functions

cozify.cloud_api.**cloudBase**

API endpoint including version

Type
str

Module Contents

Functions

get(call, headers=None, base=cloudBase, no_headers=False, json_output=True, raw=False, **kwargs)	GET method for calling hub API.
post(call, headers=None, data=None, params=None, base=cloudBase, no_headers=False, raw=False, **kwargs)	PUT method for calling hub API. For rest of kwargs parameters see get()
put(call, headers=None, data=None, base=cloudBase, no_headers=False, raw=False, **kwargs)	PUT method for calling hub API. For rest of kwargs parameters see get()
requestlogin(email, **kwargs)	Raw Cloud API call, request OTP to be sent to account email address.
emaillogin(email, otp, **kwargs)	Raw Cloud API call, request cloud token with email address & OTP.
lan_ip(**kwargs)	1:1 implementation of hub/lan_ip
hubkeys(cloud_token, **kwargs)	1:1 implementation of user/hubkeys
refreshsession(cloud_token, **kwargs)	1:1 implementation of user/refreshsession
remote(apicall, headers, data=None)	1:1 implementation of 'hub/remote'
_call(call, method, headers, params=None, data=None, no_headers=False, json_output=True, raw=False, **kwargs)	Backend for get & post

Attributes

cloudBase

cozify.cloud_api.**cloudBase = https://cloud2.cozify.fi/ui/0.2**

cozify.cloud_api.**get**(*call*, *headers=None*, *base=cloudBase*, *no_headers=False*, *json_output=True*, *raw=False*, ***kwargs*)

GET method for calling hub API.

Parameters

- **call** (*str*) – API path to call after the base, needs to include leading /.
- **headers** (*dict*) – Header dictionary to pass along to the request.
- **base** (*str*) – Base path to call from API instead of global base. Defaults to cloudBase.
- **no_headers** (*bool*) – Allow calling without headers or data.
- **json_output** (*bool*) – Assume API will return json and decode it.

cozify.cloud_api.**post**(*call*, *headers=None*, *data=None*, *params=None*, *base=cloudBase*, *no_headers=False*, *raw=False*, ***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 /.
- **headers** (*dict*) – Header dictionary to pass along to the request.
- **data** (*dict*) – Payload dictionary to POST.
- **params** (*dict*) – Payload dictionary to include as GET parameters.
- **base** (*str*) – Base path to call from API instead of global base. Defaults to cloudBase.
- **no_headers** (*bool*) – Allow calling without headers or data.

cozify.cloud_api.**put**(*call*, *headers=None*, *data=None*, *base=cloudBase*, *no_headers=False*, *raw=False*, ***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 /.
- **headers** (*dict*) – Header dictionary to pass along to the request.
- **data** (*dict*) – Payload dictionary to PUT.
- **base** (*str*) – Base path to call from API instead of global base. Defaults to cloudBase.
- **no_headers** (*bool*) – Allow calling without headers or data.

cozify.cloud_api.**requestlogin**(*email*, ***kwargs*)

Raw Cloud API call, request OTP to be sent to account email address.

Parameters

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

cozify.cloud_api.**emaillogin**(*email*, *otp*, ***kwargs*)

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.**lan_ip**(***kwargs*)

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.**hubkeys**(*cloud_token*, ***kwargs*)

1:1 implementation of user/hubkeys

Parameters
cloud_token (*str*) –

Returns
Map of hub_id: hub_token pairs.

Return type
dict

cozify.cloud_api.**refreshsession**(*cloud_token*, ***kwargs*)

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**(*apicall*, *headers*, *data=None*)

1:1 implementation of 'hub/remote'

Parameters

- **apicall** (*str*) – Full API call that would normally go directly to hub, e.g. '/cc/1.6/hub/colors'
- **headers** (*dict*) – Headers to send with request. Must contain Authorization & X-Hub-Key data.
- **data** (*str*) – json string to use as payload, changes method to PUT.

Returns
Requests response object.

Return type
requests.response

cozify.cloud_api._call(*call*, *method*, *headers*, *params=None*, *data=None*, *no_headers=False*, *json_output=True*, *raw=False*, ***kwargs*)

Backend for get & post

Parameters

- **call** (*str*) – Full API path to call.
- **method** (*function*) – requests.get|put function to use for call.
- **headers** (*dict*) – Header dictionary to pass along to the request.
- **params** (*dict*) – Params dictionary to POST.
- **data** (*dict*) – Payload dictionary to PUT.
- **no_headers** (*bool*) – Allow calling without headers, data or args.
- **json_output** (*bool*) – Assume API will return json and decode it.
- **raw** (*bool*) – Do no decoding, return requests.response object.

1.2.4 cozify.config

Module for handling consistent state storage.

cozify.config.**state_file**

file path where state storage is kept. By default XDG conventions are used. (Most likely ~/.config/python-cozify/python-cozify.cfg)

Type
str

cozify.config.**state**

State object used for in-memory state. By default initialized with _initState.

Type
configparser.ConfigParser

Module Contents

Functions

_initXDG()	Initialize config path per XDG basedir-spec and resolve the final location of state file storage.
stateWrite(tmpstate=None)	Write current state to file storage.
setStatePath(filepath=_initXDG(), copy_current=False)	Set state storage path. Useful for example for testing without affecting your normal state. Call with no arguments to reset back to autoconfigured location.
dump_state()	Print out current state file to stdout. Long values are truncated since this is only for visualization.
_initState(state_file)	Initialize state on cold start. Any stored state is read in or a new basic state is initialized.

Attributes

state_file
state

cozify.config.**_initXDG**()

Initialize config path per XDG basedir-spec and resolve the final location of state file storage.

Returns
file path to state file as per XDG spec and current env.

Return type
str

cozify.config.**stateWrite**(*tmpstate=None*)

Write current state to file storage.

Parameters
tmpstate (*configparser.ConfigParser*) – State object to store instead of default state.

cozify.config.**setStatePath**(*filepath=_initXDG()*, *copy_current=False*)

Set state storage path. Useful for example for testing without affecting your normal state. Call with no arguments to reset back to autoconfigured location.

Parameters

- **filepath** (*str*) – file path to use as new storage location. Defaults to XDG defined path.
- **copy_current** (*bool*) – Instead of initializing target file, dump previous state into it.

cozify.config.**dump_state**()

Print out current state file to stdout. Long values are truncated since this is only for visualization.

cozify.config.**_initState**(*state_file*)

Initialize state on cold start. Any stored state is read in or a new basic state is initialized.

Parameters
state_file (*str*) – State storage filepath to attempt to read from.

Returns
State object.

Return type
configparser.ConfigParser

cozify.config.**state_file**

cozify.config.**state**

1.2.5 cozify.hub

Module for handling highlevel Cozify Hub operations.

Module Contents

Functions

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.
device(device_id, **kwargs)	Get up to date device data set as a dict.
await_state(device_id, state, timeout=10, **kwargs)	Wait for a device to reach a desired state
has_state(device_id, state, **kwargs)	Check if device state matches the provided state keys. Keys not provided are ignored.
device_reachable(device_id, **kwargs)	Check if device is reachable.
device_exists(device_id, devs=None, state=None, **kwargs)	Check if device exists.
device_eligible(device_id, capability_filter, devs=None, state=None, **kwargs)	Check if device matches a AND devices filter.
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.
device_state_replace(device_id, state, **kwargs)	Replace the entire state of a device with the provided state. Useful for example for returning to a stored state.
device_on(device_id, **kwargs)	Turn on a device that is capable of turning on. Eligibility is determined by the capability ON_OFF.
device_off(device_id, **kwargs)	Turn off a device that is capable of turning off. Eligibility is determined by the capability ON_OFF.
light_temperature(device_id, temperature=2700, transition=0, **kwargs)	Set temperature of a light.
light_color(device_id, hue, saturation=1.0, transition=0, **kwargs)	Set color (hue & saturation) of a light.
light_brightness(device_id, brightness, transition=0, **kwargs)	Set brightness of a light.
scenes(filters=None, **kwargs)	Get full scene data set as a dict. Optionally filters scenes by on/off status.
scene(scene_id, **kwargs)	Get scene data set as a dict.
scene_toggle(scene_id, **kwargs)	Toggle on/off state of given scene.
scene_on(scene_id, **kwargs)	Turn on a scene.
scene_off(scene_id, **kwargs)	Turn off a scene.

continues on next page

Table 1 – continued from previous page

remote(hub_id, new_state=None)	Get remote status of matching hub_id or set a new value for it. Always returns current state at the end.
autoremote(hub_id, new_state=None)	Get autoremote status of matching hub_id or set a new value for it. Always returns current state at the end.
tz(**kwargs)	Get timezone of given hub or default hub if no id is specified. For more optional kwargs see cozify.hub_api.get()
ping(autorefresh=True, **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()
name(hub_id)	Get hub name by it's id.
host(hub_id)	Get hostname of matching hub_id
token(hub_id, new_token=None)	Get hub_token of matching hub_id or set a new value for it.
hub_id(hub_name)	Get hub id by it's name.
exists(hub_id)	Check for existance of hub in local state.
default()	Return id of default Hub.
_getAttr(hub_id, attr, default=None, boolean=False)	Get hub state attributes by attr name. Optionally set a default value if attribute not found.
_setAttr(hub_id, attr, value, commit=True)	Set hub state attributes by hub_id and attr name
_get_id(**kwargs)	Get a hub_id from various sources, meant so that you can just throw kwargs at it and get a valid id.
_fill_kwargs(kwargs)	Check that common items are present in kwargs and fill them if not.
_clean_state(state)	Return purged state of values so only wanted values can be modified.
_in_range(value, low, high, description='undefined')	Check that the value is in the given range, raise an error if not.
getDevices(**kwargs)	Deprecated, will be removed in v0.3. Get up to date full devices data set as a dict.
getDefaultHub()	Deprecated, use default(). Return id of default Hub.
getHubId(hub_name)	Deprecated, use hub_id(). Return id of hub by it's name.

Attributes

capability

cozify.hub.**capability**

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.**device**(*device_id*, ***kwargs*)

Get up to date device data set as a dict.

Parameters

- **device_id** (*str*) – ID of the device to retrieve.
- ****hub_name** (*str*) – optional name of hub to query. Will get converted to hub_id 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.

Returns
full live device data as returned by the API

Return type
dict

cozify.hub.**await_state**(*device_id*, *state*, *timeout=10*, ***kwargs*)

Wait for a device to reach a desired state

Parameters

- **device_id** (*str*) – ID of the device to check.
- **state** (*dict*) – Full state dictionary to expect. Timestamp fields are ignored.
- **timeout** (*int*) – Timeout of wait time.

Returns
True when state matches or False on timeout.

Return type
bool

cozify.hub.**has_state**(*device_id*, *state*, ***kwargs*)

Check if device state matches the provided state keys. Keys not provided are ignored.

Parameters

- **device_id** (*str*) – ID of the device to check.
- **state** (*dict*) – State dictionary to expect. Only provide keys you care about.

Returns
If given state values match.

Return type
bool

cozify.hub.**device_reachable**(*device_id*, ***kwargs*)

Check if device is reachable.

Parameters
device_id (*str*) – ID of the device to check.

Returns
Reachability as defined by the API.

Return type
bool

cozify.hub.**device_exists**(*device_id*, *devs=None*, *state=None*, ***kwargs*)

Check if device exists.

Parameters

- **device_id** (*str*) – ID of the device to check.
- **devs** (*dict*) – Optional devices dictionary to use. If not defined, will be retrieved live.
- **state** (*dict*) – Optional state dictionary, will be updated with state of checked device if device is eligible. Previous data in the dict is preserved unless it's overwritten by new values.

Returns
True if filter matches.

Return type
bool

cozify.hub.**device_eligible**(*device_id*, *capability_filter*, *devs=None*, *state=None*, ***kwargs*)

Check if device matches a AND devices filter.

Parameters

- **device_id** (*str*) – ID of the device to check.
- **capability_filter** (*hub.capability*) – Single hub.capability or a list of them to match against.
- **devs** (*dict*) – Optional devices dictionary to use. If not defined, will be retrieved live.
- **state** (*dict*) – Optional state dictionary, will be updated with state of checked device if device is eligible. Previous data in the dict is preserved unless it's overwritten by new values.

Returns
True if filter matches.

Return type
bool

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.**device_state_replace**(*device_id*, *state*, ***kwargs*)

Replace the entire state of a device with the provided state. Useful for example for returning to a stored state.

Parameters

- **device_id** (*str*) – ID of the device to toggle.
- **state** (*dict*) – State dictionary to push out.
- ****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.**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_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.**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.**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 a ValueError is raised.
- **saturation** (*float*) – Saturation in the range of [0, 1]. If outside the range a ValueError is raised. Defaults to 1.0 (full saturation.)
- **transition** (*int*) – Transition length in milliseconds. Defaults to instant.

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 a ValueError is raised.

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

cozify.hub.**scenes**(*filters=None*, ***kwargs*)

Get full scene data set as a dict. Optionally filters scenes by on/off status.

Parameters

- **filters** (*dict*) – Filter scenes by their values by defining key value pairs as a dict. Defaults to all scenes.
- ****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
scene data as returned by the API

Return type
dict

cozify.hub.**scene**(*scene_id*, ***kwargs*)

Get scene data set as a dict.

Parameters

- **scene_id** (*str*) – ID of the scene to retrieve.
- ****hub_name** (*str*) – optional name of hub to query. Will get converted to hub_id 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.

Returns
scene data as returned by the API

Return type
dict

cozify.hub.**scene_toggle**(*scene_id*, ***kwargs*)

Toggle on/off state of given scene.

Parameters

- **scene_id** (*str*) – ID of the scene 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.**scene_on**(*scene_id*, ***kwargs*)

Turn on a scene.

Parameters
scene_id (*str*) – ID of the scene to operate on.

cozify.hub.**scene_off**(*scene_id*, ***kwargs*)

Turn off a scene.

Parameters
scene_id (*str*) – ID of the scene to operate on.

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

Get remote status of matching hub_id or set a new value for it. Always returns current state at the end.

Parameters

- **hub_id** (*str*) – Id of hub to query. The id is a string of hexadecimal sections used internally to represent a hub.
- **new_state** (*bool*) – New remoteness state to set for hub. True means remote. Defaults to None when only the current value will be returned.

Returns
True for a hub considered remote.

Return type
bool

cozify.hub.**autoremote**(*hub_id*, *new_state=None*)

Get autoremote status of matching hub_id or set a new value for it. Always returns current state at the end.

Parameters

- **hub_id** (*str*) – Id of hub to query. The id is a string of hexadecimal sections used internally to represent a hub.
- **new_state** (*bool*) – New autoremoteness state to set for hub. True means remote will be automanaged. Defaults to None when only the current value will be returned.

Returns
True for a hub with autoremote enabled.

Return type
bool

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()

Parameters
****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

cozify.hub.**ping**(*autorefresh=True*, ***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

- **autorefresh** (*bool*) – Wether to perform a autorefresh after an initially failed ping. If successful, will still return True. Defaults to True.
- ****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.**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.**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.**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.**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.**exists**(*hub_id*)

Check for existance of hub in local state.

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

cozify.hub.**default**()

Return id of default Hub.

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

cozify.hub.**_getAttr**(*hub_id*, *attr*, *default=None*, *boolean=False*)

Get hub state attributes by attr name. Optionally set a default value if attribute not found.

Parameters

- **hub_id** (*str*) – Id of hub to query. The id is a string of hexadecimal sections used internally to represent a hub.
- **attr** (*str*) – Name of hub attribute to retrieve
- **default** – Optional default value to set for unset attributes. If no default is provided these raise an AttributeError.
- **boolean** – Retrieve and return value as a boolean instead of string. Defaults to False.

Returns
Value of attribute or exception on failure.

Return type
str

cozify.hub.**_setAttr**(*hub_id*, *attr*, *value*, *commit=True*)

Set hub state attributes by hub_id and attr name

Parameters

- **hub_id** (*str*) – Id of hub to store for. The id is a string of hexadecimal sections used internally to represent a hub.
- **attr** (*str*) – Name of cloud state attribute to overwrite. Attribute will be created if it doesn't exist.
- **value** (*str*) – Value to store
- **commit** (*bool*) – True to commit state after set. Defaults to True.

cozify.hub.**_get_id**(***kwargs*)

Get a hub_id from various sources, meant so that you can just throw kwargs at it and get a valid id. If no data is available to determine which hub was meant, will default to the default hub. If even that fails, will raise an AttributeError.

Parameters

- ****hub_id** (*str*) – Will be returned as-is if defined.
- ****hub_name** (*str*) – Name of hub.
- **hubName** (*str*) – Deprecated. Compatibility keyword for hub_name, to be removed in v0.3
- **hubId** (*str*) – Deprecated. Compatibility keyword for hub_id, to be removed in v0.3

cozify.hub.**_fill_kwargs**(*kwargs*)

Check that common items are present in kwargs and fill them if not.

Args: kwargs(dict): kwargs dictionary to fill. Operated on directly.

cozify.hub.**_clean_state**(*state*)

Return purged state of values so only wanted values can be modified.

Args: state(dict): device state dictionary. Original won't be modified.

cozify.hub.**_in_range**(*value*, *low*, *high*, *description='undefined'*)

Check that the value is in the given range, raise an error if not. None is always considered a valid value.

Returns
True if value in range. Otherwise a ValueError is raised.

Return type
bool

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.**getDefaultHub**()

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

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

1.2.6 cozify.hub_api

Module for all Cozify Hub API 1:1 calls

cozify.hub_api.**apiPath**

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.

Type
str

Module Contents

Functions

_getBase(host, port=8893, **kwargs)	
get(call, hub_token_header=True, base=apiPath, **kwargs)	GET method for calling hub API.
put(call, data, hub_token_header=True, base=apiPath, **kwargs)	PUT method for calling hub API. For rest of kwargs parameters see get()
_call(call, method, hub_token_header, data=None, **kwargs)	Backend for get & put
hub(**kwargs)	1:1 implementation of /hub API call. For kwargs see cozify.hub_api.get()
tz(**kwargs)	1:1 implementation of /hub/tz API call. For kwargs see cozify.hub_api.get()
devices(**kwargs)	1:1 implementation of /devices API call. For remaining kwargs see cozify.hub_api.get()
devices_command(command, **kwargs)	1:1 implementation of /devices/command. For kwargs see cozify.hub_api.put()
devices_command_generic(device_id, command=None, request_type, **kwargs)	Command helper for CMD type of actions.
devices_command_state(device_id, state, **kwargs)	Command helper for CMD type of actions.
devices_command_on(device_id, **kwargs)	Command helper for CMD_DEVICE_ON.
devices_command_off(device_id, **kwargs)	Command helper for CMD_DEVICE_OFF.
scenes(**kwargs)	Implementation of /scenes API call. For kwargs see cozify.hub_api.get()
scenes_command_state(scene_id, request_type, **kwargs)	Commands changing a scene's state. For kwargs see cozify.hub_api.put()
scenes_command_off(scene_id, **kwargs)	Command helper for CMD_SCENE_OFF.
scenes_command_on(scene_id, **kwargs)	Command helper for CMD_SCENE_ON.

Attributes

apiPath

cozify.hub_api.**apiPath = /cc/1.14**

cozify.hub_api.**_getBase**(*host*, *port=8893*, ***kwargs*)

cozify.hub_api.**get**(*call*, *hub_token_header=True*, *base=apiPath*, ***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.**put**(*call*, *data*, *hub_token_header=True*, *base=apiPath*, ***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 /.
- **data** (*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.**_call**(*call*, *method*, *hub_token_header*, *data=None*, ***kwargs*)

Backend for get & put

Parameters

- **call** (*str*) – Full API path to call.
- **method** (*function*) – requests.get|put function to use for call.

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.**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

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 APIEerror 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 wether 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_state**(*device_id*, *state*, ***kwargs*)

Command helper for CMD type of actions. No checks are made wether 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.**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_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.**scenes**(***kwargs*)

Implementation of /scenes API call. For kwargs see cozify.hub_api.get()

Returns
Full scene state as returned by the API

Return type
dict

cozify.hub_api.**scenes_command_state**(*scene_id*, *request_type*, ***kwargs*)

Commands changing a scene's state. For kwargs see cozify.hub_api.put()

Parameters

- **scene_id** (*str*) – ID of the scene to operate on.
- **request_type** (*str*) – the request type, i.e. CMD_SCENE_ON or CMD_SCENE_OFF.

Returns
Whatever the API replied or raises an APIError on failure.

Return type
str

cozify.hub_api.**scenes_command_off**(*scene_id*, ***kwargs*)

Command helper for CMD_SCENE_OFF.

Parameters
scene_id (*str*) – ID of the scene to operate on.

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

Return type
str

cozify.hub_api.**scenes_command_on**(*scene_id*, ***kwargs*)

Command helper for CMD_SCENE_ON.

Parameters
scene_id (*str*) – ID of the scene to operate on.

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

Return type
str

1.2.7 cozify.multisensor

Module Contents

Functions

getMultisensorData(data)	Deprecated, will be removed in v0.3

cozify.multisensor.**getMultisensorData**(*data*)
Deprecated, will be removed in v0.3

1.3 Package Contents

cozify.**__version__ = 0.2.34**

CHAPTER

TWO

PYTHON-COZIFY

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.

2.1 Installation

Python3.8 is the current minimum supported version of Python. For example Ubuntu 20.04 LTS is still supported out of the box, older versions may need a manual Python upgrade or PPA.

The recommended way is to install from PyPi:

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

To benefit from new features you'll need to update the library (pip does not auto-update):

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

or if developing, clone the main branch of this repo (main stays at current release) and:

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

2.2 Basic usage

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

2.2.1 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']))
```

2.2.2 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
```

2.2.3 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
```

2.3 On Capabilities

The most practical way to "find" devices for operating on is currently to filter the devices list by their capabilties. 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)
```

2.4 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))
```

2.5 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 overriden in most if not all funcions by the boolean keyword arguments 'remote' and 'autoremote'. They can also be queried or permanently changed by the hub.remote() and hub.autoremote() functions.

2.6 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.

2.7 Enconding 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
```

2.8 Sample projects

- 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

2.9 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.

2.9.1 Tests

pytest is used for unit tests.

- 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, i.e. they will not change device state.
- Some tests are marked as "destructive" and will cause changes such as a light being turned on or tokens getting invalidated on purpose. Every effort is made to return devices to their original state and to perform saved state manipulation on copies of the live state.
- A few tests are marked as "remote" and are only expected to succeed when testing remotely, i.e. outside the LAN of the hub.
- A few tests are marked as "mbtest" and will only work if a MonteBank server is available. If a non-local instance is desired, provide a .env file with MBTEST_HOST set. The easiest way to get a local instance is to use Docker: *docker run –rm –network host bbyars/mountebank:latest mb start*
- Most tests are marked as "logic" and do not require anything external. If no set is defined, only logic tests are run.

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

```
pytest # or: poetry run pytest
# or run only live tests:
pytest -m live
```

(continues on next page)

(continued from previous page)

```
# run everything except destructive * MonteBank tests:
pytest -m "not destructive and not mbtest"
```

To run the test suite on an already installed python-cozify (defining a set is mandatory, otherwise ALL sets are run including destructive):

```
pytest -v -m logic --pyargs cozify
```

2.9.2 Roadmap, aka. Current Limitations

- Authentication flow has been improved quite a bit but it would benefit a lot from real-world feedback.
- Read call coverage is decent and support for most light interaction is done. If there's something specific you want to use sooner than later file an issue so it can get prioritized!
- Device model is not object oriented yet and instead relies on capability filtering and device id's.

PYTHON MODULE INDEX

C

INDEX

Symbols

A

B

C

D

E

G

H

L

M

N

O

P

R

S

T

U

V