Package ‘PhilipsHue’

October 12, 2022

Title R Interface to the Philips Hue API

Version 1.0.0

Description Control Philips Hue smart lighting. Use this package to connect to a Hue bridge on your local network (remote authentication not yet supported) and control your smart lights through the Philips Hue API. All API V1 endpoints are supported. See API documentation at <https://developers.meethue.com/>.

License GPL (>= 3)

URL https://fascinatingfingers.gitlab.io/philipshue, https://gitlab.com/fascinatingfingers/philipshue

BugReports https://gitlab.com/fascinatingfingers/philipshue/-/issues

Depends R (>= 4.1)

Imports httr, methods, pkgload, purrr, utils, yaml

Suggests covr, DT, fs, knitr, lubridate, mockery, rmarkdown, spelling, testthat, uuid, withr

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

RoxygenNote 7.1.2

NeedsCompilation no

Author Justin Brantley [aut, cre]

Maintainer Justin Brantley <fascinatingfingers@icloud.com>

Repository CRAN

Date/Publication 2022-05-06 11:00:05 UTC

R topics documented:

`auth_helpers` *Authentication helpers*

Description

These functions help manage the environment variables that the PhilipsHue package uses to store authentication secrets.

Usage

```
has_local_auth()

write_auth(path = ".Renviron", append = TRUE)

reset_auth()
```

Arguments

`path`	file path to write secrets to
`append`	passed to `write()`

Details

Local authentication requires setting two environment variables: `PHILIPS_HUE_BRIDGE_IP` and `PHILIPS_HUE_BRIDGE_USERNAME`. `has_local_auth()` uses regular expressions to check if these variables are set (but does not check if the credentials actually work). `reset_auth()` sets these variables to empty strings, and `write_auth()` writes the current values to a file (e.g. an `.Renviron` file for use during development).

Value

`has_local_auth()` returns a logical value; `write_auth()` and `reset_auth()` return `TRUE` invisibly upon success.

`auth_local` *Authentication – local*

Description

This function helps check and set the necessary environment variables to authenticate to a Hue bridge on the local network.

Usage

```
auth_local(
  ip = Sys.getenv("PHILIPS_HUE_BRIDGE_IP"),
  username = Sys.getenv("PHILIPS_HUE_BRIDGE_USERNAME")
)
```

Arguments

`ip`	the IP address of your Hue bridge
`username`	the username with access to your Hue bridge

Value

Returns `TRUE` (invisibly) if options were successfully set

See Also

https://developers.meethue.com/develop/get-started-2/

`capabilities` *Hue API:* `capabilities` *endpoints*

Description

Hue API: `capabilities` endpoints

Usage

```
get_capabilities()
```

Value

get_capabilities returns a list structure with the capabilities of bridge resources.

See Also

https://developers.meethue.com/develop/hue-api/10-capabilities-api/

---

configs *Hue API:* `configuration` *endpoints*

Description

Hue API: `configuration` endpoints

Usage

```
get_config()

set_config_attributes(...)

get_state()
```

Arguments

`...`	named parameters describing configuration

Value

Requests that create resources return the ID of the newly created item, requests with side effects return `TRUE` upon success, and GET requests return the response content, parsed into a list.

See Also

https://developers.meethue.com/develop/hue-api/7-configuration-api/

---

configure_daylight_sensor *Configure Built-In Daylight Sensor*

Description

Supported sensors for the Hue bridge include a virtual daylight sensor that calculates sunrise and sunset times based on your location. This function helps configure the built-in daylight sensor (`id = 1`).

Usage

```
configure_daylight_sensor(
  lat,
  lon,
  sunriseoffset = 30,
  sunsetoffset = -30,
  id = 1
)
```

Arguments

`lat`	latitude (in decimal degrees). Positive north; negative south.
`lon`	longitude (in decimal degrees). Positive east; negative west.
`sunriseoffset`	"daylight" begins `sunriseoffset` minutes after sunrise
`sunsetoffset`	"daylight" ends `sunsetoffset` minutes after sunset
`id`	ID of the daylight sensor

Value

Returns `TRUE` (invisibly) upon success.

See Also

https://developers.meethue.com/develop/hue-api/supported-devices/#supportred-sensors

---

`create_user` *Create local user*

Description

The `create_user()` function allows you to create a user on a local Hue network, but it requires pressing the button on the Hue bridge then executing this command within 30 seconds. `delete_user()` is not implemented because, apparently, it can only be executed through remote authentication.

Usage

```
create_user(devicetype)
```

Arguments

`devicetype`	a string naming your app and the device it's running on; suggested format: `<application_name>#<devicename>` (e.g. `Hue#iPhone13`).

Value

Returns a list with the newly created `username` and `clientkey`

See Also

https://developers.meethue.com/develop/hue-api/7-configuration-api/#create-user

---

`groups` *Hue API:* `groups` *endpoints*

Description

Hue API: `groups` endpoints

Usage

```
create_group(...)

get_groups()

get_group(id)

set_group_attributes(id, ...)

set_group_state(id, ...)

delete_group(id)
```

Arguments

`...`	named parameters describing group attributes or state (e.g. `name = "foo"`; `on = TRUE`)
`id`	ID of a specific group

Value

Requests that create resources return the ID of the newly created item, requests with side effects return `TRUE` upon success, and GET requests return the response content, parsed into a list.

See Also

https://developers.meethue.com/develop/hue-api/groupds-api/

lights *Hue API:* `lights` *endpoints*

Description

Hue API: `lights` endpoints

Usage

```
search_for_new_lights()

get_new_lights()

rename_light(id, name)

get_lights()

get_light(id)

set_light_state(id, ...)

delete_light(id)
```

Arguments

`id`	ID of a specific light
`name`	name to assign to the light
`...`	named parameters describing light state (e.g. `on = TRUE`)

Value

Requests that create resources return the ID of the newly created item, requests with side effects return `TRUE` upon success, and GET requests return the response content, parsed into a list.

See Also

https://developers.meethue.com/develop/hue-api/lights-api/

resourcelinks *Hue API:* resourcelinks *endpoints*

Description

Hue API: resourcelinks endpoints

Usage

```
create_resourcelink(...)

get_resourcelinks()

get_resourcelink(id)

set_resourcelink_attributes(id, ...)

delete_resourcelink(id)
```

Arguments

...	named parameters describing resourcelink attributes (e.g. name = "foo")
id	ID of a specific resourcelink

Value

Requests that create resources return the ID of the newly created item, requests with side effects return TRUE upon success, and GET requests return the response content, parsed into a list.

See Also

https://developers.meethue.com/develop/hue-api/9-resourcelinks-api/

rules *Hue API:* rules *endpoints*

Description

Hue API: rules endpoints

Usage

```
create_rule(name, conditions, actions)

get_rules()

get_rule(id)

set_rule_attributes(id, name, conditions, actions)

delete_rule(id)
```

Arguments

`name`	name to assign to the rule
`conditions`	a list of conditions (e.g. the result of a call to `condition()`)
`actions`	a list of actions (e.g. the result of a call to `action()`)
`id`	ID of a specific rule

Value

Requests that create resources return the ID of the newly created item, requests with side effects return `TRUE` upon success, and GET requests return the response content, parsed into a list.

See Also

`https://developers.meethue.com/develop/hue-api/6-rules-api/`

`rule_helpers` *Rule Helpers*

Description

Defining rules can become quite verbose, and it can be tricky to prepare the proper list structure for the POST or PUT request. These functions simplify things a bit and provide a leaner, more semantic interface.

Usage

```
condition(address, operator, value)

action(address, method, ...)
```

Arguments

`address`	path to attribute or resource
`operator`	one of: eq, gt, lt, dx, ddx, stable, not stable, in, not in
`value`	the value a condition will compare against
`method`	the HTTP method used to send the body to the given address
`...`	named parameters to include in action body

Value

Returns a list-like structure suitable for `create_rule()` or `set_rule_attributes()`.

scenes — *Hue API:* `scenes` *endpoints*

Description

Hue API: `scenes` endpoints

Usage

```
create_scene(name, lights, recycle = TRUE, transitiontime = 4)

create_group_scene(name, group_id, recycle = TRUE, transitiontime = 4)

get_scenes()

get_scene(id)

set_scene_attributes(id, ...)

set_scene_lightstate(scene_id, light_id, ...)

delete_scene(id)
```

Arguments

`name`	name to assign to the scene
`lights`	vector of light IDs included in the scene
`recycle`	logical indicating whether the scene can be automatically deleted by the bridge
`transitiontime`	duration (in milliseconds) of the scene transition
`group_id`	ID of group that scene belongs to
`id, scene_id`	ID of a specific scene
`...`	named parameters describing scene attributes or light state (e.g. `name = "foo"`; `on = TRUE`)
`light_id`	ID of a specific light in the scene

Value

Requests that create resources return the ID of the newly created item, requests with side effects return `TRUE` upon success, and GET requests return the response content, parsed into a list.

See Also

https://developers.meethue.com/develop/hue-api/4-scenes/

`schedules` *Hue API:* `schedules` *endpoints*

Description

Hue API: `schedules` endpoints

Usage

```
create_schedule(...)

get_schedules()

get_schedule(id)

set_schedule_attributes(id, ...)

delete_schedule(id)
```

Arguments

`...`	named parameters describing schedule attributes (e.g. `name = "foo"`)
`id`	ID of a specific schedule

Value

Requests that create resources return the ID of the newly created item, requests with side effects return `TRUE` upon success, and GET requests return the response content, parsed into a list.

See Also

https://developers.meethue.com/develop/hue-api/3-schedules-api/

sensors — *Hue API:* sensors *endpoints*

Description

Hue API: sensors endpoints

Usage

```
create_sensor(...)

search_for_new_sensors()

get_new_sensors()

rename_sensor(id, name)

get_sensors()

get_sensor(id)

set_sensor_config(id, ...)

set_sensor_state(id, ...)

delete_sensor(id)
```

Arguments

`...`	named parameters describing sensor state (e.g. `on = TRUE`)
`id`	ID of a specific sensor
`name`	name to assign to the sensor

Value

Requests that create resources return the ID of the newly created item, requests with side effects return `TRUE` upon success, and GET requests return the response content, parsed into a list.

See Also

https://developers.meethue.com/develop/hue-api/5-sensors-api/

Index