Package ‘lifx’

June 24, 2020

Type Package
Title Control ‘LIFX’ Smart Light Bulbs
Version 0.2.0
Author Martin Barner <m@martinbarner.de>
Maintainer Martin Barner <m@martinbarner.de>
Description Allows you to read and change the state of ‘LIFX’ smart light bulbs via the ‘LIFX’ developer api <https://api.developer.lifx.com/>.
Covers most ‘LIFX’ api endpoints, including changing light color and brightness, selecting lights by id, group or location as well as activating effects.
License GPL-3
Encoding UTF-8
LazyData true
Imports curl, httr, assertthat, jsonlite, crayon, utils
RoxygenNote 7.1.0
Suggests covr, knitr, rmarkdown, testthat
NeedsCompilation no
Repository CRAN
Date/Publication 2020-06-24 12:10:06 UTC

R topics documented:

check_lifx_response .............. 2
lifx .................................. 3
lx_check_color .................. 3
lx_color ........................... 4
lx_color_name .................. 5
lx_delta ......................... 6
lx_effect_breathe .............. 7
lx_effect_flame ............... 9
lx_effect_morph ............. 10
lx_effect_move ................ 11
check_lifx_response

Description
react to 'LIFX' api response error codes

Usage
check_lifx_response(response)

Arguments
response the api response received from http::PUT / POST / GET

Value
depending on the status either: an error; a warning and the response as is; the response as is without any message.

References
error messages copied from https://api.developer.lifx.com/docs/errors
### Description

The `LIFX` R package is an interface to the [LIFX smart bulb api](https://api.developer.lifx.com/docs).

#### most important `LIFX` functions

- `lx_save_token`
- `lx_list_lights`
- `lx_color`
- `lx_effect_breathe` and other effects

### Usage

```r
lx_check_color(color_name, token = lx_get_token())
```

### Arguments

- **color_name**: a color string in `LIFX` api format (can be made with `lx_color_name`)
- **token**: API token (see `lx_save_token`). If left empty, the token is retrieved from the environmental variable if available. (see `lx_save_token`)

### Details

calls the API endpoint https://api.lifx.com/v1/color to check if the color is valid.

### Value

logical `TRUE` if the color name is valid; `FALSE` if not; throws an error if the API could not be reached or another issue occurred.
Examples

```r
## Not run:
lx_check_color("INVALID_COLOR_NAME") # invalid 'LIFX' color string returns FALSE
lx_check_color("#FFFFFF") # valid 'LIFX' color string returns TRUE
lx_check_color("orange") # valid 'LIFX' color string returns TRUE
lx_check_color('brightness:1 hue:50') # valid 'LIFX' color string returns TRUE

## End(Not run)
```

### lx_color

`lx_color` change the state of 'LIFX' lamps

#### Description

change the state of 'LIFX' lamps

#### Usage

```r
lx_color(
  hue = NULL,
  saturation = NULL,
  brightness = NULL,
  kelvin = NULL,
  duration = NULL,
  infrared = NULL,
  color_name = NULL,
  fast = FALSE,
  delta = FALSE,
  selector = "all",
  power = NULL,
  token = lx_get_token()
)
```

#### Arguments

- **hue**: set the hue (0-255)
- **saturation**: set the saturation (0-1)
- **brightness**: set the brightness (0-1)
- **kelvin**: set the color temperature. limits depend on the specific lamp; limits are likely in the range of 2500-9000
- **duration**: in seconds, how long to perform the transition
- **infrared**: infrared brightness (0-1)
- **color_name**: a color name (i.e. "red"), hexadecimal color code (i.e. "#FF0000") or output from `lx_color()` (in 'LIFX' api format (see https://api.developer.lifx.com/docs/colors). If this parameter is used, other parameters may be ignored.
Fast

Executes the query fast, without initial state checks and wait for no results. See https://api.developer.lifx.com/docs/set-state

Delta

If set to TRUE, color values (hue, saturation, brightness, kelvin, infrared) are added to the lights’ current values. Can not be used in combination with ‘color_name’

Selector

’LIFX’ api "selector” such as "all", "id:12345", or "location:kitchen". Can be created with lx_selector or written manually (see https://api.developer.lifx.com/docs/selectors

Power

String - if set to "on", turns the light on, if set to "off" turns it off.

Token

API token (see ?lx_save_token). If left empty, the token is retrieved from the environmental variable if available. (see lx_save_token)

Value

An ’httr’ response object (see response)

Examples

```r
## Not run:
lx_color(hue = 200)
lx_color(saturation = 0.8)
lx_color(hue = 200, saturation = 0.5, brightness = 0.5)
lx_color(color_name = 'cyan', brightness = 1)
lx_color(kelvin = 5000, fast = TRUE)
lx_color(brightness = -0.3, delta = TRUE)

## End(Not run)
```

---

**lx_color_name**

Picking a color by name or hsbk

**Description**

Picking a color by name or hsbk

**Usage**

```r
lx_color_name(
  hue = NULL,
  saturation = NULL,
  brightness = NULL,
  kelvin = NULL,
  color_name = NULL,
  check = TRUE,
  token = lx_get_token()
)
```
Arguments

- **hue**: set the hue (0-255)
- **saturation**: set the saturation (0-1)
- **brightness**: set the brightness (0-1)
- **kelvin**: set the color temperature. Limits depend on the specific lamp; limits are likely in the range of 2500-9000
- **color_name**: a color name (i.e., "red"), hexadecimal color code (i.e., "#FF0000") or output from `lx_color()` (in 'LIFX' API format (see https://api.developer.lifx.com/docs/colors)). If this parameter is used, other parameters may be ignored.
- **check**: if FALSE does not call the API to check if the color is valid
- **token**: API token (see ?lx_save_token). If left empty, the token is retrieved from the environmental variable if available. (see `lx_save_token`)

Value

A character string specifying a light color as expected by the 'LIFX' API. Outputs from this function have their own class and printing style, but a pure character string can be used just as well.

Examples

```r
## Not run:
strong_red <- lx_color_name(hue = 0, saturation = 1, brightness = 1)
lx_color(color_name = strong_red)

dim_green <- lx_color_name(color_name = "#00FF00", saturation = 1, brightness = 0.1)
lx_color(color_name = dim_green)

unsaturated_cyan <- lx_color_name(color_name = "cyan", saturation = 0.3)
lx_color(color_name = unsaturated_cyan)
## End(Not run)
```

---

**lx_delta**

*Change light state relative to current state (wrapper for POST state delta)*

Description

Change light state relative to current state (wrapper for POST state delta)
**lx_effect_breathe**

**Usage**

```r
lx_delta(
  hue = NULL,
  saturation = NULL,
  brightness = NULL,
  kelvin = NULL,
  infrared = NULL,
  duration = 0,
  power = NULL,
  selector = "all",
  token = lx_get_token()
)
```

**Arguments**

- **hue**
  set the hue (0-255)
- **saturation**
  set the saturation (0-1)
- **brightness**
  set the brightness (0-1)
- **kelvin**
  set the color temperature. limits depend on the specific lamp; limits are likely in the range of 2500-9000
- **infrared**
  infrared brightness (0-1)
- **duration**
  in seconds, how long to perform the transition
- **power**
  string - if set to "on", turns the light on, if set to "off" turns it off.
- **selector**
  'LIFX' api "selector" such as "all", "id:12345", or "location:kitchen". Can be created with `lx_selector` or written manually (see [https://api.developer.lifx.com/docs/selectors](https://api.developer.lifx.com/docs/selectors))
- **token**
  API token (see `lx_save_token`). If left empty, the token is retrieved from the environmental variable if available. (see `lx_save_token`)

**Value**

an 'httr' response object (see `response`)

**References**

[https://api.developer.lifx.com/docs/state-delta](https://api.developer.lifx.com/docs/state-delta)

---

**lx_effect_breathe**

"Breathe" effect

**Description**

"Breathe" effect
Usage

```r
lx_effect_breathe(
  color,
  from_color = NULL,
  period = 1,
  cycles = 1,
  persist = FALSE,
  power_on = TRUE,
  peak = 0.5,
  selector = "all",
  token = lx_get_token()
)
```

Arguments

- **color**: The color to use for the breathe effect. Use `lx_color()` as input.
- **from_color**: The color to start the effect from. If this parameter is omitted then the color the bulb is currently set to is used instead.
- **period**: The time in seconds for one cycle of the effect.
- **cycles**: The number of times to repeat the effect.
- **persist**: Boolean; If FALSE set the light back to its previous value when effect ends, if true leave the last effect color.
- **power_on**: If FALSE, does not turn light on if it is off.
- **peak**: Defines where in a period the target color is at its maximum. Minimum 0.0, maximum 1.0.
- **selector**: 'LIFX' API "selector" such as "all", "id:12345", or "location:kitchen". Can be created with `lx_selector` or written manually (see [https://api.developer.lifx.com/docs/selectors](https://api.developer.lifx.com/docs/selectors)).
- **token**: API token (see `lx_save_token`). If left empty, the token is retrieved from the environmental variable if available. (see `lx_save_token`)

Value

- an 'httr' response object (see `response`)

Examples

```r
## Not run:
lx_effect_breathe(color = "red",from_color = "blue",period = 3,cycles = 5,power_on = TRUE)
```
Description

"Flame" effect

Usage

```r
lx_effect_flame(
    period = 5,
    duration = 10^10,
    power_on = TRUE,
    fast = FALSE,
    selector = "all",
    token = lx_get_token()
)
```

Arguments

- **period**: This controls how quickly the flame runs. It is measured in seconds. A lower number means the animation is faster.
- **duration**: How long the animation lasts for in seconds. Not specifying a duration makes the animation never stop. Specifying 0 makes the animation stop. Note that there is a known bug where the tile remains in the animation once it has completed if duration is nonzero.
- **power_on**: if TRUE (default), switch any selected device that is off to on before performing the effect.
- **fast**: Executes the query fast, without initial state checks and wait for no results. See [https://api.developer.lifx.com/docs/set-state](https://api.developer.lifx.com/docs/set-state)
- **selector**: 'LIFX' api "selector" such as "all", "id:12345", or "location:kitchen". Can be created with `lx_selector` or written manually (see [https://api.developer.lifx.com/docs/selectors](https://api.developer.lifx.com/docs/selectors)
- **token**: API token (see `lx_save_token`). If left empty, the token is retrieved from the environmental variable if available. (see `lx_save_token`)

Value

an 'httr' response object (see `response`)

Examples

```r
## Not run:
lx_effect_flame(period = 2, duration = 3)
## End(Not run)
```
lx_effect_morph  "Morph" effect

Description

"Morph" effect

Usage

```r
lx_effect_morph(
    period = 5,
    duration = 10^10,
    palette,
    power_on = TRUE,
    fast = FALSE,
    selector = "all",
    token = lx_get_token()
)
```

Arguments

- **period**: This controls how quickly the morph runs. It is measured in seconds. A lower number means the animation is faster.
- **duration**: How long the animation lasts for in seconds. Not specifying a duration makes the animation "never" stop (10^100 cycles). Specifying 0 makes the animation stop. Note that there is a known bug where the tile remains in the animation once it has completed if duration is nonzero.
- **palette**: array of strings (7 colors across the spectrum). You can control the colors in the animation by specifying a list of color specifiers. See `lx_color_name`
- **power_on**: if TRUE (default), switch any selected device that is off to on before performing the effect.
- **fast**: Executes the query fast, without initial state checks and wait for no results. See https://api.developer.lifx.com/docs/set-state
- **selector**: 'LIFX' api "selector" such as "all", "id:12345", or "location:kitchen". Can be created with `lx_selector` or written manually (see https://api.developer.lifx.com/docs/selectors
- **token**: API token (see `lx_save_token`). If left empty, the token is retrieved from the environmental variable if available. (see `lx_save_token`)

Value

- an `httr` response object (see `response`)
Examples

```r
## Not run:
lx_effect_morph(period = 2, palette = c("red", "blue"))

## End(Not run)
```

### Description

"Move" effect

### Usage

```r
lx_effect_move(
  direction = "forward",
  period = 1,
  cycles = 10^10,
  power_on = TRUE,
  fast = FALSE,
  selector = "all",
  token = lx_get_token()
)
```

### Arguments

- **direction**: Move direction, can be "forward" or "backward".
- **period**: The time in seconds for one cycle of the effect.
- **cycles**: The number of times to move the pattern across the device. Special cases are 0 to switch the effect off, and unspecified to continue near indefinitely (10^10 times).
- **power_on**: Switch any selected device that is off to on before performing the effect.
- **fast**: Executes the query fast, without initial state checks and wait for no results. See https://api.developer.lifx.com/docs/set-state
- **selector**: 'LIFX' api "selector" such as "all", "id:12345", or "location:kitchen". Can be created with `lx_selector` or written manually (see https://api.developer.lifx.com/docs/selectors
- **token**: API token (see ?lx_save_token). If left empty, the token is retrieved from the environmental variable if available. (see `lx_save_token`)

### Value

an 'httr' response object (see `response`)
Examples

```r
## Not run:
lx_effect_move(direction = "backward", period = 2, cycles = 5)

## End(Not run)
```

---

**lx_effect_off**  
*Turn effects off*

---

### Description

Turn effects off

### Usage

```r
lx_effect_off(power_off = FALSE, selector = "all", token = lx_get_token())
```

### Arguments

- **power_off**: If TRUE, also turns the light(s) off
- **selector**: 'LIFX' api "selector" such as "all", "id:12345", or "location:kitchen". Can be created with `lx_selector` or written manually (see [https://api.developer.lifx.com/docs/selectors](https://api.developer.lifx.com/docs/selectors))
- **token**: API token (see `lx_save_token`). If left empty, the token is retrieved from the environmental variable if available. (see `lx_save_token`)

### Value

an 'httr' response object (see `response`)

### Examples

```r
## Not run: lx_effect_off()
```

---

**lx_effect_pulse**  
"Pulse" effect

---

### Description

"Pulse" effect
lx_effect_pulse

Usage

lx_effect_pulse(
  color,
  from_color = NULL,
  period = 1,
  cycles = 1,
  persist = FALSE,
  power_on = TRUE,
  selector = "all",
  token = lx_get_token()
)

Arguments

color          The color to use for the pulse effect. use lx_color() as input
from_color     The color to start the effect from. If this parameter is omitted then the color the bulb is currently set to is used instead.
period         The time in seconds for one cycle of the effect.
cycles         The number of times to repeat the effect.
persist        boolean; If FALSE set the light back to its previous value when effect ends, if true leave the last effect color.
power_on       If FALSE, does not turn light on if it is off
selector       'LIFX' api "selector" such as "all", "id:12345", or "location:kitchen". Can be created with lx_selector or written manually (see https://api.developer.lifx.com/docs/selectors

token          API token (see ?lx_save_token). If left empty, the token is retrieved from the environmental variable if available. (see lx_save_token)

Value

an 'httr' response object (see response)

Examples

## Not run:
lx_effect_pulse(color = "red",from_color = "blue", period = 3,cycles = 5, persist = TRUE)

## End(Not run)
**lx_GET**  

*GET request*

**Description**  
GET request

**Usage**  
```r  
lx_GET(selector = "all", endpoint, token = lx_get_token())  
```

**Arguments**

- **selector**: 'LIFX' api "selector" such as "all", "id:12345", or "location:kitchen". Can be created with `lx_selector` or written manually (see https://api.developer.lifx.com/docs/selectors)
- **endpoint**: the API endpoint to call; basically the last part of the API url after the light selector
- **token**: API token (see `lx_save_token`). If left empty, the token is retrieved from the environmental variable if available. (see `lx_save_token`)

**Value**

an 'httr' response object (see `response`)

---

**lx_get_token**  

*retrieve lifx_token from R environment*

**Description**  
retrieve lifx_token from R environment

**Usage**  
```r  
lx_get_token()  
```

**Details**

To use the 'LIFX' API, you need to get a personal access token from your 'LIFX' account. Usually you save API tokens in your r environment file; that way you only have to enter it once per system. How to get a token: 1. go to https://cloud.lifx.com/sign_in and sign in (if you do not have an account, you must download the mobile app and register there. 2. generate or look up your access token

You do not need to save the token in the environment; you can use all functions in this package by passing a valid 'token' argument.
Value
the 'LIFX' api token found in environmental variables

See Also
lx_has_token, lx_save_token

Examples
## Not run:
lx_get_token()
## End(Not run)

lx_has_token
check whether a lifx api token is stored in the R environment file.

Description
check whether a lifx api token is stored in the R environment file.

Usage
lx_has_token()

Details
To use the 'LIFX' API, you need to get a personal access token from your 'LIFX' account. Usually you save API tokens in your r environment file; that way you only have to enter it once per system. How to get a token: 1. go to https://cloud.lifx.com/sign_in and sign in (if you do not have an account, you must download the mobile app and register there. 2. generate or look up your access token
You do not need to save the token in the environment; you can use all functions in this package by passing a valid 'token' argument.

Value
logical TRUE if a token was found

See Also
lx_save_token, lx_get_token

Examples
lx_has_token()
\textbf{lx_list_lights} \hspace{1cm} \textit{list available lights}

\section*{Description}
list available lights

\section*{Usage}
\begin{verbatim}
lx_list_lights(selector = "all", token = lx_get_token())
\end{verbatim}

\section*{Arguments}
\begin{itemize}
  \item \textbf{selector} \hspace{1cm} 'LIFX' api "selector" such as "all", "id:12345", or "location:kitchen". Can be created with \texttt{lx_selector} or written manually (see \url{https://api.developer.lifx.com/docs/selectors})
  \item \textbf{token} \hspace{1cm} API token (see \texttt{lx_save_token}). If left empty, the token is retrieved from the environmental variable if available. (see \texttt{lx_save_token})
\end{itemize}

\section*{Details}
each item in the returned list contains (depending on the type of lamp), the following named items:
\begin{itemize}
  \item Reachability: \texttt{connected}, \texttt{last_seen}, \texttt{seconds_since_seen}
  \item Light identifiers / selectors: \texttt{id}, \texttt{uuid}, \texttt{label}, \texttt{group}, \texttt{location}
  \item Status: \texttt{power}, \texttt{color}, \texttt{brightness}, \texttt{effect}
  \item Hardware information: \texttt{product}
\end{itemize}

\section*{Value}
a list with each item representing one light. Each item itself is a list with all relevant information about the light and it's state

\section*{Examples}
\begin{verbatim}
## Not run:
lx_list_lights()

lights <- lx_list_lights(
  lx_selector(location = "kitchen")
)

first_kitchen_light <- lights[[1]]

first_kitchen_light$power
first_kitchen_light=color$hue
first_kitchen_light=color$saturation
\end{verbatim}
### Description

POST request

### Usage

```r
lx_POST(selector = "all", endpoint, token, ...)
```

### Arguments

- **selector**: 'LIFX' api "selector" such as "all", "id:12345", or "location:kitchen". Can be created with `lx_selector` or written manually (see [https://api.developer.lifx.com/docs/selectors](https://api.developer.lifx.com/docs/selectors)).
- **endpoint**: the API endpoint to call; basically the last part of the API url after the light selector.
- **token**: API token (see `lx_save_token`). If left empty, the token is retrieved from the environmental variable if available. (see `lx_save_token`)
- **...**: named values to add to the request body

### Value

an `httr` response object (see `response`)

---

### Description

PUT request

### Usage

```r
lx_PUT(selector = "all", endpoint, token, ...)
```
lx_rate_limit

Description
get 'LIFX' API rate limit

Usage
lx_rate_limit(token = lx_get_token())

Arguments
token

Value
a named vector of three numbers:
1. ‘limit’: The rate limit
2. ‘remaining’: how many calls are remaining
3. ‘reset’: the Unix timestamp for when the next window begins. Usually every minute.

Examples
## Not run: lx_rate_limit()
**lx_save_token**

*save a lifx API token in your r environment file*

**Description**

save a lifx API token in your r environment file

**Usage**

`lx_save_token(token)`

**Arguments**

- `token` API token (see `?lx_save_token`). If left empty, the token is retrieved from the environmental variable if available. (see `lx_save_token`)

**Details**

To use the 'LIFX' API, you need to get a personal access token from your 'LIFX' account. Usually you save API tokens in your r environment file; that way you only have to enter it once per system. How to get a token: 1. go to [https://cloud.lifx.com/sign_in](https://cloud.lifx.com/sign_in) and sign in (if you do not have an account, you must download the mobile app and register there. 2. generate or look up your access token.

You do not need to save the token in the environment; you can use all functions in this package by passing a valid ‘token’ argument.

**Value**

logical TRUE if saving token has been successful

**See Also**

`lx_has_token, lx_get_token`

**Examples**

```r
## Not run:
lx_save_token("asodfjawa9499fao8u4a09fw0s8fu439wfrsud7") # put your token here

## End(Not run)
```
lx_selector  

select lights

Description

use this function to select lights that you want to communicate with

Usage

```r
lx_selector(
  id = NULL,
  label = NULL,
  group_id = NULL,
  group = NULL,
  location_id = NULL,
  location = NULL,
  zones = NULL
)
```

Arguments

- **id**: the id of the lamp(s) to select
- **label**: the label of the lamp(s) to select
- **group_id**: the group_id of the lamp(s) to select
- **group**: the group of the lamp(s) to select
- **location_id**: the location_id of the lamp(s) to select
- **location**: the location of the lamp(s) to select
- **zones**: the zones of the lamp(s) to select

Details

this creates strings to select lamps in the format that the 'LIFX' api expects (see https://api.developer.lifx.com/docs/selectors).

This function is intended to be used to create a 'selector' that is then passed to a function that changes the state of the lamps.

Value

a character string in the format expected by the 'LIFX' API for selectors. It has it's own class and printing style, but a regular character string can be used just as well.

Examples

```r
lx_selector(id = '1234')
lx_selector(label = "my_light")
lx_selector(location = 'kitchen', zone = 3)
lx_selector(location = 'kitchen', group = 'ceiling')
```
lx_state

set light state (lifx API endpoint PUT set state)

Description

set light state (lifx API endpoint PUT set state)

Usage

```
lx_state(
  power = NULL,
  color_name = NULL,
  brightness = NULL,
  infrared = NULL,
  duration = 0,
  fast = FALSE,
  selector = "all",
  token = lx_get_token()
)
```

Arguments

- **power** string - if set to "on", turns the light on, if set to "off" turns it off.
- **color_name** a color name (i.e. "red"), hexadecimal color code (i.e. "#FF0000") or output from lx_color() (in 'LIFX' api format (see https://api.developer.lifx.com/docs/colors). If this parameter is used, other parameters may be ignored.
- **brightness** set the brightness (0-1)
- **infrared** infrared brightness (0-1)
- **duration** in seconds, how long to perform the transition
- **fast** Executes the query fast, without initial state checks and wait for no results. See https://api.developer.lifx.com/docs/set-state
- **selector** 'LIFX' api "selector" such as "all", "id:12345", or "location:kitchen". Can be created with lx_selector or written manually (see https://api.developer.lifx.com/docs/selectors
- **token** API token (see ?lx_save_token). If left empty, the token is retrieved from the environmental variable if available. (see lx_save_token)

Value

an 'httr' response object (see response)

References

https://api.developer.lifx.com/docs/set-state
lx_toggle

Description
Toggle light

Usage
lx_toggle(duration = 0, selector = "all", token = lx_get_token())

Arguments
- duration: in seconds, how long to perform the transition
- selector: 'LIFX' api "selector" such as "all", "id:12345", or "location:kitchen". Can be created with lx_selector or written manually (see https://api.developer.lifx.com/docs/selectors)
- token: API token (see lx_save_token). If left empty, the token is retrieved from the environmental variable if available. (see lx_save_token)

Examples
```r
## Not run:
lx_toggle(duration = 5)
lx_toggle(selector = lx_selector(location = "kitchen"))
```
## End(Not run)
Index

check_lifx_response, 2

lifx, 3
lx_check_color, 3
lx_color, 3, 4
lx_color_name, 3, 5, 10
lx_delta, 6
lx_effect_breathe, 3, 7
lx_effect_flame, 9
lx_effect_morph, 10
lx_effect_move, 11
lx_effect_off, 12
lx_effect_pulse, 12
lx_GET, 14
lx_get_token, 14, 15, 19
lx_has_token, 15, 15, 19
lx_list_lights, 3, 16
lx_POST, 17
lx_PUT, 17
lx_rate_limit, 18
lx_save_token, 3, 5–19, 19, 21, 22
lx_selector, 5, 7–14, 16–18, 20, 21, 22
lx_state, 21
lx_toggle, 22

response, 5, 7–14, 17, 18, 21