Version: 1.0

Light

The Light component handles a dimmable light output with additional on/off control. It has night mode capability that can reduce brightness in selected period of time. It uses `Light` as RPC namespace and provides the methods:

- `Light.GetConfig` to obtain the component's configuration
- `Light.SetConfig` to update the component's configuration
- `Light.GetStatus` to obtain the component's status
- `Light.Set` to control the output state and brightness level
- `Light.Toggle` to toggle the output state
- `Light.DimUp` to dim up
- `Light.DimDown` to dim down
- `Light.DimStop` to stop dimming
- `Light.SetAll` (if applicable)
- `Light.Calibrate` (if applicable)
- `Light.ResetCounters` to reset component's energy counters (if applicable)

Light components are identified with `light:<id>` in objects containing multiple component payloads.

Methods:

Light.SetConfig

Property	Type	Description
`id`	*number*	Id of the Light component instance
`config`	*object*	Configuration that the method takes

Find more about the config properties in config section

Light.GetConfig

Properties:

Property	Type	Description
`id`	*number*	Id of the Light component instance

Find the Light.GetConfig response properties in config section

Light.GetStatus

Properties:

Property	Type	Description
`id`	*number*	Id of the Light component instance

Find more about the status response properties in status section

Light.Set

This method sets the output and brightness level of the Light component. It can be used to trigger webhooks. More information about the events triggering webhooks available for this component can be found below.

Request

Parameters:

Property	Type	Description
`id`	*number*	Id of the Light component instance. **Required**
`on`	*boolean*	True for light on, false otherwise. **Optional**
`brightness`	*number*	Brightness level **Optional**
`transition_duration`	*number*	Transition time in seconds - time between change from current brightness level to desired brightness level in request **Optional**
`toggle_after`	*number*	Optional flip-back timer in seconds. **Optional**
`offset`	*number*	Set current brightness level with applied offset. Can not be used together with `brightness`. Boundaries [-100, 100] **Optional**

> **INFO**
>
> At least one of the `on` and `brightness` parameters is required.

Light.Toggle

This method toggles the output state. It can be used to trigger webhooks. More information about the events triggering webhooks available for this component can be found below.

Request

Parameters:

Property	Type	Description
`id`	*number*	Id of the Light component instance. **Required**

Light.DimUp

This method dims up the brightness level. Dimming stops with Light.DimStop.

Request

Parameters:

Property	Type	Description
`id`	*number*	Id of the Light component instance. **Required**
`fade_rate`	*number*	Fade rate of the brightness level dimming. Range `[1,5]` where `5` is fastest, `1` is slowest. If not provided, value is defaulted to `button_fade_rate`. **Optional**

Light.DimDown

This method dims down the brightness level. Dimming stops with Light.DimStop.

Request

Parameters:

Property	Type	Description
`id`	*number*	Id of the Light component instance. **Required**
`fade_rate`	*number*	Fade rate of the brightness level dimming. Range `[1,5]` where `5` is fastest, `1` is slowest. If not provided, value is defaulted to `button_fade_rate`. **Optional**

Light.DimStop

This method stops the dimming of the brightness level.

Request

Parameters:

Property	Type	Description
`id`	*number*	Id of the Light component instance. **Required**

Light.SetAll

This method (if applicalbe) sets the output and brightness level of all `Light` components in the device. It can be used to trigger webhooks. More information about the events triggering webhooks available for this component can be found below.

Request

Parameters:

Property	Type	Description
`on`	*boolean*	True for light on, false otherwise. **Optional**
`brightness`	*number*	Brightness level **Optional**
`transition_duration`	*number*	Transition time in seconds - time between change from current brightness level to desired brightness level in request **Optional**
`toggle_after`	*number*	Optional flip-back timer in seconds. **Optional**
`offset`	*number*	Set current brightness level with applied offset. Can not be used together with `brightness`. Boundaries [-100, 100] **Optional**

> **INFO**
>
> At least one of the `on` and `brightness` parameters is required.

Light.Calibrate

This method (if applicalbe) starts calibration of device's outputs.

Request

Parameters:

Property	Type	Description
`id`	*number*	Id of the Light component instance*

- In PlusRGBWPM all `light` instances are calibrated with one call to `Light.Calibrate` and the `id` parameter is ignored.

Response

The result from this method is `null` when calibration is started succesfully.

If invoked while calibration is already in progress, an error is returned:

```
{
  "code": -114,
  "message": "Callibration is in progress"
}
```

Light.ResetCounters

This method resets associated counters (if applicable).

Request

Parameters:

Property	Type	Description
`id`	*number*	Id of the Light component instance. **Required**
`type`	*array of strings*	Array of strings, selects which counter to reset **Optional**

> **NOTE**
> If no 'type' is provided, the method will reset all available counters.

Response

Attributes in the result:

<table>
<tr><th>Property</th><th>Type</th><th>Description</th></tr>
<tr><td><code>aenergy</code></td><td><em>object</em></td><td>Information about the active energy counter prior to reset
<table>
<tr><th>Property</th><th>Type</th><th>Description</th></tr>
<tr><td><code>total</code></td><td><em>number</em></td><td>Last counter value of the total energy consumed in Watt-hours</td></tr>
</table></td></tr>
</table>

HTTP Endpoint: /light/`id`

Through this endpoint a light can be turned on/off with or without a timer, brightness can be changed also. This can be used to trigger webhooks. More information about the events triggering webhooks available for this component can be found below.

Request

Parameters:

Property	Type	Description
`turn`	*string*	Action to be executed. Range of values: `on`, `off`, `toggle`. **Required**
`timer`	*number*	A one-shot flip-back timer in seconds.
`brightness`	*number*	Brightness, 0..100 %
`transition`	*number*	One-shot transition, 500..10800000 [ms]

Response

Received attributes:

Property	Type	Description
`ison`	*boolean*	True if the light is turned on, false otherwise
`brightness`	*number*	Brightness, 0..100 %
`transition`	*number*	One-shot transition, 500..10800000 [ms]
`has_timer`	*boolean*	True if the light is turned on, false otherwise
`timer_started_at`	*number*	Unix timestamp, start time of the timer (in UTC)
`timer_duration`	*number*	Duration of the timer in seconds
`timer_remaining`	*number*	Time remaining (in seconds) until the request is executed
`source`	*string*	Source of the last command, for example: `init`, `WS_in`, `http`, ...

Configuration

The configuration of the Light component contains information about night mode settings, button presets and the timers of the chosen light instance. To Get/Set the configuration of the Light component its `id` must be specified.

Properties:

Property	Type	Description
`id`	*number*	Id of the Light component instance
`name`	*string or null*	Name of the light instance
`in_mode`	*string*	Mode of the associated input. Range of values: `follow`, `flip`, `activate`, `detached`, `dim` (if applicable), `dual_dim` (if applicable)
`op_mode`	*number*	Operational mode (if applicable), which is device-specific. See the table below.
`initial_state`	*string*	Output state to set on power_on. Range of values: `off`, `on`, `restore_last`
`auto_on`	*boolean*	True if the "Automatic ON" function is enabled, false otherwise
`auto_on_delay`	*number*	Seconds to pass until the component is switched back on
`auto_off`	*boolean*	True if the "Automatic OFF" function is enabled, false otherwise
`auto_off_delay`	*number*	Seconds to pass until the component is switched back off
`transition_duration`	*number*	Transition time (in seconds) - time to change from 0% to 100% of brightness (if applicable)
`min_brightness_on_toggle`	*number*	Brightness level (in percent) applied when there is a toggle and current brightness is lower than `min_brightness_on_toggle`.
`night_mode.enable`	*boolean*	Enable or disable night mode
`night_mode.brightness`	*number*	Brightness level limit when night mode is active. Default value `50`.
`night_mode.active_between`	*array*	Containing 2 elements of type string, the first element indicates the start of the period during which the night mode will be active, the second indicates the end of that period. Both start and end are strings in the format HH:MM, where HH and MM are hours and minutes with optinal leading zeros

<table>
<tr><th>Property</th><th>Type</th><th>Description</th></tr>
<tr><td>`button_fade_rate`</td><td>*number*</td><td>Controls how quickly the output level changes while a button is held down for dimming (if applicable). Default value `3`. Range `[1,5]` where `5` is fastest, `1` is slowest</td></tr>
<tr><td>`button_presets`</td><td>*object*</td><td>Button presets config
<table>
<tr><th>Property</th><th>Type</th><th>Description</th></tr>
<tr><td>`button_doublepush`</td><td>*object*</td><td>`null` disables `button_doublepush` preset
<table>
<tr><th>Property</th><th>Type</th><th>Description</th></tr>
<tr><td>`brightness`</td><td>*number*</td><td>Brightness level (in percent) set on double click (if applicable), default: `100`</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>`range_map`</td><td>*array or null*</td><td>Remaps output 0%-100% range to values in array (if applicable). First value in array is `min` setting, second value is `max` setting. Array elements are of type number. Accepted range for values is from 0% to 100%. Default values are `[0, 100]`. `max` must be greater than `min`.</td></tr>
<tr><td>`power_limit`</td><td>*number*</td><td>Limit (in Watts) over which overpower condition occurs (shown if applicable)</td></tr>
<tr><td>`voltage_limit`</td><td>*number*</td><td>Limit (in Volts) over which overvoltage condition occurs (shown if applicable)</td></tr>
<tr><td>`undervoltage_limit`</td><td>*number*</td><td>Limit (in Volts) under which undervoltage condition occurs (shown if applicable)</td></tr>
<tr><td>`current_limit`</td><td>*number*</td><td>Limit (in Amperes) over which overcurrent condition occurs (shown if applicable). For PlusRGBWPM shown if device is calibrated.</td></tr>
</table>

Valid `op_mode` values depend on the device.

op_mode	0	1
Dimmer0/1-10V PM Gen3	0-10VDC (default)	1-10VDC

INFO

- `follow`: the state of the output is the same as the state of the input (e.g. when the input is off => the output is off). For `type`:`analog` sets to output current `%` of input.
- `flip`: change of the state of the input causes change of the state of the output (e.g. when input is toggled the output is also toggled)
- `activate`: when input state is `on` sets output to `on` (and activates `auto_off` if enabled), input state `off` does nothing.
- `detached`: the state of the input doesn't affect the state of the output.
- `dim`: short press toggles output, long press starts dimming with alternating directions. If output is currently off, holding the button will turn on and start dimming from 0%. Stops when reached 100% or 1%.
- `dual_dim`: short press toggles output, long press starts dimming up/down depending on which button is pressed. Stops when reached 100% or 1%.

  `follow` and `flip` are available for `type`:`switch`.

  `activate` available for `type`:`switch` and `type`:`button`

  `dim` available for `type`:`button`

  `dual_dim` available for `type`:`button`

Status

The status of the Light component contains information about the brightness level and output state of the light instance. To obtain the status of the Light component its `id` must be specified.

Properties:

Property	Type	Description
`id`	*number*	Id of the Light component instance
`source`	*string*	Source of the last command, for example: `init`, `WS_in`, `http`, ...
`output`	*boolean*	True if the output channel is currently on, false otherwise
`brightness`	*number*	Current brightness level (in percent)
`timer_started_at`	*number*	Unix timestamp, start time of the timer (in UTC) (shown if the timer is triggered)

<table>
<tr><th>Property</th><th>Type</th><th>Description</th></tr>
<tr><td>`timer_duration`</td><td>*number*</td><td>Duration of the timer in seconds (shown if the timer is triggered)</td></tr>
<tr><td>`transition`</td><td>*object*</td><td>Information about the transition (shown if transition is triggered)
<table>
<tr><th>Property</th><th>Type</th><th>Description</th></tr>
<tr><td>`target.output`</td><td>*boolean*</td><td>True if the output channel becomes on, false otherwise</td></tr>
<tr><td>`target.brightness`</td><td>*number*</td><td>Brightness level (in percent)</td></tr>
<tr><td>`started_at`</td><td>*number*</td><td>Unix timestamp, start time of the transition (in UTC)</td></tr>
<tr><td>`duration`</td><td>*number*</td><td>Duration of the transition in seconds</td></tr>
</table>
</td></tr>
<tr><td>`temperature`</td><td>*object*</td><td>Information about the temperature (if applicable)
<table>
<tr><th>Property</th><th>Type</th><th>Description</th></tr>
<tr><td>`tC`</td><td>*number or null*</td><td>Temperature in Celsius (`null` if temperature is out of the measurement range)</td></tr>
<tr><td>`tF`</td><td>*number or null*</td><td>Temperature in Fahrenheit (`null` if temperature is out of the measurement range)</td></tr>
</table>
</td></tr>
<tr><td>`aenergy`</td><td>*object*</td><td>Information about the active energy counter (shown if applicable)
<table>
<tr><th>Property</th><th>Type</th><th>Description</th></tr>
<tr><td>`total`</td><td>*number*</td><td>Total energy consumed in **Watt-hours**</td></tr>
<tr><td>`by_minute`</td><td>*array of type number*</td><td>Energy consumption in **Milliwatt-hours** for the last three complete minutes. The 0-th element indicates the counts accumulated during the minute preceding `minute_ts`. Present only if the device clock is synced.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Property</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td><table><tr><th>Property</th><th>Type</th><th>Description</th></tr><tr><td><code>minute_ts</code></td><td><em>number</em></td><td>Unix timestamp marking the start of the current minute (in UTC).</td></tr></table></td></tr>
<tr><td><code>apower</code></td><td><em>number</em></td><td>Last measured instantaneous active power (in Watts) delivered to the attached load (shown if applicable)</td></tr>
<tr><td><code>voltage</code></td><td><em>number</em></td><td>Last measured voltage in Volts (shown if applicable)</td></tr>
<tr><td><code>current</code></td><td><em>number</em></td><td>Last measured current in Amperes (shown if applicable)</td></tr>
<tr><td><code>calibration</code></td><td><em>object</em></td><td>Information about the calibration process, only present when calibration is running (shown if applicable).<table><tr><th>Property</th><th>Type</th><th>Description</th></tr><tr><td><code>progess</code></td><td><em>number</em></td><td>Calibration progress in percent</td></tr></table></td></tr>
<tr><td><code>errors</code></td><td><em>array of type string</em></td><td>Error conditions occurred, shown if at least one error is present. Depending on component capabilities may contain: <code>overtemp</code>, <code>overpower</code>, <code>overvoltage</code>, <code>undervoltage</code>, <code>overcurrent</code>, <code>unsupported_load</code>, <code>cal_abort:interrupted</code>, <code>cal_abort:power_read</code>, <code>cal_abort:no_load</code>, <code>cal_abort:no_synchro</code>, <code>cal_abort:non_dimmable</code>, <code>cal_abort:overpower</code>, <code>cal_abort:unsupported_load</code></td></tr>
<tr><td><code>flags</code></td><td><em>array of type string</em></td><td>Communicates present conditions, shown if at least one flag is set. Depending on component capabilites may contain: <code>no_load</code>, <code>uncalibrated</code></td></tr>
</table>

INFO

- If applicable, calibration is performed by `Light.Calibrate`. The calibration procedure measures consumption for each channel and should be run when there is change in loads. Consumed power of the load should be >= 3.5W.

  The following errors can occur during the calibration procedure:

  Errors:

  - `cal_abort:interrupted`: input or output are triggered
  - `cal_abort:power_read`: there is an error in reading the powermeter
  - `cal_abort:no_load`: there is no load attached

  - `cal_abort:non_dimmable`: attached load is not dimmable
  - `cal_abort:overpower`: load power exceeds max limit
  - `cal_abort:unsupported_load`: load can not be calibrated

Webhook Events

Currently, there are two events related to the Light component that can trigger webhooks:

- `light.on` - produced when the light changes its state from `off` to `on`
- `light.off` - produced when the light changes its state from `on` to `off`

MQTT Control

Since version 1.6.0

- Shelly will subscribe to the following topics, accepting commands:
  - `<topic_prefix>/command/light:<id>`
- Shelly publishes on:
  - `<topic_prefix>/status/light:<id>` - topic where the result of the `status_update` command is published. The command can be sent either on the common device topic or the component-specific topic. This topic is already used for existing status notifications if enabled.
  - `<topic_prefix>/error/light:<id>` - error message is published if the light command receives incorrect parameters or results in an error.
- Accepted commands are:
  - `status_update` - causes the status of the corresponding component to be published on its `<topic_prefix>/status/light:<id>` topic.
  - `toggle` - toggles the light.
  - `set,[boolean,number,number,number]` - sets the light state to `true` or `false` (respectively on/off) and/or its `brightness` [0-100], optional `transition_duration`, `toggle_after` parameters. Having at least one of either state (`true`/`false`) or brightness is mandatory.

> **NOTE**
>
> `<topic_prefix>` is a custom prefix if set or defaults to `<device_id>`

Examples

Light.SetConfig example

Light.SetConfig HTTP GET Request Light.SetConfig Curl Request Light.SetConfig Mos Request

```
http://192.168.33.1/rpc/Light.SetConfig?id=0&config={"name":"Light0"}
```

Response

```
null
```

Light.GetConfig example

Light.GetConfig HTTP GET Request   Light.GetConfig Curl Request   Light.GetConfig Mos Request

```
http://192.168.33.1/rpc/Light.GetConfig?id=0
```

Response

```
{
  "id": 0,
  "name": null,
  "initial_state": "restore_last",
  "auto_on": true,
  "auto_on_delay": 60,
  "auto_off": true,
  "auto_off_delay": 60,
  "transition_duration": 3,
  "min_brightness_on_toggle": 3,
  "button_presets": {
    "button_doublepush": {
      "brightness": 100
    }
  },
  "night_mode": {
    "enable": true,
    "brightness": 50,
    "active_between": [
      "21:45",
      "05:30"
    ]
  }
}
```

Light.GetStatus example

Light.GetStatus HTTP GET Request   Light.GetStatus Curl Request   Light.GetStatus Mos Request

```
http://192.168.33.1/rpc/Light.GetStatus?id=0
```

Response

```
{
  "id": 0,
  "source": "timer",
  "output": false,
  "brightness": 50,
  "timer_started_at": 1626942399.36,
  "timer_duration": 60,
  "transition": {
    "target": {
      "output": true,
      "brightness": 100
    },
    "started_at": 1626942399.36,
    "duration": 20
  }
}
```

Light.Set example

Light.Set HTTP GET Request | Light.Set Curl Request | Light.Set Mos Request

```
http://192.168.33.1/rpc/Light.Set?id=0&on=true&brightness=50&transition_duration=20
```

Response

```
null
```

Light.Toggle example

Light.Toggle HTTP GET Request | Light.Toggle Curl Request | Light.Toggle Mos Request

```
http://192.168.33.1/rpc/Light.Toggle?id=0
```

Response

```
null
```

Light.DimUp example

Light.DimUp HTTP GET Request | Light.DimUp Curl Request | Light.DimUp Mos Request

```
http://192.168.33.1/rpc/Light.DimUp?id=0&fade_rate=3
```

Response

```
null
```

Light.DimDown example

Light.DimDown HTTP GET Request Light.DimDown Curl Request Light.DimDown Mos Request

```
http://192.168.33.1/rpc/Light.DimDown?id=0&fade_rate=3
```

Response

```
null
```

Light.DimStop example

Light.DimStop HTTP GET Request Light.DimStop Curl Request Light.DimStop Mos Request

```
http://192.168.33.1/rpc/Light.DimStop?id=0
```

Response

```
null
```

Light.SetAll example

Light.SetAll HTTP GET Request Light.SetAll Curl Request Light.SetAll Mos Request

```
http://192.168.33.1/rpc/Light.SetAll?on=true&brightness=50&transition_duration=20
```

Response

```
null
```

Light.Calibrate example

Light.Calibrate HTTP GET Request Light.Calibrate Curl Request Light.Calibrate Mos Request

```
http://192.168.33.1/rpc/Light.Calibrate?id=0
```

Response

```
null
```

Light.ResetCounters example

Light.ResetCounters HTTP GET Request Light.ResetCounters Curl Request Light.ResetCounters Mos Request

```
http://192.168.33.1/rpc/Light.ResetCounters?id=0&type=["aenergy"]
```

Response

```
{
  "aenergy": {
    "total": 11.679
  }
}
```

HTTP Endpoint example

> *Example:*
>
> ```
> curl http://${SHELLY}/light/0?turn=on
> ```

> *Example:*
>
> ```
> {
>   "ison": true,
>   "brightness": 100,
>   "transition": 10,
>   "has_timer": true,
>   "timer_started_at": 258,
>   "timer_duration": 10,
>   "timer_remaining": 1.98,
>   "source": "http"
> }
> ```

Set Light over MQTT example

Set Light on with brightness to max, transition_duration of 5 seconds and toggle_after 10 seconds over MQTT example

```
export MQTT_SERVER="broker.hivemq.com"
export MQTT_PORT=1883
export SHELLY_ID="shellyprodm1pm-441793ce3f08" # The <shelly-id> of your device
mosquitto_pub -h ${MQTT_SERVER} -p ${MQTT_PORT} -t ${SHELLY_ID}/command/light:0 -m set,true,100,5,10
```