Version: 1.0

RGB

The RGB component handles an output with possibility to change color and brightness of RGB LED load. Component has additional on/off control. It has night mode capability that can reduce brightness in selected period of time. It uses `RGB` as RPC namespace and provides the methods:

- `RGB.GetConfig` to obtain the component's configuration
- `RGB.SetConfig` to update the component's configuration
- `RGB.GetStatus` to obtain the component's status
- `RGB.Set` to control the output state, color, transition and brightness level
- `RGB.Toggle` to toggle the output state
- `RGB.DimUp` to dim up
- `RGB.DimDown` to dim down
- `RGB.DimStop` to stop dimming

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

Methods:

RGB.SetConfig

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

Find more about the config properties in config section

RGB.GetConfig

Properties:

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

Find the RGB.GetConfig response properties in config section

RGB.GetStatus

Properties:

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

Find more about the status response properties in status section

RGB.Set

This method sets the output, color, transition and brightness level of the RGB 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 RGB component instance. **Required**
`on`	*boolean*	True for output on, false otherwise. **Optional**
`brightness`	*number*	Brightness level 1%-100% **Optional**
`rgb`	*array of type number*	Red, Green, Blue [r,g,b] - each value represents level between 0..255 **Optional**
`transition_duration`	*number*	Transition time in seconds - time between change from current brightness level and color to desired brightness level and color in request **Optional**
`toggle_after`	*number*	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.

RGB.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 RGB component instance. **Required**

RGB.DimUp

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

Request

Parameters:

Property	Type	Description
`id`	*number*	Id of the RGB 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**

RGB.DimDown

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

Request

Parameters:

Property	Type	Description
`id`	*number*	Id of the RGB 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**

RGB.DimStop

This method stops the dimming of the brightness level.

Request

Parameters:

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

HTTP Endpoint: /color/`id`

Through this endpoint a light can be turned on/off with or without a timer, brightness and color 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 %
`red`	*number*	Intensity of the color red, 0..255
`green`	*number*	Intensity of the color green, 0..255
`blue`	*number*	Intensity of the color blue, 0..255
`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 %

Property	Type	Description
`red`	*number*	Intensity of the color red, 0..255
`green`	*number*	Intensity of the color green, 0..255
`blue`	*number*	Intensity of the color blue, 0..255
`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 RGB component contains information about night mode settings, the timers of the chosen light instance. To Get/Set the configuration of the RGB component its `id` must be specified.

Properties:

Property	Type	Description
`id`	*number*	Id of the RGB component instance
`name`	*string or null*	Name of the RGB instance
`in_mode`	*string*	Mode of the associated input. Range of values: `follow`, `flip`, `activate`, `detached`, `dim`
`initial_state`	*string*	Output state to set on power_on. Range of values: `off`, `on`, `restore_last`

Property	Type	Description
`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 or RGB levels
`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 or null*	Brightness level limit (in percent) when night mode is active. `null` overrides `night_mode.brightness` with current brightness when night mode starts. Default value `50`.
`night_mode.rgb`	*array of type number or null*	Color level when night mode is active. Red, Green, Blue [r,g,b] - each value represents level between 0..255. `null` overrides `night_mode.rgb` array with current `rgb` array when night mode starts. Default value `255` for each color
`night_mode.active_between`	*array of type string*	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
`button_fade_rate`	*number*	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
`button_presets`	*object*	Button presets config | Property | Type | Description | |---|---|---| | `button_doublepush` | *object* | `null` disables `button_doublepush` preset |

Property	Type	Description				
		Property	Type	Description		
				Property	Type	Description
				`brightness`	*number or null*	Brightness level (in percent) set on double click (if applicable). `null` overrides `brightness` with current brightness when preset is applied. Default `100`
				`rgb`	*array of type number or null*	Color level set on double click (if applicable). Red, Green, Blue [r,g,b] - each value represents level between 0..255. `null` overrides `rgb` array with current `rgb` array when preset is applied. Default value `255` for each color

Property	Type	Description
`current_limit`	*number*	Limit (in Amperes) over which overcurrent condition occurs (shown if applicable). For specific devices applies for color channels, not RGB component
`power_limit`	*number*	Limit (in Watts) over which overpower condition occurs (shown if applicable)
`voltage_limit`	*number*	Limit (in Volts) over which overvoltage condition occurs (shown if applicable)

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%.

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

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

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

Status

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

Properties:

Property	Type	Description
`id`	*number*	Id of the RGB component instance
`source`	*string*	Source of the last command, for example: `init`, `WS_in`, `http`, ...

Property	Type	Description
`output`	*boolean*	True if the output channel is currently on, false otherwise
`rgb`	*array of type number*	Current Red, Green, Blue [r,g,b] level 0..255
`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)
`timer_duration`	*number*	Duration of the timer in seconds (shown if the timer is triggered)
`transition`	*object*	Information about the transition (shown if transition is triggered)

`transition`:

Property	Type	Description
`target.output`	*boolean*	True if the output channel becomes on, false otherwise
`target.rgb`	*array of type number*	Red, Green, Blue [r,g,b] level 0..255
`target.brightness`	*number*	Brightness level (in percent)
`started_at`	*number*	Unix timestamp, start time of the transition (in UTC)
`duration`	*number*	Duration of the transition in seconds

Property	Type	Description
`temperature`	*object*	Information about the temperature (shown if applicable) **Property** / **Type** / **Description**: `tC` — *number or null* — Temperature in Celsius (`null` if temperature is out of the measurement range) `tF` — *number or null* — Temperature in Fahrenheit (`null` if temperature is out of the measurement range)
`aenergy`	*object*	Information about the active energy counter (shown if applicable) **Property** / **Type** / **Description**: `total` — *number* — Total energy consumed in **Watt-hours** `by_minute` — *array of type number* — 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. `minute_ts` — *number* — Unix timestamp marking the start of the current minute (in UTC).
`apower`	*number*	Last measured instantaneous active power (in Watts) delivered to the attached load (shown if applicable)
`voltage`	*number*	Last measured voltage in Volts (shown if applicable)
`current`	*number*	Last measured current in Amperes (shown if applicable)
`errors`	*array of type string*	Error conditions occurred. May contain `overtemp`, (shown if at least one error is present)

Webhook Events

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

- `rgb.on` - produced when the output changes its state from `off` to `on`
- `rgb.off` - produced when the output 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/rgb:<id>`
- Shelly publishes on:
  - `<topic_prefix>/status/rgb:<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/rgb:<id>` - error message is published if the rgb 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/rgb:<id>` topic.
  - `toggle` - toggles the rgb.
  - `set,[boolean,number,number,number,number,number,number]` - sets the rgb state to `true` or `false` (respectively on/off) and/or its `brightness` [0-100], optional `red` (0-255), `green` [0-255], `blue` [0-255], `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

RGB.SetConfig example

RGB.SetConfig HTTP GET Request | RGB.SetConfig Curl Request | RGB.SetConfig Mos Request

```
http://192.168.33.1/rpc/RGB.SetConfig?id=0&config={"name":"RGB0"}
```

Response

```
null
```

RGB.GetConfig example

RGB.GetConfig HTTP GET Request RGB.GetConfig Curl Request RGB.GetConfig Mos Request

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

Response

```
{
  "id": 0,
  "name": null,
  "in_mode": "dim",
  "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,
      "rgb": [
        255,
        255,
        255
      ]
    }
  },
  "night_mode": {
    "enable": true,
    "brightness": 50,
    "rgb": [
      255,
      255,
      255
    ],
    "active_between": [
      "21:45",
      "05:30"
    ]
  },
  "button_fade_rate": 3
}
```

RGB.GetStatus example

RGB.GetStatus HTTP GET Request RGB.GetStatus Curl Request RGB.GetStatus Mos Request

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

Response

```
{
  "id": 0,
  "source": "timer",
  "output": true,
  "rgb": [
    0,
    0,
    127
  ],
  "brightness": 50,
  "timer_started_at": 1626942399.36,
  "timer_duration": 60,
  "transition": {
    "target": {
      "output": true,
      "brightness": 50,
      "rgb": [
        20,
        30,
        40
      ]
    },
    "started_at": 1626942399.36,
    "duration": 20
  },
  "temperature": {
    "tC": 53.1,
    "tF": 127.6
  },
  "aenergy": {
    "total": 0.55
  },
  "apower": 2,
  "current": 0.165,
  "voltage": 12.1
}
```

RGB.Set example

RGB.Set HTTP GET Request **RGB.Set Curl Request** **RGB.Set Mos Request**

```
http://192.168.33.1/rpc/RGB.Set?id=0&on=true&brightness=50&rgb=[20,30,40]&transition_duration=20
```

Response

```
null
```

RGB.Toggle example

RGB.Toggle HTTP GET Request **RGB.Toggle Curl Request** **RGB.Toggle Mos Request**

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

Response

```
null
```

RGB.DimUp example

RGB.DimUp HTTP GET Request RGB.DimUp Curl Request RGB.DimUp Mos Request

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

Response

```
null
```

RGB.DimDown example

RGB.DimDown HTTP GET Request RGB.DimDown Curl Request RGB.DimDown Mos Request

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

Response

```
null
```

RGB.DimStop example

RGB.DimStop HTTP GET Request RGB.DimStop Curl Request RGB.DimStop Mos Request

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

Response

```
null
```

HTTP Endpoint example

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

> *Example:*
>
> ```
> {
>   "ison": true,
>   "brightness": 100,
> ```

```
    "red": 127,
    "green": 0,
    "blue": 0,
    "transition": 10,
    "has_timer": true,
    "timer_started_at": 258,
    "timer_duration": 10,
    "timer_remaining": 1.98,
    "source": "http"
  }
```

Set RGB over MQTT example

Set RGB on with brightness to max, all RGB values set to max, transition_duration of 5 seconds and toggle_after 10 seconds

```
export MQTT_SERVER="broker.hivemq.com"
export MQTT_PORT=1883
export SHELLY_ID="shellyplusrgbwpm-b48a0a123db4" # The <shelly-id> of your device
mosquitto_pub -h ${MQTT_SERVER} -p ${MQTT_PORT} -t ${SHELLY_ID}/command/rgb:0 -m
set,true,100,255,255,255,5,10
```