Version: 1.0

CCT

The CCT component handles a dimmable light output with adjustable hues of white (cold/warm) and on/off control. It has night mode capability that can reduce brightness and color temperature in selected period of time. It uses `CCT` as RPC namespace and provides the methods:

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

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

Methods:

CCT.SetConfig

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

Find more about the config properties in config section

CCT.GetConfig

Properties:

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

Find the CCT.GetConfig response properties in config section

CCT.GetStatus

Properties:

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

Find more about the status response properties in status section

CCT.Set

This method sets the output, brightness and color temperature level of the CCT 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 CCT component instance. **Required**
`on`	*boolean*	True for light on, false otherwise. **Optional**
`brightness`	*number*	Brightness level **Optional**
`ct`	*number*	Color temperature level (in Kelvin) **Optional**
`transition_duration`	*number*	Transition time in seconds - time between change from current brightness level to desired brightness level and current color temperature level to desired color temperature 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`, `brightness` and `ct` parameters is required.

CCT.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 CCT component instance. **Required**

CCT.DimUp

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

Request

Parameters:

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

CCT.DimDown

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

Request

Parameters:

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

CCT.DimStop

This method stops the dimming of the brightness level.

Request

Parameters:

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

HTTP Endpoint: /cct/`id`

Through this endpoint a light can be turned on/off with or without a timer, brightness and color temperature level 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 %
`temp`	*number*	Color temperature level (in Kelvin), accepted value is device specific. Default range for DuoBulbG3: `[2700, 6500]`.
`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 %
`temp`	*number*	Color temperature level (in Kelvin)
`transition`	*number*	One-shot transition, 500..10800000 [ms]

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

Properties:

Property	Type	Description
`id`	*number*	Id of the CCT component instance
`name`	*string or null*	Name of the CCT instance
`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)

<table>
<tr><th>Property</th><th>Type</th><th>Description</th></tr>
<tr><td>`min_brightness_on_toggle`</td><td>*number*</td><td>Brightness level (in percent) applied when there is a toggle and current brightness is lower than `min_brightness_on_toggle`.</td></tr>
<tr><td>`night_mode.enable`</td><td>*boolean*</td><td>Enable or disable night mode</td></tr>
<tr><td>`night_mode.brightness`</td><td>*number or null*</td><td>Brightness level limit when night mode is active. `null` overrides `night_mode.brightness` with current brightness when night mode starts. Default value `50`.</td></tr>
<tr><td>`night_mode.ct`</td><td>*number or null*</td><td>Color temperature level limit (in Kelvin) when night mode is active. `null` overrides `night_mode.ct` value with current `ct` value when night mode starts. Default value: `50%` of `ct_range`. For DuoBulbG3: `4600`</td></tr>
<tr><td>`night_mode.active_between`</td><td>*array*</td><td>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</td></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 (if applicable)
<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 or null*</td><td>Brightness level (in percent) set on double click (if applicable). `null` overrides `brightness` with current brightness when preset is applied.</td></tr>
</table>
</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></td><td></td><td>
<table>
<tr><th>Property</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>Default: <code>100</code></td></tr>
<tr><td><code>ct</code></td><td><i>number or null</i></td><td>Color temperature level (in Kelvin) set on double click (if applicable). <code>null</code> overrides <code>ct</code> with current ct when preset is applied. Default: <code>max</code> setting of <code>ct_range</code> (device specific)</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td><code>range_map</code></td><td><i>array or null</i></td><td>Remaps output 0%-100% range to values in array (if applicable). First value in array is <code>min</code> setting, second value is <code>max</code> setting. Array elements are of type number. Accepted range for values is from 0% to 100%. Default values are <code>[0, 100]</code>. <code>max</code> must be greater than <code>min</code>.</td></tr>
<tr><td><code>ct_range</code></td><td><i>array or null</i></td><td>Sets the color temperature operating range in Kelvin (if applicable). First value in array is <code>min</code> setting, second value is <code>max</code> setting. Array elements are of type number. Accepted range for values is from 1000K to 10000K. Default values are device specific. Default values for DuoBulbG3: <code>[2700, 6500]</code>. <code>max</code> must be greater than <code>min</code>.</td></tr>
<tr><td><code>current_limit</code></td><td><i>number</i></td><td>Limit (in Amperes) over which overcurrent condition occurs (shown if applicable). For specific devices applies for color channels, not CCT component</td></tr>
<tr><td><code>power_limit</code></td><td><i>number</i></td><td>Limit (in Watts) over which overpower condition occurs (shown if applicable)</td></tr>
</table>

Property	Type	Description
`voltage_limit`	*number*	Limit (in Volts) over which overvoltage condition occurs (shown if applicable)

Status

The status of the CCT component contains information about the brightness level, color temperature level and the output state of the CCT instance. To obtain the status of the CCT component its `id` must be specified.

Properties:

<table>
<tr><th>Property</th><th>Type</th><th>Description</th></tr>
<tr><td><code>id</code></td><td><i>number</i></td><td>Id of the CCT component instance</td></tr>
<tr><td><code>source</code></td><td><i>string</i></td><td>Source of the last command, for example: <code>init</code>, <code>WS_in</code>, <code>http</code>, ...</td></tr>
<tr><td><code>output</code></td><td><i>boolean</i></td><td>True if the output channel is currently on, false otherwise</td></tr>
<tr><td><code>brightness</code></td><td><i>number</i></td><td>Current brightness level (in percent)</td></tr>
<tr><td><code>ct</code></td><td><i>number</i></td><td>Current color temperature level (in Kelvin)</td></tr>
<tr><td><code>timer_started_at</code></td><td><i>number</i></td><td>Unix timestamp, start time of the timer (in UTC) (shown if the timer is triggered)</td></tr>
<tr><td><code>timer_duration</code></td><td><i>number</i></td><td>Duration of the timer in seconds (shown if the timer is triggered)</td></tr>
<tr><td><code>transition</code></td><td><i>object</i></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><code>target.output</code></td><td><i>boolean</i></td><td>True if the output channel becomes on, false otherwise</td></tr>
<tr><td><code>target.brightness</code></td><td><i>number</i></td><td>Brightness level (in percent)</td></tr>
<tr><td><code>target.ct</code></td><td><i>number</i></td><td>Color temperature level (in Kelvin)</td></tr>
<tr><td><code>started_at</code></td><td><i>number</i></td><td>Unix timestamp, start time of the transition (in UTC)</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>duration</code></td><td><em>number</em></td><td>Duration of the transition in seconds</td></tr></table></td></tr>
<tr><td><code>temperature</code></td><td><em>object</em></td><td>Information about the temperature (if applicable)<table><tr><th>Property</th><th>Type</th><th>Description</th></tr><tr><td><code>tC</code></td><td><em>number or null</em></td><td>Temperature in Celsius (<code>null</code> if temperature is out of the measurement range)</td></tr><tr><td><code>tF</code></td><td><em>number or null</em></td><td>Temperature in Fahrenheit (<code>null</code> if temperature is out of the measurement range)</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>errors</code></td><td><em>array of type string</em></td><td>Error conditions occurred. May contain <code>overtemp</code>, (shown if at least one error is present)</td></tr>
</table>

Webhook Events

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

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

Examples

CCT.SetConfig example

CCT.SetConfig HTTP GET Request CCT.SetConfig Curl Request CCT.SetConfig Mos Request

```
http://192.168.33.1/rpc/CCT.SetConfig?id=0&config={"name":"CCT0"}
```

Response

```
null
```

CCT.GetConfig example

CCT.GetConfig HTTP GET Request

CCT.GetConfig Curl Request

CCT.GetConfig Mos Request

```
http://192.168.33.1/rpc/CCT.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,
      "ct": 6500
    }
  },
  "night_mode": {
    "enable": true,
    "brightness": 50,
    "ct": 4600,
    "active_between": [
      "21:45",
      "05:30"
    ]
  }
}
```

CCT.GetStatus example

CCT.GetStatus HTTP GET Request

CCT.GetStatus Curl Request

CCT.GetStatus Mos Request

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

Response

```
{
  "id": 0,
  "source": "timer",
  "output": false,
  "brightness": 50,
  "ct": 4600,
  "timer_started_at": 1626942399.36,
  "timer_duration": 60,
  "transition": {
    "target": {
      "output": true,
      "brightness": 100,
      "ct": 6500
    },
    "started_at": 1626942399.36,
    "duration": 20
  }
}
```

CCT.Set example

CCT.Set HTTP GET Request CCT.Set Curl Request CCT.Set Mos Request

```
http://192.168.33.1/rpc/CCT.Set?id=0&on=true&brightness=50&ct=4600&transition_duration=20
```

Response

```
null
```

CCT.Toggle example

CCT.Toggle HTTP GET Request CCT.Toggle Curl Request CCT.Toggle Mos Request

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

Response

```
null
```

CCT.DimUp example

CCT.DimUp HTTP GET Request CCT.DimUp Curl Request CCT.DimUp Mos Request

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

Response

```
null
```

CCT.DimDown example

CCT.DimDown HTTP GET Request CCT.DimDown Curl Request CCT.DimDown Mos Request

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

Response

```
null
```

CCT.DimStop example

CCT.DimStop HTTP GET Request CCT.DimStop Curl Request CCT.DimStop Mos Request

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

Response

```
null
```

HTTP Endpoint example

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

> *Example:*
>
> ```
> {
>   "ison": true,
>   "brightness": 100,
>   "temp": 4600,
>   "transition": 10,
>   "has_timer": true,
>   "timer_started_at": 258,
>   "timer_duration": 10,
>   "timer_remaining": 1.98,
>   "source": "http"
> }
> ```