PHILIPS

hue PERSONAL WIRELESS LIGHTING

How to **develop for Hue?**

Message Structure and Response

The hue API interface allows developers to interface with and make use of the functionality of the Philips hue System. Using this interface they can find information about the available devices in their local network, control these devices and much more.

The hue API consists of a set of commands that can be called over a REST web service. What this means is that every device, group etc in the Philips hue system is represented by a unique URL, which can be interacted with.

Typically the API commands require an authenticated whitelist user in the URL.

The API commands fall into one of 4 categories, depending on the HTTPS method used:

Method: GET

Used for: Reading specific data from the bridge.
Returns: JSON containing the requested resource.

Method: PUT

Used for: Modifying existing data on the bridge.
Returns: A list containing one item per parameter modified.
Example: `[{"success":{"/lights/1/state/hue":254}}]`

Method: POST

Used for: Adding new data to the bridge.
Returns: A list containing one item per resource created.
Example: `[{"success":{"id":"/schedules/7"}}]`

Method: DELETE

Used for: Deleting data from the bridge
Returns: A list containing one item per resource deleted.
Example: `[{"success":"/groups/1 deleted"}]`

Commands using `PUT` and `POST` methods will normally require a message body to be attached to the request. The message body must be formatted using JavaScript Object Notation (JSON). More details and examples for formatting the message body can be found in the documentation for each command.

Access types

Access types for attributes:

R	Returned by GET
RO	Resource might not support attribute
W	Required for PUT
WO	Optional for PUT
C	Required for POST
CO	Optional for POST

P Resource path. Can be separately addressed by HTTPS method
D Deprecated. Might be removed in the future

Bridge Information without a valid user

Generally, all Rest API commands will only retrieve bridge information if a valid authenticated Whitelist user is provided in the URL. There are some exceptions. Creating the username obviously, and getting a basic bridge configuration.

To get bridge information you can use:

`https://<IP ADDRESS>/api/<INVALID USER>/config`

or

`https://<IP ADDRESS>/api/config`

Depending if the user has an original Hue Bridge or a Bridge v2 (HomeKit) a typical response would be:

```
{
    "name": "Philips hue",
    "apiversion": "1.10.0",
    "swversion": "01028090",
    "mac": "00:17:88:20:01:0a",
    "bridgeid": "001788FFFE20010A",
    "replacesbridgeid": null,
    "factorynew": false,
    "modelid": "BSB001"
}
```

or on a Bridge 2 bridge that has had it's data transferred from an old bridge:

```
{
    "name": "My home bridge",
    "apiversion": "1.10.0",
    "swversion": "01028090",
    "mac": "00:17:88:20:0a:bc",
    "bridgeid": "001788FFFE200ABC",
    "replacesbridgeid": "001788FFFE0ABCDE",
    "factorynew": false,
    "modelid": "BSB002"
}
```

Note : The response may differ slightly depending on the API Version number. For example: replacesbridgeid , factorynew and modelid

[1] http://en.wikipedia.org/wiki/Representational_state_transfer