

# Domoticz Web App Site Control

---

## Domoticz Home Automation System

Robert W.B. Linn

12.01.2020

### **DISCLAIMER**

THIS DOCUMENT IS PROVIDED BY THE AUTHOR "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

# Table of Contents

Table of Contents .....	1
Purpose .....	2
Solution.....	3
Installation.....	4
Concept .....	5
Communication .....	5
HTTP API Requests .....	6
Subflow dom property .....	6
Example Requests.....	6
Configuration .....	9
Purpose.....	9
Flow .....	10
Thermostats.....	11
Purpose.....	11
Flow .....	11
Lights .....	12
Purpose.....	12
Flow .....	12
Weather .....	13
Purpose.....	13
Flow .....	13
Postbox .....	14
Purpose.....	14
Flow .....	14
Information .....	15
Purpose.....	15
Flow .....	15

# Purpose

To control the [Domoticz](#) Home Automation System, running on a Raspberry Pi 3B+, via a browser-based application (WebUI, built with Node-RED).

- Easy-to-use User Interface accessible from any device capable running a browser (smartphones, tablets, PC, SBC, TV)
- Control heating (HomematicIP) & lights (Philips Hue)
- Information only for weather data, key dates and status postbox

The goal has been to develop a prototype first, i.e. workout the concept with initial functionalitz, to be able to enhance or modify the application further.

This solution is also ment as a trigger for ideas or as a suggestion building an alternate GUI (WebUI) to access the Domoticz Homeautomation System.

The User Interface is translated to the German language.

## Notes

- In the mean time, after having used the application for a while, it has become the main application to access the Home Automation system from various devices.
- The application is rather tailer made.  
If want to use for own purposes, it will need a deep dive into the solution, which is based on Node-RED and JavaScript (used in the many function nodes).  
The flows are commented, so recommend to checkout.  
In this document, the flows are described more at higher level.
- It is not the intention to explain Node-RED in great detail.  
Visit the Node-RED site to explore more.  
The author is also on a steep learning curve on how to develop Node-RED solutions.  
This means, there might be better ways to accomplish a function, but so far this solution is working fine.
- The solution is being developed further but this documentation might not be up-to-date. Check out the comments in the flows.

# Solution

As mentioned, the solution is based on [Node-RED](#) which is running on the Domoticz Home Automation System (on both production and test).

The goal is to use the standard Node-RED nodes, except for the User Interface (WebUI) which is based upon the [Node-RED Dashboard Module](#).

In the WebUI there is a tab for each function with one or more groups:

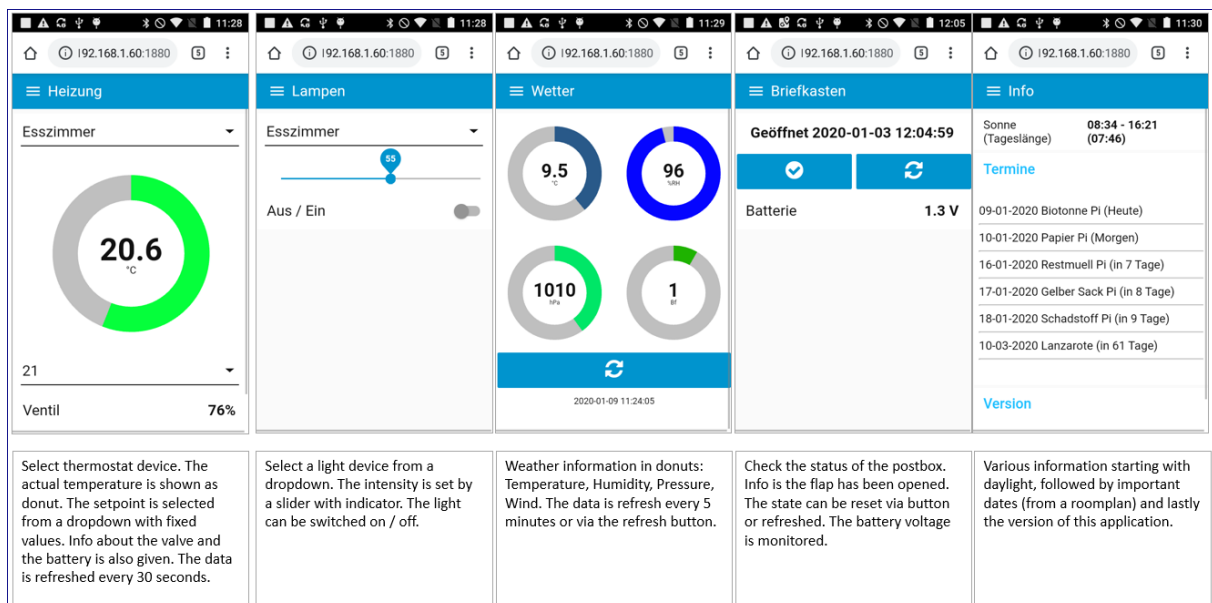
- Heizung (Heating)
- Lampen (Lights)
- Wetter (Weather)
- Briefkasten (Postbox)
- Info (Information) with groups Sonne (Daylight) Termine (Dates), Version (Version)

To access this web app, use the IP address of the Domoticz system with Node-RED port and parameter /ui.

Example: <http://192.168.NNN.NNN:1880/ui>

## Screenshots

The application is accessed from a browser on an Android phone.



## Software Versions

- Linux: 4.19.66-v7+ #1253
- Domoticz: 4.10717
- Node-RED: 1.0.3
- Node-RED Dashboard: 2.19.3

# Installation

In Node-RED import the various flow files as new flows (each flow has its own tab).  
The file names start with *wasc\** and have the extension *.flow*.

wasc-configuration	flow
wasc-dom-property	subflow
wasc-info	flow
wasc-lights	flow
wasc-postbox	flow
wasc-thermostats	flow
wasc-weather	flow

See the comment node of each tab.

## Screenshot

It shows the Node-RED tabs, the tabs & groups and the menu.

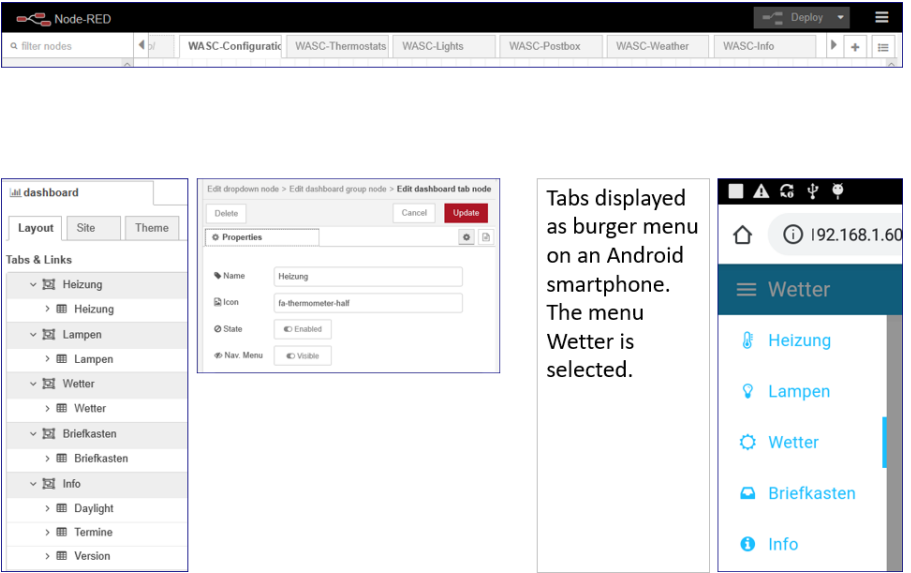
Tabs defined in Node-RED.  
Each function has its own tab with groups, except for the Configuration.

Tabs & Groups for the dashboard.

The menu & group order can be set here.

For a tab, special icons are set – Font Awesome icon (see example Tab Heizung).

Tabs displayed as burger menu on an Android smartphone. The menu Wetter is selected.



The screenshot displays the Node-RED web interface. At the top, a row of tabs is visible: 'WASC-Configurat...', 'WASC-Thermostats', 'WASC-Lights', 'WASC-Postbox', 'WASC-Weather', and 'WASC-Info'. Below this, the 'dashboard' tab is active, showing a 'Layout' panel on the left with a tree view of 'Tabs & Links'. The tree includes categories like 'Heizung', 'Lampen', 'Wetter', 'Briefkasten', and 'Info', each with sub-items. A 'Properties' panel on the right shows settings for a selected item, including 'Name', 'Icon' (fa-thermometer-half), 'State' (Enabled), and 'Nav Menu' (Visible). On the far right, a mobile device mockup shows the dashboard as a burger menu with the 'Wetter' tab selected.

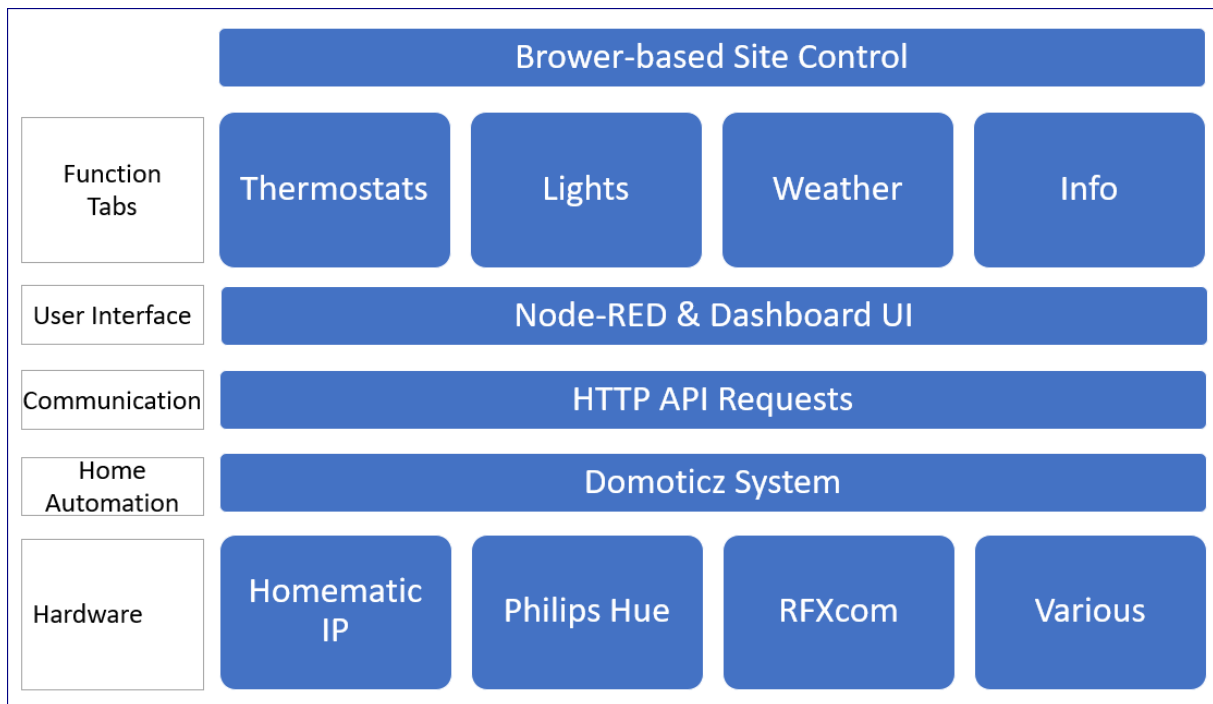
The various flows require to change the idx of the many devices used.

# Concept

There are several concepts developed depending the functionality.  
Each functionality has its own flow defined in a Node-RED tab.  
The flows are making use of global and flow context values.

## Communication

The communication between the Node-RED flows and Domoticz is via HTTP API requests (API-calls) and handling the JSON response accordingly.



### IMPORTANT

Although possible to use MQTT, decided to stick to the Domoticz recommended way by using HTTP and JSON as it gives easy access to data properties.

The concept used in various flows:

- configure the idx of the devices used as JSON string (function node)
- select a device (ui\_dropdown node)
- set the flow context for the device selected (function node)
- set the flow context properties (function node)
- set the URL for the HTTP API-call for the selected device (function node)
- send HTTP API-call (http request node)
- convert HTTP JSON response string to JavaScript object (json)
- use the message payload property of the device (various nodes, i.e. ui, function)

Other nodes are also used esp. the switch and change node.

If regular data updates are required, this is handled by the inject node interval property.

# HTTP API Requests

Sending an HTTP API request (message url defined in a function node and send by the HTTP request node) is always followed by a response which is handled depending functionality.

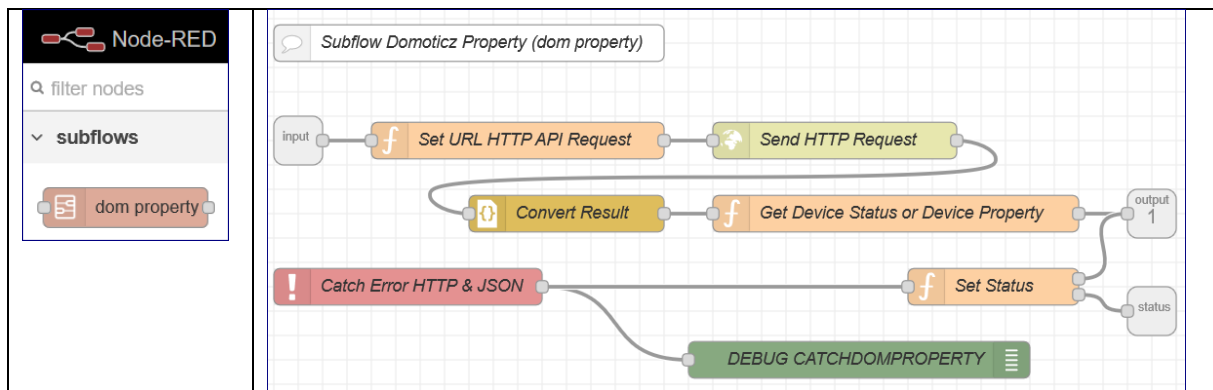
Most of the flows use a GET request, means pulling for data either for a single device, for all devices or for the devices assigned to a roomplan.

The HTTP API response (JSON string) is converted to a JavaScript object to easy access properties.

Because this process is used across various flows, a subflow has been defined to handle HTTP API requests.

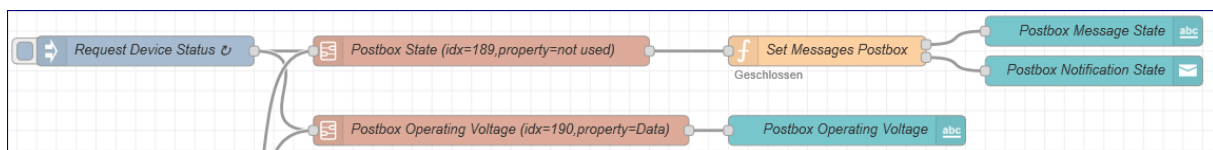
## Subflow dom property

Get a property or all properties for a specific device or all device properties.



See the subflow node description (file wasc-dom-property.subflow).

Example usage for Function Postbox



## Example Requests

The example requests are used to analyse the result JSON string key:value pairs.

### Single Device

Get the temperature property of a single device.

**Function Node** - define the msg.url property

```
msg.url = global.get("domip") + "?type=devices&rid=" + flow.get("temperature");
return msg;
```

The msg.url is used as the input for the http request node.

It uses the context values "domip" (with a value like `http://127.0.0.1:8080/json.htm`) and "temperature" (which contains the idx of the temperature device, i.e. 196).

### HTTP request status single device

```
http://127.0.0.1:8080/json.htm?type=devices&rid=196
```

#### Note

As mentioned, use these kind of HTTP requests for tests.

Change the IP accordingly when submitting from another device as the Domoticz system.

The http request node sends the request and the output JSON string is converted to a JavaScript object.

### HTTP Response JSON String (snippet)

```
{
  ...
  "ServerTime" : "2020-01-04 10:36:06",
  "app_version" : "4.10717",
  "result" : [
    {
      ...
      "Data" : "22.3 C",
      "HardwareType" : "homematicIP Radiator Thermostat (HmIP-eTRV)",
      "LastUpdate" : "2020-01-04 09:56:27",
      "Name" : "Hzg MakeLab Temperatur",
      "PlanID" : "15",
      "SubType" : "LaCrosse TX3",
      "Temp" : 22.300000000000001,
      "Type" : "Temp",
      "idx" : "196",
      "trend" : 0
    }
  ],
  "status" : "OK",
  "title" : "Devices"
}
```

From the JSON response string, the device data is in the array "result".

For a single device, the array as one entry. The properties of the array entry are accessed by using index 0.

### Example get property "Temp"

```
msg.payload.result[0].Temp
```

## Multiple Devices

If the JSON response string contains multiple devices, then the array "result" size is greater 1 and can be accessed using index 0 to array length - 1.

### HTTP request status all favorite devices

```
http://127.0.0.1:8080/json.htm?type=devices&used=true&filter=all&favorite=1
```

### HTTP response JSON String (snippet)

```
{
  "ServerTime" : "2020-01-04 10:55:36",
```



```
"app_version" : "4.10717",
"result" : [
  {
    "Data" : "Coffee Off 10:30",
    "Name" : "Alarm Meldungen",
    "idx" : "55"
  },
  {
    "Data" : "04:53 - 21:55 (17:02)",
    "Name" : "Sonne (Tageslänge)",
    "idx" : "121"
  },
  {
    "Data" : "6.2 C, 96 %",
    "Name" : "Aussen Temperatur",
    "idx" : "120",
  }
],
"status" : "OK",
"title" : "Devices"
}
```

The array "result" contains 3 entries and can be accessed using index 0 to 2.

In JavaScript the `forEach()` method can be used to access properties or extract certain devices by its `idx`.

Example selecting devices for a specific planid (roomplan) from the full list of devices. The roomplan `idx` is defined in a flow context value.

The array "result" is assigned to a local variable as a JavaScript object from which the properties are selected.

```
// Get the all devices result array property
var allDevices = msg.payload.result;
// Show number of device entries from the full list
var allDevicesCnt = allDevices.length;
// node.warn("All devices found="+allDevicesCnt);
// Get the planid to search for
var searchPlanID = flow.get("planid");
// Create new array holding the idx of the devices found
var selectedDevices = [];
// Create the array of selected devices
function getDevices(item, index) {
  // check if the idx is in the searchDevices array
  if (searchPlanID == item.PlanID){
    data = item.Data + ": " + item.Name;
    // add the device info
    selectedDevices.push(data);
    // node.warn(data);
  }
}
allDevices.forEach(getDevices);
```

This concept is used throughout the various flows.

# Configuration

## Purpose

To set global context values accessed by the various flows.

Set the global context injected first (property “inject once after 0.1 seconds”) at start of the WASC flows.

Every other inject node must start after the configuration has been injected, by setting the property “inject once after” to 0.2 seconds or higher.

### **global.domip** - HTTP IP address

Sets the IP address of the Domoticz system.

The web app production flows are by default defined on the production system, means Node-RED can access Domoticz by using localhost.

Example\_: `http://127.0.0.1:8080/json.htm`

The required API-calls are added where required by several nodes.

To explore flows, an other device can be used. Change the IP from localhost to the IP of the Domoticz system.

### **global.debug** - 0 | 1

In several function flows, the `node.warn()` function is defined.

Setting the debug value to 1 will log content to the Node-RED debug tab.

### **global.version** - Version string

The version information is displayed in tab Info group Version.

## Style

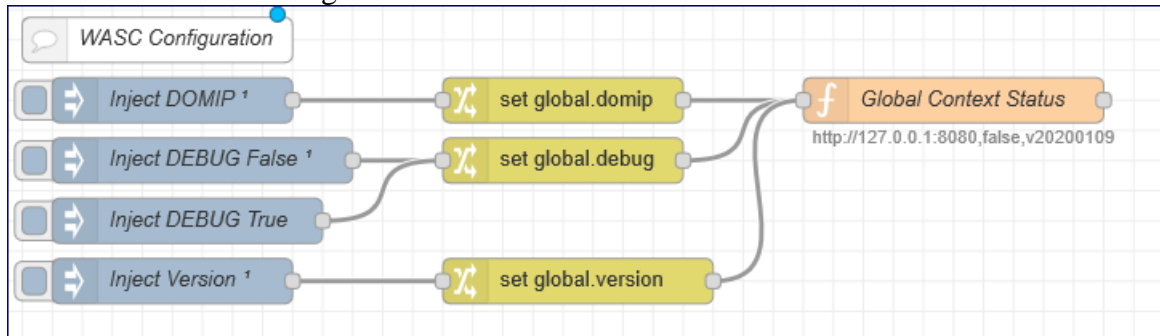
Each tab has its own style setting in a `ui_template`.

### Sample for Tab Heizung

```
<style>
#Heizung_Heizung_cards {
  font-size: 20px !important;
  background-color: "#FF0000" !important;
}
</style>
```

# Flow

Source code: wasc-configuration.flow



## Notes

Web App Site Control (WASC)

Web based GUI, using Node-RED, to control a Domoticz Home Automation System with connected hardware.

## Configuration

### Global Context

domip = Set the IP address of the Domoticz system

debug = Set debug on (true) or off (false)

version = Set the date YYYYMMDD of the last change

The function node is only used to show the status of the global context.

Change values in the Inject nodes.

The debug flag is set to false. Click on the Inject DEBUG True node to turn debug on. Then refresh the various flows or wait for a refresh.

## Style

Each tab has its own style setting in a ui\_template.

Sample for Tab Heizung

```
<style>
#Heizung_Heizung_cards {
  font-size: 20px !important;
  background-color: "#FF0000" !important;
}
</style>
```

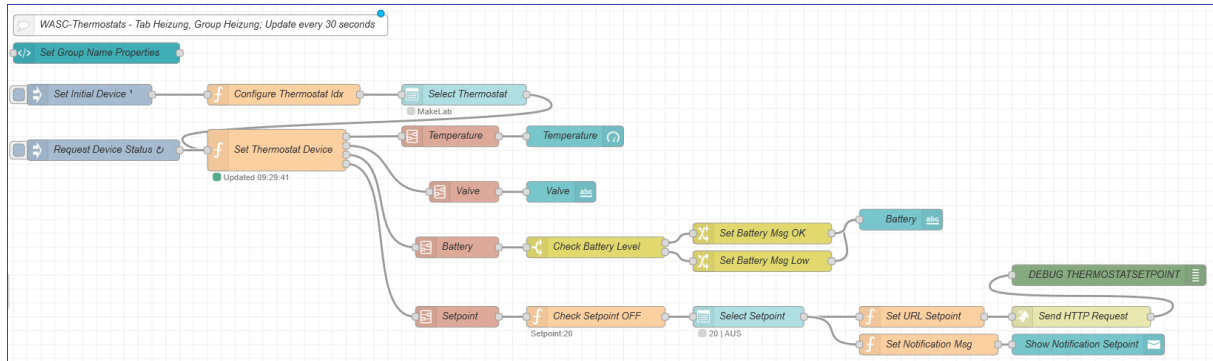
# Thermostats

## Purpose

To change the setpoint and monitor the room temperature of an HomematicIP thermostat.

## Flow

Source code: wasc-thermostats.flow



### Notes

WASC-Thermostats - Tab Heizung, Group Heizung; Update every 30 seconds

### Control the HomematicIP thermostat devices - brief description

The devices are created by the Domoticz Plugin homematicIP Radiator Thermostat (HmIP-eTRV). Each thermostat has its own hardware with the devices (Type,SubType): Setpoint (Thermostat,Setpoint),Temperature (Temp,LaCrosse TX3),Battery (General,Alert),Valve (General,Percentage).

The values (properties from the result of the HTTP API request) are used to display in dashboard ui nodes and to control the setpoint.

The function node "Configure ..." defines the idx of the thermostats and build the ui\_dropdown select options. Add or change devices in this node.

Each thermostat has a number of devices with its own idx, i.e. Temperature, Valve, Battery, Setpoint. Each of these idx are used in dedicated flow context used to obtain the properties. After device selection, from the ui\_dropdown, the flow context for the selected device are set, i.e. name, temperature, valve, battery, setpoint. The required device property is "get" via subflow "dom property" and assigned to the specific ui widget.

To set a setpoint, the ui\_dropdown node is used which has predefined values (i.e. 17,18,19 etc and AUS (OFF)). A HTTP API request is triggered to Domoticz to action.

The battery information, displayed in a ui\_text node, is set between OK and Low.

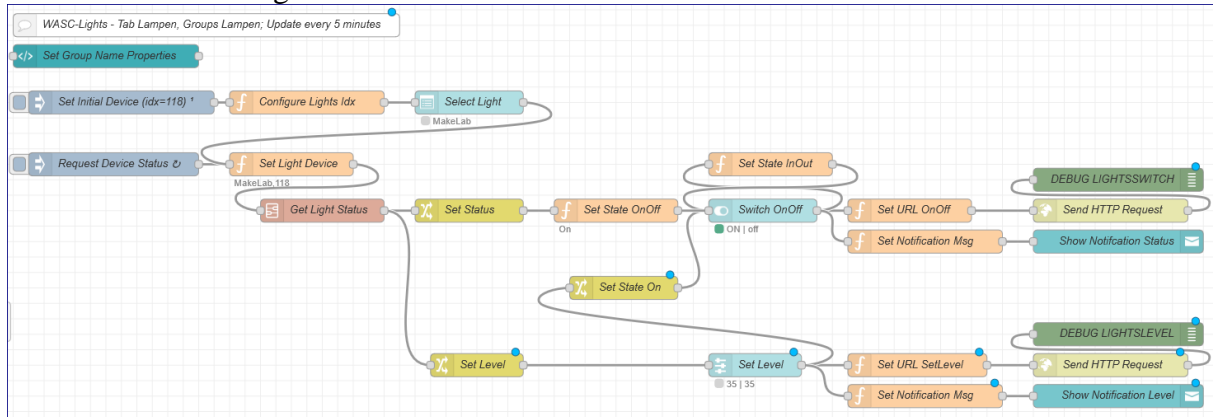
# Lights

## Purpose

To set the level (brightness) or switch on/off of a Philips Hue light.

## Flow

Source code: wasc-lights.flow



### Notes

WASC-Lights - Tab Lampen, Groups Lampen; Update every 5 minutes

### Control the Philips Hue lights - brief description

The devices are created by the hardware Philips Hue Bridge. If a Hue device is added to the Hue App, Domoticz recognizes and creates the new device as Type;SubType Light;Switch Switch-Color or Switch;WW.

The function node "Configure ..." defines the idx of the lights and builds the ui\_dropdown select options. For each light the name and idx is used.  
Add or change devices in this function node.

The idx is required to get the light status On/Off and the level (0-100%).

To set the light level, a ui\_slider is used which sets the new level when it is released.

The light can be switched on/off using a ui\_switch.

If the light is off and the slider is moved, the switch is turned on.

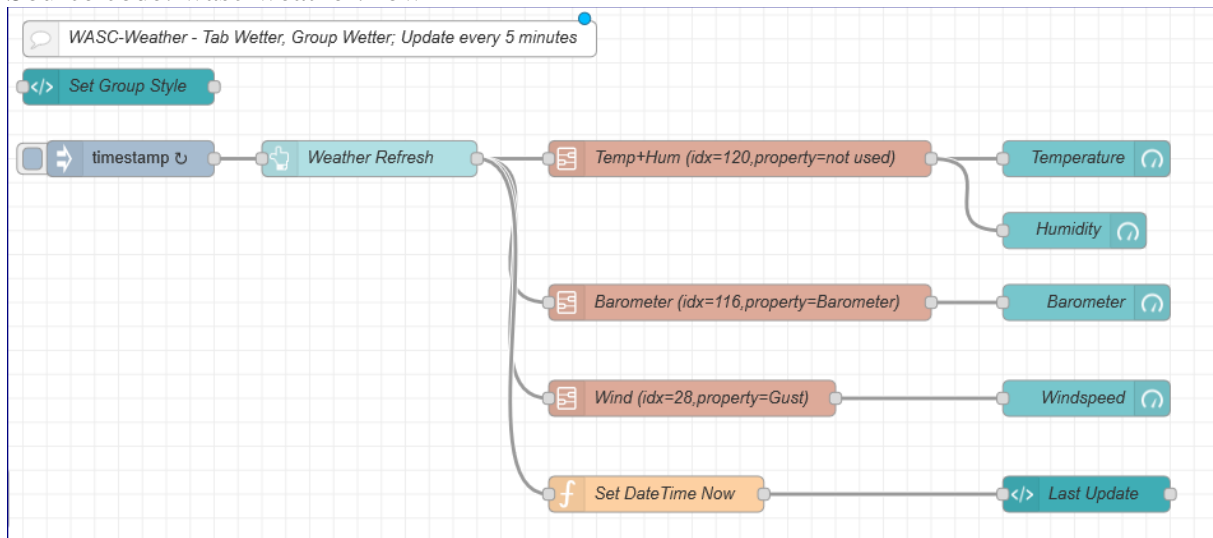
# Weather

## Purpose

To monitor weather information Temperature (°C), Humidity (%RH), Airpressure (hPa) and Windspeed (Bf).

## Flow

Source code: wasc-weather.flow



### Notes

WASC-Weather - Tab Wetter, Group Wetter; Update every 5 minutes

### Display Weather information - brief description

The weather data is displayed using ui\_gauges, type donut.

Each device uses the subflow "dom property" to get one or more properties:  
Temperature=Temp (°C), Humidity=Humidity (%RH), Airpressure=Barometer (hPa),  
Windspeed=Gust (Bf).

The refresh button obtains the latest information.

The refresh date & time (Last Update) is displayed in an ui\_template node.

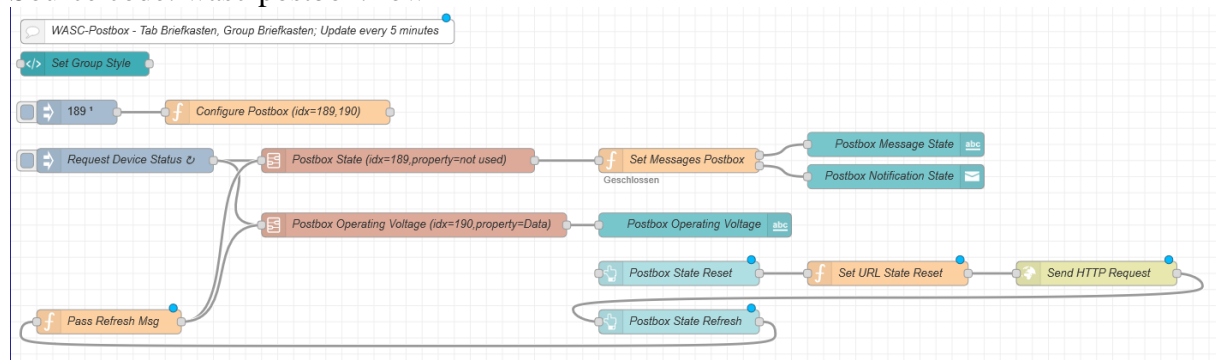
# Postbox

## Purpose

To monitor the postbox state. If the flap is opened a notification is triggered.

## Flow

Source code: wasc-postbox.flow



### Notes

WASC-Postbox - Tab Briefkasten, Group Briefkasten; Update every 5 minutes

### Control the postbox - brief description

The function node "Configure ..." defines the idx of the devices:

(idx- name,type, subtype)

189 - Status, Light/Switch, Switch

190 - Batterie, General, Voltage

If the postbox flap is opened, the ui\_text node shows state opened with date&time.

The state can be refreshed or reset after checking the postbox. If reset, the ui\_text states closed ("geschlossen").

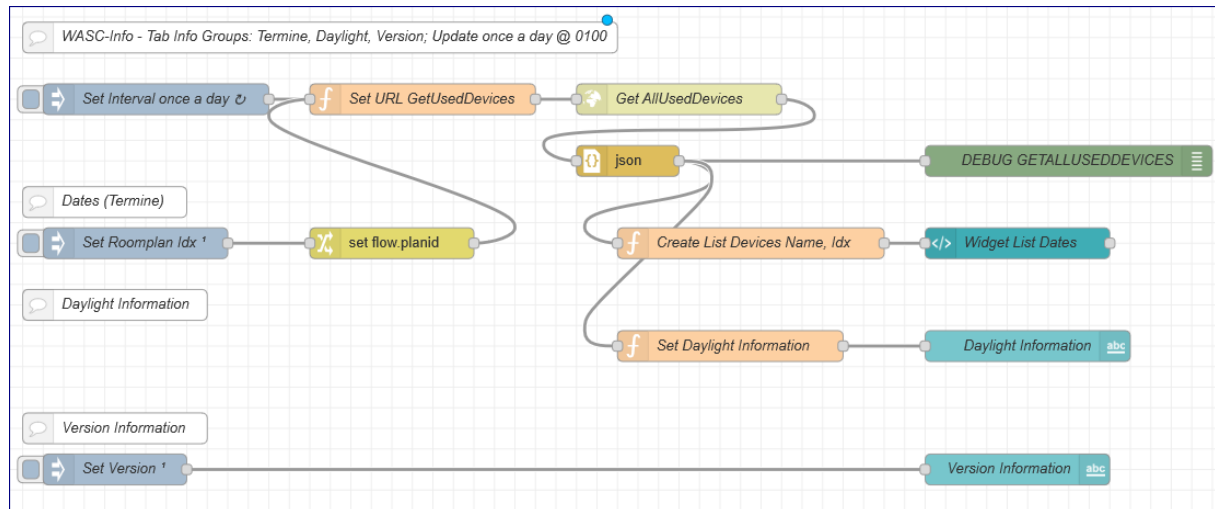
# Information

## Purpose

To display information about daylight, dates and the application version.

## Flow

Source code: wasc-info.flow



### Notes

WASC-Info - Tab Info Groups: Termine, Daylight, Version; Update once a day @ 0100

Display various information- brief description.

Daylight information from the general information of the HTTP API response from GetAllUsedDevices. No Domoticz device used.

Dates shows a list in a ui\_template with data from a roomplan (idx=4).

Version information is shown in a ui\_text with data taken from global context version (global.version).