

# Domoticz Homematic IP

## Workbook

---

Learn | Build | Share | Reference

**Robert W.B. Linn**

**07.02.2023**

### **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.....	5
Objectives.....	5
Remarks.....	5
Functions.....	6
Explore .....	6
Hard-/Software Versions.....	7
Abbreviations.....	8
Credits.....	8
Setup.....	9
Homematic .....	9
Remote Homematic Script API .....	10
Addon XML-API.....	10
Addon CCU-Jack.....	10
Addon CUxD .....	11
Addon Script-Parser .....	11
RaspberryMatic .....	12
Domoticz .....	13
Node-RED .....	13
Communication .....	14
Concept.....	14
Examples .....	14
Domoticz Trigger State Change Homematic Device .....	15
Homematic Trigger State Change Domoticz Switch.....	17
Functions.....	18
Alert Indicator (HmIP-MOD-OC8) .....	18
Purpose .....	18
Solution .....	18
Homematic Configuration .....	20
Domoticz Configuration .....	21
Arduino .....	24
Enhancement Ideas.....	26
Battery Check (HmIP All Devices) .....	27
Purpose .....	27
Solution .....	27
Solution Domoticz.....	28
Solution Homematic.....	30
Coffee Machine Monitor (HmIP-PSM).....	33

Purpose .....	33
Solution .....	33
Homematic Configuration .....	33
Domoticz Configuration .....	35
Duty-Cycle Monitor (HmIP-CCU3) .....	38
Purpose .....	38
Solution .....	38
Solution Device Object HmIP-CCU3 (In Use) .....	39
Solution System Command (Old).....	42
E-Paper Status Display (HmIP-WRCD) .....	47
Purpose .....	47
Solution .....	47
Homematic Configuration .....	48
Domoticz Configuration .....	54
Reference Display Configuration .....	65
Reference Display Set REST-API.....	68
Garage Door Monitor (HmIP-SWDM) .....	70
Purpose .....	70
Solution .....	70
Homematic Script .....	72
Automation Event .....	73
Heating Unit Gas Usage (HmIP-FCI1) .....	74
Purpose .....	74
Solution .....	74
Homematic Configuration .....	76
Domoticz Configuration .....	82
Duty-Cycle.....	84
Battery.....	85
Heating Unit Temperature (HmIP-STE2-PCB) .....	86
Purpose .....	86
Solution .....	86
Homematic Configuration .....	88
Domoticz Configuration .....	89
Solution PULL.....	93
Solution PUSH.....	96
Enhancements.....	98
Pluggable Switch and Meter (HmIP-PSM) .....	99
Purpose .....	99
Solution .....	99
Homematic Configuration .....	99

Domoticz Configuration .....	100
PSM Switch .....	101
PSM Energy .....	105
Postbox Notifier (HmIP-SWDO).....	106
Purpose .....	106
Solution .....	106
Homematic Configuration .....	108
Domoticz Configuration .....	110
Radiator Thermostats (HmIP-eTRV).....	112
Purpose .....	112
Solution Thermostat SetPoint Device (In Use) .....	113
Solution Selector Switch & Thermostat (Explored).....	122
Solution Plugin (Explored) .....	126
Solution Thermostat SetPoint Device Homematic Script (Explored) .....	133
Thermostat State Custom Page.....	143
Remote Control (HmIP-RC8) .....	147
Purpose .....	147
Solution .....	147
Homematic Configuration .....	148
Solution Custom Event (In Use).....	149
Solution Selector Switch (Explored).....	151
Explore .....	153
Purpose .....	153
Homematic Scripting .....	154
Script Development .....	154
Development Hints .....	155
Script Examples.....	157
Domoticz Examples .....	165
Remote Homematic Script API .....	177
Information .....	177
Settings .....	177
CLI Examples .....	178
Domoticz Examples .....	180
Node-RED Examples.....	190
Python Examples.....	201
Homematic JSON API .....	202
Information .....	202
Method Overview.....	202
HTTP Request Examples .....	203
Domoticz Examples .....	206

CLI Examples .....	213
Node-RED Examples.....	215
Python Examples.....	217
Addon XML-API.....	221
Information .....	221
CGI Scripts .....	221
Domoticz Examples .....	228
Experiments.....	246
Node-RED Examples.....	250
Plugin Considerations.....	262
Addon CCU-Jack.....	263
Information .....	263
MQTT-API .....	286
Addon CUxD .....	307
Information .....	307
Update Domoticz Text Device .....	309
Appendix Tools.....	311
Homematic Statelist.....	311
Purpose .....	311
Solution .....	311
Node-RED .....	312

# Purpose

The purpose is to build a **Smart Home System**, based on the [Domoticz](#) Home Automation System including [Homematic IP](#) devices connected to the Homematic Smart Home Central Control Unit [CCU3](#).

Homematic is a registered trademark of [eQ-3 AG](#).

## Why this workbook?

This workbook is an addendum to the [Domoticz Homeautomation Workbook](#).

Whilst building a Domoticz Home Automation System, started to explore Homematic IP with [RaspberryMatic](#) - free open source-based "Homematic CCU", running on a Raspberry Pi 3B+.

In the meantime

- Replaced RaspberryMatic by a Homematic Smart Home Central Control Unit CCU3.  
*Some screenshots show the RaspberryMatic WebUI instead of Homematic WebUI.*
- Integrated several Homematic IP devices - the core of the Smart Home System.
- In progress integrating more Homematic IP devices.

Therefor decided extracting all Homematic IP related information from the Domoticz Homeautomation Workbook to this specific workbook = bundling the information around Homematic IP.

Another reason is that the Domoticz Homeautomation Workbook had become a rather large document – not easy to maintain.

# Objectives

- Integrate the Homematic Smart Home Central Control Unit CCU3 into Domoticz.
- Explore controlling Homematic IP devices.
- Learn Homematic scripting.
- Explore & integrate Homematic addons and JSON-API.
- Develop solutions (called functions) and tools.
- Having fun developing & share experience.
- Use this workbook as a supplemental reference.

# Remarks

- **The information shared is for personal use only.  
Use at your own risk (see LICENSE).**
- This is a working document = concept changes & new idea's whilst progressing.
- There might be better solutions = changes depending on learning curve.
- To-Do actions are tagged with //TODO and captured in the file TODO.md.
- Names are (in cases) in German as the system is used in Germany.
- Domoticz Automation events with dzVents scripts.
- Domoticz Hardware Plugins developed with Python.
- Domoticz Custom Pages developed with JavaScript.
- Node-RED is explored as a tool or Dashboard-UI.
- Some functions are described in the Domoticz Homeautomation Workbook, like the Alert Message.
- Hard- and Software versions are subject to change.

- For latest source codes check the src folder of the GitHub repository.

## Functions

The shared solutions are called **functions**, which have a specific purpose.

The Homematic IP device(s) used for a function is listed in brackets.

A function can also be a framework used by several other functions.

For each of the devices, started first to explore on how to integrate into Domoticz, followed by a function.

Some of the functions created (used Homematic IP devices in brackets).

- Alert Indicator (HmIP-MOD-OC8)
- Battery Check (All HmIP-Devices)
- Coffee Machine Monitor (HmIP-PSM)
- E-Paper Status Display (HmIP-WRCD)
- Garage Door Monitor (HmIP-SWDM)
- Heating Unit Gas Usage (HmIP-FCI1)
- Heating Unit Temperature (HmIP-STE2-PCB)
- Duty-Cycle Monitor (HmIP-CCU3 Device Object)
- Pluggable Switch and Meter (HmIP-PSM)
- Postbox Notifier (HmIP-SWDO)
- Radiator Thermostats (HmIP-eTRV-B/eTRV-2)
- Remote Control (HmIP-RC8)
- Tool: Homematic Statelist devices and datapoints (XML-API CCU3 Addon)

## Explore

The chapter [Explore](#) describes in various topics how to use Homematic IP devices, programs & scripting, CCU addons, Domoticz communication, Domoticz Automation Events using dzVents, Node-RED and more.

# Hard-/Software Versions

<b>Component</b>	<b>Software</b>	<b>Hardware</b>	<b>Remarks</b>
<b>Homematic IP</b>			
CCU3 Firmware	3.67.10		
Addon XML-API	1.22		
Addon CCU-Jack	2.7		
Addon CUx-Daemon	2.10.1		
Addon Script-Parser	1.11		
<b>Domoticz Production</b>			
Raspberry Pi OS	5.10.103-v7+	3B+	
Domoticz Stable	2022.2		
Node-RED	3.0.2		
Python	3.7.3		
GCC	8.3.0		
Java OpenJDK	11.0.11+9		
Mosquitto MQTT	1.5.7		
Lua	5.1.5		
<b>RaspberryMatic</b>			
CCU3	3.51.6.20200621		Not used anymore
<b>Domoticz Development</b>			
Raspberry Pi OS	5.10.103-v7+	4B 2GB	
Domoticz BETA	2022.2 #NNNNN		Port 8080
Node-RED	3.0.2		Port 1880
Python	3.7.3		Develop scripts or Domoticz plugins & tools
GCC	8.3.0		Develop tools
Java OpenJDK	11.0.11+9		Used by B4J
Mosquitto MQTT	1.5.7		Send messages Homematic <> Domoticz and Node-RED. Port 1883
Lua	5.1.5		Create & test Domoticz scripts
B4J	9.8		Create desktop & console tools
B4R	3.9		Create microcontroller solutions Arduino & ESP
ESP Easy	mega-20210503		Create microcontroller solutions ESP8266 & ESP32

*Be aware: Hard- and Software versions are subject to change.*

# Abbreviations

CCU	Homematic Smart Home Central Control Unit CCU3
ccu-ip:port	Homematic CCU3 IP address and port
CCU-Jack	Homematic addon CCU-Jack
CLI	Terminal Command Line Interface
CuxD	Homematic addon CUx-Daemon
domoticz-ip:port	Domoticz system IP address and port
~domoticz	Path to the Domoticz home directory
dzVents	Domoticz Easy Events Mainly used for automation events using dzVents scripts (Lua)
Hm, HmIP	Homematic, Homematic IP
NR	Node-RED
RaspMatic	RaspberryMatic running a CCU3
RHMS	Remote Homematic Script API
RPi	Raspberry Pi
UI	User Interface or Graphical User Interface
XML-API	Homematic addon XML-API
XML-RPC	Homematic XML-RPC interface

# Credits

A big thanks, to the developers of Domoticz, Homematic, RaspberryMatic, Homematic addons and to all sharing information about Domoticz and/or Homematic.

Without these, it would not be possible to build this project and write the workbook.

# Setup

## Homematic

Installed the **Homematic Smart Home Control Unit CCU3** according Homematic [Guidelines](#).

The left panel shows the "CCU maintenance" screen with the following details:

- CCU software: Current software version: 3.57.5, Available software version: 3.57.5.
- Perform software update**: Load directly to CCU and install.
- Alternative procedure:**
  - Step 1: Download new software [Download](#)
  - Step 2: Select downloaded software [Choose File](#) No file chosen
  - Step 3: Upload software to CCU [Upload](#)
  - Step 4: Start update

The right panel shows the "HomeMatic Admin" interface with the following information:

- Admin homematic Home page
- Alarm messages (0), Service messages (0)
- Time: 10:59, Date: 21.05.2021, Sunrise: 05:09, Sunset: 21:55, Current firmware version: 3.57.5
- Duty Cycle CCU3: 2%

- Updates are performed by direct loading to the CCU.
- Experienced that updates can take up-to several minutes including reboot of the CCU.

Enabled CCU access for

- Remote Homematic Script API** – Run Homematic scripts remotely (tclregae.exe).

Installed CCU additional software (addons):

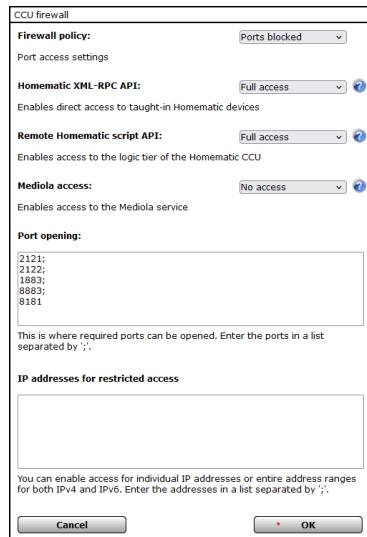
- XML-API** - read/write device datapoints using XML & URL parameter.
- CCU-Jack** - alternative to the XML-API - read/write JSON instead XML.
- CUx-Daemon** - HTTP API/JSON requests to Domoticz.
- Script-Parser** – Develop & test Homematic scripts prior implementing into programs.

Additional software for CCU			
XML-API	Installed version: 1.20 Available version: 1.20 <a href="#">Download</a> <a href="#">Uninstall</a> <a href="#">Set</a>	<b>XML-API CCU Addon</b> <a href="https://github.com/hobbyquaker/XML-API">https://github.com/hobbyquaker/XML-API</a>	
CCU-Jack	Installed version: 1.1.1 Available version: 1.1.1 <a href="#">Download</a> <a href="#">Restart</a> <a href="#">Uninstall</a> <a href="#">Set</a>	<b>CCU-Jack Add-On</b> (c) 2019-2021 info@ccu-historian.de <a href="https://github.com/mdzio/ccu-jack">https://github.com/mdzio/ccu-jack</a>	
CUx-Daemon	Installed version: 2.6 Available version: 2.6 <a href="#">Download</a> <a href="#">Restart</a> <a href="#">Uninstall</a> <a href="#">Set</a>	<b>CUx-Daemon 2.6</b> 	
Script-Parser	Installed version: 1.9 Available version: 1.9 <a href="#">Download</a> <a href="#">Uninstall</a> <a href="#">Set</a>	<b>Script-Parser CCU Addon</b> Erweiterter ReGa-Skript Parser/Tool (c) 2015-2020 Andre Lüfin, Jens Maus <a href="https://github.com/homematic-community/scriptparser">https://github.com/homematic-community/scriptparser</a>	
Install / update additional software	Select additional software: <a href="#">Choose File</a> No file chosen <a href="#">Install</a>	<b>Please note:</b> Additional software installed by the user can lead to unexpected results, data loss or even system instability. <b>eQ-3 AG does not assume any liability for additional software installed by the user.</b> To finish installation, the CCU will be restarted automatically.	

# Remote Homematic Script API

To enable [Remote Homematic Script API](#), change the Homematic settings:

- Homematic WebUI > Settings > Control Panel > Firewall > Set Full Access for Remote Script API (Enables access to the logic tier of the Homematic CCU).
- The port used is 8181.



## Addon XML-API

The addon [XML-API](#) (read/write device datapoints using CGI scripts) has been initially used. The HTTP response is an XML tree, which can be parsed with, for example the dzVents function “domoticz\_applyXPath()”.

The advantage of this addon, is that all commands can be handled as URL GET or PUT requests. This makes the Domoticz implementation easy in for example dzVents scripts (function “openURL”) or device actions.

## Addon CCU-Jack

The addon [CCU-Jack](#) (read/write JSON format) is an alternative to the XML-API.

There are two APIs available, [REST-API](#) and [MQTT-API](#).

The HTTP REST-API response is JSON data and can be parsed for example by Domoticz dzVents when received.

The MQTT-API messages require an intermediate solution (like Node-RED or Python script) to forward received Homematic message to Domoticz.

The CCU-Jack documentation is in German.

In this workbook both addons are used  
Have not finally made up my mind (yet) for using CCU-Jack only and replace XML-API = still got used to the XML-API (in place since few years).  
In the meantime, explored another option Remote Homematic Script API which does not require both addons.

## Addon CUxD

To handle HTTP API/JSON request to Domoticz, the [CUx-Daemon](#) addon is required.  
So far the Homematic scripts developed, use a single channel to place HTTP requests via  
“wget” mainly to Domoticz.  
//TODO – explore advanced usage of the CuxD.

## Addon Script-Parser

The [Script-Parser](#) supports developing & testing scripts (“ReGa Script Parser/Tool”) before these are implemented in programs.  
This tool is easier to use then the default Homematic Test script solution.

### **Caution:**

This addon enables everyone who has access to your network to execute commands on the CCU without authentication.

# RaspberryMatic

Before using the Homematic Smart Home Central Control Unit CCU3, a [RaspberryMatic](#) operating system has been setup and used for exploring Homematic IP functionality.

The solution runs the RaspberryMatic CCU3 on a Raspberry Pi 3B+ with an RPI-RF-MOD GPIO Radio Module HAT.

The Raspberry Pi 3B+ has been setup according [these](#) guidelines.  
RaspberryMatic version 3.51.6.20200621 is used (check out for newer versions [here](#)).

Installed additional software via the RaspberryMatic WebUI > Home page > Settings > Control panel > Additional Software - same as for the Homematic CCU3.

Some of the screenshots still show the RaspberryMatic WebUI.

Additional software for RaspMatic			
System-Update	Installed version: 1.13.12  Available version: 1.13.13	<a href="#">Download</a>  <a href="#">Uninstall</a> <a href="#">Set</a>	<b>rmupdate addon</b> <a href="https://github.com/j-a-n/raspberrymatic-addon-rmupdate">https://github.com/j-a-n/raspberrymatic-addon-rmupdate</a>
XML-API	Installed version: 1.20  Available version: 1.20	<a href="#">Download</a>  <a href="#">Uninstall</a> <a href="#">Set</a>	<b>XML-API CCU Addon</b> <a href="https://github.com/hobbyquaker/XML-API">https://github.com/hobbyquaker/XML-API</a>
CUX-Daemon	Installed version: 2.3.4  Available version: 2.3.4	<a href="#">Download</a>  <a href="#">Restart</a> <a href="#">Uninstall</a> <a href="#">Set</a>	<b>CUX-Daemon 2.3.4</b> 

The previous mentioned addons CCU-Jack & Script-Parser have not been used with RaspberryMatic.

# Domoticz

There are two Domoticz systems used:

- Domoticz Production System version Stable 2022.2.
- Domoticz Development System version Beta 2022.2 (build 15022 or higher).  
This system is used to explore or test functions.

Both systems have additional tools installed, like Node-RED, mosquitto (MQTT messaging).  
The Domoticz Homeautomation Workbook describes in detail the setup.

## Node-RED

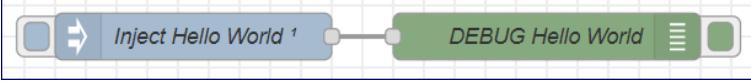
[Node-RED](#) - Low-code programming tool for event-driven applications.

Node-RED can act as a message handler between Homematic and Domoticz v.v. but also as an alternative UI to get or set data.

In this workbook, Node-RED is used to test API's and dashboards.

For the Node-RED UI flows, used the additional modules:  
node-red-dashboard and node-red-node-ui-list.

Example of a Node-RED visual flow with debug information and the flow source.

Node-RED Flow	Node-RED Debug
 <p>The inject node triggers sending the text "Hello World" to the Debug node which logs the message payload.</p>	<pre>21/05/2021, 11:20:05 node: DEBUG Hello World msg.payload : string[11] "Hello World"</pre>

### Node-RED Flow Source (JSON string compact)

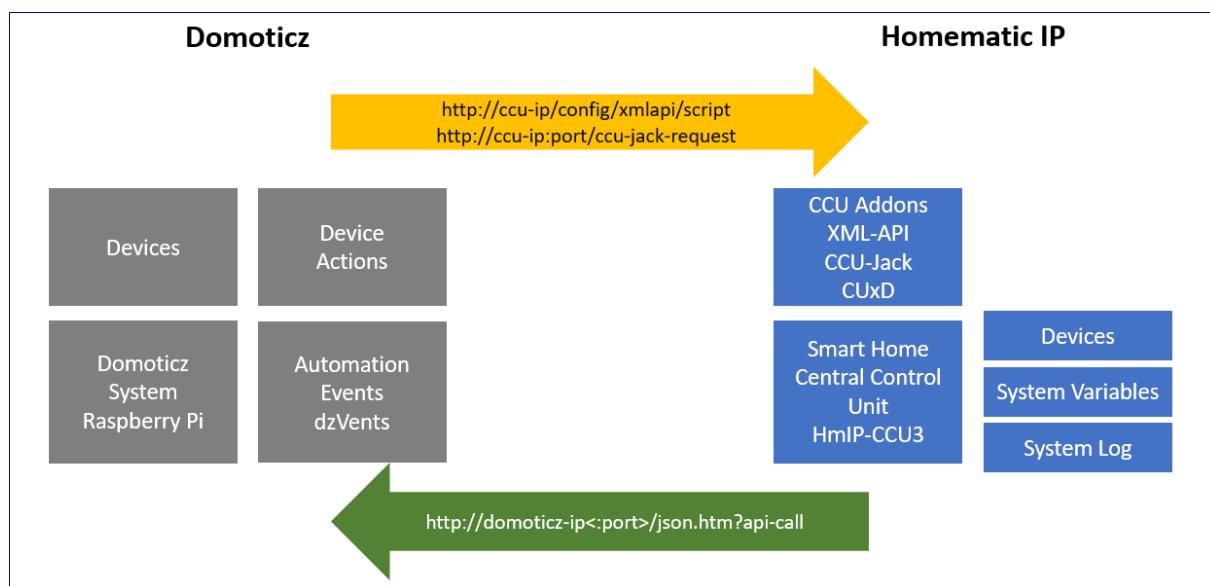
```
[{"id": "9006d730.120958", "type": "inject", "z": "b9202e63.88a7", "name": "Inject Hello World", "props": [{"p": "payload"}], "repeat": "", "crontab": "", "once": true, "onceDelay": 0.1, "topic": "", "payload": "Hello World", "payloadType": "str", "x": 130, "y": 240, "wires": [[[{"id": "c99f1707.c87b38"}]]}, {"id": "c99f1707.c87b38", "type": "debug", "z": "b9202e63.88a7", "name": "DEBUG Hello World", "active": true, "tosidebar": true, "console": false, "tostatus": false, "complete": "payload", "targetType": "msg", "statusVal": "", "statusType": "auto", "x": 360, "y": 240, "wires": []}]
```

The flow source can be imported in Node-RED via the Import menu Copy & Paste  
(Import nodes > Clipboard > Paste flow).

# Communication

## Concept

The communication between Domoticz and the CCU is handled via HTTP API requests.



The requests are submitted from

- **Domoticz to CCU** (yellow arrow) via Domoticz Device Actions or Domoticz dzVents automation events.
  - Using one of the two addons XML-API or CCU-Jack.
  - Domoticz triggers an HTTP XML-API request.  
(with parameters depending on the CGI script) to the CCU.
  - Domoticz to parse the HTTP response, which is XML format using XPath (XML-API) or JSON format (CCU-Jack).
- **CCU to Domoticz** (green arrow) via Addon CUxD “wget” commands embedded in Homematic programs with scripts.
  - The wget command holds a Domoticz HTTP JSON/API request to directly set a device value or trigger a dzVents Custom Event.

## Examples

See next two examples with Domoticz dzVents Automation Event code snippets and Homematic scripts.

Look up chapter [Explore](#) for further examples.

## Domoticz Trigger State Change Homematic Device

Domoticz to trigger the state change of a Homematic device.

The action is to **SET** for the device HmIP-PSM (Homematic IP Pluggable Switch and Meter) the **Switch actuator to true (On)**.

The datapoint ise\_id of the channel is 1485.



Screenshot Homematic WebUI > Status and Control > Devices > MakeLab PSM - the name of the HMIP-PSM with channel 0001D3C99C6AB3:3.

### Domoticz dzVents script trigger using XML-API

The XML-API script “statechange.cgi” sets the new value (state to true) for the datapoint with ise\_id=1485.

```
-- dzVents script snippet
-- Declare the XML-API url to set the new state of the Homematic device datapoint
local url = 'http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id= 1485&new_value=true'

-- Submit the url PUT request - no postData required
domoticz.openURL({url = url, method = 'PUT', callback = RES_XMLAPI})
```

The HTTP request can also be used in a Domoticz device On or Off action.  
Define in the device widget edit.

On Action:	<a href="http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id=1485&amp;new_value=true">http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id=1485&amp;new_value=true</a>
Off Action:	<a href="http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id=1485&amp;new_value=false">http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id=1485&amp;new_value=false</a>

### Domoticz dzVents script trigger using CCU-Jack

CCU-Jack REST-API request to set the new state by submitting post data in JSON format.

```
-- dzVents script snippet
-- Declare the url with port to set the new state of the datapoint
local url = 'http://ccu-ip:port/device/0001D3C99C6AB3/3/STATE/~pv'

-- Declare the new state to true for the datapoint
local newState = true

-- Declare post data as JSON object, the name 'v' set the value for the new state
local Data = { v = newState }

-- Submit the url PUT request with post data
domoticz.openURL({ url = url, method = 'PUT', callback = 'RES_CCUJACK', postData = Data })
```

## Domoticz dzVents script trigger using Remote Homematic Script API

Remote Homematic script submitted via HTTP POST request to set the state of the HmIP-PSM device.

```
-- Define the IDX of the switch device
local IDX_SWITCH = 1

-- Define the Remote Homematic Script URL.
local URL_HMAPI = 'http://ccu-ip:8181/tclregae.exe'
-- HTTP Response - MUST be UNIQUE across all events
local RES_HMAPI = "SWITCH-MAKELAB"
-- Define the Homematic script as long string. Do not encode, i.e., leave spaces etc.
-- The placeholder #STATE# sets the state.
local function setscript(domoticz, state)
    local script = [[! Homematic script
        ! Set the state to on or off of a psm device
        var result = dom.GetObject("HMIP-PSM 0001D3C99C6AB3:3").DPByHssDP("STATE").State(#STATE#);
    ]]
    if (state == 0) then state = "false" else state = "true" end
    return string.gsub(script, "#STATE#", state)
end

return {
    on = { devices = { IDX_SWITCH }, httpResponses = { RES_HMAPI } },
    logging = { level = domoticz.LOG_INFO, marker = RES_HMAPI, },
    execute = function(domoticz, item)
        if (item.isDevice) then
            -- Submit the HTTP POST request = Remote Homematic Script
            domoticz.openURL({
                url = URL_HMAPI,
                headers = { ['content-type'] = 'application/x-www-form-urlencoded' },
                method = 'POST', postData = setscript(domoticz, item.nValue), callback = RES_HMAPI,
            })
        end
        if (item.isHTTPResponse) then
            if (item.ok) then
                local result = domoticz_applyXPath(item.data,'/xml/result/text()')
                domoticz.log(string.format('Switch state changed. Result: %s', result))
            else
                domoticz.log('There was a problem handling the request', domoticz.LOG_ERROR)
                domoticz.log(item, domoticz.LOG_ERROR)
            end
        end
    end
}
```

## Homematic Trigger State Change Domoticz Switch

Homematic to trigger the state change of a Domoticz device.

Homematic to **SET** the state of the Domoticz Switch Device (IDX=99) to **On** – via HTTP JSON/API Request.

### Homematic Program Script with CUxD

CUxD to run a “wget” command with the Domoticz HTTP JSON/API request parameter.

```
! Define the url to switch the Domoticz switch state to On (ensure right case for On)
string url = "http://domoticz-ip:port/json.htm?type=command&param=switchlight&idx=99&switchcmd=On";

! Run CuxD CMD_EXEC command with wget - result is true or false
var cmdRes = dom.GetObject("CUxD.CUX2801001:1.CMD_EXEC").State("wget -q -O - "#url);
WriteLine("Exec Result: " # cmdRes);
```

Prior implementing the Domoticz HTTP/API request, test the URL via Browser and check the Domoticz HTTP response (JSON object).

### HTTP XML-API Request

```
http://domoticz-ip:port/json.htm?type=command&param=switchlight&idx=83&switchcmd=On
```

### HTTP XML-API Response

```
{ "status" : "OK", "title" : "SwitchLight"}
```

# Functions

## Alert Indicator (HmIP-MOD-OC8)

### Purpose

To visual indicate, via coloured LEDs an alert state or alert level.

Usage Examples – not all implemented (just some ideas):

- Covid-19 Incidence with threshold below (LED GREEN) or above (LED RED). Show the indicator daily between 07:00 – 08:00.
- Postbox hatch opened (“post delivered”) (LED BLUE). Show the indicator between 08:00 and 18:00; Hourly reset.
- Doorbell ringing (LED RED).
- Sunset (LED RED).
- Domoticz system issue (LED RED Blinking).
- Domoticz device level or value displayed as LED progress bar 8 LEDs with for example levels 0 to 8.
- More to explore ... not all states are implemented = just thoughts.

### Solution

This solution uses 3 out of the 8 channels of the HmIP-MOD-OC8.

A Domoticz virtual sensor Selector Switch (4 states) triggers the state ON (true) or OFF (false) of the Homematic IP device HmIP-MOD-OC8.

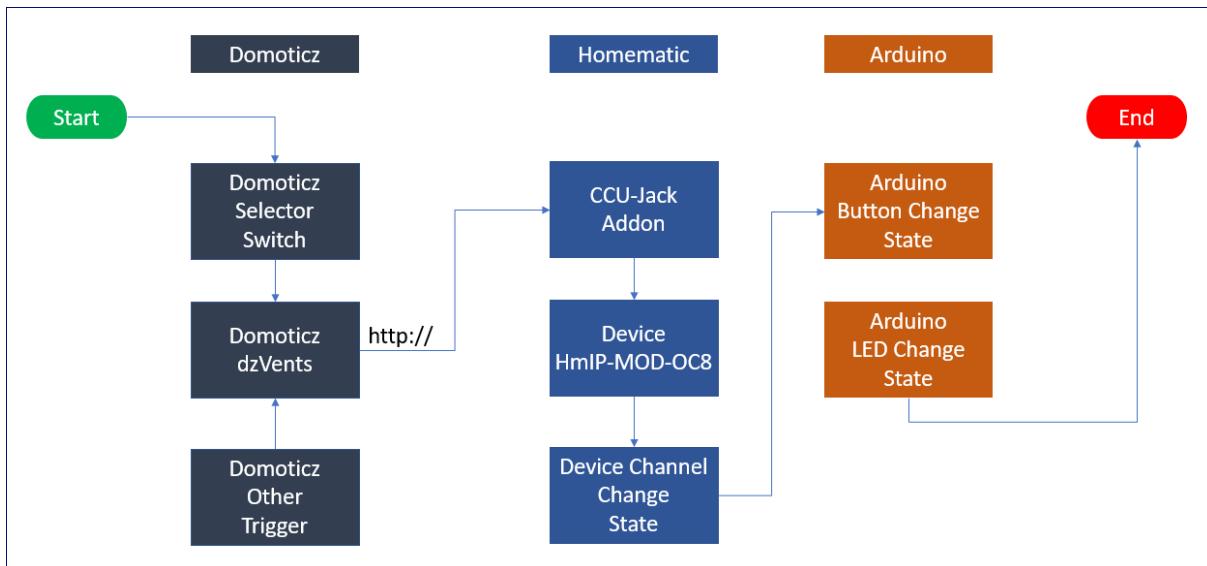
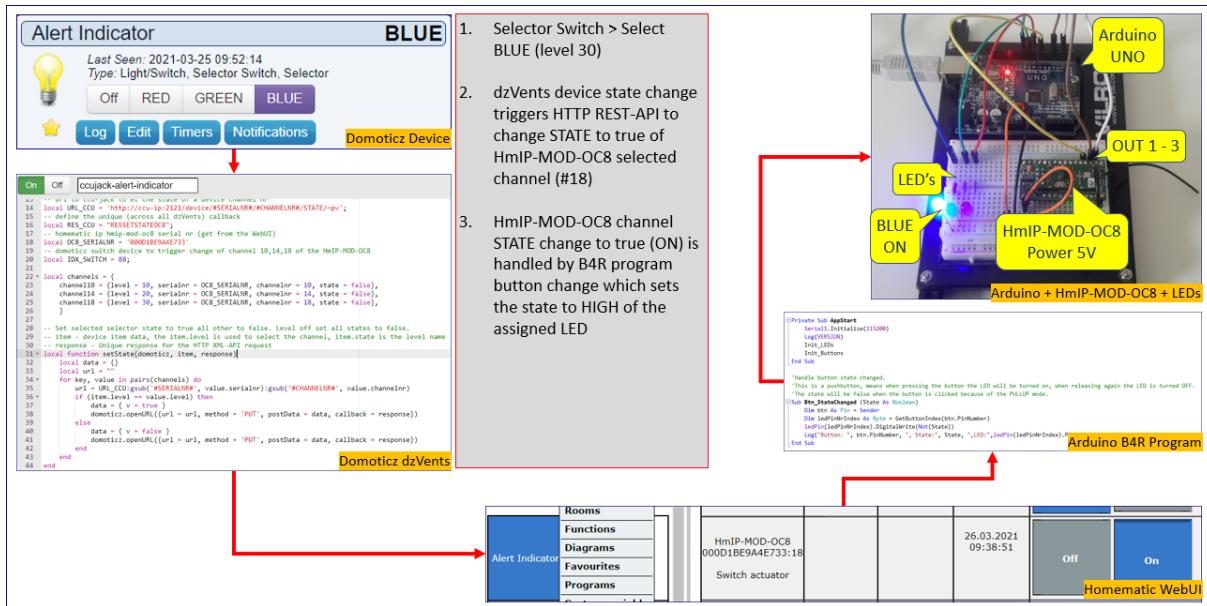
The state change of a HmIP-MOD-OC8 channel is handled via a HTTP REST-API PUT request (using the Homematic addon CCU-Jack) with JSON object as data.

The HmIP-MOD-OC8 is connected to an Arduino UNO with 3 LED's assigned to an HmIP-MOD-OC8 channel.

The HmIP-MOD-OC8 gets 5V power supply from the Arduino and common ground.

Depending which HmIP-MOD-OC8 channel state is changed, the associated LED is turned ON (HIGH) or OFF (LOW).

Domoticz triggers changing the state of a LED via Homematic, but it is also possible that Homematic triggers a state change of a Domoticz device.



## Homematic Configuration

An HmIP-MOD-OC8 is added to the CCU. Default settings used.

Name	Type description	Picture	Description		Serial number	Interface	Firmware																												
Alert Indicator	HmIP-MOD-OC8		Homematic IP Switch Actuator with OC-Output		000D1BE9A4E733	HmIP-RF	Version: 1.8.12																												
		<table border="1"> <thead> <tr> <th>Name</th> <th>Room</th> <th>Function</th> <th>Last modified</th> <th colspan="2">Control</th> </tr> <tr> <th>Filter</th> <th>Filter</th> <th>Filter</th> <th></th> <th>Off</th> <th>On</th> </tr> </thead> <tbody> <tr> <td>HmIP-MOD-OC8 000D1BE9A4E733:10 Switch actuator</td> <td></td> <td></td> <td>22.03.2021 13:44:11</td> <td>Off</td> <td>On</td> </tr> <tr> <td>HmIP-MOD-OC8 000D1BE9A4E733:14 Switch actuator</td> <td></td> <td></td> <td>19.03.2021 07:35:33</td> <td>Off</td> <td>On</td> </tr> <tr> <td>HmIP-MOD-OC8 000D1BE9A4E733:18 Switch actuator</td> <td></td> <td></td> <td>18.03.2021 15:39:43</td> <td>Off</td> <td>On</td> </tr> </tbody> </table>				Name	Room	Function	Last modified	Control		Filter	Filter	Filter		Off	On	HmIP-MOD-OC8 000D1BE9A4E733:10 Switch actuator			22.03.2021 13:44:11	Off	On	HmIP-MOD-OC8 000D1BE9A4E733:14 Switch actuator			19.03.2021 07:35:33	Off	On	HmIP-MOD-OC8 000D1BE9A4E733:18 Switch actuator			18.03.2021 15:39:43	Off	On
Name	Room	Function	Last modified	Control																															
Filter	Filter	Filter		Off	On																														
HmIP-MOD-OC8 000D1BE9A4E733:10 Switch actuator			22.03.2021 13:44:11	Off	On																														
HmIP-MOD-OC8 000D1BE9A4E733:14 Switch actuator			19.03.2021 07:35:33	Off	On																														
HmIP-MOD-OC8 000D1BE9A4E733:18 Switch actuator			18.03.2021 15:39:43	Off	On																														

The screenshot shows channel 10 is On (state is true). Channel 10 is assigned to OUT1 pin.

### Channel to Pin Mapping

HmIP-MOD-OC8 Channel	HmIP-MOD-OC8 Pin	Arduino Pin & LED
10	OUT1	#D2 = RED
14	OUT2	#D4 = GREEN
18	OUT3	#D7 = BLUE
Add more ...	OUT4-8	

# Domoticz Configuration

## Devices

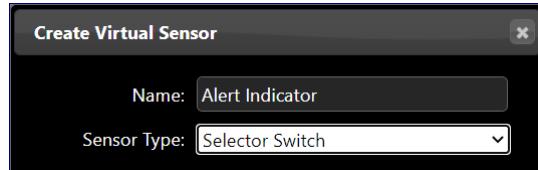
Create a Virtual Sensor Type Selector Switch using the Hardware Dummy.

After creating, add the Selector Levels to the switch.

For this example, 3 levels plus Off are used.

As the HmIP-MOD-OC8 has 8 channels assigned to OUT1-8, up-to 8 levels can be assigned to the selector switch.

Idx	Name	Enabled	Type	Address
4	VirtualSensors	Yes	Dummy (Does nothing, use for virtual switches only) <a href="#">Create Virtual Sensors</a>	



Devices

Idx	Hardware	ID	Unit	Name	Type	Sub Type	Data
88	VirtualSensors	000140A8	1	Alert Indicator	Light/Switch	Selector Switch	Off

Device Widget GUI Tab Switches

RED

Last Seen: 2021-03-24 09:54:33  
Type: LightSwitch, Selector Switch, Selector

Off RED GREEN BLUE

Log Edit Timers Notifications

Idx: 88

Name: Alert Indicator

Switch Type: Selector

Switch Icon:

On Delay: 0 (Seconds) 0 = Disabled

Off Delay: 0 (Seconds) 0 = Disabled

Protected:

Selector Style:  Button set  Select menu

Hide Off level:

Selector Levels:

Level	Level name
0	Off
10	RED
20	GREEN
30	BLUE

## Automation Event

The Domoticz Automation Event dzVents script handles the selector switch levels changes. If a level is selected, all LEDs are turned off except the selected level. If the selected level is OFF (0), all LEDs remain off.

```
-- url to ccu-jack to set the state of a device channel nr
local URL_CCU3 = 'http://ccu-ip:port/device/#SERIALNR#/CHANNELNR#/STATE/~pv';
-- define the unique (across all dzVents) callback
local RES_CCU3 = "RESETSTATEOC8";
-- Homematic ip hmip-mod-oc8 serial nr (get from the Homematic WebUI)
local OC8_SERIALNR = '000D1BE9A4E733'
-- domoticz switch device to trigger change of channel 10,14,18 of the HmIP-MOD-OC8
local IDX_SWITCH = 88;

local channels = {
    channel10 = {level = 10, serialnr = OC8_SERIALNR, channelnr = 10, state = false},
    channel14 = {level = 20, serialnr = OC8_SERIALNR, channelnr = 14, state = false},
    channel18 = {level = 30, serialnr = OC8_SERIALNR, channelnr = 18, state = false},
}

-- Set selected selector state to true all other to false. Level off set all states to false.
-- item - device item data, the item.level is used to select the channel, item.state is the level name
-- response - Unique response for the HTTP XML-API request
local function setState(domoticz, item, response)
    local data = {}
    local url = ""
    for key, value in pairs(channels) do
        url = URL_CCU3:gsub('#SERIALNR#', value.serialnr):gsub('#CHANNELNR#', value.channelnr)
        if (item.level == value.level) then
            data = { v = true }
            domoticz.openURL({url = url, method = 'PUT', postData = data, callback = response})
        else
            data = { v = false }
            domoticz.openURL({url = url, method = 'PUT', postData = data, callback = response})
        end
    end
end

return {
    on = { devices = { IDX_SWITCH }, httpResponses = { RES_CCU3 } },
    execute = function(domoticz, item)
        if (item.isDevice) then
            setState(domoticz, item, RES_CCU3)
        end
        if (item.isHTTPResponse) then
            if (item.statusCode == 200) then
                if (item.callback == RES_CCU3) then
                    domoticz.log(item.statusText)
                end
            else
                domoticz.log('[ERROR] Request:' .. item.statusText, domoticz.LOG_ERROR)
            end
        end
    end
}
```

The function setAlertIndicator() can be triggered by a timer or device.

Timer ON/OFF example:

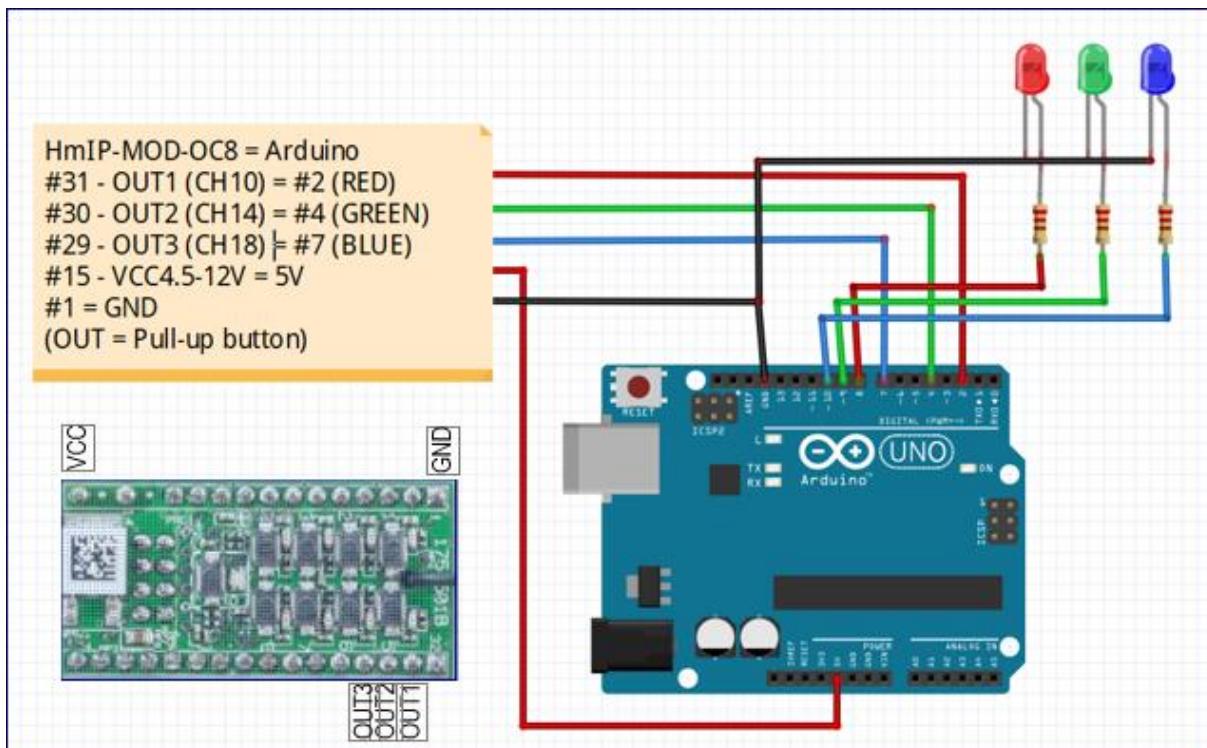
```
local TIMERRULE_ON      = 'at 07:00'
local TIMERRULE_OFF     = 'at 08:00'
return {
    on = { timer = { TIMERRULE_ON, TIMERRULE_OFF} },
    execute = function(domoticz, triggeredItem)
        if triggeredItem.isTimer then
            if triggeredItem.trigger == TIMERRULE_ON then
                -- Check something and set the bulb idx, value, threshold.
                -- The value 38 is below threshold 50, so the bulb is turned on green for an hour
                setAlertIndicator(domoticz, IDX_HUE, 38, 50)
            end
        end
    end
}
```

```
    if triggeredItem.trigger == TIMERRULE_OFF then -- Switch the bulb off
        domoticz.devices(IDX_HUE).switchOff()
    end
end
end
}
```

# Arduino

## Circuit

The 3 HmIP-MOD-OC8 channels are connected to digital pins. The pins are configured as buttons in the B4R program. The 3 LEDs are also connected to digital pins with resistors. The HmIP-MOD-OC8 gets 5V power supply from the Arduino and common ground.



## Program

The Arduino program is written in [B4R](#) - development tool for native Arduino, ESP8266 and ESP32 programs.

```
#Region Project Notes
' alertindicator
' Project: domoticz-homematicip-workbook
' Set LED Blinking state on / off via Homematic WebUI or XML-API using HmIP-MOD-OC8 Out
' Hardware: Arduino UNO, HmIP-MOD-OC8
#End Region

#Region Wiring
' HmIP-MOD-OC8 = Arduino UNO
' OUT 1 = PIN 2 (GPIO2)
' OUT 2 = PIN 4 (GPIO4)
' OUT 3 = PIN 7 (GPIO7)

' LED = Arduino UNO
' (+) > R330Ohm = PIN 8 (GPIO8) RED
' (-) = GND
' (+) > R330Ohm = PIN 9 (GPIO9) GREEN
' (-) = GND
' (+) > R330Ohm = PIN 10 (GPIO10) BLUE
' (-) = GND
#End Region

#Region Project Attributes
#AutoFlushLogs: True
```

```

#StackBufferSize: 300
#End Region

Sub Process_Globals
    Private VERSION As String = "Alert Indicator HmIP-MOD-OC8 v20210322"
    Public Serial1 As Serial
    Private Const OUTCHANNELS As Byte = 3
    Private ledPin(3) As Pin
    Private ledPinNr(3) As Byte = Array As Byte(0x08, 0x09, 0x0A)
    Private btnPin(3) As Pin
    Private btnPinNr(3) As Byte = Array As Byte(0x02, 0x04, 0x07)
End Sub

Private Sub Init_LEDs
    Dim i As Int
    For i = 0 To OUTCHANNELS - 1
        ledPin(i).Initialize(ledPinNr(i), ledPin(i).MODE_OUTPUT)
        ledPin(i).DigitalWrite(False)
    Next
End Sub

'Init the button pin. Using the internal pull up resistor to prevent the pin from floating.
Private Sub Init.Buttons
    Dim i As Int
    For i = 0 To OUTCHANNELS - 1
        btnPin(i).Initialize(btnPinNr(i), btnPin(i).MODE_INPUT_PULLUP)
        btnPin(i).AddListener("Btn_StateChanged")
    Next
End Sub

'Get the button position index from the array. This position is used to get the led from the led array.
'Basically, a mapping ledPinNr(8,9,10) = btnPinNr(2,4,7)
'Example <code>
'Dim ledPinNrIndex As Byte = GetButtonIndex(btn.PinNumber)
'</code>
Private Sub GetButtonIndex(PinNr As Byte) As Byte
    Dim i As Int
    For i = 0 To OUTCHANNELS - 1
        If btnPinNr(i) = PinNr Then
            Log("Found:",btnPinNr(i), "=",PinNr,",Result:",i)
            Return i
        End If
    Next
    Return -1
End Sub

Private Sub AppStart
    Serial1.Initialize(115200)
    Log(VERSION)
    Init_LEDs
    Init.Buttons
End Sub

'Handle button state changed.
'This is a pushbutton, means when pressing the button, the LED will be turned on, when releasing again the LED is turned OFF.
'The state will be false when the button is clicked because of the PULLUP mode.
Sub Btn_StateChanged (State As Boolean)
    Dim btn As Pin = Sender
    Dim ledPinNrIndex As Byte = GetButtonIndex(btn.PinNumber)
    ledPin(ledPinNrIndex).DigitalWrite(Not(State))
    Log("Button: ", btn.PinNumber, ", State:", State, ",LED:", ledPin(ledPinNrIndex).PinNumber)
End Sub

```

## Enhancement Ideas

- Use all 8 channels with 4 RED and 4 GREEN LEDs acting as pairs for a function with either RED or GREEN constant turned on.  
Example: Duty Cycle: OK=Green, Above Threshold=RED.
- Explore if the pins COM and TX could be used by the Arduino.  
Instead sending HTTP requests for each channel, use the COM or TX pin to send the state of all channels as for example a byte.
- Domoticz device level or value displayed as LED progress bar.  
Use 8 RED LEDs with, for example, levels 0 to 8. Level 5 would turn on channels 1-5.
- TM1637 4-Digit-Segment Display - Bit of fun thinking:  
As there are 8 channels, a value between 0-255 could be displayed on a TM1637.  
Each channel is a bit; 8 channels = 8 bit = byte, max value is  $2^8 - 1$ .
- TM1637 - As previous but to be able to set value 0000 – 9999.
  - The channels 1-4 to set a value between 0-9 for the digits 1-4.
  - The value of the digit set by channels 5-8.  
Each channel is a bit; 4 channels enable to set bin 0000 (=0) – 1001 (=9).
  - Use value 9999 to clear the display.
  - A solution using HTTP requests would cause lots of requests and complex to handle by the B4R program. For each of the 4 digits, repeat:
    - First HTTP request is to select the digit 1-4 by selecting channel 1–4.
    - Followed by 4 HTTP requests to set the value for the selected digit by setting the state ON (1) or OFF (0) for each of the channels 5-8.
    - Means for the 4 digits, send  $4 * 5 = 20$  HTTP requests.
    - No other solution found yet using an Arduino without Ethernet or WiFi.

# Battery Check (HmIP All Devices)

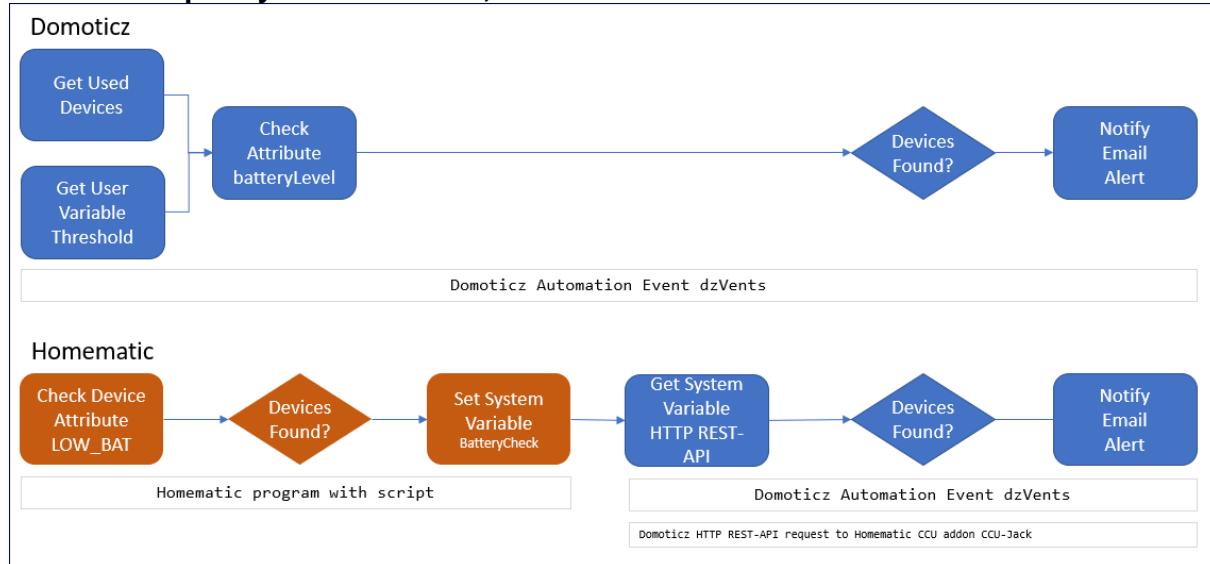
## Purpose

- To daily check the battery low level status of Domoticz & Homematic IP devices.
- To notify for devices having battery low level:
  - Sent email per system, i.e., Domoticz, Homematic.
  - Update a Domoticz Alert message.
- To disable checking the battery low level status.

## Solution

Each system has its own solution to **check battery low level** status but resulting in a common way to notify via email & alert message.

### Action Flow per System Domoticz, Homematic



# Solution Domoticz

## Concept

Daily at 01:00 the battery level for all devices is checked, via a Domoticz Automation Event dzVents against a threshold (defined as Domoticz User Variable).  
 If the battery level for a device is below threshold, the device name & battery level are added to a message.  
 After check completion, the message with the devices found is sent via email and to an Alert device. No action is taken if there are no devices found.

### Example Email and Alert Message

From	Subject	
rwbllin@outlook.de	Domoticz Batterie Prüfung Keller Großraum: 0%; Windmesser: 0%; <end>	Meldungen <b>Domoticz Batterie Prüfung: Keller Großraum: 0%; Windmesser: 0%;</b>

## User Variable

Idx	Variable name	Variable type	Current value
24	TH_BATTERY_LEVEL	Integer	20

The current value is the battery level threshold in percent (used in the Domoticz Automation Event)

### Disable Battery Level Check

Set the current value of the user variable to 0 to disable the daily battery level check.

#### Note

The user variable can also be set via the Custom page – see Domoticz Homeautomation Workbook, chapter Custom Settings.

The screenshot shows the Domoticz interface with the "Custom" tab selected. Under "Custom Settings", there is a form with two fields: "Name" containing "TH\_BATTERY\_LEVEL" and "Value" containing "20". Below the value field is a note "(%, 0 = Disabled)". A blue "Set" button is at the bottom left of the form.

## Automation Event

```
--[[  
battery_check_domoticz.dzents  
Check battery all domoticz devices in regular intervals (see below TIMERRULE) against threshold.  
Threshold = 0 battery check is disabled.  
Battery level below threshold, add the device name and battery level to a message.  
If devices found, sent an email and update an alert sensor.  
]]--  
  
-- Subject of the message for email and alert  
local MSGSUBJECT = 'Domoticz Batterie Stand Niedrig'  
-- Idx of the user variable holding the battery level threshold (n %)  
local IDX_TH_BATTERYLEVEL = 24  
-- Set a default threshold in case the user var is not found  
local DEF_THRESHOLD = 20  
-- Timer Rule 12 34 567  
local TIMERRULE = 'at 01:00 on mon,wed,sat'  
-- local TIMERRULE = 'every minute'  
  
return {  
    on = {  
        timer = { TIMERRULE },  
        logging = { level = domoticz.LOG_INFO, marker = 'BATTERYCHECKDOMOTICZ' },  
    },  
    execute = function(domoticz, timer)  
        local threshold = domoticz.variables(IDX_TH_BATTERYLEVEL).value  
        if threshold == nil then threshold = DEF_THRESHOLD end  
        if threshold > 0 then  
            -- Loop over all devices and check the property device.batteryLevel  
            -- !Ensure the right case for the device attribute -- see dzVents documentation  
            local message = 'OK'  
            domoticz.devices().forEach(function(device)  
                if device.batteryLevel ~= nil then  
                    if message == 'OK' then message = '' end  
                    if device.batteryLevel < threshold then  
                        message = message .. ('%s:%d%%; '):format(device.name, device.batteryLevel)  
                    end  
                end  
            end)  
            domoticz.log(message)  
            -- Always send notification via mail  
            domoticz.notify(MSGSUBJECT, message, domoticz.PRIORITY_HIGH)  
            if message ~= 'OK' then  
                domoticz.helpers.alertmsg(domoticz,domoticz.ALERTLEVEL_YELLOW, ('%s:%s'):format(MSGSUBJECT,  
message))  
            end  
        end  
    end  
}
```

# Solution Homematic

## Concept

Daily at 00:00 a Homematic program executes a script which loops over all devices to check the device attribute LOW\_BAT. If there are devices found, the device name is added to a message. If there are no devices found, the message contains the text OK. The message is used to update a Homematic system variable BatteryCheck.

At 01:00 (3 times per week) a Domoticz Automation Event dzVents requests via HTTP REST-API request (handled by the CCU addon CCU-Jack) the value of the Homematic system variable BatteryCheck via its datapoint id (attribute ise\_id).

The HTTP response is used to send a notification. If the response does not contain the text OK, to response is also used to update an Domoticz Alert device.

## System Variable

The system variable BatteryCheck definition.

Name	Description	Variable type	Values	Unit of measurement	Channel assignment
BatteryCheck	Daily Device Battery Check	Character string ▾			<input checked="" type="radio"/> without <input type="radio"/> with <input type="checkbox"/> Channel selection

## Program

The Homematic program runs daily at 00:00 to check for all devices the LOW\_BAT attribute against value true.

The system variable BatteryCheck is updated either with value OK or list of devices with LOW\_BAT.

Name	Description	Condition (if...)	Activity (then..., or else...)	Action
Domoticz Battery Check	Check regular intervals the battery state LOW_BAT to Domoticz alert device.	Time: Daily at 00:00 o'clock starting at 17.02.2021 trigger at point of time	Script: ... immediately run	<input type="checkbox"/> intrinsic

**Condition: If...**

Time control ▾ Daily at 00:00 o'clock starting at 17.02.2021 trigger at point of time ▾

**Activity: Then...**  Stop all current delays before performing the activity (e.g. retriggering).

Script ▾ !battery\_check\_homematic.script ! Once per day at 00:00, loop\_over\_all... immediately ▾

**Activity: Else...**  Stop all current delays before performing the activity (e.g. retriggering).

## Script

```

! battery_check_homematic.script
! Once per day at 00:00, loop over all devices to check the value of data point LOW_BAT against true
! (means the battery of the device is low)
! Update system variable BatteryCheck with devices found or OK DD.MM if all devices are fine.
! Using OK plus date to show the last date checked.
! 20210218 rwbl

! Get the actual date and time DD.MM HH:MM
string actDate = system.Date("%d.%m%"); ! sDate = "09.08"; "%d.%m.%Y" = "09.08.2020";
string actTime = system.Date("%H:%M%"); ! sTime = "07:32"; "%H:%M:%S") = "07:32:00";

! System variable holding the text of the battery check.
string BATTERYCHECK = "BatteryCheck";
! Message text to save in the system variable BatteryCheck.
string text = "";

```

```
! Number of devices with LOW_BAT = true
integer counter = 0;

! Get all devices objects.
var objIDs = dom.GetObject(ID_DEVICES).EnumUsedIDs();

! Loop over all datapoints for the devices found.
string id;
foreach(id, objIDs){
    !Get datapoint.
    var object = dom.GetObject(id);
    ! Name: Briefkasten Status: ID=2530
    ! WriteLine("Name: " # object.Name() # ":" # ID=" # object.ID());

    ! Get the device name from the object.
    string DeviceName = object.Name();
    ! Get the the device object from the device name
    var Device = dom.GetObject(DeviceName);
    string id;
    ! Loop over all channels to get the datapoint LOW_BAT.
    foreach(id, object.Channels()) {
        ! Get the channel.
        var ch = dom.GetObject(id);
        ! Get the datapoint for the attribute LOW_BAT from the channel
        var dp = ch.DPByHssDP("LOW_BAT");
        if (dp) {
            ! dpValue() true|false. If true the battery is low
            if (dp.Value() == true) {
                counter = counter + 1;
                text = text # DeviceName # ";";
                ! WriteLine(DeviceName # ", Battery LOW = " # dp.Value());
            }
        }
    }
}

! If no devices with low battery found, set text to OK DD.MM HH:MM
if (counter == 0) {
    text = "OK " # actDate # " " # actTime;
}

! Update the systemvariable BatteryCheck.
var objBatteryCheck = dom.GetObject(BATTERYCHECK);
objBatteryCheck.Variable(text);
! WriteLine(objBatteryCheck.Variable());

! Log for tests the content of the system variable
! WriteLine(objBatteryCheck.Variable());
```

## Automation Event

```
--[[  
    battery_check_homematic.dzvents  
    Check the battery level of all Homematic devices in regular intervals (see below TIMERRULE)  
    A Homematic script runs every night and sets a system variable BatteryCheck type string.  
    The Homematic device datapoint LOW_BAT is checked true (battery is low) or false (battery ok).  
    Get the list of sysvars:  
    http://ccu-ip:port/sysvar  
    Get the ID (href) of the sysvar DutyCycle from previous result JSON Data:  
    "rel": "sysvar","href": "2799","title": "BatteryCheck"  
    Get the value of the sysvar with ID 3032 from result JSON Data:  
    http://ccu-ip:port/sysvar/2799/~pv  
    {"ts":1614954055000,"v":"OK 08.03 00:00","s":0}  
    If the value does not start with "OK", there are devices with low battery state.  
    Threshold = 0 battery check is disabled.  
    If devices found, sent an email and update an alert sensor.  
]]--  
-- Subject of the message for email and alert  
local MSGSUBJECT = 'Homematic Batterie Prüfung'  
-- Domoticz idx of the user variable setting the Homematic ccu ip address & port (without / at the end)  
local IDX_UV_URL_HOMEMATIC = 26  
-- Homematic sysvar ise_id BatteryCheck  
local HM_SYSVAR_BATTERYCHECK_ID = 2799  
-- Domoticz idx of the user variable holding the battery level threshold (n %)  
local IDX_TH_BATTERYLEVEL = 24  
-- Set a default threshold (%) in case the user var is not found  
local DEF_THRESHOLD = 20  
-- define the unique (across all dzVents) callback  
local RES_HOMEMATIC = "RES_BATTERYCHECKHOMEMATIC";  
-- Timer Rule 12 34 567  
local TIMERRULE = 'at 01:00 on mon,wed,sat'  
-- local TIMERRULE = 'every minute'  
  
return {  
    on = {  
        timer = { TIMERRULE },  
        httpResponses = { RES_HOMEMATIC },  
        logging = { level = domoticz.LOG_INFO, marker = RES_HOMEMATIC },  
    },  
    execute = function(domoticz, item)  
        if (item.isTimer) then  
            local threshold = domoticz.variables(IDX_TH_BATTERYLEVEL).value  
            if threshold == nil then threshold = DEF_THRESHOLD end  
            if threshold > 0 then  
                local url = ('%s/sysvar/%d/~pv'):format(domoticz.variables(IDX_UV_URL_HOMEMATIC).value,  
HM_SYSVAR_BATTERYCHECK_ID)  
                domoticz.openURL({url = url, method = 'GET', callback = RES_HOMEMATIC,})  
            end  
        end  
  
        if (item.isHTTPResponse) then  
            if (item.statusCode == 200) then  
                if (item.callback == RES_HOMEMATIC) then  
                    local message = item.json.v  
                    domoticz.notify(MSGSUBJECT, message, domoticz.PRIORITY_HIGH)  
                    if message:sub(1,2) ~= "OK" then  
                        domoticz.helpers.alertmsg(domoticz, domoticz.ALERTLEVEL_YELLOW, ('%s:%s'):format(MSGSUBJECT,  
message))  
                    end  
                end  
            else  
                domoticz.log('[ERROR] Request:' .. item.statusText, domoticz.LOG_ERROR)  
            end  
        end  
    end  
}
```

# Coffee Machine Monitor (HmIP-PSM)

## Purpose

To monitor, the start & end time of the Coffee Machine running early morning.

If the coffee machine is tuned on, an Alert Message is set.

The Coffee Machine switches OFF after two hours.

## Solution

The value of the Homematic HmIP-PSM datapoint POWER is obtained, every 2 minutes between 07:00 and 09:30, via a Domoticz automation event and assigned to a Domoticz Virtual Sensor Type Usage, SubType Electric.

If the value of the Domoticz device is above a threshold, set by a User Variable (type Float, value 500 Wh) an Alert sensor is set (message and level RED).

Monitoring is handled by a dzVents script triggered by a timer.

The script uses a persistent variable (data) "notified" to handle the Alert Message notifications.

## Homematic Configuration

### Device

Setup: The device HMIP-PSM is teached in and renamed to "Coffee Machine PSM".

### Screenshot of the device listed in the devices (Menu Status and control)

			Name	Room	Function	Last modified	Control
			Filter	Filter	Filter		
Coffee Machine PSM			HMIP-PSM 0001D3C99C6AB3:3				
Doorbell Detector			Switch actuator				
Dusche			HMIP-PSM 0001D3C99C6AB3:6			03.06.2021 10:53:20	 11.243 kWh 11.24 EUR   9914.90 Wh 9.91 EUR   234.00 V 0 mA   0.00 W 50.02 Hz
Esszimmer			Status report measured value channel				
Flur			HMIP-PSM 0001D3C99C6AB3:8				 3 Auto mode Manu mode  
			Weekly program				

The device attribute **POWER** from channel **HMIP-PSM 0001D3C99C6AB3:6** (Status report measured value channel) is used to monitor & trigger action.

## Datapoints

The datapoints for the channel 6 are retrieved from the device statelist via an HTTP XML-API Request.

### The datapoint POWER attributes

```
<datapoint name="HmIP-RF.0001D3C99C6AB3:6.POWER" type="POWER" ise_id="1507" value="0.130000" valueuetype="4" valueunit="W" timestamp="1622723540" operations="5"/>

...
<channel name="HMIP-PSM 0001D3C99C6AB3:6" ise_id="1500" index="6" visible="true" operate="true">
  <datapoint name="HmIP-RF.0001D3C99C6AB3:6.CURRENT" type="CURRENT" ise_id="1501" value="17.000000" valueuetype="4" valueunit="mA" timestamp="1622723540" operations="5"/>
  <datapoint name="HmIP-RF.0001D3C99C6AB3:6.CURRENT_STATUS" type="CURRENT_STATUS" ise_id="1502" value="0" valueuetype="16" valueunit="" timestamp="1622723540" operations="5"/>
  <datapoint name="HmIP-RF.0001D3C99C6AB3:6.ENERGY_COUNTER" type="ENERGY_COUNTER" ise_id="1503" value="0.200000" valueuetype="4" valueunit="Wh" timestamp="1622723540" operations="5"/>
  <datapoint name="HmIP-RF.0001D3C99C6AB3:6.ENERGY_COUNTER_OVERFLOW" type="ENERGY_COUNTER_OVERFLOW" ise_id="1504" value="false" valueuetype="2" valueunit="" timestamp="1622723540" operations="5"/>
  <datapoint name="HmIP-RF.0001D3C99C6AB3:6.FREQUENCY" type="FREQUENCY" ise_id="1505" value="50.010000" valueuetype="4" valueunit="Hz" timestamp="1622723540" operations="5"/>
  <datapoint name="HmIP-RF.0001D3C99C6AB3:6.FREQUENCY_STATUS" type="FREQUENCY_STATUS" ise_id="1506" value="0" valueuetype="16" valueunit="" timestamp="1622723540" operations="5"/>
  <datapoint name="HmIP-RF.0001D3C99C6AB3:6.POWER" type="POWER" ise_id="1507" value="0.130000" valueuetype="4" valueunit="W" timestamp="1622723540" operations="5"/>
  <datapoint name="HmIP-RF.0001D3C99C6AB3:6.POWER_STATUS" type="POWER_STATUS" ise_id="1508" value="0" valueuetype="16" valueunit="" timestamp="1622723540" operations="5"/>
  <datapoint name="HmIP-RF.0001D3C99C6AB3:6.VOLTAGE" type="VOLTAGE" ise_id="1509" value="230.200000" valueuetype="4" valueunit="V" timestamp="1622723540" operations="5"/>
  <datapoint name="HmIP-RF.0001D3C99C6AB3:6.VOLTAGE_STATUS" type="VOLTAGE_STATUS" ise_id="1510" value="0" valueuetype="16" valueunit="" timestamp="1622723540" operations="5"/>
</channel>
...
```

*The XML tree is showing the elements for channel 6 only. The power datapoint has ise\_id 1507.*

# Domoticz Configuration

## Devices

Required are 2 devices (virtual sensors) and a user variable:

- Device Usage (Electric)**  
Measure the power of the coffee machine connected to the HmIP-PSM.
- Device General (Alert)**  
Display an alert message, level RED, when the coffee machine turns off, if the power exceeds a given threshold set by a user variable.
- User Variable**  
Threshold to display an alert message.

## Hardware

Idx	Name	Enabled	Type	Address	Port	Data Timeout
4	VirtualSensors	Yes	Dummy (Does nothing, use for virtual switches only) <a href="#">Create Virtual Sensors</a>			Disabled

The hardware type Dummy used to create virtual sensors. All virtual sensors are assigned to this device.

### Power Sensor

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
94	VirtualSensors	82094	1	Coffee Machine Power	Usage	Electric	0.26 Watt

**Coffee Machine Power** 0.26 Watt



Last Seen: 2021-06-03 16:48:00  
Type: Usage, Electric

! [Log](#) [Edit](#) [Notifications](#)

### Alert Sensor

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
26	VirtualSensors	82026	1	Alert Message	General	Alert	Kaffee AUS 18:44

**Alert Message**



Kaffee AUS 18:44  
Last Seen: 2021-06-03 16:44:00  
Type: General, Alert

! [Log](#) [Edit](#) [Notifications](#)

### User Variable

Idx	Variable name	Variable type	Current value
2	TH_COFFEE_MACHINE_MONITOR	Float	500

## Automation Event

```
--[[[
coffee_machine_monitor.dzevents
If the power consumption exceeds a threshold, set an alert message (alert sensor).
Homematic device: HmIP-PSM.
Dependencies: Homematic addon CCU-Jack REST-API
20210604 rwbl
]]]

-- Domoticz devices
local IDX_COFFEE_MACHINE_POWER = 94      -- Type Usage, SubType Electric
local IDX_ALERT_MESSAGE = 26                -- Type General, Alert
-- Domoticz user variables
local TH_COFFEE_MACHINE_MONITOR = 2        -- Name TH_COFFEE_MACHINE_MONITOR, Type Float, Value 500

-- Homematic CCU API Request & Response using addon CCU-Jack
-- Hint: To test run the url from a browser and check JSON response
-- {"ts":1622728746584,"v":3.32,"s":0}
local URL_REQUEST = 'http://ccu-ip:2121/device/0001D3C99C6AB3/6/POWER/~pv'
local RES_REQUEST = 'RES_COFFEE_MACHINE_MONITOR'

-- Alert Message
local MONITOR_MESSAGE    = 'Kaffee %s %02d:%02d'
local MONITOR_STATE_ON   = 'AN'
local MONITOR_STATE_OFF  = 'AUS'

-- Check the HmIP-PSM POWER attribute in regular intervals
local TIMER_RULE = 'every 2 minutes between 07:00 and 09:30'

return {
  on = {
    timer = { TIMER_RULE },
    httpResponses = { RES_REQUEST }
  },
  data = {
    -- set a flag if already notified within timeframe
    notified = { initial = 0 }
  },
  execute = function(domoticz, item)
    -- check if the triggered item is the timer, then request information
    if (item.isTimer) then
      -- Homematic url http rest-api request system variable with ise_id
      domoticz.openURL({url = URL_REQUEST, method = 'GET', callback = RES_REQUEST,})
    end

    -- Check if the item is a httpresponse from the openurl callback
    -- Select the callback - in this case there is only one (any next developments here might be more)
    if (item.isHTTPResponse) then
      -- domoticz.log(('Request: %s (%d)':format(item.statusText, item.statusCode))
      if (item.statusCode == 200) then
        -- domoticz.log('Request: %s (%s)':format(item.callback, tostring(item.isJSON)))
        if (item.callback == RES_REQUEST and item.isJSON) then
          -- Get the threshold
          local threshold = domoticz.variables(TH_COFFEE_MACHINE_MONITOR).value
          -- Get the value from the http JSON response
          local value = item.json.v
          -- Update the device energy
          domoticz.devices(IDX_COFFEE_MACHINE_POWER).updateEnergy(value)
          -- Check if notified
          domoticz.log(('Homematic Coffee Machine Power %.1f, Threshold %.1f'):format(value,
threshold))
          -- Check the actual value against the threshold
          if (value >= threshold) and (domoticz.data.notified == 0) then
            -- set notified flag
            domoticz.data.notified = 1
            -- get the time on and log
            local timeon = os.date("*t")
            local message = (MONITOR_MESSAGE):format(MONITOR_STATE_ON, timeon.hour, timeon.min)
            domoticz.log(message, domoticz.LOG_INFO)
            -- calculate the time off = timeon + 2 hours
            local timeoff = os.date("*t", os.time() + 2*60*60)
            -- define message to notify the time coffee machine switches off
        end
      end
    end
  end
}
```

```
timeoff.min)
    message = (MONITOR_MESSAGE):format(MONITOR_STATE_OFF, timeoff.hour,
message)
    -- update alert message
    domoticz.devices(IDX_ALERT_MESSAGE).updateAlertSensor(domoticz.ALERTLEVEL_RED,
domoticz.log(message, domoticz.LOG_INFO)
end
-- Check if the value is below threshold and a notification was set
-- Action: reset notified
if (value < threshold) and (domoticz.data.notified == 1) then
    domoticz.data.notified = 0
end
end
else
    domoticz.log('[ERROR] Request: %s'):format(item.statusText), domoticz.LOG_ERROR)
end
end
end
}
```

# Duty-Cycle Monitor (HmIP-CCU3)

## Purpose

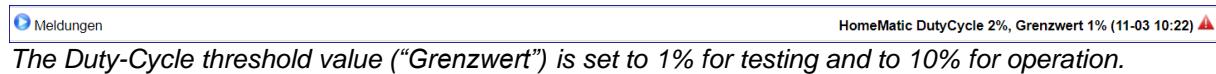
To monitor, in regular intervals, the Duty-Cycle (%) of the CCU (HmIP-CCU3) and alert in case the Duty-Cycle exceeds a threshold.

### Screenshot Domoticz Duty-Cycle Device Widget and Month Log



### Screenshot Domoticz Dashboard

Alert message with Duty-Cycle above threshold ("Grenzwert") 1%.



## Solution

To monitor the Duty-Cycle, two solutions have been worked out.

The first solution has been in place since 2023 after updating the CCU3 to firmware 3.67.10.

### 1 Homematic Device Object HmIP-CCU3

Monitor the Homematic Duty-Cycle, by using the Homematic Device Object HmIP-CCU3 datapoint DUTY\_CYCLE\_LEVEL.

#### Requirements

- CCU3 Firmware 3.67.10.
- [XML-API](#) CCU Addon.

### 2 System Command

Monitor the Homematic Duty-Cycle, by executing, via Homematic Script, a system command and assign the result to a system variable.

#### Requirements

- [CCU-Jack](#) CCU Addon.
- [CuxD](#) CCU Addon.

## Solution Device Object HmIP-CCU3 (In Use)

Since CCU3 firmware 3.67.10, there is a device object HmIP-CCU3 with datapoint **DUTY\_CYCLE\_LEVEL** (type="DUTY\_CYCLE\_LEVEL" ise\_id="5963" valuetype="4" valueunit="%").

The Domoticz Automation Event dzVents "Homematic\_DutyCycle\_Monitor" runs every 15 minutes and requests via XML-API request, the value of the datapoint DUTY\_CYCLE\_LEVEL.

The value is assigned to the Domoticz Device "Homematic Duty-Cycle" (%).

The threshold (in pct) to send an alert message and logerror, is set by uservariable TH\_HOMEMATIC\_DUTYCYCLE.

## HTTP XML-API Requests

Communication between Domoticz and the CCU is based on the [XML-API](#) CCU Addon.

The Duty-Cycle value is stored in the Homematic device object HmIP-CCU3 with datapoint **DUTY\_CYCLE\_LEVEL** (type="DUTY\_CYCLE\_LEVEL" ise\_id="5963" valuetype="4" valueunit="%") as previous described.

To get the Duty-Cycle value, the **ise\_id** of the datapoint **DUTY\_CYCLE\_LEVEL** is required.

The ise\_id is obtained from the Homematic state list.

The ise\_id of the datapoint is **DUTY\_CYCLE\_LEVEL** is 5963 (see below HTTP Response).

## Get Device Object HmIP-CCU3

Get the devices statelist, by running in a browser the XML-API script "statelist.cgi" and select the HmIP-CCU3 device.

From the HmIP-CCU3 device, select the datapoint DUTY\_CYCLE\_LEVEL.

```
http://ccu-ip/config/xmlapi/statelist.cgi
```

From the HTTP response, select, in the XML tree, the device HmIP-CCU3:

```
<stateList>
...
<device name="HmIP-CCU3 001F9A49943DFB" ise_id="5960">
    ...
    <channel name="HmIP-CCU3 001F9A49943DFB:0" ise_id="5961" index="0" visible="true" operate="true">
        <datapoint name="HmIP-RF.001F9A49943DFB:0.CARRIER_SENSE_LEVEL" type="CARRIER_SENSE_LEVEL"
        ise_id="5962" value="0.000000" valuetype="4" valueunit "%" timestamp="1674055203" operations="5"/>
        <datapoint name="HmIP-RF.001F9A49943DFB:0.DUTY_CYCLE_LEVEL" type="DUTY_CYCLE_LEVEL" ise_id="5963"
        value="28.500000" valuetype="4" valueunit "%" timestamp="1674055203" operations="5"/>
        <datapoint name="HmIP-RF.001F9A49943DFB:0.INCLUSION_UNSUPPORTED_DEVICE"
        type="INCLUSION_UNSUPPORTED_DEVICE" ise_id="5964" value="" valuetype="20" valueunit="" timestamp="0"
        operations="5"/>
    </channel>
</device>
...
<stateList>
```

The datapoint DUTY\_CYCLE\_LEVEL (ise\_id=5963) is used for monitoring the Duty-Cycle.

## Get Device Datapoint State

Get the device datapoint, by running in a browser the XML-API script “state.cgi” with parameter “datapoint\_id” and ise\_id 5963 and select the value.

```
http://ccu-ip/config/xmlapi/state.cgi?datapoint_id=5963
```

```
<state>
<datapoint ise_id="5963" value="29.500000"/>
</state>
```

## Homematic Configuration

There is no configuration required, except to ensure the CCU3 firmware is up to date.  
As mentioned requires 3.67.10 or higher).

## Domoticz Configuration

### Device

The Homematic device Homematic Duty-Cycle is configured as a Percentage device in Domoticz - Created as a virtual sensor for hardware Dummy named Virtual Sensor.

	Idx	Hardware	ID	Unit	Name	Type	SubType	Data
	301	VirtualSensors	00082301	1	Homematic Duty-Cycle	General	Percentage	2%

### Automation Event

The dzVents script requests, via HTTP XML-API GET method, in regular intervals the status of the Homematic HmIP-CCU3 datapoint DUTY\_CYCLE\_LEVEL.

The HTTP XML-API response is parsed to get the DutyCycle value from the XML data and updates the Domoticz Device named “Homematic Duty-Cycle”.

```
-- homematic_dutycycle_monitor.dzvents
-- Set the loglevel
local LOGLEVEL = domoticz.LOG_INFO --domoticz.LOG_DEBUG

-- Homematic
local DATAPOINT_ISE_ID = 5963;

-- url http xml-api request get value datapoint with ise_id. ise_id set in the function openURL.
-- http response example: <state><datapoint ise_id="5963" value="29.500000"/></state>
local URL_XMLAPI = 'http://ccu-ip/addons/xmlapi/state.cgi?datapoint_id=';
-- define the unique (across all dzVents) callback
local RES_XMLAPI = "RES_HOMEMATIC_DUTYCYCLE_MONITOR";

-- Domoticz dutycycle device (Custom Sensor, type=General, subtype=Percentage)
local IDX_HOMEMATIC_DUTYCYCLE = 301;

-- set the alert message if dutycycle value is above threshold as set by uservariable
TH_HOMEMATIC_DUTYCYCLE
-- (idx,name,type,value): 23, TH_HOMEMATIC_DUTYCYCLE, Integer, 10
local IDX_UV_TH_HOMEMATIC_DUTYCYCLE = 23

-- Timer interval
local TIMERRULE = 'every 15 minutes'
-- local TIMERRULE = 'every minute'

return {
  on = { timer = { TIMERRULE }, httpResponses = { RES_XMLAPI } },
  logging = { level = LOGLEVEL, marker = RES_XMLAPI },
  execute = function(domoticz, item)
    -- check if the triggered item is the timer, then request information
    if (item.isTimer) then
```

```
-- Homematic url http restapi request
domoticz.openURL({url = URL_XMLAPI .. DATAPOINT_ISE_ID, method = 'GET', callback = RES_XMLAPI})
end

-- Check if the item is a httpresponse from the openurl callback
if (item.isHTTPResponse) then
    if (item.statusCode == 200) then
        if (item.callback == RES_XMLAPI) then
            -- Get the key value from the http response:
            -- <state><datapoint ise_id="5963" value="29.50000"/></state>
            local dutycycle = tonumber(domoticz_applyXPath(item.data,
'//datapoint[@ise_id="'.DATAPOINT_ISE_ID..']/@value'))
            domoticz.devices(IDX_HOMEMATIC_DUTYCYCLE).updatePercentage(dutycycle);

            -- Get the threshold
            local threshold = domoticz.variables(IDX_UV_TH_HOMEMATIC_DUTYCYCLE).value
            local message = ('HomeMatic DutyCycle %.0f%% (%d%%)':format(dutycycle, threshold));
            domoticz.log(message)
            -- Check the actual value against the threshold
            if dutycycle > threshold then
                domoticz.helpers.alertmsg(domoticz, domoticz.ALERTLEVEL_RED, message)
            end
        end
    else
        domoticz.log('[ERROR] HomeMatic DutyCycle Request:' .. item.statusText, domoticz.LOG_ERROR)
    end
end
end
}
```

## Solution System Command (Old)

**Note: Not used anymore as replaced by previous solution.**

A Homematic program runs every 5 minutes and updates the Homematic system variable DutyCycle (number, min value -1, max value 100, Unit %)

In Domoticz, the dzVents script “Homematic\_Dutycycle\_Monitor”

- requests from the Homematic system (HTTP REST-API GET Request using CCU-Jack) in regular intervals the value of the System Variable DutyCycle and updates the Domoticz Device “Homematic Duty-Cycle” (%),
- checks if the DutyCycle value is above a threshold, set by a User Variable and updates an Alert sensor to level 4 (Red).

Before explaining the solution, some information on handling HTTP REST-API requests via CCU-Jack.

## HTTP REST-API Requests

Communication between Domoticz and the CCU is based on the addon [CCU-Jack](#). The HTTP REST-API of the addon is used to GET or PUT values from/to the CCU.

The Duty-Cycle value is stored in the Homematic system variable DutyCycle as previously described.

To get the Duty-Cycle value, the ise\_id of the system variable is required.

The ise\_id is obtained from the list of Homematic system variables.

The ise\_id of the system variable DutyCycle is 3032 (see below HTTP Response).

### HTTP GET All System Variables

```
http://ccu-ip:port/sysvar
```

```
{
  "description": "System variables of the ReGaHss", "identifier": "sysvar", "title": "System variables",
  "~links": [
    {"rel": "sysvar", "href": "1518", "title": "svEnergyCounter_1500_0001D3C99C6AB3:6"},
    {"rel": "sysvar", "href": "1236", "title": "${sysVarAlarmZone1}" },
    {"rel": "sysvar", "href": "950", "title": "${sysVarPresence}" },
    {"rel": "sysvar", "href": "2799", "title": "BatteryCheck" },
    {"rel": "sysvar", "href": "3032", "title": "DutyCycle" },
    {"rel": "sysvar", "href": "1517", "title": "svEnergyCounterOldVal_1500" },
    {"rel": "root", "href": "...", "title": "Root" }
  ]
}
```

The ID of the system variable **DutyCycle**: 3032.

### HTTP GET System Variable Value

```
http://ccu-ip:port/sysvar/3032/~pv
```

```
{"ts":1615126980000, "v":2, "s":0}
```

The JSON key "v" holds the Duty-Cycle value 0-100% or -1 if error. Key "s" is not used.

## Homematic Configuration

### System Variable DutyCycle

Name	Description	Variable type	Values	Unit of measurement	Channel assignment	Action	Connections
DutyCycle	DutyCycle CCU3	Number	Minimal value: -1 Maximum value: 100	%		<input type="button" value="Delete"/> <input checked="" type="checkbox"/> visible <input type="button" value="Edit"/> <input type="checkbox"/> logged	<input type="button" value="Programs"/>

Name	Description	Variable type	Values	Unit of measurement	Channel assignment
DutyCycle	DutyCycle CCU3	Number	Value range: Minimal value = -1 Maximum value = 100	%	<input type="radio"/> without <input type="radio"/> with <input type="button" value="Channel selection"/>

## Program

Name	Description	Condition (if...)	Activity (then..., or else...)	Action
Duty Cycle Monitor	Monitor in regular intervals the CCU Duty Cycle.	Time: Recurring All-day starting at 17.02.2021 trigger at point of time	Script: ... immediately run	<input type="checkbox"/> intrinsic

**Condition: If...**  
Time control  AND   
  
Activity: Then...  Stop all current delays before performing the activity (e.g. retriggering).  
Activity: Else...  Stop all current delays before performing the activity (e.g. retriggering).

**Set time module**  
When using the time module, we recommend to use no time intervals shorter than 15 minutes. This ensures that all configured programmes can be executed.

**Time**

- Period
  - Beginning: 08:26
  - All-day
  - Astro function day
  - Astro function night
- Time 08:26

**Serial pattern**

- Non-recurring All [5] Minutes
- Time interval
- Daily
- Weekly
- Monthly
- Yearly

**Validity period**

Start 17.02.2021

No end date  
 Ends after [ ] iterations  
 Ends on [ ]

## Script

```
! duty_cycle_monitor.script
! Get the DutyCycle value (%) by running a script with xmlrpc method
! 20210603

! Run system command to get the duty cycle value
dom.GetObject ("CUxD.CUX2801001:1.CMD_SETS").State("echo 'load tclrpc.so; puts [xmlrpc
http://127.0.0.1:2001/ listBidcosInterfaces ]'|tclsh | awk '{
printf $10 }'");

! Get the query result state with 1 as parameter
dom.GetObject ("CUxD.CUX2801001:1.CMD_QUERY_RET").State(1);

! Get the state object
string objState = dom.GetObject("CUxD.CUX2801001:1.CMD_RETs").State();

! Set the system variable from the state object (as float)
dom.GetObject("DutyCycle").State(objState.ToFloat());

! WriteLine("DutyCycle:#objState.ToFloat());
! DutyCycle:3.000000
```

## Domoticz Configuration

### Device

The Homematic system variable DutyCycle is configured as a Percentage device in Domoticz - Created as a virtual sensor for hardware Dummy named Virtual Sensor.

	Idx	Hardware	ID	Unit	Name	Type	SubType	Data
	301	VirtualSensors	00082301	1	Homematic Duty-Cycle	General	Percentage	2%

### User Variable

A user variable is defined to set the threshold (in pct) of the DutyCycle. If the actual value of the DutyCycle exceeds the threshold, a log entry or an alert is sent.

Idx	Variable name	Variable type	Current value
23	TH_HOMEMATIC_DUTYCYCLE	Integer	10

### Automation Event

The dzVents script requests, via HTTP REST-API GET method, in regular intervals the status of the Homematic system variable DutyCycle (ID: 3032).

The HTTP REST-API response is parsed to get the DutyCycle value from the JSON data and updates the Domoticz Device named "Homematic DutyCycle".

```
-- [[
homematic_dutycycle_monitor.dzvents
Perform regular checks on the Homematic systemvariable "DutyCycle" requested via CCU3 Addon CCU-Jack.
The URL of the Homematic CCU3 is set in the uservariable:
Idx,Variable name,Variable type,Current value
26,URL_HOMEMATIC,String,http://ccu-ip:port
The threshold (in pct) to send an alert message and logerror is set by uservariable
TH_HOMEMATIC_DUTYCYCLE.
Get the list of sysvars:
http://ccu-ip:port/sysvar
Get the ID (href) of the sysvar DutyCycle from previous result JSON Data:
"rel": "sysvar", "href": "3032", "title": "DutyCycle"
Get the value of the sysvar with ID 3032 from result JSON Data:
http://ccu-ip:port/sysvar/3032/~pv
HTTP JSON result: the key "v" holds the dutycycle value 0-100% or -1 if error. Key s is not used.
{"ts":1615126980000,"v":2,"s":0}
Dependencies: Homematic CCU3 Addon CCU-Jack
]]-- 

-- idx of the user variable setting the Homematic ccu ip address & port (without / at the end)
local IDX_UV_URL_HOMEMATIC = 26

-- ise_id of the Homematic sysvar DutyCycle
local HM_SYSVAR_DUTYCYCLE_ID = 3032

-- define the unique (across all dzVents) callback
local RES_HOMEMATIC = "RES_HOMEMATIC_DUTYCYCLE_MONITOR";

-- domoticz dutycycle device
local IDX_HOMEMATIC_DUTYCYCLE = 301;

-- set the alert message if dutycycle value is above threshold as set by uservariable
TH_HOMEMATIC_DUTYCYCLE
-- (idx,name,type,value): 23, TH_HOMEMATIC_DUTYCYCLE, Integer, 10
local IDX_UV_TH_HOMEMATIC_DUTYCYCLE = 23

-- Timer interval
local TIMERRULE = 'every 15 minutes'
-- local TIMERRULE = 'every minute'

return {
```

```
on = {
    timer = { TIMERRULE },
    httpResponses = { RES_HOMEMATIC }
},
execute = function(domoticz, item)
    -- check if the triggered item is the timer, then request information
    if (item.isTimer) then
        -- Homematic url http rest-api request system variable with ise_id
        -- ('http://ccu-ip:port/sysvar/ISE_ID/~pv')
        local url = ('%s/sysvar/%d/~pv'):format(domoticz.variables(IDX_UV_URL_HOMEMATIC).value,
HM_SYSVAR_DUTYCYCLE_ID)
        domoticz.openURL({url = url, method = 'GET', callback = RES_HOMEMATIC,})
    end

    -- Check if the item is a httpresponse from the openurl callback
    -- Select the callback - in this case there is only one (any next developments here might be more)
    if (item.isHTTPResponse) then
        if (item.statusCode == 200) then
            if (item.callback == RES_HOMEMATIC and item.isJSON) then
                -- get the key value from the http JSON response and update the domoticz percentage device
                local value = item.json.v
                domoticz.devices(IDX_HOMEMATIC_DUTYCYCLE).updatePercentage(value);
                -- Get the threshold
                local threshold = domoticz.variables(IDX_UV_TH_HOMEMATIC_DUTYCYCLE).value
                local message = ('Homematic DutyCycle %.0f%%, Grenzwert %d%% (%s)'):format(value, threshold,
domoticz.helpers.isnowshort(domoticz));
                -- Check the actual value against the threshold
                if value > threshold then
                    domoticz.helpers.alertmsg(domoticz, domoticz.ALERTLEVEL_RED, message)
                end
            end
        else
            domoticz.log('[ERROR] Request: ' .. item.statusText, domoticz.LOG_ERROR)
        end
    end
}
}
```

# E-Paper Status Display (HmIP-WRCD)

## Purpose

To view on a small e-paper display, information grouped by topic, like Weather, Information, Covid-19 and more.

## Solution

The information is shown on a Homematic IP Wall-mount Remote Control with status display (HmIP-WRCD). The device has been bought as a kit and assembled.

	<p>HmIP-WRCD device displaying Domoticz Weather ("Wetter") data.</p> <p><u>Display Rows</u></p> <ul style="list-style-type: none"> <li>Row 1 = Title (or Header)</li> <li>Row 2-4 = Domoticz device data</li> <li>Row 5 = Current date</li> </ul>
	<p>As previous Weather data page, but</p> <ul style="list-style-type: none"> <li>• with an icon for the Pressure (changes depending on Domoticz device forecast value)</li> <li>• Text LEFT aligned</li> <li>• Special character % for RH Humidity</li> </ul>

The data to be displayed is put together and triggered to the CCU by a Domoticz automation event (dzVents).

The CCU picks up the updated data and refreshes the content of the status display.  
In Domoticz several data pages are defined and can be set via a selector switch (Virtual Sensor).

It is also possible to change the page via the two push-buttons of the status display – push-button top is the Weather page, push-button bottom the Covid-19 information page.  
Communication between the CCU and Domoticz v.v. is via HTTP REST-API of the CCU Addon CCU-Jack.

The trigger for the Domoticz event to action, is either via

Trigger	Control	Description
Custom Event	Homematic Push-button	Handles the HmIP-WRCD channel data 1 or 2. Example trigger page 2 via buttonnr 2: { "channel": "HmIP-WRCD 002A5D8989D599:2", "buttonnr":2 } The Homematic HmIP-WRCD device push-buttons 1 and 2 are used to select max 2 pages: Weather or Covid-19 status information.
Timer	Domoticz	Updates the selected page in regular intervals.
Selector Switch	Domoticz Virtual sensor	Select a page. The Domoticz selector switch supports more than 2 pages as with the HmIP-WRCD push-buttons.

*Hint*  
Do not switch to fast between pages, the device needs time to update.  
Wait till the display is built.  
  
Whilst setting a new line(s), the display is updating few times.

## Homematic Configuration

### Device

The device HmIP-WRCD is teached in and renamed to “Coffee Machine PSM”.

#### Screenshot of the device listed in the devices (Menu Status and control)

Name	Room	Function	Last modified	Control
Filter	Filter	Filter		
HmIP-WRCD 002A5D8989D599:3				Display configuration

The channel HmIP-WRCD 002A5D8989D599:3 is used to configure the display via Domoticz.

## System Variable

A system variable, named “StatusDisplayCombinedParameter” is used to save the combined parameter to set the display.

The content of the system variable is set by Domoticz HTTP CCU-Jack REST-API request.

Edit system variable					
Name	Description	Variable type	Values	Unit of measurement	Channel assignment
StatusDisplayCombinedPar	Combined parameter (com)	Character string ▾			<input checked="" type="radio"/> without <input type="radio"/> with Channel selection
StatusDisplayCombinedParameter	Combined parameter (command) to set the status display.	30.05.2021 07:30:00			<pre>{DDBC=BLACK,DDTC=WHITE,DDI=0,DDA=CENTER,DDS=Wetter,DDI=1},  {DDBC=WHITE,DDTC=BLACK,DDI=0,DDA=CENTER,DDS=Temp:9.3,DDID=2},  {DDBC=WHITE,DDTC=BLACK,DDI=0,DDA=CENTER,DDS=Feuchte:89,DDID=3},  {DDBC=WHITE,DDTC=BLACK,DDI=0,DDA=CENTER,DDS=Druck:1027,DDID=4},  {DDBC=WHITE,DDTC=BLACK,DDI=0,DDA=CENTER,DDS=30.05.2021,DDID=5,DDC=true}</pre>

*Example content Combined Parameter for 5 rows with commit command.*

## Program Status Display Update

To update the status display, by running a script, when the content of the system variable "StatusDisplayCombinedParameter" is updated.

## Logic

Name	Description	Programs & CCU connection	Condition (if...)	Activity (then..., or else...)	Action
Status Display Update			System status: StatusDisplayCombinedParameter when <i>within value range</i> from and less than trigger when updated	Script: ... immediately run	<input type="checkbox"/> intrinsic
<b>Condition: If...</b>					
System state   StatusDisplayCombinedParameter when <input type="text"/> trigger when updated OR System state   Presence when <input type="text"/> check only OR System state   Presence when <input type="text"/> check only					
  Activity: Then... <input checked="" type="checkbox"/> Stop all current delays before performing the activity (e.g. retriggering). Script   !Script: homematic-status-display-update.script !Update all... <input type="text"/> immediately					
 Activity: Else... <input type="checkbox"/> Stop all current delays before performing the activity (e.g. retriggering).					

To ensure the script is activated, the “Presence” system variable is checked against state present and absent.

## Script

### This script

- gets the content of the system variable “StatusDisplayCombinedParameter”
- updates the value of the status display datapoint “HmIP-RF.002A5D8989D599:3.COMBINED\_PARAMETER”.
- update result is logged to the system variable “DomoticzLog”  
Example “Status Display Update: 30.05 07:30:OK”.

```
! Script: homematic-status-display-update.script
! Update all lines of the Status-display HmIP-WRCD.
! Command to update the display is set by system variable StatusDisplayCommand (ise_id 4930), String
! The result of updating the display using the COMBINED_PARAMETER is logged in the sysvar DomoticzLog
! See README.md
! 20210527 rwbl

! Define Testmode true or false
boolean TESTMODE = true;
if (TESTMODE == true) { WriteLine("TESTMODE"); }

! Constants
string CHECK_DDC_PARAMETER = "DDC=true";

! Result object from GetObject
var resultObject;

! Get the actual date and time DD.MM HH:MM
string actDate = system.Date("%d.%m%"); ! sDate = "09.08"; "%d.%m.%Y" = "09.08.2020";
string actTime = system.Date("%H:%M%"); ! sTime = "07:32"; "%H:%M:%S") = "07:32:00";

! Declare sysvar DomoticzLog holding the result of updating the display state with the combined
parameter
string sysvarDomoticzLogName = "DomoticzLog";
string sysvarDomoticzLogValue = "OK";
string logPrefix = "Status Display Update: ";

! Declare status-display device datapoint to set the state with he combined parameter
string deviceDatapoint = "HmIP-RF.002A5D8989D599:3.COMBINED_PARAMETER";

! Declare the name of the sysvar holding the display combined parameter
string sysvarCombinedParameterName = "StatusDisplayCombinedParameter";

! Declare the var holding the combined parameter value as string
string sysvarCombinedParameterValue = "";

! Get the value of the sysvar - this is the combined parameter
! In TESTMODE the value is direct taken from the sysvar name, else the "$src$" value
if (TESTMODE == true) { sysvarCombinedParameterValue =
dom.GetObject(sysvarCombinedParameterName).Value(); }
! Read the combined parameter from the source, i.e., sysvar
if (TESTMODE == false) { sysvarCombinedParameterValue = dom.GetObject("$src$").Value(); }
```

```

! Replace special characters if combined parameter submitted via url (xml-api) = %2C or %2c=Comma ,
sysvarCombinedParameterValue = sysvarCombinedParameterValue.Replace("%2c", ",");
sysvarCombinedParameterValue = sysvarCombinedParameterValue.Replace("%2C", ",");

if (TESTMODE == true) { WriteLine(sysvarCombinedParameterName # ", Value=" #
sysvarCombinedParameterValue); }

! Set the device state, i.e. update with the combined parameter. important to commit with DDC=true
! Example: dom.GetObject("HmIP-
RF.SERIENNR:3.COMBINED_PARAMETER").State("{DDBC=WHITE,DDTC=BLACK,DDI=1,DDA=CENTER,DDS=Zeile1,DDID=1},{D
DBC=WHITE,DDTC=BLACK,DDI=2,DDA=CENTER,DDS=Zeile2,DDID=2},{DDBC=WHITE,DDTC=BLACK,DDI=3,DDA=CENTER,DDS=Ze
ile3,DDID=3},{DDBC=WHITE,DDTC=BLACK,DDI=5,DDA=CENTER,DDS=Zeile4,DDID=4},{DDBC=WHITE,DDTC=BLACK,DDI=3,DD
A=CENTER,DDS=Zeile5,DDID=5,DDC=true},{R=1,IN=5,ANS=4}")"
if (TESTMODE == true) { WriteLine("Progres - Set State ..."); }

! Check if the combined parameter has at least the command DDC=true
if (sysvarCombinedParameterValue.Contains(CHECK_DDC_PARAMETER) == false) {
    sysvarDomoticzLogValue = "ERROR - Check combined Parameter:#sysvarCombinedParameterValue;
}
else {
    ! Set the state
    resultObject = dom.GetObject(deviceDatapoint).State(sysvarCombinedParameterValue);
    if (resultObject) {
        ! OK
    } else {
        sysvarDomoticzLogValue = "ERROR - Status Display Update. Check combined parameter.";
    }
}
! Add datetime & prefix
sysvarDomoticzLogValue = logPrefix + actDate + " " + actTime + ":" + sysvarDomoticzLogValue;

if (TESTMODE == true) { WriteLine("SysVar Update:#sysvarDomoticzLogName#",
Value="#sysvarDomoticzLogValue"); }
resultObject = dom.GetObject(sysvarDomoticzLogName).State(sysvarDomoticzLogValue);
if (TESTMODE == true) {
    if (resultObject) { WriteLine(dom.GetObject(sysvarDomoticzLogName).Value()); }
}

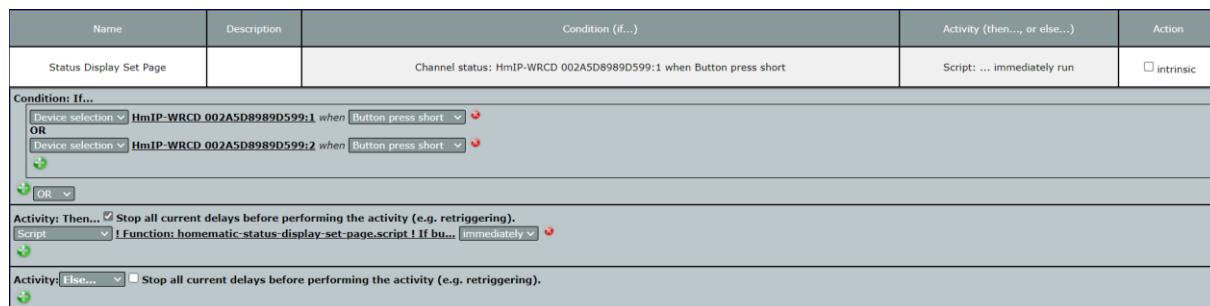
! Close
if (TESTMODE == true) { WriteLine("DONE"); }

```

## Program Status Display Set Page

To update the status display, by running a script, when push-button 1 or 2 is pressed.

### Logic



The channel 1 is for push-button 1 (bottom) and 2 for push-button 2 (top).

### Script

```

! Function: homematic-status-display-set-page.script
! If buttons 1-2 of the status display HmIP-WRCD is pressed, send custom event with buttonnr to
Domoticz to action accordingly.
! Via cuxd an http api request for a customevent is submitted to domoticz.
! The Domoticz custom event data parameter must be in json format.
! Example: {[{"channel"]="HmIP-RC8 000B1BE98D94DE:1", ["buttonnr"]=1}
! 20210528 rwbl

```

```
string cAmp = "&";  
! Domoticz production system url base for the http api request  
string urlBase = "http://domoticz-ip:8080/json.htm";  
! string urlBase = "http://domoticz-ip:8080/json.htm";  
! Custom event name = MUST match name of the Domoticz dzVents script  
string customEvent = "homematic_status_display_set_page";  
  
! Get the object datapoint of the remote control from $src$  
object objDatapoint = dom.GetObject ("$src$");  
if (objDatapoint) {  
    ! Channelobject 1-2 according buttons 1-2, i.e., HmIP-WRCD 002A5D8989D599:1 or :2  
    var objChannel = dom.GetObject(objDatapoint.Channel());  
    ! Get the buttonnr 1-2 from the channel  
    integer buttonNr = (objChannel.Name()).StrValueByIndex(":",1).ToInteger();  
  
    ! Custom event data parameter must be in json format  
    string data = '{"channel":"' + objChannel + '","buttonnr":"' + buttonNr + '"}';  
    ! Build the Domoticz http rest request url to update the device using customevent  
    string urlRequest =  
urlBase#"?type=command#&param=customevent#&event="#customEvent#&data="#data#"";  
    ! Run the command without a return result  
    var cmdRes = dom.GetObject("CUxD.CUX2801001:1.CMD_EXEC").State("wget -q -O - "#urlRequest);  
    WriteLine("CMD_EXEC: " # cmdRes);  
}
```

# Domoticz Configuration

## Purpose

To populate the data (“Combined Parameter”) to be displayed and sent the data to the Homematic System Variable to update the display. The data update can be triggered by a custom event (Homematic program), Timer (Domoticz event), Selector Switch (Domoticz).

## Automation Event

### Script Generic Version

```
--[[[  
    Homematic_status_display_set_page.dzvents  
    Set the page of the homematic status display HmIP-WRCD.  
    There are 3 triggers:  
        Customevent - Homematic - Handles the channel data: {"channel":"HmIP-WRCD  
002A5D8989D599:2","buttonnr":2}  
        Timer - Domoticz - Updates the selected page in regular intervals  
        Selector Switch - Domoticz - Select a page (do not switch to fast between pages, the device needs  
time to update)  
  
    Page Configuration  
    The pages are defined in a lua table array with elements describing each page.  
    n - Page Number = 1,2  
    rowN - describes the rows N 1-5  
        text - text to display (mandatory)  
            Get the device value from the device attribute rawData (Table). By default the first table  
element is taken.  
            Examples are Temp+Hum = {"14.6", "71", "3"}, Airpressure = {"1026", "1"}. The last element  
is not used  
        Placeholders: #DATETIME# (current date in format DD-MM HH:MM), #DATE# (current date YYYY-  
MM-DD),#TIME# (current time HH:MM), #NAME# (domoticz device name)  
        idx - domoticz device idx from which attribute rawData is used (mandatory)  
        index - Index of the rawdata entry. Start with 1.  
        icon - icon number according the HmIP-WRCD documentation (mandatory)  
        bc - Background color WHITE or BLACK (optional)  
        tc - Text color WHITE or BLACK (optional)  
    The last row holds the commit command DDC=true.  
  
    The display is updated by submitting an HTTP REST-API POST request to the CCU to update the system  
variable "StatusDisplayCombinedParameter" (sysvar ise_id 5097).  
    The sysvar holds the channel attribute COMBINED_PARAMETER.  
    The CCU program "Status Display Update" runs when the sysvar 5097 gets updated.  
    The script assigned to the CCU program, updates the HmIP-WRCD device state with the  
COMBINED_PARAMETER content.  
  
    Dependencies: Homematic CCU Addon CCU-Jack REST-API  
    20210529 rwbl  
]]--  
  
-- Domoticz selector switch to select a page - page 1=level 10,page 2=level 20  
local IDX_STATUSDISPLAYPAGE = 371  
  
-- Set the combined parameter direct to the device hmip-wrcd  
-- local HTTP_REQ = 'http://ccu-ip:2121/device/002A5D8989D599/3/COMBINED_PARAMETER/~pv'  
-- Set the combined parameter via the sysvar  
local HTTP_REQ = 'http://ccu-ip:2121/sysvar/5097/~pv'  
-- HTTP unique response  
local HTTP_RES = 'CCUJACKWRCD'  
-- Custom event triggered by the homematic script  
local CUSTOMEVENT = 'homematic_status_display_set_page'  
  
-- Define the devices for the pages  
local MAXPAGES = 2  
-- idx=0 - device not used; icon=0 - no icon; text=space - delete line; text can be used to set prefix  
info.  
local pages = {
```

```

{nr = 1, row1={text='7-Tage Inzidenz',idx=0,icon=0,bc='BLACK',tc='WHITE'}, row2={text='#NAME#',
',idx=367,icon=0}, row3={text='#NAME# ',idx=369,icon=0}, row4={text='#NAME#',
',idx=370,icon=0},row5={text='#DATE#',idx=0,icon=0}, state = false},
 {nr = 2, row1={text='Wetter',idx=0,icon=0,bc='BLACK',tc='WHITE'}, row2={text='Temp:
',idx=344,icon=0,index=1,align='LEFT'}, row3={text='Feuchte: ',idx=344,icon=0,index=2},
row4={text='Druck: ',idx=116,icon=0},row5={text='#DATE#',idx=0,icon=0}, state = true},
 {nr = 3, row1={text='Information',idx=0,icon=0,bc='BLACK',tc='WHITE'}, row2={text='#NAME#',
',idx=82,icon=0}, row3={text='#DATETIME#',idx=0,icon=0},
row4={text='#TIME#',idx=0,icon=0},row5={text='#DATE#',idx=0,icon=0}, state = false},
}

-- Define the combined parameter with placeholders for Icon(DDI),Text(DDS),Row(DDID). DDC is true to
commit.
local CMDROW = '{DDBC=#DDBC#,DDTC=#DDTC#,DDI=#DDI#,DDA=#DDA#,DDS=#DDS#,DDID=#DDID#}'
local CMDDDC = ',DDC=true'

-- Timer rule updating the page selected
local TIMERRULE = 'at 07:30'

-- 
-- FUNCTIONS
-- 

-- Helper - Return the date time in format DD-MM HH:MM with leading zeros - 29-05 11:19
function getDateDateTimeShort(domoticz)
    return ('%s.%s %s:%s'):format(
        string.format("%02d", domoticz.time.day),
        string.format("%02d", domoticz.time.month),
        string.format("%02d", domoticz.time.hour),
        string.format("%02d", domoticz.time.minutes))
end

function getDate(domoticz)
    return ('%s.%s.%s'):format(
        string.format("%02d", domoticz.time.day),
        string.format("%02d", domoticz.time.month),
        string.format("%02d", domoticz.time.year))
end

function getTime(domoticz)
    return ('%s:%s'):format(
        string.format("%02d", domoticz.time.hour),
        string.format("%02d", domoticz.time.minutes))
end

-- Set the combined parameter for a row
local function setRow(domoticz, rownr, rowdata)
    local rowstr = CMDROW -- Command string with placeholder
    local index = 1 -- Index of the rawData table, i.e. the first element as default
    local value = '' -- Value of the device taken from the rawData table index
    local bc = 'WHITE' -- Background color of the row
    local tc = 'BLACK' -- Textcolor of the row
    local align = 'CENTER'
    -- Check the attributes
    if rowdata.index == nil then rawData.index = index end
    if rawData.bc == nil then rawData.bc = bc end
    if rawData.tc == nil then rawData.tc = tc end
    if rawData.align == nil then rawData.align = align end
    -- Replace the colors placeholder DDBC, DDTC
    rowstr = string.gsub(rowstr, '#DDBC#', rawData.bc)
    rowstr = string.gsub(rowstr, '#DDTC#', rawData.tc)
    -- Replace the text alignment placeholder DDA
    rowstr = string.gsub(rowstr, '#DDA#', rawData.align)
    -- Replace the rownumber placeholder DDID
    rowstr = string.gsub(rowstr, '#DDID#', rownr)
    -- Replace the icon placeholder DDI
    rowstr = string.gsub(rowstr, '#DDI#', tostring(rawData.icon))
    -- Get the device value from the device attribute rawData
    if rawData.idx > 0 then
        -- Get the value from the table rawData - use index 1,2...
        value = domoticz.devices(rawData.idx).rawData[rawData.index]
        -- domoticz.log(domoticz.devices(rawData.idx).rawData)
    end
    -- Replace the string (=text) with text + device value
    -- Get the text from the rawData

```

```

local text = rowdata.text
-- Replace text placeholder - ensure to use the right case for the device attributes & methods
if string.match(text, '#DATETIME#') then text = string.gsub(text, '#DATETIME#',
getDateTimeShort(domoticz)) end
if string.match(text, '#DATE#') then text = string.gsub(text, '#DATE#', getDate(domoticz)) end
if string.match(text, '#TIME#') then text = string.gsub(text, '#TIME#', getTime(domoticz)) end
if string.match(text, '#NAME#') and rowdata.idx > 0 then text = string.gsub(text, '#NAME#',
domoticz.devices(rowdata.idx).name) end
-- Replace the text placeholder DDS
rowstr = string.gsub(rowstr, '#DDS#', text .. value)
-- Built the final row string
rowstr = rowstr .. ','
return rowstr
end

-- Set the display page with 5 rows.
-- The data is taken from the global table pages.
-- The data for the last row is amended with the commit command DDC=true.
local function setPage(domoticz, pagenr)
local data = {}
local url = ""
domoticz.log('Setting page: %d'):format(pagenr)
for key, value in pairs(pages) do
if (value.nr == pagenr) then
-- domoticz.log(value)
local rows = ''
rows = rows .. setRow(domoticz, 1, value.row1)
rows = rows .. setRow(domoticz, 2, value.row2)
rows = rows .. setRow(domoticz, 3, value.row3)
rows = rows .. setRow(domoticz, 4, value.row4)
rows = rows .. setRow(domoticz, 5, value.row5)
-- Replace the last characters "}," with ",DDC=true}" to trigger commit
rows = rows:sub(0, -3) .. CMDDDC
-- Define the postdata for the http rest-api request using homematic addon ccu-jack
local spData = { v = rows }
-- domoticz.log(spData)
domoticz.openURL({url = HTTP_REQ, method = 'PUT', callback = HTTP_RES, postData = spData})
end
end
end

-- MAIN
return {
on = {
timer = { TIMERRULE },
customEvents = { CUSTOMEVENT },
devices = { IDX_STATUSDISPLAYPAGE }
},
data = { pageSelected = { initial = 0 } },
logging = {},

execute = function(domoticz, triggeredItem)
-- domoticz.log(domoticz.data)
-- Homematic script trigger
if (triggeredItem.isCustomEvent) then
domoticz.data.pageSelected = triggeredItem.json.buttonnr
setPage(domoticz, domoticz.data.pageSelected)
end

-- Update the display depending page with the device items (group)
if (triggeredItem.isTimer) then
setPage(domoticz, domoticz.data.pageSelected)
end

-- Update the display depending page selected from the page selector switch
if (triggeredItem.isDevice and triggeredItem.idx == IDX_STATUSDISPLAYPAGE) then
-- Convert the selector switch level 0,10,20... to 0,1,2...
domoticz.data.pageSelected = math.floor(triggeredItem.rawData[1] * 0.1)
setPage(domoticz, domoticz.data.pageSelected)
end

end
}
}

```

## Device Attribute rawData

The automation event takes the device rawData attribute to select the value for a row. The reason for taking the rawData attribute, is that a device can have multiple values, like the device types of the examples.

Device Type	Device Attributes	Example
Temp + Humidity	Temperature (Degrees) Humidity (%RH) Status (N)	{"16.4", "66", "1"}
Wind	Bearing (Degrees) Direction (N, S, NNW etc) Speed (m/s) Gust (m/s) Temperature (Celsius) Chill (Celsius)	{"112.00", "ESE", "6", "0", "22.7", "22.7"}

## Device Attributes Logger

### Domoticz Automation Event dzVents Script

The example listens to a specific device idx changes and logs the rawData attribute. To log all the selected device rawData attributes, use a timer, rule 'every minute' and run once.

```
local IDX_DEVICE = 344

-- Timer Rule used to log all devices data
-- local TIMERRULE = 'every minute'

local function logDevice(domoticz, device)
    domoticz.log('%s: %s-%s'):format(device.name, device.deviceType, deviceSubType))
    domoticz.log(device.rawData)
    -- domoticz.log(device.dump())
end

return {
    on = { timer = { TIMERRULE }, devices = { IDX_DEVICE } },
    logging = { level = domoticz.LOG_INFO, marker = 'DEVICESRAWDATA' },
    execute = function(domoticz, item)
        -- All devices
        if (item.isTimer) then
            domoticz.devices().forEach(function(device)
                logDevice(domoticz, device)
            end)
        end
        -- Single device when changed
        if (item.isDevice and item.idx == IDX_DEVICE) then
            logDevice(domoticz, item)
        end
    end
}
```

## Display Page Configuration

In the automation event, the pages to be displayed are defined in a Lua table array with elements describing each page.

The page configuration Lua table with multiple entries.

```
local pages = {Page1, ..., PageN}
```

### Page Definition

```
{
  nr = NN,
  row1 = {text='ROWTEXT', idx=NNNN, icon=NN, bc='BLACK', tc='WHITE', align='LEFT'},
  row2 = {text='ROWTEXT', idx=NNNN, icon=NN},
  row3 = {text='ROWTEXT', idx=NNNN, icon=NN},
  row4 = {text='ROWTEXT', idx=NNNN, icon=NN},
  row5 = {text='ROWTEXT', idx=NNNN, icon=NN},
  state = false},
  DDC=true
}
```

Element	Description	Attribute	Description
nr	Page number N. N = 1 .. N		
rowN	Row description. N = row number 1 – 5	text (mandatory)	<p>The text to display. Get the device value from the device attribute rawData (Lua Table). By default, the first table element is taken. Examples are</p> <pre>Temp+Hum = {"14.6", "71", "3"} Airpressure = {"1026", "1"}</pre> <p>The last rawData element “Status” is not used Placeholders:</p> <ul style="list-style-type: none"> <li>• #DATETIME# (current date in format DD-MM HH:MM)</li> <li>• #DATE# (current date YYYY-MM-DD)</li> <li>• #TIME# (current time HH:MM)</li> <li>• #NAME# (Domoticz device name)</li> </ul>
		idx (mandatory)	Domoticz device idx from which attribute rawData is used.
		index (optional)	Index of the rawData entry. Starts with 1.
		icon (mandatory)	Icon number according the HmIP-WRCD documentation.
		bc (optional)	Background color WHITE or BLACK Define in uppercase. Default: WHITE
		tc (optional)	Text color WHITE or BLACK Define in uppercase. Default: BLACK
		align (optional)	Text row alignment: LEFT, CENTER, RIGHT Default: CENTER
state	true   false		Not used. Planned next version.
			The last row holds the commit command

		DDC=true.
--	--	-----------

## Script Custom Page Version

This is an enhanced version of the generic script. It has page specific device actions, i.e., the function “setPageWeather” using the row number for the device actions to be displayed on a row.

The script below uses all previous script declarations and functions, with the additional age number declarations, the function “setPageWeather” and the variable “dopage” to set the selected page.

In the function “setPageWeather”, the default table elements for the rows are overwritten.

The timer runs every hour during daytime:

```
local TIMERRULE = 'every hour between 07:00 and 20:00'
```

The result display is shown in this chapter, section [Purpose](#) 2<sup>nd</sup> picture.

*BTW*

A gimmick for the weather page: if outside temperature > 25 the flame icon is displayed.

```
if (domoticz.devices(value.row2.idx).temperature > 25) then value.row2.icon = 23 end -- very hot=flame

--[[[
    homematic_status_display_set_page.dzvents
    Set the page of the homematic status display HmIP-WRCD.
    There are 3 triggers:
        Customevent - Homematic - Handles the channel data: {"channel":"HmIP-WRCD
002A5D8989D599:2","buttonnr":2}. The channel attribute is for info only.
        Timer - Domoticz - Updates the weather page in regular intervals (see TIMERRULE)
        Selector Switch - Domoticz - Select a page (do not switch to fast between pages, the device needs
time to update)

    Page Configuration
    The pages are defined in a lua table array with elements describing each page.
    n - Page Number = 1,2
    rowN - describes the rows N 1-5
        text - text to display (mandatory)
            Get the device value from the device attribute rawData (Table). By default the first table
element is taken.
            Examples are Temp+Hum = {"14.6", "71", "3"}, Airpressure = {"1026", "1"}. The last element
"status" is not used.
            Placeholders: #DATETIME# (current date in format DD-MM HH:MM), #DATE# (current date YYYY-
MM-DD),#TIME# (current time HH:MM), #NAME# (domoticz device name).
        idx - domoticz device idx from which attribute rawData is used (mandatory)
        index - Index of the rawdata entry. Start with 1.
        icon - icon number according the HmIP-WRCD documentation (mandatory)
        bc - Background color WHITE or BLACK (optional)
        tc - Text color WHITE or BLACK (optional)
    The last row holds the commit command DDC=true.

    There is a special page setPageWeather - page number 2.
    // Workout other special pages

    The display is updated by submitting an HTTP REST-API POST request to the CCU to update the system
variable "StatusDisplayCombinedParameter" (sysvar ise_id 5097).
    The sysvar holds the channel attribute COMBINED_PARAMETER.
    The CCU program "Status Display Update" runs when the sysvar 5097 gets updated.
    The script assigned to the CCU program, updates the HmIP-WRCD device state with the
COMBINED_PARAMETER content.

    Dependencies: Homematic CCU Addon CCU-Jack REST-API
    20210606 rwbl
]]--


-- Domoticz selector switch to select a page - page 1=level 10,page 2=level 20
local IDX_STATUSDISPLAYPAGE = 371

-- Set the combined parameter direct to the device hmip-wrcd
-- local HTTP_REQ = 'http://ccu-ip:2121/device/002A5D8989D599/3/COMBINED_PARAMETER/~pv'
```

```

-- Set the combined parameter via the sysvar
local HTTP_REQ = 'http://ccu-ip:2121/sysvar/5097/~pv'
-- HTTP unique response
local HTTP_RES = 'CCUJACKWRCD'
-- Custom event triggered by the homematic script
local CUSTOMEVENT = 'homematic_status_display_set_page'

-- Define the devices for the pages
local MAXPAGES = 2
local COVIDPAGE = 1
local WEATHERPAGE = 2
local INFOPAGE = 3
-- idx=0 - device not used; icon=0 - no icon; text=space - delete line; text can be used to set prefix
info.
local pages = {
    {nr = COVIDPAGE, row1={text='7-Tage Inzidenz',idx=0,icon=0,bc='BLACK',tc='WHITE'},
    row2={text='#NAME#: ',idx=367,icon=0}, row3={text='#NAME#: ',idx=369,icon=0}, row4={text='#NAME#',
    'idx=370,icon=0},row5={text='#DATE#',idx=0,icon=0}, state = false},
    {nr = WEATHERPAGE, row1={text='Wetter',idx=0,icon=0,bc='BLACK',tc='WHITE'}, row2={text='Temp:
    ',idx=344,icon=0,index=1,align='CENTER'}, row3={text='Feuchte: ',idx=344,icon=0,index=2},
    row4={text='Druck: ',idx=116,icon=0},row5={text='#DATETIME#',idx=0,icon=0}, state = true},
    {nr = INFOPAGE, row1={text='Information',idx=0,icon=0,bc='BLACK',tc='WHITE'}, row2={text='#NAME#':
    ',idx=0,icon=0}, row3={text='#DATETIME#',idx=0,icon=0},
    row4={text='#TIME#',idx=0,icon=0},row5={text='#DATE#',idx=0,icon=0}, state = false},
    }

-- Define the combined parameter with placeholders for Icon(DDI),Text(DDS),Row(DDID). DDC is true to
commit.
local CMDROW = '{DDBC=#DDBC#,DDTC=#DDTC#,DDI=#DDI#,DDA=#DDA#,DDS=#DDS#,DDID=#DDID#}'
local CMDDDC = ',DDC=true'

-- Timer rule updating the page selected
local TIMERRULE = 'every hour between 07:00 and 20:00'
-- local TIMERRULE = 'at 07:30'

--
-- FUNCTIONS
--

-- Helper - Return the date time in format DD-MM HH:MM with leading zeros - 29-05 11:19
function getDateTimeShort(domoticz)
    return ('%s.%s %s:%s'):format(
        string.format("%02d", domoticz.time.day),
        string.format("%02d", domoticz.time.month),
        string.format("%02d", domoticz.time.hour),
        string.format("%02d", domoticz.time.minutes))
end

function getDate(domoticz)
    return ('%s.%s.%s'):format(
        string.format("%02d", domoticz.time.day),
        string.format("%02d", domoticz.time.month),
        string.format("%02d", domoticz.time.year))
end

function getTime(domoticz)
    return ('%s:%s'):format(
        string.format("%02d", domoticz.time.hour),
        string.format("%02d", domoticz.time.minutes))
end

-- Set the combined parameter for a row
local function setRow(domoticz, rownr, rowdata)
    local rowstr = CMDROW -- Command string with placeholder
    local index = 1 -- Index of the rawData table, i.e. the first element as default
    local value = '' -- Value of the device taken from the rawData table index
    local bc = 'WHITE' -- Background color of the row
    local tc = 'BLACK' -- Textcolor of the row
    local align = 'CENTER' -- Check the attributes
    if rowdata.index == nil then rowdata.index = index end
    if rowdata.bc == nil then rowdata.bc = bc end
    if rowdata.tc == nil then rowdata.tc = tc end
    if rowdata.align == nil then rowdata.align = align end
    -- Replace the colors placeholder DDBC, DDTC

```

```

rowstr = string.gsub(rowstr, '#DDBC#', rowdata.bc)
rowstr = string.gsub(rowstr, '#DDTC#', rowdata.tc)
-- Replace the text alignment placeholder DDA
rowstr = string.gsub(rowstr, '#DDA#', rowdata.align)
-- Replace the rownumber placeholder DDID
rowstr = string.gsub(rowstr, '#DDID#', rownr)
-- Replace the icon placeholder DDI
rowstr = string.gsub(rowstr, '#DDI#', tostring(rowdata.icon))
-- Get the device value from the device attribute rawData
if rowdata.idx > 0 then
    -- Get the value from the table rawdata - use index 1,2...
    value = domoticz.devices(rowdata.idx).rawData[rowdata.index]
    -- domoticz.log(domoticz.devices(rowdata.idx).rawData)
end
-- Replace the string (=text) with text + device value
-- Get the text from the rowdata
local text = rowdata.text
-- Replace text placeholder - ensure to use the right case for the device attributes & methods
if string.match(text, '#DATETIME#') then text = string.gsub(text, '#DATETIME#',
getDateTimeShort(domoticz)) end
if string.match(text, '#DATE#') then text = string.gsub(text, '#DATE#', getDate(domoticz)) end
if string.match(text, '#TIME#') then text = string.gsub(text, '#TIME#', getTime(domoticz)) end
if string.match(text, '#NAME#') and rowdata.idx > 0 then text = string.gsub(text, '#NAME#',
domoticz.devices(rowdata.idx).name) end
-- Replace the text placeholder DDS
rowstr = string.gsub(rowstr, '#DDS#', text .. value)
-- Built the final row string
rowstr = rowstr .. ','
return rowstr
end

-- Set the display page with 5 rows.
-- The data is taken from the global table pages.
-- The data for the last row is amended with the commit command DDC=true.
local function setPage(domoticz, pagenr)
    local data = {}
    local url = ""
    domoticz.log(('Setting page: %d'):format(pagenr))
    for key, value in pairs(pages) do
        if (value.nr == pagenr) then
            -- domoticz.log(value)
            local rows = ''
            rows = rows .. setRow(domoticz, 1, value.row1)
            rows = rows .. setRow(domoticz, 2, value.row2)
            rows = rows .. setRow(domoticz, 3, value.row3)
            rows = rows .. setRow(domoticz, 4, value.row4)
            rows = rows .. setRow(domoticz, 5, value.row5)
            -- Replace the last characters "}" with ",DDC=true}" to trigger commit
            rows = rows:sub(0, -3) .. CMDDDC
            -- Define the postdata for the http rest-api request using homematic addon ccu-jack
            local spData = { v = rows }
            -- domoticz.log(spData)
            domoticz.openURL({url = HTTP_REQ, method = 'PUT', callback = HTTP_RES, postData = spData})
        end
    end
end

-- Weather Page - Set the display page with 5 rows.
-- The data is taken from the global table pages.
-- The data for the last row is amended with the commit command DDC=true.
local function setPageWeather(domoticz, pagenr)
    local data = {}
    local url = ""
    domoticz.log(('Setting Weather Page: %d'):format(pagenr))
    for key, value in pairs(pages) do
        if (value.nr == pagenr) then
            -- domoticz.log(value)
            local rows = ''
            -- Update the row data depending device statatus
            -- ROW 1 = Title
            value.row1.text = '-Wetter-'
            -- value.row1.text = '\x28Wetter\x29'
            value.row1.icon = 0
            value.row1.align = 'CENTER'
            -- ROW 2 = Temperature set flame icon if above 25 C
        end
    end
end

```

```

        value.row2.text = ' T C: '
        if (domoticz.devices(value.row2.idx).temperature > 25) then value.row2.icon = 23 end -- -
very hot = flame
        value.row2.align = 'LEFT'
-- ROW 3 = Humidity
        value.row3.text = ' RH %: '
        if (domoticz.devices(value.row2.idx).temperature > 90) then value.row2.icon = 17 end
        value.row3.align = 'LEFT'
-- ROW 4 = Barometer set forecast icon
        value.row4.text = ' P hPa: '
        -- 0 BARO_STABLE, 1 BARO_SUNNY, 2 BARO_CLOUDY, 3 BARO_UNSTABLE, 4 BARO_THUNDERSTORM
        if (domoticz.devices(value.row4.idx).forecast == 0) then value.row4.icon = 0 end -- no
icon for stable
        if (domoticz.devices(value.row4.idx).forecast == 1) then value.row4.icon = 10 end -- -
sunny
        if (domoticz.devices(value.row4.idx).forecast == 2) then value.row4.icon = 13 end -- -
cloudy
        if (domoticz.devices(value.row4.idx).forecast == 3) then value.row4.icon = 12 end -- wind
        if (domoticz.devices(value.row4.idx).forecast == 4) then value.row4.icon = 14 end -- -
lightning
        value.row4.align = 'LEFT'
        -- domoticz.log(domoticz.devices(value.row4.idx).forecast .. '#' ..
domoticz.devices(value.row4.idx).forecastString .. '#ICON=' .. value.row4.icon .. '#FC=' ..
domoticz.BARO_CLOUDY)
        -- Set the rows
        rows = rows .. setRow(domoticz, 1, value.row1)
        rows = rows .. setRow(domoticz, 2, value.row2)
        rows = rows .. setRow(domoticz, 3, value.row3)
        rows = rows .. setRow(domoticz, 4, value.row4)
        rows = rows .. setRow(domoticz, 5, value.row5)
        -- Replace the last characters "}," with ",DDC=true}" to trigger commit
        rows = rows:sub(0, -3) .. CMDDDC
        -- Define the postdata for the http rest-api request using homematic addon ccu-jack
        local spData = { v = rows }
        -- domoticz.log(spData)
        domoticz.openURL({url = HTTP_REQ, method = 'PUT', callback = HTTP_RES, postData = spData})
    end
end

-- MAIN
return {
on = {
    timer = { TIMERRULE },
    customEvents = { CUSTOMEVENT },
    devices = { IDX_STATUSDISPLAYPAGE }
},
data = { pageSelected = { initial = 0 } },
logging = {},

execute = function(domoticz, triggeredItem)
    local dopage = false
    -- domoticz.log(domoticz.data)
    -- Homematic script trigger
    if (triggeredItem.isCustomEvent) then
        domoticz.data.pageSelected = triggeredItem.json.buttonnr
        dopage = true
    end

    -- Update the weather display
    if (triggeredItem.isTimer) then
        domoticz.data.pageSelected = WEATHERPAGE
        dopage = true
    end

    -- Update the display depending page selected from the page selector switch
    if (triggeredItem.isDevice and triggeredItem.idx == IDX_STATUSDISPLAYPAGE) then
        -- Convert the selector switch level 0,10,20... to 0,1,2...
        domoticz.data.pageSelected = math.floor(triggeredItem.rawData[1] * 0.1)
        domoticz.log("Page selected:" .. domoticz.data.pageSelected)
        dopage = true
    end

    -- Select the page specific function
    if (dopage) then

```

```
        if (domoticz.data.pageSelected == COVIDPAGE) then setPage(domoticz,  
domoticz.data.pageSelected) end  
        if (domoticz.data.pageSelected == WEATHERPAGE) then setPageWeather(domoticz,  
domoticz.data.pageSelected) end  
        if (domoticz.data.pageSelected == INFOPAGE) then setPage(domoticz,  
domoticz.data.pageSelected) end  
    end  
end  
}
```

## Reference Display Configuration

### Syntax Single Line

```
"{DDBC=WHITE|BLACK,DDTC=WHITE|BLACK,DDI=NN,DDA=LEFT|CENTER|RIGHT,DDS=STRING,DDID=1-5,DDC=true}"
```

### Syntax Multiple Lines

As above but with comma separated entries. Last line to close off with DDC=true to commit.

```
"{..DDID=1}, ... , { ..DDID=5, DDC=true}"
```

### Line Keys

Key	Name	Description
DDBC	Background Color	Line Background color WHITE, BLACK Default: WHITE
DDTC	Text Color	Line Text color WHITE, BLACK Default: BLACK
DDI	Icon	Icon after text Default: 0 = No icon See icon list below
DDA	Alignment	Line Alignment LEFT, CENTER, RIGHT Default: CENTER
DDS	String	Line Text (String) Enter SPACE to delete a line (leave empty). (See also special character list below)
DDID	ID	Line number 1 to 5
DDC	Commit	Commit the changes. Set in the last line or leave unset (and set as a separate command) (true)

Each line can be updated separately without changing the other lines.

Multiple lines can be updated within one command, use comma to separate each line. Here an example for a rule file.

## Special Characters & Icons

Special Characters DDS	Special Characters HEX Code	Icons DDI
[ - Ä	- \x20	0 - No Icon
# - Ö	! - \x21	1 - Light off
\$ - Ü	" - \x22	2 - Light on
{ - ä	% - \x25	3 - Locked
- ö	& - \x26	4 - Unlocked
}	= - \x27	5 - X
-	( - \x28	6 - Check
]	) - \x29	7 - Information
' - =	* - \x2A	8 - Envelope
;	+ - \x2B	9 - Spanner,
< - Arrow Down	,	10 - Sun
= - Arrow Up	- - \x2D	11 - Moon
> - Arrow Up Right	. - \x2E	12 - Wind
@ - Arrow Down Right	/ - \x2F	13 - Cloud,
	Ä - \x5B	14 - Cloud/Lightning
	Ö - \x23	15 - Cloud/Light Rain
	Ü - \x24	16 - Cloud/Moon
	ä - \x7B	17 - Cloud/Rain
	ö - \x7C	18 - Cloud/Snow
	ü - \x7D	19 - Cloud/Sun
	ß - \x5F	20 - Cloud/Sun/Rain
	[ - \x5B	21 - Cloud/Snowflake
	] - \x5D	22 - Cloud/Raindrop
	:	23 - Flame
	;	24 - Window Open
	@ - \x40	25 - Roller Shutter
	> - \x3E	26 - Eco
	Example: text = '\x28Wetter\x29' displays (Wetter)	27 - Protection Deactivated
	Be aware that the Special Characters DDS override the HEX code character definition. Example: text = '\x5BWetter\x5D' displays ÄWetterß	28 - Presence 29 - Absent 30 - Bell 31 - Clock

## Audible Signal

This feature is not currently used.

To use the audible signal, define as COMBINED\_COMMAND:

```
{R=0, IN=10, ANS=0}
```

The commit command, DDC=true, is not required for signals.  
The signal command can be combined with line updates

```
"{..DDID=1}, ... , { ..DDID=5, DDC=true}, {R=0,IN=10,ANS=0}"
```

Key	Purpose	Values
R	Repetitions	0 to 15 15 = infinite
IN	Interval	5 to 80 In steps of 5
ANS	Beep sound	-1 to 7 <ul style="list-style-type: none"> <li>• -1 - No Sound</li> <li>• 0 - Empty Battery</li> <li>• 1 - Alarm Off</li> <li>• 2 - External Alarm activated</li> <li>• 3 - Internal Alarm activated</li> <li>• 4 - External Alarm delayed activated</li> <li>• 5 - Internal Alarm delayed activated</li> <li>• 6 - Event</li> <li>• 7 - Error</li> </ul>

## Hints

- Text length: Recommend max text length 14 characters.  
If also an icon is displayed, then max 11 characters else the icon is not displayed.
- Be aware of special characters.  
//TODO: How to display the character used to display a special character. Example how to display the "[" or "]" characters.
- In dzVents define special characters with % prefix. Example: To display % use %%.

## Reference Display Set REST-API

Some examples setting a display line or several lines by using the addons XML-API and CCU-Jack or via Node-RED.

### XML-API

#### **Set Row 2 to string "Hello World" with icon "Unlocked".**

The XML-API “statechange” script is used to set the new value of the system variable with id 5097.

The comma in the URL parameter “new\_value” must be escaped to %2c or %2C and the space to %20.

#### *HTTP XML-API Request*

```
http://ccu-
ip/addons/xmlapi/statechange.cgi?ise_id=5097&new_value={DDBC=BLACK%2CDDTC=WHITE%2cDDI=3%2cDDA=CENTER%2c
DDS>Hello%20World%20%2cDDID=2%2cDDC=true}
```

#### *HTTP XML-API Response*

```
<result>
  <changed id="5097" new_value="{DDBC=BLACK%2CDDTC=WHITE%2cDDI=3%2cDDA=CENTER%2cDDS>Hello World
%2cDDID=2%2cDDC=true}"/>
</result>
```

## Node-RED & CCU-Jack

### Set Row 2 to string "Hello World" with icon "House with person".

Combined parameter submitted via Node-RED flow Inject > Node triggered after 0.1 sec  
Function Node > HTTP Request > Debug Node.

The HTTP Request uses CCU-Jack REST-API with JSON object with name "v" for the value containing the combined parameter.

#### Function Node

```
msg.url="http://ccu-ip:2121/device/002A5D8989D599/3/COMBINED_PARAMETER/~pv";

// DDBC=Row background black, DDTC=text white, DDI=icon 28, DDID=row 2, DDC=true to commit.
var cmd = "{DDBC=BLACK,DDTC=WHITE,DDI=28,DDA=CENTER,DDS>Hello World,DDID=2,DDC=true}";

msg.payload= {"v":cmd};

return msg;
```

#### Node-RED Flow Source

```
[
{"id":"9c8b77fe.4e7b48","type":"inject","z":"b9202e63.88a7","name":"Set HTTP Request
URL","props":[],"repeat":"","crontab":"","once":true,"onceDelay":0.1,"topic":"","x":150,"y":380,"wires":[]},
{"id":"ccc06a93.2fe048"}],{"id":"85b8f1da.ed0d4","type":"http request","z":"b9202e63.88a7","name":"Send
HTTP
Request","method":"PUT","ret":"txt","paytoqs":"ignore","url":"","tls":"","persist":false,"proxy":"","au
thType":"","x":520,"y":380,"wires":[]},
{"id":"be0bb79f.b55438","type":"debug","z":"b
9202e63.88a7","name":"DEBUG HmIP-
WRCD","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"payload","targetType"
:"msg","statusVal":"","statusType":"auto","x":740,"y":380,"wires":[]},
{"id":"ccc06a93.2fe048","type":"f
unction","z":"b9202e63.88a7","name":"","func": "// Set row 2 of the HmIP-WRCD\nmsg.url=\\"http://ccu-
ip:2121/device/002A5D8989D599/3/COMBINED_PARAMETER/~pv\\";
\n// DDBC=Row background black, DDTC=text
white, DDI=icon 28, DDID=row 2, DDC=true to commit.\nvar cmd =
\"{DDBC=BLACK,DDTC=WHITE,DDI=28,DDA=CENTER,DDS>Hello World,DDID=2,DDC=true}\\";
\nmsg.payload=
{\"v\":cmd};\n\nreturn
msg","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":340,"y":380,"wires":[]},
{"id":"ed0d4"}]
```

# Garage Door Monitor (HmIP-SWDM)

## Purpose

- To monitor open & close state of the garage door
- To check daily at 20:00 if the garage door state is open and notify via email

## Solution

### Homematic IP

The solution makes use of the Homematic IP Window/Door Contact with magnet device (HMIP-SWDM).

The HMIP-SWDM is integrated in the CCU.

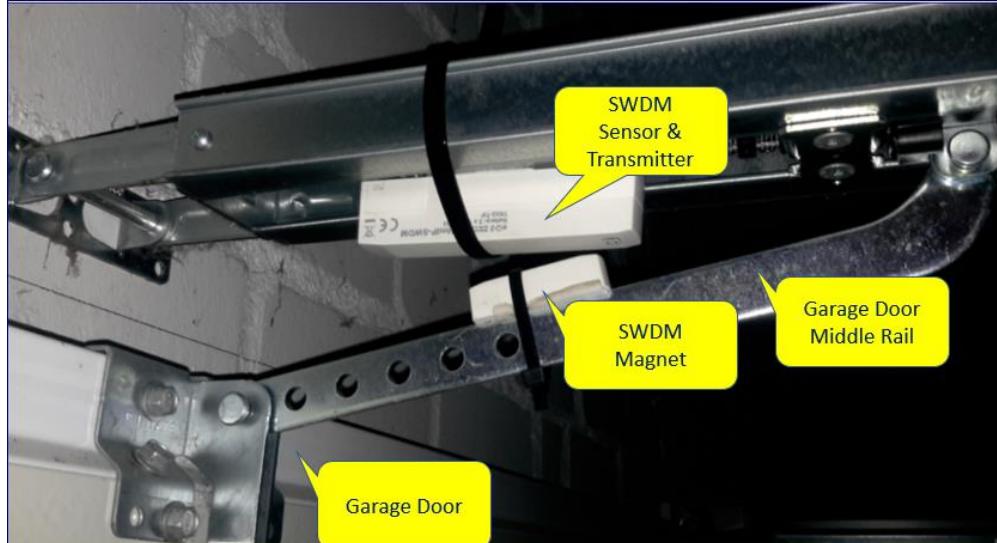
The HMIP-SWDM is mounted inside the garage door using the middle rail. The max distance between magnet and the sensor is ~5mm to ensure a signal is generated when moving the garage door = detecting the magnet. If the magnet is detecting, the LED of the sensor flashes short.

### Domoticz

A Domoticz virtual sensor type Alert, indicates the state of the HMIP-SWDM device:  
Level 1 = Green = Garage Door state closed, Level 4 = Red = Garage Door state open.

## Screenshots

### Garage Door with SWDM Device mounted inside



Requires some fiddling to get the Sensor & Magnet at right position for Open & Close state

## Homematic SWDM Device

The screenshot shows a device configuration window for a "Garagentür" (Garage door). The device is identified as "HmIP-SWDM" with ID "001558A99D5A78:1". It is categorized as a "Window / Door Contact with magnet". The status is "Secured" and "Lock" is selected. The battery level is at 2.9 V and signal strength is -71 dBm. On the right, there are buttons for "Set", "Delete", "Direct", and "Programs". Below this, the main interface shows a list of devices under "Status and control > Devices". One device, "HmIP-SWDM 001558A99D5A78:1", is listed with the name "Garage", room "Filter", function "Lock", last modified "27.03.2022 12:23:33", and control icons for "Open" and "Closed".

## Domoticz Alert Device

The screenshot shows the Domoticz interface for managing sensors. A modal window titled "Create Virtual Sensor" is open, showing fields for "Name" (Garagensor) and "Sensor Type" (Alert). The main table lists a "VirtualSensors" entry with ID 6, Name "VirtualSensors", Enabled "Yes", Type "Dummy (Does nothing, use for virtual switches only)", Address "Disabled", and Port "Disabled". Below this, two sensor cards are shown: one for "Garagensor" indicating it is "Offen (27.03 12:03)" and another for "Garagensor" indicating it is "Geschlossen (27.03 12:23)".

## Process Flow Steps

1. Homematic listens to HMIP-SWDM device channel state changes (Homematic Script).  
Channel status: HmIP-SWDM 001558A99D5A78:1 when open trigger when changed  
...
2. On channel state change, update the Domoticz Alert Sensor via HTTP API Request (Homematic Script)
3. Domoticz Alert Sensor Level indicating garage door state:  
**Green** = Closed, **Red** = Open
4. Domoticz Automation Event (dzVents) checks daily at 20:00 garage door state and notifies via email if open.

## Homematic Script

```

! Function: Garage Door Monitor
! Ensure to escape special characters
! 20220327 rwbl

! Get the actual date and time DD.MM HH:MM
string actDate = system.Date("dd.mm");      ! sDate = "09.08"; "%d.%m.%Y" = "09.08.2020";
string actTime = system.Date("%H:%M");       ! sTime = "07:32"; "%H:%M:%S") = "07:32:00";

! HomematicIP Device
! Get the state & object of the SWDM device via XMLAPI request: http://ccu-
ip/config/xmlapi/statelist.cgi
! <datapoint name="HmIP-RF.001558A99D5A78:1.STATE" type="STATE" ise_id="3622" value="0" valuetype="16"
valueunit="" timestamp="0" operations="5"/>
string swdmState = dom.GetObject("HmIP-RF.001558A99D5A78:1.STATE").State();  ! 0=Closed, 1=Open

! Domoticz Device
! A Domoticz virtual sensor type Alert is used to how the SWDM state
! 324,VirtualSensors,Garagentor,General,Alert,No Alert!
string alertIdx = 324;   ! Type: General; SubType: Alert
string alertLevel = "0";
string alertState = "Unknown";

! Check SWDM state 0=Closed,1=Open
if (swdmState == 0) {
    alertLevel = "1";      ! GREEN
    alertState = "Geschlossen";
}
if (swdmState == 1) {
    alertLevel = "4";      ! RED
    alertState = "Offen";
}
string alertText = alertState#"%20(" # actDate # "%20" # actTime # ")";
! string alertText = "Garagentor%20"#alertState#"%20(" # actDate # "%20" # actTime # ")";
WriteLine(alertText);

! Build the Domoticz http rest request url to update the device
! /json.htm?type=command&m=udevice&idx=IDX&nvalue=LEVEL&svalue=TEXT
! Level: 0=gray, 1=green, 2=yellow, 3=orange, 4=red
string cAmp = "&";
string urlBase = "'http://domoticz-ip:port/json.htm'; ! Production
string urlRequest =
urlBase#"?type=command#cAmp#"param=udevice#cAmp#"idx="#alertIdx#cAmp#"nvalue="#alertLevel#cAmp#"svalu
e="#alertText#'";
WriteLine(urlRequest);

! Run the command without a return result
cmdRes = dom.GetObject("CUxD.CUX2801001:1.CMD_EXEC").State("wget -q -O - "#urlRequest);

```

### Test Execute Script

```

Geschlossen%20(21.08%2009:07)
'http://domoticz-
ip:port/json.htm?type=command&param=udevice&idx=324&nvalue=1&svalue=Geschlossen%20(21.08%2009:07)'

```

## Automation Event

```
--[[  
garage_door_monitor.dzvents  
Trigger: Monitor the state change open or close.  
The Alert Device Garagentor is set to GREEN (level 1) or RED (level 4) by Homematic script (trigger  
Device State Open or Close).  
  
Trigger Timer: Check daily at 2000, if the garagedoor is closed. OPTIONAL: Send email notification if  
open.  
20220327 rwbl  
--]]  
  
-- IDX of the devices used  
local IDX_GARAGEDOOR_ALERT = 324  
local IDX_MESSAGES_ALERT = 55  
  
local DOORCLOSED_STATE = 1 -- alert level 1 green  
local DOOROPEN_STATE = 4 -- alert level 4 red  
  
-- local TIMER_RULE = 'every minute'  
local TIMER_RULE = 'at 20:00'  
  
return {  
    on = {  
        timer = { TIMER_RULE },  
        devices = { IDX_GARAGEDOOR_ALERT }  
    },  
    logging = {  
        level = domoticz.LOG_DEBUG,  
        marker = 'GARAGEDOOR',  
    },  
    execute = function(domoticz, item)  
        -- Check at timestamp if the garagedoor is open  
        if item.isTimer then  
            -- Get state garagedoor alert device  
            local state = domoticz.devices(IDX_GARAGEDOOR_ALERT).nValue  
            domoticz.log('Garagedoor state: %d'):format(state)  
            if (state == DOOROPEN_STATE) then  
                domoticz.log('Garagedoor is open. Please Close...')  
                -- subject,message,mailto  
                -- domoticz.email('Garagentor', 'Garagedoor is OPEN', 'EMAILADDRESS')  
                end  
            end  
            -- Device is triggered by Homematic, i.e. Garage door has moved.  
            if item.isDevice then  
                -- The state is set by Homematic script according alert level: 1=closed, 4=open  
                domoticz.log( ('Garagentor ist %s (%d).'):format(domoticz.devices(IDX_GARAGEDOOR_ALERT).text,  
domoticz.devices(IDX_GARAGEDOOR_ALERT).nValue) )  
                end  
            end  
    }  
}
```

# Heating Unit Gas Usage (HmIP-FCI1)

## Purpose

To read the gas meter counter and update a Homematic system variable keeping the gas meter counter value in m<sup>3</sup> with 3 digits.

Domoticz to pull the value of the system variable holding the counter value and updates various devices.

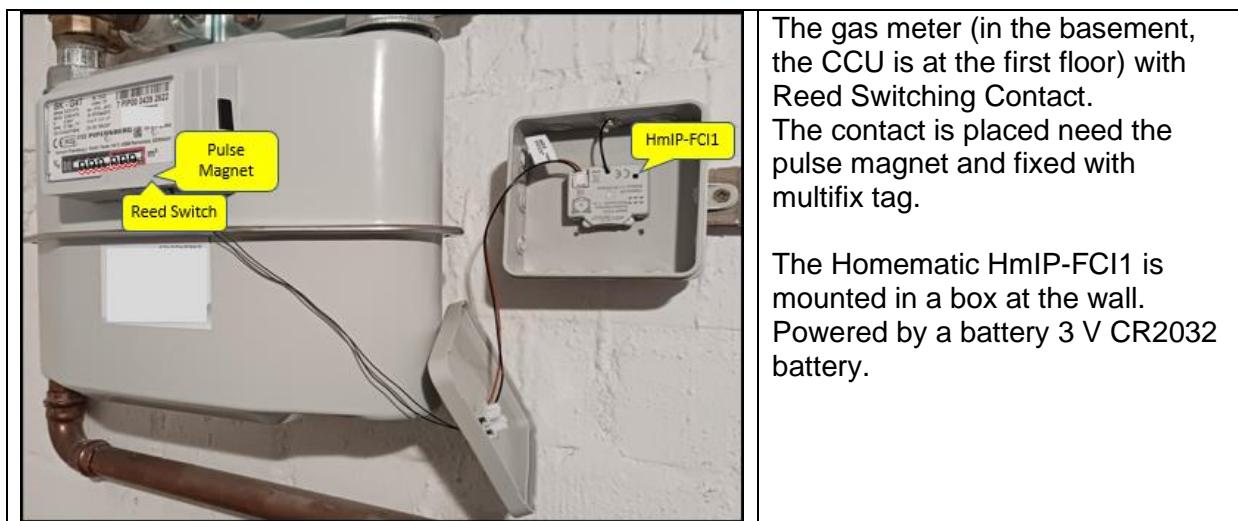
## Solution

The Homematic device HmIP-FCI1, Homematic IP Contact Interface flush-mount – 1 channel (manufactured by ELV), is used to read the pulse signal of a Reed Switch (using a model railroad device manufactured by Roco, part no: 61193 named Roco Switching Contact), connected to the Gas Meter (model DS\_BK\_4\_EN).

The gas meter sends a pulse every 10dm<sup>3</sup> (0.01m<sup>3</sup>) which triggers closing the Reed Switch. The HmIP-FCI1 state changes and updates the Homematic System Variable "HeatingUnitGasMeter" (type Number), which keeps the value of the gas meter in NNNNN.NNN m<sup>3</sup>.

Domoticz *pulls* in regular intervals via Automation Event dzVents HTTP XML-API request, the value of the Homematic system variable and updates various Homematic devices (P1 meter, custom).

The gas meter counter value is managed by Homematic thus independent from Domoticz. Any application can pull the actual gas meter counter value from Homematic.



## Domoticz Bar Chart Gas Meter (Type RFXMeter, Subtype RFXMeter Counter, Type Gas)



# Homematic Configuration

## Device

The Homematic device HmIP-FCI1, Homematic IP Contact Interface flush-mount – 1 channel, is used.

The device is obtained from the company [ELV](#) as a kit and assembled and teached-in according its manual.

Name: Heizung Gas Verbrauch Sensor

Type description: HmIP FCI1

Serial number: 001FDD89BA99D3

### The device settings

Name	Type description	Picture	Description	Serial number	Interface	Firmware
Heizung Gas Verbrauch Sensor	HmIP-FCI1		Homematic IP Contact Interface flush-mount - 1 channel	001FDD89BA99D3	HmIP-RF	Version: 1.0.14

Homematic WebUI: Home page > Settings > Devices > Set device / channel parameter

### Battery

The parameter “Low. bat. Threshold” is set to 1.5V – this could probably set lower depending required signal range. If high contact rate, the battery drains within a week.

See chapter Battery.

## Device State XML-API

The device HmIP-FCI1 attributes are obtained from the XML-API script "statelist.cgi". In the XML-tree response search for device name "Heizung Gas Verbrauch Sensor".

```
http://ccu-ip/config/xmlapi/statelist.cgi
```

```
<device name="Heizung Gas Verbrauch Sensor" ise_id="5930" unreach="false" config_pending="false">
  <channel name="Heizung Gas Verbrauch Sensor:0" ise_id="5931" index="0" visible="true" operate="true">
    <datapoint name="HmIP-RF.001FDD89BA99D3:0.CONFIG_PENDING" type="CONFIG_PENDING" ise_id="5932" value="false" valuetype="2" valueunit="" timestamp="1673963930" operations="5"/>
    <datapoint name="HmIP-RF.001FDD89BA99D3:0.DUTY_CYCLE" type="DUTY_CYCLE" ise_id="5936" value="false" valuetype="2" valueunit="" timestamp="1673963929" operations="5"/>
    <datapoint name="HmIP-RF.001FDD89BA99D3:0.LOW_BAT" type="LOW_BAT" ise_id="5938" value="false" valuetype="2" valueunit="" timestamp="1673963929" operations="5"/>
    <datapoint name="HmIP-RF.001FDD89BA99D3:0.OPERATING_VOLTAGE" type="OPERATING_VOLTAGE" ise_id="5942" value="2.600000" valuetype="4" valueunit="" timestamp="1673963929" operations="5"/>
    <datapoint name="HmIP-RF.001FDD89BA99D3:0.OPERATING_VOLTAGE_STATUS" type="OPERATING_VOLTAGE_STATUS" ise_id="5943" value="0" valuetype="16" valueunit="" timestamp="1673963929" operations="5"/>
    <datapoint name="HmIP-RF.001FDD89BA99D3:0.RSSI_DEVICE" type="RSSI_DEVICE" ise_id="5944" value="188" valuetype="8" valueunit="" timestamp="1673963930" operations="5"/>
    <datapoint name="HmIP-RF.001FDD89BA99D3:0.RSSI_PEER" type="RSSI_PEER" ise_id="5945" value="0" valuetype="8" valueunit="" timestamp="0" operations="5"/>
    <datapoint name="HmIP-RF.001FDD89BA99D3:0.UNREACH" type="UNREACH" ise_id="5946" value="false" valuetype="2" valueunit="" timestamp="1673963930" operations="5"/>
    <datapoint name="HmIP-RF.001FDD89BA99D3:0.UPDATE_PENDING" type="UPDATE_PENDING" ise_id="5950" value="false" valuetype="2" valueunit="" timestamp="1673963948" operations="5"/>
  </channel>
  <channel name="HmIP-FCI1 001FDD89BA99D3:1" ise_id="5954" index="1" visible="true" operate="true">
    <datapoint name="HmIP-RF.001FDD89BA99D3:1.PRESS_LONG" type="PRESS_LONG" ise_id="5955" value="" valuetype="2" valueunit="" timestamp="0" operations="4"/>
    <datapoint name="HmIP-RF.001FDD89BA99D3:1.PRESS_LONG_RELEASE" type="PRESS_LONG_RELEASE" ise_id="5956" value="" valuetype="2" valueunit="" timestamp="0" operations="4"/>
    <datapoint name="HmIP-RF.001FDD89BA99D3:1.PRESS_LONG_START" type="PRESS_LONG_START" ise_id="5957" value="" valuetype="2" valueunit="" timestamp="0" operations="4"/>
    <datapoint name="HmIP-RF.001FDD89BA99D3:1.PRESS_SHORT" type="PRESS_SHORT" ise_id="5958" value="" valuetype="2" valueunit="" timestamp="0" operations="4"/>
    <datapoint name="HmIP-RF.001FDD89BA99D3:1.STATE" type="STATE" ise_id="5959" value="true" valuetype="2" valueunit="" timestamp="1673963929" operations="5"/>
  </channel>
</device>
```

The datapoint "HmIP-RF.001FDD89BA99D3:1.STATE" is used to read the state (attribute value with true or false) of the Reed Switch connected to the Gas Meter.

```
<datapoint name="HmIP-RF.001FDD89BA99D3:1.STATE" type="STATE" ise_id="5959" value="true" valuetype="2" valueunit="" timestamp="1673963929" operations="5"/>
```

## Homematic System Variable

### Definition

The system variable “HeatingUnitGasMeter” has type Number with range 0 – 99999.999 m3 (max gas meter counter).

Name	Description	Variable type	Values	Unit of measurement	Channel assignment	Action	Connections
Filter		Filter		Filter	Filter		
HeatingUnitGasMeter	Heating Unit Gas Meter counter value	Number	Minimal value: 0 Maximum value: 99999.999	m3		<a href="#">Delete</a> <input checked="" type="checkbox"/> visible <a href="#">Edit</a> <input type="checkbox"/> logged	<a href="#">Programs</a>

Screenshot Homematic WebUI: Home page > Settings > System variable

Name	Description	Last modified	Status
Filter			
HeatingUnitGasMeter	Heating Unit Gas Meter counter value	20.01.2023 12:18:04	650.86 m3

Screenshot Homematic WebUI: Home page > Status and control > System variable

### IMPORTANT

If the Homematic device is installed or restarted or battery changed, then set the value of the gas meter counter via URL HTTP XML-API request.

See below.

## XML-API System Variable

Examples of XML-API addon scripts to get or set attributes of the System Variable.

### GET System Variables List

Get the list of all system variables, to obtain the ise\_id of the system variable named "HeatingUnitGasMeter".

```
http://ccu-ip/config/xmlapi/sysvarlist.cgi
```

```
<systemVariables>
..
<systemVariable name="HeatingUnitGasMeter" variable="0.000000" value="0.000000" value_list=""
ise_id="5703" min="0" max="99999.999" unit="m3" type="4" subtype="0" logged="false" visible="true"
timestamp="1674034220" value_name_0="" value_name_1=""/>
..
</systemVariables>
```

The ise\_id is 5703, which is used next to get the value of the system variable.

### GET Value

Get the value of the system variable with ise\_id=5703.

```
http://ccu-ip/addons/xmlapi/sysvar.cgi?ise_id=5703
```

```
<systemVariables>
<systemVariable name="HeatingUnitGasMeter" variable="632.710000" value="632.710000" value_list=""
value_text="" ise_id="5703" min="0" max="99999.999" unit="m3" type="4" subtype="0"
timestamp="1674034358" value_name_0="" value_name_1=""/>
</systemVariables>
```

The HTTP response, as XML tree, can be parsed for example in a Domoticz Automation Event dzVents.

The value is stored in the XML attribute "value".

```
local newgascounter =
tonumber(domoticz_applyXPath(item.data,'//systemVariable[@ise_id="'.DATAPOINT_ISE_ID..'" ]/@value'))
domoticz.log(newgascounter) -- 632.850 m3
-- Update the gasusage P1 meter (in dm3) --> 1 imp = 0.01 m3 = 10 dm3
-- i.e 632.710 * 1000 = 632710 dm3
domoticz.devices(IDX_GASUSAGE_P1METER).updateGas(newgascounter * 1000)
```

### SET Value

Set the value of the system variable with ise\_id=5703.

```
http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id=5703&new_value=632.710
```

```
<result>
<changed id="5703" new_value="632.730"/>
</result>
```

## Sync System Variable Value

### IMPORTANT

In case:

- the Homematic device HmIP-FCI1 is installed or
- the Homematic device HmIP-FCI1 battery replaced or
- the Homematic CCU is restarted or
- the Gas Meter is updated or replaced.

the value of the system variable “HeatingUnitGasMeter” (ise\_id=5703) must be synced with the value of the gas meter counter.

This is done by executing the XML-API script “statechange.cgi”.

Example:

```
http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id=5703&new_value=632.710
```

```
<result>
  <changed id="5703" new_value="632.730"/>
</result>
```

## Steps to Sync the Gas Meter Counter Value

1 - Open Browser

2 - Set URL without setting the value for the parameter “new\_value”:

```
http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id=5703&new_value=
```

3 - Read the gas meter counter, i.e., 632.710.

4 - Add the counter value in the URL parameter “new\_value”:

```
http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id=5703&new_value=632.710
```

5 - Submit and check the HTTP response.

```
<result>
  <changed id="5703" new_value="632.730"/>
</result>
```

## Homematic Program

### Logic

The Homematic program “Heating Unit Gas Usage PULL” logic:

Condition: If device HmIP-FCI1 001FDD89BA99D3:1 when closed trigger when changed  
Activity: Then execute script “Heating Unit Gas Usage PULL” immediately.

### Script

```

! Function: Heating Unit Gas Usage PULL
! heating_unit_gas_usage_pull.script
! Listen to HmIP-FCI1 State changes.
! If the state is closed, then increase the value of the system variable HeatingUnitGasMeter by 0.01
! 20230118 rwbl

string stateReed = dom.GetObject("HmIP-RF.001FDD89BA99D3:1.STATE").State(); ! true or false
WriteLine(stateReed);

! System Variable
string gasMeter = "HeatingUnitGasMeter";

integer counter;

! Get the actual counter value from the sysvar
counter = dom.GetObject(gasMeter).Value();
WriteLine(counter);

! Increase the counter by 10dm3
counter = counter + 0.01;

! Update the sysvar
dom.GetObject(gasMeter).State(counter);
WriteLine(dom.GetObject(gasMeter).Value());

```

## Domoticz Configuration

This solution makes use of the function Gas Usage (Manual) as described in the [Domoticz Homeautomation Workbook](#), chapter Gas Usage (Manual).

### Purpose

Domoticz to pull the value of the Homematic System Variable “HeatingUnitGasMeter”, which holds the Gas Meter counter value and update the Domoticz device “Gaszähler”.

### Devices

For the pull solution, there are no additional devices required because the next Automation Event updates the devices used for the Gas Usage (Manual) solution.

The device “Gaszähler” stores the Gas Meter counter value.  
The type is RFXMeter, subtype RFXMeter counter and type Gas.

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
395	VirtualSensors	141DB	1	Gaszähler	RFXMeter	RFXMeter counter	672.010 m3

Domoticz GUI > Setup > Devices

The screenshot shows the Domoticz GUI interface. On the left, a list of devices is displayed with one item selected: "Gaszähler" (Idx: 395, Type: RFXMeter, SubType: RFXMeter counter, Value: 672.010 m3). On the right, a detailed "Edit Device" dialog is open for the same device. The dialog fields include:

- Idx: 395
- Name: Gaszähler
- Description: (empty)
- Sensor Icon: Default
- Type: Gas
- Counter Divider: 1000 (0 = Application default)
- Meter Offset: 0

At the bottom of the dialog are buttons for Update, Remove Device, Replace, and Cancel.

Domoticz GUI > Tab Utility

The device log, bar charts, example:



## Automation Event dzVents

The event pulls in regular intervals, via HTTP XML-API requests the value of the Homematic system variable “HeatingUnitGasMeter” and updates the Domoticz device “Gaszähler” (IDX=395).

```
--[[[heating_unit_gas_usage_pull.dzvents]]]

-- Domoticz Custom device
local IDX_GASUSAGE_COUNTER = 395

-- Homematic
local DATAPOINT_ISE_ID = 5703;

-- url http xml-api request get value datapoint with ise_id. ise_id is set in the function openURL.
local URL_XMLAPI = 'http://ccu-ip/addons/xmlapi/sysvar.cgi?ise_id=';
-- define the unique (across all dzVents) callback
local RES_XMLAPI = "RESXMLAPIHEATINGUNITGASUSAGE";

-- Timer
-- local TIMERRULE = 'every 5 minutes'
local TIMERRULE = 'every minute'

-- Helpers
function round(number, decimals)
    local power = 10^decimals
    return math.floor(number * power) / power
end

return {
    on = { timer = { TIMERRULE }, httpResponses = { RES_XMLAPI } },
    execute = function(domoticz, item)
        if (item.isTimer) then
            domoticz.openURL({url = URL_XMLAPI .. DATAPOINT_ISE_ID, method = 'GET', callback = RES_XMLAPI})
        end

        if (item.isHTTPResponse) then
            if (item.statusCode == 200) then
                if (item.callback == RES_XMLAPI) then
                    local newgascounter =
tonumber(domoticz_applyXPath(item.data,'//systemVariable[@ise_id="'. .. DATAPOINT_ISE_ID .. '"]/@value'))
                    domoticz.log(newgascounter) -- 632.850 m3
                    -- Update the gasusage P1 meter (in dm3) --> 1 imp = 0.01 m3 = 10 dm3
                    -- i.e 632.850 * 1000 = 632850 dm3
                    domoticz.devices(IDX_GASUSAGE_COUNTER).updateCounter(newgascounter * 1000)
                end
            else
                domoticz.log('[ERROR] Request:' .. item.statusText, domoticz.LOG_ERROR)
            end
        end
    end
}
```

When the Domoticz devices are updated, the Domoticz Automation Event dzVents “heating\_gas\_usage” is triggered.

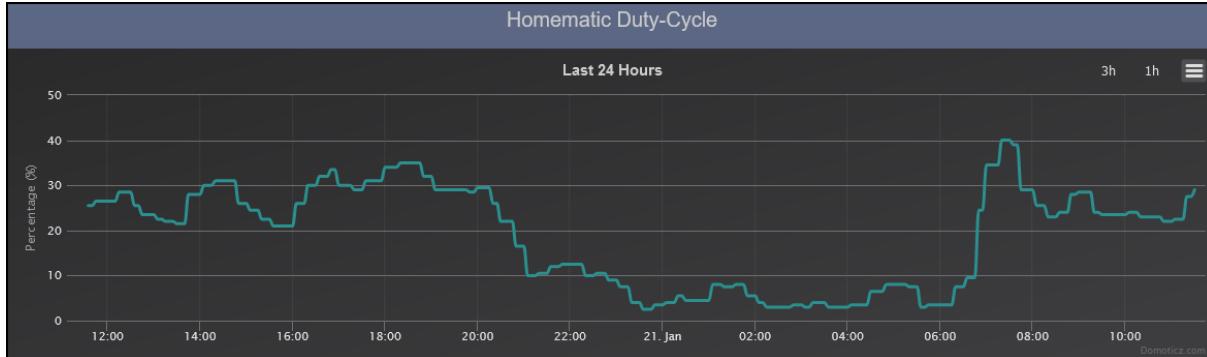
The “heating\_gas\_usage” is a separate script which has been used to manually update the Gas Meter counter device (IDX=395).

The manual solution was in place prior the automatic solution but can still be used in case the automatic solution is not working.

## Duty-Cycle

### Issue

This solution results in an increase of the Homematic Duty-Cycle depending on the gas flow.



The chart shows during the day (winter, temperature -5 to 2 C) a Duty-Cycle around 30% and overnight 5-8%. The overnight value will probably apply for summer month as well (to be verified).

### Considerations

Because of the high Duty-Cycle, explore possible alternate solutions.

#### **Homematic ES-GAS-2 & HM-ES-TX-WM**

The device „ELV Energy-Sensor Gas 2 für Gaszähler“ (ES-GAS-2) connected to the ELV Homematic Bausatz Zählersensor-Sendeeinheit Strom/Gas HM-ES-TX-WM.

#### **Raspberry Pi Pico**

Use a Raspberry Pi Pico with the Reed Switch and the Homematic HmIP-FCI1 connected. An internal counter is updated on every gas meter magnet pulse and closes the HmIP-FCI1 contact every 10 or 100 gas meter counter pulses. Then the Homematic system variable "HeatingUnitGasMeter" is updated. Domoticz to pull the value via HTTP XML-API request.

#### **Raspberry Pi Pico W**

Use a Raspberry Pi Pico W with the Reed Switch connected, to update a Domoticz User Variable via Domoticz HTTP JSON/API request.

A Domoticz Automation Event dzVents is then triggered because the User Variable is changed. The event updates the related Domoticz Device(s).

#### **ESPEasy with NodeMCU**

Use an ESP8266 NodeMCU with ESPEasy. In ESPEasy, create a device Pulse Counter and send the pulse value to Domoticz HTTP controller.

## Battery

### Issue

The battery lifetime depends on the contact rate.

With a high contact rate (2 times/minute), the lifetime decreases rapidly = observed a decrease of 0.1V over 3 days.

HTTP XML-API Request to check the Battery attributes.

```
http://ccu-ip/addons/xmlapi/state.cgi?device_id=5930
```

From the HTTP response XML tree, regularly check the attributes LOW\_BAT and OPERATING\_VOLTAGE.

```
<state>
<device name="Heizung Gas Verbrauch Sensor" ise_id="5930" unreach="false" config_pending="false">
<channel name="Heizung Gas Verbrauch Sensor:0" ise_id="5931">
<datapoint name="HmIP-RF.001FDD89BA99D3:0.CONFIG_PENDING" type="CONFIG_PENDING" ise_id="5932"
value="false" valuetype="2" valueunit="" timestamp="1674379161"/>
<datapoint name="HmIP-RF.001FDD89BA99D3:0.DUTY_CYCLE" type="DUTY_CYCLE" ise_id="5936" value="false"
valuetype="2" valueunit="" timestamp="1674379161"/>
<datapoint name="HmIP-RF.001FDD89BA99D3:0.LOW_BAT" type="LOW_BAT" ise_id="5938" value="false"
valuetype="2" valueunit="" timestamp="1674379161"/>
<datapoint name="HmIP-RF.001FDD89BA99D3:0.OPERATING_VOLTAGE" type="OPERATING_VOLTAGE" ise_id="5942"
value="2.300000" valuetype="4" valueunit="" timestamp="1674379161"/>
<datapoint name="HmIP-RF.001FDD89BA99D3:0.OPERATING_VOLTAGE_STATUS" type="OPERATING_VOLTAGE_STATUS"
ise_id="5943" value="0" valuetype="16" valueunit="" timestamp="1674379161"/>
<datapoint name="HmIP-RF.001FDD89BA99D3:0.RSSI_DEVICE" type="RSSI_DEVICE" ise_id="5944" value="185"
valuetype="8" valueunit="" timestamp="1674379161"/>
<datapoint name="HmIP-RF.001FDD89BA99D3:0.RSSI_PEER" type="RSSI_PEER" ise_id="5945" value="0"
valuetype="8" valueunit="" timestamp="0"/>
<datapoint name="HmIP-RF.001FDD89BA99D3:0.UNREACH" type="UNREACH" ise_id="5946" value="false"
valuetype="2" valueunit="" timestamp="1674379161"/>
<datapoint name="HmIP-RF.001FDD89BA99D3:0.UPDATE_PENDING" type="UPDATE_PENDING" ise_id="5950"
value="false" valuetype="2" valueunit="" timestamp="1673963948"/>
</channel>
<channel name="HmIP-FCI1 001FDD89BA99D3:1" ise_id="5954">
<datapoint name="HmIP-RF.001FDD89BA99D3:1.PRESS_LONG" type="PRESS_LONG" ise_id="5955" value=""
valuetype="2" valueunit="" timestamp="0"/>
<datapoint name="HmIP-RF.001FDD89BA99D3:1.PRESS_LONG_RELEASE" type="PRESS_LONG_RELEASE" ise_id="5956"
value="" valuetype="2" valueunit="" timestamp="0"/>
<datapoint name="HmIP-RF.001FDD89BA99D3:1.PRESS_LONG_START" type="PRESS_LONG_START" ise_id="5957"
value="" valuetype="2" valueunit="" timestamp="0"/>
<datapoint name="HmIP-RF.001FDD89BA99D3:1.PRESS_SHORT" type="PRESS_SHORT" ise_id="5958" value="false"
valuetype="2" valueunit="" timestamp="1674379161"/>
<datapoint name="HmIP-RF.001FDD89BA99D3:1.STATE" type="STATE" ise_id="5959" value="true" valuetype="2"
valueunit="" timestamp="1674379161"/>
</channel>
</device>
</state>
```

### Considerations

See previous Duty-Cycle considerations.

# Heating Unit Temperature (HmIP-STE2-PCB)

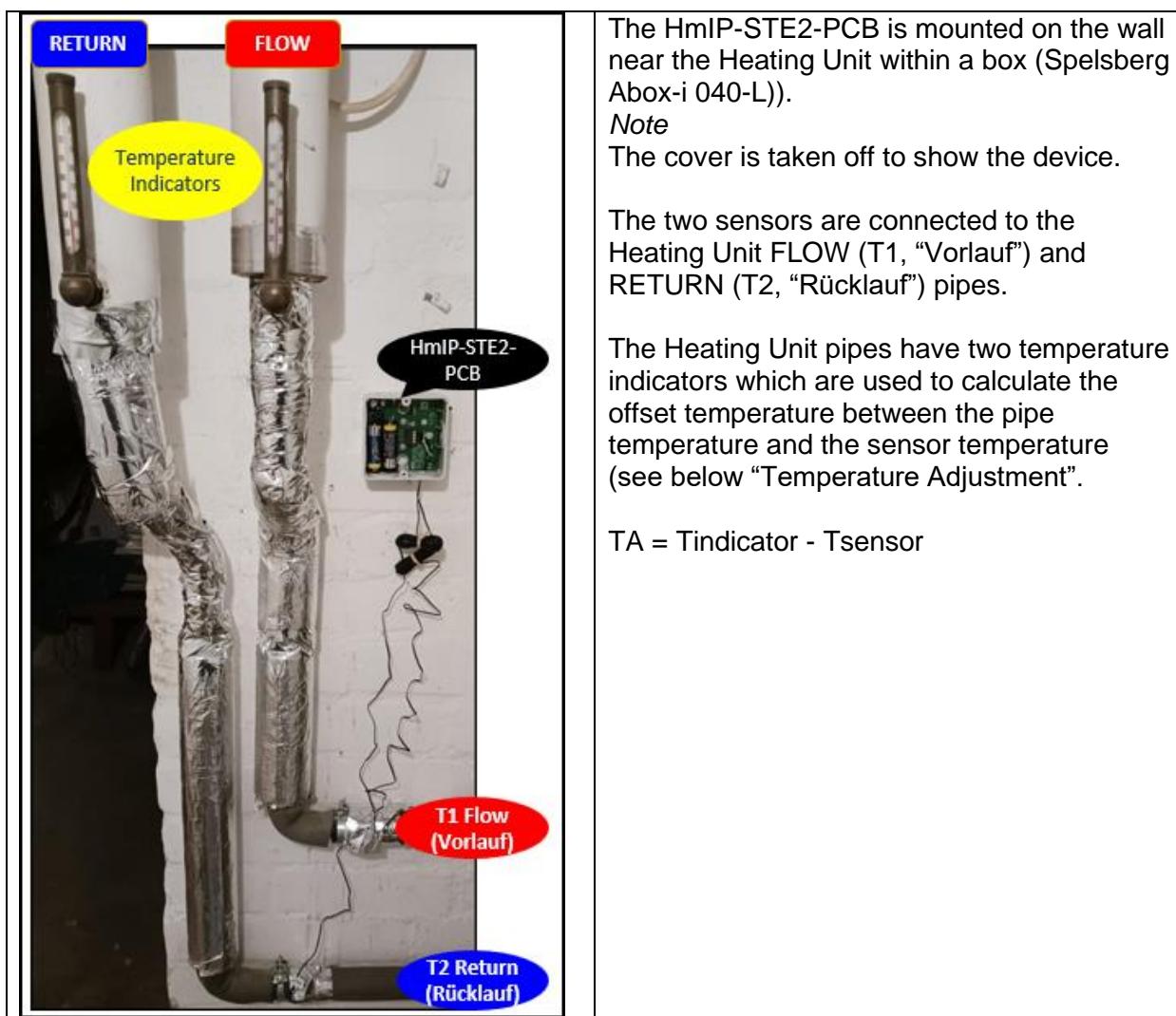
## Purpose

To monitor the heating unit flow and return temperature.

## Solution

The solution makes use of the Homematic IP 2-channel-Temperature Sensor HmIP-STE2-PCB.

The HmIP-STE2-PCB has two temperature sensors which are connected to the Heating Unit pipes flow and return.



There are two solutions worked out to display, in Domoticz, the Heating Unit Flow T1 and Return T2 temperature:

- **PULL** = Domoticz pulls in regular intervals the Homematic HmIP-STE2-PCB device temperature data via HTTP XML-API requests triggered by a dzVents Automation Event.
- **PUSH** = Homematic HmIP-STE2-PCB device temperature data updates are pushed using a Homematic Program with Script to Domoticz via HTTP JSON/API request triggering a dzVents Custom Event.

**The PULL solution is in place.**

The PUSH solution has been explored.

# Homematic Configuration

## Device

The device HmIP-WRCD is teached in and renamed to “Heizung Temperatur”.

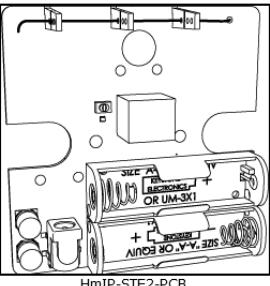
### Device Inbox after teach-in.

Device inbox											
Type	Description	Serial number	Interface category	Transmission mode	Name	Function	Room	Functional test	Action	Done	
HmIP-ST E2-PCB	Homematic IP Temperature Sensor with external probes - 2 channels	00281F2996368C	HmIP-RF	Secured	HmIP-STE2-PCB 00281F2996368C			<span style="background-color: green; color: white;">OK</span>	<input type="checkbox"/> operable <input checked="" type="checkbox"/> visible <input type="checkbox"/> logged	<input type="button" value="Done"/>	

After pressing done, go to Settings > Devices and select the new device. Change the name accordingly. Start the test which will respond if the reset button on the device is pressed.

## Device Settings

General device settings: 00281F2996368C



Name:   
 Type description:   
 Serial number:   
 Operable:   
 Logged:

**Functional test**

--:--:--

During the functional test the error-free communication to the device is checked.  
Therefore, switching commands will be send to all actuators connected to the device. Sensors (e.g. remote controls) are usually sending signals only if they are operated manually. The test is passed as soon as the first feedback will be received by the device.

Homematic WebUI > Settings > Devices > Device selected.

## Device Status and Control

Status and control					
Devices					
Name	Room	Function	Last modified	Control	
Filter	Filter	Filter			
HmIP-STE2-PCB 00281F2996368C:1 Connection partner Temperature			04.01.2023 11:37:09	Temperature sensor 1 47.9 °C	
HmIP-STE2-PCB 00281F2996368C:2 Connection partner Temperature			04.01.2023 11:37:09	Temperature sensor 2 43.2 °C	
HmIP-STE2-PCB 00281F2996368C:3 Connection partner Differential temperature			04.01.2023 11:37:09	Difference temperature sensor 1 - sensor 2 4.7 °C	

The device with actual data.

# Domoticz Configuration

## Purpose

To display the Heating Unit Flow and Return Temperatures provided by Homematic device 2-channel Temperature Sensor HmIP-STE2-PCB.

## Devices

Two Domoticz virtual sensors are used from type Temp, SubType LaCrosse TX3, which are named "Heizung Vorlauf" and "Heizung Rücklauf"

### Domoticz Hardware & Devices

The Domoticz devices assigned to Homematic IP devices are Virtual Sensors created with the Hardware "Dummy".

Idx	Name	Enabled	Type
6	VirtualSensors	Yes	Dummy (Does nothing, use for virtual switches only) <a href="#">Create Virtual Sensors</a>

Click on [Create Virtual Sensors] to create two new devices Named **Heizung Vorlauf** and **Heizung Rücklauf**, from Sensor Type "Temperature" to show Homematic HmIP-STE2-PCB sensors T1 (flow temperature) and T2 (return temperature).

The temperature data for the two devices are updated via the Domoticz Custom Event **homematic\_heatingunittemperature**.

### Domoticz Devices List

	Idx	Hardware	ID	Unit	Name	Type	SubType	Data
	375	VirtualSensors	141C7	1	Heizung Vorlauf	Temp	LaCrosse TX3	55.0 C
	397	VirtualSensors	141DD	1	Heizung Rücklauf	Temp	LaCrosse TX3	50.2 C

### Domoticz Device Widgets

**Heizung Vorlauf** ↑ 56.9° C



Last Seen: 2023-01-04 11:29:14

★ Log Edit Notifications

**Heizung Rücklauf** ↑ 49° C



Last Seen: 2023-01-04 11:29:14

★ Log Edit Notifications

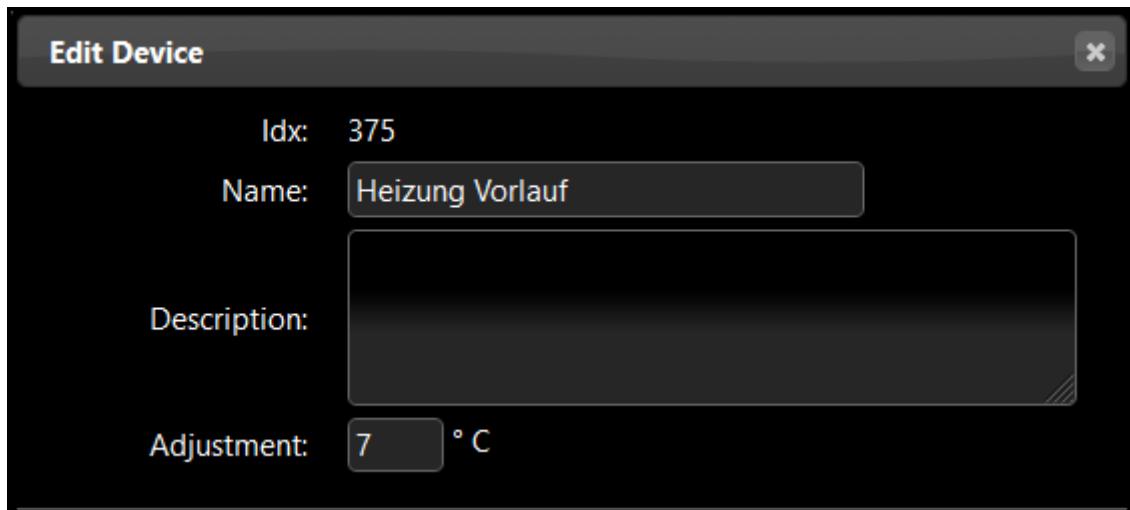
## Temperature Adjustment

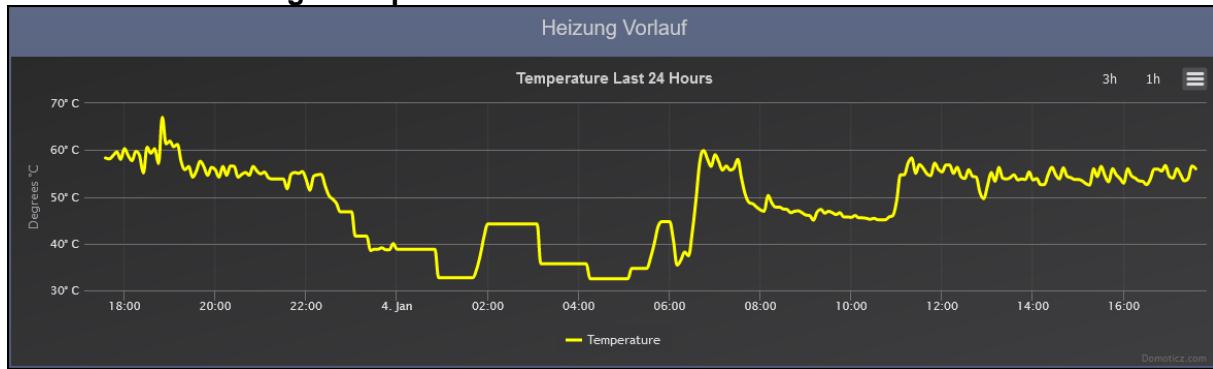
### IMPORTANT

The temperature of the two devices must be adjusted, because the Heating Unit piping temperatures are higher than the HmIP-STE2-PCB sensors.

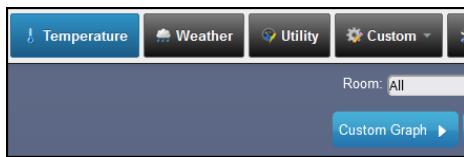
The HmIP-STE2-PCB sensors are connected at the outside of the pipes.

The adjustment temperature value is calculated between the piping temperature indicator and the HmIP-STE2-PCB measured sensor temperature.



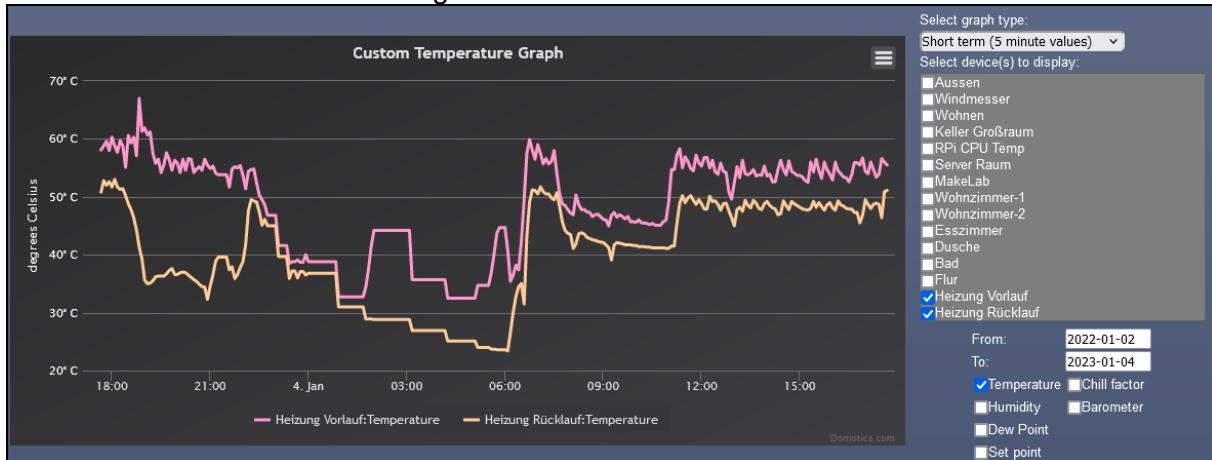
**Domoticz Device Log Example**

## Example Custom Temperature Graph



(GUI > Tab Temperature > Custom Graph)

The devices selected are Heizung Vorlauf and Rücklauf



(URL: <http://domoticz-ip:8080/#/Temperature/CustomTempLog>)

## Solution PULL

Domoticz PULLS in regular intervals, Homematic device data via HTTP XML-API requests triggered by a dzVents Automation Event.

This solution is rather simple i.e., just a Domoticz Automation Event dzVents to request data from Homematic followed by updating the Domoticz temperature devices.

This solution has been explored only on the Domoticz Development System.

### Homematic Configuration

There are no programs or scripts required.

The XML-API addon must be installed as used to get the device data via script state.cgi.

The XML-API statelist is required to get the datapoint ise\_id from the T1 and T2 sensors.

Run in a browser:

```
http://ccu-ip/config/xmlapi/statelist.cgi
```

Select the device named "Heizung Temperatur".

Get, for the channels 1 and 2, the ise\_id of the datapoint ACTUAL\_TEMPERATURE.

```
<datapoint name="HmIP-RF.00281F2996368C:1.ACTUAL_TEMPERATURE" type="ACTUAL_TEMPERATURE" ise_id="5729" value="50.300000" valuetype="4" valueunit="°C" timestamp="1672851621" operations="5"/>
<datapoint name="HmIP-RF.00281F2996368C:2.ACTUAL_TEMPERATURE" type="ACTUAL_TEMPERATURE" ise_id="5732" value="39.100000" valuetype="4" valueunit="°C" timestamp="1672851621" operations="5"/>
```

The ise\_id is used to request via HTTP XML-API request the actual temperature.

### HTTP Request

```
http://ccu-ip/config/xmlapi/state.cgi?datapoint_id=5729,5732
```

### HTTP Response

```
<state>
  <datapoint ise_id="5729" value="44.100000"/>
  <datapoint ise_id="5732" value="40.400000"/>
</state>
```

For this PULL solution, the Domoticz Automation Event dzVents triggers via timer, the datapoints. These are parsed to get the attribute value, which is used to update the Domoticz devices.

## Domoticz Configuration

### Automation Event

```
--[[{"homematic_heatingunittemperature.dzvents": "Update the Heating Unit Flow and Return Temperature measured by Homematic HmIP-STE2-PCB device.\nThe sensor data is pulled Homematic pushes to update the device data.\nThe event is triggered by the homematic program ("Heating Unit Temperature") sending an HTTP JSON/API request for a Custom Event.\nThe T1 temperature (Flow or Vorlauf)of the HmIP-STE2-PCB device is used.\n\n20230105 rwbl"}]]--\n\n-- Define the datapoint ise_id for the attribute ACTUAL_TEMPERATURE of the device channels 1 & 2.\n-- Use the XML-API script statelist (http://domoticz-ip/config/xmlapi/statelist.cgi) to get the device datapoints\n-- <datapoint name="HmIP-RF.00281F2996368C:1.ACTUAL_TEMPERATURE" type="ACTUAL_TEMPERATURE"\nise_id="5729" value="20.20000" valuetype="4" valueunit="°C" timestamp="1672744699" operations="5"/>\nlocal DATAPOINT_ISE_ID_T1 = 5729;\nlocal DATAPOINT_ISE_ID_T2 = 5732;\n-- Define the URL to get the sensor data from the CCU using the XML-API\nlocal URL_XMLAPI = ''..DATAPOINT\_ISE\_ID\_T2;\n\n-- define the unique (across all dzVents) callback\nlocal RES_XMLAPI = "RESXMLAPIHEATINGUNITTEMPERATURE";\n\n-- Define the Domoticz temperature devices (virtual sensors)\nlocal IDX_HEATING_UNIT_T1 = 375;\nlocal IDX_HEATING_UNIT_T2 = 397;\n\n-- Timer - use every minute for tests\nlocal TIMERRULE = 'every 5 minutes'\n-- local TIMERRULE = 'every minute'\n\nreturn {\non = { timer = { TIMERRULE }, httpResponses = { RES_XMLAPI } },\nexecute = function(domoticz, item)\n    -- check if the item is a timer, then request information\n    if (item.isTimer) then\n        domoticz.openURL({url = URL_XMLAPI, method = 'GET', callback = RES_XMLAPI})\n    end\n\n    -- check if the item is a httpresponse from the openurl callback\n    if (item.isHTTPResponse) then\n        if (item.statusCode == 200) then\n            -- Select callback - in this case there is only one\n            if (item.callback == RES_XMLAPI) then\n                -- Get the key value from the http response:\n                -- <state><datapoint ise_id="5729" value="44.10000"/><datapoint ise_id="5732"\nvalue="40.40000"/></state>\n                -- Update the domoticz devices\n                local valuet1 =\ntonumber(domoticz_applyXPath(item.data,'//datapoint[@ise_id="'.DATAPOINT_ISE_ID_T1..'" ]/@value'))\n                domoticz.devices(IDX_HEATING_UNIT_T1).updateTemperature(valuet1);\n                local valuet2 =\ntonumber(domoticz_applyXPath(item.data,'//datapoint[@ise_id="'.DATAPOINT_ISE_ID_T2..'" ]/@value'))\n                domoticz.devices(IDX_HEATING_UNIT_T2).updateTemperature(valuet2);\n                domoticz.log((('T1=%1f, T2=%1f'):format(valuet1, valuet2))\n            end\n        else\n            domoticz.log('[ERROR] Request:' .. item.statusText, domoticz.LOG_ERROR)\n        end\n    end\nend\n}
```

### Domoticz Log

```
2023-01-05 09:50:00.722 Status: dzVents: Info: ----- Start internal script:  
homematic_heatingunittemperature:, trigger: "every 5 minutes"  
2023-01-05 09:50:00.722 Status: dzVents: Info: ----- Finished homematic_heatingunittemperature  
  
2023-01-05 09:50:00.904 Status: dzVents: Info: Handling httpResponse-events for:  
"RESXMLAPIHEATINGUNITTEMPERATURE"  
2023-01-05 09:50:00.904 Status: dzVents: Info: ----- Start internal script:  
homematic_heatingunittemperature: HTTPResponse: "RESXMLAPIHEATINGUNITTEMPERATURE"  
2023-01-05 09:50:00.926 Status: dzVents: Info: T1=39.6, T2=35.2  
2023-01-05 09:50:00.926 Status: dzVents: Info: ----- Finished homematic_heatingunittemperature
```

## Solution PUSH

### PUSH Data Homematic to Domoticz

Homematic pushes the flow (T1) and return (T2) temperature values to Domoticz Virtual Sensors via HTTP JSON/API request to a Domoticz Custom Event.

The custom event data is a JSON object containing for the flow & return temperature the IDX and value.

The custom event data is pushed to Domoticz when the flow temperature is update within a value range of 4 °C.

There are two Domoticz Virtual Sensors from Type Temp and SubType LaCrosse TX3 named "Heizung Vorlauf" and "Heizung Rücklauf".

Example Custom Data:

vl is Flow (Heizung Vorlauf), rl is Return (Heizung Rücklauf), idx is the Domoticz Idx of the device.

```
{["idxrl1"]=397, ["temprl1"]=37.1, ["tempvl1"]=42.5, ["idxvl1"]=375}
```

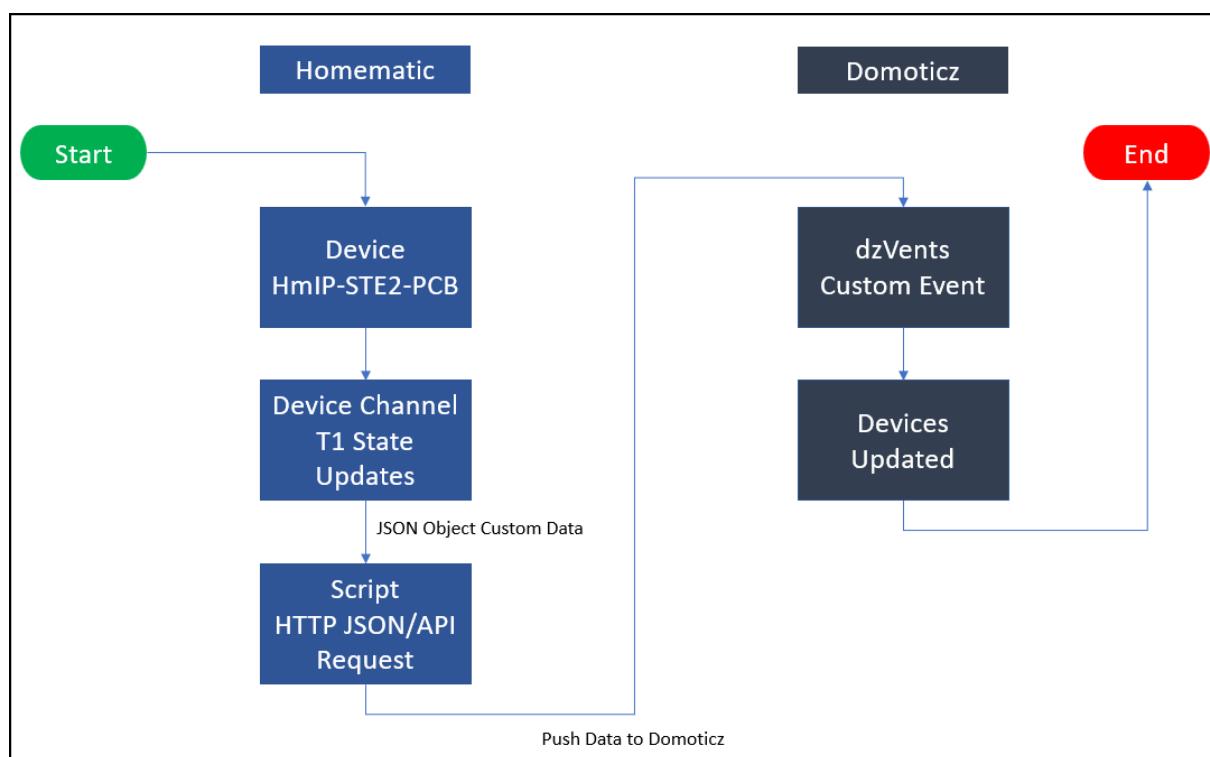
The Domoticz Event "homematic\_heatingunittemperature" is triggered by the Custom Event "homematic\_heatingunittemperature" which updates the Domoticz devices data.

### Summary PUSH Data Flow

Homematic listens to temperature updates of the T1 sensor of the HmIP-STE2-PCB device.

If the temperature is updated in the range of more than 4 °C, Homematic triggers the HTTP JSON/API request to a Domoticz Custom Event.

The Domoticz Custom Event picks up the JSON data and updates the two virtual devices "Heizung Vorlauf" and "Heizung Rücklauf".



## Homematic Configuration

The program Heating Unit Temperature listens to temperature updates of the T1 sensor of the HmIP-STE2-PCB device.

If the temperature is updated in the range of more than 4 °C, Homematic triggers the HTTP JSON/API request to a Domoticz Custom Event.

The Domoticz Custom Event picks up the JSON data and updates the two virtual devices "Heizung Vorlauf" and "Heizung Rücklauf".

**Example JSON Data submitted to Domoticz:**

```
{["idxvl"] = 375, ["tempvl"] = 49.1, ["temprl"] = 44.1, ["idxrl"] = 397}
```

## Logic

The screenshot shows the HomeMatic WebUI interface under the 'Programs and connections' > 'Programs' section. A logic configuration is displayed with the following details:

- Condition:** If... (selected)
  - Device selection: HmIP-STE2-PCB 00281F2996368C:1
  - when: actual temperature
  - within value range: more than 4.00°C
  - trigger when updated
- Activity: Then:**
  - Script: ... immediately run
  - checkbox: Stop all current delays before performing the activity (e.g. retriggering) is checked.
  - Function: Heating Unit Temperature ! homematic\_heatingunittemperature
- Activity: Else:**
  - checkbox: Stop all current delays before performing the activity (e.g. retriggering) is checked.

## Script

(snippet)

```
string cAmp = "&";
! Domoticz system url base for the http api request
string urlBase = "'http://domoticz-ip:8080/json.htm";
string customEvent = "homematic_heatingunittemperature";

! HomematicIP Device
! Get object from the Homematic WebUI > Devices OR via XMLAPI request: http://ccu-
ip/config/xmlapi/statelist.cgi

! T1 = Flow (Vorlauf) = VL, T2 = Return (Rücklauf) = RL
string tempVL = dom.GetObject("HmIP-RF.00281F2996368C:1.ACTUAL_TEMPERATURE").State(); ! 42.500000
string tempRL = dom.GetObject("HmIP-RF.00281F2996368C:2.ACTUAL_TEMPERATURE").State(); ! 37.100000

! Domoticz Devices = Domoticz virtual sensor type Temp, LaCrosse TX3
string idxVL = 375;
string idxRL = 397;

! Custom event data parameter must be in json format
string data = '{"idxvl":#idxVL#, "idxrl":#idxRL#, "tempvl":#tempVL#, "temprl":#tempRL#}'';

! Build the Domoticz http rest request url to update the device usig customevent
string urlRequest =
urlBase#"?type=command#&param=customevent#&event=#customEvent#&data=#data#";
WriteLine(urlRequest);

! Run the command without a return result
cmdRes = dom.GetObject("CUxD.CUX2801001:1.CMD_EXEC").State("wget -q -O - "#urlRequest);
```

## Domoticz Configuration

### Automation Event

The Automation Event dzVents is triggered by the Homematic script.

```
--[[  
    homematic_heatingunittemperature.dzvents  
    Update the Heating Unit Flow and Return Temperature measured by Homematic HmIP-STE2-PCB device.  
    The event is triggered by the homematic program ("Heating Unit Temperature").  
    The T1 temperature (Flow or Vorlauf)of the HmIP-STE2-PCB device is used.  
    20230103 rwbl  
]]--  
  
local CUSTOMEVENT_NAME = "homematic_heatingunittemperature"  
  
return {  
    on = { customEvents = { CUSTOMEVENT_NAME } },  
    data = {},  
    logging = {},  
    execute = function(domoticz, item)  
        if (item.isCustomEvent) then  
            -- {[{"idxrl":397, "temprl":37.1, "tempv1":42.5, "idxv1":375}  
            domoticz.log(item.json);  
            domoticz.devices(item.json.idxv1).updateTemperature(item.json.tempv1)  
            domoticz.devices(item.json.idxrl).updateTemperature(item.json.temprl)  
        end  
    end  
}
```

### Domoticz Log Entry

```
2023-01-03 16:09:41.483 Status: dzVents: Info: Handling Domoticz custom event for:  
"homematic_heatingunittemperature"  
2023-01-03 16:09:41.483 Status: dzVents: Info: ----- Start internal script:  
homematic_heatingunittemperature: Custom event: "homematic_heatingunittemperature"  
2023-01-03 16:09:41.484 Status: dzVents: Info: {[{"idxrl":397, "temprl":37.1, "tempv1":42.5,  
"idxv1":375}  
2023-01-03 16:09:41.484 Status: dzVents: Info: ----- Finished homematic_heatingunittemperature
```

## Enhancements

These are just some thoughts on enhancing the solution.

### Alert Flow Temperature

Set a low threshold for the T1 “Vorlauf” temperature to check if the Heating Unit is working.

Sent a notification in case T1 below threshold.

This can be done in the Automation Event.

# Pluggable Switch and Meter (HmIP-PSM)

## Purpose

- To remote control connected devices anytime and anywhere.
- To read in regular intervals the power (W) & energy consumption (Wh) consumption used by the connected devices.

## Solution

The Homematic IP pluggable switch and meter (HmIP-PSM) enables to remote switch devices and measure power consumption.

## Homematic Configuration

The HmIP-PSM is “teached-in” like for any other Homematic device via the Homematic WebUI.

### Screenshot of a device after adding and renamed to “MakeLab PSM”.

Name	Room	Function	Last modified	Control	
Filter	Filter	Filter		Off	On
HmIP-PSM 0001D3C99C6AB3:3 Switch actuator			12.03.2021 14:26:39	<input type="button" value="Off"/>	<input type="button" value="On"/>
HmIP-PSM 0001D3C99C6AB3:6 Status report measured value channel			12.03.2021 14:19:01	<input type="button" value="Energy counter CCU&lt;br/&gt;1.178 kWh"/> <input type="button" value="Energy counter device&lt;br/&gt;16.50 Wh"/>	<input type="button" value="Reset"/>
HmIP-PSM 0001D3C99C6AB3:8				<input type="button" value="Voltage&lt;br/&gt;229.10 V"/> <input type="button" value="Power&lt;br/&gt;0.15 W"/> <input type="button" value="Frequency&lt;br/&gt;49.98 Hz"/>	<input type="button" value="3&lt;br/&gt;Auto mode"/> <input type="button" value="Manu mode"/>

The example shows the HmIP-PSM named “MakeLab PSM”.

The device has 3 channels, from which the two channels 3 & 6, are used to set or get state/values via CCU-Jack:

- HMIP-PSM 0001D3C99C6AB3:3 - Switch actuator – switch ON or OFF  
Datapoint: STATE
- HMIP-PSM 0001D3C99C6AB3:6 - Status report measured value channel – get power, current & voltage  
Datapoints: POWER, CURRENT, VOLTAGE, ENERGY\_COUNTER

## Domoticz Configuration

For the HmIP-PSM several Domoticz devices can be created via the Hardware "Dummy". Each of the devices holds data from a PSM datapoint.

The data can be set or get in various ways:

- *GET* via a Domoticz automation event dzVents script = PULL using HTTP REST-API GET request,
- *SET* via a Domoticz automation event dzVents script = PUSH using HTTP REST-API PUT request,
- *SET* via Homematic program triggering Domoticz JSON/API request = PUSH the value of a Domoticz device.
- Other like Python script or Node-RED

### Hardware

Idx	Name	Enabled	Type	Address
4	VirtualSensors	Yes	Dummy (Does nothing, use for virtual switches only)	<a href="#">Create Virtual Sensors</a>

### Devices

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
87	VirtualSensors	000140A7	1	MakeLab Switch	Light/Switch	Switch	Off
86	VirtualSensors	00082086	1	MakeLab Current	General	Current	0 A
85	VirtualSensors	00082085	1	MakeLab Voltage	General	Voltage	0 V
84	VirtualSensors	00082084	1	MakeLab Energy	General	kWh	0.000 kWh

More devices can be created depending on the PSM datapoint required, like the device channel 6 datapoint ENERGY\_COUNTER (could be assigned to a Domoticz Custom Sensor Energy Counter Device Wh – see Homematic WebUI device information) or use less devices if for example only the switch is used

The devices created as an example for the room MakeLab added to a room plan named MakeLab Energy.

2021-03-12 15:20:10 ★▲06:42 ▼18:19 Room: MakeLab Energy

Light/Switch Devices:

- MakeLab Switch **On**  
Last Seen: 2021-03-12 15:05:08

Utility Sensors:

- MakeLab Current **0 A**  
Last Seen: 2021-03-12 13:53:25
- MakeLab Energy **0 Watt**  
Today: 0.000 kWh  
Last Seen: 2021-03-12 13:52:15
- MakeLab Voltage **0 V**  
Last Seen: 2021-03-12 13:52:45

#### Note

As the devices have been created initially and not updated, there is no data yet available.

## PSM Switch

Checkout the datapoints for the device channel 3 - Switch actuator:

### *HTTP XML-API Request*

```
http://ccu-ip:port/device/0001D3C99C6AB3/3
```

### *HTTP XML-API Response*

From the HTTP response, the datapoint (listed as parameter) required is: **STATE**.  
See snippet:

```
{
  "address": "0001D3C99C6AB3:3", "parentType": "HMIP-PSM", ...
  "title": "HMIP-PSM 0001D3C99C6AB3:3", "type": "SWITCH_VIRTUAL_RECEIVER",
  "~links": [...,
    {"rel": "parameter", "href": "STATE", "title": "HMIP-PSM 0001D3C99C6AB3:3 - STATE"}, ...
  ]
}
```

Prior creating the Domoticz Automation Event script, test switching the PSM ON / OFF from the CLI via curl HTTP REST-API request.

- The HTTP REST-API PUT request requires:  
device serial number, channel number, datapoint parameter and the state,
- The serial number and channel are taken from the Homematic WebUI (see previous): 0001D3C99C6AB3:3,
- The datapoint is taken from the channel information via HTTP REST-API request: <http://ccu-ip:port/device/serialnr/channelnr/>.  
As already mentioned, for a switch, STATE is used.

The curl command to switch the PSM OFF (STATE is set to false):  
The JSON data - The key “v” holds the new value (STATE) false.

```
{v=false}
```

```
curl -X PUT -d '{"v":false}' http://ccu-ip:port/device/0001D3C99C6AB3/3/STATE/~pv
```

After switching the PSM, check if the state has changed to false - can also be checked via the Homematic WebUI.

The JSON response must have value false for the key “v”.

```
curl http://ccu-ip:port/device/0001D3C99C6AB3/3/STATE/~pv
{"ts":1615554829272, "v":false, "s":0}
```

## Automation Event – Trigger Device

### Purpose

To switch the HmIP-PSM on/off triggered by one or more Domoticz device.

This is done by submitting an HTTP REST-API request to the CCU using CCU-Jack.  
The next script enables to define more than one PSM switch.

### Script

```
-- hmip-psm_switch.dzvents
-- Define domoticz devices idx and the devices table for the psm switches
-- The DEVICES idx must match the PSMSWITCHES idx
local DEVICES = { 87, 99999 }
local PSMSWITCHES = {
    ["makelab"] = { idx=87, serialnr = "0001D3C99C6AB3" },
    ["room"] = { idx=99999, serialnr = "NNNNNNNNNNNN" }
}
local URL_PSM_SWITCH = 'http://ccu-ip:port/device/#SERIALNR#/3/STATE/~pv'
local RES_CCU3 = 'HMIP-PSM_SWITCH'

-- Switch a PSM on or off: switchPSM(domoticz, item.idx, item.state)
local function switchPSM(domoticz, idx, state)
    local bstate = true
    if state == "Off" then bstate = false end
    local jsonData = {v=bstate}
    for key, value in pairs(PSMSWITCHES) do
        if value.idx == idx then
            local url = URL_PSM_SWITCH:gsub('#SERIALNR#', value.serialnr)
            domoticz.openURL({url = url, method = 'PUT', callback = RES_CCU3, postData = jsonData})
        end
    end
end

return {
    on = { devices = DEVICES, httpResponses = { RES_CCU3 } },
    logging = { level = domoticz.LOG_INFO },
    execute = function(domoticz, item)
        if (item.isDevice) then switchPSM(domoticz, item.id, item.state) end
        if (item.isHTTPResponse) then domoticz.log(("Status: %s"):format(item.statusText)) end
    end
}
```

### Domoticz Log

```
2021-03-13 11:15:25.146 Dummy: Light/Switch (MakeLab Switch)
2021-03-13 11:15:25.141 Status: User: Admin (IP: NNN.NNN.NNN.NNN) initiated a switch command
(87/MakeLab Switch/On)
2021-03-13 11:15:25.255 Status: dzVents: Info: Handling events for: "MakeLab Switch", value: "On"
2021-03-13 11:15:25.255 Status: dzVents: Info: ----- Start internal script: hmip-psm_switch: Device:
"MakeLab Switch (VirtualSensors)", Index: 87
2021-03-13 11:15:25.256 Status: dzVents: Info: ----- Finished hmip-psm_switch
2021-03-13 11:15:25.256 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
2021-03-13 11:15:25.545 Status: dzVents: Info: Handling httpResponse-events for: "HMIP-PSM_SWITCH"
2021-03-13 11:15:25.545 Status: dzVents: Info: ----- Start internal script: hmip-psm_switch:
HTTPResponse: "HMIP-PSM_SWITCH"
2021-03-13 11:15:25.546 Status: dzVents: Info: Status: OK
2021-03-13 11:15:25.546 Status: dzVents: Info: ----- Finished hmip-psm_switch
```

## Automation Event – Trigger Timer

### Purpose

To switch the PSM ON or OFF triggered by a timer.

For example, switching the coffee machine on at 07:00 and off at 08:00.

### Script

```
--[[  
    hmip-psm_switch_timer.dzvents  
    Switch am HmIP-PSM ON/OFF (Datapoint STATE true/false) at specific time.  
    Dependencies: CCU3 Addon CCU-Jack  
]]--  
  
-- Define domoticz devices idx and the devices table for the psm switches  
-- The DEVICES idx must match the PSMSWITCHES idx  
local DEVICES = { 87, 99999 }  
local PSMSWITCHES = {  
    ["makelab"] = { idx=87, serialnr = "0001D3C99C6AB3" },  
    ["room"] = { idx=99999, serialnr = "NNNNNNNNNNNN" }  
}  
local URL_PSM_SWITCH = 'http://ccu-ip:port/device/#SERIALNR#/3/STATE/~pv'  
local RES_CCU3 = 'HMIP-PSM_SWITCH_TIMER'  
  
local TIMERRULE_ON = 'at 07:00'  
local TIMERRULE_OFF = 'at 08:00'  
  
-- local TIMERRULE = 'every minute'  
  
-- Switch a PSM on or off: switchPSM(domoticz, item.idx, item.state)  
local function switchPSM(domoticz, idx, state)  
    domoticz.log('PSM %d set state to %s'):format(idx, state)  
    local bstate = true  
    if state == "Off" then bstate = false end  
    local jsonData = {v=bstate}  
    for key, value in pairs(PSMSWITCHES) do  
        if value.idx == idx then  
            local url = URL_PSM_SWITCH:gsub('#SERIALNR#', value.serialnr)  
            domoticz.openURL({url = url, method = 'PUT', callback = RES_CCU3, postData = jsonData})  
        end  
    end  
end  
  
return {  
    on = { timer = {TIMERRULE_ON, TIMERRULE_OFF}, httpResponses = { RES_CCU3 } },  
    logging = { level = domoticz.LOG_INFO },  
    execute = function(domoticz, item)  
        -- domoticz.log(item)  
        if (item.isTimer) then  
            domoticz.log(item.trigger)  
            local state  
            if (item.trigger == TIMERRULE_ON) then state = 'On' end  
            if (item.trigger == TIMERRULE_OFF) then state = 'Off' end  
            switchPSM(domoticz, PSMSWITCHES.makelab.idx, state)  
        end  
        if (item.isHTTPResponse) then  
            domoticz.log("Status: %s"):format(item.statusText)  
        end  
    end  
}
```

### Domoticz Log

```
ON
2021-03-13 11:33:00.307 Status: dzVents: Info: ----- Start internal script: hmip-psm_switch_timer:, trigger: "at 11:33"
2021-03-13 11:33:00.307 Status: dzVents: Info: at 11:33
2021-03-13 11:33:00.307 Status: dzVents: Info: PSM 87 set state to On
2021-03-13 11:33:00.307 Status: dzVents: Info: ----- Finished hmip-psm_switch_timer
2021-03-13 11:33:00.308 Status: EventSystem: Script event triggered: /home/pi/domoticz/dzVents/runtime/dzVents.lua
2021-03-13 11:33:00.580 Status: dzVents: Info: Handling httpResponse-events for: "HMIP-PSM_SWITCH_TIMER"
2021-03-13 11:33:00.580 Status: dzVents: Info: ----- Start internal script: hmip-psm_switch_timer: HTTPResponse: "HMIP-PSM_SWITCH_TIMER"
2021-03-13 11:33:00.580 Status: dzVents: Info: Status: OK
2021-03-13 11:33:00.581 Status: dzVents: Info: ----- Finished hmip-psm_switch_timer

OFF
2021-03-13 11:34:00.313 Status: dzVents: Info: ----- Start internal script: hmip-psm_switch_timer:, trigger: "at 11:34"
2021-03-13 11:34:00.313 Status: dzVents: Info: at 11:34
2021-03-13 11:34:00.313 Status: dzVents: Info: PSM 87 set state to Off
2021-03-13 11:34:00.313 Status: dzVents: Info: ----- Finished hmip-psm_switch_timer
2021-03-13 11:34:00.314 Status: EventSystem: Script event triggered: /home/pi/domoticz/dzVents/runtime/dzVents.lua
2021-03-13 11:34:00.595 Status: dzVents: Info: Handling httpResponse-events for: "HMIP-PSM_SWITCH_TIMER"
2021-03-13 11:34:00.595 Status: dzVents: Info: ----- Start internal script: hmip-psm_switch_timer: HTTPResponse: "HMIP-PSM_SWITCH_TIMER"
2021-03-13 11:34:00.595 Status: dzVents: Info: Status: OK
2021-03-13 11:34:00.595 Status: dzVents: Info: ----- Finished hmip-psm_switch_timer
```

# PSM Energy

## Automation Event

### Purpose

To read in regular intervals the HmIP-PSM channel 6 attribute POWER value (in W).

### Script

```
--[[ hmip-psm_power.dzvents
Get, in regular intervals, the value from datapoint POWER (Channel 6 - Homematic WebUI Devices)
Dependencies: CCU3 Addon CCU-Jack

Test getting the value via browser HTTP REST-API request:
http://ccu-ip:2121/device/0001D3C99C6AB3/6/POWER/~pv
The HTTP response JSON data with key "v" holding the power value in W.
{"ts":1615552951348,"v":8.51,"s":0}
]]-
-- Define the devices domoticz idx and the devices table for the psm switches
local PSM_MAKELAB = { idx=84, serialnr = "0001D3C99C6AB3", response = "PSMPOWER_MAKELAB" }
local URL_PSM_POWER = 'http://ccu-ip:port/device/#SERIALNR#/6/POWER/~pv'
local TIMERRULE = 'every minute'

-- Request PSM power
-- requestPSMPower(domoticz, item.idx, item.state)
local function requestPSMPower(domoticz, device)
    local url = URL_PSM_POWER:gsub('#SERIALNR#', device.serialnr)
    domoticz.openURL({url = url, method = 'GET', callback = device.response})
end

-- Set device Electric Instant+Counter Power value only
-- setPSMPower(domoticz, PSM_MAKELAB, item.json.v)
local function setPSMPower(domoticz, device, value)
    -- Updating 84 with value 0.6
    domoticz.log(('Updating %s with value %.1f'):format(device.idx, value))
    domoticz.devices(device.idx).updateElectricity(value, 0)
end

return {
    on = {
        timer = { TIMERRULE },
        httpResponses = { PSM_MAKELAB.response }
    },
    logging = { level = domoticz.LOG_INFO },
    execute = function(domoticz, item)
        if (item.isTimer) then
            requestPSMPower(domoticz, PSM_MAKELAB)
        end
        if (item.isHTTPResponse and item.isJSON) then
            -- PSMPOWER_MAKELAB Status: OK
            domoticz.log(("%" .. item.callback .. " Status: %s"):format(item.statusText))
            if (item.callback == PSM_MAKELAB.response) then
                setPSMPower(domoticz, PSM_MAKELAB, item.json.v)
            end
        end
    end
}
```

### Domoticz Log

```
2021-03-12 16:44:00.535 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
2021-03-12 16:44:00.656 Status: dzVents: Info: Handling httpResponse-events for: "PSMPOWER_MAKELAB"
2021-03-12 16:44:00.656 Status: dzVents: Info: ----- Start internal script: hmip-psm_power:
HTTPResponse: "PSMPOWER_MAKELAB"
2021-03-12 16:44:00.656 Status: dzVents: Info: PSMPOWER_MAKELAB Status: OK
2021-03-12 16:44:00.656 Status: dzVents: Info: Updating 84 with value 0.6
2021-03-12 16:44:00.679 Status: dzVents: Info: ----- Finished hmip-psm_power
```

# Postbox Notifier (HmIP-SWDO)

## Purpose

To watch the postbox for post and notify via Email Notification and Alert Message.

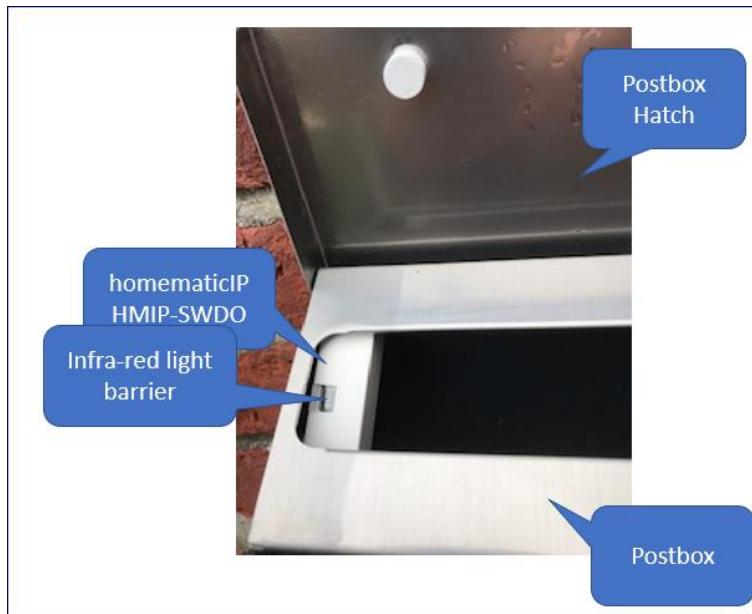
## Solution

A detector detects opening the hatch of the postbox and triggers sending an Email Notification displayed in the Alert Message and Alert Indicator.

The communicating between Domoticz is handled via XML-API or Domoticz HTTP API requests.

The hardware used is a Homematic IP Window and Door Contact – optical (HMIP-SWDO) and Metal Postbox with dimensions (W x H x D) 35 x 40 x 10 cm.

The postbox is ~40 m distance from the CCU (which is in range of 300 m according HMIP-SWDO datasheet).



### Hint

The device HmIP-SWDO is light sensitive, means if it's dark then there might be no signal. If that is an issue, use the device HmIP-SWDM = window / door contact which magnet.

## Communication Flow

1. HMIP-SWDO detects opening the hatch of the postbox.
2. HMIP-SWDO triggers a script which submits a HTTP API Request to the Domoticz system to update Alert Message device to Level 4 with message "Postbox opened".
3. Domoticz updates the device and sends out Email notification as triggered by comparing Alert Level 4 against user variable TH\_ALERTTOEMAIL (which has integer value 4) and sets the Alert Indicator.

### Issue

If the hatch remains open, the HMIP-SWDO keeps on sending alert messages.

To avoid this, the alert message script checks if an alert is sent again within a short timeframe threshold, but

- in case other alert messages are received, these are not captured and
- also, if the threshold is reached, alerts are still sent.

Therefor option two is developed which requires manual intervention.

# Homematic Configuration

## Setup

### Step 1

The device HMIP-SWDO is added to the CCU following the Teach-in devices procedure (used entering the Key and SGTIN data).

The screenshot shows two parts of the HomeMatic Web-UI interface:

- (1) Add new device:** A table titled "Device inbox" lists a single entry: "HMIP-SW DO". The row includes columns for Type, Description, Serial number, Interface category, Transmission mode, Name, Function, Room, Functional test, Action, and Done. The "Action" column shows a green "OK" button.
- (2) Device Status and control:** A table titled "Devices" lists several devices, including "HMIP-SWDO". The "Control" column for "HMIP-SWDO" shows icons for "Open" (blue outline) and "Closed" (black outline).

RaspberryMatic has been replaced by the Homematic CCU3 (HmIP-CCU3).

### Step 2

Create a Homematic program to trigger sending the Alert Message Level 4 (RED) to Domoticz in case the device state changes to open.

Domoticz handles the Alert via dzVents script event "alertmsg\_monitor".

```
IF HMIP-SWDO...:1 is OPEN THEN trigger WHEN updated SCRIPT
```

The screenshot shows a program configuration window:

- Condition (if...):** "Channel status: HMIP-SWDO 0000DA498D5859:1 when open trigger when updated".
- Activity (then..., or else...):** "Script: ... immediately run" with "intrinsic" checked.
- Activity: Then...:** "Stop all current delays before performing the activity (e.g. retriggering). Script: !Function: Postbox Alert string:sfunction = "Postbox Alert"... immediately".
- Activity: Else...:** "Stop all current delays before performing the activity (e.g. retriggering)."

## Script

The script makes use of a CUxD device (see Addon CUxD) to run system commands.

The system command, which triggers a HTTP API Request, to update the Domoticz Alert Message device (IDX=55) with Alert Level 4 (nvalue) and the message (svalue):

```
wget -q -O - 'http://domoticz-ip:8080/json.htm?type=command&param=udevice&idx=IDX&nvalue=4&svalue=MESSAGE'
```

The CUxD device used is “CUxD (28) System” with Function Exec – Remote Control with 19 keys. The key 1 is used to sent HTTP requests.

```
! postbox_notifier.script
! 20200910 by rwbl
string sFunction = "Postbox Notifier";
string sDate = system.Date("%d.%m"); ! sDate = "09.08";
! string sDate = system.Date("%d.%m.%Y"); ! sDate = "09.08.2019";
string sTime = system.Date("%H:%M"); ! sTime = "07:32";
! string sTime = system.Date("%H:%M:%S"); ! sTime = "07:32:00";

! A domoticz alert sensor is used
string nIdx = 189; ! Type: Light/Switch; SubType: On/Off
string sOn = "On"; ! Red
string sText = "Postbox%20opened%20(" # sDate # "%20" # sTime # ")";

! Build the domoticz http rest request url
string cAmp = "&";
string sDomUrl = "'http://domoticz-ip:8080/json.htm'; ! Production
string sUrl =
sDomUrl#"?type=command#&param=switchlight#&idx=#nIdx#&switchcmd=#sOn#'";
WriteLine(sUrl);

! OPTION: Run the command without a return result
! http://domoticz-ip:8080/json.htm?type=command&m=switchlight&idx=189&switchcmd=On
res = dom.GetObject("CUxD.CUX2801001:1.CMD_EXEC").State("wget -q -O - "#sUrl);
```

## Domoticz Configuration

This solution makes use of an Alert Message and Alert Indicator.  
No additional configuration required.

### Automation Event

```
-- postbox_notifier.dzvents
-- Domoticz Idx
-- Switch to keep the state of the postbox flap
local IDX_POSTBOXNOTIFIER_STATE = 189
-- hue bulb used to indicate if postbox has been opened
local IDX_HUE = 355
-- hue color indicating postbox has been openend = mid blue
local HUE_BLUE = 240

-- Timer rules at midnight reset the state else check every hour
local TIMERRULEA = 'at 00:30'
local TIMERRULEB = 'every hour between 8:00 and 20:00'

-- Set the hue on with the color
-- setPostboxIndicator(domoticz, IDX_HUE, HUE_BLUE)
local function setPostboxIndicator(domoticz, idx, hue)
    local level = 20
    local iswhite = false
    domoticz.devices(idx).setHue(hue, level, iswhite)
end

return {
    on = { timer = { TIMERRULEA, TIMERRULEB }, devices = { IDX_POSTBOXNOTIFIER_STATE }, },
    data = { pbnnotified = { initial = 0 } },
    execute = function(domoticz, item)
        if (item.isDevice) then
            if (domoticz.devices(IDX_POSTBOXNOTIFIER_STATE).state=='On') and (domoticz.data.pbnnotified==0) then
                local subject = ('Briefkasten geöffnet (%s)':format(domoticz.helpers.isnowshort(domoticz)))
                local message = 'Aktion: Status zurücksetzen.\r\n'
                domoticz.data.pbnnotified = 1
                domoticz.helpers.alertmsg(domoticz, domoticz.ALERTLEVEL_YELLOW, subject)
                domoticz.notify(subject, message, domoticz.PRIORITY_HIGH)
                setPostboxIndicator(domoticz, IDX_HUE, HUE_BLUE)
            end
            if (domoticz.devices(IDX_POSTBOXNOTIFIER_STATE).state=='Off') and
                (domoticz.data.pbnnotified==1) then
                local message = ('Briefkasten zurückgesetzt (%s)':format(domoticz.helpers.isnowshort(domoticz)))
                domoticz.helpers.alertmsg(domoticz, domoticz.ALERTLEVEL_YELLOW, message)
                domoticz.devices(IDX_HUE).switchOff()
                domoticz.data.pbnnotified = 0
            end
        end
        if (item.isTimer) then
            if (domoticz.devices(IDX_POSTBOXNOTIFIER_STATE).state == 'On') then
                domoticz.devices(IDX_POSTBOXNOTIFIER_STATE).switchOff()
                domoticz.devices(IDX_HUE).switchOff()
                domoticz.data.pbnnotified = 0
            end
        end
    end
}
```

*To avoid sending bulk emails within short timeframe, the notified time is checked against the previous notified time.*

## dzVents Global Data Snippet Alert Message

```

return {
    -- global helper functions
    helpers = {

        -- Idx of devices used (do not forget the comma at the end of the list)
        IDX_ALERTMSG = 55,

        -- Update the alert message with level and text
        alertmsg = function(domoticz, level, msg)
            domoticz.devices(domoticz.helpers.IDX_ALERTMSG).updateAlertSensor(level, msg)
            -- Send email notification in case level = 4 (or other as set by uservar TH_ALERTTOEMAIL)
            if (level == domoticz.variables(IDX_TH_ALERTTOEMAIL).value) then
                domoticz.notify('ALERT', msg, domoticz.PRIORITY_HIGH)
            end
            domoticz.log(( '%s'):format(msg),domoticz.LOG_INFO)
        end,
    }
}

```

### User Variable

Idx	Variable name	Variable type	Current value
14	TH_ALERTTOEMAIL	Integer	4

Depending on the current value, an email is sent.

The function alertmsg parameter level is checked against the current value of the user variable.

# Radiator Thermostats (HmIP-eTRV)

The Homematic IP Radiator Thermostats HmIP-eTRV-B & HmIP-eTRV-2 are installed at several radiators in the house.

The Homematic IP devices are connection partner Radiator Thermostats (manual operation, transmitter) and connected to the CCU.

## Purpose

- To set the temperature setpoint of a radiator thermostat.
- To measure, in regular intervals, the room temperature.
- To check, in regular intervals, the low battery state of the device.

Various solutions have been developed to explore what's possible.

The first solution is currently in place for several thermostats.

1. Domoticz Thermostat SetPoint device  
(USED)
2. Domoticz Selector Switch Device connected to a Thermostat SetPoint device  
(EXPLORED only)
3. Domoticz Python Plugin for a Thermostat with various devices  
(EXPLORED only)

## Solution Thermostat SetPoint Device (In Use)

This solution uses for each Homematic IP Thermostat device a dedicated Domoticz Thermostat SetPoint device.

There are two actions:

### 1. Update Domoticz Thermostat Setpoint triggered by Homematic.

To silent (this avoids endless communication between Homematic and Domoticz) set the Domoticz Thermostat Setpoint triggered by either changing via the Homematic WebUI or the knob/buttons on the Homematic IP Thermostat device.

Communication Homematic to Domoticz, via the Domoticz HTTP API triggering a Domoticz Custom Event with JSON Data. This is handled by a Homematic Script.

```
http://domoticz-ip:<port>/json.htm?type=command&param=customevent&event=homematic_thermostat_sync&data={"id":NNNN,"idx":NNN,"setpoint":NN.N,"temperature":NN.N,"sync":1}
```

Homematic Script	Domoticz Custom Event
homematic_thermostat_sync.script	homematic_thermostat_sync.dzvents

### 2. Set the Homematic Thermostat Setpoint triggered by Domoticz

To change the setpoint of the Homematic IP Thermostat Setpoint via a Domoticz Thermostat Setpoint device.

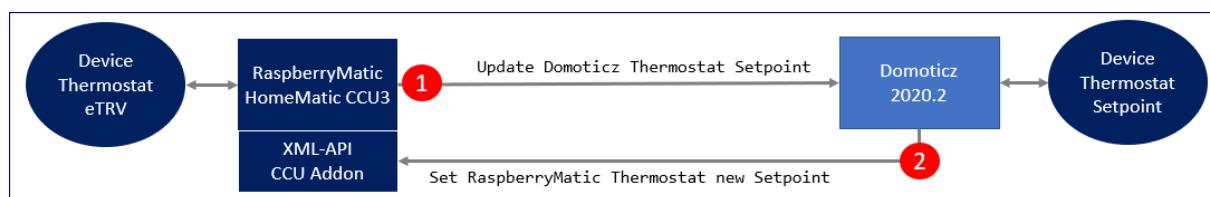
Communication Domoticz to Homematic using the XML-API.

This is handled by a Domoticz automation event with dzVents script.

```
http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id=NNNN&new_value=NN.N
```

Homematic Script	Domoticz Event
n/a	homematic_thermostat_setsetpoint.dzvents

### Communication Flow



## Domoticz Configuration

For each of the Homematic IP thermostats, a Domoticz Virtual Sensor Type Thermostat, SubType SetPoint is defined.

The Domoticz thermostat device description has the Homematic datapoint id JSON format:

```
{"iseid":NNN}
```

The id is used in the Domoticz Automation Event "homematic\_thermostat\_setsetpoint.dzvents" for the HTTP XML-API request to change the new value of the setpoint in HomeMatic.

Domoticz Devices – just one of many devices listed								
Idx	Hardware	ID	Unit	Name	Type	SubType		
341	VirtualSensors	00141A5	1	Bad	Thermostat	SetPoint	21.0	

**Widget Idx=341**

**Dashboard Mobile with setpoint change dialog**

**Widget Edit Device with RaspberryMatic device datapoint defined as JSON**

## Homematic Datapoints

The list of Homematic defined thermostat devices is obtained using the XML-API addon:

```
http://ccu-ip/addons/xmlapi/statelist.cgi
```

Only the datapoint SET\_POINT\_TEMPERATURE is shown as used to control the thermostat setpoint via HTTP XML-API GET or SET requests.

**GET** the thermostat SET\_POINT\_TEMPERATURE:

```
http://ccu-ip/config/xmlapi/state.cgi?datapoint_id=DATAPOINT_ISE_ID
```

**SET** the thermostat SET\_POINT\_TEMPERATURE:

```
http://ccu-ip/config/xmlapi/statechange.cgi?ise_id=DATAPOINT_ISE_ID
```

Example Thermostat datapoint SET\_POINT\_TEMPERATURE.  
The "Thermostat Bad" is used as an example, with ise\_id: 3478.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<stateList>
<device ise_id="3435" name="Thermostat Bad" config_pending="false" unreach="false">
<datapoint ise_id="3478" name="HmIP-RF.00201A99D2097:1 SET_POINT_TEMPERATURE" type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588144602" valueunit="°C" valuetype="4" value="21.000000"/>
</device>
<device ise_id="3489" name="Thermostat Dusche" config_pending="false" unreach="false">
<datapoint ise_id="3532" name="HmIP-RF.00201A49952C8E:1 SET_POINT_TEMPERATURE" type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588139539" valueunit="°C" valuetype="4" value="6.000000"/>
</device>
<device ise_id="3637" name="Thermostat Esszimmer" config_pending="false" unreach="false">
<datapoint ise_id="3680" name="HmIP-RF.000A1A49A0D878:1 SET_POINT_TEMPERATURE" type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588144916" valueunit="°C" valuetype="4" value="21.000000"/>
</device>
<device ise_id="3691" name="Thermostat Flur" config_pending="false" unreach="false">
<datapoint ise_id="3734" name="HmIP-RF.000A1A49A0D8A5:1 SET_POINT_TEMPERATURE" type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588144818" valueunit="°C" valuetype="4" value="21.000000"/>
</device>
<device ise_id="1541" name="Thermostat MakeLab" config_pending="false" unreach="false">
<datapoint ise_id="1584" name="HmIP-RF.000A18A9A64DAC:1 SET_POINT_TEMPERATURE" type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588144325" valueunit="°C" valuetype="4" value="20.000000"/>
</device>
<device ise_id="3543" name="Thermostat Wohnzimmer-1" config_pending="false" unreach="false">
<datapoint ise_id="3586" name="HmIP-RF.00201A49952CB1:1 SET_POINT_TEMPERATURE" type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588144907" valueunit="°C" valuetype="4" value="12.000000"/>
</device>
<device ise_id="3375" name="Thermostat Wohnzimmer-2" config_pending="false" unreach="false">
<datapoint ise_id="3418" name="HmIP-RF.00201A499D76F8:1 SET_POINT_TEMPERATURE" type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588144908" valueunit="°C" valuetype="4" value="21.000000"/>
</device>
</stateList>
```

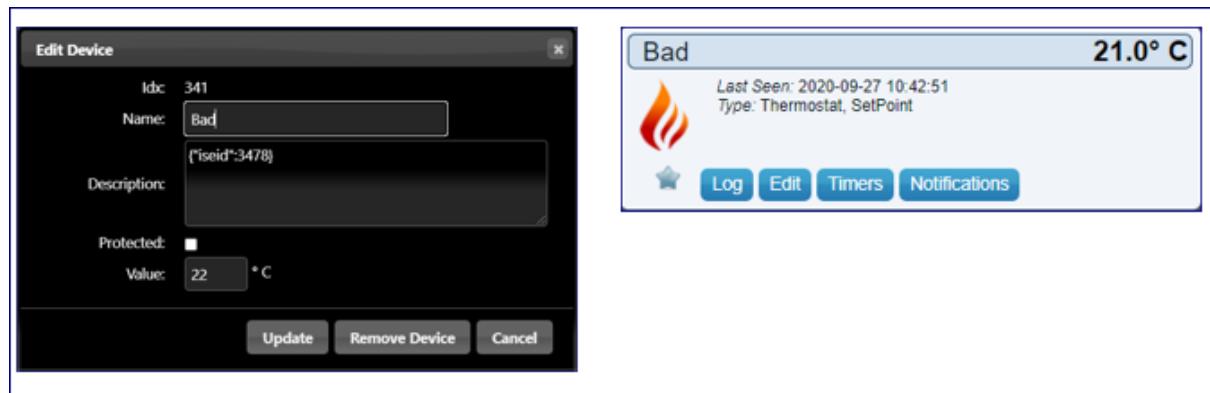
## Domoticz Thermostat Device Define Datapoint

The datapoints are defined in the Domoticz Thermostat Description as JSON.

For the Domoticz automation event with dzVents script, the Homematic Thermostats datapoint ise\_id for the type of SET\_POINT\_TEMPERATURE is required.  
This is defined for each Domoticz Thermostat Device in the device description as JSON string:

```
{"iseid":NNNN}
```

The datapoint ise\_id is taken from the XML-API statelist request (see previous).  
This is an example for the device “Thermostat Bad” with datapoint 3478.

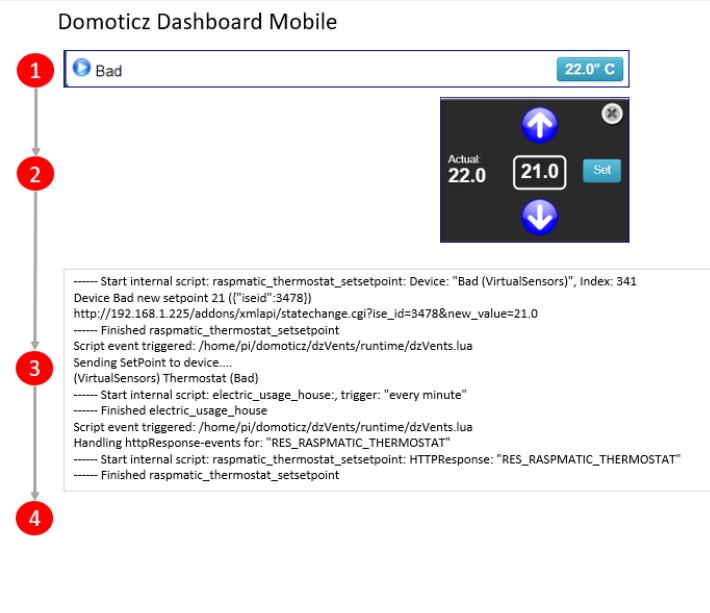


## Action: Domoticz Thermostat Setpoint Change

The flow of actions changing the setpoint of a Domoticz thermostat device via the Domoticz Dashboard.

- Click the temperature button [22.0 °C] of the Thermostat device from the Domoticz Dashboard
- Set the new setpoint to 21.0 and click button [Set] to change the setpoint
- Automation Event dzVents Lua “raspmatic\_thermostat\_setsetpoint” is triggered to submit the new setpoint to RaspberryMatic via HTTP XML-API request datapoint SET\_POINT\_TEMPERATURE
- RaspberryMatic to update the thermostat device setpoint (Homematic WebUI)

Channel	Room	Function	Last modified	Control
Filter	Filter	Filter		
HE02018M9902097.1 Radiator-thermostat (transmitter, receiver, transmitter)	Bad		27.09.2020 11:36:44	<div style="display: flex; justify-content: space-between;"> <div style="flex: 1;"> <p>Temperature: 23.10°C</p> <p>Window status: Closed</p> <p>Valve opening: 41%</p> <p>5°C Auto mode</p> <p>Manual mode</p> <p>30°C</p> <p>Week profile: 1</p> </div> <div style="flex: 1;"> <p>21.0°C</p> <p>Up</p> <p>Down</p> <p>Set</p> </div> </div>



## Automation Event

```
-- homematic_thermostat_setsetpoint.dzvents
-- Set new setpoint Homematic IP thermostat triggered by a Domoticz thermostat device setpoint change.
-- The Homematic device datapoint is deviced as JSON string in the Domoticz device description

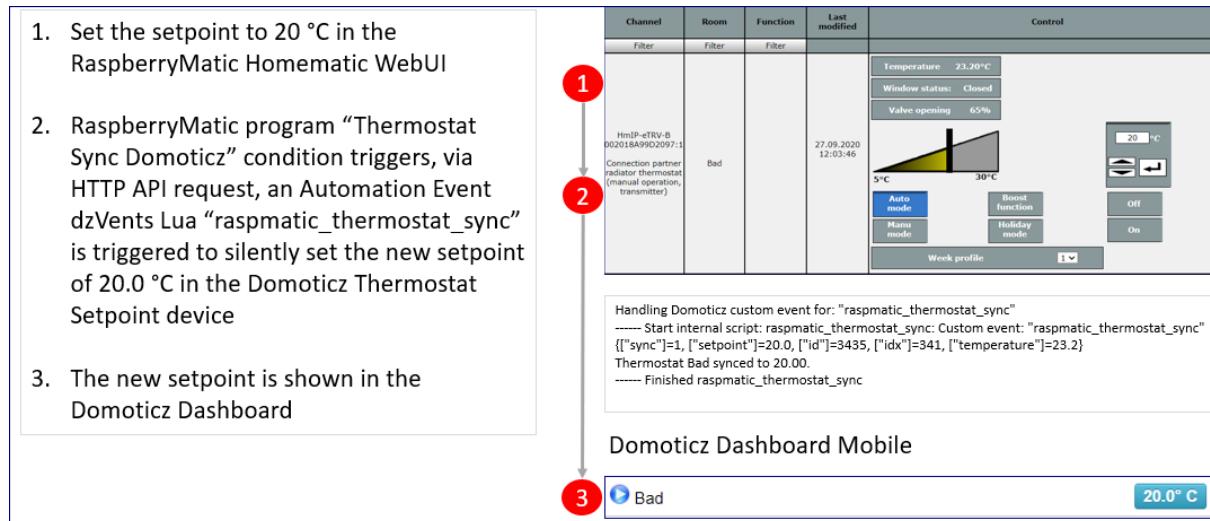
-- url http xml-api request to set a new value
local URL_CCU3 = 'http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id=#ISEID#&new_value=#NEWVALUE#';
-- define the unique (across all dzVents) callback
local RES_CCU3 = "RES_CCU_THERMOSTAT";
-- list of the idx of all thermostats (see Domoticz WebUI > Setup > Devices)
local idxdevicestable = {336,337,338,339,340,341,342};

return {
  on = {
    devices = idxdevicestable,
    httpResponses = { RES_CCU3 }
  },
  execute = function(domoticz, item)
    -- Device changes a setpoint
    if (item.isDevice) then
      local device = domoticz.devices(item.id)
      local setpoint = device.setPoint;
      domoticz.log(string.format('Device %s new setpoint %.0f (%s)', device.name, device.setPoint,
device.description), domoticz.LOG_INFO)
      -- Read and convert the JSON description to a table
      local jsondesc = domoticz.utils.fromJSON(device.description)
      -- Get the Homematic datapoint id for the device setup
      local iseid = jsondesc.iseid
      -- Build the url to set the new setpoint in Homematic
      -- Replace the placeholders with the id and newvalue
      local url = URL_CCU3:gsub("#ISEID#", iseid):gsub("#NEWVALUE#",setpoint)
      domoticz.log(url)
      domoticz.openURL({ url = url, method = 'POST', callback = RES_CCU3 })
    end
    --
    if (item.isHTTPResponse) then
      if (item.ok and item.callback == RES_CCU3 ) then
        -- domoticz.log(item.data, domoticz.LOG_INFO)
      else
        domoticz.log(string.format('Problem handling the request: %s',url), domoticz.LOG_ERROR)
        domoticz.log(item.data, domoticz.LOG_ERROR)
      end
    end
  end
}
```

## Action: Homematic Thermostat Setpoint Change

The flow of actions Homematic trigger Thermostat Setpoint Change via Homematic WebUI. The flow of actions is also triggered in case the thermostat is manually changed via knob or +/- buttons (depending on thermostat model).

The change is also reflected in the Homematic WebUI.



## Homematic Program updates Domoticz Thermostat Setpoint

Define condition for all thermostat devices to trigger a script to update the Domoticz thermostat setpoint silently:  
if the thermostat setpoint temperature is updated (“trigger when updated”) having a value of more than 4.5°C

**Condition (if...)**

Name	Description	Condition (if...)	Activity (then..., or else...)	Action
Thermostat Sync Domoticz	Sync the thermostat setpoint with the corresponding Domoticz device.	Channel status: HmIP-eTRV-2 000A18A9A64DAC:1 when Setpoint temperature within value range / with value more than 4.50° C trigger when updated	Script: ... immediately run	<input type="checkbox"/> intrinsic

**Condition: If...**

Set value range

Value range / with value / Value:  
 from 4.50 °C and less than 0.00 °C  
 equal 4.50 °C  
 not equal 4.50 °C  
 more than 4.50 °C  
 less than 4.50 °C  
 more or equal 4.50 °C  
 less or equal 4.50 °C

**OK**

**Activity: Then...**  Stop all current delays before performing the activity (e.g. retriggering).  
**Script**

**Activity: Else...**  Stop all current delays before performing the activity (e.g. retriggering).

## Homematic Script updates Domoticz Thermostat Setpoint

If the previous defined conditions apply, then the activity is to execute the script below which submits a HTTP API request to Domoticz, handled by a Custom Event. The Custom Event parameter “data” contains in JSON format the required fields to update the setpoint of the thermostat device in Domoticz.

```

! If the setpoint of a thermostat gets updated, update the corresponding domoticz thermostat silent (no
active setpoint change).
! Via cuxd an http api request for a customevent is submitted to domoticz.
! Custom event data parameter must be in json format.
! Note: Use http://ccu-ip/config/xmlapi/statelist.cgi to get device iseid and datapoints.
! The thermostat trigger is value > 4.5 when updated
string cAmp = "&";
! Domoticz system url base for the http api request
string urlBase = "'http://domoticz-ip:port/json.htm"; ! Production
! string urlBase = "'http://domoticz-ip:port/json.htm"; ! Development
string customEvent = "homematic_thermostat_sync";
string deviceID = "UNKNOWN";      ! 1541
string setPoint = "";            ! 20.5
string actTemperature = "";      ! 20.3
string deviceIdx = "UNKNOWN";    ! 336
string syncState = "1";          ! must be 1 for silent update in domoticz
! Get the object datapoint of the thermostat which setpoint is updated
object objDatapoint = dom.GetObject ("$src$");
if (objDatapoint) {
    if (objDatapoint.Value()) {
        object objChannel = dom.GetObject (objDatapoint.Channel());
        deviceID = objChannel.Device();
        setPoint = objChannel.DPByHssDP("SET_POINT_TEMPERATURE").Value();
        actTemperature = objChannel.DPByHssDP("ACTUAL_TEMPERATURE").Value();
    }
}
! Map the Homematic thermostat datapoint to the domoticz idx
if (deviceID == "3489") { deviceIdx = 342; }  ! Dusche
if (deviceID == "3435") { deviceIdx = 341; }  ! Bad
if (deviceID == "3691") { deviceIdx = 340; }  ! Flur
if (deviceID == "3375") { deviceIdx = 339; }  ! Wohnzimmer-2
if (deviceID == "3543") { deviceIdx = 338; }  ! Wohnzimmer-1
if (deviceID == "3637") { deviceIdx = 337; }  ! Esszimmer
if (deviceID == "1541") { deviceIdx = 336; }  ! MakeLab
! Custom event data parameter must be in json format
string data =
'{"id": "#deviceID#", "idx": "#deviceIdx#", "setpoint": "#setPoint#", "temperature": "#actTemperature#", "sync": "#syncState#"}';
! Build the Domoticz http rest request url to update the device using customevent
string urlRequest =
urlBase#"?type=command#cAmp#"param=customevent#cAmp#"event="#customEvent#cAmp#"data="#data#"";;
WriteLine(urlRequest);
! Run the command without a return result
cmdRes = dom.GetObject("CUxD.CUX2801001:1.CMD_EXEC").State("wget -q -O - "#urlRequest);

```

## Automation Event

Domoticz dzVents automation event to sync the new Homematic Thermostat setpoint with the Domoticz thermostat device.

```
-- homematic_thermostat_sync.dzvents
-- Sync the domoticz thermostat setpoint with the Homematic thermostat setpoint.
-- The event is triggered by the Homematic setpoint change program.

local CUSTOMEVENT_NAME = "homematic_thermostat_sync"

return {
  on = {
    customEvents = {
      CUSTOMEVENT_NAME
    }
  },
  data = {},
  logging = {},
  execute = function(domoticz, item)
    if (item.isCustomEvent) then
      -- {[{"setpoint":23.0, "temperature":22.1, "sync":1, "id":3489, "idx":342}]
      domoticz.log(item.data);
      if item.data.sync == 1 then
        domoticz.devices(item.data.idx).updateSetPoint(item.data.setpoint).silent()
        domoticz.log(string.format("Thermostat %s synced to %.2f.",,
domoticz.devices(item.data.idx).name, item.data.setpoint))
      end
    else
      -- second parameter can be anything, number, string, boolean or table
      -- domoticz.emitEvent('MyEvent', 'Some data')
    end
  end
}
```

## Action: Set All Thermostats On Off

Set for all thermostats the setpoint On = 5°C or Off = 21°C.

### Domoticz Configuration

#### Create Device

Create virtual sensor (Hardware Dummy): Name=Thermostats, Sensor Type: Switch

The devices list (GUI > Setup > Devices) shows the new device with properties:

```
IDX=389, Hardware=VirtualSensors, Name=Thermostats, Type=Light/Switch, SubType=Switch, Data=Off
```

The switch device is added as a widget to the GUI > Tab Switches.

Change the device icon to Heating Device via the Widget Edit Button > Switch Icon > Select Heating (Heating Device).



### Automation Event

Create an automation event dzVents to switch all thermostats On or Off.

The temperature settings for the state On and Off are defined as User Variables:

#### IDX, Name, Type, Value

```
21, DEF_THERMOSTAT_ON, Integer, 21
22, DEF_THERMOSTAT_OFF, Integer, 5
```

```
local IDX_SWITCH_THERMOSTATS = 389
local IDX_DEF_THERMOSTAT_ON = 21;
local IDX_DEF_THERMOSTAT_OFF = 22;
local DEVTYPE_THERMOSTAT = 'Thermostat'
return {
    on = { devices = { IDX_SWITCH_THERMOSTATS } },
    logging = { level = domoticz.LOG_INFO, marker = 'HOMEMATICSWITCHTHERMOSTATS', },
    execute = function(domoticz, device)
        local setpoint = 0
        -- Set the setpoint from the user variable depending on switch state
        if device.state == 'On' then
            setpoint = domoticz.variables(IDX_DEF_THERMOSTAT_ON).value;
        else
            setpoint = domoticz.variables(IDX_DEF_THERMOSTAT_OFF).value;
        end
        -- Loop over all devices, select the thermostat device (Type) and set the setpoint
        domoticz.devices().forEach(function(device)
            if (device.deviceType == DEVTYPE_THERMOSTAT) then
                domoticz.devices(device.idx).updateSetPoint(setpoint).afterSec(2)
                domoticz.log(string.format("Thermostat %s setpoint changed to %.2f.", device.name, setpoint))
            end
        end)
    end
}
```

## Solution Selector Switch & Thermostat (Explored)

This is a simple solution making use of a Domoticz Selector Switch to select a thermostat from the list of thermostats.

The setpoint for the selected thermostat is set by using a single thermostat device.

The communication between Domoticz and Homematic is handled with the XML-API CCU addon.

### Domoticz Configuration

Domoticz Devices – just one of many devices listed

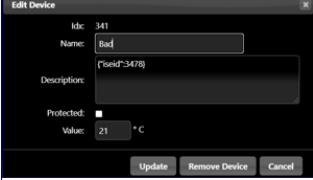
Idx	Hardware	ID	Unit	Name	Type	SubType
341	VirtualSensors	00141A5	1	Bad	Thermostat	SetPoint

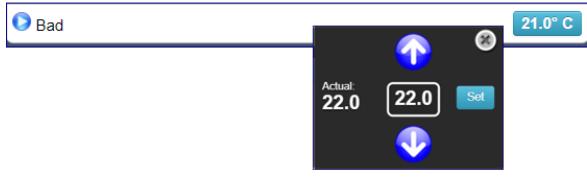
Widget Idx=341

Widget Edit Device with RaspberryMatic device datapoint

Dashboard Mobile with setpoint change dialog

Domoticz Log Automation Events dYvents Lua

```

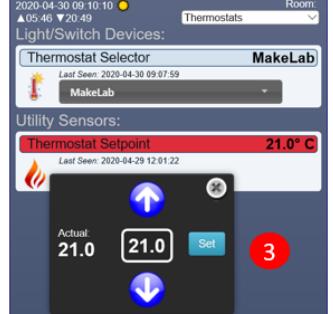
2020-09-27.... Handling events for: "Bad", value: "22.0"
2020-09-27.... ----- Start internal script: rasmatic_thermostat_setsetpoint: Device: "Bad (VirtualSensors)", Index: 341
2020-09-27.... Device Bad never setpoint 22 (["seid":3478])
2020-09-27.... http://192.168.1.225/addons/xmlapi/statechange.cgi?se_id=3478&new_value=22.0
2020-09-27.... ----- Finished rasmatic_thermostat_setsetpoint

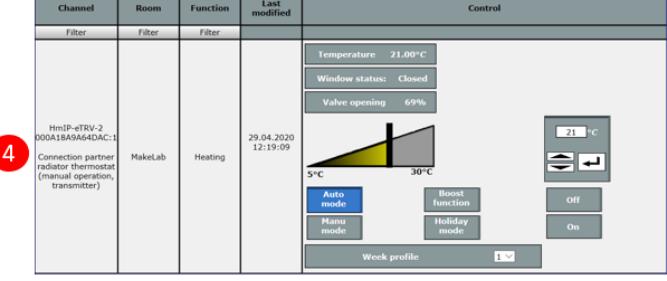
2020-09-27.... Handling httpResponse-events for: "RES_RASPMATIC_THERMOSTAT"
2020-09-27.... ----- Start internal script: rasmatic_thermostat_setsetpoint: HTTPResponse: "RES_RASPMATIC_THERMOSTAT"
2020-09-27.... ----- Finished rasmatic_thermostat_setsetpoint
2020-09-27.... Handling Domoticz custom event for: "rasmatic_thermostat_sync"
2020-09-27.... ----- Start internal script: rasmatic_thermostat_sync: Custom event: "rasmatic_thermostat_sync"
2020-09-27.... [{"setpoint":22.0, ["idx":341, {"temperature":23.1, ["id":3435, "sync":1}]}
2020-09-27.... Thermostat Bad synced to 22.0
2020-09-27.... ----- Finished rasmatic_thermostat_sync

```

### Select Thermostat & Change Setpoint

1. Select the thermostat:  
Domoticz device: "Thermostat Selector"
2. Get the RaspberryMatic selected thermostat device value for the datapoint SET\_POINT\_TEMPERATURE & update the Domoticz device: "Thermostat Setpoint" (via HTTP XML-API request)
3. Change the setpoint:  
Domoticz device: "Thermostat Setpoint"
4. Update RaspberryMatic thermostat device setpoint (via HTTP XML-API request)





### Thermostat Devices

Get all devices defined in CCU by sending an HTTP XML-API request to the CCU with parameter the script “statelist.cgi”.

```
http://ccu-ip/addons/xmlapi/statelist.cgi
```

From the HTTP XML response, only the datapoint type SET\_POINT\_TEMPERATURE is required as used to control the thermostat, gain, via HTTP XML-API requests with script as parameter.

(extract from the statelist request)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<statelist>
<device ise_id="3435" name="Thermostat Bad" config_pending="false" unreach="false">
<datapoint ise_id="3478" name="HmIP-RF.002018A99D2097:1.SET_POINT_TEMPERATURE"
type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588144602" valueunit="°C" valuetype="4"
value="21.00000"/>
</device>
<device ise_id="3489" name="Thermostat Dusche" config_pending="false" unreach="false">
<datapoint ise_id="3532" name="HmIP-RF.00201A49952CB8:1.SET_POINT_TEMPERATURE"
type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588139539" valueunit="°C" valuetype="4"
value="6.00000"/>
</device>
<device ise_id="3637" name="Thermostat Esszimmer" config_pending="false" unreach="false">
<datapoint ise_id="3680" name="HmIP-RF.000A1A49A0D878:1.SET_POINT_TEMPERATURE"
type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588144916" valueunit="°C" valuetype="4"
value="21.00000"/>
</device>
<device ise_id="3691" name="Thermostat Flur" config_pending="false" unreach="false">
<datapoint ise_id="3734" name="HmIP-RF.000A1A49A0D8A5:1.SET_POINT_TEMPERATURE"
type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588144818" valueunit="°C" valuetype="4"
value="21.00000"/>
</device>
<device ise_id="1541" name="Thermostat MakeLab" config_pending="false" unreach="false">
<datapoint ise_id="1584" name="HmIP-RF.000A18A9A64DAC:1.SET_POINT_TEMPERATURE"
type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588144325" valueunit="°C" valuetype="4"
value="20.00000"/>
</device>
<device ise_id="3543" name="Thermostat Wohnzimmer-1" config_pending="false" unreach="false">
<datapoint ise_id="3586" name="HmIP-RF.00201A49952CB1:1.SET_POINT_TEMPERATURE"
type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588144907" valueunit="°C" valuetype="4"
value="12.00000"/>
</device>
<device ise_id="3375" name="Thermostat Wohnzimmer-2" config_pending="false" unreach="false">
<datapoint ise_id="3418" name="HmIP-RF.00201A499D76F8:1.SET_POINT_TEMPERATURE"
type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588144908" valueunit="°C" valuetype="4"
value="21.00000"/>
</device>
</statelist>
```

## GET the thermostat SET\_POINT\_TEMPERATURE

```
http://ccu-ip/config/xmlapi/state.cgi?datapoint_id=DATAPOINT_ISE_ID
```

## SET the thermostat SET\_POINT\_TEMPERATURE

```
http://ccu-ip/config/xmlapi/statechange.cgi?ise_id=DATAPOINT_ISE_ID
```

*The datapoints are defined in a dzVents automation event.*

## Automation Event

For the dzVents automation event, the Homematic Thermostat datapoint ise\_id for the attribute type SET\_POINT\_TEMPERATURE, is required.

The datapoints are used to get or set the setpoint.

In dzVents an array is defined according to the thermostats defined in the selector switch:

```
[level]=ise_id SET_POINT_TEMPERATURE
```

```
local ID_THERMOSTAT_DATAPOINTS = {[10]=1584, [20]=3586, [30]=3418, [40]=3680, [50]=3478, [60]=3734, [70]=3532};
```

Name	Ise_id	Level	Level Name	
Thermostat MakeLab	1584	10	MakeLab	
Thermostat Wohnzimmer-1	3586	20	Wohn-1	
Thermostat Wohnzimmer-2	3418	30	Wohn-2	
Thermostat Esszimmer	3680	40	Essen	
Thermostat Bad	3478	50	Bad	
Thermostat Flur	3734	60	Flur	
Thermostat Dusche	3532	70	Dusche	

**IMPORTANT: if the level order is changed in the device widget, the array must be changed as well!**

Event name: thermostat\_control\_selector\_switch.dzvents  
(below code is stripped)

```
local IDX_THERMOSTAT_SELECTOR = 114, IDX_THERMOSTAT_SETPOINT = 115;
local ID_THERMOSTAT_DATAPOINTS = {[10]=1584, [20]=3586, [30]=3418, [40]=3680, [50]=3478, [60]=3734, [70]=3532};
local datapointSelected;
local URL_CCU_STATECHANGE = 'http://ccu-ip/config/xmlapi/statechange.cgi?ise_id=' , URL_CCU_STATE = 'http://ccu-ip/config/xmlapi/state.cgi?datapoint_id=';
local RES_CCU_SETSETPOINT = 'res_homematic_setsetpoint', RES_CCU_GETSETPOINT = 'res_homematic_getsetpoint';
local DECIMALS = 1;
function round(number, decimals)
    local power = 10^decimals
    return math.floor(number * power) / power
end
return {
    on = {devices = {IDX_THERMOSTAT_SELECTOR, IDX_THERMOSTAT_SETPOINT},
          httpResponses = {RES_CCU_SETSETPOINT,RES_CCU_GETSETPOINT,{}}},
          data = {updateHomematic = { initial = 1 }},
          execute = function(domoticz, item)
            if (item.isDevice) then
                datapointSelected = ID_THERMOSTAT_DATAPOINTS[domoticz.devices( IDX_THERMOSTAT_SELECTOR ).level];
                if (item.idx == IDX_THERMOSTAT_SELECTOR) then
                    domoticz.data.updateHomematic = 0;
                    local urlrm = string.format("%s%d", URL_CCU_STATE, datapointSelected);
                    domoticz.openURL({url = urlrm, method = 'POST', callback = RES_CCU_GETSETPOINT});
                end
                if (item.idx == IDX_THERMOSTAT_SETPOINT) then
                    if (domoticz.data.updateHomematic == 1) then
                        local newsetpoint = round(item.setPoint,DECIMALS);
                        local urlrm = string.format("%s%d&new_value=%2f", URL_CCU_STATECHANGE,
                        datapointSelected, newsetpoint);
                        domoticz.openURL({url = urlrm, method = 'POST', callback = RES_CCU_SETSETPOINT});
                    end
                    if (domoticz.data.updateHomematic == 0) then domoticz.data.updateHomematic = 1; end
                end
            end
            if (item.isHTTPResponse) then
                if (item.statusCode == 200) then

```

```
        if (item.callback == RES_CCU_GETSETPOINT) then
            datapointSelected =
ID_THERMOSTAT_DATAPOINTS[domoticz.devices(IDX_THERMOSTAT_SELECTOR).level];
            local setpoint = domoticz_applyXPath(item.data,'//datapoint[@ise_id="" ..
datapointSelected .. """]/@value');
            setpoint = round(setpoint, DECIMALS);
            local urldom = string.format("http://domoticz-
ip:8080/json.htm?type=command&param=setsetpoint&idx=%d&setpoint=%.2f",IDX_THERMOSTAT_SETPOINT,setpoint)
;
            domoticz.openURL({url = urldom, method = 'POST'});
        end
        if (item.callback == RES_CCU_SETSETPOINT) then datapointSelected =
ID_THERMOSTAT_DATAPOINTS[domoticz.devices(IDX_THERMOSTAT_SELECTOR).level]; end
        else
            domoticz.log('[ERROR] handling HTTP request:' .. item.statusText, domoticz.LOG_ERROR)
        end
    end
end
}
```

## Solution Plugin (Explored)

- Various Domoticz device timers are put in place to control the temperature depending on the radiator location.
- If an open window is detected by the thermostat, the setpoint is decreased to 12 °C until the window is closed, then the setpoint is back to previous value.
- Each device has a battery which is regularly checked, and alert given if below threshold 2.2V (defined in the Homematic WebUI for each device).
- Just as an experiment: a [Node-RED Dashboard](#).

A Domoticz Python Plugin has been created, called **Homematic IP Radiator Thermostat (HmIP-eTRV)**.

## Installation

These steps apply for every Radiator Thermostat Device:

### Homematic WebUI:

Add the thermostat device to Homematic (Teach-in)

Download the HmIP-eTRV plugin repository from [here](#)

On the Domoticz system, create folder: /home/pi/domoticz/plugins/hmip-etriv

Copy the file plugin.py to the folder: /home/pi/domoticz/plugins/hmip-etriv

Restart Domoticz either via command line or Domoticz WebUI:

- Command: “sudo service domoticz.sh restart” OR
- Domoticz WebUI > Setup > More Options > Restart System

The plugin requires the parameters:

- CCU IP Address
- Device id of the thermostat device HmIP-eTRV-B or HmIP-eTRV-2
- Datapoint ids for the selected device:  
SET\_POINT\_TEMPERATURE, ACTUAL\_TEMPERATURE, LOW\_BAT, LEVEL  
as comma separated list in this order.

### Note

Obtain the device & datapoint id's by requesting the device state list from the CCU via URL:

<http://ccu-ip/addons/xmlapi/statelist.cgi>

Get the devices state list: <http://ccu-ip-address/addons/xmlapi/statelist.cgi> with XML tree result.

1. Select Device HMIP-eTRV-B with name "Thermostat Dining Room"
2. Get the Device ID **3637**
3. Get the Datapoints IDs for the properties:  
SET\_POINT\_TEMPERATURE=3680,ACTUAL\_TEMPERATURE=3663,LOW\_BAT=3645,LEVEL=3672

The ids are used to

1. get the actual temperature, battery low voltage status & level
2. set the thermostat setpoint.

```
<device config_pending="False" unreach="False" ise_id="3637" name="Thermostat Dining Room">
  <channel ise_id="3639" name="HmIP-RF-0001A49A0D878:0.CONFIG_PENDING" operations="5" timestamp="1577009479" valueunit="" valueType="2" value="0" type="CONFIG_PENDING"/>
  <channel ise_id="3640" name="HmIP-RF-0001A49A0D878:0.DUTY_CYCLE" operations="5" timestamp="1577009479" valueunit="" valueType="2" value="0" type="DUTY_CYCLE"/>
  <channel ise_id="3645" name="HmIP-RF-0001A49A0D878:0.LOW_BAT" operations="5" timestamp="1577009479" valueunit="" valueType="2" value="0" type="LOW_BAT"/>
  <channel ise_id="3649" name="HmIP-RF-0001A49A0D878:0.OPERATING_VOLTAGE" operations="5" timestamp="1577009479" valueunit="" valueType="4" value="3.00000" type="OPERATING_VOLTAGE"/>
  <channel ise_id="3650" name="HmIP-RF-0001A49A0D878:0.OPERATING_VOLTAGE_STATUS" operations="5" timestamp="1577009479" valueunit="" valueType="16" value="0" type="OPERATING_VOLTAGE_STATUS"/>
  <channel ise_id="3651" name="HmIP-RF-0001A49A0D878:0.RSSI_DEVICE" operations="5" timestamp="1577009479" valueunit="" value="206" type="RSSI_DEVICE"/>
  <channel ise_id="3652" name="HmIP-RF-0001A49A0D878:0.RSSI_PEER" operations="5" timestamp="157696804" valueunit="" value="200" type="RSSI_PEER"/>
  <channel ise_id="3653" name="HmIP-RF-0001A49A0D878:0.UNREACH" operations="5" timestamp="1577009479" valueunit="" valueType="2" value="0" type="UNREACH"/>
  <channel ise_id="3657" name="HmIP-RF-0001A49A0D878:0.UPDATE_PENDING" operations="5" timestamp="1576940478" valueunit="" valueType="2" value="0" type="UPDATE_PENDING"/>
</channels>
<datapoint ise_id="3661" name="HmIP-eTRV-2 0001A49A0D878:1.OPERATE" operations="1" value="true" visible="true" index="1" type="ACTIVE_PROFILE"/>
<datapoint ise_id="3662" name="HmIP-RF-0001A49A0D878:1.ACTUAL_TEMPERATURE_PROFILE" operations="2" timestamp="1577009479" valueunit="" valueType="16" value="1" type="ACTUAL_TEMPERATURE"/>
<datapoint ise_id="3663" name="HmIP-RF-0001A49A0D878:1.ACTUAL_TEMPERATURE_STATUS" operations="5" timestamp="1577009479" valueunit="" valueType="16" value="0" type="ACTUAL_TEMPERATURE_STATUS"/>
<datapoint ise_id="3664" name="HmIP-RF-0001A49A0D878:1.BOOT_MODE" operations="6" timestamp="1577009479" valueunit="" valueType="1" value="false" type="BOOT_MODE"/>
<datapoint ise_id="3665" name="HmIP-RF-0001A49A0D878:1.BOOT_TIME" operations="5" timestamp="1577009479" valueunit="" value="16" type="BOOT_TIME"/>
<datapoint ise_id="3666" name="HmIP-RF-0001A49A0D878:1.BOOT_TIMER" operations="5" timestamp="1577009479" valueunit="" value="16" type="BOOT_TIMER"/>
<datapoint ise_id="3667" name="HmIP-RF-0001A49A0D878:1.CONTROL_DIFFERENTIAL_TEMPERATURE" operations="2" timestamp="0" valueunit="" valueType="4" type="CONTROL_DIFFERENTIAL_TEMPERATURE"/>
<datapoint ise_id="3668" name="HmIP-RF-0001A49A0D878:1.CONTROL_MODE" operations="2" timestamp="0" valueunit="" valueType="16" value="1" type="CONTROL_MODE"/>
<datapoint ise_id="3669" name="HmIP-RF-0001A49A0D878:1.DURATION_UNIT" operations="2" timestamp="0" valueunit="" valueType="16" value="1" type="DURATION_UNIT"/>
<datapoint ise_id="3670" name="HmIP-RF-0001A49A0D878:1.FROST_PROTECTION" operations="2" timestamp="0" valueunit="" valueType="16" type="FROST_PROTECTION"/>
<datapoint ise_id="3671" name="HmIP-RF-0001A49A0D878:1.FROST_PROTECTION_SETPOINT" operations="5" timestamp="1577009479" valueunit="" value="0" type="FROST_PROTECTION_SETPOINT"/>
<datapoint ise_id="3672" name="HmIP-RF-0001A49A0D878:1.LEVEL" operations="7" timestamp="1577009479" valueunit="" valueType="4" value="0.12000" type="LEVEL"/>
<datapoint ise_id="3673" name="HmIP-RF-0001A49A0D878:1.LEVEL_STATUS" operations="5" timestamp="1577009479" valueunit="" valueType="16" value="0" type="LEVEL_STATUS"/>
<datapoint ise_id="3674" name="HmIP-RF-0001A49A0D878:1.PARTY_MODE" operations="5" timestamp="1577009479" valueunit="" valueType="2" value="0" type="PARTY_MODE"/>
<datapoint ise_id="3675" name="HmIP-RF-0001A49A0D878:1.PARTY_SETPOINT_TEMPERATURE" operations="5" timestamp="1577009479" valueunit="" value="0" type="PARTY_SETPOINT_TEMPERATURE"/>
<datapoint ise_id="3676" name="HmIP-RF-0001A49A0D878:1.PARTY_TIME_END" operations="7" timestamp="0" valueunit="" valueType="20" type="PARTY_TIME_END"/>
<datapoint ise_id="3677" name="HmIP-RF-0001A49A0D878:1.PARTY_TIME_START" operations="7" timestamp="0" valueunit="" valueType="20" value="1" type="PARTY_TIME_START"/>
<datapoint ise_id="3678" name="HmIP-RF-0001A49A0D878:1.QUICK_VETO_TIME" operations="5" timestamp="1577009479" valueunit="" valueType="16" value="0" type="QUICK_VETO_TIME"/>
<datapoint ise_id="3679" name="HmIP-RF-0001A49A0D878:1.SET_POINT_TEMPERATURE" operations="5" timestamp="1577009479" valueunit="" value="25" type="SET_POINT_TEMPERATURE"/>
<datapoint ise_id="3680" name="HmIP-RF-0001A49A0D878:1.SET_POINT_TEMPERATURE_SETPOINT" operations="5" timestamp="1577009479" valueunit="" value="25" type="SET_POINT_TEMPERATURE_SETPOINT"/>
<datapoint ise_id="3681" name="HmIP-RF-0001A49A0D878:1.SWITCH_POINT_OCCURRED" operations="5" timestamp="1577009479" valueunit="" valueType="2" value="0" type="SWITCH_POINT_OCCURRED"/>
<datapoint ise_id="3682" name="HmIP-RF-0001A49A0D878:1.VALVE_ADAPTION" operations="5" timestamp="0" valueunit="" valueType="2" value="0" type="VALVE_ADAPTION"/>
<datapoint ise_id="3683" name="HmIP-RF-0001A49A0D878:1.WINDOW_STATE" operations="5" timestamp="1577009479" valueunit="" valueType="16" value="0" type="WINDOW_STATE"/>
</channels>
<channel ise_id="3685" name="HmIP-eTRV-2 0001A49A0D878:2.OPERATE" operations="1" value="true" visible="true" index="2" type="ACTIVE_PROFILE"/>
<channel ise_id="3686" name="HmIP-eTRV-2 0001A49A0D878:3.OPERATE" operations="1" value="true" visible="true" index="3" type="ACTIVE_PROFILE"/>
<channel ise_id="3687" name="HmIP-eTRV-2 0001A49A0D878:4.OPERATE" operations="1" value="true" visible="true" index="4" type="ACTIVE_PROFILE"/>
<channel ise_id="3688" name="HmIP-eTRV-2 0001A49A0D878:5.OPERATE" operations="1" value="true" visible="true" index="5" type="ACTIVE_PROFILE"/>
<channel ise_id="3689" name="HmIP-eTRV-2 0001A49A0D878:6.OPERATE" operations="1" value="true" visible="true" index="6" type="ACTIVE_PROFILE"/>
<channel ise_id="3690" name="HmIP-eTRV-2 0001A49A0D878:7.OPERATE" operations="1" value="true" visible="true" index="7" type="ACTIVE_PROFILE"/>
</device>
```

## Important

Prior adding new hardware, ensure Domoticz accepts new hardware as devices will be created when adding the plugin.

Check: Domoticz WebUI > Setup > Settings > Hardware

Either mark accept or click Allow for 5 minutes

Goto Domoticz WebUI > Setup > Hardware

Select Type: Homematic IP Radiator Thermostat (HMIP-eTRV)

Give a name, i.e., Thermostat Dining

Set initially the parameter: Keep debug to true

Click button Add

Check the Domoticz log if the four devices are created and updated:

Thermostat Dining - Setpoint, Thermostat Dining - Temperature, Thermostat Dining - Battery, Thermostat Dining - Valve

Check the device widgets on the Domoticz WebUI tabs.

If all OK, set the hardware parameter Debug to false

Optional steps (see also screenshots below):

- Add the devices to a room plan, i.e., Dining Room
- Add a device timer to ensure the thermostat is turned Off to avoid heating overnight
- Add a switch device to turn the thermostat off

## Notes

If adding device timers for the setpoint, set the thermostat to "Manu mode" in the Homematic WebUI else if "Auto mode" is set, the selected Week profile is used.

## Add new Hardware (Domoticz WebUI > Setup > Hardware)

Enabled:

Name: Thermostat Dining

Type: homematicIP Radiator Thermostat (HmIP-eTRV)

Data Timeout: Disabled

Specifying a Data Timeout will restart the hardware device if no data is received for the specified time.  
Do not enable this option for devices that do not receive data!

homematicIP Radiator Thermostat (HMIP-eTRV) v1.2.0

- Set the setpoint (degrees C).
- Get the actual temperature (degrees C).
- Get the low battery state (true or false). Threshold is set in the HomeMatic WebUI
- Get the valve position (0 - 100%).
- Supported are the devices HmIP-eTRV-B, HmIP-eTRV-2

Domoticz Devices (Type,SubType)

- Setpoint (Thermostat,Setpoint)
- Temperature (Temp,LaCrosse TX3)
- Battery (General,Alert)
- Valve (General,Percentage)

Hardware Configuration

- Address (CCU IP address, default: 192.168.1.225)
- IDs (obtained via XML-API script <http://ccu-ip-address/addons/xmlapi/statelist.cgi>):
  - Device ID HmIP-eTRV-B or HmIP-eTRV-2 (default: 1541)
  - Datapoint IDs(#4): SET\_POINT\_TEMPERATURE, ACTUAL\_TEMPERATURE, LOW\_BAT, LEVEL as comma separated list in this order (defaults: 1584,1567,1549,1576)
- Note: After configuration update, the setpoint is 0. Click the setpoint to set the value.

CCU IP: 192.168.1.225

Device ID: 3637

Datapoint IDs: 3680,3663,3645,3672 |

Check Interval (sec): 60

Debug: True

## Devices, Roomplan & Dashboard for selected room

The screenshot displays three main sections of the Domoticz interface:

- Devices "Dining Room":** A table listing devices in the Dining Room, including Thermostat Dining - Valve, Thermostat Dining - Battery, Thermostat Dining - Temperature, and Thermostat Dining - Setpoint.
- Roomplan "Dining Room":** A list of room plans for the Dining Room, showing four entries related to the thermostat.
- Dashboard "Dining Room":** A control panel for the Dining Room. It shows the current temperature (21.9°C), battery level (22%), and status (Batteriestand Ok). It includes a "Change setpoint" button with up and down arrows, currently set at 22.0.

## Enhancement Push Off Button

### Purpose

To define a switch to handle the thermostat device state change action to Off.

### How

Define a VirtualSensor device with properties:

- Name, i.e., "MakeLab Thermostat - Off"
- Sensor Type Switch.

After adding the device, select the device widget and change the properties:

- Switch Type:  
Push Off Button
- Off Action:  
[http://ccu-ip/addons/xmlapi/statechange.cgi?ise\\_id=1584&new\\_value=0](http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id=1584&new_value=0)

The datapoint SET\_POINT\_TEMPERATURE of the device is required, i.e., 1584.

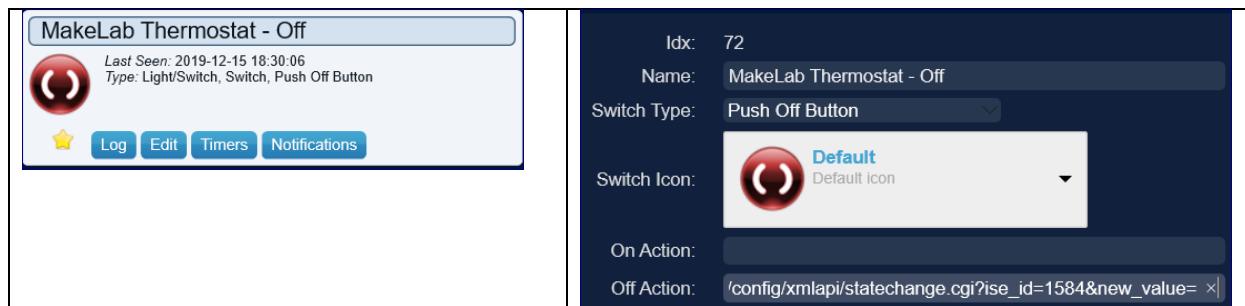
Obtain this datapoint id (ise\_id), by requesting the state of the thermostat device with

id=1541. The HTTP XML response holds various datapoints, select:

SET\_POINT\_TEMPERATURE.

```
http://ccu-ip/addons/xmlapi/state.cgi?device_id=1541
```

```
<datapoint ise_id="1584" name="HmIP-RF.000A18A9A64DAC:1.SET_POINT_TEMPERATURE" timestamp="1576829294" valueunit="°C" valuetype="4" value="20.00000" type="SET_POINT_TEMPERATURE"/>
```



### Domoticz Log

Performed test by pushing the icon on the widget, which send the HTTP XML-API request to the CCU to set the setpoint of the MakeLab thermostat to 0.

```
2019-12-15 18:30:06.767 (VirtualDevices) Light/Switch (MakeLab Thermostat - Off)
2019-12-15 18:30:06.758 Status: User: Admin initiated a switch command (72/MakeLab Thermostat - Off/Off)
```

## Timers

Each radiator thermostat has one or more timers to control the temperature.  
The timers are defined in the Domoticz Database table SetpointTimers.

Examples SQL SELECT statement to obtain the setpoint timers from the Domoticz Production database. The device names are in German.

```
SELECT d.Name, t.* FROM DeviceStatus d, SetpointTimers t WHERE d.ID=t.DeviceRowID;
```

### Output Tool SQLiteSpy

Name	ID	Active	DeviceRowID	Date	Time	Type	Temperature	TimerPlan	Days	Month	MDay	Occurence
HZG Bad Sollwert	1	1	207	2019-12-15	06:00	2	20.0	0	128	0	0	0
HZG Bad Sollwert	2	1	207	2019-12-15	11:00	2	19.0	0	128	0	0	0
HZG Bad Sollwert	3	1	207	2019-12-14	23:00	2	17.0	0	128	0	0	0
HZG EZ Sollwert	4	1	204	2019-12-14	06:00	2	20.0	0	128	0	0	0
HZG EZ Sollwert	5	1	204	2019-12-14	22:00	2	17.0	0	128	0	0	0
HZG WZ Sollwert	6	1	201	2019-12-15	06:15	2	21.0	0	128	0	0	0
HZG WZ Sollwert	7	1	201	2019-12-14	22:00	2	0.0	0	128	0	0	0
HZG Dusche Sollwert	8	1	198	2019-12-14	21:30	2	21.0	0	128	0	0	0
HZG Dusche Sollwert	9	1	198	2019-12-14	23:00	2	0.0	0	128	0	0	0
HZG MakeLab Sollwert	10	1	195	2019-12-14	19:00	2	0.0	0	128	0	0	0

### Examples SQLite Shell

The header (.header on) and mode CSV (.mode csv) are set.  
More information read chapter [SQLite Shell](#).

### Full Overview

```
sqlite> SELECT d.Name, t.* FROM DeviceStatus d, SetpointTimers t WHERE d.ID=t.DeviceRowID ORDER BY d.name;
```

```
Name,ID,Active,DeviceRowID,Date,Time,Type,Temperature,TimerPlan,Days,Month,MDay,Occurrence
"HZG Bad Sollwert",1,1,207,2019-12-17,06:00,2,21.0,0,128,0,0,0
"HZG Bad Sollwert",2,1,207,2019-12-15,11:00,2,19.0,0,128,0,0,0
"HZG Bad Sollwert",3,1,207,2019-12-17,19:00,2,21.0,0,128,0,0,0
"HZG Bad Sollwert",11,1,207,2019-12-17,23:00,2,17.0,0,128,0,0,0
"HZG Dusche Sollwert",8,1,198,2019-12-14,21:30,2,21.0,0,128,0,0,0
"HZG Dusche Sollwert",9,1,198,2019-12-14,23:00,2,0.0,0,128,0,0,0
"HZG EZ Sollwert",12,1,210,2019-12-21,22:00,2,17.0,0,128,0,0,0
...and more ...
```

### Brief Overview

```
sqlite> SELECT d.Name, t.Time, t.Temperature FROM DeviceStatus d, SetpointTimers t WHERE d.ID=t.DeviceRowID ORDER BY d.name,t.Time;
```

```
Name,Time,Temperature
"HZG Bad Sollwert",06:00,21.0
"HZG Bad Sollwert",11:00,19.0
... and more
```

## Example Device Timer Definition

Name: Hzg Bad Sollwert				
2019-12-15 19:27:34 ☀️ ▲08:31 ▼16:01				
Show	25	▼ entries	Search:	
Active	Type	Date	Time	Command
Yes	On Time		06:00	Temperature, 20
Yes	On Time		11:00	Temperature, 19
Yes	On Time		23:00	Temperature, 17

## Timerplans

The timerplan default (ID=0) is assigned.

In the Domoticz WebUI > Setup > Settings > Other: the timer plan can be set.



The selected timer plan ID, can be found in the Domoticz database, table Preferences , field Key with content ActiveTimerPlan.

## Solution Thermostat SetPoint Device Homematic Script (Explored)

This solution uses for each Homematic IP Thermostat device a dedicated Domoticz Thermostat SetPoint device.

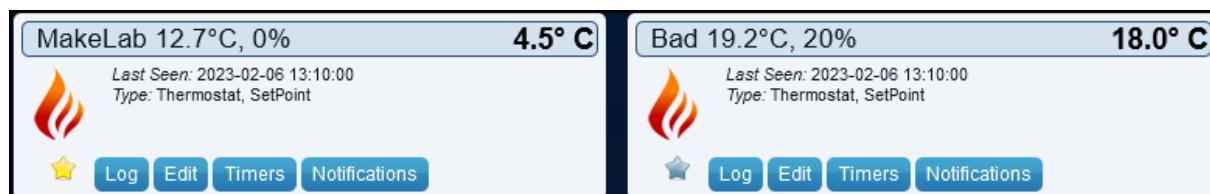
The Domoticz Thermostat SetPoint devices are updated in regular intervals with the device name (containing the Homematic device name, the actual temperature, and the level of the valve) and the device setpoint.

The setpoint of the thermostat setpoint can be changed as well.

For exploring this solution, two Domoticz thermostat devices created, shown here with actual data.

If more information required, like the operating voltage, then the solution can be enhanced easily.

### Domoticz Device Widgets



	<p>The device description contains the Homematic device channel used to change the setpoint via Remote Homematic Script API (see Domoticz Configuration below).</p> <p>HmIP-eTRV-B 002018A99D2097:1</p>
--	---

### Domoticz Dashboard (settings mobile mode)



## Communication

The communication between Domoticz and Homematic is handled with the Remote Homematic Script API triggered by a Domoticz Automation Event dzVents (PULL via HTTP POST request).

### Note

If the Domoticz thermostat setpoint devices are updated every 5 minutes and a thermostat setpoint is changed via the Homematic device at the radiator or via the Homematic WebUI, it will take up-to 5 minutes to see the new setpoint on the Domoticz devices.

Check out the Homematic Duty-Cycle when setting a high interval, like 1 minute. If Duty-Cycle gets too high, increase the interval.

If the Domoticz thermostat setpoint is changed via its widget or dashboard entry, the new setpoint is seen immediately.

## Homematic Configuration

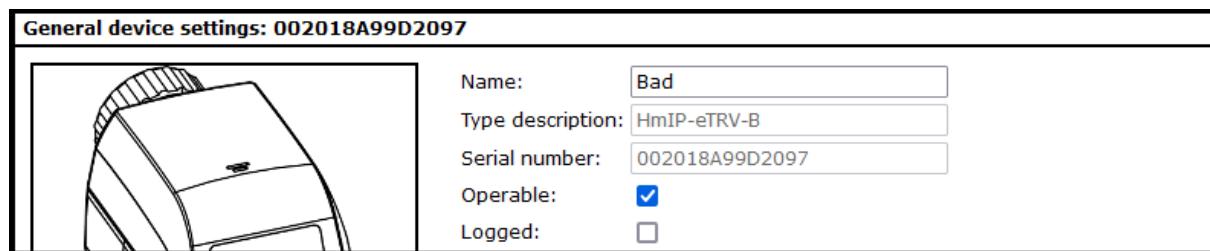
There is no additional Homematic configuration required, besides having the thermostats in use.

### Important

The device name is used by the Domoticz Automation Event dzVents to get the thermostat data (see below Domoticz Automation Event).

The type description with serial number and channel is used to set the setpoint triggered by Domoticz thermostat device.

### Example Thermostat named Bad



### Homematic Script

The Domoticz Automation Event dzVents uses a more advanced Homematic script to get the Homematic device attributes.

The script is developed and tested first using the Homematic CCU Addon Script-Parser (Homematic Script Executor)..

Ensure to use for every script change prior integrating in the Domoticz Automation Event dzVents script.

**HomeMatic Script Executor v1.9**

Achtung! Geben Sie das Webinterface Ihrer CCU / RaspberryMatic etc. **NIEMLS** über eine Portfreigabe im Router für den externen Zugriff frei! Ihre Hausautomatisierung wird damit trotz gesetztem WebUI-Passwort von außen angreif- und verwundbar! Entfernen Sie die Portfreigabe und nutzen Sie Techniken wie VPN oder ReverseProxy um einen sicheren Fernzugriff auf Ihre CCU zu erhalten.

Eingabe:

CCU-Inventar (Geräte, Kanäle, Datapunkte) zeigen

Variablen zeigen

Programme zeigen

gecachte Seiten zeigen

Um detaillierte Fehlermeldungen zu sehen: ggf. SSH in CCU aktivieren, putty-Session zur CCU starten (putty), einloggen, Befehl 'tail -f /var/log/messages | grep "Error.\*near"' (ohne führendes und endendes ' eingeben!) eingeben.

```

Um Thermostats - Define the thermostats as CSV list
var thermostats = "Bad,Dusche,Esszimmer,Bad,Esszimmer,Wohnzimmer-1,Wohnzimmer-2";
var thermostatid = "12,0,0,0,9,0,0";
Integer thermostatcount = 7;
Integer thermostatindex = 0;
String thermostatlist = "Bad,Dusche,Esszimmer,Bad,Esszimmer,Wohnzimmer-1,Wohnzimmer-2";
String thermostatlistsplit = thermostatlist.Split(",");
String result = "[";
for(thermostatindex=0;thermostatindex<thermostatcount;thermostatindex++)
{
    result += "[";
    var thermostat = thermostatlistsplit[thermostatindex];
    var attributes = thermostat.Split(",");
    var realtemperature = attributes[0];
    var realsetpoint = attributes[1];
    var reallevel = attributes[2];
    result += "real temperature: " + realtemperature + ",";
    result += "real setpoint: " + realsetpoint + ",";
    result += "real level: " + reallevel + ",";
    result += "],";
}
result = result.Substring(0,result.Length-1);
result += "]";

```

Ausgabe:

```
[{"n": "Bad", "l": "12", "c": "HmIP-eTRV-B 002018A99D2097:1", "t": "19-1,s": "18-0,t": "28}, {"n": "Dusche", "l": "0", "c": "HmIP-eTRV-B 00201A49952C08:1", "t": "12-1,s": "5-0,t": "0}, {"n": "Esszimmer", "l": "0", "c": "HmIP-eTRV-2 000A1A49A0D878:1", "t": "29-7,s": "0} ]
```

**Ausführen**

The script is embedded in the Domoticz Automation Event dzVents script as a long string (via cut and paste).

Do not forget to comment out all WriteLine() statements.

```
-- Homematic script
local script = [[!
  INSERT YOUR SCRIPT HERE
]]
```

If the script has placeholder, then change like:

```
script = string.gsub(script, "#THERMOSTATS#", thermostats)
```

## Domoticz Configuration

### Devices

For each of the Homematic IP thermostats, a Domoticz Virtual Sensor (Hardware Type Dummy) Type Thermostat, SubType SetPoint is defined.

For exploring, two thermostats are defined.

Idx	Name	Enabled	Type	Address	Port	Data Timeout
3	VirtualSensors	Yes	Dummy (Does nothing, use for virtual switches only) <a href="#">Create Virtual Sensors</a>			Disabled

Idx	Hardware	ID	Unit	Name	Type	SubType	Data	Last Seen
9	VirtualSensors	0014059	1	Makelab 12.7°C, 0%	Thermostat	SetPoint	4.5	2023-02-06 13:13:00
12	VirtualSensors	001405C	1	Bad 19.2°C, 20%	Thermostat	SetPoint	18.0	2023-02-06 13:13:00

## Automation Event Update Devices

### Concept

The Domoticz Automation Event dzVents defines the thermostat as a CSV string with thermostat names as defined in Homematic WebUI.

Important: When changing the name in Homematic WebUI, set the new name in the list below.

```
-- Thermostat csv string with the names as defined in the Homematic WebUI
local thermostats = "Bad,Dusche,Esszimmer,Flur,MakeLab,Wohnzimmer-1,Wohnzimmer-2"
-- Thermostat csv string with the thermostat device idx defined in Domoticz
local thermostatidxs = "12,0,0,0,9,0,0"
-- Number of thermostats = MUST match the previous csv string items
local thermostatcnt = 7
```

The event triggers, in regular intervals, an HTTP POST request (to the CCU) containing a (more complex) Remote Homematic Script.

The Homematic embedded script loops over the thermostats list to get the attributes Name, Actual Temperature, Setpoint and Level.

The result is a JSON array with thermostat elements as key:value pairs.

The keys are:

- n = Thermostat name,
- i = Domoticz thermostat IDX,
- t = Actual temperature °C,
- s = Current setpoint °C,
- l = Level valve opening 0-100%,
- c = Device channel used to change the setpoint.

### Example for two thermostats

```
[{"n":"Bad","i":"12","c":"HmIP-eTRV-B 002018A99D2097:1","t":19.1,"s":18.0,"l":20}, {"n":"Dusche","i":"0","c":"HmIP-eTRV-B 00201A49952CB8:1","t":12.1,"s":5.0,"l":0}, and more]
```

The HTTP response is an XML object containing various XML tags.

The XML tag <result> contains a JSON string with data for each of the defined thermostats.

The JSON string is converted to a Lua table.

The Domoticz thermostats are updated by looping over the Lua table and assign the key:value pairs accordingly to the Thermostat device.

The Domoticz device description is checked if empty. If so, the Homematic device channel is inserted.

The channel is required by the event to handle setpoint changes from the Domoticz Thermostat Setpoint device.

## Script

```
-- homematic_thermostat_update.dzvents
-- Get, in regular intervals, the temperature (°C), setpoint (°C) and level (0-100%) of a TRV device
via Remote Homematic Script API.
-- In addition the thermostat name and channel are requested to check which thermostat to change (name)
and which thermostat setpoint is changed (channel, used by other event).
-- The thermostat channel name for chaning the setpoint is stored in the Domoticz thermostat
description.
-- The object channel is obtained from the Homematic WebUI Devices list for the selected thermostat.
-- Datapoints ACTUAL_TEMPERATURE, SET_POINT_TEMPERATURE, LEVEL

-- Define the Remote Homematic Script URL.
local URL_HMAPI = 'http://192.168.1.70:8181/tclregga.exe'
-- HTTP Response - MUST be UNIQUE across all events.
local RES_HMAPI = "THERMOSTAT-UPDATE"

-- Thermostat csv string with the names as defined in the Homematic WebUI
local thermostats = "Bad,Dusche,Esszimmer,Flur,MakeLab,Wohnzimmer-1,Wohnzimmer-2"
-- Thermostat csv string with the thermostat device idx defined in Domoticz
local thermostatidxs = "12,0,0,0,9,0,0"
-- Number of thermostats = MUST match the previous csv string items
local thermostatcnt = 7

-- Timer rule
local TIMER_RULE = "every minute"
-- local TIMER_RULE = "every 5 minutes"

-- Homematic script to get all thermostat selective attributes
-- Create a JSON object containing for selective thermostats the name, temperature, setpoint and
level.
-- Result is a JSON array, like:
-- [{"n": "Bad", "i": "12", "c": "HmIP-eTRV-B
002018A99D2097:1", "t": "19.1", "s": "18.0", "l": "20"}, {"n": "Dusche", "i": "0", "c": "HmIP-eTRV-B
00201A49952CB8:1", "t": "12.1", "s": "5.0", "l": "0}], and more]
local script = [!

! Thermostats - Define the thermostats as csv list
var thermostats = "#THERMOSTATS#";
! var thermostats = "Bad,Dusche,Esszimmer,Flur,MakeLab,Wohnzimmer-1,Wohnzimmer-2";
integer thermostatcnt = "#THERMOSTATCNT#";
!integer thermostatcnt = 7;
var thermostatidxs = "#THERMOSTATIDXS#";

! Split the thermostats to a list
var thermostatlist = thermostats.Split(",");
! Define the result JSON array [{},{}]
string result = "[";
! Thermostat attributes
real temperature;
real setpoint;
real level;
! Helpers
string thermostat;
var obj;
var objchannel;
integer devcount = 0;
integer index;
! Get all devices
var devs = dom.GetObject(ID_DEVICES).EnumUsedIDs();
string devid;
string device;
! Loop over all devices
foreach(devid, devs){
    obj = dom.GetObject(devid);
    ! WriteLine(obj.Name() # "=" # obj.Address() # "=" # obj.HssType());
    ! Loop over the defined thermostats list to get the data
    ! The index is used to get the thermostat idx from the csv string thermostatidxs
    index = 0;
    foreach(thermostat, thermostatlist){
        ! Check if the thermostat is found
        if (thermostat == obj.Name()) {
            devcount = devcount + 1;
            ! WriteLine(obj.Name() # "=" # obj.Address() # "=" # obj.HssType());
            ! Build the JSON object for the device
            device = '{';

```

```

! Set the name
device = device # "'n':'" # obj.Name() # '",';
! Set the idx
device = device # "'i':'" # thermostatidxs.StrValueByIndex(", index) # '",';
! Get the channel :1 for the thermostat attributes
objchannel = obj.HssType() # " " # obj.Address() # ":1";
! WriteLine(objchannel);
! Get the channel object - resuses the var obj
obj = dom.GetObject(objchannel);
! Set the objchannel
device = device # "'c':'" # objchannel # '",';
! Set the attributes - using ToString(decimals)
temperature = obj.DPByHssDP("ACTUAL_TEMPERATURE").Value().ToString(1);
device = device # "'t':'" # temperature # ',';
setpoint = obj.DPByHssDP("SET_POINT_TEMPERATURE").Value().ToString(1);
device = device # "'s':'" # setpoint # ',';
level = obj.DPByHssDP("LEVEL").Value() * 100;
device = device # "'l':'" # level.ToString(0);
device = device # '}';
! Add comma if device count less number thermostats
if (devcount < thermostatcnt) {
    device = device # ',';
}
result = result # device;
}
index = index + 1;
}
result = result # "]";
! WriteLine(result);
[]]

-- Update the Domoticz thermostat device name and setpoint (do silent to avoid endless loop)
local function updateThermostat(domoticz, thermostat)
    local name = string.format('%s %.1f°C, %.0f%%', thermostat['n'], thermostat['t'], thermostat['l'])
    local idx = tonumber(thermostat['i'])
    if (idx > 0) then
        domoticz.devices(idx).rename(name)
        domoticz.devices(idx).updateSetPoint(thermostat['s']).silent() -- .afterSec(2)
        -- Check if the device descriptions needs set
        local descr = domoticz.devices(idx).description
        if (descr == "") then
            descr = thermostat['c']
            domoticz.devices(idx).setDescription(descr)
            domoticz.log(string.format('Set description: %s %s', thermostat['n'], descr))
        end
    end
end

return {
    on = { timer = { TIMER_RULE }, httpResponses = { RES_HMAPI } },
    logging = { level = domoticz.LOG_INFO, marker = RES_HMAPI, },
    execute = function(domoticz, item)
        if (item.isTimer) then
            local url = URL_HMAPI
            script = string.gsub(script, "#THERMOSTATS#", thermostats)
            script = string.gsub(script, "#THERMOSTATIDXS#", thermostatidxs)
            script = string.gsub(script, "#THERMOSTATCNT#", thermostatcnt)
            domoticz.openURL({
                url = url,
                headers = { ['content-type'] = 'application/x-www-form-urlencoded' },
                postData = script, method = 'POST', callback = RES_HMAPI,
            })
        end
        if (item.isHTTPResponse) then
            if (item.ok) then
                -- The item data contains the XML response from the CCU
                -- Get the various XML tag result, replace &quot; by " to get JSON string
                local data = domoticz_applyXPath(item.data,'/xml/result/text()')
                data = string.gsub(data, "&quot;", "'")
                domoticz.log(data)
                -- Convert JSON string (array) to Lua table
                local datatable = domoticz.utils.fromJSON(data)
                -- domoticz.log(datatable)
            end
        end
    end
}

```

```

-- Loop over the table containing the key as thermostat
-- {[ "i" ]= "9", [ "s" ]= 4.5, [ "t" ]= 14.6, [ "n" ]= "MakeLab", [ "l" ]= 0}
for key,value in pairs(datatable) do
    domoticz.log(datatable[key])
    updateThermostat(domoticz, datatable[key])
end
else
    domoticz.log('There was a problem handling the request', domoticz.LOG_ERROR)
    domoticz.log(item, domoticz.LOG_ERROR)
end
end
end
}

```

## Domoticz Log

```

2023-02-06 15:59:00.248 Status: dzVents: Info: THERMOSTAT-UPDATE: ----- Start internal script:
homematic_thermostat_update:, trigger: "every minute"
2023-02-06 15:59:00.249 Status: dzVents: Info: THERMOSTAT-UPDATE: ----- Finished
homematic_thermostat_update
2023-02-06 15:59:00.249 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
2023-02-06 15:59:00.422 Status: dzVents: Info: Handling httpResponse-events for: "THERMOSTAT-UPDATE"
2023-02-06 15:59:00.422 Status: dzVents: Info: THERMOSTAT-UPDATE: ----- Start internal script:
homematic_thermostat_update: HTTPResponse: "THERMOSTAT-UPDATE"
2023-02-06 15:59:00.431 Status: dzVents: Info: THERMOSTAT-UPDATE: {[ "l" ]= 20, [ "n" ]= "Bad", [ "c" ]= "HmIP-eTRV-B 002018A99D2097:1", [ "t" ]= 19.1, [ "i" ]= 12, [ "s" ]= 18.0}
2023-02-06 15:59:00.448 Status: dzVents: Info: THERMOSTAT-UPDATE: {[ "l" ]= 0, [ "n" ]= "Dusche", [ "c" ]= "HmIP-eTRV-B 00201A49952CB8:1", [ "t" ]= 12.1, [ "i" ]= 0, [ "s" ]= 5.0}
2023-02-06 15:59:00.448 Status: dzVents: Info: THERMOSTAT-UPDATE: {[ "l" ]= 78, [ "n" ]= "Esszimmer", [ "c" ]= "HmIP-eTRV-2 000A1A49A0D878:1", [ "t" ]= 20.7, [ "i" ]= 0, [ "s" ]= 20.0}
2023-02-06 15:59:00.448 Status: dzVents: Info: THERMOSTAT-UPDATE: {[ "l" ]= 25, [ "n" ]= "Flur", [ "c" ]= "HmIP-eTRV-2 000A1A49A0D8A5:1", [ "t" ]= 18.3, [ "i" ]= 0, [ "s" ]= 18.0}
2023-02-06 15:59:00.448 Status: dzVents: Info: THERMOSTAT-UPDATE: {[ "l" ]= 0, [ "n" ]= "MakeLab", [ "c" ]= "HmIP-eTRV-2 000A18A9A64DAC:1", [ "t" ]= 14.5, [ "i" ]= 9, [ "s" ]= 4.5}
2023-02-06 15:59:00.449 Status: dzVents: Info: THERMOSTAT-UPDATE: Set description: MakeLab HmIP-eTRV-2 000A18A9A64DAC:1
2023-02-06 15:59:00.449 Status: dzVents: Info: THERMOSTAT-UPDATE: {[ "l" ]= 100, [ "n" ]= "Wohnzimmer-1", [ "c" ]= "HmIP-eTRV-B 00201A49952CB1:1", [ "t" ]= 18.8, [ "i" ]= 0, [ "s" ]= 21.0}
2023-02-06 15:59:00.449 Status: dzVents: Info: THERMOSTAT-UPDATE: {[ "l" ]= 100, [ "n" ]= "Wohnzimmer-2", [ "c" ]= "HmIP-eTRV-B 00201A499D76F8:1", [ "t" ]= 18.8, [ "i" ]= 0, [ "s" ]= 21.0}
2023-02-06 15:59:00.449 Status: dzVents: Info: THERMOSTAT-UPDATE: ----- Finished
homematic_thermostat_update

```

## Automation Event Setpoint Change

```
-- homematic_thermostat_set.dzevents
-- Change the setpoint of a TRV-2 device via Remote Homematic Script API.
-- Bbject channel obtained from Domoticz device description (see event homematic_thermostat_update).
-- The datapoint is SET_POINT_TEMPERATURE.

-- Define the IDX of the thermostat devices
local IDX_THERMOSTAT_BAD = 12
local IDX_THERMOSTAT_MAKELAB = 9
local THERMOSTAT_DEVICES = { IDX_THERMOSTAT_BAD, IDX_THERMOSTAT_MAKELAB}
-- Define the Remote Homematic Script URL.
local URL_HMAPI = 'http://192.168.1.70:8181/tclreg.exe'
-- HTTP Response - MUST be UNIQUE across all events
local RES_HMAPI = "THERMOSTAT-SET"

-- Define the Homematic script as long string. Do not encode, i.e. leave spaces etc.
local function setscript(domoticz, channel, setpoint)
    local script = [[! Homematic script
        var result = dom.GetObject("#CHANNEL#").DPByHssDP("SET_POINT_TEMPERATURE").State(#SP#);
    ]]
    script = string.gsub(script, "#CHANNEL#", channel)
    script = string.gsub(script, "#SP#", setpoint)
    domoticz.log(script)
    return script
end

return {
    on = { devices = THERMOSTAT_DEVICES , httpResponses = { RES_HMAPI } },
    logging = { level = domoticz.LOG_INFO, marker = RES_HMAPI, },
    execute = function(domoticz, item)
        -- The device setpoint has changed. Sent HTTP request to CCU.
        if (item.isDevice) then
            -- Submit the HTTP POST request = Remote Homematic Script
            domoticz.openURL({
                url = URL_HMAPI,
                headers = { ['content-type'] = 'application/x-www-form-urlencoded' },
                method = 'POST',
                postData = setscript(domoticz, item.description, item.setPoint),
                callback = RES_HMAPI,
            })
        end
        if (item.isHTTPResponse) then
            if (item.ok) then
                -- Get the setpoint tag
                -- <xml><exec>/tclreg.exe</exec><sessionId></sessionId><httpUserAgent>User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/603.37 (KHTML, like Gecko) Chrome/54.0.30045.0 Safari/603.38</httpUserAgent><result>true</result></xml>
                -- Get the text of the result tag: true=OK, null=ERROR
                local result = domoticz_applyXPath(item.data,'/xml/result/text()')
                domoticz.log(string.format('Setpoint changed. Result: %s', result))
            else
                domoticz.log('There was a problem handling the request', domoticz.LOG_ERROR)
                domoticz.log(item, domoticz.LOG_ERROR)
            end
        end
    end
}
```

## Domoticz Log

Change the Thermostat MakeLab Setpoint to 21 °C.

```
2023-02-06 16:43:03.688 VirtualSensors: Thermostat (MakeLab 14.7°C, 0%)
2023-02-06 16:43:03.686 Status: User: admin initiated a SetPoint command
2023-02-06 16:43:03.743 Status: dzVents: Info: Handling events for: "MakeLab 14.7°C, 0%", value: "21.00"
2023-02-06 16:43:03.744 Status: dzVents: Info: THERMOSTAT-SET: ----- Start internal script: homematic_thermostat_set: Device: "MakeLab 14.7°C, 0% (VirtualSensors)", Index: 9
2023-02-06 16:43:03.744 Status: dzVents: Info: THERMOSTAT-SET: http://192.168.1.70:8181/tclrega.exe
2023-02-06 16:43:03.744 Status: dzVents: Info: THERMOSTAT-SET: ! Homematic script
2023-02-06 16:43:03.744 ! Set the setpoint of a thermostat device
2023-02-06 16:43:03.744 var result = dom.GetObject("HmIP-eTRV-2
000A18A9A64DAC:1").DPByHssDP("SET_POINT_TEMPERATURE").State(21.0);
2023-02-06 16:43:03.744
2023-02-06 16:43:03.744 Status: dzVents: Info: THERMOSTAT-SET: ----- Finished homematic_thermostat_set
2023-02-06 16:43:03.744 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
2023-02-06 16:43:05.303 Status: dzVents: Info: Handling httpResponse-events for: "THERMOSTAT-SET"
2023-02-06 16:43:05.303 Status: dzVents: Info: THERMOSTAT-SET: ----- Start internal script: homematic_thermostat_set: HTTPResponse: "THERMOSTAT-SET"
2023-02-06 16:43:05.306 Status: dzVents: Info: THERMOSTAT-SET: Setpoint changed. Result: true
2023-02-06 16:43:05.306 Status: dzVents: Info: THERMOSTAT-SET: ----- Finished homematic_thermostat_set
```

## Thermostat State Custom Page

### Purpose

To display the list of thermostats with selective datapoints in an HTML table embedded in a Domoticz Custom Page.

Homematic Thermostat State						Refresh
Name	Setpoint	Temperature	Difference	Level	Voltage	
Bad	19.0	19.9	0.9	24	2.4	
Dusche	6.0	14.9	8.9	0	2.6	
Esszimmer	20.0	21.5	1.5	13	3.0	
Flur	20.0	21.0	1.0	28	2.3	
MakeLab	21.0	20.1	-0.9	90	2.7	
Wohnzimmer-1	23.0	19.1	-3.9	100	2.5	
Wohnzimmer-2	23.0	20.1	-2.9	100	2.5	

20220315 by rwbl

The custom page is accessed via the Domoticz GUI > Tab Custom > Thermostat\_State. It is also possible to access the custom page direct in a browser using the URL:

[http://domoticz-ip:8080/#/Custom/Termostat\\_State](http://domoticz-ip:8080/#/Custom/Termostat_State)

Refresh reloads the table with latest data.

The solution is a good example of performing an asynchronous HTTP (Ajax) API request to the CCU, handled by the XML-API addon, and parsing the HTTP XML response.

## Concept

The custom page gets the homematic statelist via HTTP XML-API request

```
http://ccu-ip/addons/xmlapi/statelist.cgi
```

The HTTP response is a devices statelist in XML format.

```
<stateList>
  <device name="Alert Indicator" ise_id="3884" unreach="false" config_pending="false">
    ...
  </device>
  <device name="Bad" ise_id="1786" unreach="false" config_pending="false">
    <channel name="Bad:0" ise_id="1787" index="0" visible="true" operate="true">
      <datapoint name="HmIP-RF.002018A99D2097:0.CONFIG_PENDING" type="CONFIG_PENDING" ise_id="1788" value="0"/>
      <datapoint name="HmIP-RF.002018A99D2097:0.DUTY_CYCLE" type="DUTY_CYCLE" ise_id="1792" value="0"/>
      <datapoint name="HmIP-RF.002018A99D2097:0.LOW_BAT" type="LOW_BAT" ise_id="1794" value="0"/>
      <datapoint name="HmIP-RF.002018A99D2097:0.OPERATING_VOLTAGE" type="OPERATING_VOLTAGE" ise_id="1795" value="0"/>
      <datapoint name="HmIP-RF.002018A99D2097:0.OPERATING_VOLTAGE_STATUS" type="OPERATING_VOLTAGE_STATUS" ise_id="1796" value="0"/>
      <datapoint name="HmIP-RF.002018A99D2097:0.RSSI_DEVICE" type="RSSI_DEVICE" ise_id="1800" value="0"/>
      <datapoint name="HmIP-RF.002018A99D2097:0.RSSI_PEER" type="RSSI_PEER" ise_id="1801" value="0"/>
      <datapoint name="HmIP-RF.002018A99D2097:0.UNREACH" type="UNREACH" ise_id="1802" value="0"/>
    </channel>
    <channel name="HmIP-eTRV-B 002018A99D2097:1" ise_id="1806" index="1" visible="true" operate="true">
      <datapoint name="HmIP-RF.002018A99D2097:1.ACTIVE_PROFILE" type="ACTIVE_PROFILE" ise_id="1807" value="0"/>
      <datapoint name="HmIP-RF.002018A99D2097:1.ACTUAL_TEMPERATURE" type="ACTUAL_TEMPERATURE" ise_id="1808" value="21"/>
      <datapoint name="HmIP-RF.002018A99D2097:1.ACTUAL_TEMPERATURE_STATUS" type="ACTUAL_TEMPERATURE_STATUS" ise_id="1809" value="0"/>
    </channel>
  </device>
```

*Extract of the HTTP XML response with the root node stateList and the device nodes with channels and datapoints.*

The XML tree is parsed to get selective datapoints for the thermostat devices:  
NAME, ACTUAL\_TEMPERATURE, SET\_POINT\_TEMPERATURE , LEVEL,  
OPERATING\_VOLTAGE

A HTML table is used to show the thermostat selective datapoints.  
Each table row contains the attribute value of the datapoints.

## Custom Page

The custom page *Thermostat\_State.html* is in the folder:

```
~domoticz/www/templates
```

Because this is a Domoticz Custom Page there are not HTML and BODY tags.

```
<style>
/* hint style text small */
.hint {
  font-size: smaller;
  color: white;
}
</style>

<!-- HTML -->
<div class="container">
  <p id="demo"></p>
  <table class="page-header-small" border="0" cellpadding="0" cellspacing="0" width="100%">
    <tr>
      <td align="left">
        <h4 data-i18n="Homematic Thermostat State"></h4>
      </td>
      <td align="right">
```

```

        <a class="btnstyle3" onclick="refreshList();"><span data-i18n="Refresh"></span></a>
    </td>
</tr>
</table>
<table id="thermostats" class="display" border="1" cellpadding="1" cellspacing="0"
style="width:600px;align=left">
</table>
<div id="pageversion" class="hint"></div>
</div>

<!-- JavaScript -->
<script type="text/javascript" charset="utf-8">

// Get the statelist and parse the XML tree to build the table with Thermostat devices
// Parameter:
// None
function getStateList() {
    var requestUrl = "http://ccu-ip/addons/xmlapi/statelist.cgi";
    $.ajax({
        url: requestUrl,
        async: false,
        dataType: 'xml',
        success: function (data) {
            // Extract relevant data from XML
            var thermostattable = "";
            thermostattable += "<tr>" +
                "<th style='width:110px;'>" + "Name" + "</th>" +
                "<th style='width:80px;'>" + "Setpoint" + "</th>" +
                "<th style='width:80px;'>" + "Temperature" + "</th>" +
                "<th style='width:80px;'>" + "Difference" + "</th>" +
                "<th style='width:80px;'>" + "Level" + "</th>" +
                "<th style='width:80px;'>" + "Voltage" + "</th>" +
                "</tr>";
            var $devices = $('#stateList',data);
            $devices.find("device").each(function(index, elem){
                // <device name="Alert Indicator" ise_id="3884" unreach="false" config_pending="false">
                // <channel name="Alert Indicator:0" ise_id="3885" index="0" visible="true" operate="false">
                //   <datapoint name="HmIP-RF.000D1BE9A4E733:0.ACTUAL_TEMPERATURE" type="ACTUAL_TEMPERATURE"
ise_id="3886" value="0.000000" valuetype="4" valueunit="°C" timestamp="0" operations="5"/>
                // ...
                // Device attributes from xml:
                // <device name="Alert Indicator" ise_id="3884" unreach="false" config_pending="false">
                var $device = $(this);
                var device_name = $device.attr("name");
                var device_iseid = $device.attr("ise_id");

                // List of device channels
                // <channel name="Alert Indicator:0" ise_id="3885" index="0" visible="true" operate="false">
                var $channels = $("channel", elem);

                // List of device channel datapoints
                // <datapoint name="HmIP-RF.000D1BE9A4E733:0.ACTUAL_TEMPERATURE" type="ACTUAL_TEMPERATURE"
ise_id="3886" value="0.000000" valuetype="4" valueunit="°C" timestamp="0" operations="5"/>
                var temperature = -1;
                var setpoint = -1;
                var level = -1;
                var voltage = -1;
                // Loop over each device channel, i.e., there might be one or more channels
                $channels.each(function(){
                    var $datapoints = $(this).find("datapoint");
                    // Get the datapoints
                    if ($datapoints.length > 0){
                        // Loop over each datapoint of the channel and get selective attributes
                        $datapoints.each(function(){
                            var $datapoint = $(this);
                            var dtype = $datapoint.attr("type");
                            switch (dtype) {
                                case "ACTUAL_TEMPERATURE":
                                    temperature = Number($datapoint.attr("value"));
                                    break;
                                case "SET_POINT_TEMPERATURE":
                                    setpoint = Number($datapoint.attr("value"));
                                    break;
                                case "LEVEL":

```

```

        level = Number($datapoint.attr("value")) * 100;
        break;
    case "OPERATING_VOLTAGE":
        voltage = Number($datapoint.attr("value"));
        break;
    default:
        // console.log("");
    }
});
if (temperature > -1 && setpoint > -1 && level > -1 && voltage > -1){
    thermostattable += "<tr>" +
        "<td>" + device_name + "</td>" +
        "<td align=center>" + setpoint.toFixed(1) + "</td>" +
        "<td align=center>" + temperature.toFixed(1) + "</td>" +
        "<td align=center>" + (temperature - setpoint).toFixed(1) + "</td>" +
        "<td align=center>" + level.toFixed(0) + "</td>" +
        "<td align=center>" + voltage.toFixed(1) + "</td>" +
        "</tr>";
    }
});
// Set the selector with the table data
$("#thermostats").html(thermostattable);
},
error: function () {
    console.log("[ERROR] getStateList. Can not communicate to the Domoticz server!");
    return false;
}
});
};

// Refresh the page. Assigned to a button.
function refreshList() {
    window.location.reload();
    console.log("List refreshed");
}

// Set the page version below the thermostat list
function setVersion(pageVersion) {
    $('#pageversion').html(pageVersion);
};

// Load the document
$(document).ready(function(){
    setVersion("<br>" + "20220315 by rwbl");
    getStateList();
});
</script>
```

## Enhancements

Some enhancement ideas:

- If the difference between Setpoint and Actual Temperature is negative, show the value in red.
- Instead, battery voltage, show the low battery state (attribute LOW\_BAT).

# Remote Control (HmIP-RC8)

## Purpose

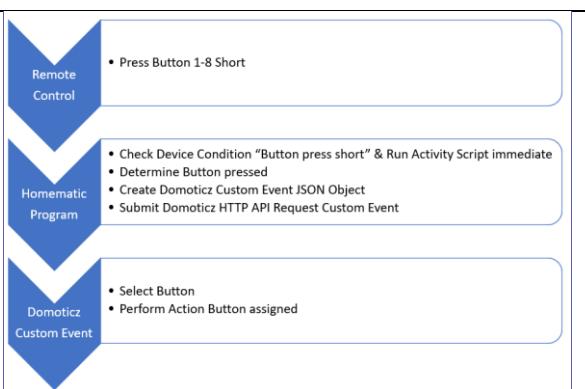
To remote control selected functions via a handheld Remote-Control device.

## Solution

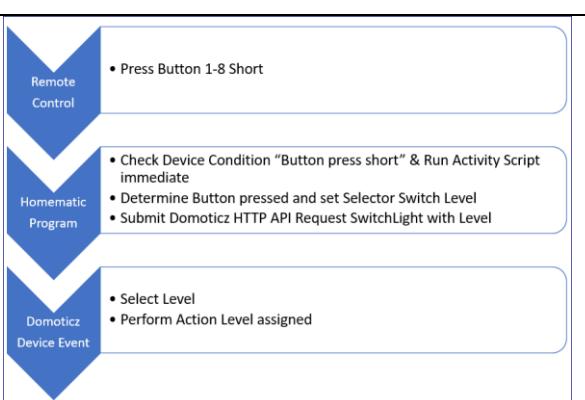
This solution makes use of the Homematic IP Remote Control - 8 buttons (HmIP-RC8).

	<p>The HmIP-RC8 device is integrated in the CCU. Two solutions worked out on using the Remote-Control.</p>
---	--

### Solution Custom Event – In Use

<p>The Remote-Control device submits, via a Homematic program, a HTTP API request to Domoticz with Custom Event data holding the button number (and channel of the button - not used).</p> <p>An Automation Event dzVents with trigger customEvents acts according to button number pressed.</p> <p>There are no Domoticz devices required.</p>	 <pre> graph TD     RC[Remote Control] --&gt; HP[Homematic Program]     HP --&gt; DC[Domoticz Custom Event]     </pre> <ul style="list-style-type: none"> <li>• Press Button 1-8 Short</li> <li>• Check Device Condition "Button press short" &amp; Run Activity Script immediate</li> <li>• Determine Button pressed</li> <li>• Create Domoticz Custom Event JSON Object</li> <li>• Submit Domoticz HTTP API Request Custom Event</li> </ul> <ul style="list-style-type: none"> <li>• Select Button</li> <li>• Perform Action Button assigned</li> </ul>
---	--

### Solution Selector Switch

<p>The Remote-Control device submits, via a Homematic program, a HTTP API request to Domoticz with Selector Switch Level depending on button number.</p> <p>An Automation Event dzVents with trigger device, acts according to level selected.</p> <p>A Domoticz Selector Switch device is required Hardware Dummy, Virtual Sensor.</p>	 <pre> graph TD     RC[Remote Control] --&gt; HP[Homematic Program]     HP --&gt; DE[Domoticz Device Event]     </pre> <ul style="list-style-type: none"> <li>• Press Button 1-8 Short</li> <li>• Check Device Condition "Button press short" &amp; Run Activity Script immediate</li> <li>• Determine Button pressed and set Selector Switch Level</li> <li>• Submit Domoticz HTTP API Request SwitchLight with Level</li> </ul> <ul style="list-style-type: none"> <li>• Select Level</li> <li>• Perform Action Level assigned</li> </ul>
---	---

The first option is rather straight forward, and the button actions can only be used via the Remote-Control device.

The second option enables to use the button actions also from the Domoticz WebUI (i.e., widget or dashboard) or integrate in Custom Web Pages or UI.

# Homematic Configuration

## Teach-In

The Remote-Control device is “teach-in” via the Homematic WebUI - standard approach to add devices.

The screenshot shows two pages of the Homematic WebUI. The top page is titled "Device inbox" and has a "Teach-in devices" button highlighted with a red circle. A red bracket on the right side points from this button to a larger callout box. The callout box contains text about teaching in via Internet connection and a "Done" button, also highlighted with a red circle. A red bracket on the left side points to the "Device inbox" table, which lists a device named "HmIP-RC 8". The bottom page is also titled "Device inbox" and shows the device has been renamed to "Remote Control". A red bracket on the left side points to this renamed entry. A red bracket on the right side points to a table showing the device's configuration, specifically the "Push button" section which lists "8 Push buttons".

This screenshot shows the "General device settings" page for the device with serial number 000B1BE98D94DE. On the left is a photograph of the "HmIP-RC8" remote control. The main form contains the following fields:

- Name: RC-Wohnzimmer
- Type description: HmIP-RC8
- Serial number: 000B1BE98D94DE
- Operable:
- Logged:

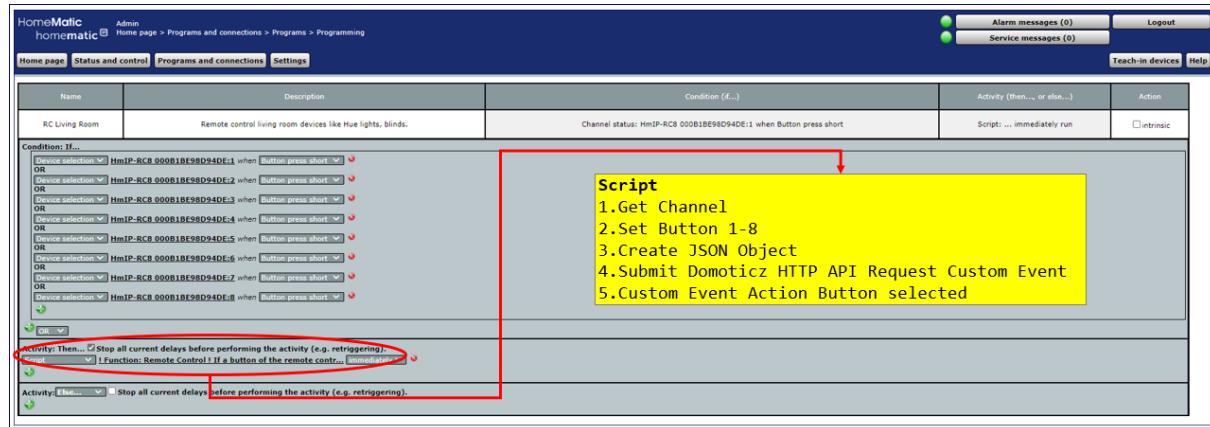
**Functional test**

**Start test**      --:--:--

During the functional test the error-free communication to the device is checked. Therefore, switching commands will be send to all actuators connected to the device. Sensors (e.g. remote controls) are usually sending signals only if they are operated manually. The test is passed as soon as the first feedback will be received by the device.

## Homematic Program

The screenshot shows the Homematic program with the condition based on the Remote-Control device button short pressed. Immediate action is taken by executing the script. The script assigns the datapoint id of the button short to the button. A JSON object is created with id and buttonnr, used for the data parameter of the HTTP API custom event request to Domoticz.



## Solution Custom Event (In Use)

### Homematic Script

```
! Function: Remote Control
! remote_control_custom_event.script.
! If buttons 1-8 of the remote control HmIP-RC8 is pressed, send custom event with buttonnr to Domoticz
! to action accordingly.
! Via CUxD an http api request for a custom event is submitted to domoticz.
! The Domoticz custom event data parameter must be in json format.
! Example: {[{"channel"]}="HmIP-RC8 000B1BE98D94DE:1", ["buttonnr"]=1}

string cAmp = "&";
! Domoticz production system url base for the http api request
string urlBase = "'http://domoticz-ip:port/json.htm";
! Custom event name = MUST match name of the Domoticz dzVents script
string customEvent = "Homematic_remote_control";

! Get the object datapoint of the remote control from $src$
object objDatapoint = dom.GetObject ("$src$");
if (objDatapoint) {
    ! Channel object 1-8 according to buttons 1-8, i.e., HmIP-RC8 000B1BE98D94DE:1
    var objChannel = dom.GetObject(objDatapoint.Channel());
    ! Get the buttonnr 1-8 from the channel
    integer buttonNr = (objChannel.Name().StrValueByIndex(":",1)).ToInteger();

    ! Custom event data parameter must be in json format
    string data = '{"channel": "#objChannel#", "buttonnr": "#buttonNr#"}';
    ! Build the Domoticz http rest request url to update the device using customEvent
    string urlRequest =
urlBase#"?type=command#&param=customevent#&event=#customEvent#&data=#data#";
    ! Run the command without a return result
    cmdRes = dom.GetObject("CUxD.CUX2801001:1.CMD_EXEC").State("wget -q -O - #urlRequest");
}
```

## Domoticz Configuration

### Automation Event

```
--[[  
remote_control_custom_event.dzvents  
Trigger: customEvent  
To handle remote control buttons from the homematicip HmIP-RC8 device connected to a Homematic system.  
A Homematic script creates a custom event with data:  
{["channel"]="HmIP-RC8 000B1BE98D94DE:1", ["buttonnr"]=1}  
An action is assigned to every button.  
]]--  
  
local CUSTOMEVENTNAME = ' Homematic_remote_control'  
return {  
    on = {  
        customEvents = { CUSTOMEVENTNAME, }  
    },  
    logging = {  
        level = domoticz.LOG_INFO, marker = 'REMOTECONTROL',  
    },  
    execute = function(domoticz, item)  
        local data = item.data  
        local buttonNr = data.buttonnr  
        if buttonNr == 1 then  
            -- action  
        end  
        if buttonNr == 2 then  
            -- action  
        end  
        -- more buttons ...  
    end  
}
```

## Solution Selector Switch (Explored)

### Homematic Script

If a Remote-Control device button short is pressed, an immediate action is taken by executing the script.  
The script assigns the datapoint id of the button short to the switch level 10-80 for buttons 1-8.  
The HTTP API switch light request with Selector Switch Idx and level are submitted to Domoticz.

```
! Function: Remote Control Selector Switch
! remote_control_selector_switch.script.
! If a button of the remote control is pressed short, send switch light command with level according
button nr.
! Via cuxd an http api request is submitted to domoticz.
! /json.htm?type=command & parameter=switchlight&idx=IDX&switchcmd=Set%20Level&level=LEVEL
string cAmp = "&";
! Domoticz system url base for the http api request
string urlBase = "'http://domoticz-ip:port/json.htm'";
string switchLevel = 0;
string deviceIdx = 41;
! Get the object datapoint of the remote control from $src$
object objDatapoint = dom.GetObject ("$src$");
if (objDatapoint) {
    ! Channelobject 1-8 according buttons 1-8, i.e., HmIP-RC8 000B1BE98D94DE:1
    var objChannel = dom.GetObject(objDatapoint.Channel());
    ! Get the buttonnr 1-8 from the channel
    integer buttonNr = (objChannel.Name().StrValueByIndex(":",1)).ToInteger();
    switchLevel = buttonNr * 10;
    ! Build the Domoticz http rest request url
    string urlRequest =
urlBase#"?type=command#cAmp#param=switchlight#cAmp#"idx="#deviceIdx#cAmp#"switchcmd=Set%20Level&leve
l="#switchLevel#'";
    WriteLine(urlRequest);
    ! Run the command without a return result
    cmdRes = dom.GetObject("CUxD.CUX2801001:1.CMD_EXEC").State("wget -q -O - "#urlRequest");
}
```

## Domoticz Configuration

### Device

Domoticz GUI > Setup > Devices

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
41	VirtualSensors	00014079	1	Remote Control	Light/Switch	Selector Switch	Off

Domoticz GUI > Switches

Device has 9 levels.  
Level 0 Off is not used (hidden).

Levels 10-80 have level names 1-8 according Remote-Control device buttons.

The Level property is used in the Automation Event to set the action for the button pressed.

### Automation Event

Domoticz Device Event triggered by the Homematic Script

```
--[[  
    remote_control_selector_switch.dzevents  
    To take action based on a selector switch devices trigger.  
    The selector switch level is set by a Homematic script triggered by a remote control device pressing short.  
    The Homematic script submits a HTTP API request with level 10-80 for button 1-8  
    http://domoticz-ip:port/json.htm?type=command&param=switchlight&idx=IDX&switchcmd=Set%20Level&level=LEVEL  
]]--  
  
-- Idx f the device*s(  
local IDX_SELECTOR_SWITCH_RC = 41  
  
return {  
    on = {  
        devices = {IDX_SELECTOR_SWITCH_RC}  
    },  
    execute = function(domoticz, device)  
        -- Button 1  
        if device.level == 10 then  
            domoticz.log(("Remote Control Button %d action"):format(device.level / 10))  
        end  
        -- Button 2  
        if device.level == 20 then  
            domoticz.log(("Remote Control Button %d action"):format(device.level / 10))  
        end  
    end  
}
```

# Explore

## Purpose

To explore interfacing [Homematic IP](#) compatible devices with Domoticz by using Homematic programs with scripts and addons.

# Homematic Scripting

The scripts are:

- used by various functions and integrated in programs,
- created to develop & test before these are integrated in the final programs.

## Script Development

Developing & testing scripts can be done in various ways.

From the two options outlined below, the author uses the Homematic Addon Script-Parser.

### Homematic WebUI

From the Homematic WebUI go to Programs and connections > Programs > Test script.

Example	Notes
<p>Test script</p> <p>Input:</p> <pre> // Declare channel name of the device string nameChannel = "HmIP-eTRV-2 000A18A9A64DAC:1"; // Get channel object var objChannel = obj1.GetObject(nameChannel); // Get channel properties var devicename = objChannel.Name(); var deviceID = objChannel.Device(); // Get specific datapoint by name var deviceValue = objChannel.DPByHssDP("ACTUAL_TEMPERATURE").Value(); // Log WriteLine(deviceName # "=" # deviceID # "=" # deviceValue); </pre> <p>Output:</p> <pre>HmIP-eTRV-2 000A18A9A64DAC:1=1390=19.300000</pre> <p><b>Execute</b> <b>Close</b></p>	<p>The Input field contains the script code.</p> <p>The Output field logs the result of the function WriteLn(Text).</p> <p>Button Execute to run the code defined in the Input field.</p> <p>Button Close closes the Test script window. The code is not saved, so ensure to copy the code first if to be used further.</p>

## Addon Script-Parser

As an alternative, implemented the addon [Script-Parser](#), which is easier to use than the Homematic WebUI Test Script dialog.

From the Homematic WebUI go to Settings > Control Panel > Additional Software > Script-Parser > Set.

A new browser tab is opened.

Example	Notes
<p><b>HomeMatic Script Executor v1.9</b></p> <p>Achtung! Geben Sie das Webinterface Ihrer CCU / RaspberryMatic etc. <b>NIEMALS</b> über eine Portfreigabe im Router für den externen Zugriff frei! Ihre Hausautomatisierung wird damit trotz gesetztem WebUI-Passwort von außen angreif- und verwundbar! Entfernen Sie die Portfreigabe und nutzen Sie Techniken wie VPN oder ReverseProxy um einen sicheren Fernzugriff auf Ihre CCU zu erhalten.</p> <p>Eingabe:</p> <div style="border: 1px solid #ccc; padding: 5px; width: fit-content;"> <input type="button" value="CCU-Inventur (Geräte, Kanäle, Datenpunkte) zeigen"/> <input type="button" value="Variablen zeigen"/> <input type="button" value="Programme zeigen"/>  <input type="button" value="gecachte Seiten zeigen"/> </div> <p>Um detaillierte Fehlermeldungen zu sehen: ggf. SSH in CCU aktivieren, putty-Session zur CCU starten (<a href="#">putty</a>), einloggen, Befehl 'tail -f /var/log/messages   grep "Error.*near"' (ohne führendes und endendes ' eingeben!) eingeben.</p> <pre>! SHORT VERSION var nameChannel = "HmIP-MOD-OC8 000D1BE9A4E733:10"; var typeDatapoint = "STATE"; var deviceValue = dom.GetObject(nameChannel).DPByHssDP(typeDatapoint).Value(); WriteLine(nameChannel # "-" # typeDatapoint # ":" # deviceValue);</pre> <p>Ausgabe:</p> <div style="border: 1px solid #ccc; padding: 5px; width: fit-content;">HmIP-MOD-OC8 000D1BE9A4E733:10-STATE:false</div>	<p>Enter the script code in field "Eingabe".</p> <p>Press "Ausführen" to see the output of methods WriteLine in field "Ausgabe"</p>

## Development Hints

*Taken from Homematic Forum*

It is advised to keep scripts short & simple to avoid scripts from running long as this might cause issues with other CCU processes – scripts are running single threaded.  
(state 20210513)

For complex scripting, consider external solutions [RedMatic](#), [CCU-Jack](#) or other.

Build the script and iterate = execute line-by-line.

Use the function WriteLine(string) line-by-line.

Check Output from function WriteLine:

Variable content or return state of a command.

If Output remains empty, there is an error in the code.

If the final code OK, integrate the code (copy & paste) in the target Homematic program.  
Ensure to remove or comment out the function WriteLine().

If closing the Test Script Window, the code is gone, ensure to copy first.

Ensure to document (comment with ! follows by space) each code line or group of lines.

! This is a code comment.
---------------------------

Device channel names can be obtained from the Homematic WebUI  
(Homepage > Status and control > Devices):

```
string nameChannel = "HmIP-eTRV-2 000A18A9A64DAC:1";
```

The channel name is used to get device datapoints from the channel.

Declare special characters as variables:

For URL's executed by CuxD declare the string cAmp = "&";  
The string is concatenated in the URL declaration.

#### *Possible Code Errors*

- Missing semicolon ";" at the end of a code line
- Missing type declaration
  - Wrong: resCmd = ..
  - Right: var resCmd = ..
- Missing " character in a string
  - Wrong: WriteLine(actData # " # actTime);
  - Right: WriteLine(actData # " " # actTime);

To output the actual date and time:

```
string actDate = system.Date("%d.%m"); ! sDate = "09.08"; "%d.%m.%Y" = "09.08.2020";
string actTime = system.Date("%H:%M"); ! sTime = "07:32"; "%H:%M:%S" = "07:32:00";
WriteLine(actDate # " " # actTime);
```

# Script Examples

## List Devices

### List all Devices with Name, ID and HSS Type

```
! Get all devices objects.
var objIDs = dom.GetObject(ID_DEVICES).EnumUsedIDs();

! Loop over all datapoints for the devices found.
string id;
foreach(id, objIDs){
    !Get datapoint.
    var item = dom.GetObject(id);
    ! var device = dom.GetObject(item.Device());
    ! Name: Briefkasten Status: ID=2530
    WriteLine("Name: " # item.Name() # ":" # item.ID() # ":" # HSS=" # item.HssType());
}
```

```
Name: Alert Indicator: ID=3884: HSS=HmIP-MOD-OC8
Name: Bad: ID=1786: HSS=HmIP-eTRV-B
and more
```

### List Thermostat Devices with Name, ID and HSS Type (“TRV”).

The HSS type is compared against TRV only.

```
! Get all devices objects.
var objIDs = dom.GetObject(ID_DEVICES).EnumUsedIDs();

! Loop over all datapoints for the devices found.
string id;
foreach(id, objIDs){
    !Get datapoint.
    var item = dom.GetObject(id);
    ! Name: MakeLab: ID=1390: HSS=HmIP-eTRV-2
    var hssType = item.HssType();
    if (hssType.Find("TRV") > -1) {
        WriteLine("Name: " # item.Name() # ":" # item.ID() # ":" # HSS=" # hssType);
    }
}
```

```
Name: Bad: ID=1786: HSS=HmIP-eTRV-B
Name: Dusche: ID=1836: HSS=HmIP-eTRV-B
Name: Esszimmer: ID=1684: HSS=HmIP-eTRV-2
Name: Flur: ID=1530: HSS=HmIP-eTRV-2
Name: MakeLab: ID=1390: HSS=HmIP-eTRV-2
Name: Wohnzimmer-1: ID=1634: HSS=HmIP-eTRV-B
Name: Wohnzimmer-2: ID=1584: HSS=HmIP-eTRV-B
```

## List Thermostat Devices with datapoints

Datapoints: Channel Name, Datapoint Name, ACTUAL\_TEMPERATURE and SET\_POINT\_TEMPERATURE

### Steps

1. Get all the devices as objects.
2. Loop over the devices.
3. Get for each device the HSS type.
4. If HSS type is a thermostat.
5. Loop over all device channels to get the datapoints.

```
WriteLine("Channel Name, Datapoint Name, ACTUAL_TEMPERATURE, SET_POINT_TEMPERATURE");
var objIDs = dom.GetObject(ID_DEVICES).EnumUsedIDs();
string id;
foreach(id, objIDs){
    var item = dom.GetObject(id);
    var hssType = item.HssType();
    if (hssType.Find("TRV") > -1) {
        string chid;
        foreach(chid, item.Channels()) {
            var n = dom.GetObject(chid).Name();
            var pv = dom.GetObject(chid).DPByHssDP("ACTUAL_TEMPERATURE");
            var sp = dom.GetObject(chid).DPByHssDP("SET_POINT_TEMPERATURE");
            if (pv && sp) {
                WriteLine(n # " Name: " # item.Name() # " PV:" # pv.Value() # " SP:" # sp.Value());
            }
        }
    }
}
```

Channel Name, Datapoint Name, ACTUAL_TEMPERATURE, SET_POINT_TEMPERATURE
HmIP-eTRV-B 002018A99D2097:1, Name: Bad, PV:23.500000, SP:23.000000
HmIP-eTRV-B 00201A49952CB8:1, Name: Dusche, PV:18.300000, SP:6.000000
HmIP-eTRV-2 000A1A49A0D878:1, Name: Esszimmer, PV:22.300000, SP:23.000000
HmIP-eTRV-2 000A1A49A0D8A5:1, Name: Flur, PV:19.700000, SP:6.000000
HmIP-eTRV-2 000A18A9A64DAC:1, Name: MakeLab, PV:19.300000, SP:7.000000
HmIP-eTRV-B 00201A49952CB1:1, Name: Wohnzimmer-1, PV:19.900000, SP:23.000000
HmIP-eTRV-B 00201A499D76F8:1, Name: Wohnzimmer-2, PV:20.500000, SP:23.000000

## Get Device Value

Get the actual temperature (Channel 1, Datapoint “ACTUAL\_TEMPERATURE”) of a thermostat device HmIP-eTRV-2 (Channel name: HmIP-eTRV-2 000A18A9A64DAC:1).  
The channel name is taken from the Homematic WebUI  
(Homepage > Status and control > Devices):

MakeLab		HmIP-eTRV-2 000A18A9A64DAC:1	
		Connection partner radiator thermostat (manual operation, transmitter)	

### Long and Short Version

```

! LONG VERSION
! Declare channel name of the device.
string nameChannel = "HmIP-eTRV-2 000A18A9A64DAC:1";

! Get channel object.
var objChannel = dom.GetObject(nameChannel);

! Get channel properties.
var deviceName = objChannel.Name();
var deviceID = objChannel.Device();

! Get specific datapoint by name.
var deviceValue = objChannel.DPByHssDP("ACTUAL_TEMPERATURE").Value();

! Log the name, datapoint id, device value.
WriteLine(deviceName # "=" # deviceID # "=" # deviceValue);

! SHORT VERSION
var nameChannel = "HmIP-MOD-OC8 000D1BE9A4E733:10";
var typeDatapoint = "STATE";
var deviceValue = dom.GetObject(nameChannel).DPByHssDP(typeDatapoint).Value();
WriteLine(nameChannel # "-" # typeDatapoint # ":" # deviceValue);
! HmIP-MOD-OC8 000D1BE9A4E733:10-STATE:false

```

```
HmIP-eTRV-2 000A18A9A64DAC:1=1390=19.300000
```

## Set Device Value

Set the value of a device channel.

```
! LONG VERSION
! Declare channel name of the device.
string nameChannel = "HmIP-MOD-OC8 000D1BE9A4E733:10";
string typeDatapoint = "STATE";
boolean channelState = false;

! Get channel object.
var objChannel = dom.GetObject(nameChannel);
! Set datapoint value.
objChannel.DPByHssDP(typeDatapoint).State(channelState);
! Get datapoint by type.
var deviceValue = objChannel.DPByHssDP(typeDatapoint).Value();
! Log the name, datapoint id, device value.
WriteLine(nameChannel # "=" # typeDatapoint # ":" # deviceValue);

! SHORT VERSION
var nameChannel = "HmIP-MOD-OC8 000D1BE9A4E733:10";
var typeDatapoint = "STATE";
var channelState = true;
dom.GetObject(nameChannel).DPByHssDP(typeDatapoint).State(channelState);
WriteLine(nameChannel # "-" # typeDatapoint # ":" # channelState);
```

```
! HmIP-MOD-OC8 000D1BE9A4E733:10-STATE:true
```

## Get Device Channel Datapoints

```

! Get all device channel datapoints.
! Name of the channel as shown in the Homematic WebUI
string nameChannel = "HmIP-eTRV-2 000A18A9A64DAC:1";

! Get channel object.
var objChannel = dom.GetObject(nameChannel);

! Get channel properties.
var deviceName = objChannel.Name();
var deviceID = objChannel.Device();
WriteLine("Device Name: " # deviceName # ", ID: " # deviceID);

! ID for each of the channel datapoints - i.e., 1415
string id;

! Loop over all datapoints
foreach(id, objChannel.DPs()) {
    ! Get the datapoint.
    var dp = dom.GetObject(id);
    ! Log Name (ID): Value - note: id is same as dp.ID()
    WriteLine(dp.Name() # "(" # dp.ID() # ") : " # dp.Value());
}

```

```

Device Name: HmIP-eTRV-2 000A18A9A64DAC:1, ID: 1390
HmIP-RF.000A18A9A64DAC:1.ACTIVE_PROFILE (1415) : 2
HmIP-RF.000A18A9A64DAC:1.ACTUAL_TEMPERATURE (1416) : 19.900000
HmIP-RF.000A18A9A64DAC:1.ACTUAL_TEMPERATURE_STATUS (1417) : 0
HmIP-RF.000A18A9A64DAC:1.BOOST_MODE (1418) : false
HmIP-RF.000A18A9A64DAC:1.BOOST_TIME (1419) : 0
HmIP-RF.000A18A9A64DAC:1.CONTROL_DIFFERENTIAL_TEMPERATURE (1420) :
HmIP-RF.000A18A9A64DAC:1.CONTROL_MODE (1421) :
HmIP-RF.000A18A9A64DAC:1.DURATION_UNIT (1422) :
HmIP-RF.000A18A9A64DAC:1.DURATION_VALUE (1423) :
HmIP-RF.000A18A9A64DAC:1.FROST_PROTECTION (1424) : false
HmIP-RF.000A18A9A64DAC:1.LEVEL (1425) : 0.000000
HmIP-RF.000A18A9A64DAC:1.LEVEL_STATUS (1426) : 0
HmIP-RF.000A18A9A64DAC:1.PARTY_MODE (1427) : false
HmIP-RF.000A18A9A64DAC:1.PARTY_SET_POINT_TEMPERATURE (1428) : 0.000000
HmIP-RF.000A18A9A64DAC:1.PARTY_TIME_END (1429) :
HmIP-RF.000A18A9A64DAC:1.PARTY_TIME_START (1430) :
HmIP-RF.000A18A9A64DAC:1.QUICK_VETO_TIME (1431) : 0
HmIP-RF.000A18A9A64DAC:1.SET_POINT_MODE (1432) : 1
HmIP-RF.000A18A9A64DAC:1.SET_POINT_TEMPERATURE (1433) : 4.500000
HmIP-RF.000A18A9A64DAC:1.SWITCH_POINT_OCCURRED (1434) : false
HmIP-RF.000A18A9A64DAC:1.VALVE_ADAPTION (1435) : false
HmIP-RF.000A18A9A64DAC:1.VALVE_STATE (1436) : 4
HmIP-RF.000A18A9A64DAC:1.WINDOW_STATE (1437) : 0

```

## Get System Variables

```

! List all system variables.
var objIDs = dom.GetObject(ID_SYSTEM_VARIABLES).EnumUsedIDs();
string id = "";

! Loop over all datapoints
foreach(id, objIDs){
    ! Get the system variable datapoint object.
    var sysVar = dom.GetObject(id);
    ! Show system variable name and value.
    WriteLine("Name: " # sysVar.Name() # " ID: " + sysVar.ID() # " Value: " + sysVar.Value());
}

WriteLine("*****");
! Get specific system var.
string id = "DutyCycle";
var sysVar = dom.GetObject(id);
! Show system variable name and value.
WriteLine("Name: " # sysVar.Name() # " ID: " + sysVar.ID() # " Value: " + sysVar.Value());

```

```

Name: ${sysVarAlarmZone1} ID: 1236 Value:
Name: ${sysVarPresence} ID: 950 Value:
Name: BatteryCheck ID: 2799 Value: OK 12.05 00:00
Name: DutyCycle ID: 3032 Value: 1.000000
*****
Name: DutyCycle ID: 3032 Value: 1.000000

```

## Set System Variable by ID

Set the value of a system variable named “DutyCycle”.

```

! Get specific system var.
string name = "DutyCycle";
var sysVar = dom.GetObject(ID_SYSTEM_VARIABLES).Get(name);
! Show system variable name and value.
WriteLine("Name: " # sysVar.Name() # " ID: " + sysVar.ID() # " Value: " + sysVar.Value());

! Set the value of a sysVar.
WriteLine("Set new value");
sysVar.State(1.2);
WriteLine("Name: " # sysVar.Name() # " ID: " + sysVar.ID() # " Value: " + sysVar.Value());

```

```

Name: DutyCycle ID: 3032 Value: 1.000000
Set new value.
Name: DutyCycle ID: 3032 Value: 1.200000

```

## Set System Variable from Device Value

Set the value of a system variable, named “Domoticz.Data”, from a device value (Thermostat ACTUAL\_TEMPERATURE).

The system variable is defined via Domoticz WebUI > Settings > System variable > New.

The script could be triggered by a program every 15 minutes to update the system variable.

### Example: Type: String

Set the device value to the system variable as string.

```
! Get device value and assign to system variable.
var actTemperature = dom.GetObject("HmIP-eTRV-2
000A18A9A64DAC:1").DPByHssDP("ACTUAL_TEMPERATURE").Value();

! Get system variable.
var sysVar = dom.GetObject("Domoticz.Data");
WriteLine(sysVar.Value());

! Set value.
sysVar.State(actTemperature);
WriteLine(sysVar.Value());
```

```
???
19.900000
```

### Example: ValueList

The type ValueList(A;B;C...) could be used to collect data and sent these to a Domoticz Custom Event, which splits the data and assigns the values to the devices.

The next script holds two values, ACTUAL\_TEMPERATURE and SET\_POINT\_TEMPERATURE, from a thermostat device.

```
! Get device values and assign to system variable type ValueList.

! Get the device data: actual temperature & setpoint.
var actTemperature = dom.GetObject("HmIP-eTRV-2
000A18A9A64DAC:1").DPByHssDP("ACTUAL_TEMPERATURE").Value();
var actSetPoint = dom.GetObject("HmIP-eTRV-2
000A18A9A64DAC:1").DPByHssDP("SET_POINT_TEMPERATURE").Value();

! Get system variable containing ValueList.
var sysVar = dom.GetObject("Domoticz.Data");
! Log previous ValueList data
WriteLine(sysVar.ValueList());

! Set the new ValueList values.
sysVar.ValueList(actTemperature # ";" # actSetPoint);
! Log the updated values.
WriteLine(sysVar.ValueList());
```

```
19.900000;4.500000
20.000000;6.000000
```

*The thermostat setpoint was changed from 4.5 (=Manual state OFF) to 6.*

### Example ValueList Index Change

In this example the second entry of the ValueList() is changed.  
This is done by using the method Replace() with StrValueByIndex().

```
! Get device values and assign to system variable type valuelist
! Get the device data, actual temperature & setpoint.
var actTemperature = dom.GetObject("HmIP-eTRV-2
000A18A9A64DAC:1").DPByHssDP("ACTUAL_TEMPERATURE").Value();
var actSetPoint = dom.GetObject("HmIP-eTRV-2
000A18A9A64DAC:1").DPByHssDP("SET_POINT_TEMPERATURE").Value();

! Get system variable containing valuelist
var sysVar = dom.GetObject("Domoticz.Data");
! Log previous valueist data
WriteLine(sysVar.ValueList());

! Set the new value list values.
sysVar.ValueList(actTemperature # ";" # actSetPoint);
! Log the updated values.
WriteLine(sysVar.ValueList());
WriteLine("****");

! Example changing ValueList entry 2 (Index 1)
string x = sysVar.ValueList().StrValueByIndex(";",1);
WriteLine(x);

var newlist = sysVar.ValueList().Replace(sysVar.ValueList().StrValueByIndex(";", 1), 9);
sysVar.ValueList(newlist);
WriteLine(sysVar.ValueList());
```

```
20.000000;9
20.000000;6.000000
6.000000
20.000000;9
```

#### Lines

1. Current values.
2. Updated values from device data.
3. Value list second value (index 1).
4. Value list with updated second value.

## Domoticz Examples

### Get Device Value & Update Domoticz Device Value

#### Purpose

Get the actual temperature (datapoint "ACTUAL\_TEMPERATURE") of a Homematic thermostat device HmIP-eTRV-2 and update the Domoticz Virtual Sensor (Type: Temp, SubType: LaCrosse TX3, Idx: 31).

The Domoticz device is updated by running the Homematic Addon CuxD command CMD\_EXEC with "wget url", which returns true or false.

The URL is a Domoticz [HTTP JSON/API](#) command to update the Temperature device value:

```
http://domoticz-ip:port/json.htm?type=command&param=udevice&idx=IDX&nvalue=0&svalue=TEMP
```

#### Homematic Script

```
! Get the actual date & time for logging
string actDate = system.Date("%d.%m"); ! sDate = "09.08"; "%d.%m.%Y" = "09.08.2020";
string actTime = system.Date("%H:%M"); ! sTime = "07:32"; "%H:%M:%S" = "07:32:00";
WriteLine(actDate # " " # actTime);

! Get the device channel properties.
string nameChannel = "HmIP-eTRV-2 000A18A9A64DAC:1";
var objChannel = dom.GetObject(nameChannel);
var deviceName = objChannel.Name();
var deviceID = objChannel.Device();
var deviceValue = objChannel.DPBvHsDP("ACTUAL_TEMPERATURE").Value();
! HmIP-eTRV-2 000A18A9A64DAC:1: 1390=19.900000
WriteLine(deviceName # ":" # deviceID # "=" # deviceValue);

! Built the url for the Domoticz HTTP JSON/API request.
string cAmp = "&";
string urlBase = "'http://domoticz-ip:8080/json.htm'";
string deviceIdx = "31";

! Build the Domoticz http rest request url to update the device.
! http://domoticz-ip:port/json.htm?type=command&param=udevice&idx=IDX&nvalue=0&svalue=TEMP
string urlRequest =
urlBase#"?type=command#cAmp#param=udevice#cAmp#idx="#deviceIdx#cAmp#"nvalue=0#cAmp#"svalue="#deviceValue#'";
WriteLine(urlRequest);

! Run CuxD wget with the url request and check output is true.
var cmdRes = dom.GetObject("CUxD.CUX2801001:1.CMD_EXEC").State("wget -q -O - "#urlRequest);
WriteLine("Exec Result: " # cmdRes);
```

```
12.05 10:53
HmIP-eTRV-2 000A18A9A64DAC:1: 1390=19.900000
'http://domoticz-ip:8080/json.htm?type=command&param=udevice&idx=31&nvalue=0&svalue=19.900000'
Exec Result: true
```

## Domoticz

Domoticz log level is set in Domoticz WebUI > Settings > Other dzVents: Enabled & Log Level.

If Log Level is Debug, the log with Debug info:

```
2021-05-12 10:58:09.030 Status: Incoming connection from: ccu-ip
2021-05-12 10:58:09.073 Status: dzVents: Debug: Dumping domoticz data to
/home/pi/domoticz/scripts/dzVents/domoticzData.lua
2021-05-12 10:58:09.110 Status: dzVents: Debug: Processing device-adapter for MakeLab Temperature:
Temperature device adapter
2021-05-12 10:58:09.110 Status: dzVents: Debug: dzVents version: 3.1.8
2021-05-12 10:58:09.110 Status: dzVents: Debug: Event triggers:
2021-05-12 10:58:09.110 Status: dzVents: Debug: - Device: MakeLab Temperature
```

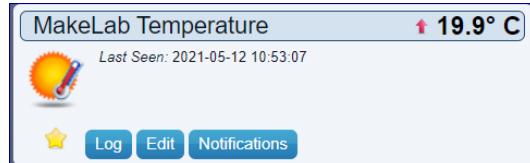
If Log Level is “Errors + Minimal Execution Info + Generic Info”, the log info:

```
2021-05-12 10:53:07.560 Status: Incoming connection from: ccu-ip
```

The device information from the Domoticz WebUI > Setup > Devices:

Idx	Hardware	ID	Unit	Name	Type	SubType	Data	Last Seen
31	VirtualSensors	1406F	1	MakeLab Temperature	Temp	LaCrosse TX3	19.9 C	2021-05-12 10:53:07

The device widget from the Domoticz WebUI > Temperature:



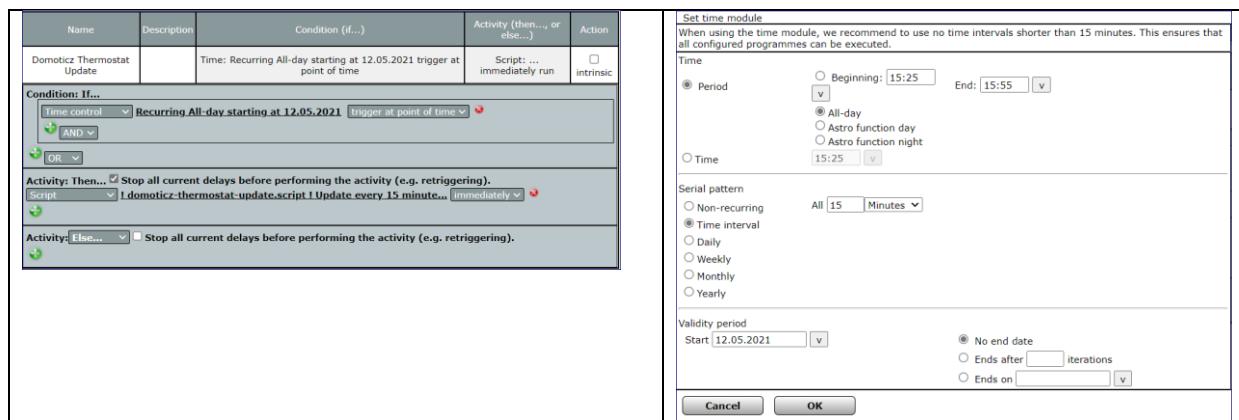
## Update Domoticz Devices from System Variable ValueList

### Purpose

Send the data of a Homematic System Variable type ValueList() to Domoticz. The system variable contains the thermostat ACTUAL\_TEMPERATURE and SET\_POINT\_TEMPERATURE.

The Domoticz dzVents system uses a custom event to handle the data received, i.e., updates the Temperature device and sets the SetPoint of the Thermostat device (as silent without acting back to Homematic).

### Homematic Program



### Condition & Activity

If Time Control Time interval 15 Minutes then Activity Script domoticz-thermostat-update.

### Homematic Script

```
! domoticz-thermostat-update.script
! Update every 15 min value of a Domoticz device via HTTP JSON/API request from Homematic Device.
! Homematic Device: HmIP-eTRV-2 000A18A9A64DAC:1
! Domoticz Devices:
! Temperature (T:Temp, ST:LaCrosse TX3, Idx: 31)
! SetPoint (T:Thermostat, ST:SetPoint, Idx: 30)
! Domoticz Custom Event Data send:
! {"sp":4.5, "pv":20.2}
! 20210512 rwbl

! Get actual date & time
string actDate = system.Date("%d.%m"); ! sDate = "09.08"; "%d.%m.%Y" = "09.08.2020";
string actTime = system.Date("%H:%M"); ! sTime = "07:32"; "%H:%M:%S" = "07:32:00";
WriteLine(actDate # " " # actTime);

! Get the device data: actual temperature & setpoint
var actTemperature = dom.GetObject("HmIP-eTRV-2
000A18A9A64DAC:1").DPByHssDP("ACTUAL_TEMPERATURE").Value();
var actSetPoint = dom.GetObject("HmIP-eTRV-2
000A18A9A64DAC:1").DPByHssDP("SET_POINT_TEMPERATURE").Value();

! Get system variable containing ValueList
var sysVar = dom.GetObject("Domoticz.Data");
WriteLine(sysVar.ValueList());

! Set the new ValueList values as custom event data json format
string data = "pv' #' :# actTemperature #' , #' sp' #' :# actSetPoint";
sysVar.ValueList(data);
WriteLine(sysVar.ValueList());

! Domoticz system url base for the http api request
string urlBase = 'http://domoticz-ip:8080/json.htm';
string customEvent = "homematic_thermostat_update";
```

```

! Build the Domoticz http rest request url to update the device using customEvent
! Make JSON format for the data submitted
data = "{" # data # "}";
string cAmp = "&";
string urlRequest =
urlBase#"?type=command#cAmp#param=customevent#cAmp#event="#customEvent#cAmp#"data="#data#"';
WriteLine(urlRequest);

! Run the command without a return result
var cmdRes = dom.GetObject("CUxD.CUX2801001:1.CMD_EXEC").State("wget -q -O - "#urlRequest);
WriteLine("CMD_EXEC: " # cmdRes);

```

```

12.05 14:28
"pv":20.200000,"sp":4.500000
"pv":20.200000,"sp":4.500000
'http://domoticz-
ip:8080/json.htm?type=command&param=customevent&event=Homematic_thermostat_test&data={"pv":20.200000,"s
p":4.500000}'
CMD_EXEC: true

```

## Domoticz Automation Event dzVents Script

```

local CUSTOMEVENT_NAME = "homematic_thermostat_update"
local IDX_PV = 31
local IDX_SP = 30

return {
  on = { customEvents = { CUSTOMEVENT_NAME } },
  data = {},
  logging = {},
  execute = function(domoticz, item)
    if (item.isCustomEvent and item.isJSON) then
      -- {[{"sp":4.5, "pv":20.2}]
      domoticz.log(item.json)
      domoticz.devices(IDX_SP).updateSetPoint(item.json.sp).silent()
      domoticz.devices(IDX_PV).updateTemperature(item.json.pv).silent()
      domoticz.log(string.format("Thermostat: SP=%.2f, PV=%.2f.",
        item.json.sp,
        item.json.pv
      ))
    end
  end
}

```



## Update Domoticz Devices

### Purpose

This example updates the value of Domoticz devices Temperature & SetPoint from a Homematic thermostat device, datapoints ACTUAL\_TEMPERATURE and SET\_POINT\_TEMPERATURE.

The Domoticz dzVents system uses a custom event to handle the data received from Homematic, i.e., updates the Temperature device and sets the SetPoint of the Thermostat device (as silent without acting back to Homematic).

1. Homematic Script
  - a. reads in regular intervals the device datapoint values.
  - b. creates JSON object with for each datapoint name:value pairs idx,value.
  - c. submits a Domoticz HTTP JSON/API Custom Event request via CuxD ("wget").
2. Domoticz Automation Event dzVents Custom Event
  - a. parses the received JSON object into devices.
  - b. sets the Domoticz device value (silent).

This script can handle multiple thermostats.

### Example JSON Object

```
{
  ["timestamp"]="12.05 15:28",
  ["thermostats"]={
    {
      ["idxsp"]=30, ["pv"]=20.1, ["idxpv"]=31, ["sp"]=4.5
    }
  }
}
```

### Homematic Program

Name	Description	Condition (if...)	Activity (then... or else...)	Action
Domoticz Bulk Update		Time: Recurring at 19:37 o'clock starting at 11.05.2021 trigger at point of time	Script: ... immediately run	<input type="checkbox"/> intrinsic
<b>Condition: If...</b>				
<b>Time control</b> Recurring at 19:37 o'clock starting at 11.05.2021 trigger at point of time				
<b>Activity: Then...</b> <input checked="" type="checkbox"/> Stop all current delays before performing the activity (e.g. retriggering).				
Script: ! Domoticz-Bulk-Update 120210512.rwbt ! Get actual date & time immediately				
<b>Activity: Else...</b> <input type="checkbox"/> Stop all current delays before performing the activity (e.g. retriggering).				

Set time module  
When using the time module, we recommend to use no time intervals shorter than 15 minutes. This ensures that all configured programmes can be executed.

Time
  Period
  Beginning: 15:25 
 End: 15:55 
  
 All-day
  Astro function day
  Astro function night
  Time
 15:25

Serial pattern
  Non-recurring
  Time interval
  Daily
  Weekly
  Monthly
  Yearly

Validity period
 Start: 12.05.2021 
 No end date
  Ends after  iterations
  Ends on

### Condition & Activity

If Time Control Time interval 15 Minutes then Activity Script domoticz-thermostat-update

## Homematic Script

```

! Get actual date & time
string actDate = system.Date("%d.%m"); ! sDate = "09.08"; "%d.%m.%Y" = "09.08.2020";
string actTime = system.Date("%H:%M"); ! sTime = "07:32"; "%H:%M:%S" = "07:32:00";
WriteLine(actDate # " " # actTime);

! for each device get the data: idx, actual temperature & setpoint
! for tests do manual build json string first
string idxPV = "31";
string idxSP = "30";
var actPV = dom.GetObject("HmIP-eTRV-2 000A18A9A64DAC:1").DPByHssDP("ACTUAL_TEMPERATURE").Value();
var actSP = dom.GetObject("HmIP-eTRV-2 000A18A9A64DAC:1").DPByHssDP("SET_POINT_TEMPERATURE").Value();
string jsonDevData = '{' # '"idxpv"' # ':' # idxPV # ',' # '"pv"' # ':' # actPV # ',' # '"idxsp"' # ':' #
# idxSP # ',' # '"sp"' # ':' # actSP # '}';
WriteLine(jsonDevData);

string jsonData = '{' # '"timestamp"' # ':' # actDate # " " # actTime # '' , ' # '"thermostats"' # '['
# jsonDevData # ']'}';
WriteLine(jsonData);

! Domoticz system url base for the http api request
string urlBase = "'http://domoticz-ip:8080/json.htm";
string customEvent = "domoticz-Homematic-bulk-update";

! Build the Domoticz http rest request url to update the device using customEvent
! Make JSON format for the data submitted
string cAmp = "&";
string urlRequest =
urlBase#"?type=command#cAmp#param=customevent#cAmp#event="#customEvent#cAmp#"data="#jsonData#""";
WriteLine(urlRequest);

! Run the command without a return result
var cmdRes = dom.GetObject("CUXD.CUX2801001:1.CMD_EXEC").State("wget -q -O - "#urlRequest);
WriteLine("CMD_EXEC: " # cmdRes);

```

```

13.05 11:13
{"idxpv":31,"pv":19.000000,"idxsp":30,"sp":4.500000}
{"timestamp":"13.05 11:13","thermostats":[{"idxpv":31,"pv":19.000000,"idxsp":30,"sp":4.500000}]}
'http://domoticz-ip:8080/json.htm?type=command&param=customevent&event=domoticz-Homematic-bulk-
update&data={"timestamp":"13.05
11:13","thermostats":[{"idxpv":31,"pv":19.000000,"idxsp":30,"sp":4.500000}]}'
CMD_EXEC: true

```

## Domoticz dzVents Script

```

local CUSTOMEVENT_NAME = "domoticz-Homematic-bulk-update"

-- Set the state of the thermostats
local function setThermostats(domoticz, result)
    local data = {}
    local url = ""
    for key, value in pairs(result) do
        -- domoticz.log(value.idxpv .. "," .. value.idxsp);
        domoticz.devices(value.idxsp).updateSetPoint(value.sp).silent()
        domoticz.devices(value.idxpv).updateTemperature(value.pv).silent()
        domoticz.log(string.format("Thermostat: SP=%.2f, PV=%.2f.", value.sp, value.pv))
    end
end

return {
    on = { customEvents = { CUSTOMEVENT_NAME } },
    data = {},
    logging = {},
    execute = function(domoticz, item)
        if (item.isCustomEvent and item.isJSON) then
            -- domoticz.log(item.json)
            -- result
            local thermostats = item.json.thermostats
            -- {[{"sp":4.5, "idxsp":30, "pv":20.1, "idxpv":31}]
            -- domoticz.log(thermostats)
            setThermostats(domoticz, thermostats)
        end
    end
}
}

```



## Update Domoticz Devices Bulk

### Purpose

As a Homematic recurring All-Day event send (using CuxD CMD\_EXEC) every 15 minutes, selective Homematic devices data in JSON format to Domoticz.

A Domoticz Custom Event handles updating the related Domoticz Devices.

### Steps

#### 1. Homematic Script

- a. Get all used devices as list.
- b. Loop over devices list and select HSS type “TRV” = Thermostats eTRV-B-eTRV-2 ...
- c. Loop over all device channels
  - get channel name and
  - select datapoints ACTUAL\_TEMPERATURE and SET\_POINT\_TEMPERATURE.
- d. Build channel JSON object with name:value pairs:
  - name:value.
  - pv:value – this is the Thermostat actual temperature.
  - sp:value – this is the Thermostat set point.
- e. Add the channel JSON object to the JSON data object used to send to Domoticz Custom Event.
- f. Finalize the JSON data object by adding the timestamp.
- g. Build the URL for the Domoticz HTTP JSON/API request.
- h. Submit the HTTP JSON /API request to Domoticz using CuxD CMD\_EXEC command.
- i. Optional: assign the result to a Homematic System Variable.

#### 2. Domoticz Automation Event dzVents Custom Event

- a. If triggered then get the Custom Event data as JSON object item.
- b. Get the JSOM member named thermostats and assign to Lua table.
- c. Loop over the Lua table with thermostats data and select from a defined Lua table thermostats, the thermostat idx of the devices Temperature & Setpoint.
- d. Update the devices Temperature & Setpoint.

## Homematic Script

```

! domoticz-Homematic-bulk-update
! Get all devices
var listDevices = dom.GetObject(ID_DEVICES).EnumUsedIDs();
! Declare JSON data object to be sent to Domoticz via HTTP JSON/API custom event
string jsonData = "";
string deviceID;
! Loop over the devices list
foreach(deviceID, listDevices) {
    ! Get the device
    var deviceItem = dom.GetObject(deviceID);
    ! Check the HSS type
    var hssType = deviceItem.HssType();
    ! Device type thermostat TRV found?
    if (hssType.Find("TRV") > -1) {
        ! Loop over the device channels to get selective datapoints
        string channelID;
        foreach(channelID, deviceItem.Channels()) {
            var name = dom.GetObject(channelID).Name();
            var pv = dom.GetObject(channelID).DPByHssDP("ACTUAL_TEMPERATURE");
            var sp = dom.GetObject(channelID).DPByHssDP("SET_POINT_TEMPERATURE");
            ! Create the JSON object with name, pv, sp
            if (pv && sp) {
                var jsonObject = '{}';
                jsonObject = jsonObject + '"name"' # ':' # ''# name # ''# ',';
                jsonObject = jsonObject + '"pv"' # ':' # pv.Value() # ',';
                jsonObject = jsonObject + '"sp"' # ':' # sp.Value();
                jsonObject = jsonObject + '}';
                jsonData = jsonData + jsonObject;
            }
        }
    }
}
! Workaround to remove the last "," from the JSON object. Add # and remove ',#' by ''.
jsonData = jsonData + '#';
jsonData = jsonData.Replace(',#', '');
! Add timestamp
! Get actual date & time
string actDate = system.Date("%d.%m"); ! sDate = "09.08"; "%d.%m.%Y" = "09.08.2020";
string actTime = system.Date("%H:%M"); ! sTime = "07:32"; "%H:%M:%S" = "07:32:00";
string jsonTimeStamp = "'timestamp'" # ':' # actDate # ' ' # actTime # '';
! Final JSON object
jsonData = '{' # jsonTimeStamp # ',' + '"thermostats"' # ':' # '[' # jsonData # ']}'';
! Domoticz system url base for the http api request
string urlBase = "'http://domoticz-ip:8080/json.htm";
string customEvent = "domoticz-Homematic-bulk-update";
! Build the Domoticz http rest request url to update the device using customEvent
! Make JSON format for the data submitted
string cAmp = "&";
string urlRequest =
urlBase#"?type=command#cAmp#"param=customevent#cAmp#"event="#customEvent#cAmp#"data="#jsonData#'";
WriteLine(urlRequest);
! Run the exec command "wget"
var cmdRes = dom.GetObject("CUxD.CUX2801001:1.CMD_EXEC").State("wget -q -O - "#urlRequest);
WriteLine("CMD_EXEC: "# cmdRes);
! Optional update system variable DomoticzLog
string sysVarName = "DomoticzLog";
var sysVar = dom.GetObject(ID_SYSTEM_VARIABLES).Get(sysVarName);
! Set the sysvar value
string logEntry = actDate # " " # actTime # " - Domoticz Homematic Bulk Update: ";
if (cmdRes == true) {
    logEntry = logEntry # "OK";
}
else {
    logEntry = logEntry # "ERROR";
}
sysVar.State(logEntry);
WriteLine("Name: "# sysVar.Name() # " ID: " + sysVar.ID() # " Value: " + sysVar.Value());

```

```
'http://domoticz-ip:8080/json.htm?type=command&param=customevent&event=domoticz-Homematic-bulk-
update&data={
  "timestamp": "14.05 09:48",
  "thermostats": [
    {"name": "HmIP-eTRV-B 002018A99D2097:1", "pv": 23.000000, "sp": 23.000000},
    {"name": "HmIP-eTRV-B 00201A49952CB8:1", "pv": 18.100000, "sp": 6.000000},
    {"name": "HmIP-eTRV-2 000A1A49A0D878:1", "pv": 22.000000, "sp": 23.000000},
    {"name": "HmIP-eTRV-2 000A1A49A0D8A5:1", "pv": 19.500000, "sp": 6.000000},
    {"name": "HmIP-eTRV-2 000A18A9A64DAC:1", "pv": 18.600000, "sp": 4.500000},
    {"name": "HmIP-eTRV-B 00201A49952CB1:1", "pv": 19.200000, "sp": 23.000000},
    {"name": "HmIP-eTRV-B 00201A499D76F8:1", "pv": 19.900000, "sp": 23.000000}
  ]
}'
```

CMD\_EXEC: true

Name: DomoticzLog ID: 4765 Value: 14.05 10:14 - Domoticz Homematic Bulk Update: OK

## Domoticz Automation Event dzVents Script

The next Domoticz Automation Event dzVents script updates the selected Domoticz devices with the received thermostat data Name, PV (=Datapoint type ACTUAL\_TEMPERATURE), SP (=Datapoint type SET\_POINT\_TEMPERATURE).

The data is, as outlined previous, in JSON format. The main JSON name used is "thermostats" holding the name, pv and sp for each device.

The serial number of the device name is used to determine the Domoticz devices Idx to set the new values.

```
{
  "thermostats": [
    {"name": "HmIP-eTRV-B 002018A99D2097:1", "pv": 22.7, "sp": 23.0},
    {"name": "HmIP-eTRV-B 00201A49952CB8:1", "pv": 18.1, "sp": 6.0},
    {"name": "HmIP-eTRV-2 000A1A49A0D878:1", "pv": 22.6, "sp": 23.0},
    {"name": "HmIP-eTRV-2 000A1A49A0D8A5:1", "pv": 19.5, "sp": 6.0},
    {"name": "HmIP-eTRV-2 000A18A9A64DAC:1", "pv": 18.8, "sp": 4.5},
    {"name": "HmIP-eTRV-B 0201A49952CB1:1", "pv": 19.9, "sp": 23.0},
    {"name": "HmIP-eTRV-B 00201A499D76F8:1", "pv": 20.2, "sp": 23.0}
  ],
  "timestamp": "14.05 10:28"
}
```

```
-- [[
domoticz-Homematic-bulk-update.dzvents
Update several Domoticz devices from Homematic data.
The device data is grouped. In next example there is only a single group named "thermostats".
Each thermostat member has following data: name, pv, sp.
The actual temperature is named pv, set point temperature is sp.
]]-- 

-- The custom event name must match the name used in the Homematic script
local CUSTOMEVENT_NAME = "domoticz-Homematic-bulk-update"

-- Table holding all thermostats with serial number and Domoticz devices Temperature & Thermostat
SetPoint
-- TESTS: 2 thermostats are defined: makelab, wohnzimmer1
local tableThermostats = {
  ["makelab"] = {[{"serialnr": "000A18A9A64DAC", "idxpv": 31, "idxsp": 30},
                 {"serialnr": "000A1A49A0D878", "idxpv": 0, "idxsp": 0},
                 {"serialnr": "00201A49952CB1", "idxpv": 90, "idxsp": 89},
                 {"serialnr": "00201A499D76F8", "idxpv": 0, "idxsp": 0},
                 {"serialnr": "000A1A49A0D8A5", "idxpv": 0, "idxsp": 0},
                 {"serialnr": "002018A99D2097", "idxpv": 0, "idxsp": 0},
                 {"serialnr": "00201A49952CB8", "idxpv": 0, "idxsp": 0}
  ]
}

-- Get the idx for thermostat setpoint & setpoint for device selected from tableThermostats
-- The idx is a table with 3 keys location, pv and sp. If nothing found return empty table {}
-- Parameter:
-- channelName - Channel name of the thermostat device
-- Example:
-- local idx = getThermostatSetpoint(domoticz, value.name)
-- idx.location = Makelab, idx.pv=20.2, idx.sp=23.0
function getThermostatIdx(domoticz, channelName)
  local result = {}
  -- Get the thermostat serialnr from the thermostats table matching the channelName
  for key, value in pairs(tableThermostats) do
    if string.match(channelName, value.serialnr) then
      result = {location = key, pv = value.idxpv, sp = value.idxsp}
      return result
    end
  end
end

-- Set the value of all thermostats
-- thermostats = table with data provided by Homematic
local function setThermostats(domoticz, thermostats)
  -- loop over the thermostats table
  for key, value in pairs(thermostats) do
    local idxsp = 0
    local idxpv = 0
```

```

-- Select the thermostat by name and assign the idx of the temperature & setpoint devices
local idx = getThermostatIdx(domoticz, value.name)
if idx ~= {} then
    if (idx.sp > 0 and idx.pv > 0) then
        domoticz.devices(idx.sp).updateSetPoint(value.sp).silent()
        domoticz.devices(idx.pv).updateTemperature(value.pv).silent()
        domoticz.log(string.format(
            "Update Thermostat: %s Name=%s SP=%.2f (idx=%d), PV=%.2f (idx=%d).",
            idx.location, value.name, value.sp, idx.sp, value.pv, idx.pv))
    end
end
end

-- Handle the custom event triggered by recurring Homematic program script
return {
    on = { customEvents = { CUSTOMEVENT_NAME } },
    data = {},
    logging = {},
    execute = function(domoticz, item)
        if (item.isCustomEvent and item.isJSON) then
            -- {[{"timestamp"]}="12.05 20:24",
            -- [{"thermostats"]}=[{{"sp"}=4.5, {"idxsp"}=30, {"pv"}=20.5, {"idxpv"}=31}]}
            -- domoticz.log(item.json)
            domoticz.log(item.json.timestamp)
            -- set the thermostat devices Temperature & SSetPoint
            local thermostats = item.json.thermostats
            setThermostats(domoticz, thermostats)
        end
    end
}
}

```

## Domoticz Log

The logs shows the 2 thermostats updating their devices Temperature & Setpoint.

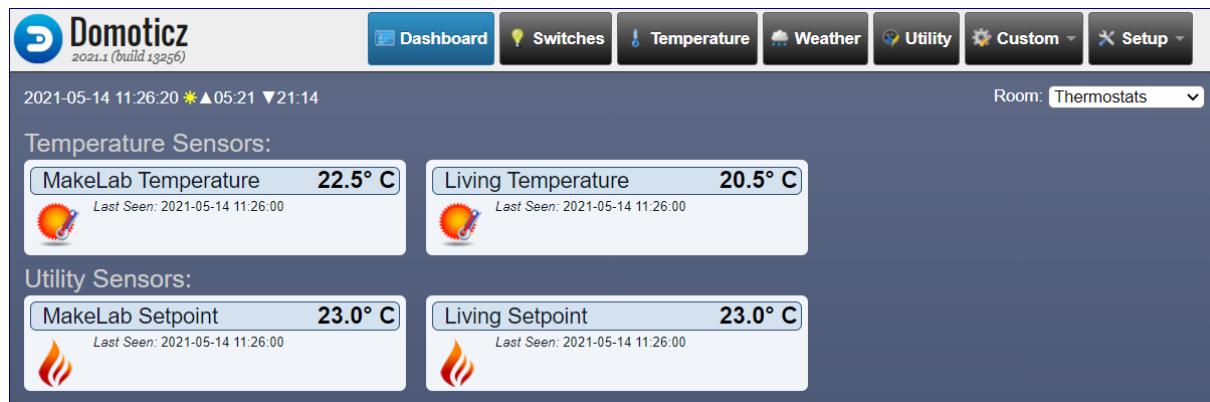
```

2021-05-14 11:22:00.797 Status: dzVents: Info: ----- Start internal script: Domoticz-Homematic-bulk-
update: Custom event: "domoticz-Homematic-bulk-update"
2021-05-14 11:22:00.798 Status: dzVents: Info: 14.05 11:22
2021-05-14 11:22:00.811 Status: dzVents: Info: Update Thermostat: makelab Name=HmIP-eTRV-2
000A18A9A64DAC:1 SP=22.50 (idx=30), PV=22.20 (idx=31).
2021-05-14 11:22:00.812 Status: dzVents: Info: Update Thermostat: wohnzimmer1 Name=HmIP-eTRV-B
00201A49952CB1:1 SP=23.00 (idx=89), PV=20.50 (idx=90).
2021-05-14 11:22:00.813 Status: dzVents: Info: ----- Finished Domoticz-Homematic-bulk-update

```

## **Domoticz Dashboard**

The dashboard shows the 2 thermostats for the locations MakeLab and Living.  
The devices are assigned to a roomplan Thermostats.



# Remote Homematic Script API

## Information

The **Remote Homematic Script API** enables to access/control Homematic actuators, sensors and more using:

- HTTP as a transport mechanism,
- XML (Extensible Markup Language) to encode the HTTP response.

This chapter explores how write & read data to & from the CCU using Homematic scripts submitted remotely via HTTP GET or POST requests.

The HTTP GET or POST request:

- requires the application tclrega.exe, which runs the Homematic scripts,
- contains the Homematic script code as data (POST) or an URL parameter with script code (GET),
- can be submitted from the CLI via curl or wget or from development tools, like Node-RED, JavaScript, Python and other.

Whilst exploring, the concept of running Homematic scripts remotely returning an XML object containing XML tags from the Homematic script vars with its value, is rather interesting.

Parsing the XML object enables to get the scripts vars for further use = see examples Domoticz, Node-RED or Python.



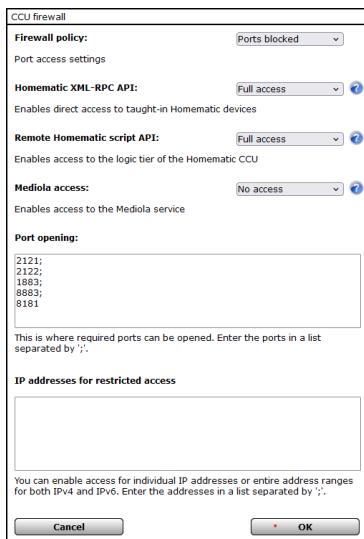
It is advised to keep the Homematic Remote Scripts simple to avoid CCU load or blocking the CCU.

The next script solutions explored are one-liners or have just a few lines to get or set data.

## Settings

To enable Remote Homematic Script API, change the Homematic settings:

- Homematic WebUI > Settings > Control Panel > Firewall > Set Full Access for Remote Script API (Enables access to the logic tier of the Homematic CCU).
- The port used is 8181.



## CLI Examples

These are some Remote Script examples submitted from Command Line Interface (CLI).

### Get System Variable

Get the value of the Homematic system variable named "DomoticzLog" assigned to the XML tag <value> which stores the result of the request.

Any XML tag can be set for the result of a Homematic function or code return.

#### Console Command

```
wget -q -O - "http://ccu-
ip:8181/tclreg.exe?value=dom.GetObject(ID_SYSTEM_VARIABLES).Get('DomoticzLog').Value()"
```

#### Console Output

```
<xml>
<exec>/tclreg.exe</exec>
<sessionId></sessionId>
<httpUserAgent>User-Agent: Wget/1.20.1 (linux-gnueabihf)</httpUserAgent>
<value>
  Status Display Update: 27.05 15:27:0K
</value>
</xml>
```

The XML tag <value> contains the system variable value.

Another example with the result stored in the variable "sysvarvalue".

```
wget -q -O - "http://ccu-
ip:8181/tclreg.exe?sysvarvalue=dom.GetObject(ID_SYSTEM_VARIABLES).Get('DomoticzLog').Value()"

<xml>
<sysvarvalue>
  Status Display Update: 27.05 15:27:0K
</sysvarvalue>
</xml>
```

## Post-Data WriteLine

Write the text (string) Hello World to the console using the post-data parameter.  
The WriteLine output is not stored in a tag in the returned XML tree.

### Note

Better to use script variable(s) because these are stored in XML tags.

### Console Command

```
wget -q -O - --post-data "WriteLine('Hello World');" "http://ccu-ip:8181/tclreg.exe"
```

### Console Output

```
Hello World
<xml>
<exec>/tclreg.exe</exec>
<sessionId></sessionId>
<httpUserAgent>User-Agent: Wget/1.20.1 (linux-gnueabihf)</httpUserAgent>
</xml>
```

## Post-Data Script Code

Define a short script code as post-data parameter.  
The WriteLine output is not stored in a variable in the returned XML tree, but the variable "count" is stored in the XML tag <count>.

### Console Command

```
wget --no-check-certificate -q -O - --post-data "var count = 1;WriteLine('Hallo Welt! '#count#' OK');"
"http://ccu-ip:8181/tclreg.exe"
```

### Console Output

```
Haloo Welt! 1 OK
<xml>
<exec>/tclreg.exe</exec>
<sessionId></sessionId><httpUserAgent>User-Agent: Wget/1.20.1 (linux-gnueabihf)</httpUserAgent>
<count>1</count>
</xml>
```

### Note

If variable "count" is changed to counter, the XML tag <counter> holds the counter value 1.

## Domoticz Examples

Domoticz examples which get or set Homematic devices values or state by using Domoticz Automation Events scripted in dzVents.

Info: [Read](#) about dzVents - next generation Lua scripting.

### Concept

The dzVents script is triggered by, for example, the state change of a Domoticz device or by a timer event.

The url, containing the Homematic Script, is defined, either as a one-line-script (HTTP GET, for simple scripts) or as multiple-line-script (HTTP POST, for more advanced scripts).

The url is submitted, containing tclreg.exe and the dzVents scripts waits for the response from the CCU.

The HTTP response received is an XML object containing tags which hold the Homematic scripts variables.

The XML tags are parsed, using [XPath](#), and update Domoticz devices or to get an acknowledge if the Homematic script has been executed ok remotely.

Next simple examples for a Homematic one-line and multi-line script in a Domoticz Automation Event dzVents with the openURL snippet.

**Recommend** using Multi-Line-Scripts as better readable, error handling and advanced solutions.

### One-Line-Script

```
-- Define the Remote Homematic Script URL. Ensure to escape space to %20 (used in the channelname).
local URL_HMAPI = 'http://ccu-ip:8181/tclreg.exe?result=dom.GetObject("HmIP-eTRV-
2%20000A18A9A64DAC:1").DPByHssDP("ACTUAL_TEMPERATURE").Value()'

-- Example HTTP Request
local url = URL_HMAPI
-- Submit the HTTP GET request = Remote Homematic Script
domoticz.openURL({
  url = url,
  headers = { ['content-type'] = 'application/x-www-form-urlencoded' },
  method = 'GET', callback = 'HMGETACTUALTEMPERATURE'
})
```

### Multi-Line-Script

```
-- Define the Homematic script as long string. Do not encode, i.e., leave spaces etc.
local script = [[! Homematic script
  var obj = dom.GetObject("DomoticzLog");
  if (obj) {var value = obj.Value();}
]

Local URL_HMAPI = 'http://ccu-ip:8181/tclreg.exe'

-- Example HTTP Request
local url = URL_HMAPI
-- Submit the HTTP GET request = Remote Homematic Script
domoticz.openURL({
  url = url,
  headers = { ['content-type'] = 'application/x-www-form-urlencoded' },
  postData = script, method = 'POST', callback = 'HMGETSYSVAR'
}])
```

## Get System Variable

Get the value of the system variable named “HeatingUnitGasMeter” assigned to the XML tag <value>.

```
-- homematic_get_sysvar.dzvents
-- Get, in regular intervals, the value of a system variable via Remote Homematic Script API.
-- The Homematic script is a long string which is assigned to the HTTP request postdata parameter.
]]--

-- Homematic
-- Define the Remote Homematic Script URL.
local URL_HMAPI = 'http://ccu-ip:8181/tclreg.exe'

-- HTTP Response - MUST be UNIQUE across all events
local RES_HMAPI = "GETSYSVAR"

-- Define the Homematic sysvar name
local SYS_VAR = "HeatingUnitGasMeter"

-- Define the Homematic script as a long string.
-- Do not encode, i.e. leave spaces etc.
-- The sysvar name is a placeholder which is replaced prior openURL.
local script = [[! Homematic script
    var obj = dom.GetObject("#SYSVAR#");
    if (obj) {var value = obj.Value();}
]]
-- Assign the IDX for the sysvar to be assigned to
local IDX_HEATINGUNIT_GASMETER = 8

-- Timer rule
local TIMER_RULE = "every minute"
-- local TIMER_RULE = "every 5 minutes"

return {
  on = {
    timer = { TIMER_RULE },
    httpResponses = { RES_HMAPI }
  },
  logging = { level = domoticz.LOG_INFO, marker = RES_HMAPI, },
  execute = function(domoticz, item)

    -- The timer is triggered to get the sysvar from Homematic.
    if (item.isTimer) then
      local url = URL_HMAPI
      -- For Debug log the url and the script
      -- domoticz.log(url)
      script = string.gsub(script, "#SYSVAR#", SYS_VAR)
      domoticz.log(script)
      -- Submit the HTTP GET request = Remote Homematic Script
      domoticz.openURL({
        url = url,
        headers = { ['content-type'] = 'application/x-www-form-urlencoded' },
        postData = script,
        method = 'POST',
        callback = RES_HMAPI,
      })
    end

    -- Handle the HTTP Remote Homematic Script API response
    if (item.isHTTPResponse) then
      -- domoticz.log(item)
      if (item.ok) then
        -- The item data contains the XML response from the CCU
        domoticz.log(item.data)
        -- <xml><exec>/tclreg.exe</exec><sessionId></sessionId><httpUserAgent>User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/603.37 (KHTML, like Gecko) Chrome/54.0.30045.0 Safari/603.38</httpUserAgent><obj>HeatingUnitGasMeter</obj><value>672.52000</value></xml>
        -- Get the text of the sysvar <value> tag: value=OK, null=ERROR
        local value = domoticz_applyXPath(item.data,'/xml/value/text()')
        domoticz.log(string.format('Sysvar=%s, value=%s', SYS_VAR, value))
      end
    end
  end
}
```

```

    -- Update the domoticz device = convert m3 to dm3.
    domoticz.devices(IDX_HEATINGUNIT_GASMETER).updateCounter(value * 100);
else
    domoticz.log('There was a problem handling the request', domoticz.LOG_ERROR)
    domoticz.log(item, domoticz.LOG_ERROR)
end
end
end
}
}

```

## Domoticz Log

```

2023-02-03 10:06:00.439 Status: dzVents: Info: GETSYSVAR: ----- Start internal script:
homematic_get_sysvar:, trigger: "every minute"
2023-02-03 10:06:00.439 Status: dzVents: Info: GETSYSVAR: ! Homematic script
2023-02-03 10:06:00.439 var obj = dom.GetObject("HeatingUnitGasMeter");
2023-02-03 10:06:00.439 if (obj) {var value = obj.Value();}
2023-02-03 10:06:00.439
2023-02-03 10:06:00.439 Status: dzVents: Info: GETSYSVAR: ----- Finished homematic_get_sysvar
2023-02-03 10:06:00.440 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
2023-02-03 10:06:00.519 Status: dzVents: Info: Handling httpResponse-events for: "GETSYSVAR"
2023-02-03 10:06:00.520 Status: dzVents: Info: GETSYSVAR: ----- Start internal script:
homematic_get_sysvar: HTTPResponse: "GETSYSVAR"
2023-02-03 10:06:00.524 Status: dzVents: Info: GETSYSVAR:
<xml><exec>/tclrega.exe</exec><sessionId></sessionId><httpUserAgent>User-Agent: Mozilla/5.0 (Windows NT
10.0; WOW64) AppleWebKit/603.37 (KHTML, like Gecko) Chrome/54.0.30045.0
Safari/603.38</httpUserAgent><obj>HeatingUnitGasMeter</obj><value>672.520000</value></xml>
2023-02-03 10:06:00.524 Status: dzVents: Info: GETSYSVAR: Sysvar=HeatingUnitGasMeter, value=672.520000
2023-02-03 10:06:00.542 Status: dzVents: Info: GETSYSVAR: ----- Finished homematic_get_sysvar
2023-02-03 10:06:00.543 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua

```

## Domoticz Device



## Get Thermostats Temperature

Get the actual temperature value of the two Homematic thermostat devices and update the related Domoticz Temperature devices.

```
-- homematic_thermostats_temperature.dzvents
-- Get, in regular intervals, the temperature of thermostat devices via Remote Homematic Script API.
-- The object channel for a device is obtained from the Homematic WebUI Devices list for the selected
thermostat.
-- The datapoint is ACTUAL_TEMPERATURE.
-- The Homematic script is a long string which is assigned to the HTTP request postdata parameter.

-- Homematic
-- Define the Remote Homematic Script URL.
local URL_HMAPI = 'http://ccu-ip:8181/tclregae.exe'

-- HTTP Response - MUST be UNIQUE across all events
local RES_HMAPI = "THERMOSTATS-TEMPERATURE"

-- Define the Homematic script as long string. Do not encode, i.e., leave spaces etc.
local script = [[! Homematic script
    ! Get the temperature of thermostat devices
    var temp_makelab = dom.GetObject("HmIP-eTRV-2
000A18A9A64DAC:1").DPByHssDP("ACTUAL_TEMPERATURE").Value();
    var temp_bad = dom.GetObject("HmIP-eTRV-B
002018A99D2097:1").DPByHssDP("ACTUAL_TEMPERATURE").Value();
    ]]

-- Domoticz
local IDX_TEMPERATURE_MAKELAB = 10
local IDX_TEMPERATURE_BAD = 11
-- Add more ...

-- Timer rule
-- local TIMER_RULE = "every minute"
local TIMER_RULE = "every 5 minutes"

return {
  on = {
    timer = { TIMER_RULE },
    httpResponses = { RES_HMAPI }
  },
  logging = {
    level = domoticz.LOG_INFO, marker = RES_HMAPI,
  },
  execute = function(domoticz, item)

    -- The timer is triggered to get the device temperature.
    if (item.isTimer) then
      local url = URL_HMAPI
      -- For Debug log the url and the script
      -- domoticz.log(url)
      domoticz.log(script)
      -- Submit the HTTP GET request = Remote Homematic Script
      domoticz.openURL({
        url = url,
        headers = { ['content-type'] = 'application/x-www-form-urlencoded' },
        postData = script,
        method = 'POST',
        callback = RES_HMAPI,
      })
    end

    -- Handle the HTTP Remote Homematic Script API response
    if (item.isHTTPResponse) then
      -- domoticz.log(item)
      if (item.ok) then
        -- The item data contains the XML response from the CCU
        domoticz.log(item.data)
        -- <xml><exec>/tclregae.exe</exec><sessionId></sessionId><httpUserAgent>User-Agent:
Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/603.37 (KHTML, like Gecko) Chrome/54.0.30045.0
Safari/603.38</httpUserAgent><temp_makelab>13.300000</temp_makelab><temp_bad>19.000000</temp_bad></xml>

```

```
-- Get the text of the temperature tags: NN.N=OK, null=ERROR
local temp_makelab = domoticz_applyXPath(item.data,'/xml/temp_makelab/text()')
-- Update the temperature device
domoticz.devices(IDX_TEMPERATURE_MAKELAB).updateTemperature(temp_makelab);

local temp_bad = domoticz_applyXPath(item.data,'/xml/temp_bad/text()')
domoticz.devices(IDX_TEMPERATURE_BAD).updateTemperature(temp_bad);

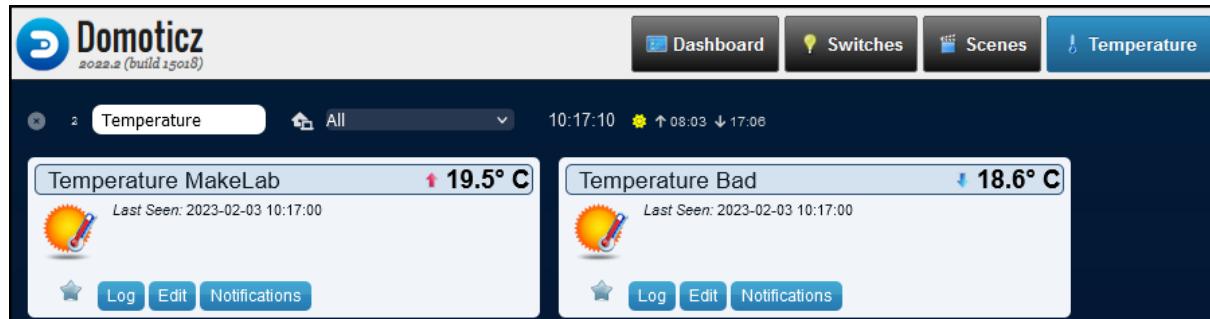
domoticz.log(string.format('Temperatures MakeLab: %.1f °C, Bad: %.1f °C',
tonumber(temp_makelab), tonumber(temp_bad)))
else
    domoticz.log('There was a problem handling the request', domoticz.LOG_ERROR)
    domoticz.log(item, domoticz.LOG_ERROR)
end
end

end
}
```

## Domoticz Log

```
2023-02-02 19:05:01.073 Status: dzVents: Info: THERMOSTATS-TEMPERATURE: ----- Start internal script:
homematic_thermostats_temperature:, trigger: "every 5 minutes"
2023-02-02 19:05:01.073 Status: dzVents: Info: THERMOSTATS-TEMPERATURE: ! Homematic script
2023-02-02 19:05:01.073 ! Get the temperature of thermostat devices
2023-02-02 19:05:01.073 var temp_makelab = dom.GetObject("HmIP-eTRV-2
000A18A9A64DAC:1").DPByHssDP("ACTUAL_TEMPERATURE").Value();
2023-02-02 19:05:01.073 var temp_bad = dom.GetObject("HmIP-eTRV-B
002018A99D2097:1").DPByHssDP("ACTUAL_TEMPERATURE").Value();
2023-02-02 19:05:01.073
2023-02-02 19:05:01.073 Status: dzVents: Info: THERMOSTATS-TEMPERATURE: ----- Finished
homematic_thermostats_temperature
2023-02-02 19:05:01.074 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
2023-02-02 19:05:01.168 Status: dzVents: Info: Handling httpResponse-events for: "THERMOSTATS-
TEMPERATURE"
2023-02-02 19:05:01.169 Status: dzVents: Info: THERMOSTATS-TEMPERATURE: ----- Start internal script:
homematic_thermostats_temperature: HTTPResponse: "THERMOSTATS-TEMPERATURE"
2023-02-02 19:05:01.173 Status: dzVents: Info: THERMOSTATS-TEMPERATURE:
<xml><exec>/tclrega.exe</exec><sessionId></sessionId><httpUserAgent>User-Agent: Mozilla/5.0 (Windows NT
10.0; WOW64) AppleWebKit/603.37 (KHTML, like Gecko) Chrome/54.0.30045.0
Safari/603.38</httpUserAgent><temp_makelab>13.600000</temp_makelab><temp_bad>18.700000</temp_bad></xml>
2023-02-02 19:05:01.193 Status: dzVents: Info: THERMOSTATS-TEMPERATURE: Temperatures MakeLab: 13.6 °C,
Bad: 18.7 °C
2023-02-02 19:05:01.193 Status: dzVents: Info: THERMOSTATS-TEMPERATURE: ----- Finished
homematic_thermostats_temperature
2023-02-02 19:05:01.193 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
```

## Domoticz Devices



Domoticz GUI > Tab Temperature > Filter on string Temperature.

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
10	VirtualSensors	1405A	1	Temperature MakeLab	Temp	LaCrosse TX3	19.8 C
11	VirtualSensors	1405B	1	Temperature Bad	Temp	LaCrosse TX3	18.3 C

Domoticz GUI > Settings > Devices > Filter on string Temperature.

## Set Thermostat Setpoint

Set the setpoint of a Homematic thermostat device by using a Domoticz Thermostat device.

### Homematic Script One-Liner

This example uses a Homematic script one-liner HTTP GET request executed by a Domoticz Automation Event dzVents.

```
-- homematic_thermostat_setpoint.dzvents
-- Change the setpoint of a TRV-2 device via Remote Homematic Script API.
-- The object channel is obtained from the Homematic WebUI Devices list for the selected thermostat.
-- The datapoint is SET_POINT_TEMPERATURE.
-- The Homematic script is a one-liner with the channel state used to set the new setpoint.
-- Hint: If the channel state function (.State()) has an empty string as argument, the current setpoint
is returned or use function .Value().

-- Define the IDX of the thermostat device
local IDX_THERMOSTAT_MAKELAB = 9

-- Define Remote Homematic Script URL. Ensure escape space to %20 (used in the channelname).
-- The placeholder #SP# sets the setpoint.
local URL_HMAPI = 'http://ccu-ip:8181/tclreg.exe?result=dom.GetObject("HmIP-eTRV-
2%2000A18A9A64DAC:1").DPByHssDP("SET_POINT_TEMPERATURE").State(#SP#)'
-- HTTP Response - MUST be UNIQUE across all events
local RES_HMAPI = "THERMOSTAT-MAKELAB"

return {
  on = {
    devices = { IDX_THERMOSTAT_MAKELAB },
    httpResponses = { RES_HMAPI }
  },
  logging = { level = domoticz.LOG_INFO, marker = RES_HMAPI, },
  execute = function(domoticz, item)

    -- The device setpoint has changed. Sent HTTP request to CCU.
    if (item.isDevice) then
      -- Replace the SP placeholder with the new setpoint from the device item
      local url = string.gsub(URL_HMAPI, "#SP#", item.setPoint)
      -- Just as info log the url
      domoticz.log(url)
      -- Submit the HTTP GET request = Remote Homematic Script
      domoticz.openURL({
        url = url,
        headers = { ['content-type'] = 'application/x-www-form-urlencoded' },
        method = 'GET',
        callback = RES_HMAPI,
      })
    end

    -- Handle the HTTP Remote Homematic Script API response
    if (item.isHTTPResponse) then
      -- domoticz.log(item)
      if (item.ok) then
        -- The item data contains the XML response from the CCU
        -- domoticz.log(item.data)
        -- Get the setpoint tag
        -- <xml><exec>/tclreg.exe</exec><sessionId></sessionId><httpUserAgent>User-Agent: Mozilla/5.0
        (Windows NT 10.0; WOW64) AppleWebKit/603.37 (KHTML, like Gecko) Chrome/54.0.30045.0
        Safari/603.38</httpUserAgent><result>true</result></xml>
        -- Get the text of the result tag: true=OK, null=ERROR
        local result = domoticz_applyXPath(item.data,'/xml/result/text()')
        domoticz.log(string.format('Setpoint changed. Result: %s', result))
      else
        domoticz.log('There was a problem handling the request', domoticz.LOG_ERROR)
        domoticz.log(item, domoticz.LOG_ERROR)
      end
    end
  end
}
```

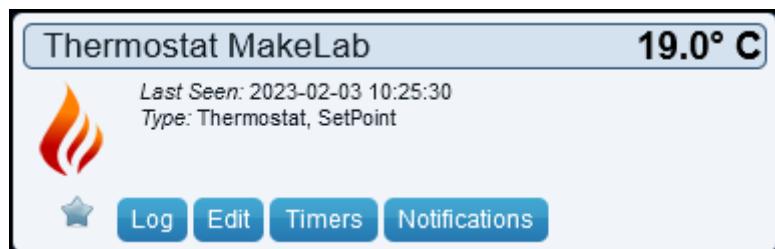
## Domoticz Log

```

2023-02-01 18:52:03.125 VirtualSensors: Thermostat (Thermostat-MakeLab)
2023-02-01 18:52:03.122 Status: User: admin initiated a SetPoint command
2023-02-01 18:52:03.219 Status: dzVents: Info: Handling events for: "Thermostat-MakeLab", value: "19.00"
2023-02-01 18:52:03.219 Status: dzVents: Info: THERMOSTAT-MAKELAB: ----- Start internal script: homematic_thermostat_setpoint: Device: "Thermostat-MakeLab (VirtualSensors)", Index: 9
2023-02-01 18:52:03.219 Status: dzVents: Info: THERMOSTAT-MAKELAB: http://ccu-ip:8181/tclreg.exe?result=dom.GetObject("HmIP-eTRV-2%20000A18A9A64DAC:1").DPByHssDP("SET_POINT_TEMPERATURE").State(6.0)
2023-02-01 18:52:03.219 Status: dzVents: Info: THERMOSTAT-MAKELAB: ----- Finished homematic_thermostat_setpoint
2023-02-01 18:52:03.220 Status: EventSystem: Script event triggered: /home/pi/domoticz/dzVents/runtime/dzVents.lua
2023-02-01 18:52:05.383 Status: dzVents: Info: Handling httpResponse-events for: "THERMOSTAT-MAKELAB"
2023-02-01 18:52:05.383 Status: dzVents: Info: THERMOSTAT-MAKELAB: ----- Start internal script: homematic_thermostat_setpoint: HttpResponseMessage: "THERMOSTAT-MAKELAB"
2023-02-01 18:52:05.388 Status: dzVents: Info: THERMOSTAT-MAKELAB: Setpoint changed. Result: true
2023-02-01 18:52:05.389 Status: dzVents: Info: THERMOSTAT-MAKELAB: ----- Finished homematic_thermostat_setpoint

```

## Domoticz Device



## Homematic Script Multi-Liner

This example uses a Homematic script multi-liner HTTP POST request executed by a Domoticz Automation Event dzVents.

```

local IDX_THERMOSTAT_MAKELAB = 9
local URL_HMAPI = 'http://ccu-ip:8181/tclreg.exe'
local RES_HMAPI = "THERMOSTAT-MAKELAB"
local script = [[! Homematic script Set the setpoint of a thermostat device
    var result = dom.GetObject("HmIP-eTRV-2
000A18A9A64DAC:1").DPByHssDP("SET_POINT_TEMPERATURE").State(#SP#);
]]

return {
    on = { devices = { IDX_THERMOSTAT_MAKELAB }, httpResponses = { RES_HMAPI } },
    logging = { level = domoticz.LOG_INFO, marker = RES_HMAPI, },
    execute = function(domoticz, item)
        if (item.isDevice) then
            script = string.gsub(script, "#SP#", item.setPoint)
            domoticz.openURL({
                url = URL_HMAPI,
                headers = { ['content-type'] = 'application/x-www-form-urlencoded' },
                method = 'POST', postData = script, callback = RES_HMAPI,
            })
        end
        if (item.isHTTPResponse) then
            if (item.ok) then
                local result = domoticz_applyXPath(item.data,'/xml/result/text()')
                domoticz.log(string.format('Setpoint changed. Result: %s', result))
            else
                domoticz.log('There was a problem handling the request', domoticz.LOG_ERROR)
                domoticz.log(item, domoticz.LOG_ERROR)
            end
        end
    end
}

```

```
}
```

**Note**

Instead using a variable for the script and replace the placeholder for the setpoint, it can also be done via a function.

```
local function setscript(domoticz, setpoint)
    local script = [[! Homematic script
        ! Set the setpoint of a thermostat device
        var result = dom.GetObject("HmIP-eTRV-2
000A18A9A64DAC:1").DPByHssDP("SET_POINT_TEMPERATURE").State(#SP#);
    ]]
    script = string.gsub(script, "#SP#", setpoint)
    domoticz.log(script)
    return script
end
```

The openURL function with the function setscript as postdata. The device item setPoint property is set as the function argument.

```
-- Submit the HTTP POST request = Remote Homematic Script
domoticz.openURL({
    url = url,
    headers = { ['content-type'] = 'application/x-www-form-urlencoded' },
    method = 'POST',
    postData = setscript(domoticz, item.setPoint),
    callback = RES_HMAPI,
})
```

## Set Domoticz Thermostat Widget

Get the actual temperature, setpoint and level value of an Homematic thermostat device and update the related Domoticz Thermostat device.

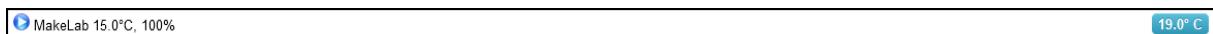
Beside the 3 properties, more can be added like voltage or low battery state.

### Domoticz Thermostat Widget

The device name contains the thermostat location, actual temperature, and the level (percentage valve opened).



### Domoticz Thermostat Mobile Dashboard



### Domoticz Automation Event dzVents

The dzVents script runs every 5 minutes (for tests every minute).

The script uses a function defined in the shared helper global\_data.lua.

#### Script global\_data.lua

```
return {
    helpers = {
        hms_get_thermostat = function(devname)
            local script = [[! Homematic script
                var obj = dom.GetObject("#NAME#");
                if (obj) {
                    var temperature = obj.DPByHssDP("ACTUAL_TEMPERATURE").Value();
                    var setpoint = obj.DPByHssDP("SET_POINT_TEMPERATURE").Value();
                    var level = obj.DPByHssDP("LEVEL").Value();
                }
            ]]
            return string.gsub(script, "#NAME#", devname)
        end
    }
}
```

#### Script homematic\_thermostat.dzvents

```
-- homematic_thermostat.dzvents
-- Get, regular intervals, temperature (°C), setpoint (°C), level (0-100%) TRV device.
-- The object channel is obtained from the Homematic WebUI Devices list for the selected thermostat.
-- Datapoints ACTUAL_TEMPERATURE, SET_POINT_TEMPERATURE, LEVEL
-- Define the IDX of the thermostat device
local IDX_THERMOSTAT_MAKELAB = 9
local NAME_THERMOSTAT_MAKELAB = "MakeLab"
-- Define the Remote Homematic Script URL.
local URL_HMAPI = 'http://192.168.1.70:8181/tclreg.exe'
-- HTTP Response - MUST be UNIQUE across all events.
local RES_HMAPI = "THERMOSTAT-MAKELAB"
-- Define the Homematic thermostat device name used for the function to get thermostat properties
local THERMOSTAT = "HmIP-eTRV-2 000A18A9A64DAC:1"
-- Timer rule
local TIMER_RULE = "every minute"
-- local TIMER_RULE = "every 5 minutes"

return {
    on = { timer = { TIMER_RULE }, httpResponses = { RES_HMAPI } },
    logging = { level = domoticz.LOG_INFO, marker = RES_HMAPI, },
```

```

execute = function(domoticz, item)

if (item.isTimer) then
    domoticz.openURL({
        url = URL_HMAPI,
        headers = { ['content-type'] = 'application/x-www-form-urlencoded' },
        postData = domoticz.helpers.hms_get_thermostat(THERMOSTAT),
        method = 'POST', callback = RES_HMAPI,
    })
end

if (item.isHTTPResponse) then
    if (item.ok) then
        local temperature = tonumber(domoticz_applyXPath(item.data,'/xml/temperature/text()'))
        local setpoint = tonumber(domoticz_applyXPath(item.data,'/xml/setpoint/text()'))
        local level = tonumber(domoticz_applyXPath(item.data,'/xml/level/text())) * 100
        domoticz.log(string.format('temperature=%1f, setpoint=%1f, level=%0f', temperature, setpoint, level))
        -- Update the Domoticz thermostat device name and setpoint (do silent to avoid endless loop)
        local name = string.format('%s %.1f°C, %.0f%%', NAME_THERMOSTAT_MAKELAB, temperature, level)
        domoticz.devices(IDX_THERMOSTAT_MAKELAB).rename(name)
        domoticz.devices(IDX_THERMOSTAT_MAKELAB).updateSetPoint(setpoint).silent()
    else
        domoticz.log('There was a problem handling the request', domoticz.LOG_ERROR)
        domoticz.log(item, domoticz.LOG_ERROR)
    end
end
end
}
}

```

## Domoticz Log

```

2023-02-04 11:45:00.290 Status: dzVents: Info: THERMOSTAT-MAKELAB: ----- Start internal script:
homematic_thermostat:, trigger: "every minute"
2023-02-04 11:45:00.290 Status: dzVents: Info: THERMOSTAT-MAKELAB: http://192.168.1.70:8181/tclreg.exe
2023-02-04 11:45:00.290 Status: dzVents: Info: THERMOSTAT-MAKELAB: ----- Finished homematic_thermostat
2023-02-04 11:45:00.291 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
2023-02-04 11:45:00.387 Status: dzVents: Info: Handling httpResponse-events for: "THERMOSTAT-MAKELAB"
2023-02-04 11:45:00.387 Status: dzVents: Info: THERMOSTAT-MAKELAB: ----- Start internal script:
homematic_thermostat: HTTPResponse: "THERMOSTAT-MAKELAB"
2023-02-04 11:45:00.391 Status: dzVents: Info: THERMOSTAT-MAKELAB:
<xml><exec>/tclreg.exe</exec><sessionId></sessionId><httpUserAgent>User-Agent: Mozilla/5.0 (Windows NT
10.0; WOW64) AppleWebKit/601.36 (KHTML, like Gecko) Chrome/53.0.50337.0
Safari/601.36</httpUserAgent><obj>HmIP-eTRV-2
000A18A9A64DAC:1</obj><temperature>14.600000</temperature><setpoint>19.000000</setpoint><level>1.000000
</level></xml>
2023-02-04 11:45:00.392 Status: dzVents: Info: THERMOSTAT-MAKELAB: temperature=14.6, setpoint=19.0,
level=100
2023-02-04 11:45:00.409 Status: dzVents: Info: THERMOSTAT-MAKELAB: ----- Finished homematic_thermostat

```

## Node-RED Examples

A few Node-RED examples using **Remote Homematic Script Calls** send via the HTTP Request node (with tclregae.exe Remote API).

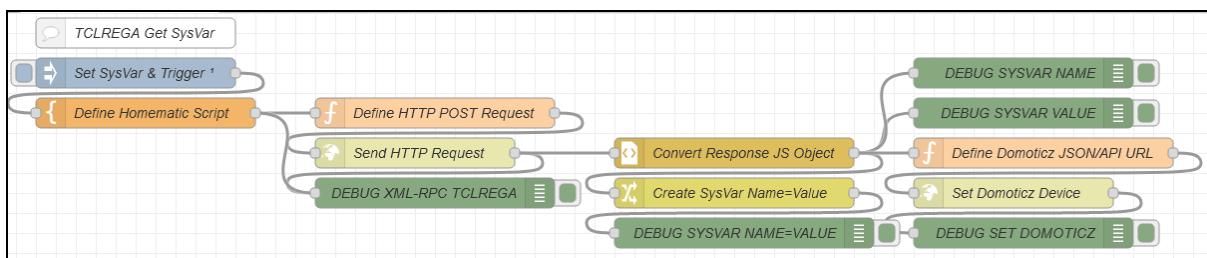
Node-RED could act as an interface between Homematic and Domoticz vv.

### Get System Variable

#### Purpose

Get the value of the Homematic system variable HeatingUnitGasMeter and update the Domoticz Device (idx=8, RFXMeter).

#### Node-RED Flow



#### Node-RED Nodes

- **Inject** node set the System Variable name as string (HeatingUnitGasMeter) and triggers the flow.
- **Template** node defines the Homematic script – test first with the Homematic addon Script Executor.
- **Function** node defines the HTTP POST data to get the sysvar value.
- **HTTP Request** node
  - sets the HTTP method POST (mandatory by Node-RED to define in this node and not in the function node as msg.method) and
  - submits the request to Homematic.
- **Debug** node logs the HTTP response.
- **XML** node to convert HTTP response XML tree to Javascript object.
- **Debug** node logs the system variable value.
- **Function** node defines the Domoticz HTTP JSON/API request to update the Domoticz device.
- **HTTP Request** node submits the URL to Domoticz.
- **Debug** node to log the Domoticz HTTP response as JSON object: {"status" : "ERR"} or {"status" : "OK", "title" : "Update Device"}.

## Template Node Define Homematic Script

The screenshot shows the configuration of a Template Node in Domoticz. The node is named "Define Homematic Script" and is set to use the "msg.payload" property. The template code is a Mustache template that defines a sysvar from msg.payload and retrieves its value if it exists.

```

1 ! Define the Homematic script.
2 ! Test first with the Homematic Addon Script Executor.
3 ! Use var to get XML tags; Comment out WriteLine (only for debug/testing).
4 ! Set the sysvar, given as msg.payload in "".
5 ! Use format Mustache template to be able to use the msg.payload.
6
7 ! Define the sysvar in "" from the msg.payload. Use payload instead msg.payload.
8 var SYSVAR = "{{payload}}";
9
10 ! Get the sysvar as object to access properties.
11 var sysvarobj = dom.GetObject(SYSVAR);
12
13 ! If there is a sysvar object, get the value.
14 if (sysvarobj) {
15     var value = sysvarobj.Value();
16 }
17
18

```

The "Format" dropdown is set to "Mustache template" and the "Output as" dropdown is set to "Plain text".

Important to set the format to Mustache template, to be able to use the msg.payload (defined as {{payload}}).

## Function Node Define HTTP POST Request

```

// Define the HTTP POST request using tclrega.exe

// Define the script
let script = msg.payload;

// Define the HTTP headers
let headers = {};
headers["Content-Length"] = script.length;
headers["Content-Type"] = "application/x-www-form-urlencoded";
msg.headers = headers;

// The HTTP method POST needs to be set in the httprequest node
// msg.method = "POST";
// The port is required else HTTP 403 error
msg.url = "http://ccu-ip:8181/tclrega.exe";
msg.payload = script;

// Return the msg
return msg;

```

## Debug Output Nodes XML-RPC TCLREGA and SYSVAR VALUE

```
<xml>
<exec>/tclregga.exe</exec>
<sessionId></sessionId><httpUserAgent>user-agent: got
(https://github.com/sindresorhus/got)</httpUserAgent>
<SYSVAR>HeatingUnitGasMeter</SYSVAR>
<sysvarobj>HeatingUnitGasMeter</sysvarobj>
<value>672.420000</value>
</xml>
```

672.140000

The XML tag <value> contains the system variable value.

The other vars defined in the Homematic script are also stored as XML tag.

## Function Node Define Domoticz JSON/API URL

```
// Domoticz Device Counter URL to update the device
let domurl = "http://domoticz-ip:8080/json.htm?type=command&param=udevice&idx=#IDX##&svalue=#SVALUE#";

// Domoticz device idx
let IDX = 8;

// Counter value in m3
let counter = msg.payload.xml.value[0] * 100;

// Replace the placeholder in the url
domurl = domurl.replace("#IDX#", IDX);
domurl = domurl.replace("#SVALUE#", counter);

// Assign the url
msg.url = domurl;
node.warn(msg.url)

// Return to the HTTP request node
return msg;
```

## Node-RED Flow Source

```
[{"id": "2037a45990be96e4", "type": "tab", "label": "TCLREGA Get SysVar", "disabled": false, "info": "", "env": []}, {"id": "fcd26fe2.c1f278", "type": "inject", "z": "2037a45990be96e4", "name": "Set SysVar & Trigger", "props": [{"p": "payload"}], "repeat": "", "crontab": "", "once": true, "onceDelay": 0.1, "topic": "", "payload": "HeatingUnitGasMeter", "payloadType": "str", "x": 140, "y": 60, "wires": [{"8eaa049a980f350a"}]}, {"id": "6ab91ffb.3f7c88", "type": "function", "z": "2037a45990be96e4", "name": "Define HTTP POST Request", "func": "// Define the HTTP POST request using tclregaa.exe\n// Define the script\nlet script = msg.payload;\n\n// Define the HTTP headers\nlet headers = {};\nheaders[\"Content-Length\"] = script.length;\nheaders[\"Content-Type\"] = \"application/x-www-form-urlencoded\";\nnmsg.headers = headers;\n\n// The HTTP method POST needs to be set in the httprequest node\n// msg.method = \"POST\";\n\n// The port is required else HTTP 403 error\nnmsg.url = \"http://ccu-ip:8181/tclregaa.exe\";\nnmsg.payload = script;\n\n// Return the msg\nreturn msg;\n\n// Example Response\nHTTP POST request\n/*\n<xml><exec>tclregaa.exe</exec><sessionId></sessionId><httpUserAgent>user-agent: got\n(https://github.com/sindresorhus/got)</httpUserAgent><SYSVAR>HeatingUnitGasMeter</SYSVAR><sysvarobj>HeatingUnitGasMeter</sysvarobj><value>672.420000</value></xml>\n*, \"outputs\": 1, \"noerr\": 0, \"initialize\": \"\", \"finalize\": \"\", \"libs\": [], \"x\": 440, \"y\": 100, \"wires\": [[\"5e083760.e8d558\"]]}, {"id": "5e083760.e8d558", "type": "http request", "z": "2037a45990be96e4", "name": "Send HTTP Request", "method": "POST", "ret": "txt", "paytoqs": "ignore", "url": "", "tls": "", "persist": false, "proxy": "", "insecureHTTPParser": false, "authType": "", "senderr": false, "headers": [], "x": 420, "y": 140, "wires": [{"f4819927.093148", "e37f04132e81aa2d"}]}, {"id": "f4819927.093148", "type": "debug", "z": "2037a45990be96e4", "name": "DEBUG XML-RPC"}, {"id": "e37f04132e81aa2d", "type": "TCLREGA", "active": true, "tosidebar": true, "console": false, "tostatus": false, "complete": "payload", "targetType": "msg", "statusVal": "", "statusType": "auto", "x": 440, "y": 180, "wires": []}, {"id": "305cb4ff5b38e077", "type": "function", "z": "2037a45990be96e4", "name": "Convert Response JS Object", "property": "payload", "attr": "$", "chr": "_", "x": 740, "y": 140, "wires": [{"442fc7484eb00846", "305cb4ff5b38e077", "b1c90face81ebfe1", "bb808a196ab1ae0b"}]}, {"id": "442fc7484eb00846", "type": "debug", "z": "2037a45990be96e4", "name": "DEBUG SYSVAR"}, {"id": "305cb4ff5b38e077", "type": "function", "z": "2037a45990be96e4", "name": "DEBUG SYSVAR"}, {"id": "b1c90face81ebfe1", "type": "change", "z": "2037a45990be96e4", "name": "Create SysVar Name=Value", "rules": [{"t": "set", "p": "payload", "pt": "msg", "to": "payload.xml.sysvarobj[0] & \"/=\\\" & payload.xml.value[0]", "tot": "jsonata"}, {"action": "", "property": "", "from": "", "to": "", "reg": false, "x": 740, "y": 180, "wires": [{"f1b80366232d048e"}]}]}, {"id": "f1b80366232d048e", "type": "debug", "z": "2037a45990be96e4", "name": "DEBUG SYSVAR"}, {"id": "b701b28d4aa53f0c", "type": "comment", "z": "2037a45990be96e4", "name": "TCLREGA Get SysVar", "info": "Get Homematic System Variable.\n\nThe Inject node sets the system variable Name as msg.payload used in the Template node.\n\nHTTP Result XML"}, {"id": "8eaa049a980f350a", "type": "template", "z": "2037a45990be96e4", "name": "Define Homematic Script", "field": "payload", "fieldType": "msg", "format": "text", "syntax": "mustache", "template": "# Define the Homematic script.\n\nTest first with the Homematic Addon Script Executor.\n\nUse var to get XML tags; Comment out WriteLine (only for debug/testing).\n\nSet the sysvar, given as msg.payload in \"\".\n\nUse format Mustache template to be able to use the msg.payload.\n\nDefine the sysvar in \"\" from the msg.payload. Use payload instead msg.payload.\n\nvar SYSVAR = \"{{payload}}\";\n\nGet the sysvar as object to access properties.\n\nvar sysvarobj = dom.GetObject(SYSVAR);\n\nIf there is a sysvar object, get the value.\n\nif (sysvarobj) {\n    var value = sysvarobj.Value();\n}\n\noutput: \"str\", \"x\": 150, \"y\": 100, \"wires\": [[\"6ab91ffb.3f7c88\", \"f4819927.093148\"]]}, {"id": "bb808a196ab1ae0b", "type": "function", "z": "2037a45990be96e4", "name": "Define Domoticz JSON/API URL", "func": "# Domoticz Device Counter URL to update the device\n\nlet domurl = \"http://domoticz-ip:8080/json.htm?type=command&param=udevice&idx=#IDX#&svalue=#SVALUE#\";\n\n// Domoticz device idx\n\nIDX = 8;\n\n// Counter value in m3\n\nlet counter = msg.payload.xml.value[0] * 100;\n\n// Replace the placeholder in the url\n\nlet domurl = domurl.replace(\"#IDX#\", IDX);\n\nlet domurl = domurl.replace(\"#SVALUE#\", counter);\n\n// Assign the url\n\nnmsg.url = domurl;\n\nnode.warn(msg.url);\n\n// Return to the HTTP request node\n\nreturn msg;"}, {"id": "598986a1df559ed1", "type": "http request", "z": "2037a45990be96e4", "name": "Set Domoticz Device", "method": "GET", "ret": "txt", "paytoqs": "ignore", "url": "", "tls": "", "persist": false, "proxy": "", "insecureHTTPParser": false, "authType": "", "senderr": false, "headers": [], "x": 1020, "y": 180, "wires": [{"598986a1df559ed1", "23ff994bb63ff845"}]}, {"id": "23ff994bb63ff845", "type": "function", "z": "2037a45990be96e4", "name": "DEBUG SET DOMOTICZ"}, {"id": "598986a1df559ed1", "type": "function", "z": "2037a45990be96e4", "name": "DEBUG SET DOMOTICZ", "active": true, "tosidebar": true, "console": false, "tostatus": false, "complete": "payload", "targetType": "msg", "statusVal": "", "statusType": "auto", "x": 1030, "y": 220, "wires": []}]
```

## Import into Node-RED

Select the flow, Node-RED Burger Menu > Import > Paste Flow JSON

## Domoticz Device

### Devices List

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
8	VirtualSensors	14058	1	HeatingUnitGasMeter	RFXMeter	RFXMeter counter	672.420 m3

### Device Widget

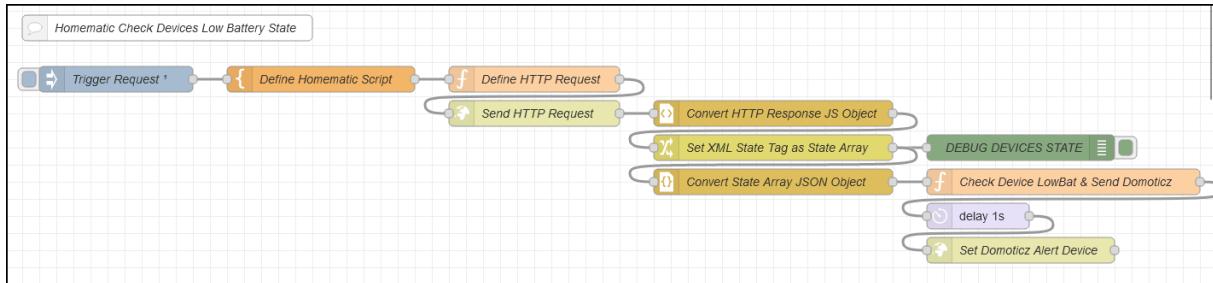


## Check Battery Status

### Purpose

Check the battery state (LOW\_BAT) for all Homematic devices and sent message to Domoticz Alert Device.

### Node-RED Flow



### Nodes

- **Inject** node triggers the HTTP request.
- **Template** node defines the Homematic script.
- **Function** node defines the HTTP request.
  - sets the HTTP POST method = mandatory by Node-RED to define in this node and not in the function node as msg.method and
  - submits the request.
- **XML** node to convert HTTP response XML tree to Javascript object.
- **Set** node to set payload as State array from the XML state tag.
- **Debug** node logs the State array.
- **Json** node to convert the State array to Json object.
- **Function** node to loop over the State array and check for Json key lowbat=true. If true send message to Domoticz via HTTP JSON/API request.
- **Delay** for 1 second to allow the previous function to complete message to Domoticz.
- **HTTP request** node to Domoticz.

## Template Node Homematic Script

```

! Homematic script to get all devices datapoints LOW_BAT and OPERATING_VOLTAGE as JSON array
!
! String JSON object holding array with devices
string state = "[";

! Get all devices objects
var objIDs = dom.GetObject(ID_DEVICES).EnumUsedIDs();

! Loop over all datapoints for the devices found
integer cnt = 0;
string objid;
foreach(objid, objIDs){
    ! Get device object
    var object = dom.GetObject(objid);
    ! Name: Briefkasten Status: ID=2530
    ! WriteLine("Name: " # object.Name() # ": ID=" # object.ID());
    string chid;
    foreach(chid, object.Channels()) {
        ! Get the channel
        var ch = dom.GetObject(chid);
        if (ch) {
            ! Get the datapoint for the attribute LOW_BAT from the channel
            var dplb = ch.DPByHssDP("LOW_BAT");
            if (dplb) {
                ! Get the datapoint for the attribute OPERATING_VOLTAGE from the channel
                var dpov = ch.DPByHssDP("OPERATING_VOLTAGE");
                ! Create JSON object name,id,lowbat,voltage
                var devstate = "{\"name\":\"" # object.Name() # "\",\"id\":\"" # object.ID() # "\",\"lowbat\":\"" # dplb.Value() # "\",\"voltage\":\"" # dpov.Value() # "\"}";
                ! WriteLine(devobj);
                if (cnt == 0) {state = state # devstate;}
                if (cnt > 0) {state = state # "," # devstate;}
                cnt = cnt + 1;
            }
        }
    }
}
state = state # "]";
! WriteLine(state);

```

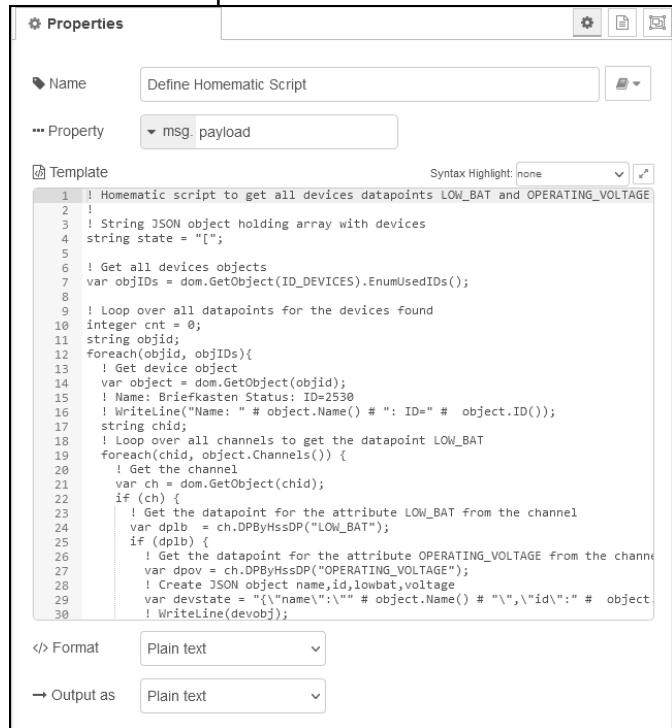
## Notes

Each Homematic script variable is defined in the HTTP XML-RPC response as XML tag.  
Ensure to use unique variable names in the Homematic script.

<pre> ▼ object ▼ xml: object   ▶ exec: array[1]   ▶ sessionId: array[1]   ▶ httpUserAgent: array[1]   ▶ state: array[1]   ▶ objIDs: array[1]   ▶ cnt: array[1]   ▶ objid: array[1]   ▶ object: array[1]   ▶ chid: array[1]   ▶ ch: array[1]   ▶ dplb: array[1]   ▶ dpov: array[1]   ▶ devstate: array[1] </pre>	<p>Each var is an XML tag defined as array with length 1.</p> <p>The tag state contains a JSON object array with all devices:</p> <pre> [   {"name":"Bad","id":1786,"lowbat":false,"voltage":3.000000},   {"name":"Briefkasten","id":2508,"lowbat":false,"voltage":1.300000},   ... ] </pre> <p>The JSON key lowbat is checked per devices.</p>
---	---

## Template Node Properties

## Format and Output as Plain Text



## Function Node Define HTTP Request

```
// Define the HTTP XML-API Request
// The script is defined in the Template node.

// Set the script from the msg payload from the Template node
let script = msg.payload;

// Define the HTTP headers
let headers = {};
headers["Content-Length"] = script.length;
headers["Content-Type"] = "application/x-www-form-urlencoded";
msg.headers = headers;

// The HTTP method POST needs to be set in the httprequest node
// msg.method = "POST";
// The port is required else HTTP 403 error
msg.url = "http://ccu-ip:8181/tclreg.exe";
msg.payload = script;

// Return the msg
return msg;
```

**Function Node Check Device LowBat & Send Domoticz**

```
// Loop over the devices state array and check for lowbat=true
// If lowbat then send message to Domotict Alert Device (level 4).
// If no lowbat devices send message to Domotict Alert Device (level 0).

// Get the devices state array as JSON object
let devices = msg.payload;
// Counter for the number of lowbat devices
let counter = 0;
// Message to log and Domoticz
let alertmsg = "No LOW_BAT devices.";
// Domoticz URL JSON/Api request
let domurl = "http://domoticz-
ip:8080/json.htm?type=command&param=udevice&idx=7&nvalue=#NVALUE##svalue=#SVALUE#";

// Loop over the devices and check
devices.forEach(checkDevice);

// Check the device if key lowbat is true
```

```
// Set Domoticz Alert Level: (0=gray=NO LOW_BAT, 4=red=LOW_BAT)
function checkDevice(device) {
    if (device.lowbat == true) {
        alertmsg = "Homematic " + device.name + "<BR>LOW_BAT (" + Math.round(device.voltage * 10) / 10
        + "V)";
        node.warn(alertmsg);
        // Domoticz Update device: IDX=7, AlertMessage, Type=General, SubType=Alert
        // HTTP response JSON format: {"status":"OK","title":"Update Device"}
        domurl = domurl.replace("#NVALUE#", 4);
        domurl = domurl.replace("#SVALUE#", alertmsg);
        var msgAlertLowBat = {}
        msgAlertLowBat.payload = alertmsg;
        msgAlertLowBat.url = domurl;
        node.send(msgAlertLowBat);
        node.warn(msgAlertLowBat.url);
        counter = counter + 1;
    }
}

if (counter == 0) {
    domurl = domurl.replace("#NVALUE#", 0);
    domurl = domurl.replace("#SVALUE#", alertmsg);
    var msgAlertLowBat = {}
    msgAlertLowBat.payload = alertmsg;
    msgAlertLowBat.url = domurl;
    node.send(msgAlertLowBat);
    node.warn(alertmsg)
}
return msg;// Loop over the devices state array and check for lowbat=true
```

#### Hint

The message contains HTML line break <BR> which is used by the Domoticz device.

#### Debug Output

```
Homematic Heizung Gas Verbrauch Sensor<BR>LOW_BAT (2.6V)
No LOW_BAT devices.
```

## Node-RED Flow Source

```
[{"id": "d867331c253e35ba", "type": "tab", "label": "TCLREGA Check Low_Bat", "disabled": false, "info": "", "env": []}, {"id": "b47eb43d6a8a8651", "type": "comment", "z": "d867331c253e35ba", "name": "Homematic Check Devices Low Battery State", "info": "", "x": 190, "y": 40, "wires": []}, {"id": "5709c35ef2ccbc6f", "type": "inject", "z": "d867331c253e35ba", "name": "Trigger Request", "props": [], "repeat": "", "crontab": "", "once": true, "onceDelay": 0.1, "topic": "", "x": 130, "y": 100, "wires": [{"id": "a0b6921cbbbf9dd5"}]}, {"id": "a0b6921cbbbf9dd5", "type": "template", "z": "d867331c253e35ba", "name": "Define Homematic Script", "field": "payload", "fieldType": "msg", "format": "text", "syntax": "plain", "template": "#! Homematic script to get all devices datapoints LOW_BAT and OPERATING_VOLTAGE as JSON array\n\nString JSON object holding array with devices\n\nstring state = \"[\";\\n\\n! Get all devices objects\n\nvar objIDs = dom.GetObject(ID_DEVICES).EnumUsedIDs();\\n\\n! Loop over all datapoints for the devices found\n\ninteger cnt = 0;\n\nstring objid; \\nforeach(objid, objIDs){\\n    ! Get device object\n    var object = dom.GetObject(objid);\\n    ! Name: Briefkasten Status: ID=2530\\n    ! WriteLine(\"Name: \" # object.Name() # \" : ID=\" # object.ID());\\n    string chid; \\n    ! Loop over all channels to get the datapoint LOW_BAT\\n    foreach(chid, object.Channels()) {\\n        ! Get the channel\n        var ch = dom.GetObject(chid);\n        if (ch) {\\n            ! Get the datapoint for the attribute LOW_BAT from the channel\n            var dblb = ch.DPByHssDP(\"LOW_BAT\");\\n            if (dblrb) {\\n                ! Get the datapoint for the attribute OPERATING_VOLTAGE from the channel\n                var dpov = ch.DPByHssDP(\"OPERATING_VOLTAGE\");\\n                ! Create JSON object\n                var devstate = \"{{\"name\":\"\"} # object.Name() # \"\",\\\"id\":\"\" # object.ID() # \"\",\\\"lowbat\":\"\" # dblrb.Value() # \"\",\\\"voltage\":\"\" # dpov.Value() # \"\"};\\n                ! WriteLine(devobj);\n                if (cnt == 0) {state = state # devstate;}\\n                if (cnt > 0) {state = state # \"\",\\\"# devstate;}\n                cnt = cnt + 1;\\n            }\n        }\n    }\n}\\n\\nstate = state # \"\";\\n\nWriteLine(state);\\n", "outputs": 1, "noerr": 0, "initialize": "", "finalize": "", "libs": [{"x": 620, "y": 100}], "wires": [{"id": "4672c5b2dcbb942c"}]}, {"id": "63af428e60eb2a2e", "type": "debug", "z": "d867331c253e35ba", "name": "DEBUG DEVICES STATE", "active": true, "tosidebar": true, "console": false, "tostatus": false, "complete": "payload", "targetType": "msg", "statusVal": "", "statusType": "auto", "x": 1190, "y": 180, "wires": []}, {"id": "4672c5b2dcbb942c", "type": "function", "z": "d867331c253e35ba", "name": "Define HTTP Request", "func": "// Define the HTTP XML-API Request\\n// The script is defined in the Template node.\\n\\n// Set the script from the msg payload from the Template node\\nlet script = msg.payload;\\n\\n// Define the HTTP headers\\nlet headers = {};\n\nheaders[\"Content-Length\"] = script.length;\nheaders[\"Content-Type\"] = \"application/x-www-form-urlencoded\";\n\nnmsg.headers = headers;\\n\\n// The HTTP method POST needs to be set in the httprequest node\\n// msg.method = \"POST\";\\n\\n// The port is required else HTTP 403 error\\nmsg.url = \"http://ccu-ip:8181/tclregae.exe\";\n\nnmsg.payload = script;\\n\\n// Return the msg\\nreturn msg;\\n", "outputs": 1, "noerr": 0, "initialize": "", "finalize": "", "libs": [{"x": 620, "y": 120}], "wires": [{"id": "084426cc9a09f0f1"}]}, {"id": "084426cc9a09f0f1", "type": "http request", "z": "d867331c253e35ba", "name": "Send HTTP Request", "method": "POST", "ret": "txt", "paytoqs": "ignore", "url": "", "tls": "", "persist": false, "proxy": "", "insecureHTTPParser": false, "authType": "", "senderr": false, "headers": [{"x": 620, "y": 140}], "wires": [{"id": "f7d2a0ad2cfdfb53"}]}, {"id": "f7d2a0adc2fdfb53", "type": "xml", "z": "d867331c253e35ba", "name": "Convert HTTP Response JS Object", "property": "payload", "attr": "$", "chr": "_", "x": 900, "y": 140, "wires": [{"id": "b589aec370327f77"}]}, {"id": "b589aec370327f77", "type": "change", "z": "d867331c253e35ba", "name": "Set XML State Tag as State Array", "rules": [{"t": "set", "p": "payload", "pt": "msg", "to": "payload.xml.state[0]", "tot": "msg"}], "action": "", "property": "", "from": "", "to": "", "reg": false, "x": 900, "y": 160, "wires": [{"id": "63af428e60eb2a2e", "x": 650, "y": 180}], {"id": "66a2c28e3f88949e", "type": "function", "z": "d867331c253e35ba", "name": "Check Device LowBat & Send Domoticz", "func": "// Loop over the devices state array and check for lowbat=true\\n// If lowbat then send message to Domotict Alert Device (level 4).\\n// If no lowbat devices send message to Domotict Alert Device (level 0).\\n\\n// get the devices state array as JSON object\\nlet devices = msg.payload;\\n\\n// Counter for the number of lowbat devices\\nlet counter = 0;\\n\\n// Message to log and Domoticz\\nlet alertmsg = \"No LOW_BAT devices.\";\\n\\n// Domoticz \\nlet domurl = \"http://domoticz-ip:8080/json.htm?type=command&param=udevice&idx=7&value=#NVALUE#&svalue=#SVALUE#\";\\n\\n// Loop over the devices and check\\ndevices.forEach(checkDevice);\\n\\n// Check the device if key lowbat is true\\n// Set Domoticz Alert Level: (0=gray=NO_LOW_BAT, 4=red=LOW_BAT)\\nfunction checkDevice(device) {\\n    if (device.lowbat == true) {\\n        alertmsg = \"Homematic \" + device.name + \"<BR>LOW_BAT (\" + Math.round(device.voltage * 10) / 10 + \"V\")\";\\n        node.warn(alertmsg);\\n        // Domoticz Update device: IDX=7, AlertMessage, Type=General, SubType=Alert\\n        // HTTP response JSON format:\\n        {\\\"status\\\": \"OK\\\", \\\"title\\\": \"Update Device\\\"}\\n        domurl = domurl.replace(\"#NVALUE#\", 4);\\n        domurl = domurl.replace(\"#SVALUE#\", alertmsg);\\n        var msgAlertLowBat = {}\\n        msgAlertLowBat.payload = alertmsg;\\n        msgAlertLowBat.url = domurl;\\n        node.send(msgAlertLowBat);\\n        node.warn(msgAlertLowBat.url);\\n        //\\n        counter = counter + 1;\\n    }\\n}\\nif (counter == 0) {\\n    domurl = domurl.replace(\"#NVALUE#\", 0);\\n}\\ndomurl = domurl.replace(\"#SVALUE#\", alertmsg);\\n        var msgAlertLowBat = {}\\n        msgAlertLowBat.payload = alertmsg;\\n        msgAlertLowBat.url = domurl;\\n        node.send(msgAlertLowBat);\\n        node.warn(alertmsg);\\n}\\nreturn msg;\\n", "outputs": 1, "noerr": 0, "initialize": "", "finalize": "", "libs": [{"x": 620, "y": 200}], "wires": [{"id": "3fb83ef845c1e928"}]}, {"id": "3fb83ef845c1e928", "type": "json", "z": "d867331c253e35ba", "name": "Convert State Array JSON Object", "property": "payload", "action": "", "pretty": false, "x": 900, "y": 220, "wires": [{"id": "66a2c28e3f88949e"}]}]
```

```
omLast": "5", "randomUnits": "seconds", "drop": false, "allowrate": false, "outputs": 1, "x": 1140, "y": 260, "wires": [{"id": "5ed221e955f88ac7"}], {"id": "5ed221e955f88ac7", "type": "http request", "z": "d867331c253e35ba", "name": "Set Domoticz Alert Device", "method": "GET", "ret": "txt", "paytoqs": "ignore", "url": "", "tls": "", "persist": false, "proxy": "", "insecureHTTPParser": false, "authType": "", "senderr": false, "headers": [], "x": 1190, "y": 300, "wires": []}]
```

## Import into Node-RED

Select the flow, Node-RED Burger Menu > Import > Paste Flow JSON

### Hints

This flow shows how to define an Homematic script in a template node.  
It could be made more generic to execute Homematic scripts.

### Domoticz

In Domoticz, a virtual sensor from type General and subtype Alert is used to get the Node-RED message (Function node Check Device LowBat).

The icon color is set according to the level set in the Node-RED Function Node, i.e. 0=Gray, 4=Red.

### Alert Device

#### Devices List

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
7	VirtualSensors	82007	1	AlertMessage	General	Alert	Homematic Heizung Gas Verbrauch Sensor LOW_BAT (2.6V)

### Device Widget



### Device Dashboard (Mode Mobile)



### Hint

The message sent by Node-RED contains HTML line break <BR> which is used by the Domoticz device.

# Python Examples

## Get System Variable

```

#!/usr/bin/python3
"""
hmapi-get-sysvar.py
Homematic JSON-API test getting system variable value
20230129 rwb1

Example run:
python getsysvar.py
Response:
<xml><exec>/tclrega.exe</exec><sessionId></sessionId><httpUserAgent>User-Agent: python-
requests/2.28.2</httpUserAgent><obj>HeatingUnitGasMeter</obj><value>672.140000</value></xml>

HeatingUnitGasMeter: 672.140000
"""

import requests
from requests.exceptions import HTTPError
import xml.etree.ElementTree as ET

# Set the CCUIP URL for the HTTP JSON-API request
URL_HMAPI = 'http://ccu-ip:8181/tclrega.exe'

SYS_VAR = "HeatingUnitGasMeter"

"""
Define the Homematic script (used as POST data) to obtain the sysvar value.
The script vars are set as tags in the XML tree response.
"""

def def_script(sysvar):
    script = "var obj = dom.GetObject(\"" + sysvar + "\");\r\n\""
    "if (obj) {var value = obj.Value();}"
    return script

"""
Get the value of a sysvar
"""

def get_sysvar_value(sysvar):
    headers = {'content-type': 'application/x-www-form-urlencoded'}
    response = requests.post(URL_HMAPI, data=def_script(sysvar), headers=headers)
    if response.status_code == 200:
        print(f'Response:\r\n{response.content.decode("utf-8")}\r\n')
        # Get the XML tree as decoded string
        root = ET.fromstring(response.content.decode("utf-8"))
        # Loop over the XML tree to get the tag value as defined in the script
        for x in root:
            # print(x.tag, x.text)
            if x.tag == "value":
                return x.text
    else:
        print(f'get_sysvar_value Error Status: {response.status_code}')
        print(response.raise_for_status())
    return ''

"""
Main
"""

def main():
    try:
        value = get_sysvar_value(SYS_VAR)
        print(f'{SYS_VAR}: {value}')
    except HTTPError as http_err:
        print(f'HTTP error occurred: {http_err}')
        # Example HTTP request without port:
        # HTTP error occurred: 403 Client Error: Forbidden for url: http://ccu-ip/tclrega.exe
    except Exception as err:
        print(f'Other error occurred: {err}')

if __name__ == "__main__":
    main()

```

# Homematic JSON API

## Information

The Homematic JSON API enables to access the CCU by using the JSON-RPC interface. The Homematic Common Gateway Interface “homematic.cgi” is used.

There is no need for the addons XML-API or CCU-Jack. If functionality is missing, scripts can be defined and executed via the method “ReGa.runScript”.

Beside submitting commands from the CLI, these could be submitted from other development tools, like Node-RED, JavaScript PHP, Python etc..

Further usage of Homematic JSON API is under review.

## Method Overview

To get an overview of JSON-API methods, send a HTTP API request to the CCU. The HTTP API response is a table with all methods containing method name, required privilege (Authorization level), short method description, parameter.

### *HTTP API Request*

```
http://ccu-ip/api/homematic.cgi
```

### *HTTP API Response*

A list of methods as a table with columns Method name, Privilege, Brief Description, JSON Parameter.

The snippet below lists the HTTP response table (in German) with first methods.

<b>Methodenübersicht</b>			
<b>Methodenname</b>	<b>Privilegstufe</b>	<b>Kurzbeschreibung</b>	<b>Parameter</b>
BidCoS.changeLanGatewayKey	Administrator	Bereitet das Setzen eines LAN Gateway Schlüssels vor.	_session_id_ lgwclass lgwserial lgwip newkey curkey
BidCoS_RF.createKeyFile	Administrator	Generiert die Datei /etc/config/keys	_session_id_ key
BidCoS_RF.getConfigurationRF	Administrator	Liefert die aktuelle Konfiguration des BidCoS-RF Dienstes	_session_id_
BidCoS_RF.isKeySet	Administrator	Ermittelt, ob ein System Sicherheitsschlüssel im ARM7 gesetzt ist	_session_id_
BidCoS_RF.setConfigurationRF	Administrator	Setzt die Konfiguration des BidCoS-RF Dienstes	_session_id_ interfaces
BidCoS_RF.validateKey	Administrator	Prüft, ob der angegebene Schlüssel dem System-Sicherheitsschlüssel entspricht	_session_id_ key

... and more ...

The parameter “params” is defined in the POST data, as JSON string for the HTTP API POST request.

```
{
  "method": "method.name",
  "params": {"paramA": "content", paramB: "content" ...}
}
```

See next for more examples.

# HTTP Request Examples

## Generic Request

### *HTTP API Request*

```
http://ccu-ip/api/homematic.cgi
```

### Post Data

```
{  
    "method": "method.name",  
    "params": { "paramA": "content", paramB: "content" ... }  
}
```

*The JSON object defining the method and parameter.*

### *HTTP API Response*

```
{"version": "1.1", "result": "DEPENDS_ON_THE_METHOD", "error": null}
```

*The JSON object with version information, result object and error object.*

If there is an error, the response “error” name contains the error details.

```
{  
    "version": "1.1",  
    "result": null,  
    "error": {  
        "name": "JSONRPCError",  
        "code": 400,  
        "message": "access denied (\\"ADMIN\\\" needed 0)"  
    }  
}
```

### Domoticz Log

```
Status: dzVents: Info: {[{"version": "1.1", "id": "error", "error": {"name": "JSONRPCError", "code": 100, "message": "invalid request (\\"{\\ expected)\"}}}}
```

*Snippet of a Domoticz log entry with invalid request error.*

## Session.login

Method: Session.login, Privilege: None, Description: User login

Parameter: username password

Returns: the session id in name "result", to be used in next requests.

### HTTP API Request

```
{  
    "method": "Session.login",  
    "params": { "username" : "Admin", "password" : "****" }  
}
```

### HTTP API Response

```
{"version": "1.1", "result": "4v4YliBdZa", "error": null}
```

The result contains the session id after successful login.

The session id is required for calling other methods depending on authorization level.

## Device.listAll

Run the method “Device.listAll” with the previous (from the method Session.login) obtained session id “4v4YliBdZa”.

### HTTP API Request

```
{  
    "method": "Device.listAll",  
    "params": { "interface" : "BidCos-RF", "_session_id_": "4v4YliBdZa" }  
}
```

### HTTP API Response

```
{  
    "version": "1.1",  
    "result": [  
        "3884", "1786", "2508", "3856", "1836", "1684", "1530", "2739", "12", "2024", "1012", "1238", "1390", "1444",  
        "1738", "5055", "1634", "1584"  
    ],  
    "error": null  
}
```

## Device.get

### *HTTP API Request*

Run the method “Device.get” with the previous (from the method Session.login) obtained session id “4v4YliBdZa”.

```
{  
    "method": "Device.get",  
    "params": { "interface" : "BidCos-RF", "_session_id_" : "4v4YliBdZa", "id": "1530" }  
}
```

### *HTTP API Response (Snippet)*

```
{  
    "version": "1.1",  
    "result": {  
        "id": "1530", "name": "Flur", "address": "000A1A49A0D8A5", "interface": "HmIP-RF",  
        "type": "HmIP-eTRV-2", "operateGroupOnly": "false",  
        "channels": [  
            {"id": "1531", "name": "Flur:0", "address": "000A1A49A0D8A5:0", "deviceId": "1530", ...,  
             "channelType": "MAINTENANCE"},  
            {"id": "1554", "name": "HmIP-eTRV-2 000A1A49A0D8A5:1", "address": "000A1A49A0D8A5:1", "deviceId": "1530",  
             ...},  
            {"id": "1583", "name": "HmIP-eTRV-2 000A1A49A0D8A5:7", "address": "000A1A49A0D8A5:7", "deviceId": "1530", ...,  
             "channelType": "HEATING_KEY_RECEIVER"}  
        ]  
    },  
    "error": null  
}
```

## Domoticz Examples

The Domoticz examples are dzVents scripts used in automation events.

The dzVents scripts are triggered by a Domoticz switch (virtual sensor, switch, push-button).

The steps performed by the dzVents scripts are:

- Login as admin,
- Get the session id,
- Obtain the method, like list of devices, get channel value etc.,
- Logout from the session.

### Pseudo Code

1. Declare the HTTP JSON API URL & POST data for the POST requests.
2. Submit HTTP JSON API asynchronous POST request to login.
3. Handle HTTP response from login, get session ID.
4. Submit HTTP JSON API asynchronous POST request for a method.
5. Handle HTTP response from the method.
6. Submit HTTP JSON API asynchronous POST request to logout.
7. Handle HTTP response from logout.

The session id is used for all subsequent POST requests.

Depending on the JSON API method, the authorization level differs.

For the examples described the login method with username “Admin” and password is used.

## Device.listAll

```
--[[  
    jsonapi-device-listall.dzvents  
    Login as admin, get the session id, obtain the list of devices, logout from the session.  
    Each of the steps submit a HTTP JSON API request with HTTP response handling.  
    Dependencies: Homematic JSON-API  
    20210607 rwbl  
]]--  
  
-- url http json-api request to execute the method set as postdata using the function openURL.  
local URL_JSONAPI = 'http://ccu-ip/api/homematic.cgi'  
local data = ''  
  
-- JSON API methods  
-- 2021-06-07 10:54:28.512 Status: dzVents: Info: {"version": "1.1","result": "5Fmp25fsEQ","error":  
null}  
local DATA_SESSION_LOGIN = '{"method":"Session.login","params":{"username":"Admin","password":"***"}}'  
local RES_SESSION_LOGIN = "RES_SESSION_LOGIN";  
local DATA_SESSION_LOGOUT = '{"method": "Session.logout", "params": {"_session_id_": "#SESSIONID#"} }'  
local RES_SESSION_LOGOUT = "RES_SESSION_LOGOUT";  
local DATA_DEVICE_LISTALL = '{"method": "Device.listAll", "params": { "_session_id_": "#SESSIONID#"} }'  
local RES_DEVICE_LISTALL = "RES_DEVICE_LISTALL"  
  
-- trigger device test switch  
local IDX_SWITCH = 83  
  
function sentRequest(domoticz, data, response)  
    domoticz.log(data)  
    domoticz.openURL({url = URL_JSONAPI, method = 'POST', postData = data, callback = response})  
end  
  
return {  
    on = {  
        timer = { TIMERRULE },  
        devices = { IDX_SWITCH },  
        httpResponses = { RES_SESSION_LOGIN, RES_DEVICE_LISTALL, RES_SESSION_LOGOUT }  
    },  
    data = {  
        sessionID = { initial = '' }  
    },  
    execute = function(domoticz, item)  
        -- check if the item is a switch, then login  
        if (item.isDevice and item.idx == IDX_SWITCH) then  
            sentRequest(domoticz, DATA_SESSION_LOGIN, RES_SESSION_LOGIN)  
        end  
        -- check if the item is a httpresponse from the openurl callback  
        if (item.isHTTPResponse) then  
            -- domoticz.log(item.data);  
            if (item.statusCode == 200) then  
                if (item.callback == RES_SESSION_LOGIN) then  
                    domoticz.data.sessionID = item.json.result  
                    domoticz.log(['SessionID: %s']:format(domoticz.data.sessionID))  
                    sentRequest(domoticz, string.gsub(DATA_DEVICE_LISTALL, '#SESSIONID#',  
domoticz.data.sessionID), RES_DEVICE_LISTALL)  
                end  
                if (item.callback == RES_DEVICE_LISTALL) then  
                    domoticz.log(item.json)  
                    sentRequest(domoticz, string.gsub(DATA_SESSION_LOGOUT, '#SESSIONID#',  
domoticz.data.sessionID), RES_SESSION_LOGOUT)  
                end  
                if (item.callback == RES_DEVICE_LOGOUT) then  
                    domoticz.log(item.json)  
                end  
            else  
                domoticz.log('[ERROR] Request: ' .. item.statusText, domoticz.LOG_ERROR)  
            end  
        end  
    end  
}
```

### Domoticz Log

```
Admin (IP: NNN.NNN.NNN.NNN) initiated a switch command (83/Test Switch/On)
Info: Handling events for: "Test Switch", value: "On"
Info: ----- Start internal script: jsonapi-device-listall: Device: "Test Switch (VirtualSensors)", Index: 83
Info: {"method": "Session.login","params": { "username" : "Admin", "password" : "***" } }
Info: ----- Finished jsonapi-device-listall
Script event triggered: /home/pi/domoticz/dzVents/runtime/dzVents.lua
Info: Handling httpResponse-events for: "RES_SESSION_LOGIN"
Info: ----- Start internal script: jsonapi-device-listall: HTTPResponse: "RES_SESSION_LOGIN"
Info: SessionID: hEf3YbDgG1
Info: {"method": "Device.listAll", "params": { "_session_id_" : "hEf3YbDgG1" } }
Info: ----- Finished jsonapi-device-listall
Script event triggered: /home/pi/domoticz/dzVents/runtime/dzVents.lua
Info: Handling httpResponse-events for: "RES_DEVICE_LISTALL"
Info: ----- Start internal script: jsonapi-device-listall: HTTPResponse: "RES_DEVICE_LISTALL"
Info: {[{"version":1.1, ["result"]=[{"3884", "1786", "2508", "1444", "3856", "1836", "1684", "1530", "2739", "12", "2024", "1012", "1238", "1390", "1738", "5055", "1634", "1584"}]}
Info: {"method": "Session.logout","params": { "_session_id_" : "" } }
Info: ----- Finished jsonapi-device-listall
Script event triggered: /home/pi/domoticz/dzVents/runtime/dzVents.lua
Info: Handling httpResponse-events for: "RES_SESSION_LOGOUT"
Info: ----- Start internal script: jsonapi-device-listall: HTTPResponse: "RES_SESSION_LOGOUT"
Info: ----- Finished jsonapi-device-listall
```

## Channel.getValue

```
--[[  
    jsonapi-device-get.dzvents  
    Login as admin, get the session id, obtain the list of devices, logout from the session.  
    Each of the steps submit a HTTP JSON API request with HTTP response handling.  
    Dependencies: Homematic JSON-API  
    20210607 rwbl  
]]--  
  
-- url http json-api request to execute the method set as postdata using the function openURL.  
local URL_JSONAPI = 'http://ccu-ip/api/homematic.cgi'  
local data = ''  
  
-- JSON API methods  
-- Info: {"version": "1.1","result": "5Fmp25fsEQ","error": null}  
local DATA_SESSION_LOGIN = '{"method":"Session.login","params":{"username":"Admin","password":"***"}}'  
local RES_SESSION_LOGIN = "RES_SESSION_LOGIN"  
  
local DATA_SESSION_LOGOUT = '{"method": "Session.logout","params":{ "_session_id_": "#SESSIONID#" }}'  
local RES_SESSION_LOGOUT = "RES_SESSION_LOGOUT"  
  
local DATA_DEVICE_GET = '{"method": "Device.get","params":{ "_session_id_": "#SESSIONID#", "id": "#ID#" }}'  
local RES_DEVICE_GET = "RES_DEVICE_GET";  
local DEVICE_ID = '1390'  
  
-- trigger device test switch  
local IDX_SWITCH = 83  
  
function sentRequest(domoticz, data, response)  
    domoticz.log(data)  
    domoticz.openURL({url = URL_JSONAPI, method = 'POST', postData = data, callback = response})  
end  
  
return {  
    on = {  
        timer = { TIMERRULE },  
        devices = { IDX_SWITCH },  
        httpResponses = { RES_SESSION_LOGIN, RES_SESSION_LOGOUT, RES_DEVICE_GET }  
    },  
    data = {  
        sessionID = { initial = "" }  
    },  
    execute = function(domoticz, item)  
        -- check if the item is a switch, then login  
        if (item.isDevice and item.idx == IDX_SWITCH) then  
            sentRequest(domoticz, DATA_SESSION_LOGIN, RES_SESSION_LOGIN)  
        end  
        -- check if the item is a httpresponse from the openurl callback  
        if (item.isHTTPResponse) then  
            if (item.statusCode == 200) then  
                if (item.callback == RES_SESSION_LOGIN) then  
                    domoticz.data.sessionID = item.json.result  
                    domoticz.log('LOGIN SessionID: %s'):format(domoticz.data.sessionID)  
                    data = string.gsub(DATA_DEVICE_GET,'#SESSIONID#',domoticz.data.sessionID)  
                    data = string.gsub(data,'#ID#', DEVICE_ID)  
                    sentRequest(domoticz, data, RES_DEVICE_GET)  
                end  
                if (item.callback == RES_DEVICE_GET) then  
                    domoticz.log(item.json)  
                    domoticz.log('LOGOUT SessionID: %s'):format(domoticz.data.sessionID)  
                    data = string.gsub(DATA_SESSION_LOGOUT,'#SESSIONID#',domoticz.data.sessionID)  
                    sentRequest(domoticz, data, RES_SESSION_LOGOUT)  
                end  
                if (item.callback == RES_DEVICE_LOGOUT) then  
                    domoticz.log(item.json)  
                end  
            else  
                domoticz.log('[ERROR] Request: ' .. item.statusText, domoticz.LOG_ERROR)  
            end  
        end  
    end  
}
```

### Domoticz Log

The channel value is logged as:

```
Info: Channel 1416 Value: 21.700000
```

```
Admin (IP: NNN.NNN.NNN.NNN) initiated a switch command (83/Test Switch/On)
Info: Handling events for: "Test Switch", value: "On"
Info: ----- Start internal script: jsonapi-channel-getvalue: Device: "Test Switch (VirtualSensors)", Index: 83
Info: {"method": "Session.login", "params": { "username" : "Admin", "password" : "***" } }
Info: ----- Finished jsonapi-channel-getvalue
Script event triggered: /home/pi/domoticz/dzVents/runtime/dzVents.lua
Info: Handling httpResponse-events for: "RES_SESSION_LOGIN"
Info: ----- Start internal script: jsonapi-channel-getvalue: HTTPResponse: "RES_SESSION_LOGIN"
Info: LOGIN SessionID: W32CX25vxI
Info: {"method": "Channel.getValue", "params": { "_session_id_" : "W32CX25vxI", "id" : "1416" } }
Info: ----- Finished jsonapi-channel-getvalue
Script event triggered: /home/pi/domoticz/dzVents/runtime/dzVents.lua
Info: Handling httpResponse-events for: "RES_CHANNEL_GETVALUE"
Info: ----- Start internal script: jsonapi-channel-getvalue: HTTPResponse: "RES_CHANNEL_GETVALUE"
Info: {[{"result": "21.900000", "version": "1.1"}]
Info: Channel 1416 Value: 21.700000
Info: LOGOUT SessionID:
Info: {"method": "Session.logout", "params": { "_session_id_" : "" } }
Info: ----- Finished jsonapi-channel-getvalue
Script event triggered: /home/pi/domoticz/dzVents/runtime/dzVents.lua
Info: Handling httpResponse-events for: "RES_SESSION_LOGOUT"
Info: ----- Start internal script: jsonapi-channel-getvalue: HTTPResponse: "RES_SESSION_LOGOUT"
Info: ----- Finished jsonapi-channel-getvalue
```

## ReGa.runScript

Execute a script defined as parameter.

Next example writes the value of a device datapoint ACTUAL\_TEMPERATURE to the output, captured in the JSON name:value pair

```
[ "result" ] = "NN.000000"
```

*This could be used to update a Domoticz device.*

```
var deviceValue = dom.GetObject('HmIP-eTRV-2
000A18A9A64DAC:1').DPByHssDP('ACTUAL_TEMPERATURE').Value();
WriteLine(deviceValue);
```

The output in the Domoticz log:  
Info: Result: 22.00000

*Use 'within the script code and not '*

```
-- [[
    jsonapi-rega-runscript.dzvents
    Login as admin, get the session id, run the script, logout from the session.
    Each of the steps submit a HTTP JSON API request with HTTP response handling.
    Dependencies: Homematic JSON-API
    20210607 rwbl
]]--

-- url http json-api request to execute the method set as postdata using the function openURL.
local URL_JSONAPI = 'http://ccu-ip/api/homematic.cgi'
local data = ''

-- JSON API methods
-- 2021-06-07 10:54:28.512 Status: dzVents: Info: {"version": "1.1","result": "5Fmp25fsEQ","error": null}
local DATA_SESSION_LOGIN = '{"method": "Session.login", "params": {"username": "Admin", "password": "***"}}'
local RES_SESSION_LOGIN = "RES_SESSION_LOGIN"

local DATA_SESSION_LOGOUT = '{"method": "Session.logout", "params": { "_session_id_": "#SESSIONID#"} }'
local RES_SESSION_LOGOUT = "RES_SESSION_LOGOUT"

local DATA_REGA_RUNSCRIPT = '{"method": "ReGa.runScript", "params": {"_session_id_": "#SESSIONID#", "script": "#SCRIPT#"} }'
local RES_REGA_RUNSCRIPT = "RES_REGA_RUNSCRIPT";
-- Playing with some scripts IMPORTANT: use ' within the script code and not "
-- The output from WriteLine is captured in the JSON result element
local REGA_SCRIPT = "var deviceValue = dom.GetObject('HmIP-eTRV-2
000A18A9A64DAC:1').DPByHssDP('ACTUAL_TEMPERATURE').Value();WriteLine(deviceValue);"
-- local REGA_SCRIPT = "var i = 1;WriteLine('Hallo Welt! '#i#' OK');"

-- trigger device switch
local IDX_SWITCH = 83

function trim(s)
    return (string.gsub(s, "^%s*(.-)%s*$", "%1"))
end

function sentRequest(domoticz, data, response)
    domoticz.log(data)
    domoticz.openURL({url = URL_JSONAPI, method = 'POST', postData = data, callback = response})
end

return {
    on = {
        timer = { TIMERRULE },
        devices = { IDX_SWITCH },
        httpResponses = { RES_SESSION_LOGIN, RES_SESSION_LOGOUT, RES_REGA_RUNSCRIPT }
    },
    data = {
        sessionID = { initial = "" }
    },
    execute = function(domoticz, item)
        -- check if the item is a switch, then login
        if (item.isDevice and item.idx == IDX_SWITCH) then
```

```

    sentRequest(domoticz, DATA_SESSION_LOGIN, RES_SESSION_LOGIN)
end
-- check if the item is a httpresponse from the openurl callback
if (item.isHTTPResponse) then
    -- domoticz.log(item.data);
    if (item.statusCode == 200) then
        if (item.callback == RES_SESSION_LOGIN) then
            domoticz.data.sessionID = item.json.result
            domoticz.log(['LOGIN SessionID: %s']:format(domoticz.data.sessionID))
            data = string.gsub(DATA_REGA_RUNSCRIPT, '#SESSIONID#', domoticz.data.sessionID)
            data = string.gsub(data, '#SCRIPT#', REGA_SCRIPT)
            domoticz.log(data)
            sentRequest(domoticz, data, RES_REGA_RUNSCRIPT)
        end
        if (item.callback == RES_REGA_RUNSCRIPT) then
            domoticz.log(item.json)
            domoticz.log(['Result: %s']:format(trim(item.json.result)))
            domoticz.log(['LOGOUT SessionID: %s']:format(domoticz.data.sessionID))
            data = string.gsub(DATA_SESSION_LOGOUT, '#SESSIONID#', domoticz.data.sessionID)
            sentRequest(domoticz, data, RES_SESSION_LOGOUT)
        end
        if (item.callback == RES_DEVICE_LOGOUT) then
            domoticz.log(item.json)
        end
    else
        domoticz.log('[ERROR] Request:' .. item.statusText, domoticz.LOG_ERROR)
    end
end
end
}

```

### Domoticz Log

The output of the WriteLine method is logged as:

Result: 22.000000

*This is the JSON name:value pair ["result"]="22.000000".*

```

VirtualSensors: Light/Switch (Test Switch)
Status: User: Admin (IP: NNN.NNN.NNN.NNN) initiated a switch command (83/Test Switch/On)
Status: dzVents: Info: Handling events for: "Test Switch", value: "On"
Status: dzVents: Info: ----- Start internal script: jsonapi-rega-runscript: Device: "Test Switch
(VirtualSensors)", Index: 83
Status: dzVents: Info: {"method": "Session.login", "params": {"username": "Admin", "password": "*****" } }
Status: dzVents: Info: ----- Finished jsonapi-rega-runscript
Status: EventSystem: Script event triggered: /home/pi/domoticz/dzVents/runtime/dzVents.lua
Status: dzVents: Info: Handling httpResponse-events for: "RES_SESSION_LOGIN"
Status: dzVents: Info: ----- Start internal script: jsonapi-rega-runscript: HTTPResponse:
"RES_SESSION_LOGIN"
Status: dzVents: Info: LOGIN SessionID: OeBSSKfIau
Status: dzVents: Info: {"method": "ReGa.runScript", "params": { "_session_id_": "OeBSSKfIau", "script":
"var deviceValue = dom.GetObject('HmIP-eTRV-2
000A18A9A64DAC:1').DPByHssDP('ACTUAL_TEMPERATURE').Value();WriteLine(deviceValue);" } }
Status: dzVents: Info: {"method": "ReGa.runScript", "params": { "_session_id_": "OeBSSKfIau", "script":
"var deviceValue = dom.GetObject('HmIP-eTRV-2
000A18A9A64DAC:1').DPByHssDP('ACTUAL_TEMPERATURE').Value();WriteLine(deviceValue);" } }
Status: dzVents: Info: ----- Finished jsonapi-rega-runscript
Status: EventSystem: Script event triggered: /home/pi/domoticz/dzVents/runtime/dzVents.lua
Status: dzVents: Info: Handling httpResponse-events for: "RES_REGA_RUNSCRIPT"
Status: dzVents: Info: ----- Start internal script: jsonapi-rega-runscript: HTTPResponse:
"RES_REGA_RUNSCRIPT"
Status: dzVents: Info: {[{"result": "22.000000", "version": "1.1"}]
Info: Result: 22.000000
Info: LOGOUT SessionID: OeBSSKfIau
Info: {"method": "Session.logout", "params": { "_session_id_": "OeBSSKfIau" } }
Info: ----- Finished jsonapi-rega-runscript
Script event triggered: /home/pi/domoticz/dzVents/runtime/dzVents.lua
Info: Handling httpResponse-events for: "RES_SESSION_LOGOUT"
Info: ----- Start internal script: jsonapi-rega-runscript: HTTPResponse: "RES_SESSION_LOGOUT"
Info: ----- Finished jsonapi-rega-runscript

```

## CLI Examples

### Session.login

Run, via CURL, the method “Session.login” with username Admin and password.

#### Command

```
curl --data "{\"method\": \"Session.login\", \"params\": { \"username\" : \"Admin\", \"password\" : \"*****\" }}" http://ccu-ip/api/homematic.cgi
```

*Ensure to escape the " character in the JSON object.*

#### Response

```
{"version": "1.1", "result": "xuNoGlzmDk", "error": null}
```

### Device.get

Run, via CURL, the method “Device.get” for the device with ID 1530 with the previous (from the method Session.login) obtained session id “xuNoGlzmDk”.

#### Command

```
curl --data "{\"method\": \"Device.get\", \"params\": { \"_session_id_\" : \"xuNoGlzmDk\", \"id\" : \"1530\" }}" http://ccu-ip/api/homematic.cgi
```

*Ensure to escape the " character in the JSON object.*

#### Response

```
{
  "version": "1.1",
  "result": {
    "id": "1530", "name": "Flur", "address": "000A1A49A0D8A5",
    "interface": "HmIP-RF", "type": "HmIP-eTRV-2", "operateGroupOnly": "false",
    "channels": [
      {
        "id": "1531", "name": "Flur:0", "address": "000A1A49A0D8A5:0", "deviceId": "1530",
        "index": 0, "partnerId": "", "mode": "MODE_AES", "category": "CATEGORY_NONE",
        "isReady": true, "isUsable": true, "isVisible": true, "isLogged": false,
        "isLogable": true, "isReadable": true, "isWritable": false, "isEventable": true, "isAesAvailable": false,
        "isVirtual": false, "channelType": "MAINTENANCE"
      },
      ...
      {
        "id": "1554", "name": "HmIP-eTRV-2 ..."
      }
    ],
    "error": null
  }
}
```

*Not all response data is shown.*

## SysVar.getAll & get

Run, via CURL, the methods

- “Sysvar.getAll” = get all system variables.
- “SysVar.get” = get a sysvar with ID 3032.
- Using the session id from the method Session.login.

### SysVar.getAll

```
curl --data "{\"method\": \"SysVar.getAll\", \"params\": { \"/_session_id_\" : \"U8lUBw6c5C\" }}"
http://ccu-ip/api/homematic.cgi
```

```
{
  "version": "1.1",
  "result": [
    {"id": "40", "name": "${sysVarAlarmMessages}", "type": "NUMBER", "unit": "", "value": "0", "channelId": "65535", "minValue": "0", "maxValue": "65000", "isLogged": false, "isVisible": false, "isInternal": true},
    {"id": "1236", "name": "${sysVarAlarmZone1}", "type": "ALARM", "unit": "", "value": "", "channelId": "65535", "valueName0": "${sysVarAlarmZone1NotTriggered}", "valueName1": "${sysVarAlarmZone1Triggered}", "isLogged": false, "isVisible": true, "isInternal": false},
    ...
    {"id": "3032", "name": "DutyCycle", "type": "NUMBER", "unit": "%", "value": "2.000000", "channelId": "65535", "minValue": "-1", "maxValue": "100", "isLogged": false, "isVisible": true, "isInternal": false},
    {"id": "1517", "name": "svEnergyCounterOldVal_1500", "type": "NUMBER", "unit": "", "value": "44.900000", "channelId": "65535", "minValue": "0", "maxValue": "1.7e+308", "isLogged": false, "isVisible": true, "isInternal": true}
  ],
  "error": null
}
```

### SysVar.get for ID 3032

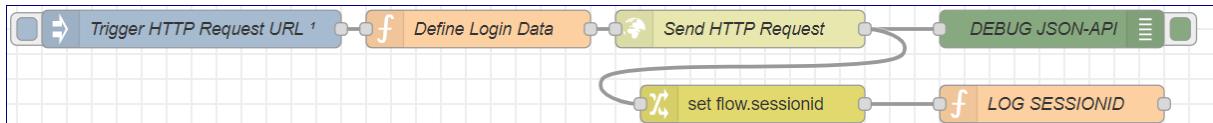
```
curl --data "{\"method\": \"SysVar.get\", \"params\": { \"/_session_id_\" : \"U8lUBw6c5C\", \"id\": \"3032\" }}"
http://ccu-ip/api/homematic.cgi
```

```
{
  "version": "1.1",
  "result": {
    "id": "3032", "name": "DutyCycle", "type": "NUMBER", "unit": "%", "value": "2.000000", "channelId": "65535", "minValue": "-1", "maxValue": "100", "isLogged": false, "isVisible": true, "isInternal": false},
    "error": null
}
```

To get the system variable value, select the JSON response name “result.value”.

## Node-RED Examples

### Session.Login



#### Nodes

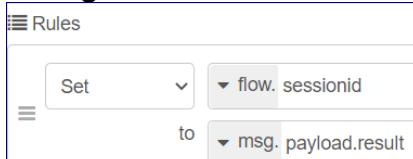
- Inject node to trigger the login flow.
- Function node to define the JSON data with method & parameter.
- HTTP request node sends the HTTP POST request with URL parameter.
- Change node to set the flow context session id used for subsequent JSON-API HTTP requests.
- Debug & Function node to log the message and the flow context.

#### Function Node

```

var data = {};
var login = true;
// Login first to get a session id {"version": "1.1","result": "4v4Y1iBdZa","error": null}
if (login) {
  data = {
    "method": "Session.login",
    "params": { "username" : "Admin", "password" : "****" }
  };
}
msg.payload = data;
return msg;
  
```

#### Change Node



#### Debug & Function Node Node.Warn Information (node: LOG SESSIONID)

```

01/06/2021, 10:37:31 node: DEBUG JS API
msg.payload.result : string[10]
"4sNv1MCN1Q"
01/06/2021, 10:37:31 node: LOG SESSIONID
function : (warn)
"4sNv1MCN1Q"
  
```

## Node-RED Flow Source

```
[{"id": "816adbde.adfaa8", "type": "inject", "z": "b9202e63.88a7", "name": "Trigger HTTP Request URL", "props": [], "repeat": "", "crontab": "", "once": true, "onceDelay": 0.1, "topic": "", "x": 160, "y": 600, "wires": [{"id": "7fdd8b9b.e682e4"}]}, {"id": "7fdd8b9b.e682e4", "type": "function", "z": "b9202e63.88a7", "name": "Define Login Data", "func": "\nvar data = {};\nvar login = true;\n\n// Login first to get a session id\n\n", "version": "1.1", "result": "\\\\"4v4YliBdZa\\\\"", "error": null, "nif (login) {\n    data = {\n        method: \\\\"Session.login\\\",
        params: {\n            username: \\\\"Admin\\\",
            password: \\\\"***\\\"
        }
    };
}\n\nmsg.payload = data;\n\nreturn msg", "outputs": 1, "noerr": 0, "initialize": "", "finalize": "", "libs": [], "x": 390, "y": 600, "wires": [{"id": "9e639cc2.0b03e"}]}, {"id": "9e639cc2.0b03e", "type": "http request", "z": "b9202e63.88a7", "name": "Send HTTP Request", "method": "POST", "ret": "obj", "paytoqs": "ignore", "url": "http://ccu-ip/api/homematic.cgi", "tls": "", "persist": false, "proxy": "", "authType": "", "x": 600, "y": 600, "wires": [{"id": "4c993f5b.0218b", "x": 850, "y": 600}], {"id": "4c993f5b.0218b", "type": "debug", "z": "b9202e63.88a7", "name": "DEBUG JSON- API", "active": true, "tosidebar": true, "console": false, "tostatus": false, "complete": "payload.result", "targetType": "msg", "statusVal": "", "statusType": "auto", "x": 850, "y": 600, "wires": []}, {"id": "ff4ac6ef.ad1618", "type": "change", "z": "b9202e63.88a7", "name": "", "rules": [{"t": "set", "p": "sessionid", "pt": "flow", "to": "payload", "d": "result", "tot": "msg"}, {"t": "set", "p": "sessionid", "pt": "flow", "to": "reg", "d": "result", "tot": "msg"}], "action": "", "property": "", "from": "", "to": "", "reg": false, "x": 610, "y": 660, "wires": [{"id": "800e33f3.76c2a"}]}, {"id": "800e33f3.76c2a", "type": "function", "z": "b9202e63.88a7", "name": "LOG SESSIONID", "func": "node.warn(flow.get(\"sessionid\"));\n", "outputs": 1, "noerr": 0, "initialize": "", "finalize": "", "libs": [], "x": 850, "y": 660, "wires": []}]]
```

## Other Methods

Based on the previous flow, the function node can be used to define other data for the HTTP POST requests.

Some examples with data to set the method and the parameter & HTTP POST response.

```
var sessionID = "QtH2NpZ53b";
Run a method with the previous obtained sessionid

/*
List all devices IDs
{"version": "1.1","result": ["3884", "1786", "2508", "3856", "1836", "1684", "1530", "2739", "12",
"2024","1012","1238","1390","1444","1738","5055","1634","1584"], "error": null}
*/
data = {
    "method": "Device.listAll",
    "params": { "interface" : "BidCos-RF", "_session_id_" : "4v4YliBdZa" }
};

/*
Get detailed device information.
{"version": "1.1","result": {"id":"1530","name":"Flur","address":"000A1A49A0D8A5","interface":"HmIP-
RF","type":"HmIP-eTRV-2","operateGroupOnly":false,"channels":[
    {"id":1531,"name":Flur:0,"address":000A1A49A0D8A5:0,"deviceId":1530,"index":0,"partnerId":"",
    "mode":MODE_AES,"category":CATEGORY_NONE,"isReady":true,"isUsable":true,"isVisible":true,"isLogged":
    false,"isLogable":true,"isReadable":true,"isWritable":false,"isEventable":true,"isAesAvailable":false,
    "isVirtual":false,"channelType":MAINTENANCE},
    ...
]
}
data = {
    "method": "Device.get",
    "params": { "interface" : "BidCos-RF", "_session_id_" : "4v4YliBdZa", "id": "1530" }
};

/*
Run a program
{"version": "1.1","result": true,"error": null}
*/
data = {
    "method": "Program.execute",
    "params": { "interface" : "BidCos-RF", "_session_id_" : sessionID, "id": "2787" }
};
```

## Python Examples

The Python 3 examples logs in as user Admin. The obtained session id is used to execute a method. After completion, the user logs out from the session.

This approach could be used in a Domoticz Python Plugin or Automation Event with Python script.

The Python HTTP library [Requests](#) is used for HTTP requests and HTTP error handling.

### SysVar.GetValueByName

Get the value of a system variable by its name.

```
#!/usr/bin/python3
"""
hmapi-get-sysvar.py
Homematic JSON-API test getting system variable value
20210609 rwb1
"""

import requests
from requests.exceptions import HTTPError
# from datetime import datetime
import time

## Set the CCUIP URL for the HTTP JSON-API request
URL_HMAPI = 'http://ccu-ip/api/homematic.cgi'

# POST data JSON string
# Session login & logout
dataLogin = {'method':'Session.login', 'params': {'username':'Admin', 'password':'*****'} }
dataLogout = {'method':'Session.logout', 'params': {'_session_id_':'#SESSIONID#'} }
# Sysvar getAll, get
# dataSysVarValueByName = {'method':'SysVar.getAll', 'params': {'_session_id_':'#SESSIONID#'} }
dataSysVarValueByName = {'method':'SysVar.getValueByName', 'params': {'_session_id_':'#SESSIONID#',
'name':'#NAME#' } }
sysVarName = 'DutyCycle'

"""
Login as Admin
Returns sessionid
"""
def login(data):
    response = requests.post(URL_HMAPI, json=data)
    if response.status_code == 200:
        jsonResponse = response.json() # {'version': '1.1', 'result': 'ovtQvZGu72', 'error': None}
        # print('Login OK - ID: {}'.format(jsonResponse['result']))
        return jsonResponse['result']
    else:
        print('Login Error Status: {}'.format(response.status_code))
        print(response.raise_for_status())
        return ''

"""
Logout with sessionid from login
"""
def logout(data, id):
    # Set the param sessionid
    data['params'][ '_session_id_ ' ] = id
    response = requests.post(URL_HMAPI, json=data)
    if response.status_code == 200:
        print('Logout - ID: {}'.format(id))
    else:
        print('Logout Error Status: {}'.format(response.status_code))
        print(response.raise_for_status())

"""
Get the value of a sysvar
"""

```

```

def get_sysvar_value_by_name(data, name, id):
    # Set the params sessionid and name
    data['params'][ '_session_id_ ' ] = id
    data['params'][ 'name' ] = name
    response = requests.post(URL_HMAPI, json=data)
    if response.status_code == 200:
        # print('sysVarValueByName OK')
        jsonResponse = response.json() #
        # print(( 'Value {}: {}'.format(name, jsonResponse[ 'result' ]) ) # Value: 2.00000
        return jsonResponse[ 'result' ]
    else:
        print('sysVarValueByName Error Status: {}'.format(response.status_code))
        print(response.raise_for_status())
        return ''

"""
Main
"""

def main():
    try:
        ## Login
        sessionid = login(dataLogin)
        print("Login - ID: {}".format(sessionid))

        ## Short delay
        time.sleep(1)

        dutycycle = get_sysvar_value_by_name(dataSysVarValueByName, sysVarName, sessionid)
        print("{}: {}".format(sysVarName, dutycycle))

        ## Short delay
        time.sleep(1)

        ## Logout
        logout(dataLogout, sessionid)

    except HTTPError as http_err:
        print(f'HTTP error occurred: {http_err}')
    except Exception as err:
        print(f'Other error occurred: {err}')

if __name__ == "__main__":
    main()

```

*Run from the Command Line with Output*

```

python3 hmapi-get-sysvar.py

Login - ID: TC5wby007v
DutyCycle: 2.000000
Logout - ID: TC5wby007v

```

## ReGa.runScript

Set the value of a system variable by running a Homematic script.

```
#!/usr/bin/python3
"""

hmapi-set-sysvar.py
Homematic JSON-API test setting the state of a system variable type string
Set the value of a sysvar type string.
The json-api does not have a method SysVar.setString.
The sysvar value is set via short script (ReGa.runScript with parameter sessionid, script
Log:
Login - ID: Aj2jWCWL9u
{'method': 'ReGa.runScript', 'params': {'_session_id': 'Aj2jWCWL9u', 'script': 'var result = dom.GetObject("DomoticzLog").State("NEW STATE"); WriteLine(result);'}}
Set sysvar state DomoticzLog, result: true
Logout - ID: Aj2jWCWL9u
20210609 rwb1
"""

import requests
from requests.exceptions import HTTPError
# from datetime import datetime
import time

## Set the CCUIP URL for the HTTP JSON-API request
URL_HMAPI = 'http://ccu-ip/api/homematic.cgi'

# POST data JSON string
# Session login & logout
dataLogin = {'method': 'Session.login', 'params': {'username': 'Admin', 'password': 'shrdlu'} }
dataLogout = {'method': 'Session.logout', 'params': {'_session_id': '#SESSIONID#'} }
dataSysVar = {'method': 'ReGa.runScript', 'params': {'_session_id': '#SESSIONID#', 'script': '#SCRIPT#'} }
}
sysVarName = 'DomoticzLog'
sysVarScript = 'var result = dom.GetObject("{}").State("{}");
WriteLine(result);'.format(sysVarName, "#STATE#")

"""

Login as Admin
Returns sessionid
"""

def login(data):
    response = requests.post(URL_HMAPI, json=data)
    if response.status_code == 200:
        jsonResponse = response.json() # {'version': '1.1', 'result': 'ovtQvZGu72', 'error': None}
        # print('Login OK - ID: {}'.format(jsonResponse['result']))
        return jsonResponse['result']
    else:
        print('Login Error Status: {}'.format(response.status_code))
        print(response.raise_for_status())
        return ''

"""

Logout with sessionid from login
"""

def logout(data, id):
    # Set the param sessionid
    data['params'][ '_session_id' ] = id
    response = requests.post(URL_HMAPI, json=data)
    if response.status_code == 200:
        print('Logout - ID: {}'.format(id))
    else:
        print('Logout Error Status: {}'.format(response.status_code))
        print(response.raise_for_status())

"""

Get the value of a sysvar
"""

def set_sysvar_state(data, script, state, id):
    # Set the params sessionid and
    data['params'][ '_session_id' ] = id
    data['params'][ 'script' ] = script.replace('#STATE#', state)
```

```

print(data)
response = requests.post(URL_HMAPI, json=data)
if response.status_code == 200:
    # print('set_sysvar_state OK')
    jsonResponse = response.json() #
    print(jsonResponse)
    return jsonResponse['result'].replace("\r\n", "")
else:
    print('sysVarValueByName Error Status: {}'.format(response.status_code))
    print(response.raise_for_status())
    return ''

"""
Main
"""
def main():
    try:
        ## Login
        sessionid = login(dataLogin)
        print("Login - ID: {}".format(sessionid))

        ## Short delay
        time.sleep(1)

        result = set_sysvar_state(dataSysVar, sysVarScript, 'NEW STATE', sessionid)
        print('Set sysvar state {}, result: {}'.format(sysVarName, result))

        ## Short delay
        time.sleep(1)

        ## Logout
        logout(dataLogout, sessionid)

    except HTTPError as http_err:
        print(f'HTTP error occurred: {http_err}')
    except Exception as err:
        print(f'Other error occurred: {err}')

if __name__ == "__main__":
    main()

```

*Run from the Command Line with Output*

```

python3 hmapi-set-sysvar.py

Login - ID: qkvJyNxEIH
{'method': 'ReGa.runScript', 'params': {'_session_id_': 'qkvJyNxEIH', 'script': 'var result = dom.GetObject("DomoticzLog").State("NEW STATE"); WriteLine(result);'}}
{'version': '1.1', 'result': 'true\r\n', 'error': None}
Set sysvar state DomoticzLog, result: true
Logout - ID: qkvJyNxEIH

```

# Addon XML-API

## Information

The [XML-API](#) addon is used to communicate between the CCU and Domoticz. Understanding the XML-API concept is essential to communicate between Domoticz and the CCU v.v.. Recommend keeping the XML-API [reference](#) at hand.

## CGI Scripts

There are several CGI scripts available:

- see the [XML-API](#) documentation or
- from Homematic WebUI > Home page > Settings > Control panel > Additional Software, the option **Set** lists the available scripts with their parameter.

The CGI scripts are executed via URL requests and used by the Domoticz automation events or Domoticz Device actions - with relevant parameters.

Domoticz triggers an HTTP XML-API request (with parameters depending on the CGI script) to the CCU and parses the HTTP response in XML format using XPath.

Below are the three XML-API CGI scripts mostly used in this document.

### **Script statelist.cgi**

Lists all devices with channels and current values.

This is a key script to get ids required by automation events to get or set values.

```
http://ccu-ip/addons/xmlapi/statelist.cgi
```

### **Script state.cgi**

List channels and values for single or multiple devices (1234,5678).

```
http://ccu-ip/addons/xmlapi/state.cgi?device_id=1541
```

### **Script statechange.cgi**

Set a new value for a datapoint.

```
http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id=1584&new_value=20
```

In examples, the wording “push” and “pull” are used:

**CCU push data to Domoticz**

The CCU triggers setting Domoticz device data.

**Domoticz pull data from the CCU**

Domoticz to get Homematic device values

**Domoticz push data to the CCU**

Domoticz triggers setting Homematic device value change

## CCU Devices State List

To be able to communicate between Domoticz and the CCU, the device ids and datapoint ids are used - can be obtained via the **statelist.cgi** script.

### HTTP XML-API Request

```
http://ccu-ip/addons/xmlapi/statelist.cgi
```

### HTTP XML-API Response

The response is an XML tree with all devices, their channels and datapoints with values.

The device name is also listed in the Homematic WebUI:

Settings > Devices > Click on a device entry shows the general devices settings which enables to change the device name.

Each device has an unique serial number.

The default device name is “device type serial number”, i.e., “HmIP-eTRV-2 000A18A9A64DAC”.

For this device, changed the name to “Radiator Thermostat MakeLab”.

The screenshot shows the Homematic WebUI interface. At the top, there is a header "RaspberryMatic – General Device Settings with Name change". Below it, a message says "Example Radiator Thermostat with type HmIP-eTRV-2 and serial number 000A18A9A64DAC". A blue arrow points from the device list below to a configuration dialog on the right. The device list shows two entries: "HmIP-eTRV-2 000A18A9A64DAC" and "Radiator Thermostat MakeLab". The configuration dialog is split into two sections: "General device settings: 000A18A9A64DAC" and "General device settings: 000A18A9A64DAC". Both dialogs show the same fields: Name (HmIP-eTRV-2 000A18A9A64DAC), Type description (HmIP-eTRV-2), Serial number (000A18A9A64DAC), Operable (checked), Visible (checked), Logged (unchecked), Service messages (unchecked), and a "Functional test" section with a "Start test" button and a note about error-free communication.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<statelist>
<device sticky_unreach="false" unreach="false" ise_id="1596" name="HM-RC-19 CUX2801001">
  <channel ise_id="1597" name="HM-RC-19 CUX2801001:0" operate="" visible="" index="0">
    <datapoint type="LOWBAT" ise_id="1598" name="CUXD.CUX2801001:0.LOWBAT" operations="5"
      timestamp="1575570253" valueunit="" valuetype="2" value="false"/>
    <datapoint type="UNREACH" ise_id="1606" name="CUXD.CUX2801001:0.UNREACH" operations="5"
      timestamp="1575570253" valueunit="" valuetype="2" value="false"/>
    <datapoint type="STICKY_UNREACH" ise_id="1602" name="CUXD.CUX2801001:0.STICKY_UNREACH"
      operations="7" timestamp="1575570253" valueunit="" valuetype="2" value="false"/>
  </channel>
  ...
</device>
<device ise_id="1011" name="HM-RCV-50 BidCoS-RF">
  <channel ise_id="1012" name="HM-RCV-50 BidCoS-RF:0" operate="" visible="" index="0">
    <datapoint type="INSTALL_MODE" ise_id="1013" name="BidCos-RF.BidCos-RF:0.INSTALL_MODE"
      operations="3" timestamp="1575570253" valueunit="" valuetype="2" value="false"/>
  </channel>
  ...
</device>
<device unreach="false" ise_id="1541" name="HmIP-eTRV-2 000A18A9A64DAC" config_pending="false">
```

```

<channel ise_id="1542" name="HmIP-eTRV-2 000A18A9A64DAC:0" operate="true" visible="true" index="0">
  <datapoint type="LOW_BAT" ise_id="1549" name="HmIP-RF.000A18A9A64DAC:0.LOW_BAT" operations="5"
  timestamp="1575709506" valueunit="" valuetype="2" value="false"/>
  <datapoint type="OPERATING_VOLTAGE" ise_id="1553" name="HmIP-RF.000A18A9A64DAC:0.OPERATING_VOLTAGE"
  operations="5" timestamp="1575709506" valueunit="" valuetype="4" value="2.800000"/>
  <datapoint type="OPERATING_VOLTAGE_STATUS" ise_id="1554" name="HmIP-
  RF.000A18A9A64DAC:0.OPERATING_VOLTAGE_STATUS" operations="5" timestamp="1575709506" valueunit=""
  valuetype="16" value="0"/>
  <datapoint type="UNREACH" ise_id="1557" name="HmIP-RF.000A18A9A64DAC:0.UNREACH" operations="5"
  timestamp="1575709506" valueunit="" valuetype="2" value="false"/>
...
</channel>
<channel ise_id="1565" name="HmIP-eTRV-2 000A18A9A64DAC:1" operate="true" visible="true" index="1">
  <datapoint type="ACTUAL_TEMPERATURE" ise_id="1567" name="HmIP-
  RF.000A18A9A64DAC:1.ACTUAL_TEMPERATURE" operations="5" timestamp="1575709506" valueunit=""
  valuetype="4" value="21.600000"/>
  <datapoint type="SET_POINT_TEMPERATURE" ise_id="1584" name="HmIP-
  RF.000A18A9A64DAC:1.SET_POINT_TEMPERATURE" operations="7" timestamp="1575709506" valueunit="°C"
  valuetype="4" value="20.000000"/>
</channel>
...
</device>
<device unreach="false" ise_id="1418" name="HMIP-PSM 0001D3C99C6AB3" config_pending="false">
  <channel ise_id="1419" name="HMIP-PSM 0001D3C99C6AB3:0" operate="true" visible="true" index="0">
    <datapoint type="OPERATING_VOLTAGE" ise_id="1426" name="HmIP-RF.0001D3C99C6AB3:0.OPERATING_VOLTAGE"
    operations="5" timestamp="0" valueunit="" valuetype="4" value="0.000000"/>
    <datapoint type="ACTUAL_TEMPERATURE" ise_id="3065" name="HmIP-
    RF.0001D3C99C6AB3:0.ACTUAL_TEMPERATURE" operations="5" timestamp="0" valueunit="" valuetype="4"
    value="0.000000"/>
  </channel>
...
</device>
</stateList>
```

*The HTTP response (extract) lists few devices.*

Device HmIP-eTRV-2 000A18A9A64DAC listed with default name.

```

<device unreach="false" ise_id="1541" name="Radiator Thermostat MakeLab" config_pending="false">
  <channel ise_id="1542" name="Radiator Thermostat MakeLab:0" operate="true" visible="true" index="0">
    <datapoint type="OPERATING_VOLTAGE" ise_id="1553" name="HmIP-RF.000A18A9A64DAC:0.OPERATING_VOLTAGE"
    operations="5" timestamp="1575710600" valueunit="" valuetype="4" value="2.800000"/>
...
</channel>
<channel ise_id="1565" name="HmIP-eTRV-2 000A18A9A64DAC:1" operate="true" visible="true" index="1">
  <datapoint type="ACTIVE_PROFILE" ise_id="1566" name="HmIP-RF.000A18A9A64DAC:1.ACTIVE_PROFILE"
  operations="7" timestamp="1575710600" valueunit="" valuetype="16" value="1"/>
...
```

Device HmIP-eTRV-2 000A18A9A64DAC listed with name changed to “Radiator Thermostat MakeLab”.

## CCU Device List

### HTTP XML-API Request

```
http://ccu-ip/addons/xmlapi/devicelist.cgi
```

### HTTP XML-API Response

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<deviceList>
  <device ready_config="true" device_type="HmIP-eTRV-2" interface="HmIP-RF" ise_id="1541"
address="000A18A9A64DAC" name="HmIP-eTRV-2 000A18A9A64DAC">
    <channel ready_config="true" ise_id="1542" address="000A18A9A64DAC:0" name="HmIP-eTRV-2
000A18A9A64DAC:0" operate="true" visible="true" transmission_mode="AES" aes_available="false"
group_partner="" index="0" parent_device="1541" direction="UNKNOWN" type="30"/>
...
  </device>

  <device ready_config="true" device_type="HMIP-PSM" interface="HmIP-RF" ise_id="1418"
address="0001D3C99C6AB3" name="HMIP-PSM 0001D3C99C6AB3">
    <channel ready_config="true" ise_id="1419" address="0001D3C99C6AB3:0" name="HMIP-PSM
0001D3C99C6AB3:0" operate="true" visible="true" transmission_mode="AES" aes_available="false"
...

```

The HTTP response shows an extract of the XML tree.

The HTTP response lists two devices Thermostat & Pluggable Switch:

HmIP-eTRV-2 (device id=1541)	HmIP-eTRV-2 000A18A9A64DAC		
HMIP-PSM (device id=1418) with their device id and serial number (address).	HMIP-PSM 0001D3C99C6AB3		

## CCU Single Device

HTTP request to obtain detailed information for a single device.  
Example Thermostat HmIP-eTRV-2.

### HTTP XML-API Request

```
http://ccu-ip/addons/xmlapi/state.cgi?device_id=1541
```

### HTTP XML-API Response

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<state>
<device ise_id="1541" name="HmIP-eTRV-2 000A18A9A64DAC" config_pending="false" unreach="false">
  <channel ise_id="1542" name="HmIP-eTRV-2 000A18A9A64DAC:0">
    ...
    <datapoint ise_id="1549" name="HmIP-RF.000A18A9A64DAC:0.LOW_BAT" type="LOW_BAT"
      timestamp="1561799020" valueunit="" valuetype="2" value="false"/>
    ...
  </channel>
  <channel ise_id="1565" name="HmIP-eTRV-2 000A18A9A64DAC:1">
    ...
    <datapoint ise_id="1567" name="HmIP-RF.000A18A9A64DAC:1.ACTUAL_TEMPERATURE"
      type="ACTUAL_TEMPERATURE" timestamp="1561799020" valueunit="" valuetype="4" value="22.900000"/>
    ...
    <datapoint ise_id="1584" name="HmIP-RF.000A18A9A64DAC:1.SET_POINT_TEMPERATURE"
      type="SET_POINT_TEMPERATURE" timestamp="1561799020" valueunit="°C" valuetype="4" value="4.500000"/>
    ...
  </channel>
  ...
</device>
</state>
```

The HTTP response shows an extract of the XML tree.

The response lists all the device datapoints, which can be used to obtain information or change a value via HTTP URL XML-API request.

These requests are used by Domoticz dzVents scripts.

To test, submit the HTTP URL and check the response on a browser, but also in the Homematic Web-UI.

## CCU Multiple Devices

HTTP request to obtain detailed information for multiple devices.  
Example 2-channel Temperature device HmIP-STE-PCB.

The XML-API statelist is required to get the datapoint ise\_id from the T1 and T2 sensors.

### *HTTP XML-API Request*

```
http://ccu-ip/config/xmlapi/statelist.cgi
```

### *HTTP Response*

Select the device named "Heizung Temperatur".  
Get, for the channels 1 and 2, the ise\_id of the datapoint ACTUAL\_TEMPERATURE.

```
<datapoint name="HmIP-RF.00281F2996368C:1.ACTUAL_TEMPERATURE" type="ACTUAL_TEMPERATURE" ise_id="5729" value="50.30000" valuetype="4" valueunit="°C" timestamp="1672851621" operations="5"/>
<datapoint name="HmIP-RF.00281F2996368C:2.ACTUAL_TEMPERATURE" type="ACTUAL_TEMPERATURE" ise_id="5732" value="39.10000" valuetype="4" valueunit="°C" timestamp="1672851621" operations="5"/>
```

The ise\_id is used to request via HTTP XML-API request the actual temperature.

### *HTTP XML-API Request*

```
http://ccu-ip/config/xmlapi/state.cgi?datapoint_id=5729,5732
```

### *HTTP Response*

```
<state>
  <datapoint ise_id="5729" value="44.100000"/>
  <datapoint ise_id="5732" value="40.400000"/>
</state>
```

## Domoticz Examples

The Domoticz examples are automation events with dzVents scripts.

The next two examples show how to Get a Value and Set the State of a Homematic IP device.

The examples can be used as a generic approach to get or set devices values for both Homematic IP as well as Domoticz devices.

### Get Homematic Device Value & Set Domoticz Device Value

#### Purpose

Read in regular intervals the datapoint ACTUAL\_TEMPERATURE of a thermostat HmIP-eTRV and update the Domoticz device type Temp, subtype LaCrosse TX3 with the actual temperature.

#### dzVents script triggers

timer, httpResponses

```
--[[

xmlapi-thermostat-temperature.dzvents
Read in regular intervals the datapoint ACTUAL_TEMPERATURE of a thermostat HmIP-eTRV.
Every device with a datapoint ACTUAL_TEMPERATURE can be read.
Just check out the device state information: http://ccu-ip/addons/xmlapi/statelist.cgi
Assign the value to a Domoticz Virtual Sensor Temperature:
Idx,Hardware,Name,Type,SubType,Data
31,VirtualSensors,MakeLab Temperature,Temp,LaCrosse TX3,17.8 C
Dependencies: Homematic CCU XML-API Addon
Domoticz log example:
2021-03-13 15:16:00.519 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
2021-03-13 15:16:00.639 Status: dzVents: Info: Handling httpResponse-events for:
"RESXMLAPITHERMOSTATTEMPERATURE"
2021-03-13 15:16:00.639 Status: dzVents: Info: ----- Start internal script: xmlapi-thermostat-
temperature: HTTPResponse: "RESXMLAPITHERMOSTATTEMPERATURE"
2021-03-13 15:16:00.642 Status: dzVents: Info: 20.1
2021-03-13 15:16:00.642 Status: dzVents: Info: ----- Finished xmlapi-thermostat-temperature
]]--


-- url http xml-api request to get the value of a datapoint with ise_id. The ise_id is set in the
function openURL.
-- http response example: <state><datapoint ise_id="1416" value="19.700000"/></state>
local URL_XMLAPI = 'http://ccu-ip/addons/xmlapi/state.cgi?datapoint_id=';

-- define the unique (across all dzVents) callback
local RES_XMLAPI = "RESXMLAPITHERMOSTATTEMPERATURE";

-- set the datapoint ise_id of the thermostat ACTUAL_TEMPERATURE
-- <datapoint name="HmIP-RF.000A18A9A64DAC:1.ACTUAL_TEMPERATURE" type="ACTUAL_TEMPERATURE"
ise_id="1416" value="20.400000" valuetype="4" valueunit="" timestamp="1615644967" operations="5"/>
local DATAPOINT_ISE_ID = 1416;

-- domoticz thermostat temperature device
local IDX_THERMOSTAT_ACTUAL_TEMPERATURE = 31;

-- Timer
-- local TIMERRULE = 'every 10 minutes'
local TIMERRULE = 'every minute'

-- Helpers
function round(number, decimals)
    local power = 10^decimals
    return math.floor(number * power) / power
end

return {
    on = { timer = { TIMERRULE }, httpResponses = { RES_XMLAPI } },
    execute = function(domoticz, item)
        -- check if the item is a timer, then request information
        if (item.isTimer) then

```

```
    domoticz.openURL({url = URL_XMLAPI .. DATAPOINT_ISE_ID, method = 'GET', callback = RES_XMLAPI})
end

-- check if the item is a httpresponse from the openurl callback
if (item.isHTTPResponse) then
    if (item.statusCode == 200) then
        -- domoticz.log(item.data);
        -- Select the callback - in this case there is only one (but for any next developments
here might be more)
        if (item.callback == RES_XMLAPI) then
            -- get the key value from the http response: <state><datapoint ise_id="1416"
value="19.700000"/></state>
            local value =
tonumber(domoticz_applyXPath(item.data,'//datapoint[@ise_id="'.DATAPOINT_ISE_ID..'"']/@value'))
            domoticz.devices(IDX_THERMOSTAT_ACTUAL_TEMPERATURE).updateTemperature(value);
            domoticz.log(value)
        end
    else
        domoticz.log('[ERROR] Request:' .. item.statusText, domoticz.LOG_ERROR)
    end
end

end
}
```

## Get Domoticz Device Value & Set Homematic Device Value

### Purpose

Get the setpoint of a Domoticz thermostat device and set the setpoint of a Homematic thermostat device.

### dzVents script triggers

devices, httpResponses

```
--[[  
    xmlapi-thermostat-setsetpoint.dzvents  
    Set the datapoint SET_POINT_TEMPERATURE of a thermostat HmIP-eTRV.  
    Every device with a datapoint SET_POINT_TEMPERATURE can be set.  
    Just check out the device state information: http://ccu-ip/addons/xmlapi/statelist.cgi  
    The Domoticz Virtual Sensor Setpoint:  
    Idx,Hardware,Name,Type,SubType,Data  
    30,VirtualSensors, MakeLab Setpoint,Thermostat,SetPoint, 20.0  
    Dependencies: Homematic CCU XML-API Addon  
]]--  
  
-- url http xml-api request to set the value of the datapoint with ise_id. The ise_id is set in the  
function openURL.  
-- http response example:  
local URL_XMLAPI = 'http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id=#ID#&new_value=#VALUE#';  
  
-- define the unique (across all dzVents) callback  
local RES_XMLAPI = "RESXMLAPITHERMOSTATSETPOINT";  
  
-- set the datapoint ise_id of the thermostat SET_POINT_TEMPERATURE  
-- <datapoint name="HmIP-RF.000A18A9A64DAC:1.SET_POINT_TEMPERATURE" type="SET_POINT_TEMPERATURE"  
ise_id="1433" value="20.00000" valuetype="4" valueunit="°C" timestamp="1615712034" operations="7"/>  
local DATAPOINT_ISE_ID = 1433;  
  
-- domoticz thermostat device  
local IDX_SETPOINT = 30;  
  
return {  
    on = { devices = { IDX_SETPOINT }, httpResponses = { RES_XMLAPI } },  
    execute = function(domoticz, item)  
        -- check if the item is a timer, then request information  
        if (item.isDevice) then  
            -- http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id=1433&new_value=6.5  
            local url = URL_XMLAPI:gsub('#ID#',DATAPOINT_ISE_ID):gsub('#VALUE#', item.setPoint)  
            domoticz.openURL({url = url, method = 'PUT', callback = RES_XMLAPI})  
        end  
  
        -- check if the item is a httpresponse from the openurl callback  
        if (item.isHTTPResponse) then  
            if (item.statusCode == 200) then  
                -- Select the callback - in this case there is only one (but for any next developments  
here might be more)  
                if (item.callback == RES_XMLAPI) then  
                    domoticz.log(item.data);  
                    -- get the key value from the http response: <result><changed id="1433"  
new_value="6.5"/></result>  
                    local value =  
tonumber(domoticz_applyXPath(item.data,'//changed[@id="'.DATAPOINT_ISE_ID..'"']/@new_value'))  
                    domoticz.log(value)  
                end  
            else  
                domoticz.log('[ERROR] Request:' .. item.statusText, domoticz.LOG_ERROR)  
            end  
        end  
    end  
}
```

## HmIP-eTRV-2 Device Request

Domoticz to request ("pull") the setpoint & temperature of the Homematic IP radiator thermostat (HMIP-eTRV-2) with the device id=1541:

1. dzVents script event sends, every minute, an HTTP XML-API request (with callback) for device information (using the device id) to the CCU  
[http://ccu-ip/addons/xmlapi/state.cgi?device\\_id=1541](http://ccu-ip/addons/xmlapi/state.cgi?device_id=1541).
2. CCU sends a HTTP response, in XML format, to Domoticz system.
3. dzVents script handles the callback, parses the received XML string using XPath to get the values of the setpoint & temperature by reading their datapoints.
4. dzVents script updates the Domoticz text device with IDX=175.

### HTTP XML-API Request Test

Test in a browser the HTTP URL and the HTTP response.

The response lists the datapoints to pick, i.e., use in the dzVents automation events.

#### *HTTP XML-API Request*

```
http://ccu-ip/addons/xmlapi/state.cgi?device_id=1541
```

#### *HTTP XML-API Response*

The datapoints required are the SET\_POINT\_TEMPERATURE (ise\_id=1584) and the ACTUAL\_TEMPERATURE (ise\_id=1567).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<state>
  <device config_pending="false" unreach="false" ise_id="1541" name="HmIP-eTRV-2 000A18A9A64DAC">
    <channel ise_id="1542" name="HmIP-eTRV-2 000A18A9A64DAC:0">
      ...
      <datapoint ise_id="1553" name="HmIP-RF.000A18A9A64DAC:0.OPERATING_VOLTAGE" timestamp="1575647467"
      valueunit="" valuetype="4" value="2.800000" type="OPERATING_VOLTAGE"/>
      <datapoint ise_id="1554" name="HmIP-RF.000A18A9A64DAC:0.OPERATING_VOLTAGE_STATUS"
      timestamp="1575647467" valueunit="" valuetype="16" value="0" type="OPERATING_VOLTAGE_STATUS"/>
      ...
    </channel>
    <channel ise_id="1565" name="HmIP-eTRV-2 000A18A9A64DAC:1">
      ...
      <datapoint ise_id="1567" name="HmIP-RF.000A18A9A64DAC:1.ACTUAL_TEMPERATURE" timestamp="1575647467"
      valueunit="" valuetype="4" value="19.300000" type="ACTUAL_TEMPERATURE"/>
      <datapoint ise_id="1583" name="HmIP-RF.000A18A9A64DAC:1.SET_POINT_MODE" timestamp="1575647467"
      valueunit="" valuetype="16" value="1" type="SET_POINT_MODE"/>
      <datapoint ise_id="1584" name="HmIP-RF.000A18A9A64DAC:1.SET_POINT_TEMPERATURE"
      timestamp="1575647467" valueunit="°C" valuetype="4" value="20.000000" type="SET_POINT_TEMPERATURE"/>
      ...
    </channel>
  ...
</device>
</state>
```

*The HTTP response shows an extract of the XML tree.*

## Automation Event

```
-- thermostat_makelab_infodzvents
-- Domoticz device text (named MakeLab Thermostat Info)
local IDX_MAKELAB_THERMOSTAT_INFO = 50;
local ID_DEVICE = 1541;
-- url of the CCU3 to obtain device information
local URL_CCU3 = 'http://ccu-ip/addons/xmlapi/state.cgi?device_id=' .. ID_DEVICE;
-- callback of the url request - must be unique across all automation events
local RES_CCU3 = 'RES_makelab_thermostat_info';
-- helper to round a number to n decimals
local DECIMALS = 1;

function round(number, decimals)
    local power = 10^decimals
    return math.floor(number * power) / power
end

return {
    on = {
        timer = {
            -- 'every minute' -- for tests
            'every 5 minutes'
        },
        httpResponses = {
            RES_CCU3 -- must match with the callback passed to the openURL command
        }
    },
    execute = function(domoticz, item)
        -- check if the item is a device, then request information
        if (item.isTimer) then
            domoticz.openURL({url = URL_CCU3, method = 'GET', callback = RES_CCU3})
        end

        -- check if the item is a httpresponse from the openurl callback
        if (item.isHTTPResponse) then
            if (item.statusCode == 200) then
                -- multiple datapoints - domoticz.log(item.data);
                -- parse the response using XPath
                -- select the attribute value of the datapoint element (this is XPath syntax)
                -- SETPOINT °C
                local setpointvalue = domoticz_applyXPath(item.data,'//datapoint[@ise_id="1584"]/@value')
                -- TEMPERATURE °C
                local temperaturevalue = domoticz_applyXPath(item.data,'//datapoint[@ise_id="1567"]/@value')

                -- log
                local msg = 'Setpoint: ' .. tostring(round(setpointvalue,DECIMALS)) .. ', Temperature: ' ..
                tostring(round(temperaturevalue,DECIMALS))
                domoticz.log(msg)

                -- update the domoticz device
                domoticz.devices(IDX_MAKELAB_THERMOSTAT_INFO).updateText(msg);

            else
                domoticz.log('[ERROR] Handling request:' .. item.statusText, domoticz.LOG_ERROR)
            end
        end
    end
}
```

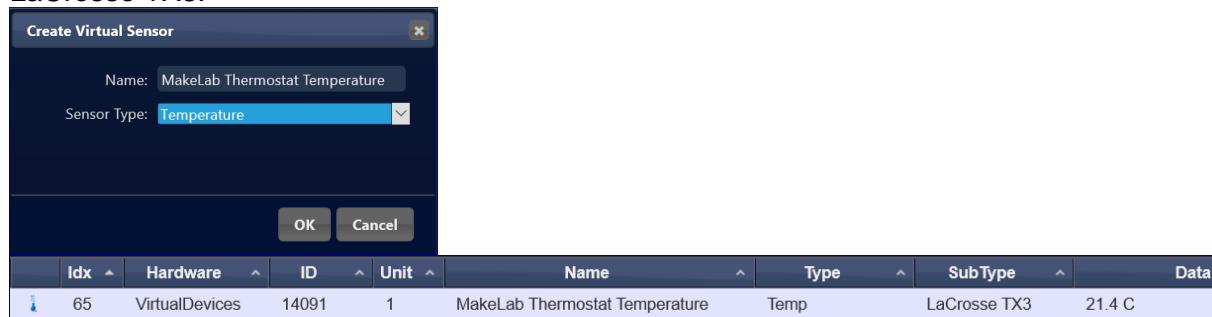
## HmIP-eTRV-2 Single Datapoint Request

Domoticz to request ("pull") the temperature of the Homematic IP radiator thermostat (HMIP-eTRV-2) with the device id=1541, datapoint id=1567:

1. dzVents script event sends, every minute, an HTTP XML-API request (with callback) for datapoint information (using datapoint id) to the CCU  
[http://ccu-ip/addons/xmlapi/state.cgi?datapoint\\_id=1567](http://ccu-ip/addons/xmlapi/state.cgi?datapoint_id=1567).
2. CCU sends a HTTP response, in XML format, to Domoticz system.
3. dzVents script handles the callback, parses the received XML string using XPath to get the values of the temperature datapoint.
4. dzVents script updates the Domoticz temp device with IDX=65.

### *Domoticz Device*

Created as Virtual Sensor Type Temperature, listed as device type Temp, SubType LaCrosse TX3.



Idx	Hardware	ID	Unit	Name	Type	SubType	Data
65	VirtualDevices	14091	1	MakeLab Thermostat Temperature	Temp	LaCrosse TX3	21.4 C

## HTTP XML-API Request Test

Test in a browser the HTTP URL and the HTTP response.

The response lists the datapoint to pick, i.e., use in the dzVents automation events.

### *HTTP XML-API Request*

```
http://ccu-ip/addons/xmlapi/state.cgi?datapoint_id=1567
```

### *HTTP XML-API Response*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<state>
  <datapoint value="22.400000" ise_id="1567"/>
</state>
```

## Automation Event

```
-- thermostat_makelab_temp.dzevents
local IDX_MAKELAB_THERMOSTAT_TEMP = 65;
local ID_DATAPOINT = 1567;
-- url of the Homematic webserver to obtain device information
local URL_CCU3 = 'http://ccu-ip/addons/xmlapi/state.cgi?datapoint_id=' .. ID_DATAPOINT;
-- callback of the url request - must be unique across all automation events
local RES_CCU3 = 'RES_makelab_thermostat_temp';
-- helper to round a number to n decimals
local DECIMALS = 1;

function round(number, decimals)
    local power = 10^decimals
    return math.floor(number * power) / power
end

return {
    on = {
        timer = {
            -- 'every minute' -- for tests
            'every 5 minutes'
        },
        httpResponses = {
            RES_CCU3 -- must match with the callback passed to the openURL command
        }
    },
    execute = function(domoticz, item)
        -- check if the item is a device, then request information
        if (item.isTimer) then
            domoticz.openURL({url = URL_CCU, method = 'GET', callback = RES_CCU})
        end

        -- check if the item is a httpresponse from the openurl callback
        if (item.isHTTPResponse) then

            if (item.statusCode == 200) then
                -- <datapoint value="22.400000" ise_id="1567"/>
                local temperaturevalue = domoticz_applyXPath(item.data,'//datapoint[@ise_id="1567"]/@value')
                -- log
                local msg = 'Temperature: ' .. tostring(round(temperaturevalue,DECIMALS))
                domoticz.log(msg)
                -- update the domoticz device
                domoticz.devices(IDX_MAKELAB_THERMOSTAT_TEMP).updateTemperature(round(temperaturevalue,1));
            else
                domoticz.log('[ERROR] Handling request:' .. item.statusText, domoticz.LOG_ERROR)
            end
        end
    end
}
```

By opening multiple URLs with unique callback string, it is possible to obtain multiple datapoints. This could be used to check the battery status of several Homematic IP devices.

## HmIP-eTRV-2 Multiple Datapoints Request

Builds upon previous example, but Domoticz to request, i.e., pull various datapoints for multiple devices.

The example below is taken from the function [Heating System Dashboard](#).

### *Principle*

Define a Lua table with properties for each device, i.e., the Homematic device datapoint and the corresponding Domoticz IDX for the devices getting Homematic datapoint attributes values assigned.

### *Example*

Get datapoints for the Homematic device with ise\_id=1541. Select the datapoint with type ACTUAL\_TEMPERATURE and assign the value to the Domoticz Temperature device with IDX=51.

### *Note*

It is also possible to let Homematic push device data triggered by value change. This requires a Homematic program (see function [Radiator Thermostat \(HmIP-eTRV\)](#)).

## Automation Event

```
-- heating_system_control.dzvents
-- Helper
JSON = (loadfile "/home/pi/domoticz/scripts/lua/JSON.lua")()

-- Create JSON object per heat exchanger (thermostat) for openURL and HTTP response
-- Device properties:
-- Homematic device datapoint ise_id
-- Domoticz IDX for TRCnnnnSP (Thermostat), TRCnnnnPV (Temperature), TCVnnnnPV (Percentage)
-- dzVents HTTP response (The response must be unique across all dzevents)
-- Keep the keys for the device properties to lowercase

-- E-6201 = Heat Exchanger MakeLab
local E6201 = JSON:decode('{"iseid":'
1541,"idxtrcsp":52,"idxtrcpv":51,"idxtcvpv":53,"response":"U6000E6201"}')
-- E-6301 = Heat Exchanger Wohnzimmer-1
local E6301 = JSON:decode('{"iseid":'
3543,"idxtrcsp":55,"idxtrcpv":54,"idxtcvpv":56,"response":"U6000E6301"}')

-- Homematic HTTP XMLAPI
-- URL of the CCU3 to obtain device information
local HTTPREQUEST = 'http://Homematic-ccu-ip/config/xmlapi/state.cgi?device_id='
-- Define the XMLAPI attributes to get data from
local TYPE_SET_POINT_TEMPERATURE = "SET_POINT_TEMPERATURE" -- TRCnnnnSP
local TYPE_ACTUAL_TEMPERATURE = "ACTUAL_TEMPERATURE" -- TRCnnnnPV
local TYPE_LEVEL = "LEVEL" -- TCVnnnnPV

-- Update the domoticz device data with the values
-- Info: ----- Start internal script: u6000_control: HTTPResponse: "U6000E6201"
-- {[{"iseid":1541, ["level":100, ["temperature":20.7, ["setpoint":23.0]}]
-- Updating Device: 1541, SP: 23.0, PV: 20.7, L: 1.0
-- Info: ----- Finished u6000_control
local function updateDevices(domoticz, deviceProperties, deviceValues)
    domoticz.log(deviceValues)
    domoticz.log(('Updating Device: %s, SP: %s, PV: %s, L: %s'):format(
        deviceValues.iseid, deviceValues.setpoint, deviceValues.temperature, deviceValues.level))
    -- Update the devices using the p&id nomenclature
    domoticz.devices(deviceProperties.idxtrcsp).updateSetPoint(deviceValues.setpoint);
    domoticz.devices(deviceProperties.idxtrcpv).updateTemperature(deviceValues.temperature);
    domoticz.devices(deviceProperties.idxtcvpv).updatePercentage(deviceValues.level);
end

return {
    on = {
        timer = {'every minute'},
        httpResponses = {E6201.response, E6301.response,},
```

```
},
execute = function(domoticz, item)
    -- check if the item is a device, then request information
    if (item.isTimer) then
        -- Submit HTTP Requests to Homematic -- wait 1 sec after each request
        domoticz.openURL({
            url = HTTPREQUEST .. E6201.iseid, method = 'GET', callback = E6201.response,})
        domoticz.openURL({
            url = HTTPREQUEST .. E6301.iseid, method = 'GET', callback = E6301.response,}).afterSec(1)
    end

    -- check if the item is a httpresponse from the openurl callback
    if (item.isHTTPResponse) then
        if (item.statusCode == 200) then
            -- From the HTTP response, check which device returned data
            local deviceValues = {}

            -- Select the datapoints by using the type name.
            -- this makes selecting datapoints easy as no need for the datapoint ise_id per device
            deviceValues.setpoint = tonumber(domoticz_applyXPath(item.data,
                ('//datapoint[@type="%s"]/@value'):format(TYPE_SET_POINT_TEMPERATURE)))
            deviceValues.temperature = tonumber(domoticz_applyXPath(item.data,
                ('//datapoint[@type="%s"]/@value'):format(TYPE_ACTUAL_TEMPERATURE)))
            deviceValues.level = tonumber(domoticz_applyXPath(item.data,
                ('//datapoint[@type="%s"]/@value'):format(TYPE_LEVEL))) * 100

            -- Select the device(s) to update
            if deviceValues.iseid == E6201.iseid then updateDevices(domoticz, E6201, deviceValues) end
            if deviceValues.iseid == E6301.iseid then updateDevices(domoticz, E6301, deviceValues) end

        else
            domoticz.log('[ERROR] Problem handling the request:' .. item.statusText, domoticz.LOG_ERROR)
        end
    end
end
}
```

## HmIP-eTRV-2 Datapoint Change Value

Domoticz to request (“push”) changing the setpoint of the Homematic IP radiator thermostat (HMIP-eTRV-2) with the device id=1541, datapoint id=1584.

Solutions have been worked:

1. Device Switch Type Selector – Selector Levels Actions
2. Automation Event with dzVents script
3. Device Switch Type Push Off Button

My preferred solution is using the automation event with dzVents script as

- easier to maintain,
- enables to handle multiple devices,
- handle HTTP response (for example log or set alert message).

### Solution Device Switch Type Selector – Selector Level Actions

For this solution, Domoticz uses device level actions (as the switch type is selector), by submitting for each level the HTTP XML-API request to the CCU.

The HTTP Response is not handled.

**MakeLab Thermostat Setpoint** 20

Last Seen: 2019-12-06 13:57:20  
Type: Light/Switch, Switch, Selector

Switch Type: Selector

Switch Icon: Heating (Heating device)

On Delay: 0 (Seconds) 0 = Disabled  
Off Delay: 0 (Seconds) 0 = Disabled  
Protected:

Selector Style:  Button set  Select menu  
Hide Off level:

Level	Level name	Order
0	0	Up, Down, Left, Right
10	18	Up, Down, Left, Right
20	19	Up, Down, Left, Right
30	20	Up, Down, Left, Right
40	21	Up, Down, Left, Right

Level name:  Add

Selector actions:

Level Action	Order
0 http://i/config/xmlapi/statechange.cgi?ise_id=1584&new_value=0	Up, Down, Left, Right
10 http://i/config/xmlapi/statechange.cgi?ise_id=1584&new_value=18	Up, Down, Left, Right
20 http://CCU-IP i/config/xmlapi/statechange.cgi?ise_id=1584&new_value=19	Up, Down, Left, Right
30 http://i/config/xmlapi/statechange.cgi?ise_id=1584&new_value=20	Up, Down, Left, Right
40 http://i/config/xmlapi/statechange.cgi?ise_id=1584&new_value=21	Up, Down, Left, Right

Description: MakeLab homematicIP Radiator Thermostat change setpoint using Level Actions via HTTP XML-API requests.

Save Delete Replace

## HTTP XML-API Request Test

Test in a browser the HTTP URL and the HTTP response.

The HTTP URL used is the same as the device level action defined.

This example request changes the setpoint to 18°C for datapoint=1584.

The response confirms the changed id and the new value.

### HTTP XML-API Request

```
http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id=1584&new_value=18
```

### HTTP XML-API Response

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<result>
  <changed id="1584" new_value="18"/>
</result>
```

## Solution Automation Event

Instead, device level actions, an option is to use a dzVents automation event triggered by device changes (the level name is used as the new setpoint), which opens the URL (HTTP XML-API request) with a HTTP callback.

The callback handles the response.

The dzVents automation event could also handle multiple devices.

## Automation Event

```
-- makelab_thermostat_setpoint.dzvents
-- set the setpoint of the radiator thermostat channel HmIP-eTRV-2 000A18A9A64DAC:1
-- set a new setpoint via url example:
-- http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id=1584&new_value=17
-- response:
-- <result><changed id="1584" new_value="17.0"/></result>

-- Domoticz device idx
local IDX_MAKELAB_THERMOSTAT_SETPOINT = 49;

-- Homematic datapoint id taken from statelist
-- <datapoint type="SET_POINT_TEMPERATURE" ise_id="1584" name="HmIP-
RF.000A18A9A64DAC:1.SET_POINT_TEMPERATURE" operations="7" timestamp="1575709506" valueunit="°C"
valuetype="4" value="20.00000"/>
local ID_DATAPOINT_SETPOINT = 1584;

-- url of the Homematic webserver to set the new setpoint
local URL_CCU3 = 'http:// ccu-ip/addons/xmlapi/statechange.cgi?ise_id=' .. ID_DATAPOINT_SETPOINT .. '&new_value='

-- callback of the url request - must be unique across all dzevents - use prefix res + script name
local RES_CCU3 = 'res_makelab_thermostat_setpoint';

-- helper to round a number to n decimals
local DECIMALS = 1;

function round(number, decimals)
  local power = 10^decimals
  return math.floor(number * power) / power
end

return {
  on = {
    devices = {
      IDX_MAKELAB_THERMOSTAT_SETPOINT
    },
    httpResponses = {
      RES_CCU3
    }
  }
}
```

```

},
execute = function(domoticz, item)
    -- check if item is device to trigger setpoint change using the levelname (0,17,18,...)
    if (item.isDevice) then
        domoticz.log('Device ' .. domoticz.devices(IDX_MAKELAB_THERMOSTAT_SETPOINT).name .. ' was changed
to ' .. tostring(domoticz.devices(IDX_MAKELAB_THERMOSTAT_SETPOINT).levelName), domoticz.LOG_INFO)
        -- get the new setpoint from the levelname
        -- 0%=0(OFF) ; 10%-40%=18-21 ; min=18, max=21
        local level = domoticz.devices(IDX_MAKELAB_THERMOSTAT_SETPOINT).levelName;
        local newsetpoint = tonumber(level);
        -- set the new setpoint
        domoticz.log('New setpoint:' .. URL_CCU3 .. tostring(newsetpoint), domoticz.LOG_INFO);
        domoticz.openURL({url = URL_CCU3 .. tostring(newsetpoint), method = 'POST', callback = RES_CCU});
    end

    -- check if the item is a httpresponse from the openurl callback
    if (item.isHTTPResponse) then
        if (item.statusCode == 200) then
            -- the full http xml response: domoticz.log(item.data);
            -- parse the response using XPath
            -- select the attribute value of the changed element (this is XPath syntax)
            -- <changed id="1584" new_value="18"/>
            domoticz.log('CCU3 Response new value: ' ..
domoticz_applyXPath(item.data, '//changed[@id="1584"]/@new_value'))
            -- The response could also be logged to an alert or control message device to log changes
        else
            domoticz.log('[ERROR] handling HTTP request:' .. item.statusText, domoticz.LOG_ERROR)
        end
    end
end
}

```

### Domoticz Log

The log shows the HTTP request and the HTTP response.

```

2019-12-07 11:48:54.970 (VirtualDevices) Light/Switch (MakeLab Thermostat Setpoint)
2019-12-07 11:48:54.963 Status: User: Admin initiated a switch command (49/MakeLab Thermostat
Setpoint/Set Level)

2019-12-07 11:48:55.103 Status: dzVents: Info: Handling events for: "MakeLab Thermostat Setpoint",
value: "19"
2019-12-07 11:48:55.103 Status: dzVents: Info: ----- Start internal script:
makelab_thermostat_setpoint: Device: "MakeLab Thermostat Setpoint (VirtualDevices)", Index: 49
2019-12-07 11:48:55.103 Status: dzVents: Info: Device MakeLab Thermostat Setpoint was changed to 19
2019-12-07 11:48:55.103 Status: dzVents: Info: New setpoint:http://ccu-
ip addons/xmlapi/statechange.cgi?se_id=1584&new_value=19
2019-12-07 11:48:55.103 Status: dzVents: Info: ----- Finished makelab_thermostat_setpoint

2019-12-07 11:48:55.104 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
2019-12-07 11:48:56.019 Status: dzVents: Info: Handling httpResponse-events for:
"res_makelab_thermostat_setpoint"
2019-12-07 11:48:56.019 Status: dzVents: Info: ----- Start internal script:
makelab_thermostat_setpoint: HTTPResponse: "res_makelab_thermostat_setpoint"
2019-12-07 11:48:56.024 Status: dzVents: Info: CCU3 Response new value: 19
2019-12-07 11:48:56.024 Status: dzVents: Info: ----- Finished makelab_thermostat_setpoint

```

## Device Switch Type Push Off Button

The state change action can also be assigned to a Switch to turn the thermostat Off.

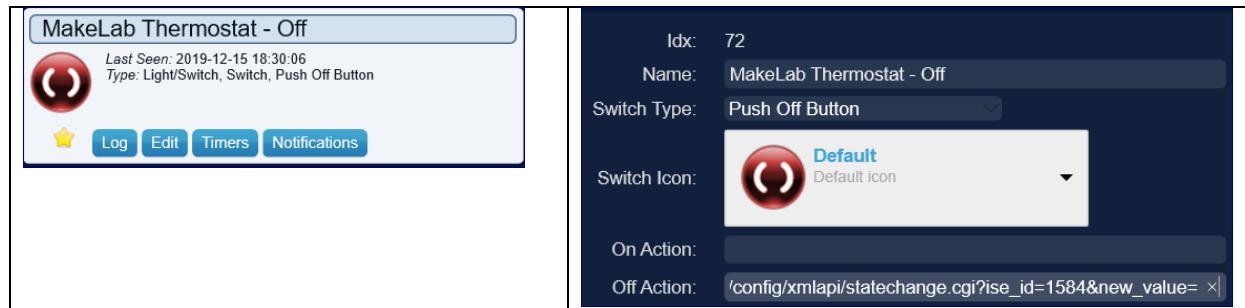
Define a VirtualSensor device with

- Name, i.e., "MakeLab Thermostat - Off"
- Sensor Type Switch.

After adding the device, select the widget and change the properties:

- Switch Type: Push Off Button
- Off Action:

[http://ccu-ip/addons/xmlapi/statechange.cgi?ise\\_id=1584&new\\_value=0](http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id=1584&new_value=0)



### Domoticz Log

Test by pushing the icon.

```
2019-12-15 18:30:06.767 (VirtualDevices) Light/Switch (MakeLab Thermostat - Off)
2019-12-15 18:30:06.758 Status: User: Admin initiated a switch command (72/MakeLab Thermostat - Off/Off)
```

## HmIP-eTRV-2 Update Domoticz Device Value

The CCU to update (“push”) the Domoticz device “MakeLab Thermostat Temperature” (IDX=65), if the “Actual Temperature Value” from the Homematic IP radiator thermostat (HmIP-eTRV-2) changes. So, every change in temperature is sent to Domoticz. For this example, the Domoticz development system is used.

1. The Homematic script “HmIP-eTRV-2-MakeLab” immediately runs if the following rule applies:  
“Channel status: HmIP-eTRV-2 000A18A9A64DAC:1 when Actual temperature within value range / with value from 1.00 and less than 0.00 trigger when changed”
2. A script reads the actual state of the datapoint “HmIP-RF.000A18A9A64DAC:1.ACTUAL\_TEMPERATURE”, builds & submits the URL of the Domoticz API request to update the value of the Domoticz device “MakeLab Thermostat Temperature” (IDX=65). The HTTP response is logged.

### Note

The solution is rather device bound, means for every device an XML-API script is required. If this is not required, to measure every value change, then consider using a Domoticz dzVents automation event to pull value(s) [of one or more devices] in regular intervals (i.e., every minute or 5 minutes). See previous Example Datapoint Request.

### Homematic Script

The script makes use of a CUxD device (see Addon CUxD) to run system commands.

The system command, which triggers a HTTP API Request, to update the Domoticz device. Example:

```
wget -q -O - 'http://domoticz-ip:8080/json.htm?type=command&param=udevice&idx=IDX&nvalue=0&svalue=20.89'
```

The CUxD device used is “CUxD (28) System” with Function Exec – Remote Control with 19 keys = key :1 is used.

HmIP-eTRV-2-MakeLab	Update the Domoticz device MakeLab Thermostat Temperature	Channel status: HmIP-eTRV-2 000A18A9A64DAC:1 when Actual temperature <i>within value range / with value from 1.00 and less than 0.00 trigger when changed</i>	Script: ... immediately run
---------------------	---	---	-----------------------------

The screenshot shows the RaspberryMatic software interface under the "Programs and connections" section. A new rule is being configured with the following details:

- Name:** HmIP-eTRV-2-MakeLab
- Description:** Update the Domoticz device MakeLab Thermostat Temperature
- Condition (if...):** Channel status: HmIP-eTRV-2 000A18A9A64DAC:1 when Actual temperature *within value range / with value from 1.00 and less than 0.00 trigger when changed*
- Activity (then..., or else...):** Script: ... immediately run
- Action:** intrinsic

The condition is set to trigger when the actual temperature is between 1.00 and 0.00. The activity is a script named "Thermostat\_MakeLab\_Notify\_Actual\_Temperature.cha..." which runs immediately. There is also an "Else..." section at the bottom.

### Homematic Script: thermostat\_makelab\_temp.script

```

! Function: Thermostat MakeLab Notify Actual Temperature change
! 20191208 by rwbl
string function = "Thermostat MakeLab Temperature";
string actdate = system.Date("%d.%m%"); ! sDate = "09.08";
! string actDate = system.Date("%d.%m.%Y"); ! sDate = "09.08.2019";
string acttime = system.Date("%H:%M%"); ! sTime = "07:32";
! string actime = system.Date("%H:%M:%S%"); ! sTime = "07:32:00";

! A domoticz virtual temperature device is used
string idx = 65; ! Type: Temp; SubType: LaCrosse TX3
! Define the parameter nvalue and svalue - see domoticz api documentation
string nvalue = 0;
! The object string is taken from the Homematic XML-API script statelist.cgi
string svalue = dom.GetObject("HmIP-RF.000A18A9A64DAC:1.ACTUAL_TEMPERATURE").State();
! for tests: string svalue = 20;

! Build the domoticz http api request url to set the actual temperature on the domoticz device
! Example: http://domoticz-ip:8080/json.htm?type=command&param=udevice&idx=65&nvalue=0&svalue=20.89
string amp = "&";
string urldom = "'http://domoticz-ip:8080/json.htm'; ! Production=60,Development=179
string urlrequest =
urldom"?type=command#amp#param=udevice#amp#idx=#idx#amp#nvalue=#nvalue#amp#svalue=#svalue#''"
;
! For Tests.
! Output'http://domoticz-ip:8080/json.htm?type=command&param=udevice&idx=65&nvalue=0&svalue=21.700000'
WriteLine(urlrequest);

! OPTION: Run the command without a return result
! res = dom.GetObject("CUxD.CUX2801001:1.CMD_EXEC").State("wget -q -O - "#sUrl);
! OPTION: Run the command with a return result
! Define the command to execute
dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State ("wget -q -O - "#urlrequest);
! Set the return flag to 1 to be able to read the json result
dom.GetObject("CUxD.CUX2801001:1.CMD_QUERY_RET").State (1);

! Start the command, wait till completed and get the result JSON string, i.e.
! {"status" : "OK","title" : "Update Device"}
! NOTE: The script running on the CCU3 waits until the completion - ensure not to execute commands
which take long time.
string res = dom.GetObject("CUxD.CUX2801001:1.CMD_RETS").State();
! WriteLine("VT="#res.VarType()#/##res); ! VT=4, {"status" : "OK","title" : "Update Device"}

! handle result
var domlog = dom.GetObject ("DomoticzLog");
! Update the var with result text
if (res.Find("OK") > 0) {
    domlog.Variable(function#"Last update OK:#actdate#" "#acttime" )
}
else {
    domlog.Variable(function#"Last update ERROR:#actdate#" "#acttime")
};
! WriteLine("VT="#domlog.VarType()); ! VT=9
WriteLine(domlog.Variable()); ! shows the last update string

```

### RaspberryMatic Log Entry

Menu: Home page > Status and control > System variables

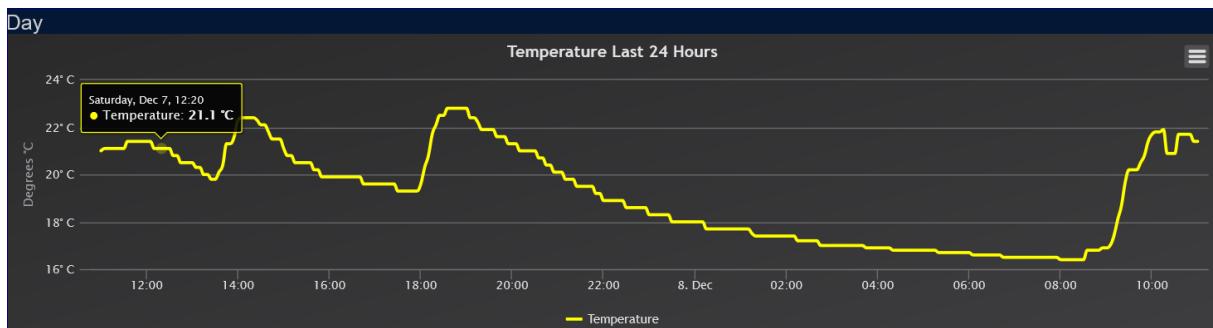
DomoticzLog	Domoticz Log Messages	03.12.2019 23:44:44	Thermostat MakeLab Temperature- Last update OK:08.12 11:14
-------------	-----------------------	---------------------	---

## Domoticz Device

The Domoticz device is a virtual device (see below properties).

The temperature is updated by the HTTP API request triggered by the CCU ("push"), there is no need to define an automation event.

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
65	VirtualDevices	14091	1	MakeLab Thermostat Temperature	Temp	LaCrosse TX3	21.4 C



## HmIP-SWDO Datapoint Request

Another example of a Domoticz Automation Event to request (“pull”) the value of a Homematic IP device datapoint value.

The device is a SWDO from which the datapoint Operating Voltage (id=2543) is pulled. The datapoint required information, i.e., ise\_id) is taken from the statelist request URL:

```
http://ccu-ip/addons/xmlapi/state.cgi
```

```
<datapoint ise_id="2543" name="HmIP-RF.0000DA498D5859:0.OPERATING_VOLTAGE" operations="5" timestamp="1575914670" valueunit="" valuetype="4" value="1.500000" type="OPERATING_VOLTAGE"/>
```

### HTTP XML-API Request Test

Test in a browser the HTTP URL and the HTTP response.

The response lists the datapoint information value and id, to be used in the dzVents automation event.

#### *HTTP XML-API Request*

```
http://ccu-ip/addons/xmlapi/state.cgi?datapoint_id=2543
```

#### *HTTP XML-API Response*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<state>
  <datapoint value="1.400000" ise_id="2543"/>
</state>
```

### Automation Event

```
--[[[

postbox_voltage.dzvents
Check the voltage of the homematicip device SWDO built into the postbox.
Log if voltage is below threshold.

]]-
local URL_CCU3 = 'http://ccu-ip/addons/xmlapi/state.cgi?datapoint_id=';
JSON = (loadfile "/home/pi/domoticz/scripts/lua/JSON.lua")() -- For Linux
local DATAPOINT2543 = JSON:decode('{"name":"Postbox"
"Voltage","id":2543,"xpath":"/datapoint[@ise_id=2543]/@value","response":"RESPOSTBOXVOLTAGE"}');
-- swdo device: set low voltage threshold (same as the Homematic device parameter.
local TH_SWDO_LOW_VOLTAGE = 1.0;

return {
  on = {
    timer = {
      'every minute' -- for tests
      -- 'every 30 minutes'
    },
    httpResponses = {
      DATAPOINT2543.response,
    }
  },
  execute = function(domoticz, item)
    -- check if the item is a device, then request information
    if (item.isTimer) then
      domoticz.openURL({url = URL_CCU3 .. DATAPOINT2543.id, method = 'GET', callback =
DATAPOINT2543.response,})
    end

    -- check if the item is a httpresponse from the openurl callback
    if (item.isHTTPResponse) then
      if (item.statusCode == 200) then
        domoticz.log(item.data);
        -- Select the callback - in case several datapoints
        if (item.callback == DATAPOINT2543.response) then
```

```

        local voltage = tonumber(domoticz_applyXPath(item.data, DATAPOINT2543.xpath))
        domoticz.log(DATAPOINT2543.name .. ':' .. voltage)
        if voltage < TH_SWDO_LOW_VOLTAGE then
            local message= DATAPOINT2543.name .. ':Low Voltage ' .. voltage .. ' ' ..
domoticz.helpers.isnowhhmm(domoticz)
            domoticz.log(message)
            -- in production system the set alert
            -- domoticz.helpers.alertmsg(domoticz, domoticz.ALERTLEVEL_GREEN, message)
        end
    end
else
    domoticz.log('[ERROR] Request:' .. item.statusText, domoticz.LOG_ERROR)
end
end
}

```

### Domoticz Log

```

2019-12-12 09:46:00.223 Status: dzVents: Info: ----- Start internal script: postbox_voltage:, trigger: every minute
2019-12-12 09:46:00.224 Status: dzVents: Info: ----- Finished postbox_voltage
2019-12-12 09:46:00.224 Status: dzVents: Info: ----- Start internal script: thermostat_makelab_temp:, trigger: every minute
2019-12-12 09:46:00.224 Status: dzVents: Info: ----- Finished thermostat_makelab_temp

2019-12-12 09:46:00.224 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
2019-12-12 09:46:00.362 Status: dzVents: Info: Handling httpResponse-events for: "RESPONSEBOXVOLTAGE"
2019-12-12 09:46:00.362 Status: dzVents: Info: ----- Start internal script: postbox_voltage:
HTTPResponse: "RESPONSEBOXVOLTAGE"
2019-12-12 09:46:00.367 Status: dzVents: Info: <?xml version="1.0" encoding="ISO-8859-1" ?><state><datapoint ise_id='2543' value='1.400000' /></state>
2019-12-12 09:46:00.367 Status: dzVents: Info: Postbox Voltage: 1.4
2019-12-12 09:46:00.367 Status: dzVents: Info: ----- Finished postbox_voltage

```

# Experiments

Next a few experiments described briefly, to test connecting and using devices.

## HMIP-PSM

### Purpose

To switch the power on/off for all devices and measure the Power (W) and Energy Consumption (Wh) of the room MakeLab.

A Homematic IP Pluggable Switch and Meter (HmIP-PSM, Device ID: 1418) connected to a Homematic operating system running a Homematic Central-Control-Unit (CCU).

Domoticz requests, in regular intervals, data from the CCU via a dzVents script and updates Domoticz device information.

### Homematic

Homematic Device Configuration					Domoticz Device Configuration (Name/Type/SubType)
Channel	Room	Function	Last modified	Control	Device
HmIP-PSM 0001D3C99C6AB3:3 Switch actuator			30.06.2019 09:10:57		<b>Powerswitch MakeLab</b> Light/switch Switch
HmIP-PSM 0001D3C99C6AB3:6 Status report measured value channel	Home office		30.06.2019 09:10:57		<b>Electric Usage MakeLab</b> General kWh

### Domoticz

#### Automation Event

```
-- powerswitch_makelab.dzvents
local IDX_POWERSWITCH_MAKELAB = 177;
local ID_DATAPOINT_STATE = 1451
local URL_CCU3 = 'http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id=' .. ID_DATAPOINT_STATE ..
'&new_value='
local RES_CCU3 = 'powerswitchmakelab';
return {
  on = {
    devices = {IDX_POWERSWITCH_MAKELAB},
    httpResponses = {RES_CCU}
  },
  execute = function(domoticz, item)
    if (item.isDevice) then
      local state = 'true';
      if domoticz.devices(IDX_POWERSWITCH_MAKELAB).state == 'Off' then
        state = 'false';
      end
      domoticz.openURL({ url = URL_CCU3 .. state, method = 'GET', callback = RES_CCU3 });
    end;
    if (item.isHTTPResponse) then
      if (item.statusCode == 200) then
        domoticz.log('[INFO] Switch : ' .. item.statusText)
      else
        domoticz.log('[ERROR] Cannot switch : ' .. item.statusText, domoticz.LOG_ERROR)
      end
    end
  end
}
```

**Event name: electric\_usage\_makelab**

```
local IDX_ELECTRICUSAGEMAKELAB = 174;
local ID_DEVICE = 1418;
local URL_CCU3 = 'http://ccu-ip/addons/xmlapi/state.cgi?device_id=' .. ID_DEVICE;
local RES_CCU3 = 'electricusagemakelab';
local DECIMALS = 2;

return {
  on = {
    timer = {'every minute'},
    httpResponses = {RES_CCU}
  },
  execute = function(domoticz, item)
    if (item.isTimer) then
      domoticz.openURL({ url = URL_CCU, method = 'GET', callback = RES_CCU3 })
    end
    if (item.isHTTPResponse) then
      if (item.statusCode == 200) then
        local powervalue = domoticz_applyXPath(item.data,'//datapoint[@ise_id="1471"]/@value') -- W
        local energyvalue = domoticz_applyXPath(item.data,'//datapoint[@ise_id="1467"]/@value') -- Wh
        domoticz.devices(IDX_ELECTRICUSAGEMAKELAB).updateElectricity(
          domoticz.helpers.roundnumber(tonumber(powervalue),2), -- W
          domoticz.helpers.roundnumber(tonumber(energyvalue),2) ) -- Wh

      else
        domoticz.log('[ERROR] Problem handling the request:' .. item.statusText, domoticz.LOG_ERROR)
      end
    end
  end
}
```

HMIP-eTRV-2

## Purpose

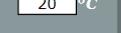
To measure and control the MakeLab room temperature.

A Homematic IP Radiator Thermostat (HmIP-eTRV-2, Device ID: 1541) connected to a Homematic operating system running a Homematic Central-Control-Unit (CCU). The thermostat setpoint is set by a Domoticz Selector Switch.

Domoticz requests, in regular intervals, the setpoint and room temperature from the CCU via a dzVents script and updates Domoticz device information.

Homematic

## Device Configuration

Channel	Room	Function	Last modified	Control	
Filter	Filter	Filter			
HmIP-eTRV-2 000A18A9A64DAC:1  Connection partner radiator thermostat (manual operation, transmitter)	Home office		30.06.2019 10:12:45	<p>Temperature 23.30°C</p> <p>Window status: Closed</p> <p>Valve opening 0%</p>  <p>5°C                    30°C</p> <p>Auto mode</p> <p>Manu mode</p> <p>Boost function</p> <p>Holiday mode</p> <p>Week profile</p> <p>1 ▾</p>	<p>20 °C</p>  <p>Off</p> <p>On</p>

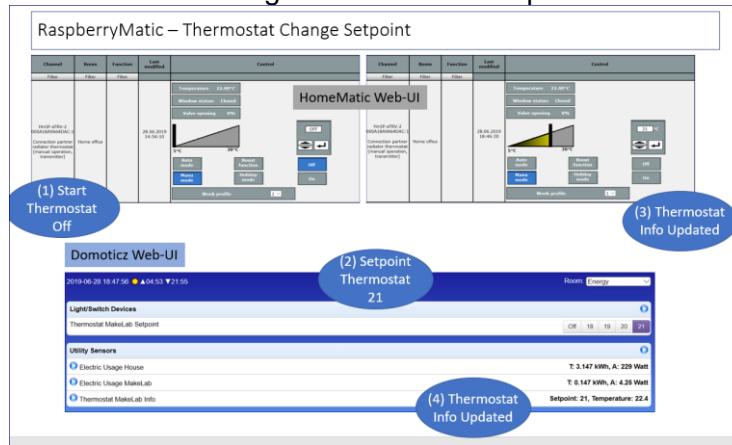
## Domoticz

## Domoticz Device Configuration

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
176	VirtualSensors	00014100	1	Thermostat MakeLab Setpoint	Light/Switch	Switch	Off
175	VirtualSensors	00082175	1	Thermostat MakeLab Info	General	Text	Setpoint: 4.5, Temperature: 23.2
<b>Thermostat MakeLab Setpoint</b>				<b>20</b>	<b>Thermostat MakeLab Info</b>		
 Last Seen: 2019-06-30 10:12:44 Type: Light/Switch, Switch, Selector				<b>20</b>	 <b>Setpoint: 20, Temperature: 23.3</b> Last Seen: 2019-06-30 10:14:01 Type: General, Text		
  Off  18  19  <b>20</b>  21					<b>Log</b>	<b>Edit</b>	
 <a href="#">Log</a> <a href="#">Edit</a> <a href="#">Timers</a> <a href="#">Notifications</a>					<b>Log</b>	<b>Edit</b>	

## Change Thermostat Setpoint

The setpoint of the thermostat is changed via dzVents script.



## Automation Event

```
-- thermostat_makelab_setpoint.dzvents
local IDX_THERMOSTAT_MAKELAB_SETPOINT = 176;
local ID_DATAPOINT_SETPOINT = 1584;
local URL_CCU3 = 'http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id=' .. ID_DATAPOINT_SETPOINT .. '&new_value='
local RES_CCU3 = 'thermostatmakelabsetpoint';
local DECIMALS = 2;

return {
on = {
    devices = {IDX_THERMOSTAT_MAKELAB_SETPOINT},
    httpResponses = {RES_CCU}
},
execute = function(domoticz, item)
if (item.isDevice) then
    -- get the new SETPOINT 0%=0(OFF); 10%-40%=18-21; min=18, max=21
    local level = domoticz.devices(IDX_THERMOSTAT_MAKELAB_SETPOINT).level;
    local newsetpoint = 0;
    if level > 0 then
        newsetpoint = 17 + (level * 0.1);
    end
    -- set the new setpoint
    domoticz.log('New setpoint:' .. URL_CCU3 .. tostring(newsetpoint), domoticz.LOG_INFO);
    domoticz.openURL({url = URL_CCU3 .. tostring(newsetpoint), method = 'POST', callback = RES_CCU});
end

-- check if the item is a httpresponse from the openurl callback
if (item.isHTTPResponse) then
    if (item.statusCode == 200) then
        -- multile datapoints
        domoticz.log(item.data);
        -- parse the response using XPath
        -- select the attribute value of the datapoint element (this is XPath syntax)
        -- domoticz.log(domoticz_applyXPath(item.data,'//datapoint[@ise_id="1584"]/@value'))
    else
        domoticz.log('[ERROR] Problem handling the request:' .. item.statusText, domoticz.LOG_ERROR)
    end
end
end
}
```

## Node-RED Examples

### Dashboard Thermostat

#### Purpose

To create a Node-RED dashboard showing thermostat datapoints

- ACTUAL\_TEMPERATURE – measured at the thermostat (°C)
- SET\_POINT\_TEMPERATURE – at the thermostat (°C)
- LEVEL - the thermostat valve position (%)

The Node-RED dashboard addon enables to create live dashboards (see [Node-RED](#)).

This experiment is developed on the Domoticz development system running Node-RED.

Node-RED communicates directly with the Homematic system to obtain the value of the datapoints by triggering HTTP XML-API requests (using the state.cgi script).

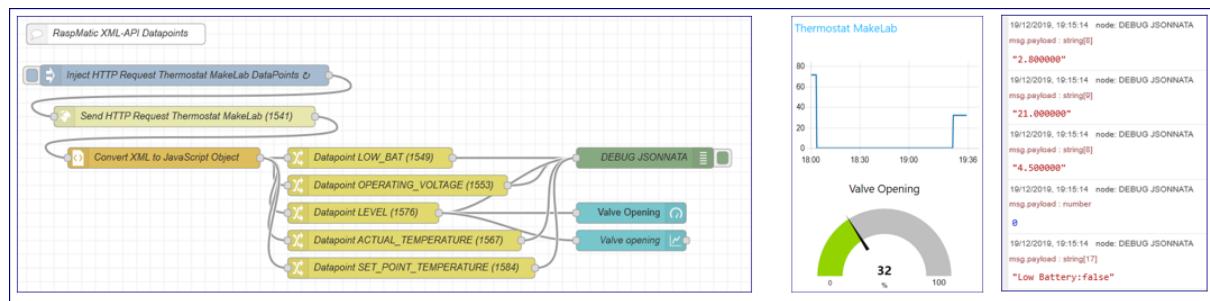
Domoticz is not involved in this experiment.

As a first test, several datapoints of the thermostat MakeLab are selected.

#### Node-RED

The picture below shows left the Node-RED flow and right a simple dashboard with chart & gauge plus the Node-RED debug log.

In a browser, enter URL <http://domoticz-ip:1880/ui> to view the Node-RED dashboard tab Homematic > Group Thermostat MakeLab.



The **Inject node** triggers every 30s, the action for the **HTTP request node** to submit the URL

```
http://ccu-ip/addons/xmlapi/state.cgi?device_id=1541
```

to get the state information for the device with datapoint id 1541.

The HTTP XML-API request returns a XML tree string with all device datapoints.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<state>
  <><device config_pending="false" unreach="false" ise_id="1541" name="Thermostat MakeLab">
    <channel ise_id="1542" name="HmIP-RF.000A18A9A64DAC:0.CONFIG_PENDING" timestamp="1576843927" valueunit="" valuetype="2" value="false" type="CONFIG_PENDING"/>
    <datapoint ise_id="1543" name="HmIP-RF.000A18A9A64DAC:0.DUTY_CYCLE" timestamp="1576843927" valueunit="" valuetype="2" value="false" type="DUTY_CYCLE"/>
    <datapoint ise_id="1544" name="HmIP-RF.000A18A9A64DAC:0.LOW_BAT" timestamp="1576843927" valueunit="" valuetype="2" value="false" type="LOW_BAT"/>
    <datapoint ise_id="1553" name="HmIP-RF.000A18A9A64DAC:0.OPERATING_VOLTAGE" timestamp="1576843927" valueunit="" valuetype="4" value="2.800000" type="OPERATING_VOLTAGE"/>
    <datapoint ise_id="1554" name="HmIP-RF.000A18A9A64DAC:0.OPERATING_VOLTAGE_STATUS" timestamp="1576843927" valueunit="" valuetype="16" value="0" type="OPERATING_VOLTAGE_STATUS"/>
    <datapoint ise_id="1555" name="HmIP-RF.000A18A9A64DAC:0.RSSI_PEER" timestamp="1576843927" valueunit="" valuetype="4" value="0" type="RSSI_PEER"/>
    <datapoint ise_id="1557" name="HmIP-RF.000A18A9A64DAC:0.UNREACH" timestamp="1576843927" valueunit="" valuetype="2" value="false" type="UNREACH"/>
    <datapoint ise_id="1561" name="HmIP-RF.000A18A9A64DAC:0.UPDATE_PENDING" timestamp="1575413140" valueunit="" valuetype="2" value="false" type="UPDATE_PENDING"/>
  </channel>
  <channel ise_id="1565" name="HmIP-eTRV-2 000A18A9A64DAC:1">
    <datapoint ise_id="1566" name="HmIP-RF.000A18A9A64DAC:1.ACTIVE_PROFILE" timestamp="1576843927" valueunit="" valuetype="16" value="1" type="ACTIVE_PROFILE"/>
    <datapoint ise_id="1567" name="HmIP-RF.000A18A9A64DAC:1.ACTUAL_TEMPERATURE" timestamp="1576843927" valueunit="" valuetype="4" value="20.900000" type="ACTUAL_TEMPERATURE"/>
    <datapoint ise_id="1568" name="HmIP-RF.000A18A9A64DAC:1.ACTUAL_TEMPERATURE_STATUS" timestamp="1576843927" valueunit="" valuetype="16" value="0" type="ACTUAL_TEMPERATURE_STATUS"/>
    <datapoint ise_id="1569" name="HmIP-RF.000A18A9A64DAC:1.BOOT_MODE" timestamp="1576843927" valueunit="" valuetype="4" value="0" type="BOOT_MODE"/>
    <datapoint ise_id="1570" name="HmIP-RF.000A18A9A64DAC:1.BOOST_TIME" timestamp="1576843927" valueunit="" valuetype="16" value="0" type="BOOST_TIME"/>
    <datapoint ise_id="1571" name="HmIP-RF.000A18A9A64DAC:1.CONTROL_DIFFERENTIAL_TEMPERATURE" timestamp="0" valueunit="" valuetype="4" value="0" type="CONTROL_DIFFERENTIAL_TEMPERATURE"/>
    <datapoint ise_id="1572" name="HmIP-RF.000A18A9A64DAC:1.CONTROL_MODE" timestamp="0" valueunit="" valuetype="16" value="0" type="CONTROL_MODE"/>
    <datapoint ise_id="1573" name="HmIP-RF.000A18A9A64DAC:1.DURATION_UNIT" timestamp="0" valueunit="" valuetype="16" value="0" type="DURATION_UNIT"/>
    <datapoint ise_id="1574" name="HmIP-RF.000A18A9A64DAC:1.DURATION_VALUE" timestamp="0" valueunit="" valuetype="16" value="0" type="DURATION_VALUE"/>
    <datapoint ise_id="1575" name="HmIP-RF.000A18A9A64DAC:1.FROST_PROTECTION" timestamp="1576843927" valueunit="" valuetype="2" value="false" type="FROST_PROTECTION"/>
    <datapoint ise_id="1576" name="HmIP-RF.000A18A9A64DAC:1.LEVEL" timestamp="1576843927" valueunit="" valuetype="0.270000" type="LEVEL"/>
    <datapoint ise_id="1577" name="HmIP-RF.000A18A9A64DAC:1.LEVEL_STATUS" timestamp="1576843927" valueunit="" valuetype="16" value="0" type="LEVEL_STATUS"/>
    <datapoint ise_id="1578" name="HmIP-RF.000A18A9A64DAC:1.PARTY_MODE" timestamp="1576843927" valueunit="" valuetype="2" value="0" type="PARTY_MODE"/>
    <datapoint ise_id="1579" name="HmIP-RF.000A18A9A64DAC:1.PARTY_POINT_TEMPERATURE" timestamp="0" valueunit="" valuetype="0.200000" type="PARTY_SET_POINT_TEMPERATURE"/>
    <datapoint ise_id="1580" name="HmIP-RF.000A18A9A64DAC:1.PARTY_TIME_END" timestamp="0" valueunit="" valuetype="20" value="0" type="PARTY_TIME_END"/>
    <datapoint ise_id="1581" name="HmIP-RF.000A18A9A64DAC:1.PARTY_TIME_START" timestamp="0" valueunit="" valuetype="20" value="0" type="PARTY_TIME_START"/>
    <datapoint ise_id="1582" name="HmIP-RF.000A18A9A64DAC:1.QUICK_VETO_TIME" timestamp="1576843927" valueunit="" valuetype="16" value="0" type="QUICK_VETO_TIME"/>
    <datapoint ise_id="1583" name="HmIP-RF.000A18A9A64DAC:1.SETPOINT_TEMPERATURE" timestamp="1576843927" valueunit="" valuetype="4" value="20.900000" type="SET_POINT_TEMPERATURE"/>
    <datapoint ise_id="1584" name="HmIP-RF.000A18A9A64DAC:1.SWITCH_POINT_OCCURRED" timestamp="1576843927" valueunit="" valuetype="2" value="false" type="SWITCH_POINT_OCCURRED"/>
    <datapoint ise_id="1586" name="HmIP-RF.000A18A9A64DAC:1.VALVE_ADAPTION" timestamp="0" valueunit="" valuetype="2" value="false" type="VALVE_ADAPTION"/>
    <datapoint ise_id="1587" name="HmIP-RF.000A18A9A64DAC:1.VALVE_STATE" timestamp="1576843927" valueunit="" valuetype="16" value="0" type="VALVE_STATE"/>
    <datapoint ise_id="1588" name="HmIP-RF.000A18A9A64DAC:1.WINDOW_STATE" timestamp="1576843927" valueunit="" valuetype="16" value="0" type="WINDOW_STATE"/>
  </channel>
  <channel ise_id="1589" name="HmIP-eTRV-2 000A18A9A64DAC:2">
    <channel ise_id="1590" name="HmIP-eTRV-2 000A18A9A64DAC:3">
      <channel ise_id="1591" name="HmIP-eTRV-2 000A18A9A64DAC:4">
        <channel ise_id="1592" name="HmIP-eTRV-2 000A18A9A64DAC:5">
        <channel ise_id="1593" name="HmIP-eTRV-2 000A18A9A64DAC:6">
    </channel>
  </channel>
  <channel ise_id="1594" name="HmIP-eTRV-2 000A18A9A64DAC:7">
  </channel>
</state>
```

All datapoints with the example datapoint ACTUAL\_TEMPERATURE highlighted as used in the example next

The **XML node** converts the XML structure to a Javascript object which is used by the various **Change nodes** to get the value of a specific datapoint by using JSONata expressions.

JSONata makes it simple to parsing this kind of data.

#### *Example JSONata expressions*

Get the ACTUAL\_TEMPERATURE value for datapoint 1567.

The payload returned is a string.

```
payload.**[ise_id="1567"].value
```

The multi-level wildcard \*\* traverses all descendants. In this case, select all "ise\_id equals 1567" values, regardless of how deeply nested the information is in the XML structure.

The XML structure snippet parsed:

```
...
<datapoint ise_id="1567" name="HmIP-RF.000A18A9A64DAC:1.ACTUAL_TEMPERATURE" timestamp="1576843927"
valueunit="" valuetype="4" value="20.900000" type="ACTUAL_TEMPERATURE"/>
...
```

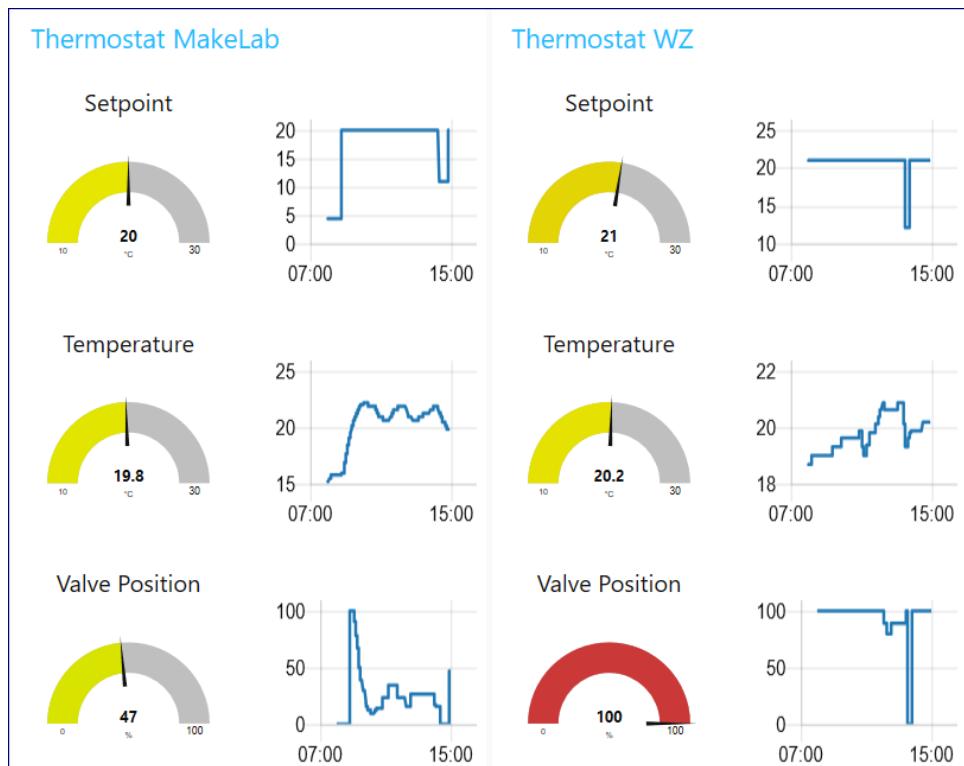
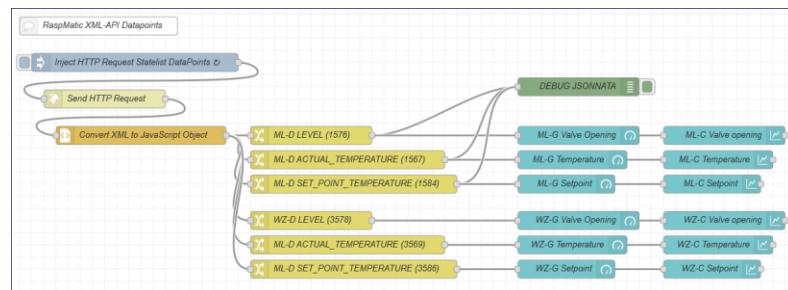
JSONata enables also to change the value, like for example the LEVEL value (=the valve position) from 0-1 to 0-100 (displayed with unit %).

The payload returned is a number.

```
$number(payload.**[ise_id=\\"1576\\"].value) * 100
```

The value is used by Node-Red Dashboard nodes like the Gauge (ui\_gauge) or Chart (ui\_chart).

The next flow has the two thermostats MakeLab and LivingRoom based on previous flow explained.



## CCU triggers Node-RED flow

### Purpose

To handle the state change of a window and door contact with magnet device (HmIP-SWDM).

Every state change is handled by a Homematic script which triggers a Node-RED HTTP GET request (CCU pushes the state information the Node-RED flow).

The HTTP GET request parameter are used by Node-RED to act accordingly.

Domoticz is not involved in this experiment, i.e., direct communication between the CCU and Node-RED.

### Homematic

The SWDM datapoints id used are: 3597 = Device ID, 3622 = STATE

If the SWDM device state changes, an HTTP request is sent (pushed) to Node-RED.

Example:

```
http://domoticz-ip:1880/3597?dp=3622&state=1
```

The parameters are the device id SWDM, datapoint id STATE and current state (0=closed, 1=open).

The request is handled by Node-RED.

### Homematic Script

The script is triggered if the state of the device changes either open or close.

Name	Description	Condition (if...)	Activity (then..., or else...)	Action
Eingang Monitor	Monitor Eingangstür auf geschlossen oder geöffnet	Channel status: HmIP-SWDM 001558A99D5A78:1 when open trigger when changed OR Device selection HmIP-SWDM 001558A99D5A78:1 when closed trigger when changed	Script: ... immediately run	<input type="checkbox"/> intrinsic

### Homematic Script

```
string sFunction = "Eingang Monitor";
string sDate = system.Date("%d.%m"); ! sDate = "09.08";
string sTime = system.Date("%H:%M"); ! sTime = "07:32";
! The object string is taken from the Homematic XML-API script statelist.cgi
string sState = dom.GetObject("HmIP-RF.001558A99D5A78:1.STATE").State();
! Node-RED Test
string cQMark = "?";
string cAmp = "&";
! XMLAPI datapoints Device & STATE
string sIDDevice = "3597";
string sIDState = "3622";
! Datapoint STATE property
string sUrlNR = "http://domoticz-ip:1880/"; ! Development
string sUrl = sUrlNR#sIDDevice#cQMark#"dp="#sIDState#cAmp#"state="#sState#"';
WriteLine(sUrl);

! Run the command with a return result
! Define the command to execute
```

```

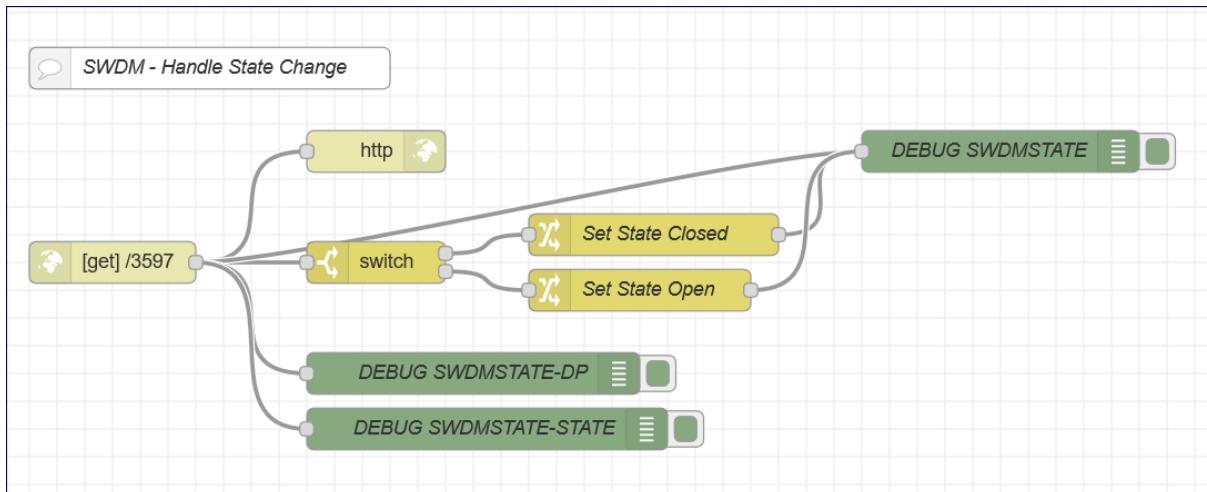
dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State ("wget -q -O - "#sUrl);

! Set the return flag to 1 to be able to read the json result
dom.GetObject("CUxD.CUX2801001:1.CMD_QUERY_RET").State (1);

! Start the command
! NOTE: script running on CCU3 waits until completion - ensure not to execute commands which take long
string sRes = dom.GetObject("CUxD.CUX2801001:1.CMD_RET").State();
! handle result
var dl = dom.GetObject ("DomoticzLog");
! Update the var with result text
if (sRes.Find("OK") > 0) {
    dl.Variable(sFunction#"-Last update OK:#sDate#" "#sTime ")
}
else {
    dl.Variable(sFunction#"Last update ERROR:#sDate#" "#sTime")
};
! WriteLine("VT="#dl.VarType()); ! VT=9
! WriteLine(dl.Variable()); ! shows the last update string

```

## Node-RED



Node	Comments
<b>http in</b>	Method: GET URL: /3597 Output to nodes: http response, switch, various debug <b>Notes</b> The output of the http in node is a JSON string containing the parameter. Example: {"dp": "3622", "state": "1"} from the URL <a href="http://domoticz-ip:1880/3597?dp=3622&amp;state=1">http://domoticz-ip:1880/3597?dp=3622&amp;state=1</a> with device id 3597 and datapoint 3622.
<b>http response</b>	No changes made, kept the defaults
<b>switch</b>	Property: msg.req.query.state == a/z: 0 == a/z: 1 Output to nodes: switch, debug There are two outputs covering 0 and 1  <b>Notes:</b> The property "state" from the URL query is used with value 0   1 (req.query.state). The property "dp" has value "3622 (req.query.dp)".

<b>change</b>	There are two change nodes handling 0 and 1. For 0, the value is to the string Closed. For 1, the value is to the string Open.
<b>debug</b>	Outputs the value of various nodes, i.e., http in, change

**Node-RED Debug log from testing the Homematic script**

The script sends HTTP request:

```
http://domoticz-ip:1880/3597?dp=3622&state=1
```

The screenshot shows the Node-RED debug log window. It displays the following log entries:

```
13/01/2020, 11:32:58 node: DEBUG SWDMSTATE
msg.payload : Object
▶ { dp: "3622", state: "1" }

13/01/2020, 11:32:58 node: DEBUG SWDMSTATE-
DP
msg.req.query.dp : string[4]
"3622"

13/01/2020, 11:32:59 node: DEBUG SWDMSTATE-
STATE
msg.req.query.state : string[1]
"1"

13/01/2020, 11:33:00 node: DEBUG SWDMSTATE
msg.payload : string[4]
"Open"
```

## Node-RED triggers HTTP GET Request

*Domoticz is not involved in this experiment, i.e., direct communication between Homematic and Node-RED.*

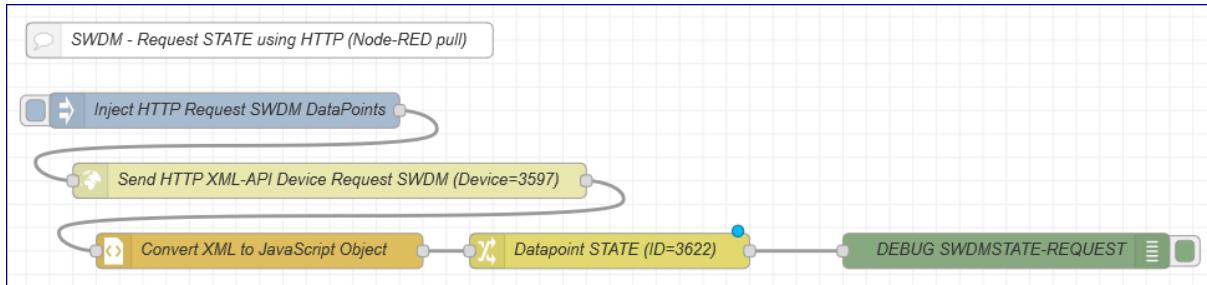
To request the state change of a window and door contact with magnet device (HmIP-SWDM).

The SWDM datapoints id used are: 3597 = Device ID, 3622 = STATE

### Node-RED

Node-RED to *pull* from Homematic the state of the SWDM device.

This is an example how to trigger, by Node-RED, the value of a datapoint via the XML-API.



Node	Comments
<b>Inject</b>	Start the request. An interval can be set to regular request device information. For this example, does not make sense as the state can be changed within an interval.
<b>http request</b>	Send the HTTP request using the defined URL. Example: http://ccu-ip/config/xmlapi/state.cgi?device_id=3597
<b>Json change</b>	Convert the HTTP XML response to a JavaScript object. Get the value from the XML tree using JSONata & xpath: \$join(["STATE:", msg.payload.**[ise_id="3622"].value])
<b>Debug</b>	Output the value of the state, i.e., 0   1 <pre> debug 13/01/2020, 12:16:38 node: DEBUG SWDMSTATE-REQUEST msg.payload : string[7] "STATE:1" 13/01/2020, 12:16:44 node: DEBUG SWDMSTATE-REQUEST msg.payload : string[7] "STATE:0" </pre>

## Node-RED triggers HTTP GET Request All Datapoints

This concept is based on the Node-RED tool [Homematic Statelist](#) and is enhanced further to

- get, in regular intervals, via HTTP XML-API request (<http://ccu-ip/config/xmlapi/statelist.cgi>),
- the device datapoint value and
- update the Domoticz device via an HTTP API request (<http://domoticz-ip:port/json.htm?type=command&param=udevice&idx=113&nvalue=0&svalue=24.1>)

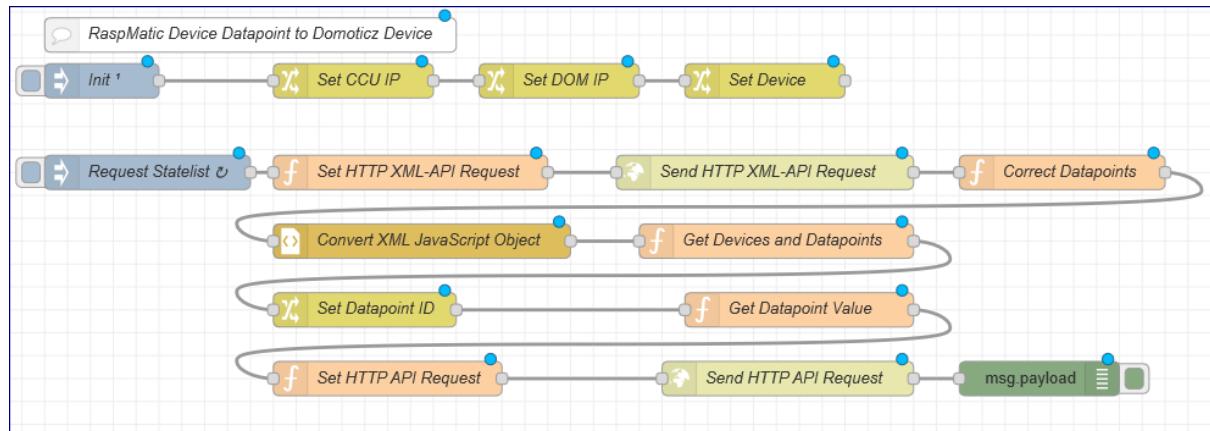
Instead of setting a single device, there is also an example how to set multiple devices from multiple datapoints.

### Single Device

Get the Homematic datapoint 1567 (Thermostat MakeLab, Type ACTUAL\_TEMPERATURE, Value °C) and assign to the Domoticz device with IDX=113.

	<b>IDX=113</b> Hardware=VirtualDevices, Name=MakeLab Temperature Type=Temp SubType=LaCrosse TX3
--	---

### Node-RED



### Function Node Get Devices and Datapoints

The purpose of this function is to

- return a json array with devices used for the ui\_dropdown
- set flow context "datapoints" for the ui\_template to display the datapoints for the selected device id

#### Notes

- Example device entry: deviceobj["Briefkasten Status"] = 2530;
- Example datapoint entry:  

```
device:"2530",datapoint:{"id":"2532","type":"CONFIG_PENDING","value":false,"name":"HmIP-RF.0000DA498D5859:0.CONFIG_PENDING"}
```
- the second array below is not used as replaced by the flow context "datapoints" - but is kept in case any enhancements planned

```
statelist = msg.payload.stateList;
```

```

devices = statelist.device
arrdevices = [];
arrdatapoints = [];
devices.forEach(function(device) {
    devicename = device.$.name;
    deviceise_id = device.$.ise_id;
    deviceobj = {}
    deviceobj[devicename]=deviceise_id;
    arrdevices.push(deviceobj);
    channels = device.channel;
    channels.forEach(function(channel){
        channeldatapoints = channel.datapoint;
        if (channeldatapoints !== undefined) {
            channeldatapoints.forEach(function(datapoint){
                obj = datapoint.$;
                if (obj !== undefined) {
                    dpobj = {
                        "device":deviceise_id,
                        "datapoint": {
                            "id":obj.ise_id,"type":obj.type,"value":obj.value,"name":obj.name
                        }
                    };
                    arrdatapoints.push(dpobj);
                }
            });
        }
    });
});
flow.set("datapoints",arrdatapoints);

msgdevices = {};
msgdevices.options = arrdevices;

// not used
msgdatapoints = {};
msgdatapoints.payload = arrdatapoints;

return [msgdevices,msgdatapoints];

```

### Function Node Get Datapoint Value

Get the value from the flow context “datapoints” using the datapoint id and returns the value property.

```

// get the datapoint id, i.e., 1657 for the selected channel
var datapointid = msg.payload;
// get all the datapoints by device
// device:"2530",datapoint:{"id":"2532","type":"CONFIG_PENDING","value":false,"name":"HmIP-
RF.0000DA498D5859:0.CONFIG_PENDING"}
datapoints = flow.get("datapoints");
datapoints.forEach(function(datapoint){
    if (datapoint.datapoint.id == datapointid) {
        msg.payload = datapoint.datapoint.value;
    }
});
return msg;

```

**Function Node Set HTTP API request**

Set the msg.url to request Domoticz to update the device with IDX=113.  
The flow context "domip" holds the Domoticz system IP address.

```
const temp = msg.payload;
const idx = flow.get("device").idx;

// Define the url to set the temperature of the Domoticz device
// /json.htm?type=command&param=udevice&idx=IDX&nvalue=0&svalue=TEMP

msg.url = "http://" + flow.get("domip") + ":8080/json.htm?type=command&param=udevice&idx=" + idx +
"&nvalue=0&svalue=" + temp;
node.warn(msg.url);

return msg;
```

## Multiple Devices

To get & set the value for multiple Domoticz devices a JSON array is defined in the Node-RED flow change node “Set Devices”. This array is used by the function node “Get Devices and Set Value”.

For this example, get 3 Homematic datapoints and set the value of three Domoticz devices.

Extract from the Homematic full statelist obtained using HTTP XML-API request:

```
<datapoint ise_id="1567" name="HmIP-RF.000A18A9A64DAC:1.ACTUAL_TEMPERATURE" operations="5" timestamp="1583752872" valueunit="" valuetype="4" value="22.800000" type="ACTUAL_TEMPERATURE"/>

<datapoint ise_id="1576" name="HmIP-RF.000A18A9A64DAC:1.LEVEL" operations="7" timestamp="1583752872" valueunit="" valuetype="4" value="0.310000" type="LEVEL"/>

<datapoint ise_id="1584" name="HmIP-RF.000A18A9A64DAC:1.SET_POINT_TEMPERATURE" operations="7" timestamp="1583752872" valueunit="°C" valuetype="4" value="21.000000" type="SET_POINT_TEMPERATURE"/>
```

The Node-RED change node to configure the datapoints in JSON format (array with several key:value pairs) and the three Domoticz devices (from the Domoticz WebUI > Setup > Devices).

The screenshot shows the Node-RED interface with two main sections. On the left, the "Edit change node > JSON editor" pane displays a JSON array with three objects, each representing a datapoint with its ID, type, and index:

```

1 [ 
2   {
3     "datapoint": 1567,
4     "type": "ACTUAL_TEMPERATURE",
5     "idx": 113
6   },
7   {
8     "datapoint": 1576,
9     "type": "LEVEL",
10    "idx": 114
11  },
12  {
13    "datapoint": 1584,
14    "type": "SET_POINT_TEMPERATURE",
15    "idx": 115
16  }
17 ]

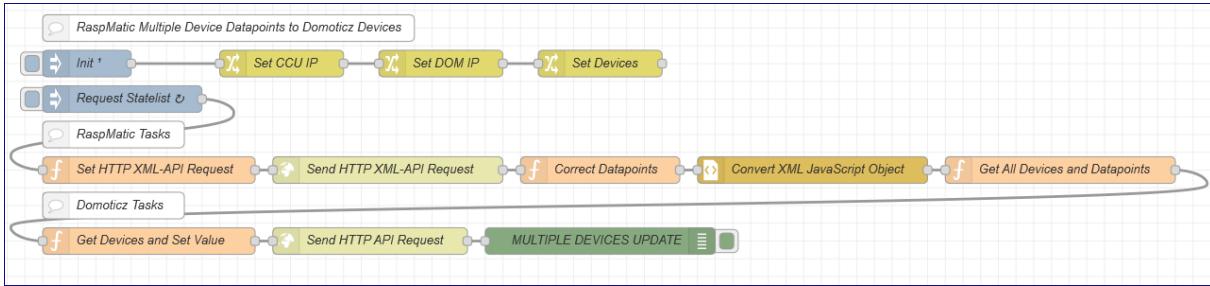
```

On the right, the "Devices" table lists three Domoticz devices with their details:

Idx	Hardware	ID	Unit	Name	Type	SubType	
115	VirtualDevices	140C3	1	MakeLab Thermostat Setpoint	Temp	LaCrosse TX3	6.0 C
114	VirtualDevices	00082114	1	MakeLab Thermostat Valve	General	Percentage	0%
113	VirtualDevices	140C1	1	MakeLab Thermostat Temperature	Temp	LaCrosse TX3	21.8 C

A note below the table states: "Homematic Datapoint = Domoticz Idx - see left."

## Node-RED



### Function Node Get Devices and Datapoints

Same as for the single device.

### Function Node Get Devices and Set Value

```

/*
The purpose of this function is to set the value for multiple domoticz devices.
This function does not return a value as msg are send during device json array loop.
*/
const DEBUG = true;
// get all the datapoints by device
// device:"2530",datapoint:{id:2532,type:"CONFIG_PENDING",value:false,name:"HmIP-"
RF.0000DA498D5859:0.CONFIG_PENDING}
datapoints = flow.get("datapoints");
// devices: [{"datapoint": 1567,type: "ACTUAL_TEMPERATURE",idx: 113}, ...]
devices = flow.get("devices");
//
devices.forEach(function(device){
    datapoints.forEach(function(datapoint){
        if (device.datapoint == datapoint.datapoint.id) {
            idx = device.idx;
            value = datapoint.datapoint.value;
            if (idx == "114") value = value * 100;
            msg.url = "http://" + flow.get("domip") + ":8080/json.htm?type=command&param=udevice&idx="
+ idx + "&value=0&svalue=" + value;
            node.send(msg);
            if (DEBUG) node.warn("URL: " + msg.url);
        }
    });
});
}

```

## Plugin Considerations

Initially started testing by adding Homematic IP devices to the CCU, determined the required device, channel and datapoint id's, triggered HTTP REST requests (XML-API) from Domoticz and parsed the CCU XML formatted HTTP response to newly created Domoticz devices as timer triggered by a dzVents script.

These steps are defined next.

Then considered, that for additional devices, it requires effort to setup devices and dzVents scripts ... which could get unclear or even confusing.

A better approach would be, to develop for each of the Homematic IP devices a Domoticz Python Plugin.

After various iterations, developed plugins for the Homematic IP devices Pluggable Switch and Meter (HMIP-PSM) and Radiator Thermostat (HMIP-eTRV-2).

These can be found in my GitHub [repositories](#) starting with domoticz-plugin-hmip-<short description>, i.e., domoticz-plugin-hmip-psm, domoticz-plugin-hmip-etrv-2.

More under consideration ...

# Addon CCU-Jack

## Information

The [CCU-Jack](#) Addon is used to communicate between the CCU and Domoticz.

**CCU-Jack** enables simple access to device datapoints both read & write via

- HTTP REST-API or
- MQTT-API - MQTT-Server V3.1.1
- 

using JSON data format – easier to use than the XML-API.

The CCU-Jack addon is an alternative to the XML-API addon.

### Installation

The installation of CCU-Jack is via the Homematic WebUI > Home page > Settings > Control panel > Additional Software.

After installation:

Add open ports to the CCU Firewall as described in the installation manual.

Access Homematic WebUI > Home page > Settings > Control panel > Configure firewall:

The ports opened are: 2121;2122;1883;8883;8181.

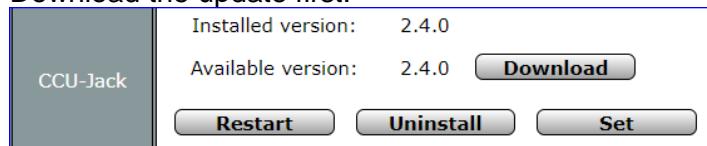
To access CCU-Jack REST-API port 2121 is used.

The option *set* opens a WebUI with the CCU-Jack Navigator to view devices, change configuration and set access rights.

### Update

CCU-Jack updates are done via the Homematic WebUI > Home page > Settings > Control panel > Additional Software.

Download the update first:

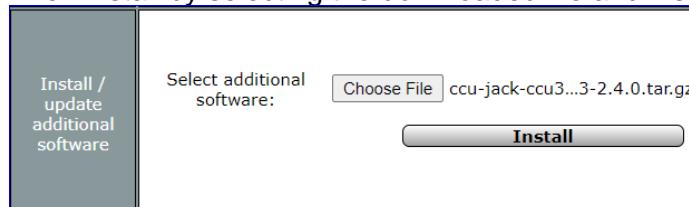


From the web site download from the Assets section, like

▼ Assets 9

[ccu-jack-ccu3-rm-rp2+3-2.4.0.tar.gz](#)

Then install by selecting the downloaded file and install:



## REST-API

Some examples on how to subscribe and publish messages to the CCU-JACK REST-API.

The tests mainly use a Homematic IP thermostat device Hm-IP-eTRV-2 as the device enables to get or set data via MQTT subscribe to messages or publish messages.

The examples start with commands issued via Domoticz automation event dzVents scripts, followed by examples from the command-line interface (CLI), Node-RED flow & Python 3.

## Domoticz Examples

The Domoticz examples are automation events with dzVents scripts.

The next two examples show how to Get a Value and Set the State of a Homematic IP device.

The examples can be used as a generic approach to get or set devices values for both Homematic IP as well as Domoticz devices.

### Get Homematic Device Value & Set Domoticz Device Value

#### Purpose

Read in regular intervals the datapoint ACTUAL\_TEMPERATURE of a thermostat HmIP-eTRV and update the Domoticz device type Temp, subtype LaCrosse TX3 with the actual temperature.

#### dzVents script triggers

timer, httpResponses

```
--[[[

    ccujack-thermostat-temperature.dzvents
    Read in regular intervals the datapoint ACTUAL_TEMPERATURE of a thermostat HmIP-eTRV.
    Assign the value to a Domoticz Virtual Sensor Temperature:
    Idx, Hardware, Name, Type, SubType, Data
    31,VirtualSensors,MakeLab Temperature, Temp, LaCrosse TX3, 17.8 C
    Dependencies: Homematic CCU Addon CCU-Jack
    20210314 rwbl
]]--


-- url http xml-api request to get the value of a datapoint with ise_id. The ise_id is set in the
function openURL.
-- http response example: {"ts":1615733608402,"v":17,"s":0}
local URL_CCUJACK = 'http://ccu-ip:2121/device/000A18A9A64DAC/1/ACTUAL_TEMPERATURE/~pv';

-- define the unique (across all dzVents) callback
local RES_CCUJACK = "RESCCUJACKTHERMOSTATTEMPERATURE";

-- domoticz thermostat temperature device
local IDX_THERMOSTAT_ACTUAL_TEMPERATURE = 31;

-- Timer
local TIMERRULE = 'every 10 minutes'
-- local TIMERRULE = 'every minute'

return {
    on = { timer = { TIMERRULE }, httpResponses = { RES_CCUJACK } },
    execute = function(domoticz, item)
        -- check if the item is a timer, then request information
        if (item.isTimer) then
            domoticz.openURL({url = URL_CCUJACK, method = 'GET', callback = RES_CCUJACK})
        end

        -- check if the item is a httpresponse from the openurl callback
        if (item.isHTTPResponse) then
            if (item.statusCode == 200) then
                -- domoticz.log(item.data);
                -- Select the callback - in this case there is only one (but for any next developments
here might be more)
                if (item.callback == RES_CCUJACK) then

```

```
        domoticz.devices(IDX_THERMOSTAT_ACTUAL_TEMPERATURE).updateTemperature(item.json.v);
        domoticz.log(item.json.v)
    end
else
    domoticz.log('[ERROR] Request:' .. item.statusText, domoticz.LOG_ERROR)
end
end
}
}
```

## Get Domoticz Device Value & Set Homematic Device Value

### Purpose

Get the setpoint of a Domoticz thermostat device and set the setpoint of a Homematic thermostat device.

### dzVents script triggers

devices, httpResponses

```
--[[
  ccujack-thermostat-setpoint.dzvents
  Dependencies: Homematic CCU Addon CCU-Jack
  20210314 rwbl
]]-- 

-- url http xml-api request to put the value of a datapoint with ise_id. The ise_id is set in the
function openURL.
local URL_CCUJACK = 'http://ccu-ip:2121/device/#SERIALNR#/1/SET_POINT_TEMPERATURE/~pv';

-- domoticz thermostat setpoint device
local makelab = { serialnr = '000A18A9A64DAC', spidx = 30, responsesp = 'RESCCUJACK-TH-MAKELAB-SP', }

-- Put a new setpoint for a Homematic thermostat
-- Example: putDatapoint(domoticz, makelab.serialnr, 'SET_POINT_TEMPERATURE', response, item.setPoint)
local function putDatapoint(domoticz, serialnr, response, setpointnew)
  local url = URL_CCUJACK:gsub('#SERIALNR#', serialnr)
  local setpointData = { v = setpointnew }
  domoticz.openURL({url = url, method = 'PUT', callback = response, postData = setpointData});
end

return {
  on = { devices = { makelab.spidx }, httpResponses = { makelab.responsesp } },
  execute = function(domoticz, item)
    if (item.isDevice) then
      putDatapoint(domoticz, makelab.serialnr, response, item.setPoint)
    end
    if item.isHTTPResponse then
      if item.statusCode == 200 then
        domoticz.log(item.json)
      else
        domoticz.log('[ERROR] Request:' .. item.statusText, domoticz.LOG_ERROR)
      end
    end
  end
}
```

## CLI CURL Examples

The HTTP REST-API example requests are submitted, on the Raspberry Pi Domoticz Development system, using the command-line tool “curl” (Client URL) for transferring data using the HTTP network protocols.

After the CLI examples, some examples using Node-RED.

## Get Thermostat Setpoint

Request the thermostat setpoint temperature, which is returned as JSON object.

```
curl http://ccu-ip:2121/device/000A18A9A64DAC/1/SET_POINT_TEMPERATURE/~pv
{"ts":1614956072812,"v":21.0,"s":0}
```

It is also possible to get the setpoint temperature as a plain text value instead JSON object:

```
curl http://ccu-ip:2121/device/000A18A9A64DAC/1/SET_POINT_TEMPERATURE/~pv?format=simple
21
```

## Set Thermostat Setpoint

```
curl -X PUT -d '{"v":7}' http://ccu-ip:2121/device/000A18A9A64DAC/1/SET_POINT_TEMPERATURE/~pv
```

### Notes

X - option to be used for request command

d - option to be used to put data on remote url

It is also possible to write the new setpoint direct as parameter (writepv=value) in the URL:

```
curl http://ccu-ip:2121/device/000A18A9A64DAC/1/SET_POINT_TEMPERATURE/~pv?writepv=20.5
```

Turn the thermostat OFF by writing value 0:

```
curl http://ccu-ip:2121/device/000A18A9A64DAC/1/SET_POINT_TEMPERATURE/~pv?writepv=0
```

## CURL Info Option (-i)

Like previous but with info option to lookup status response.

```
curl -i -X PUT -d '{"v":7}' http://ccu-ip:2121/device/000A18A9A64DAC/1/SET_POINT_TEMPERATURE/~pv
HTTP/1.1 200 OK
Content-Length: 0
Content-Type: application/json
Date: Sat, 06 Mar 2021 09:14:19 GMT
```

## CURL Verbose Option (-v)

Like previous but with verbose option to lookup status.

```
curl -v -X PUT -d '{"v":7}' http://ccu-ip:2121/device/000A18A9A64DAC/1/SET_POINT_TEMPERATURE/~pv
* Expire in 0 ms for 6 (transfer 0xc458b0)
*   Trying ccu-ip...
* TCP_NODELAY set
* Expire in 200 ms for 4 (transfer 0xc458b0)
* Connected to ccu-ip (ccu-ip) port 2121 (#0)
> PUT /device/000A18A9A64DAC/1/SET_POINT_TEMPERATURE/~pv HTTP/1.1
> Host: ccu-ip:2121
> User-Agent: curl/7.64.0
> Accept: */
> Content-Length: 7
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 7 out of 7 bytes
< HTTP/1.1 200 OK
< Content-Length: 0
< Content-Type: application/json
< Date: Sat, 06 Mar 2021 09:10:00 GMT
<
* Connection #0 to host ccu-ip left intact
```



## Get List of Devices

Get the list of devices configured in the CCU.

```
curl http://ccu-ip:2121/device
```

```
{
  "description": "CCU3 Devices",
  "identifier": "device",
  "title": "Devices",
  "~links": [
    {"rel": "device", "href": "001558A99D5A78", "title": "Garage"},
    {"rel": "device", "href": "0001D3C99C6AB3", "title": "MakeLab PSM"},
    {"rel": "device", "href": "000B1BE98D94DE", "title": "RC-Wohnzimmer"},
    {"rel": "device", "href": "000A18A9A64DAC", "title": "MakeLab"},
    {"rel": "device", "href": "00201A49952CB8", "title": "Dusche"},
    {"rel": "device", "href": "000A1A49A0D878", "title": "Esszimmer"},
    {"rel": "device", "href": "0000DA498D5859", "title": "Briefkasten"},
    {"rel": "device", "href": "HmIP-RCV-1", "title": "HmIP-RCV-50 HmIP-RCV-1"},
    {"rel": "device", "href": "BidCoS-RF", "title": "HM-RCV-50 BidCoS-RF"},
    {"rel": "device", "href": "00201A49952CB1", "title": "Wohnzimmer-1"},
    {"rel": "device", "href": "002018A99D2097", "title": "Bad"},
    {"rel": "device", "href": "000A1A49A0D8A5", "title": "Flur"},
    {"rel": "device", "href": "00201A499D76F8", "title": "Wohnzimmer-2"},
    {"rel": "root", "href": "...", "title": "Root"}
  ]
}
```

## Get Device

(Device name MakeLab, serial number 000A18A9A64DAC, response JSON object extract)

```
curl http://ccu-ip:2121/device/000A18A9A64DAC
```

```
{
  "address": "000A18A9A64DAC",
  "aesActive": 1,
  "availableFirmware": "0.0.0",
  "children": ["000A18A9A64DAC:0", "000A18A9A64DAC:1", "000A18A9A64DAC:2", "000A18A9A64DAC:3",
               "000A18A9A64DAC:4", "000A18A9A64DAC:5", "000A18A9A64DAC:6", "000A18A9A64DAC:7"],
  "direction": 0, "firmware": "2.2.8", "flags": 1, "group": "", "identifier": "000A18A9A64DAC",
  "index": 0, "interface": "", "interfaceType": "HmIP-RF", "linkSourceRoles": "", "linkTargetRoles": "",
  "paramsets": ["MASTER", "SERVICE"],
  "parent": "", "parentType": "", "rfAddress": 5847832, "roaming": 0, "rxMode": 11, "team": "",
  "teamChannels": null, "teamTag": "",
  "title": "MakeLab", "type": "HmIP-eTRV-2", "version": 4,
  "~links": [
    {"rel": "channel", "href": "$MASTER", "title": "MakeLab - $MASTER"}, {"rel": "channel", "href": "2", "title": "HmIP-eTRV-2 000A18A9A64DAC:2"}, ...
    {"rel": "devices", "href": "...", "title": "Devices"}]
}
```

## Get Device Datapoint

```
curl http://ccu-ip:2121/device/000A18A9A64DAC/1/ACTUAL_TEMPERATURE

{
  "control": "HEATING_CONTROL_HMIP.TEMPERATURE",
  "default": 0, "flags": 1,
  "id": "ACTUAL_TEMPERATURE", "identifier": "ACTUAL_TEMPERATURE",
  "maximum": 3276.7, "minimum": -3276.8,
  "mqttStatusTopic": "device/status/000A18A9A64DAC/1/ACTUAL_TEMPERATURE",
  "operations": 5, "tabOrder": 0,
  "title": "HmIP-eTRV-2 000A18A9A64DAC:1 - ACTUAL_TEMPERATURE",
  "type": "FLOAT", "unit": "",
  "~links": [
    {"rel": "channel", "href": "..", "title": "HmIP-eTRV-2 000A18A9A64DAC:1" },
    {"rel": "service", "href": "~pv", "title": "PV Service"}
  ]
}
```

*Device name MakeLab.*

## Get List of Programs

```
curl http://ccu-ip:2121/program

{
  "description": "Programs of the ReGaHss",
  "identifier": "program",
  "title": "Programs",
  "~links": [
    {"rel": "program", "href": "2762", "title": "Garage Door Monitor"},
    {"rel": "program", "href": "2540", "title": "Postbox Monitor"},
    ...
    {"rel": "root", "href": "..", "title": "Root"}
  ]
}
```

*Response JSON object extract only.*

## Get Program Information

```
curl http://ccu-ip:2121/program/2787

{
  "active": true,
  "description": "Check daily the battery state LOW_BAT to Domoticz alert device.",
  "identifier": "2787",
  "mqttGetTopic": "program/get/2787",
  "mqttSetTopic": "program/set/2787",
  "mqttStatusTopic": "program/status/2787",
  "title": "Battery Check",
  "visible": true,
  "~links": [
    {"rel": "programs", "href": "..", "title": "Programs"},
    {"rel": "service", "href": "~pv", "title": "PV Service"}
  ]
}
```

## Program Run Battery Check

Run a program (ID 2787) to check the battery state of all devices and update system variable (ID 2799).

In the JSON data object, set name “v” (=value) to true to trigger running the program

```
curl -X PUT -d '{"v":true}' http://ccu-ip:2121/program/2787/~pv
```

There is no response.

To check the result, get the value of the system variable with ID 2799 or get the timestamp of the program last run (see next).

## Program Get Timestamp Last Run

```
curl http://ccu-ip:2121/program/2787/~pv
```

```
{"ts":1615042593000,"v":false,"s":0}
```

*ts - Timestamp program last run in UNIX format milliseconds.*

## Program Get System Variable

Get the value of the system variable with ID 2799, which has been changed by the previous program (ID 2787).

```
curl http://ccu-ip:2121/sysvar/2799/~pv
```

```
{"ts":1614954055000,"v":"OK 06.03 15:57","s":0}
```

### Note

Use the parameter `format=simple` to get the sysvar value as plain text:

```
curl http://ccu-ip:2121/sysvar/2799/~pv?format=simple
OK 19.08 00:00
```

## Get Root Domain

```
curl http://ccu-ip:2121/
{
  "description": "Root of the CCU-Jack VEAP server",
  "identifier": "root",
  "title": "Root",
  "~links": [
    {"rel": "domain","href": "sysvar","title": "System variables"},
    {"rel": "domain","href": "program","title": "Programs"},
    {"rel": "domain","href": "room","title": "Rooms"},
    {"rel": "domain","href": "function","title": "Functions"},
    {"rel": "domain","href": "~vendor","title": "Vendor Information"},
    {"rel": "domain","href": "device","title": "Devices"}
  ]
}
```

## Get System Variables

```
curl http://ccu-ip:2121/sysvar

{
  "description": "System variables of the ReGaHss",
  "identifier": "sysvar",
  "title": "System variables",
  "~links": [
    {"rel": "sysvar","href": "2799","title": "BatteryCheck"},  

    {"rel": "sysvar","href": "3032","title": "DutyCycle"},  

...
    {"rel": "root","href": "...","title": "Root"}  

  ]
}
```

## Get System Variable Properties

### System variable ID 2799

```
curl http://ccu-ip:2121/sysvar/2799

{
  "description": "Daily Device Battery Check Status",
  "identifier": "2799",
  "mqttGetTopic": "sysvar/get/2799",
  "mqttSetTopic": "sysvar/set/2799",
  "mqttStatusTopic": "sysvar/status/2799",
  "operations": 7,
  "title": "BatteryCheck",
  "type": "STRING",
  "unit": "",
  "~links": [
    {"rel": "sysvars","href": "...","title": "System variables"},  

    {"rel": "service","href": "~pv","title": "PV Service"}  

  ]
}
```

## Get System Variable

Two examples: ID 2799 = BatteryCheck, ID 3032 = DutyCycle

```
curl http://ccu-ip:2121/sysvar/2799/~pv  
  

>{"ts":1614954055000,"v":"OK 06.03 15:57","s":0}  
  

curl http://ccu-ip:2121/sysvar/3032/~pv  
  

>{"ts":1615126980000,"v":2,"s":0}
```

## Set System Variable

Set the value of a system variable string.

```
curl -X PUT -d '{"v":"Hello World"}' http://ccu-ip:2121/sysvar/5097/~pv  
  

Run curl again with get command to check if content is set.  

The result should contain v = "Hello World"  

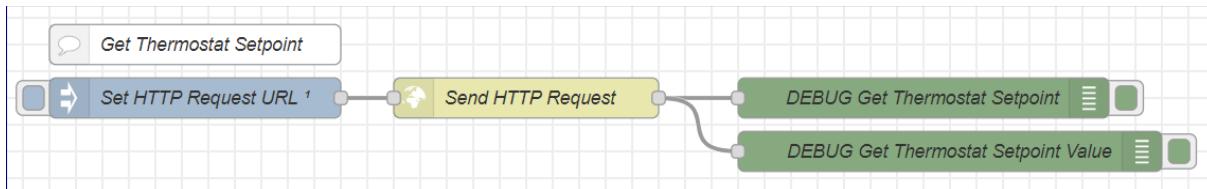
curl http://ccu-ip:2121/sysvar/5097/~pv  
  

>{"ts":1622121875000,"v":"Hello World","s":0}
```

## Node-RED Examples

Next some simple examples showing the concept of how to use Node-RED with the CCU-Jack REST-API.

## Get Thermostat Setpoint



The inject node sets the message url (String):

Name	Set HTTP Request URL
msg.url	= http://192.168.1.70:2121/device/000A18A9A64DAC/1/SET_POINT_TEMPERATURE/~pv

The http-request node submits the URL.

Edit http request node

Delete      Cancel      Done

**Properties**

Method: GET    URL: http://    Payload: Ignore

Enable secure (SSL/TLS) connection     Use authentication     Enable connection keep-alive     Use proxy

Return: a parsed JSON object    Name: Send HTTP Request

Tip: If the JSON parse fails the fetched string is returned as-is.

Set the property **Method** to GET and **Return** to a parsed JSON object to read, from the message payload, the name "v" holding the thermostat setpoint value.

The debug nodes log the HTTP response.

<pre>17/05/2021, 09:46:27 node: DEBUG Get Thermostat Setpoint msg.payload : Object ▶ { ts: 1621237148514, v: 4.5, s: 0 }  17/05/2021, 09:46:27 node: DEBUG Get Thermostat Setpoint Value msg.payload.v : number 4.5</pre>	<p>Thermostat setpoint is msg.payload.v = 4.5.</p>
---	--

## Node-RED Flow Source

```
[{"id": "cf92b259.38d34", "type": "inject", "z": "f814089e.24ee78", "name": "Set HTTP Request URL", "props": [{"p": "url", "v": "http://ccu-ip:2121/device/000A18A9A64DAC/1/SET_POINT_TEMPERATURE/~pv", "vt": "str"}, {"p": "topic", "vt": "str"}], "repeat": "", "crontab": "", "once": true, "onceDelay": 0.1, "topic": "", "payloadType": "str", "x": 150, "y": 60, "wires": [{"id": "3fa5c5ac.baa60a"}]}, {"id": "e252e5f8.460508", "type": "comment", "z": "f814089e.24ee78", "name": "Get Thermostat Setpoint", "info": "HTTP Response:\n{\n  \"ts\": 1621237148514,\n  \"v\": 4.5,\n  \"s\": 0\n}\n", "x": 300, "y": 100, "wires": []}, {"id": "3fa5c5ac.baa60a", "type": "http request", "z": "f814089e.24ee78", "name": "Send HTTP Request", "method": "GET", "ret": "obj", "paytoqs": "ignore", "url": "", "tls": "", "persist": false, "proxy": "", "authType": "", "x": 300, "y": 120, "wires": [{"id": "adb37cdd.e3fa9", "x": 450, "y": 120}], {"id": "adb37cdd.e3fa9", "type": "debug", "z": "f814089e.24ee78", "name": "DEBUG Get Thermostat Setpoint", "active": true, "tosidebar": true, "console": false, "tostatus": false, "complete": "payload", "targetType": "msg", "statusVal": "", "statusType": "auto", "x": 450, "y": 140, "wires": []}, {"id": "2ccf59a0.ae8736", "type": "debug", "z": "f814089e.24ee78", "name": "DEBUG Get Thermostat Setpoint Value", "active": true, "tosidebar": true, "console": false, "tostatus": false, "complete": "payload.v", "targetType": "msg", "statusVal": "", "statusType": "auto", "x": 450, "y": 160, "wires": []}]]
```

## Set Thermostat Setpoint (Inject)



The inject node sets the message url (String) and the payload (JSON):

Name	Set HTTP Request URL
msg. url	= http://192.168.1.70:2121/device/000A18A9A64DAC/1/SET_POINT_TEMPERATURE/~pv
msg. payload	= {} {"v":17.5}

The http-request node submits the URL.

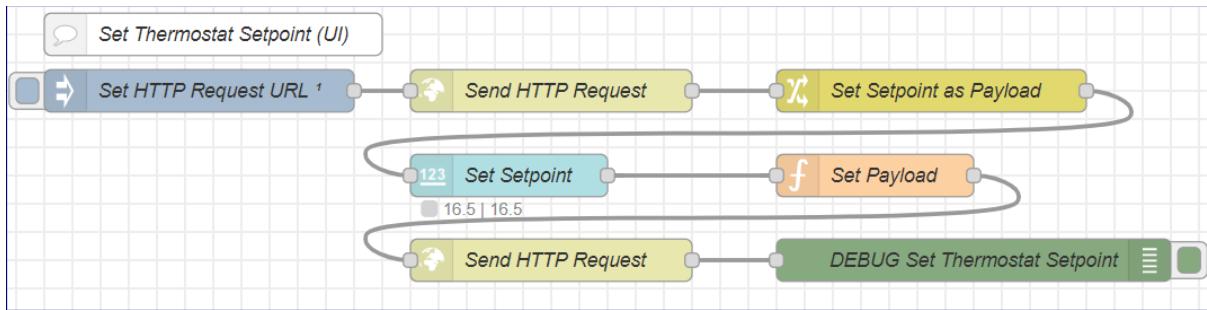
Edit http request node	
<input type="button" value="Delete"/> <input type="button" value="Cancel"/> <input type="button" value="Done"/> <b>Properties</b> <input checked="" type="radio"/> Method: PUT <input checked="" type="radio"/> URL: http:// <input type="checkbox"/> Enable secure (SSL/TLS) connection <input type="checkbox"/> Use authentication <input type="checkbox"/> Enable connection keep-alive <input type="checkbox"/> Use proxy <input type="button" value="Return"/> a UTF-8 string <input type="button" value="Name"/> Send HTTP Request	Set the property <b>Method</b> to PUT and the <b>Return</b> to a UTF-8 string.

The debug nodes log the HTTP response BUT in this case there is no response

### Node-RED Flow Source

```
[{"id": "4bb4ecce.bca224", "type": "inject", "z": "f814089e.24ee78", "name": "Set HTTP Request URL", "props": [{"p": "url", "v": "http://ccu-ip:2121/device/000A18A9A64DAC/1/SET_POINT_TEMPERATURE/~pv", "vt": "str"}, {"p": "payload"}], "repeat": "", "crontab": "", "once": true, "ondelay": "0.1", "topic": "", "payload": "{\"v\":17.5}", "payloadType": "json", "x": 150, "y": 240, "wires": [{"id": "9bb46732.0cd9a8"}]}, {"id": "48d4cd52.b19fe4", "type": "comment", "z": "f814089e.24ee78", "name": "Set Thermostat Setpoint", "info": "Message payload to set the thermostat setpoint to 17.5 \nC:\n{\\"v\\":17.5}\n", "x": 140, "y": 200, "wires": []}, {"id": "9bb46732.0cd9a8", "type": "http request", "z": "f814089e.24ee78", "name": "Send HTTP Request", "method": "PUT", "ret": "txt", "paytoqs": "ignore", "url": "", "tls": "", "persist": false, "proxy": "", "authType": "", "x": 400, "y": 240, "wires": [{"id": "291f8ff5.d5787"}]}, {"id": "291f8ff5.d5787", "type": "debug", "z": "f814089e.24ee78", "name": "DEBUG Set Thermostat Setpoint", "active": true, "tosidebar": true, "console": false, "tostatus": false, "complete": "payload", "targetType": "msg", "statusVal": "", "statusType": "auto", "x": 700, "y": 240, "wires": []}]
```

## Set Thermostat Setpoint (UI)



The ui\_numeric node enables to set the setpoint.

<div style="border: 1px solid #ccc; padding: 10px;"> <p>Edit numeric node</p> <p><b>Properties</b></p> <p>Group: [Homematic] Thermostat</p> <p>Size: auto</p> <p>Label: Setpoint</p> <p>Tooltip: optional tooltip</p> <p>Value Format: {{value}}</p> <p>Range: min 0 max 25 step 0.5</p> <p>Wrap value from max to min and min to max. <input type="checkbox"/></p> <p>If msg arrives on input, pass through to output: <input type="checkbox"/></p> <p>When changed, send:</p> <ul style="list-style-type: none"> <li>Payload: Current value</li> <li>Topic: msg.topic</li> <li>Name: Set Setpoint</li> </ul> </div>	<p>The property “If msg arrives on input...” must be disabled to avoid changing the setpoint.</p> <p>The steps are in 0.5 °C.</p> <p>If setpoint 0 is set, the thermostat is switched Off – if in Manual Mode set in the Homematic WebUI.</p>
---	---

The function node “Set Payload” defined the message payload used by the http-request node.

```

// Define the message payload for the HTTP REST-API request to set the thermostat setpoint

msg.url = "http://ccu-ip:2121/device/000A18A9A64DAC/1/SET_POINT_TEMPERATURE/~pv";

var setpoint = msg.payload;
msg.payload = {"v":setpoint};

return msg;

```

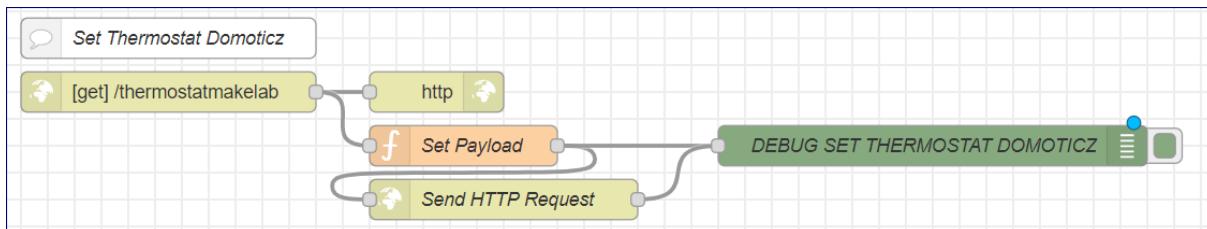
To set the thermostat new value, the flow is the same as previous described with “Set Thermostat Setpoint (Inject).

## Node-RED Flow Source

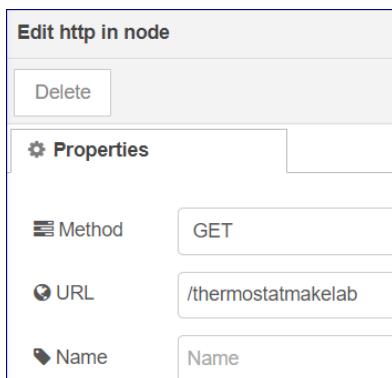
## Set Thermostat Setpoint (Domoticz)

Set the thermostat setpoint from a Domoticz Thermostat Device using a Node-RED HTTP listener node.

Method	GET
URL	/thermostatmakelab
URL Parameter	setpoint=VALUE
Test from Browser:	
<i>HTTP Request</i> http://domoticz-ip:1880/thermostatmakelab?setpoint=12.5	
<i>HTTP Response</i> {"setpoint":"12.5"}	



The flow makes use of a http in node listening to a HTTP GET request with query string "/thermostatmakelab". The parameter used is setpoint, which is not defined in this node.



The http-response node is required to ensure an HTTP end-to-end request.

The function node defines the payload for the HTTP REST-API request:

```

// Define the message payload for the HTTP REST-API request to set the thermostat setpoint

msg.url = "http://ccu-ip:2121/device/000A18A9A64DAC/1/SET_POINT_TEMPERATURE/~pv";

// The setpoint is taken from the message payload
// http://domoticz-ip:1880/thermostatmakelab?setpoint=12.5
var setpoint = Number(msg.req.query.setpoint);
msg.payload = {"v":setpoint};

return msg;
  
```

To set the thermostat new value, the flow is the same as previous described with "Set Thermostat Setpoint (Inject).

## Node-RED Flow Source

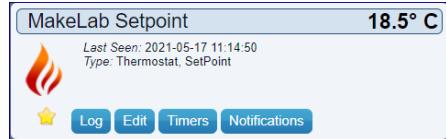
```
[{"id": "159e0051.f0c41", "type": "http_in", "z": "f814089e.24ee78", "name": "", "url": "/thermostatmakelab", "method": "get", "upload": false, "swaggerDoc": "", "x": 150, "y": 580, "wires": [{"df335e9d.622fd", "4b6a5494.df03dc"}]}, {"id": "9a374e38.d2478", "type": "debug", "z": "f814089e.24ee78", "name": "DEBUG SET THERMOSTAT DOMOTICZ", "active": true, "tostidebar": true, "console": false, "tostatus": false, "complete": "payload", "targetType": "msg", "statusVal": "", "statusType": "auto", "x": 720, "y": 620, "wires": []}, {"id": "df335e9d.622fd", "type": "http_response", "z": "f814089e.24ee78", "name": "", "statusCode": "", "headers": {}, "x": 350, "y": 580, "wires": []}, {"id": "fa080d86.e15f", "type": "comment", "z": "f814089e.24ee78", "name": "Set Thermostat Domoticz", "info": "\n", "x": 150, "y": 540, "wires": []}, {"id": "4b6a5494.df03dc", "type": "function", "z": "f814089e.24ee78", "name": "Set Payload", "func": "// Define the message payload for the HTTP REST-API request to set the thermostat setpoint\n\nmsg.url = \"http://ccu-ip:2121/device/000A18A9A64DAC/1/SET_POINT_TEMPERATURE/~pv\";\n\n// The setpoint is taken from the message payload\n\nhttp://domoticz-ip:1880/thermostatmakelab?setpoint=12.5\n\nvar setpoint = Number(msg.req.query.setpoint);\n\nmsg.payload = {\n    \":setpoint\"\n}\n\nreturn msg;\n", "outputs": 1, "noerr": 0, "initialize": "", "finalize": "", "libs": [], "x": 370, "y": 620, "wires": [{"f5675ac6.6ef428", "9a374e38.d2478"}]}, {"id": "f5675ac6.6ef428", "type": "http_request", "z": "f814089e.24ee78", "name": "Send HTTP Request", "method": "PUT", "ret": "txt", "paytoqs": "ignore", "url": "", "tls": "", "persist": false, "proxy": "", "authType": "", "x": 400, "y": 660, "wires": [{"9a374e38.d2478"}]}]
```

## Domoticz Configuration

### Device Thermostat MakeLab (IDX=30)

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
30	VirtualSensors	001406E	1	MakeLab Setpoint	Thermostat	SetPoint	18.5

### Device Thermostat MakeLab Widget



### dzVents to handle device Thermostat MakeLab setpoint changes

Trigger: Device

```
--[[
    ccujack-thermostat-node-red.dzvents
    Dependencies: Homematic CCU3 addon CCU-Jack
]]--
-- domoticz thermostat setpoint device
-- url http request to node-red http in node to set the new setpoint
local makelab = {
    url = 'http://domoticz-ip:1880/thermostatmakelab?setpoint=#SETPOINT#',
    idx = 30,
    setpoint = -1,
    response = 'RESCCUJACK-THERMOSTATMAKELAB'}

-- Put a new setpoint for a Homematic thermostat
-- Example: putDatapoint(domoticz, makelab.serialnr, 'SET_POINT_TEMPERATURE', response, item.setPoint)
local function setSetpoint(domoticz, device)
    local url = device.url:gsub('#SETPOINT#', device.setpoint)
    domoticz.log(url)
    domoticz.openURL({url = url, method = 'GET', callback = device.response});
end

return {
    on = { devices = { makelab.idx }, httpResponses = { makelab.response } },
    execute = function(domoticz, item)
        if (item.isDevice) then
            makelab.setpoint = item.setPoint
            setSetpoint(domoticz, makelab)
        end
        if item.isHTTPResponse then
            if item.statusCode == 200 then
                domoticz.log(item.json)
            else
                domoticz.log('[ERROR] Request:' .. item.statusText, domoticz.LOG_ERROR)
            end
        end
    end
}
```

## Domoticz Log

Thermostat setpoint set to 18.5 °C.

```
2021-05-17 11:14:50.834 Status: dzVents: Info: Handling events for: "MakeLab Setpoint", value: "18.50"
2021-05-17 11:14:50.834 Status: dzVents: Info: ----- Start internal script: ccujack-thermostat-node-red: Device: "MakeLab Setpoint (VirtualSensors)", Index: 30
2021-05-17 11:14:50.834 Status: dzVents: Info: http://domoticz-ip:1880/thermostatmakelab?setpoint=18.5
2021-05-17 11:14:50.834 Status: dzVents: Info: ----- Finished ccujack-thermostat-node-red
2021-05-17 11:14:50.834 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
2021-05-17 11:14:50.895 Status: dzVents: Info: Handling httpResponse-events for: "RESCCUJACK-THERMOSTATMAKELAB"
2021-05-17 11:14:50.895 Status: dzVents: Info: ----- Start internal script: ccujack-thermostat-node-red: HTTPResponse: "RESCCUJACK-THERMOSTATMAKELAB"
2021-05-17 11:14:50.895 Status: dzVents: Info: {[{"setpoint"}]="18.5"}
2021-05-17 11:14:50.895 Status: dzVents: Info: ----- Finished ccujack-thermostat-node-red
```

## Python 3 Script Examples

The next Python 3 script examples showing how to use GET & SET REST-API requests.  
The scripts are developed on a Raspberry Pi 4B running Raspberry OS with Python 3.7.3 [GCC 8.3.0].

The Python HTTP library [requests](#) (2.21.0) is used.

Installed with:

```
sudo pip3 install requests
```

### Get Device Value

Get the value of a system variable and the actual temperature of a thermostat device.

```
#!/usr/bin/python3

## ccujack-rest-get.py
## CCU-Jack REST-API Test Parsing JSON Response from get request.
## Run: python3 ccujack-rest-get.py

## System Variable DutyCycle, ID 3032
## HTTP Request: http://ccu-ip:2121/sysvar/3032/~pv
## HTTP Response: {"ts":1615126980000,"v":2,"s":0}

## Device Variable, MakeLab Thermostat, SerienNr 000A18A9A64DAC
## HTTP Request: http://ccu-ip:2121/device/000A18A9A64DAC/1/ACTUAL_TEMPERATURE/~pv
## HTTP Response: {"ts":1615201525736,"v":21.1,"s":0}

VERSION = 'ccujack-rest-get v20210518'

## Set the CCUIP URL for the REST-API request
URL_GET_SYSVAR = 'http://ccu-ip:2121/sysvar/3032/~pv'
URL_GET_DEVICE = 'http://ccu-ip:2121/device/000A18A9A64DAC/1/ACTUAL_TEMPERATURE/~pv'

import requests
from requests.exceptions import HTTPError
from datetime import datetime

print(f'{VERSION}')
try:
    # Sysvar
    response = requests.get(URL_GET_SYSVAR)
    response.raise_for_status()
    jsonResponse = response.json()
    print("SYSVAR DutyCycle: JSON Object")
    print(f'JSON: {jsonResponse}')
    ## JSON: {'ts': 1621257780000, 'v': 1, 's': 0}
    print(f'Value: {jsonResponse["v"]}')
    ## Value: 1
    print(f'Timestamp: {datetime.fromtimestamp(jsonResponse["ts"]/1000)})')
    ## Timestamp: 2021-05-17 15:23:00

    # Device
    response = requests.get(URL_GET_DEVICE)
    response.raise_for_status()
    jsonResponse = response.json()
    print("DEVICE Thermostat ACTUAL_TEMPERATURE: JSON Object")
    print(f'JSON: {jsonResponse}')
    ## JSON: {'ts': 1621322375521, 'v': 18.4, 's': 0}
    print(f'Value: {jsonResponse["v"]}')
    ## Value: 18.4
    print(f'Timestamp: {datetime.fromtimestamp(jsonResponse["ts"]/1000)})')
    ## Timestamp: 2021-05-18 09:19:35.521000

except HTTPError as http_err:
    print(f'HTTP error occurred: {http_err}')
except Exception as err:
    print(f'Other error occurred: {err}')
```

```
$ python3 ccujack-rest-get.py

ccujack-rest-get v20210518
SYSVAR DutyCycle: JSON Object
JSON: {'ts': 1621257780000, 'v': 1, 's': 0}
Value: 1
Timestamp: 2021-05-17 15:23:00
DEVICE Thermostat ACTUAL_TEMPERATURE: JSON Object
JSON: {'ts': 1621322375521, 'v': 18.4, 's': 0}
Value: 18.4
Timestamp: 2021-05-18 09:19:35.521000
```

## Set Device Value

Set the value of a thermostat device setpoint.

```
#!/usr/bin/python3

## ccujack-rest-set.py
## CCU-Jack REST-API Test setting the setpoint of a thermostat device HmIP-eTRV-B.
## Run: python3 ccujack-rest-set.py

## Device Variable, MakeLab Thermostat, SerienNr 000A18A9A64DAC
## HTTP Request: http://ccu-ip:2121/device/000A18A9A64DAC/1/SET_POINT_TEMPERATURE/~pv
## HTTP PUT Data: {"v":11.5}
## HTTP Response:

import requests
from requests.exceptions import HTTPError
from datetime import datetime
import time

VERSION = 'ccujack-rest-set v20210518'

## Set the URL for the REST-API request
URL_SET_DEVICE = 'http://ccu-ip:2121/device/000A18A9A64DAC/1/SET_POINT_TEMPERATURE/~pv'
URL_GET_DEVICE = 'http://ccu-ip:2121/device/000A18A9A64DAC/1/SET_POINT_TEMPERATURE/~pv'

## Define the data
data = {}
data['v']=12.5
device = "000A18A9A64DAC"
headers = {'Content-Type': "application/json", 'Accept': "application/json"}

print(f'{VERSION}')
try:
    print(f'Submitting HTTP PUT request...')
    response = requests.put(URL_SET_DEVICE, json=data, headers=headers)
    print(f'HTTP Response Code: {response.status_code}')
    print(f'HTTP Response Status: {response.raise_for_status()}')

    ## Short delay
    time.sleep(2)

    ## Check the result
    print(f'Submitting HTTP GET request...')
    response = requests.get(URL_GET_DEVICE)
    response.raise_for_status()
    jsonResponse = response.json()
    print(f'Device Setpoint: {jsonResponse}')

except HTTPError as http_err:
    print(f'HTTP error occurred: {http_err}')
except Exception as err:
    print(f'Other error occurred: {err}'')
```

```
$ python3 ccujack-rest-set.py
ccujack-rest-set v20210518
Submitting HTTP PUT request...
HTTP Response Code: 200
HTTP Response Status: None
Submitting HTTP GET request...
Device Setpoint: {'ts': 1621323938455, 'v': 12.5, 's': 0}
```

## Get CCU Devices

Get the list of CCU devices, name (title) and serial number (href).

```
#!/usr/bin/python3

## ccujack-rest-get-devices.py
## CCU-Jack REST-API Test getting the list of devices.
## Run: python3 ccujack-rest-get-devices.py

## Get Deviceslist
## HTTP Request: http://ccu-ip:2121/device
## HTTP Response:

## Set the CCUIP URL for the REST-API request
URL_GET_DEVICES = 'http://ccu-ip:2121/device'

import requests
from requests.exceptions import HTTPError
from datetime import datetime

try:
    response = requests.get(URL_GET_DEVICES)
    response.raise_for_status()
    jsonResponse = response.json()
    ## print(f'JSON: {jsonResponse}')
    ## Devices found = ~links key holds all devices
    print(f'Devices: {len(jsonResponse["~links"])}')
    devices = jsonResponse["~links"]
    ## Loop over the array devices
    print(f'Title (Name) = HRef (SerialNr)')
    for device in devices:
        ## {"rel": "device", "href": "000B1BE98D94DE", "title": "RC-Wohnzimmer"}
        print(f'{device["title"]} = {device["href"]}')
        ## MakeLab PSM = 0001D3C99C6AB3

except HTTPError as http_err:
    print(f'HTTP error occurred: {http_err}')
except Exception as err:
    print(f'Other error occurred: {err}'')
```

(Extract from the HTTP REST-API request, JSON response object)

```
$ python3 ccujack-rest-get-devices.py

JSON:
{
    'description': 'CCU3 Devices', 'identifier': 'device', 'title': 'Devices',
    '~links': [
        {'rel': 'device', 'href': 'CUX2801001', 'title': 'HM-RC-19 CUX2801001'},
        {'rel': 'device', 'href': '000B1BE98D94DE', 'title': 'RC-Wohnzimmer'},
        ...
        {'rel': 'device', 'href': '000A1A49A0D8A5', 'title': 'Flur'},
        {'rel': 'root', 'href': '...', 'title': 'Root'}]
}

Devices: 17
Title (Name) = HRef (SerialNr)
Wohnzimmer-1 = 00201A49952CB1
Doorbell Detector = 0026DBE998FA39
...
MakeLab = 000A18A9A64DAC
Root = ..
```

## MQTT-API

Some examples on how to subscribe and publish messages to the CCU-JACK MQTT broker but also to the Domoticz MQTT broker based on mosquitto.

The tests mainly use a Homematic IP thermostat device Hm-IP-eTRV-2 to get or set data via MQTT.

### CLI Examples

The examples are executed from the CLI on the Raspberry Pi Domoticz Development system running mosquitto (MQTT v3.1.1 broker).

After the CLI examples, some examples using Node-RED & Python.

## Quick Summary

**Homematic MQTT Subscribe Device Changes (device/status/serialnr/channel/type)**  
Thermostat HmIP-eTRV-2, Channel 1, Datapoint ACTUAL\_TEMPERATURE.

```
mosquitto_sub -h ccu-ip -t device/status/000A18A9A64DAC/1/ACTUAL_TEMPERATURE
Message received:
{"ts":1621065969946,"v":17.8,"s":0}
```

The message received is a JSON object with **name:value** pairs:  
**ts** = Timestamp UNIX format milliseconds, **v** = Value, **s** = Value Quality: 0=Good.

**Homematic MQTT Publish New Device State (device/set/serialnr/channel/type)**  
Thermostat HmIP-eTRV-2, Channel 1, Datapoint SET\_POINT\_TEMPERATURE.

```
mosquitto_pub -h ccu-ip -t device/set/000A18A9A64DAC/1/SET_POINT_TEMPERATURE -m {"v":19.5}
```

Define the new state as JSON Object with name:value pair:

**v** = Value

Ensure to escape character " to \" in the JSON Data definition or define the message payload as string '{"v":19.5}'.

**Domoticz MQTT Publish New Device State (domoticz/in)**  
Device Type Temperature, SubType LaCrosse TX3.

```
mosquitto_pub -h domoticz-ip -t domoticz/in -m '{"idx":31,"nvalue":0,"svalue":"21.2"}'
The payload (-m) must be defined as string.
```

**Domoticz MQTT Subscribe Device Changes (domoticz/out)**

```
mosquitto_sub -h domoticz-ip -t domoticz/out
Message received:
{
  "Battery": 255,
  "LastUpdate": "2021-05-16 10:46:57",
  "RSSI": 12,
  "description": "{\"serialnr\":\"000A18A9A64DAC\"}",
  "dtype": "Thermostat",
  "hwid": "4",
  "id": "001406E",
  "idx": 30,
  "name": "MakeLab Setpoint",
  "nvalue": 0,
  "stype": "SetPoint",
  "svalue1": "21.50",
  "unit": 1
}
And more ...
```

The message received is a JSON object.  
The name "svalue1" contains the actual setpoint value of the Domoticz thermostat device.  
The name description has been added just to show the link of the Domoticz device to the Homematic Thermostat device.

*Note*

### Domoticz MQTT Messages

From Domoticz version stable 2021.1, to view MQTT messages in the Domoticz log, the option "hardware debugging" must be enabled - [Read](#) > Domoticz MQTT communication.

## Subscribe Thermostat Actual Temperature

Thermostat Room MakeLab HmIP-eTRV-2, Channel 1, Datapoint ACTUAL\_TEMPERATURE.

```
mosquitto_sub -h ccu-ip -t device/status/000A18A9A64DAC/1/ACTUAL_TEMPERATURE
{"ts":1621065969946, "v":17.8, "s":0}
```

Topic (-t)	device/status/serialnr/channel/type
Message received	JSON object with <b>name</b> :value pairs <code>{"ts":long, "v":number, "s":number}</code> <b>ts</b> = Timestamp UNIX format milliseconds <b>v</b> = Value <b>s</b> = Value Quality: 0=Good.

## Publish Thermostat New Setpoint

Thermostat Room MakeLab HmIP-eTRV-2, Channel 1, Datapoint SET\_POINT\_TEMPERATURE.

```
mosquitto_pub -h ccu-ip -t device/set/000A18A9A64DAC/1/SET_POINT_TEMPERATURE -m {"v":19.5}
There is no response.
```

Topic (-t)	device/set/serialnr/channel/type
Message payload	JSON object with token <b>name</b> :value: <code>{"v":number}</code> <b>v</b> = Value  Ensure to escape character " to \' in the JSON Data definition or define the payload as string ' <code>{"v":number}</code> '.

Check if the setpoint is changed by subscribing to the datapoint type SET\_POINT\_TEMPERATURE.

```
mosquitto_sub -h ccu-ip -t device/status/000A18A9A64DAC/1/SET_POINT_TEMPERATURE
{"ts":1621067343372, "v":19.5, "s":0}
```

The MQTT message received confirms that the setpoint is 19.5 degrees - "v":19.5.

## Node-RED Examples

### **Example Node-RED handling Homematic MQTT Messages**

1. Homematic publishes messages
2. Node-RED subscribes to specific topics
3. Node-RED converts the message to Domoticz JSON object
4. Node-RED publishes the converted Homematic message to Domoticz
5. Domoticz to update the selected device(s)

### **Example Node-RED handling Domoticz MQTT Messages**

1. Node-RED subscribes to message with Domoticz topic “domoticz/out”
2. Node-RED selects a Domoticz device by property “idx”
3. Node-RED converts the Domoticz message to Homematic JSON object
4. Node-RED publishes the converted Domoticz message to Homematic
5. Homematic to update the selected device(s)

The next Node-RED flows are based on the previous terminal examples.

## Subscribe Thermostat Actual Temperature

Thermostat Room MakeLab HmIP-eTRV-2, Channel 1, Datapoint ACTUAL\_TEMPERATURE.

The received MQTT message from Homematic is logged.

### Node-RED Flow



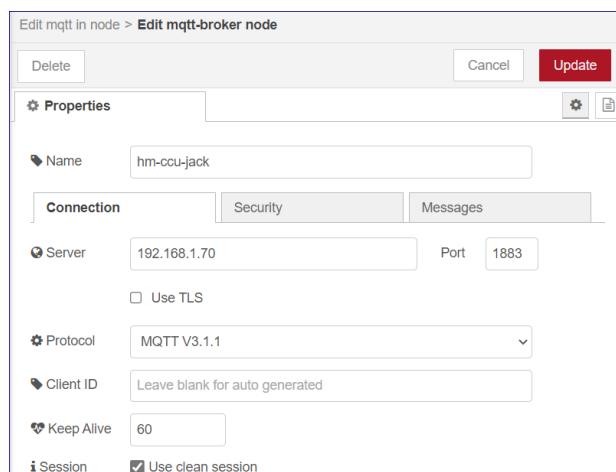
The flow has 4 nodes – 3 are nodes (visible) and 1 is a configuration node (not visible in the flow).

#### Node mqtt-in - Subscribe Thermostat Actual Temperature

Connects to the CCU-Jack MQTT broker (node “hm-ccu-jack”).

Subscribes to messages from topic “device/status/000A18A9A64DAC/1/ACTUAL\_TEMPERATURE” with setting output “Parsed JSON Object”.

This node uses a MQTT-broker node named “hm-ccu-jack”.



#### Node Function - Convert Timestamp LocaleString

This is an optional node showing how to convert the UNIX timestamp received to a readable date & time string.

Convert timestamp JSON name “ts” from UNIX long number (ms) to a locale string.

From:

```
{"ts":1621069660364,"v":18,"s":0}
```

To:

```
{"ts":"5-15-2021, 11:07:40","v":18,"s":0}
```

Ensure the msg payload is a JSON object.

```

const timeStamp = Number(msg.payload.ts);
// Log, i.e., 1621068893373
// node.warn(timeStamp);

// Create date object from UNIX timestamp in ms.
const dateVal = new Date(timeStamp);
const options = { hour12: false };
// const options = { weekday: 'long', year: 'numeric', month: 'long', day: 'numeric', hour12: false };

// Convert to local string
// Note: toLocalString does not work for de-DE. Workaround: replace / by -.

```

```
// {"ts": "5/14/2021, 18:32:58", "v": 23, "s": 0}
msg.payload.ts = dateVal.toLocaleString('de-DE', options).replace(/\\/g, "-");
// {"ts": "5-14-2021, 18:32:58", "v": 20.8, "s": 0}

return msg;
```

## Node - Debug

### Log the received message

15/05/2021, 19:04:38 node: 49fd8155.3c51c  
device/status/000A18A9A64DAC/1/ACTUAL\_TEMPERATURE:  
msg.payload : Object  
▶ { ts: "5-15-2021, 19:03:08", v: 19.3, s: 0 }

## Node-RED Flow Source

```
[{"id": "f3ea700d.49d81", "type": "mqtt_in", "z": "29c88325.94379c", "name": "Subscribe Thermostat Actual Temperature", "topic": "device/status/000A18A9A64DAC/1/ACTUAL_TEMPERATURE", "qos": "2", "datatype": "json", "broker": "6ffff726c.41220c", "nl": false, "rap": true, "rh": 0, "x": 180, "y": 80, "wires": [{"55daf40a.2b281c"}]}, {"id": "49fd8155.3c51c", "type": "debug", "z": "29c88325.94379c", "name": "", "active": true, "tosidebar": true, "console": false, "tostatus": false, "complete": "false", "statusVal": "", "statusType": "auto", "x": 810, "y": 80, "wires": []}, {"id": "55daf40a.2b281c", "type": "function", "z": "29c88325.94379c", "name": "Convert Timestamp LocaleString", "func": "/// Convert timestamp JSON name ts from UNIX (ms) to Locale String\n// Ensure the msg payload is a JSON object\nconst timeStamp = Number(msg.payload.ts);\n// Log, i.e. 1621068893373\nnode.warn(timeStamp);\n\n// Create date object from UNIX timestamp in ms.\nconst dateVal = new Date(timeStamp);\nconst options = { hour12: false };\n// const options = { weekday: 'long', year: 'numeric', month: 'long', day: 'numeric', hour12: false };\n\n// Convert to local string\n// Note: toLocalString does not work for de-DE. Workaround: replace / by -. \n// {\n// ts\n// :\n// 5/14/2021,\n// 18:32:58\n// ,\n// v\n// :\n// 23,\n// s\n// :\n// 0\n// }\nmsg.payload.ts = dateVal.toLocaleString('de-DE', options).replace(/\\/g, "-");\n\n// {\n// ts\n// :\n// \"5-14-2021, 18:32:58\",\n// v\n// :\n// 20.8,\n// s\n// :\n// 0\n// }\n\nreturn msg;\n", "outputs": 1, "noerr": 0, "initialize": "", "finalize": "", "libs": [], "x": 530, "y": 80, "wires": [{"49fd8155.3c51c"}]}, {"id": "6ffff726c.41220c", "type": "mqtt-broker", "name": "hm-ccu-jack-mqtt-broker", "broker": "ccu-ip", "port": "1883", "clientId": "", "useTls": false, "protocolVersion": "4", "keepalive": "60", "cleanSession": true, "birthTopic": "", "birthQos": "0", "birthPayload": "", "birthMsg": {}, "closeTopic": "", "closeQos": "0", "closePayload": "", "closeMsg": {}, "willTopic": "", "willQos": "0", "willPayload": "", "willMsg": {}, "sessionExpiry": ""}]
```

## Subscribe Thermostat Actual Temperature & Update Domoticz

Thermostat room MakeLab HmIP-eTRV-2, Channel 1, Datapoint ACTUAL\_TEMPERATURE.

Subscribe to Homematic MQTT message published by the CCU-Jack addon. Convert the received message payload to a Domoticz MQTT message with

- default topic: domoticz/in
- payload structure {"idx":NN,"nvalue":N,"svalue":"NN.N"}.

### Node-RED Flow



The flow has 5 nodes – 3 are nodes (visible) and 2 are configuration nodes (not visible in the flow).

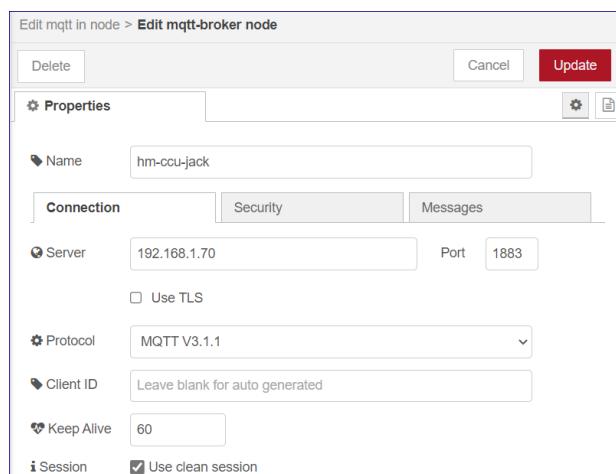
#### Node mqtt-in - Subscribe Thermostat Actual Temperature

Connects to the CCU-Jack MQTT broker (node “hm-ccu-jack”).

Subscribes to messages from topic “device/status/000A18A9A64DAC/1/ACTUAL\_TEMPERATURE” with setting output “Parsed JSON Object”.

#### Node mqtt-broker – hm-ccu-jack

This node uses a MQTT-broker node named “hm-ccu-jack”.



#### Node Function - Create Domoticz MQTT Message

The Homematic MQTT received message is converted to a Domoticz MQTT message.

From:

```
{"ts":1621072405650,"v":19.6,"s":0}
```

To:

```
{"idx":31,"nvalue":0,"svalue":"19.6"}
```

The Domoticz MQTT message is build according [this](#) documentation.

**IMPORTANT:** The name “svalue” must be a string.

The Homematic MQTT message name “v” is assigned to the Domoticz MQTT message name “svalue” as string.

The returned message has as topic the default Domoticz MQTT message in topic: "domoticz/in".

```
// Create MQTT message payload published to the Domoticz MQTT topic domoticz/in
// node.warn(msg.payload);

// Build a new message with topic domoticz/in
var msgdata = {};
msgdata.topic = "domoticz/in";

// Get the data from the incoming payload name "v"
temperature = msg.payload.v.toFixed(1).toString();

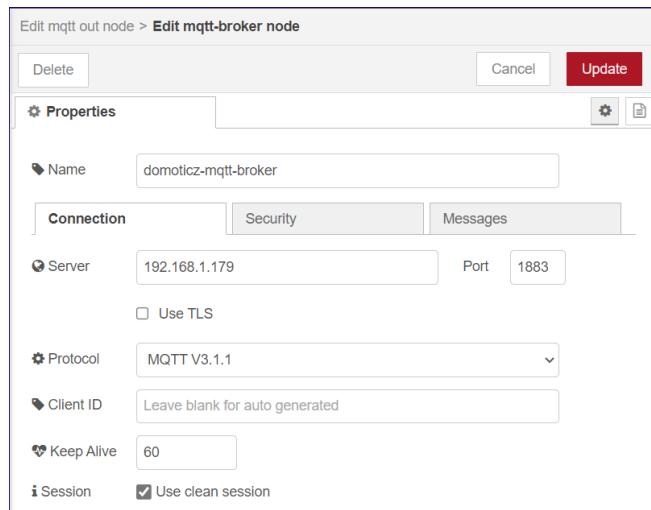
// Build the new payload (according Domoticz JSON/API documentation)
msgdata.payload = { "idx" : 31, "nvalue": 0, "svalue": temperature };
// node.warn(msgdata.payload);

// Return the message
return msgdata;
```

### Node mqtt-out – Publish Domoticz

Publishes the MQTT message to the Domoticz MQTT broker.

### Node mqtt-broker - domoticz-mqtt-broker



### Debug

Log the published message:

- topic: domoticz/in
- msg.payload: {"idx":31,"nvalue":0,"svalue":"19.6"}

```
15/05/2021, 19:07:00 node: Create Domoticz MQTT Message
function : (warn)
▶ { ts: 1621098188816, v: 19.3, s: 0 }

15/05/2021, 19:07:00 node: 6db7738c.0036bc
domoticz/in : msg.payload : Object
▶ { idx: 31, nvalue: 0, svalue: "19.3" }
```

### Domoticz Device Widget



## Hint: Manually Update the Domoticz Device

Example to manually update the Domoticz device.

```
mosquitto_pub -h domoticz-ip -t 'domoticz/in' -m '{"idx":31,"nvalue":0,"svalue":"21.2"}'
```

The topic (-t) and the payload (-m) must be defined as strings.

## Node-RED Flow Source

```
[{"id": "ef4ec627.828c78", "type": "function", "z": "29c88325.94379c", "name": "Create Domoticz MQTT Message", "func": "// Create MQTT message payload published to the Domoticz MQTT topic domoticz/in\nnode.warn(msg.payload);\n\n// Build a new message with topic domoticz/in\\nvar msgdata = {};\n\nmsgdata.topic = \"domoticz/in\";\n\n// Get the data from the incoming payload name\n\nvar temperature = msg.payload.v.toFixed(1).toString();\n\n// Build the new payload (according Domoticz JSON/API documentation)\n\nmsgdata.payload = { \"idx\": 31, \"nvalue\": 0, \"svalue\": temperature };\n\n// node.warn(msgdata.payload);\n\n// Return the message\\nreturn msgdata;\n\n", "outputs": 1, "noerr": 0, "initialize": "", "finalize": "", "libs": [], "x": 540, "y": 180, "wires": [[{"id": "6db7738c.0036bc", "x": 600, "y": 180}], {"id": "6db7738c.0036bc", "type": "debug", "z": "29c88325.94379c", "name": "", "active": true, "tostidebar": true, "console": false, "tostatus": false, "complete": "payload", "targetType": "msg", "statusVal": "", "statusType": "auto", "x": 810, "y": 240, "wires": []}], {"id": "996ef83e.c62d58", "type": "mqtt out", "z": "29c88325.94379c", "name": "Publish Domoticz", "topic": "", "qos": "", "retain": "", "respTopic": "", "contentType": "", "userProps": "", "correl": "", "expiry": "", "broker": "a3031651.0e39d8", "x": 830, "y": 180, "wires": [{"id": "c4427d68.bf9bd", "type": "mqtt in", "z": "29c88325.94379c", "name": "Subscribe Thermostat Actual Temperature", "topic": "device/status/000A18A9A64DAC/1/ACTUAL_TEMPERATURE", "qos": "2", "datatype": "json", "broker": "6ffff726c.41220c", "nl": false, "rap": true, "rh": 0, "x": 180, "y": 240, "wires": [{"id": "a3031651.0e39d8", "type": "mqtt-broker", "name": "domoticz-mqtt-broker", "broker": "domoticz-ip", "port": 1883, "clientid": "", "usetls": false, "protocolVersion": 4, "keepalive": 60, "cleansession": true, "birthTopic": "", "birthQos": 0, "birthPayload": "", "birthMsg": {}, "closeTopic": "", "closeQos": 0, "closePayload": "", "closeMsg": {}, "willTopic": "", "willQos": 0, "willPayload": "", "willMsg": {}, "sessionExpiry": ""}], {"id": "6ffff726c.41220c", "type": "mqtt-broker", "name": "hm-ccu-jack-mqtt-broker", "broker": "ccu-ip", "port": 1883, "clientid": "", "usetls": false, "protocolVersion": 4, "keepalive": 60, "cleansession": true, "birthTopic": "", "birthQos": 0, "birthPayload": "", "birthMsg": {}, "closeTopic": "", "closeQos": 0, "closePayload": "", "closeMsg": {}, "willTopic": "", "willQos": 0, "willPayload": "", "willMsg": {}, "sessionExpiry": ""}]}
```

## Publish Thermostat New Setpoint (Inject)

Thermostat room MakeLab HmIP-eTRV-2, Channel 1, Datapoint SET\_POINT\_TEMPERATURE.



The Inject node sets the new setpoint – 14.4 defined as number.

The function node creates the MQTT message to publish to Homematic MQTT broker (CCU-Jack):

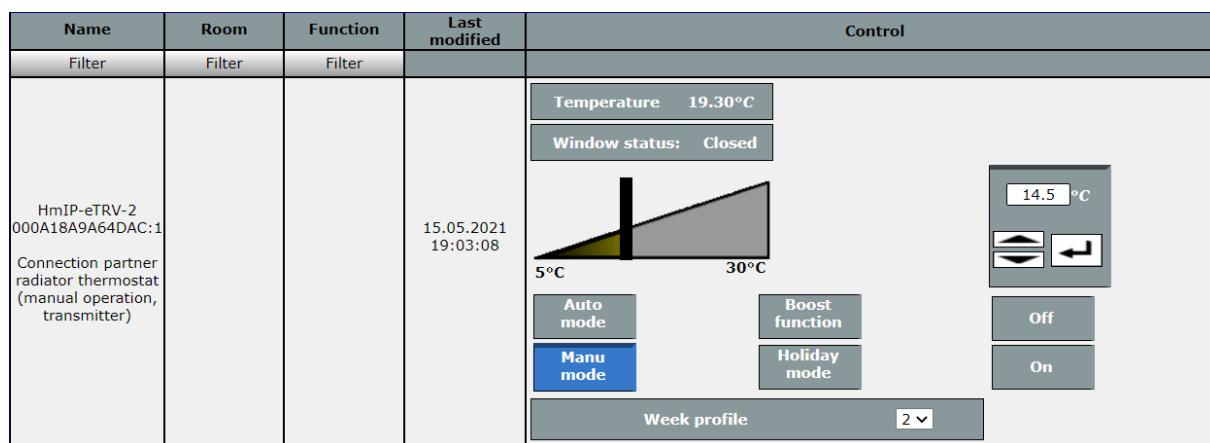
- Topic: device/set/000A18A9A64DAC/1/SET\_POINT\_TEMPERATURE
- Payload: {"v":14.4}

The mqtt-out node connects to the Homematic MQTT broker and publishes the message.

*Note*

Homematic converts the payload from 14.4 to 14.5 °C.

Homematic WebUI showing the new setpoint 14.5 °C.



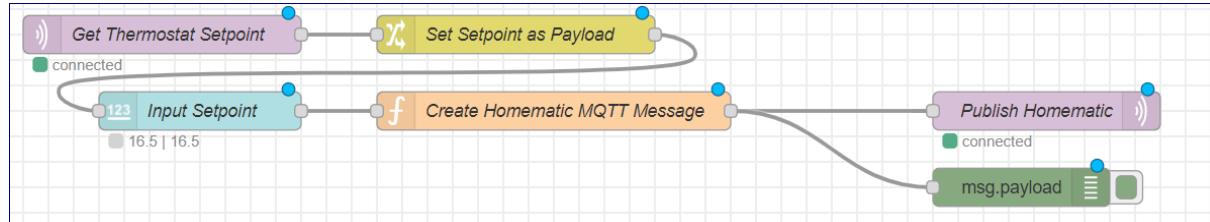
## Node-RED Flow Source

```
[{"id": "1f5dfe5f.fd0402", "type": "inject", "z": "29c88325.94379c", "name": "Set Thermostat Setpoint", "props": [{"p": "payload"}, {"p": "topic", "vt": "str"}], "repeat": "", "crontab": "", "once": false, "on": "eDelay": 0.1, "topic": "", "payload": "14.4", "payloadType": "num", "x": 140, "y": 380, "wires": [[{"b9743ff2.2ab6"}]}, {"id": "776ece92.1b54f", "type": "mqtt out", "z": "29c88325.94379c", "name": "Publish Homematic", "topic": "", "qos": "", "retain": "", "respTopic": "", "contentType": "", "userProps": "", "correl": "", "expiry": "", "broker": "6fff726c.41220c", "x": 830, "y": 380, "wires": []}, {"id": "b9743ff2.2ab6", "type": "function", "z": "29c88325.94379c", "name": "Create Homematic MQTT Message", "func": "// Define the topic to set the new setpoint\nmsg.topic = \"/device/set/000A18A9A64DAC/1/SET_POINT_TEMPERATURE\\n\\nvar newsetpoint =\nmsg.payload;\\n\\nmsg.payload = {\"v\":newsetpoint};\\nnode.warn(msg.payload);\\n\\nreturn\nmsg; ", "outputs": 1, "noerr": 0, "initialize": "", "finalize": "", "libs": [], "x": 540, "y": 380, "wires": [{"776ece92.1b54f", "f0259632.47e108"}]}, {"id": "f0259632.47e108", "type": "debug", "z": "29c88325.94379c", "name": "", "active": true, "tosidebar": true, "console": false, "tostatus": false, "complete": "payload", "targetType": "msg", "statusVal": "", "statusType": "auto", "x": 810, "y": 440, "wires": []}, {"id": "6fff726c.41220c", "type": "mqtt-broker", "name": "hm-ccu-jack-mqtt-broker", "broker": "ccu-ip", "port": "1883", "clientid": "", "usetls": false, "protocolVersion": "4", "keepalive": "60", "cleansession": true, "birthTopic": "", "birthQos": "0", "birthPayload": "", "birthMsg": {}, "closeTopic": "", "closeQos": "0", "closePayload": "", "closeMsg": {}, "willTopic": "", "willQos": "0", "willPayload": "", "willMsg": {}, "sessionExpiry": ""}]]
```

## Publish Thermostat New Setpoint (UI)

This example builds upon the previous flow, but the Inject node is replaced by a [Node-RED Dashboard](#) UI node to enter the setpoint manually.

### Node-RED Flow



The node mqtt-in “Get Thermostat Setpoints” listens to thermostat SET\_POINT\_TEMPERATURE changes.

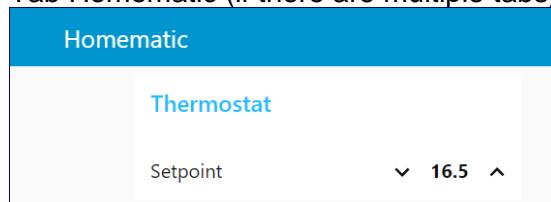
The change node “Set Setpoint as Payload” converts the “msg.payload.v” to “msg.payload” used to set the value of the node ui\_numeric “Input Setpoint”.

The node ui\_numeric “Input Setpoint” enables to change the setpoint value in 0.5 degrees steps. Each change is sent to the mqtt-out node “Publish Homematic” to publish to the Homematic MQTT broker.

<div style="border: 1px solid #ccc; padding: 5px;"> <b>Edit numeric node</b>  <input type="button" value="Delete"/>   <b>Properties</b>  <input checked="" type="checkbox"/> Group [Homematic] Thermostat  <input checked="" type="checkbox"/> Size auto  <input checked="" type="checkbox"/> Label Setpoint  <input checked="" type="checkbox"/> Tooltip optional tooltip  <input checked="" type="checkbox"/> Value Format {{value}}  <input checked="" type="checkbox"/> Range min 0 max 25 step 0.5  <input checked="" type="checkbox"/> Wrap value from max to min and min to max. <input type="checkbox"/>  <input checked="" type="checkbox"/> If msg arrives on input ... <input type="checkbox"/>  <input checked="" type="checkbox"/> When changed, send:            Payload Current value            Topic <input type="button" value="msg. topic"/>            Name Input Setpoint         </div>	<p><b>Notes</b></p> <p>The step is in 0.5 degrees.</p> <p>The option “if msg arrives on input ...” is disabled, because the MQTT received value should not be forwarded else an endless loop would occur.</p> <p>Each value change is sent to the next node mqtt-out.</p> <p>Instead ui_numeric node, could also use a slider ui_slider or ui_dropdown.</p>
---	---

### Node-RED UI in Browser (<http://domoticz-ip:1880/ui>)

Tab Homematic (if there are multiple tabs)



#### Note

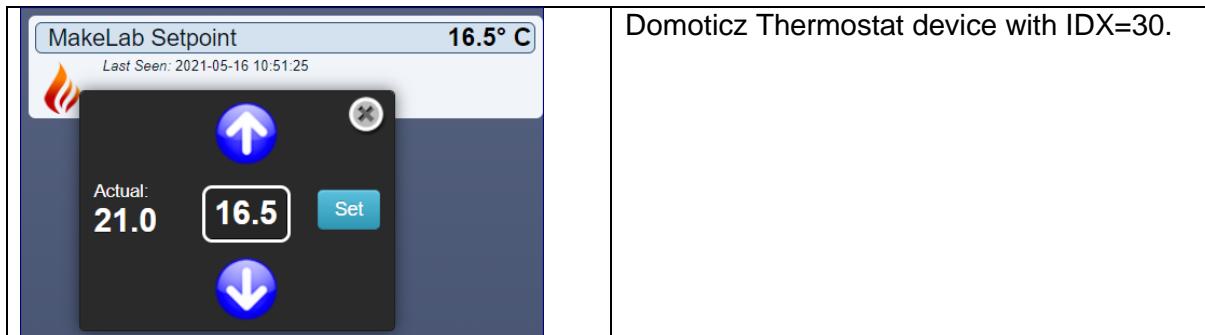
Set the value to 0 will switch the thermostat OFF - if in manual mode.  
If the setpoint is 0, Homematic sets the setpoint to 4.5.

## Node-RED Flow Source

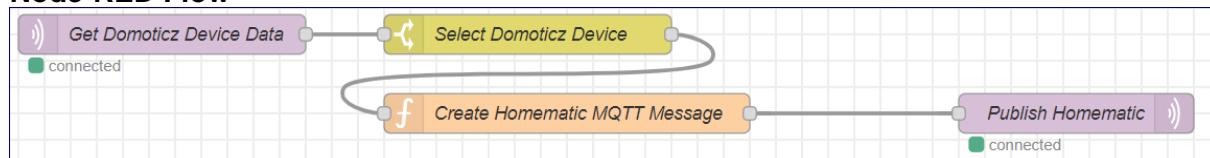
```
[{"id": "7874b8d4.dc08c8", "type": "mqtt_in", "z": "29c88325.94379c", "name": "Get Thermostat Setpoint", "topic": "device/status/000A18A9A64DAC/1/SET_POINT_TEMPERATURE", "qos": "2", "datatype": "json", "broker": "6ffff726c.41220c", "nl": false, "rap": true, "rh": 0, "x": 130, "y": 380, "wires": [{"f0990242.b3e82"}]}, {"id": "776ece92.1b54f", "type": "mqtt_out", "z": "29c88325.94379c", "name": "Publish Homematic", "topic": "", "qos": "", "retain": "", "respTopic": "", "contentType": "", "userProps": "", "correl": "", "expiry": "", "broker": "6ffff726c.41220c", "x": 830, "y": 440, "wires": []}, {"id": "b9743ff2.2ab6", "type": "function", "z": "29c88325.94379c", "name": "Create Homematic MQTT Message", "func": "// Define the topic to set the new setpoint\nmsg.topic = \"device/set/000A18A9A64DAC/1/SET_POINT_TEMPERATURE\\n\\nvar newsetpoint = msg.payload\\nmsg.payload = {\"v\":newsetpoint};\\nnode.warn(msg.payload);\\n\\nreturn msg", "outputs": 1, "noerr": 0, "initialize": "", "finalize": "", "libs": [], "x": 440, "y": 440, "wires": [{"f0990242.b3e82"}]}, {"id": "f0259632.47e108", "type": "debug", "z": "29c88325.94379c", "name": "", "active": true, "toSidebar": true, "console": false, "tostatus": false, "complete": "payload", "targetType": "msg", "statusVal": "", "statusType": "auto", "x": 810, "y": 500, "wires": []}, {"id": "2005de50.30f912", "type": "ui_numeric", "z": "29c88325.94379c", "name": "Input Setpoint", "label": "Setpoint", "tooltip": "", "group": "ba6487aa.136fd8", "order": 0, "width": 0, "height": 0, "wrap": false, "passthru": false, "topic": "topic", "topicType": "msg", "format": "{{value}}", "min": 0, "max": 25, "step": 0.5, "x": 160, "y": 440, "wires": [{"f0990242.b3e82"}]}, {"id": "f0990242.b3e82", "type": "change", "z": "29c88325.94379c", "name": "Set Setpoint as Payload", "rules": [{"t": "set", "p": "payload", "pt": "msg", "to": "payload.v", "tot": "msg"}]}, {"id": "6ffff726c.41220c", "type": "mqtt-broker", "name": "hm-ccu-jack-mqtt-broker", "broker": "ccu-ip", "port": 1883, "clientId": "", "useTls": false, "protocolVersion": 4, "keepalive": 60, "cleanSession": true, "birthTopic": "", "birthQos": 0, "birthPayload": "", "birthMsg": {}, "closeTopic": "", "closeQos": 0, "closePayload": "", "closeMsg": {}}, {"id": "ba6487aa.136fd8", "type": "ui_group", "name": "Thermostat", "tab": "f0cc92fc.5bd4e", "order": 1, "display": true, "width": 6, "collapse": false}, {"id": "f0cc92fc.5bd4e", "type": "ui_tab", "name": "Homematic", "icon": "dashboard", "disabled": false, "hidden": false}]
```

## Publish Thermostat New Setpoint (Domoticz)

Set the thermostat new setpoint triggered by a Domoticz Thermostat device.



### Node-RED Flow



The mqtt-in node “Get Domoticz Device Data” subscribes to the Domoticz topic “domoticz/out”.

The switch node “Select Domoticz Device” routes the message based on the value of the property msg.payload.idx (“== 30”).

The function node “Create Homematic MQTT Message” builds the message

- topic ( "device/set/000A18A9A64DAC/1/SET\_POINT\_TEMPERATURE")
- payload ({"v":16.5})

to be published to the Homematic MQTT broker.

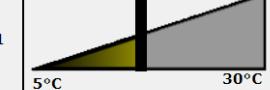
The mqtt-out node “Publish Homematic” sends the message to the Homematic MQTT broker.

### Node-RED Flow Source

```
[{"id": "39db349b.d61d5c", "type": "mqtt in", "z": "29c88325.94379c", "name": "Get Domoticz Device Data", "topic": "domoticz/out", "qos": "2", "datatype": "json", "broker": "a3031651.0e39d8", "nl": false, "rap": true, "rh": 0, "x": 130, "y": 580, "wires": [{"id": "50aaeb9f.3a7f64"}]}, {"id": "680cf203.553afc", "type": "mqtt out", "z": "29c88325.94379c", "name": "Publish Homematic", "topic": "", "qos": "", "retain": "", "respTopic": "", "contentType": "", "userProps": "", "correl": "", "expiry": "", "broker": "6fff726c.41220c", "x": 830, "y": 640, "wires": []}, {"id": "961d758f.ff1a48", "type": "function", "z": "29c88325.94379c", "name": "Create Homematic MQTT Message", "func": "// Define the topic to set the new setpoint\nmsg.topic = \"device/set/000A18A9A64DAC/1/SET_POINT_TEMPERATURE\";\n\nvar newsetpoint = Number(msg.payload.svalue1);\n\nmsg.payload = {\"\\\":newsetpoint};\nnode.warn(msg.payload);\n\nreturn msg;", "outputs": 1, "noerr": 0, "initialize": "", "finalize": "", "libs": [], "x": 440, "y": 640, "wires": [{"id": "680cf203.553afc"}]}, {"id": "50aaeb9f.3a7f64", "type": "switch", "z": "29c88325.94379c", "name": "Select Domoticz Device", "property": "payload.idx", "propertyType": "msg", "rules": [{"t": "eq", "v": "30", "vt": "num"}], "checkall": true, "repair": false, "outputs": 1, "x": 410, "y": 580, "wires": [{"id": "961d758f.ff1a48"}]}, {"id": "a3031651.0e39d8", "type": "mqtt-broker", "name": "domoticz-mqtt-broker", "broker": "domoticz-ip", "port": 1883, "clientid": "", "usetls": false, "protocolVersion": "4", "keepalive": 60, "cleansession": true, "birthTopic": "", "birthQos": 0, "birthPayload": "", "birthMsg": {}, "closeTopic": "", "closeQos": 0, "closePayload": "", "closeMsg": {}, "willTopic": "", "willQos": 0, "willPayload": "", "willMsg": {}, "sessionExpiry": ""}, {"id": "6fff726c.41220c", "type": "mqtt-broker", "name": "hm-ccu-jack-mqtt-broker", "broker": "ccu-ip", "port": 1883, "clientid": "", "usetls": false, "protocolVersion": "4", "keepalive": 60, "cleansession": true, "birthTopic": "", "birthQos": 0, "birthPayload": "", "birthMsg": {}, "closeTopic": "", "closeQos": 0, "closePayload": "", "closeMsg": {}, "willTopic": "", "willQos": 0, "willPayload": "", "willMsg": {}, "sessionExpiry": ""}]]
```

## Homematic Thermostat Device

The setpoint is set to 16.5 °C as triggered by Domoticz Thermostat Device (IDX=30).

Name	Room	Function	Last modified	Control
Filter	Filter	Filter		
HmIP-eTRV-2 000A18A9A64DAC:1 Connection partner radiator thermostat (manual operation, transmitter)			16.05.2021 10:51:26	<div style="display: flex; justify-content: space-between;"> <div style="flex: 1;"> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;">Temperature 18.60°C</div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;">Window status: Closed</div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;">  <p>5°C                    30°C</p> </div> <div style="display: flex; justify-content: space-around; margin-bottom: 5px;"> <span>Auto mode</span> <span>Boost function</span> </div> <div style="background-color: #0070C0; color: white; padding: 5px; border-radius: 5px; text-align: center; width: fit-content;">Manu mode</div> <div style="display: flex; justify-content: space-around; margin-bottom: 5px;"> <span>Holiday mode</span> </div> <div style="background-color: #0070C0; color: white; padding: 5px; border-radius: 5px; text-align: center; width: fit-content;">Week profile</div> <div style="border: 1px solid #ccc; padding: 2px 5px; margin-left: 10px;">2 ▾</div> </div> <div style="display: flex; justify-content: space-around; align-items: center; margin-top: 10px;"> <span>16.5 °C</span> <span style="border: 1px solid #ccc; padding: 2px 5px; border-radius: 5px; text-align: center;">▲</span> <span style="border: 1px solid #ccc; padding: 2px 5px; border-radius: 5px; text-align: center;">▼</span> </div> <div style="display: flex; justify-content: space-around; align-items: center; margin-top: 10px;"> <span>Off</span> <span>On</span> </div> </div>

## Get & Set System Variables

Playing around with Homematic system variables.

Homematic system variables could be used by Domoticz set set data which is then picked up by Homematic programs or Homematic to set a system variable, from a script which is then used by Domoticz to trigger action.

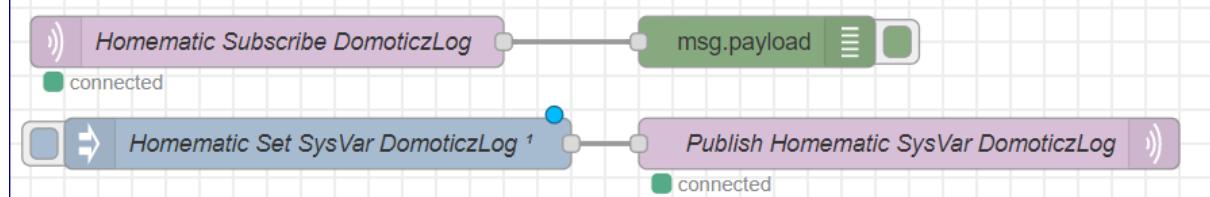
To get the list of system variables, use the CCU-Jack REST-API.

```
http://ccu-ip:2121/sysvar
```

```
{
  "description": "System variables of the ReGaHss",
  "identifier": "sysvar",
  "title": "System variables",
  "~links": [
    {"rel": "sysvar", "href": "1236", "title": "${sysVarAlarmZone1}" },
    {"rel": "sysvar", "href": "950", "title": "${sysVarPresence}" },
    {"rel": "sysvar", "href": "2799", "title": "BatteryCheck" },
    {"rel": "sysvar", "href": "4431", "title": "Domoticz.Data" },
    {"rel": "sysvar", "href": "4765", "title": "DomoticzLog" },
    {"rel": "sysvar", "href": "3032", "title": "DutyCycle" },
    {"rel": "sysvar", "href": "1517", "title": "svEnergyCounterOldVal_1500" },
    {"rel": "sysvar", "href": "1518", "title": "svEnergyCounter_1500_0001D3C99C6AB3:6" },
    {"rel": "root", "href": "...", "title": "Root" }
  ]
}
```

The name “href” is required to subscribe and publish Homematic system variables.

### Node-RED Flow



The first flow subscribes to messages from the system variable with ISE\_ID 4765.  
Topic: sysvar/status/4765.

The second flow sets the new value of the system variable with ISE\_ID 4765 and published the new value to the Homematic MQTT broker.

Topic: sysvar/set/4765

Payload: “New entry sysvar DomoticzLog.”

The ISE\_ID 4765 is taken from the HTTP REST-API request response (see previous):

```
{"rel": "sysvar", "href": "4765", "title": "DomoticzLog"},
```

### Node-RED Log

```

16/05/2021, 11:28:56 node: e5a3aa74.de4d78
sysvar/status/4765 : msg.payload : Object
▶ { ts: 1621156999000, v: "This is a
new entry in the sys...", s: 0 }

16/05/2021, 11:28:56 node: e5a3aa74.de4d78
sysvar/status/4765 : msg.payload : Object
▶ { ts: 1621157337000, v: "New entry
sysvar DomoticzLog.", s: 0 }

```

The content of the system variable is changed from “This is a...” to “New entry...”.

## Node-RED Flow Source

```
[{"id": "d9115c8f.f0c4c", "type": "mqtt_in", "z": "29c88325.94379c", "name": "Homematic Subscribe DomoticzLog", "topic": "sysvar/status/4765", "qos": "2", "datatype": "json", "broker": "6fff726c.41220c", "nl": false, "rap": true, "rh": 0, "x": 160, "y": 940, "wires": [{"id": "e5a3aa74.de4d78"}]}, {"id": "e5a3aa74.de4d78", "type": "debug", "z": "29c88325.94379c", "name": "", "active": true, "tosidebar": true, "console": false, "tostatus": false, "complete": "payload", "targetType": "msg", "statusVal": "", "statusType": "auto", "x": 450, "y": 940, "wires": []}, {"id": "389d26bd.0a934a", "type": "inject", "z": "29c88325.94379c", "name": "Homematic Set SysVar DomoticzLog", "props": [{"p": "payload"}, {"p": "topic", "vt": "str"}], "repeat": "", "crontab": "", "once": true, "onceDelay": 0.1, "topic": "sysvar/set/4765", "payload": "New entry sysvar DomoticzLog", "payloadType": "str", "x": 190, "y": 1000, "wires": [{"id": "570a695b.05e748"}]}, {"id": "570a695b.05e748", "type": "mqtt_out", "z": "29c88325.94379c", "name": "Publish Homematic SysVar DomoticzLog", "topic": "", "qos": "", "retain": "", "respTopic": "", "contentType": "", "userProps": "", "correl": "", "expiry": "", "broker": "6fff726c.41220c", "x": 540, "y": 1000, "wires": []}, {"id": "6fff726c.41220c", "type": "mqtt-broker", "name": "hm-ccu-jack-mqtt-broker", "broker": "ccu-ip", "port": 1883, "clientId": "", "useTls": false, "protocolVersion": 4, "keepalive": 60, "cleanSession": true, "birthTopic": "", "birthQos": 0, "birthPayload": "", "birthMsg": {}, "closeTopic": "", "closeQos": 0, "closePayload": "", "closeMsg": {}, "willTopic": "", "willQos": 0, "willPayload": "", "willMsg": {}, "sessionExpiry": ""}]
```

## Python3 Examples

Python3 script examples showing how to use GET & SET REST-API requests.  
The scripts are developed on a Raspberry Pi 4B running Raspberry OS with Python 3.7.3 [GCC 8.3.0].

The Python HTTP library [requests](#) (2.21.0) is used.  
Installed with sudo pip3 install requests

### Subscribe Device

Subscribe to a thermostat device setpoint.

```
#!/usr/bin/python3

## ccujack-mqtt-subscribe-device.py
## CCU-Jack MQTT-API Test subscribing thermostat device.
## Run: python3 ccujack-mqtt-subscribe-device.py

VERSION = "ccujack-mqtt-subscribe-device v20210518"

## Set MQTT parameter
MQTT_BROKER = "ccu-ip"
MQTT_TOPIC = "device/status/000A18A9A64DAC/1/SET_POINT_TEMPERATURE"
CLIENT_ID = "ccujack-test"

from paho.mqtt import client as mqtt_client

# Connect to the MQTT Broker
def connect_mqtt() -> mqtt_client:
    def on_connect(client, userdata, flags, rc):
        if rc == 0:
            print(f"Connected to MQTT Broker {MQTT_BROKER}")
        else:
            print(f"Failed to connect, return code {rc}\n")

    client = mqtt_client.Client(CLIENT_ID)
    client.on_connect = on_connect
    client.connect(MQTT_BROKER)
    return client

def subscribe(client: mqtt_client):
    # Handle new messages
    def on_message(client, userdata, msg):
        # Decode the message payload to a string
        payload_decoded = msg.payload.decode("utf-8", "ignore")
        # Convert the decoded json string to a dict object
        payload = eval(payload_decoded)
        print(f"Payload: {payload}")
        print(f"Value: {payload['v']}")

    client.subscribe(MQTT_TOPIC)
    client.on_message = on_message

def run():
    print(f"{VERSION}")
    client = connect_mqtt()
    subscribe(client)
    client.loop_forever()

if __name__ == '__main__':
    run()
```

## Output

```
$ python3 ccujack-mqtt-subscribe-device.py  
ccujack-mqtt-subscribe-device v20210518  
Connected to MQTT Broker ccu-ip  
Payload: {'ts': 1621328324292, 'v': 14.5, 's': 0}  
Value: 14.5  
Payload: {'ts': 1621328858842, 'v': 8, 's': 0}  
Value: 8
```

## Publish Device

Publish a new thermostat device setpoint. The script runs once.

```
#!/usr/bin/python3

## ccujack-mqtt-publish-device.py
## CCU-Jack MQTT-API Test subscribing thermostat device.
## Run: python3 ccujack-mqtt- publish-device.py

from paho.mqtt import client as mqtt_client
import json

VERSION = "ccujack-mqtt-subscribe-device v20210518"

## Set MQTT parameter
MQTT_BROKER = "ccu-ip"
MQTT_TOPIC = "device/set/000A18A9A64DAC/1/SET_POINT_TEMPERATURE"
CLIENT_ID = "ccujack-test"
## Define the data: v = new setpoint
data = {}
data['v']=16.5

# Connect to the MQTT Broker
def connect_mqtt() -> mqtt_client:
    # Connect as client to the MQTT broker
    def on_connect(client, userdata, flags, rc):
        if rc == 0:
            print(f"Connected to MQTT Broker {MQTT_BROKER}")
        else:
            print(f"Failed to connect, return code {rc}\n")

    client = mqtt_client.Client(CLIENT_ID)
    client.on_connect = on_connect
    client.connect(MQTT_BROKER)
    return client

# Publish the new setpoint to the MQTT broker
def publish(client):
    # Publish the JSON object - use json.dumps to convert
    result = client.publish(MQTT_TOPIC, json.dumps(data))
    print(f'Publish Result: {result}')
    # Publish Result: (0, 1) which is (result, mid)
    # result = error code; 0 = OK
    # mid = published message number assigned by the client
    if result[0] == 0:
        print(f"OK - Send data '{str(data)}' to topic '{MQTT_TOPIC}'")
    else:
        print(f"ERROR - Failed send data to topic {MQTT_TOPIC}")

def run():
    print(f"{VERSION}")
    client = connect_mqtt()
    client.loop_start()
    publish(client)

if __name__ == '__main__':
    run()
```

## Output

```
python3 ccujack-mqtt-publish-device.py
ccujack-mqtt-subscribe-device v20210518
Publish Result: (0, 1)
OK - Send data '{'v': 16.5}' to topic 'device/set/000A18A9A64DAC/1/SET_POINT_TEMPERATURE'
```

## Subscribe & Publish Thermostat New Setpoint (Domoticz)

Subscribe to a Homematic IP Thermostat HmIP-eTRV-2 datapoint SET\_POINT\_TEMPERATURE and publish the setpoint to a Domoticz Thermostat device. There are two MQTT brokers used: CCU-Jack mosquitto and Domoticz mosquitto.

```
#!/usr/bin/python3

## ccujack-mqtt-device-to-domoticz.py
## CCU-Jack MQTT-API Test subscribing thermostat device SET_POINT_TEMPERATURE
## and publish the value to Domoticz Thermostat device.
## Quit the script after publishing to Domoticz.
## Run: python3 ccujack-mqtt-device-to-domoticz.py

from paho.mqtt import client as mqtt_client

VERSION = "ccujack-mqtt-device-to-domoticz v20210518"

## Set MQTT parameter
## There are two MQTT brokers: CCU3 and Domoticz
## Subscribe setpoint Homematic Thermostat
CCU_MQTT_BROKER = "ccu-ip"
CCU_MQTT_TOPIC = "device/status/000A18A9A64DAC/1/SET_POINT_TEMPERATURE"
CCU_CLIENT_ID = "ccujack-test"

## Publish setpoint Domoticz Virtual Sensor Type: Thermostat, SubType: SetPoint
DOM_MQTT_BROKER = 'domoticz-ip'
DOM_MQTT_TOPIC = "domoticz/in"
DOM_DEVICE_IDX = 30

# Connect to the CCU3 MQTT Broker
def connect_mqtt() -> mqtt_client:
    def on_connect(client, userdata, flags, rc):
        if rc == 0:
            print(f"Connected to CCU3 MQTT Broker {CCU_MQTT_BROKER}")
        else:
            print(f"Failed to connect to the CCU3 MQTT Broker, return code {rc}\n")

    client = mqtt_client.Client(CCU_CLIENT_ID)
    client.on_connect = on_connect
    client.connect(CCU_MQTT_BROKER)
    return client

def subscribe(client: mqtt_client):

    # Handle Domoticz message published
    def on_dom_publish(client, userdata, result):                  #create function for callback
        print("Domoticz message published")
        pass

    # Handle new messages from the CCU
    def on_message(client, userdata, msg):
        # Decode the received CCU3 message payload to a string
        ccupayload_decoded = msg.payload.decode("utf-8", "ignore")
        # Convert the decoded json string to a dict object to get a value from a key
        ccupayload = eval(ccupayload_decoded)
        print(f"CCU3 Message Payload: {ccupayload}")
        # Get the setpoint value from the CCU3 message: key "v"
        setpoint = ccupayload['v']
        # Connect to Domoticz MQTT Broker
        print(f"Publish new setpoint {setpoint} to Domoticz Thermostat device IDX={DOM_DEVICE_IDX}")
        # Create Domoticz client object
        domclient = mqtt_client.Client("Domoticz Client")
        # Assign function to Domoticz on_publish callback
        domclient.on_publish = on_dom_publish
        # Connect to the Domoticz MQTT broker
        domclient.connect(DOM_MQTT_BROKER)
        print(f"Connected to Domoticz MQTT Broker {DOM_MQTT_BROKER}")
        # Create the MQTT message as string. svalue must be a string.
        dompayload = '{"idx" : %d, "nvalue" : 0, "svalue" : "%s"}' % (DOM_DEVICE_IDX, str(setpoint))
        print(f"Domoticz Message payload: {dompayload}")
        result = domclient.publish(DOM_MQTT_TOPIC, dompayload)
        print(f"Domoticz Result: {result}")


```

```
quit()

client.subscribe(CCU_MQTT_TOPIC)
client.on_message = on_message

def run():
    print(f"{{VERSION}}")
    client = connect_mqtt()
    subscribe(client)
    client.loop_forever()

if __name__ == '__main__':
    run()
```

## Output

```
$python3 ccujack-mqtt-device-to-domoticz.py

ccujack-mqtt-subscribe-device v20210518
Connected to MQTT Broker ccu-ip
Payload: {'ts': 1621358928287, 'v': 7, 's': 0}
Setpoint: 7
Domoticz publish new setpoint to Thermostat device IDX=30
Domoticz Client connected
Domoticz Publish Data: {"idx" : 30, "nvalue" : 0, "svalue" : "7"}
Domoticz data published
Domoticz Result: (0, 1)
```

# Addon CUxD

## Information

The [CUxD](#) CCU Addon is used for communication triggered by the CCU to Domoticz.  
The CCU submits Domoticz HTTP API Requests via system command wget.

This concept can be used for example to

- handle Window/Door Contact state changes and update Domoticz Alert Device
- handle manual Radiator Thermostat changes and sync the Domoticz Setpoint Device

Get started, by reading [here](#) and [download](#) latest version.

Install via the Homematic WebUI > Home page > Settings > Control panel > Additional Software.

### CUxD Configure with Set



### CUxD Set Menu



### Add a new “Geräte” device

Select CUxD Gerätetyp (28) System with Function Exec, Leave Name empty, Use Geräte Icon “Fernbedienung 19 Tasten”.

Press Button “Gerät auf CCU erzeugen !”.



If all goes well, the Homematic WebUI Device Inbox lists the new device.  
Switch to the Homematic WebUI.

## Homematic WebUI

Select Settings > Device Inbox and add the CUxD.

The CUxD device created is **HM-RC-19 CUX2801001** with 16 push-button channels (called "Push button channel").

Name	Picture
HM-RC-19 CUX2801001	
HM-RCV-50 BidCoS-RF	
HmIP-eTRV-2 000A18A9A64DAC	
HMIP-PSM 0001D3C99C6AB3	
HmIP-RCV-50 61A7D7098E047C	

Channel	Room	Function	Last modified	
HM-RC-19 CUX2801001:1			07.07.2019 08:58:05	 <b>Short button press</b>
HM-RC-19 CUX2801001:2				 <b>Long button press</b>

Check the device channels and datapoints with a HTTP XML-API Request.

### HTTP XML-API Request

```
http://ccu-ip/addons/xmlapi/statelist.cgi
```

### HTTP XML-API Response

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<stateList>
- <device sticky_unreach="false" unreach="false" ise_id="1596" name="HM-RC-19 CUX2801001">
  - <channel ise_id="1597" name="HM-RC-19 CUX2801001:0" operate="" visible="" index="0">
    <datapoint ise_id="1598" name="CUxD.CUX2801001:0.LOWBAT" operations="5" timestamp="1562428172" valueunit="" valuetype="2" value="false" type="LOWBAT"/>
    <datapoint ise_id="1606" name="CUxD.CUX2801001:0.UNREACH" operations="5" timestamp="1562428172" valueunit="" valuetype="2" value="false" type="UNREACH"/>
    <datapoint ise_id="1602" name="CUxD.CUX2801001:0.STICKY_UNREACH" operations="7" timestamp="1562428172" valueunit="" valuetype="2" value="false" type="STICKY_UNREACH"/>
  </channel>
  - <channel ise_id="1610" name="HM-RC-19 CUX2801001:1" operate="true" visible="true" index="1">
    <datapoint ise_id="1628" name="CUxD.CUX2801001:1.PRESS_SHORT" operations="6" timestamp="156242685" valueunit="" valuetype="2" value="false" type="PRESS_SHORT"/>
    <datapoint ise_id="1627" name="CUxD.CUX2801001:1.PRESS_LONG" operations="6" timestamp="0" valueunit="" valuetype="2" value="" type="PRESS_LONG"/>
    <datapoint ise_id="1612" name="CUxD.CUX2801001:1.CMD_KILL" operations="2" timestamp="0" valueunit="" valuetype="20" value="" type="CMD_KILL"/>
    <datapoint ise_id="1611" name="CUxD.CUX2801001:1.CMD_EXEC" operations="2" timestamp="0" valueunit="" valuetype="20" value="" type="CMD_EXEC"/>
    <datapoint ise_id="1634" name="CUxD.CUX2801001:1.WORKING" operations="5" timestamp="1562428036" valueunit="" valuetype="2" value="false" type="WORKING"/>
  </channel>
  - <channel ise_id="1636" name="HM-RC-19 CUX2801001:2" operate="true" visible="true" index="2">
    <datapoint ise_id="1654" name="CUxD.CUX2801001:2.PRESS_SHORT" operations="6" timestamp="0" valueunit="" valuetype="2" value="" type="PRESS_SHORT"/>
    <datapoint ise_id="1653" name="CUxD.CUX2801001:2.PRESS_LONG" operations="6" timestamp="0" valueunit="" valuetype="2" value="" type="PRESS_LONG"/>
    <datapoint ise_id="1638" name="CUxD.CUX2801001:2.CMD_KILL" operations="2" timestamp="0" valueunit="" valuetype="20" value="" type="CMD_KILL"/>
  </channel>
</device>
</stateList>
```

## Update Domoticz Text Device

Update the text of the Domoticz Text Device (IDX=36) by a short button press of the Homematic IP Remote Control (HmIP-RC8) channel:1 (name HM-RC-19 CUX2801001:1) of the device HM-RC-19 CUX2801001.

### Homematic Program

Logic: IF CHANNEL:1 button is pressed short THEN execute script.

Name	Description	Condition (if...)	Activity (then..., or else...)	Action
CUX2801001Button1	Send Message to Domoticz Text Device	Channel status: HM-RC-19 CUX2801001:1 when Button press short	Script: ... immediately run	<input type="checkbox"/> intrinsic
<b>Condition: If...</b>				
Device selection <b>HM-RC-19 CUX2801001:1</b> when <b>Button press short</b>  AND  OR <input checked="" type="checkbox"/> Stop all current delays before performing the activity (e.g. retriggering).				
<b>Activity: Then...</b> <input checked="" type="checkbox"/> Stop all current delays before performing the activity (e.g. retriggering). <b>Script</b>  <b>string sDate = system.Date("%d.%m.%Y"); ! sDate = "09.08.201..."</b>   <b>Activity: Else...</b> <input type="checkbox"/> Stop all current delays before performing the activity (e.g. retriggering). 				

### Homematic Script

Submit Domoticz HTTP API Request via system command wget, to update the Domoticz Text Device.

```
wget -q -O - URL
```

The URL is the HTTP API Request as defined per Domoticz HTTP API/JSON documentation.

```
http://domoticz-ip:8080/json.htm?type=command&param=udevice&idx=iDX&nvalue=0&svalue=MESSAGE
```

```
string sDate = system.Date("%d.%m.%Y"); ! sDate = "09.08.2019";
string sTime = system.Date("%H:%M:%S"); ! sTime = "07:32:00";
string nIdx = 36;
string sMsg = sDate # " " # sTime # " - Threshold reached! Take action...";
string cAmp = "&";
string sDomUrl = "'http://domoticz-ip:8080/json.htm'";
string sUrl =
sDomUrl"?type=command#cAmp#param=udevice#cAmp#idx="#nIdx#cAmp#"nvalue=0#cAmp#svalue="#sMsg#"';

! Run the command with a return result
! Define the command to execute
dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State ("wget -q -O - "#sUrl);
! Set the return flag to 1 to be able to read the json result
dom.GetObject("CUxD.CUX2801001:1.CMD_QUERY_RET").State (1);
! Start the command, wait till completed and get the result JSON string, i.e.
! {"status" : "OK","title" : "Update Device"}
! NOTE: script running on CCU3 waits until completion - use commands which do not take long time.
string sRes = dom.GetObject("CUxD.CUX2801001:1.CMD_RET").State();
! WriteLine("VT="#sRes.VarType()#/ "#sRes); ! VT=4, {"status" : "OK","title" : "Update Device"}
! handle result
var d1 = dom.GetObject("DomoticzLog");
! Update the var with result text
if (sRes.Find("OK") > 0) {
  d1.Variable("Last update OK")
}
else {
  d1.Variable("Last update ERROR")
};
! WriteLine("VT="#d1.VarType()); ! VT=9
! WriteLine(d1.Variable()); ! shows the last update string
```

### Note

The Domoticz HTTP response string is checked against error and a Homematic system variable named “DomoticzLog” is updated.

### Button pressed and Domoticz Device Updated

The first button on the Homematic Remote Control HM-RC-19 is pressed and the text of the Domoticz device named “IAQM Status” gets updated.

Channel	Room	Function	Last modified	
Filter HM-RC-19 CUX2801001:1 Push button channel	Filter	Filter	09.07.2019 10:21:57	 Short button press

**IAQM Status**

09.07.2019 10:21:57 - Threshold reached! Take action...  
Last Seen: 2019-07-09 10:21:56  
Type: General, Text

 [Log](#) [Edit](#)

# Appendix Tools

## Homematic Statelist

### Purpose

To obtain the list of devices and datapoints, with selected properties, from the CCU. The focus is on getting the id of the devices and datapoints configured in the CCU. The ids are used by Homematic scripts and Domoticz Automation Events (dzVents).

### Solution

The Homematic addon XML-API is used to get the data.

A HTTP XML-API request is submitted to the CCU, with the statelist script (statelist.cgi) as parameter.

The HTTP response is an XML tree (as plain text; XML is used as information storage) with all devices and datapoints.

The XML tree is parsed to get the device name(s) & id plus the associated datapoint(s) id, type, and value.

## Node-RED

Developed with Node-RED (1.3.4) and the modules [node-red-dashboard](#) (2.29.1), [node-red-node-ui-list](#) (0.3.4).

### Node-RED Dashboard

Homematic Statelist	
<b>Device</b>	<b>Datapoints</b>
MakeLab	1415 ACTUAL_TEMPERATURE=18.400000 1416 ACTUAL_TEMPERATURE_STATUS=0 1417 BOOST_MODE=false 1418 BOOST_TIME=0 1419 CONTROL_DIFFERENTIAL_TEMPERATURE= 1420 CONTROL_MODE= 1421
Device ID	1390
Last Update	2021-05-19 09:49:28
<b>REFRESH</b>	
<b>Settings</b>	
CCU IP:	192.168.1.70

The dashboard, tab Homematic Statelist, has a **Device** dropdown list which populates a list with the **Datapoints** for the device selected.

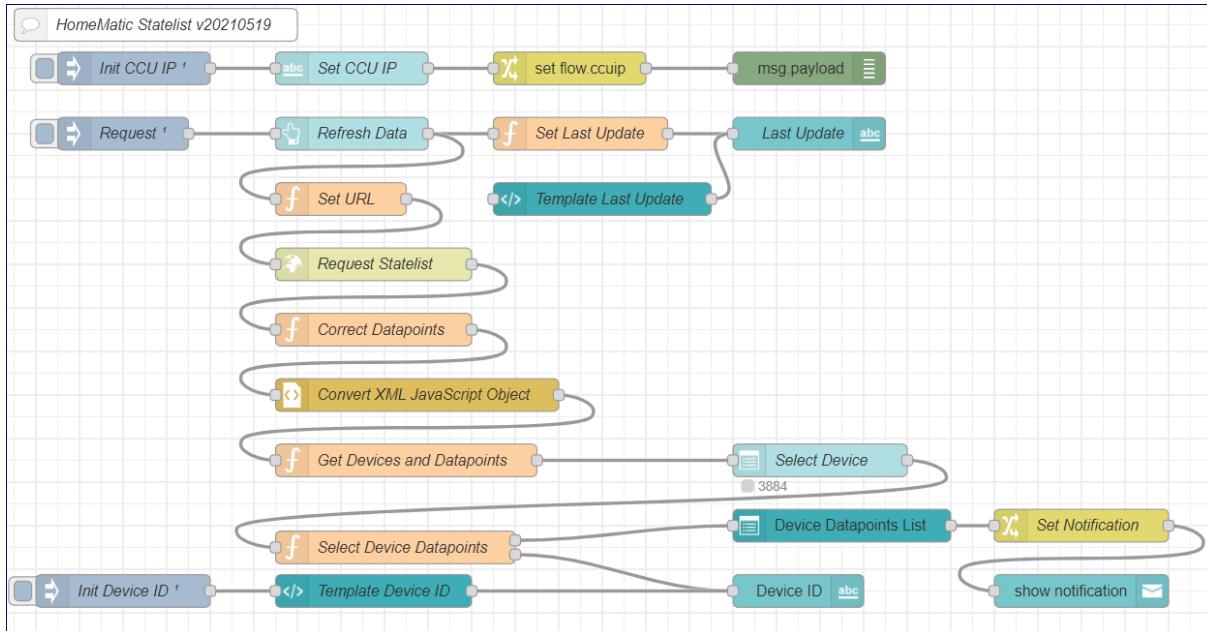
The list items are added according to the XML tree structure - by channel and by datapoint ID (property `ise_id`).

Clicking on a list item, a notification is shown with the list item data ID : Datapoint = Value.

The button **Refresh** gets the latest statelist and updates the Last Update field.

The input field **CCU IP** sets the IP address of the CCU.

## Node-RED Flow



### Summary

The http request node sends an HTTP XML-API request to the CCU to obtain the statelist in XML format.

The XML tree is converted to a JavaScript object which is parsed to arrays holding JSON formatted entries for the devices and datapoints.

These arrays are used for the UI based on the Node-RED [Dashboard](#).

The devices are added, from the devices array, to a ui\_dropdown node.

Selecting a device, lists the assigned datapoints, from the datapoints array, in a ui\_list node.

## HTTP XML-API Response

The response contains the statelist with devices, channels and datapoints.

Example with data extract:

Elements > stateList > device > channel > datapoint.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<stateList>
<device config_pending="false" unreach="false" ise_id="2530" name="Briefkasten Status">
  <channel ise_id="2531" name="Briefkasten Status:0" operate="true" visible="true" index="0">
    <datapoint ise_id="2532" name="HmIP-RF.0000DA498D5859:0.CONFIG_PENDING" operations="5"
      timestamp="1583826706" valueunit="" valuetype="2" value="false" type="CONFIG_PENDING"/>
    <datapoint ise_id="2536" name="HmIP-RF.0000DA498D5859:0.DUTY_CYCLE" operations="5"
      timestamp="1583826706" valueunit="" valuetype="2" value="false" type="DUTY_CYCLE"/>
    ...
    <datapoint ise_id="2547" name="HmIP-RF.0000DA498D5859:0.SABOTAGE" operations="5"
      timestamp="1583826706" valueunit="" valuetype="2" value="false" type="SABOTAGE"/>
    <datapoint ise_id="2551" name="HmIP-RF.0000DA498D5859:0.UNREACH" operations="5"
      timestamp="1583826706" valueunit="" valuetype="2" value="false" type="UNREACH"/>
    <datapoint ise_id="2555" name="HmIP-RF.0000DA498D5859:0.UPDATE_PENDING" operations="5"
      timestamp="1582992737" valueunit="" valuetype="2" value="false" type="UPDATE_PENDING"/>
  </channel>
  ... more channels
</device>
<device config_pending="false" unreach="true" ise_id="3597" name="Eingang Monitor">
  ... channels & datapoints
</device>
... more devices
</stateList>
```

## Function Node Get Devices and Datapoints

```
/*
 * return a json array with devices used for the ui_dropdown
 * set flow context "datapoints" for the ui_template to display the datapoints for the selected device
id
Notes:
* Example device entry: deviceobj["Briefkasten Status"] = 2530;
* Example datapoint entry:
device:"2530",datapoint:{ "id": "2532", "type": "CONFIG_PENDING", "value": "false", "name": "HmIP-"
RF.0000DA498D5859:0.CONFIG_PENDING" }
* second array not used = replaced by flow context "datapoints" - kept in case any enhancements planned
*/
const DEBUG = false;
// get the statelist from xml into json format
statelist = msg.payload.stateList;
// Get the statelist with the devices
if (DEBUG) node.warn(statelist);
// Get the devices array
devices = statelist.device
if (DEBUG) node.warn(devices.length);
arrdevices = [];
arrdatapoints = [];
// Loop over the devices
devices.forEach(function(device) {
    // node.warn(device);
    devicename = device.$.name;
    deviceise_id = device.$.ise_id;
    deviceobj = {}
    deviceobj[devicename] = deviceise_id;
    arrdevices.push(deviceobj);
    channels = device.channel;
    if (DEBUG) node.warn("Channel Name: " + device.$.name);
    channels.forEach(function(channel){
        channeldatapoints = channel.datapoint;
        if (channeldatapoints !== undefined) {
            // node.warn("Channel datapoints: " + channeldatapoints);
            channeldatapoints.forEach(function(datapoint){
                //node.warn(datapoint);
                obj = datapoint.$;
                if (obj !== undefined) {
                    if (DEBUG) node.warn(devicename + ":dps=" + obj.ise_id + ":" + obj.type + "=" + obj.value);
                    dpobj = {
                        "device": deviceise_id,
                        "datapoint": {
                            "id": obj.ise_id, "type": obj.type, "value": obj.value, "name": obj.name
                        }
                    };
                    arrdatapoints.push(dpobj);
                    //dpstring="dev:" + devicename + ",dp:" + obj.ise_id + ",t:" + obj.type + ",v:" + obj.value;
                    // arrdatapoints.push(dpstring);
                }
            });
        }
    });
});
if (DEBUG) node.warn(arrdatapoints.length);
flow.set("datapoints", arrdatapoints);
if (DEBUG) node.warn(flow.get("datapoints"));
msgdevices = {};
msgdevices.options = arrdevices;
if (arrdevices.length > 0){
    const name = Object.keys(arrdevices[0])[0];      //MakeLab
    const value = Object.values(arrdevices[0])[0];     //1541
    msgdevices.payload = value;
}
if (DEBUG) node.warn(arrdevices[0]);
if (DEBUG) node.warn(msgdevices.options)
msgdatapoints = {};
msgdatapoints.payload = arrdatapoints;

return [msgdevices, msgdatapoints];
}
```

## Function Node Select Device Data

```
/*
The purpose of this function is to
* get all datapoints for the selected device using the device id (ise_id)
* return an array with datapoint entries: id: type=value

Notes:
* The type is the short name of the datapoint
* The array is used for the ui_template html list
*/
const DEBUG = false;

// get the device id, i.e. 2530 for the selected device
var deviceid = msg.payload;
if (DEBUG) node.warn("Selected: " + deviceid);
var msgdeviceid = {};
msgdeviceid.payload = deviceid;
// if (deviceid === null) msgdeviceid.payload = "";

// get all the datapoints by device
// device:"2530",datapoint:{id:"2532",type:"CONFIG_PENDING",value:"false",name:"HmIP-
RF.0000DA498D5859:0.CONFIG_PENDING"}
datapoints = flow.get("datapoints");

msgdevicedatapoints = {};
msglist = [];
datapoints.forEach(function(datapoint){
    if (datapoint.device == deviceid) {
        dp = datapoint.datapoint;
        let obj = {};
        obj.datapoint = dp.id + ":" + dp.type + "=" + dp.value;
        // obj.title = dp.type + "=" + dp.value;
        // obj.description = dp.id;
        obj.title = "<p style='font-size:small;'>" + dp.type + "=" + dp.value + "</p>";
        obj.description = "<I>" + dp.id + "</I>";
        msglist.push(obj);
        if (DEBUG) node.warn("Added: " + obj);
    }
});
msgdevicedatapoints.payload = msglist;
if (DEBUG) node.warn(msgdevicedatapoints.payload);

return [msgdevicedatapoints,msgdeviceid];
```

## Flow Source

```

node.warn(msgdevicedatapoints.payload);\\n\\nreturn
[msgdevicedatapoints,msgdeviceid];\\n", "outputs":2, "noerr":0, "initialize": "", "finalize": "", "libs":[], "x":370, "y":500, "wires": [{"3cbfc25.4a8983e"}, {"497f0cf6.c8c974"}]}, {"id": "e57ca04b.28462", "type": "ui_text", "z": "f6e89496.2b34d8", "group": "431a79e6.122218", "order": 3, "width": "0", "height": "0", "name": "Last Update", "label": "Last Update", "format": "{msg.payload}", "layout": "row-spread", "x": 750, "y": 120, "wires": []}, {"id": "f08ed0e9.7aac9", "type": "ui_button", "z": "f6e89496.2b34d8", "name": "Refresh Data", "group": "431a79e6.122218", "order": 4, "width": "0", "height": "0", "passthru": true, "label": "Refresh", "tooltip": "", "color": "", "bgcolor": "", "icon": "", "payload": "", "payloadType": "str", "topic": "", "topicType": "str", "x": 330, "y": 120, "wires": [{"e61959f0.339d18"}, {"7b2f11e.db2ee"}]}, {"id": "e61959f0.339d18", "type": "function", "z": "f6e89496.2b34d8", "name": "Set Last Update", "func": "// Get the actual date & time in format YYYY-MM-DD HH:MM:SS.\\n// See also various formats possible.\\n// Function to convert the unix timestamp to string\\nfunction GetDateTime(){\\n    var now = new Date();\\n    var year = now.getFullYear();\\n    var month = now.getMonth() + 1; \\n    var day = now.getDate();\\n    var hours = now.getHours();\\n    var minutes = now.getMinutes();\\n    var seconds = now.getSeconds();\\n\\n    if(month.toString().length == 1) { \\n        month = '0'+month;\\n    }\\n    if(day.toString().length == 1) { \\n        day = '0'+day;\\n    }\\n    if(hours.toString().length == 1) {\\n        hours = '0'+hours;\\n    }\\n    if(seconds.toString().length == 1) {\\n        seconds = '0'+seconds;\\n    }\\n    \\n    // YYYY-MM-DD HH:MM:SS\\n    var dts = year+'-'+month+'-'+day+'\\n'+hours+':'+minutes+':'+seconds;\\n    \\n    // Other Formats\\n    // HH:MM:SS\\n    // var dts = hours+\\n'+minutes+\\n'+seconds;\\n    \\n    // YYYYMMDDHHMMSS\\n    //var dts = year+month+day+hours+minutes+seconds;\\n\\n    return dts;\\n}\\n\\nmsg.payload = GetDateTime();\\nreturn
msg", "outputs": 1, "noerr": 0, "x": 540, "y": 120, "wires": [{"e57ca04b.28462"}]}, {"id": "4ab5f513.58663c", "type": "ui_text_input", "z": "f6e89496.2b34d8", "name": "Set CCU3 IP", "label": "CCU3 IP:", "tooltip": "CCU3 IP Address", "group": "fc669d26.b9e35", "order": 7, "width": "0", "height": "0", "passthru": true, "mode": "text", "delay": "300", "topic": "", "topicType": "str", "x": 330, "y": 60, "wires": [{"b449a70b.f6aa78"}]}, {"id": "7b2f11e.db2ee", "type": "function", "z": "f6e89496.2b34d8", "name": "Set URL", "func": "\\nmsg.url = \"http://\" + flow.get(\"ccuip\") + \"/config/xmlapi/statelist.cgi\";\\n// node.warn(msg.url);\\nreturn
msg", "outputs": 1, "noerr": 0, "initialize": "", "finalize": "", "x": 320, "y": 180, "wires": [{"6029cd74.adc594"}]}, {"id": "b449a70b.f6aa78", "type": "change", "z": "f6e89496.2b34d8", "name": "", "rules": [{"t": "set", "p": "ccuip", "pt": "flow", "to": "payload", "tot": "msg"}], "action": "", "property": "", "from": "", "to": "", "reg": false, "x": 530, "y": 60, "wires": [{"9098dad0.b05908"}]}, {"id": "9098dad0.b05908", "type": "debug", "z": "f6e89496.2b34d8", "name": "", "active": true, "tosidebar": false, "console": false, "tostatus": false, "complete": "payload", "targetType": "msg", "x": 750, "y": 60, "wires": []}, {"id": "70d8204a.e6663", "type": "inject", "z": "f6e89496.2b34d8", "name": "Init CCU3 IP", "props": [{"p": "payload"}, {"p": "topic", "vt": "str"}], "repeat": "", "crontab": "", "once": true, "onceDelay": 0.1, "topic": "", "payload": "ccu-ip", "payloadType": "str", "x": 130, "y": 60, "wires": [{"4ab5f513.58663c"}]}, {"id": "497f0cf6.c8c974", "type": "ui_text", "z": "f6e89496.2b34d8", "group": "431a79e6.122218", "order": 2, "width": "0", "height": "0", "name": "", "label": "Device ID", "format": "{msg.payload}", "layout": "row-spread", "x": 740, "y": 540, "wires": []}, {"id": "920679aa.8e2e38", "type": "ui_template", "z": "f6e89496.2b34d8", "group": "431a79e6.122218", "order": 5, "width": 0, "height": 0, "format": "<!--\\nSet the background color & text size of the ui_text label \\\"Aktualisiert\\\".\\n-->\\n<style>\\n    .nr-dashboard-widget-Device_ID {\\n        background-color:#F5F5F5 !important;\\n        font-size: medium !important;\\n    }\\n</style>", "storeOutMessages": true, "fwdInMessages": true, "resendOnRefresh": true, "templateScope": "local", "x": 350, "y": 540, "wires": [{"497f0cf6.c8c974"}]}, {"id": "7f1096fd.409518", "type": "inject", "z": "f6e89496.2b34d8", "name": "Init Device ID", "repeat": "", "crontab": "", "once": true, "onceDelay": 0.2, "topic": "", "payload": "not selected", "payloadType": "str", "x": 120, "y": 540, "wires": [{"920679aa.8e2e38"}]}, {"id": "c4b87acc.01d17", "type": "ui_template", "z": "f6e89496.2b34d8", "group": "431a79e6.122218", "name": "Template Last Update", "order": 6, "width": 3, "height": 1, "format": "<!--\\nSet the background color & text size of the ui_text label \\\"Aktualisiert\\\".\\n-->\\n<style>\\n    background-color:#F5F5F5 !important;\\n    font-size: small !important;\\n</style>", "storeOutMessages": true, "fwdInMessages": true, "resendOnRefresh": true, "templateScope": "local", "x": 560, "y": 180, "wires": [{"e57ca04b.28462"}]}, {"id": "871e1921.d25df8", "type": "ui_toast", "z": "f6e89496.2b34d8", "position": "top right", "displayTime": "3", "highlight": "", "sendall": true, "outputs": 0, "ok": "OK", "cancel": "", "raw": false, "topic": "", "name": "", "x": 1000, "y": 540, "wires": []}, {"id": "f3e59ee0.8ca998", "type": "change", "z": "f6e89496.2b34d8", "name": "Set Notification", "rules": [{"t": "set", "p": "payload", "pt": "msg", "to": "$.payload.datapoint", "tot": "jsonata"}], "action": "", "property": "", "from": "", "to": "", "reg": false, "x": 1000, "y": 480, "wires": [{"871e1921.d25df8"}]}, {"id": "3cbfc25.4a8983e", "type": "ui_list", "z": "f6e89496.2b34d8", "group": "187837da.b7cfe8", "name": "Device Datapoints List", "order": 0, "width": 6, "height": 6, "lineType": "two", "actionType": "click", "allowHTML": true, "outputs": 1, "topic": "", "x": 780, "y": 480, "wires": [{"f3e59ee0.8ca998"}]}, {"id": "431a79e6.122218", "type": "ui_group", "name": "Device", "tab": "5a022734.6be7c8", "order": 1, "disp": true, "width": 6, "collapse": false}, {"id": "fc669d26.b9e35", "type": "ui_group", "name": "Settings", "tab": "5a022734.6be7c8", "order": 3, "disp": true, "width": 6, "collapse": false}, {"id": "187837da.b7cfe8", "type": "ui_group", "name": "Datapoints", "tab": "5a022734.6be7c8", "order": 2, "disp": true, "width": 6, "collapse": false}, {"id": "5a022734.6be7c8", "type": "ui_tab", "name": "Homematic Statelist", "icon": "dashboard", "disabled": false, "hidden": false}]

```

(Node-RED Import > Paste Flow json)

Last page initially left blank.