Viessmann API and Node Red – Part 4

January 19 , 2023 IoT , JSON , Node-Red , Viessmann API settings change

Contents [Hide]

Settings change

After we in the previous ones chapters learned have how to access the API and how to get data from it read out can see _ we now how to adjust the heating – e.g. the temperature _ or the operating mode change via the API can .

This tutorial is adapted to the changed API from February 2023 . Instead of the endpoint *equipment* becomes now the endpoint *features* used .

Look we us first once the JSON object from the feature query . This here shown abstract shows or . sets the normal temperature of heating circuit 2 (Feature: heating.circuits .2.operating.programs.normal): A feature starts always with “properties ” and runs until next “properties” line.

Example : normal temperature Set heating circuit 2

```
{
  "properties": {
    "active": {
      "value": false,
      "type": "boolean"
    },
    "demand": {
      "value": "unknown",
      "type": "string"
    },
    "temperature": {
      "value": 17,
      "unit": "celsius",
      "type": "number"
```

```
            }
          },
          "commands" : {
            " setTemperature " : {
              " uri " :
"https://api.viessmann.com/iot/v1/equipment/installations/999999/gateways/1234567890123456/devices/0/features/heating.circuits.2.operating.programs.normal/commands/setTemperature" ,
              "name": "setTemperature",
              "isExecutable": true,
              "params": {
                "targetTemperature": {
                  "type": "number",
                  "required": true,
                  "constraints": {
                    "min": 3,
                    "max": 37,
                    "stepping": 1
                  }
                }
              }
            }
          },
          " apiVersion " : 1 ,
          " uri " :
"https://api.viessmann.com/iot/v1/equipment/installations/999999/gateways/1234567890123456/devices/0/features/heating.circuits.2.operating.programs.normal" ,
          " gatewayId " : "1234567890123456" ,
          "feature": "heating.circuits.2.operating.programs.normal",
          "timestamp": "2022-11-24T08:22:15.667Z",
          "isEnabled": true,
          "isReady": true,
          " deviceId " : "0"
        },
```

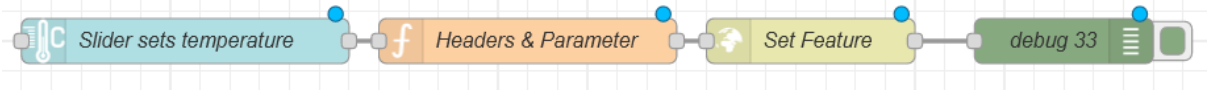
4
2
4
3

The one set Target temperature is 17 degrees Celsius – how to do it reads out , is in the previous one Chapter described .

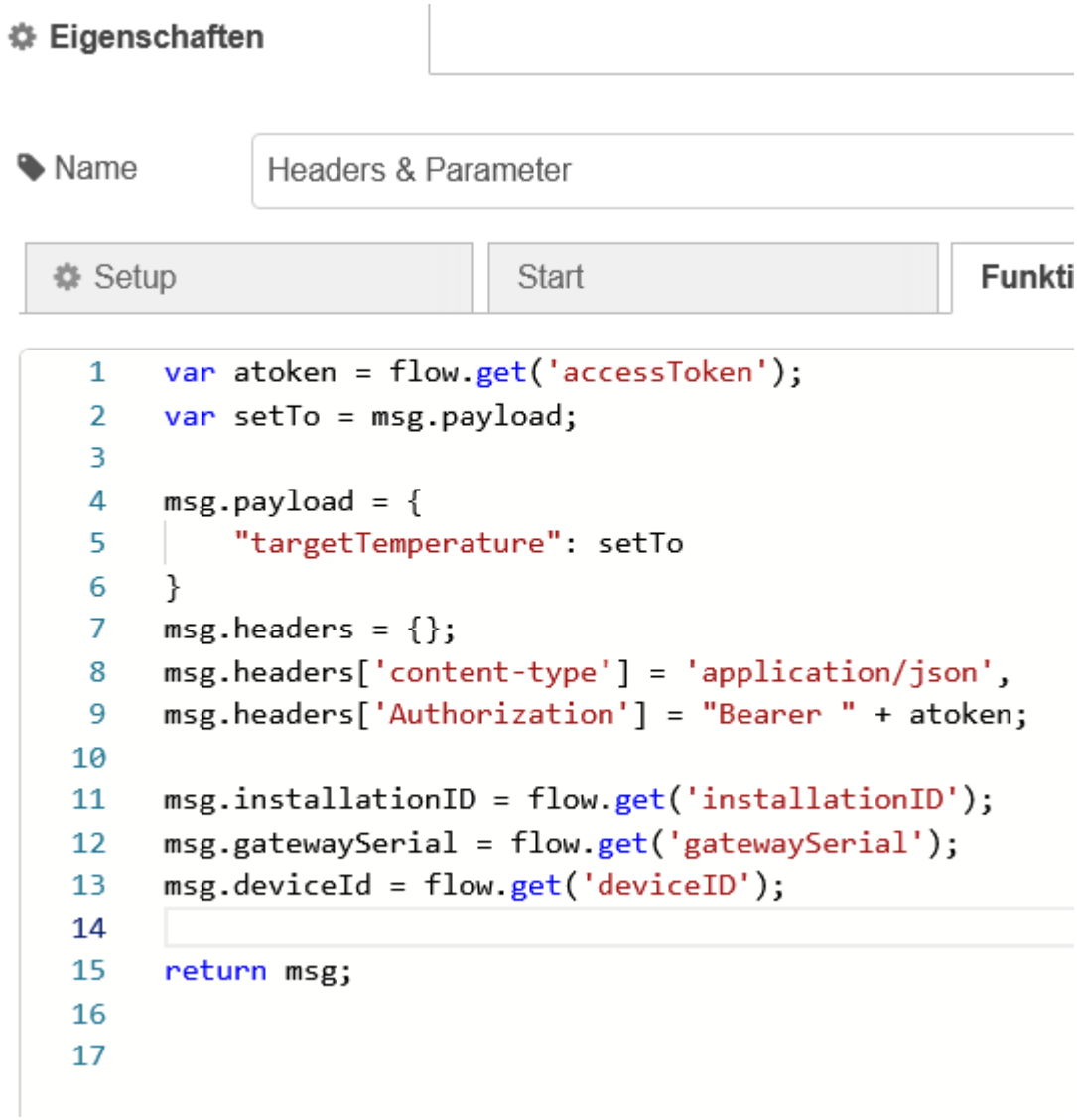
Every feature that has a “commands” not to empty Content can – theoretically – be manipulated become . In our Example If this is the case. It says there `"commands" : { " setTemperature " : {...` followed by a URL and some constraints .
The “ isExecutable ” parameter specifies whether itself a Command in a feature **below the current ones circumstances** carry out leaves or not . If “ isExecutable ” = true lets the command _ execute , otherwise not . Example : You cannot use the “ Eco” operating program set because _ the system is in standby mode .

How to ask Do we now enter the normal temperature of heating circuit 2 ?

Straight forward



First comes a Dashboard Slider Node that we set to provide “ Output only on release” . The target value plugged now in msg.payload . The function node that contains the http-Reuest header accordingly processed , is How follows programmed :



or . as Javascript snippet for copy&paste :

```
var atokens = flow.get ( ' accessToken ' );
var setTo = msg.payload;

msg.payload = {
  "targetTemperature": setTo
}
msg.headers = {};
msg.headers['content-type'] = 'application/json';
msg.headers['Authorization'] = "Bearer " + atoken;

msg.installationID = flow.get('installationID');
msg.gatewaySerial = flow.get('gatewaySerial');
msg.deviceId = flow.get('deviceID');

return msg;
```

Another one NOTE : The ones in msg.header hidden information is quite “ sticky ”, meaning there is often information out of previous http requests in it stand remain . To

one clean base too you should have before _ each Set the header this with one `msg.headers = {};` delete .
The *http Rquest Node* “set feature” contains the one for the normal temperature Heating circuit 2 required URI:

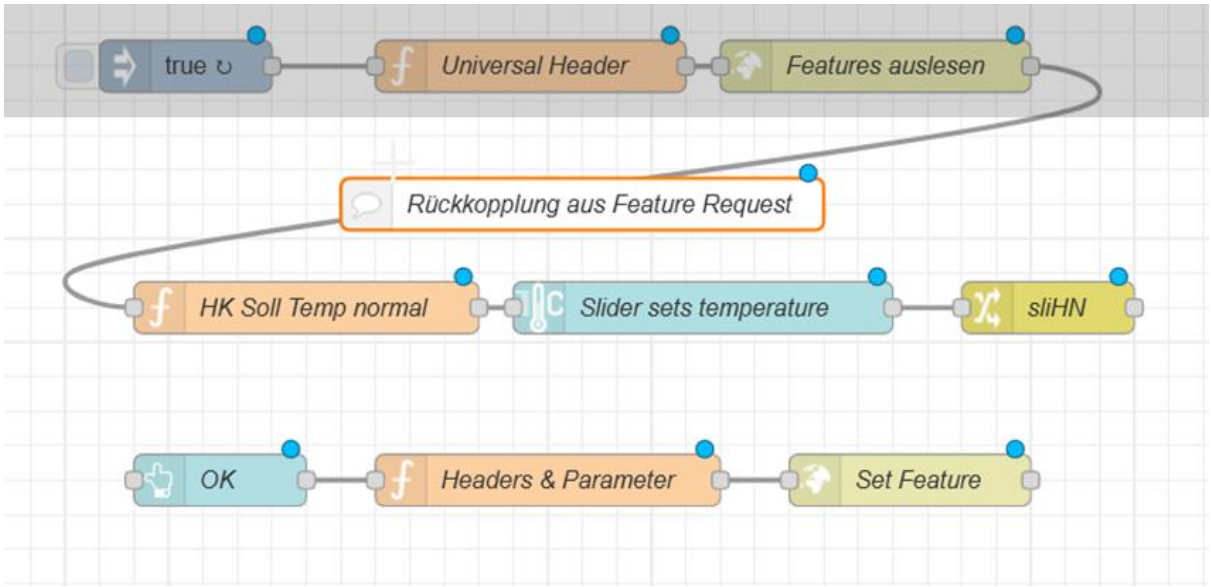
```
https://api.viessmann.com/iot/v1/features/installations/{{installationID}}/gateways/{{gatewaySerial}}/devices/{{deviceId}}/features/heating.circuits.2.operating . _ programs.normal /commands/setTemperature
```

Complete! Or not ?

Flow operationally safe make

use the dashboard one Mobile device , it can quickly happen that you use the slider temperature selector inadvertently adjusted . That is naturally not in the spirit of the inventor . I have hence the slider and the command decoupled . For the Sending the new one Temperature still has to be a Confirmation button clicked become .

That looks in total How follows out of :



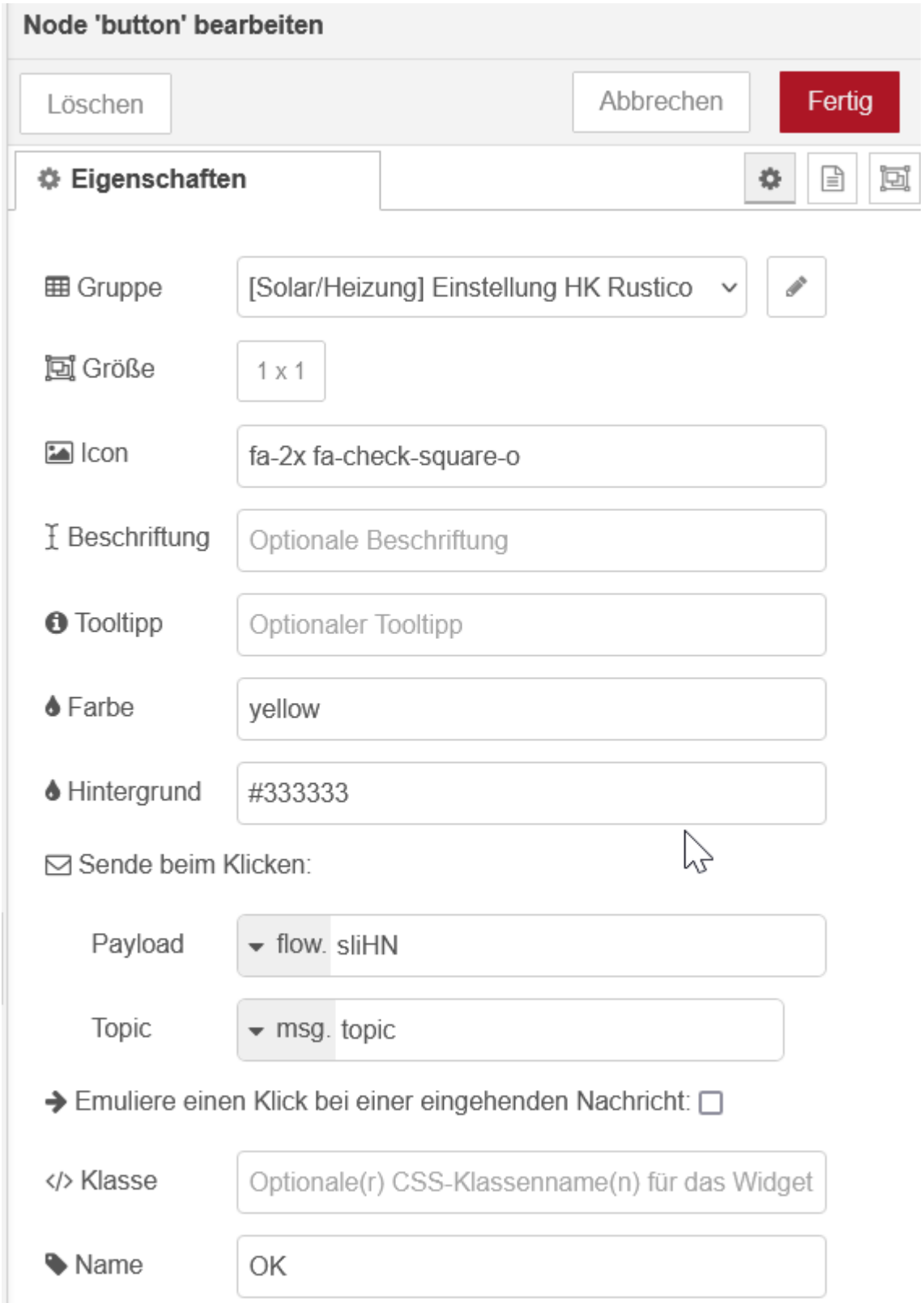
The top ones three Gray underlying nodes _ You certainly have the feature request already ; she serve just the feedback from the current in the heating set value. Since ours application Yes not the only one Input option is – there is Yes nor the ViCare app or the buttons on the device itself – is the feedback very important to always be up to date applicable Attitude to see .

The function node with the name “HK target temp normal” becomes simply with dem already existing “ Read feature ” *http request* wired . This node does nothing other than located in the heating system applicable value _ the JSON file . The logic is out of dem previous Chapter known .

```
var featureArray = msg.payload.data;
var idx = featureArray.findIndex((element) => element.feature === 'heating.circuits.2.operating.programs.normal');
msg.payload = msg.payload.data[idx].properties.temperature.value;
flow.set("hkNormal", msg.payload);
return msg;
```

Thereafter comes the Dashboard Slider Node setting How above . Last node in this strand is still a change node that provides the flow variable according to the slider value Save usage in the next step . Now when the slider moves will happen _ How wanted first _ nothing . This only happens in the next step, because only the lower ones take three nodes after dem Triggering the dashboard button triggers the setting before .

The Dashboard Button Node becomes How below shown set - where I changed the display of the button icon have :



The payload contains the flow variable with the set Temperature .

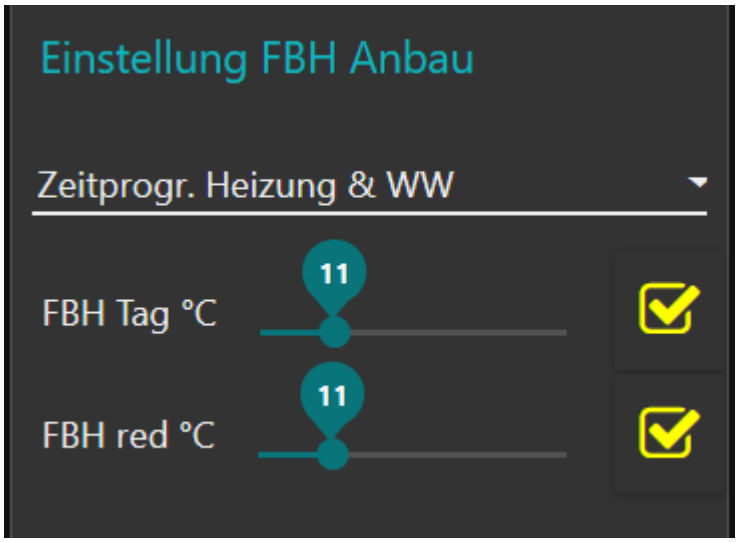
The *Function Node Headers & Parameters* looks almost like the “ straight forward ” example . Only line 2 is different

```
var atoken = flow.get('accessToken');
var setTo = flow.get('sliHN');

msg.payload = {
   "targetTemperature": setTo
}
msg.headers = {};
msg.headers['content-type'] = 'application/json';
msg.headers['Authorization'] = "Bearer " + atoken;

msg.installationID = flow.get('installationID');
msg.gatewaySerial = flow.get('gatewaySerial');
msg.deviceId = flow.get('deviceID');
;
return msg;
```

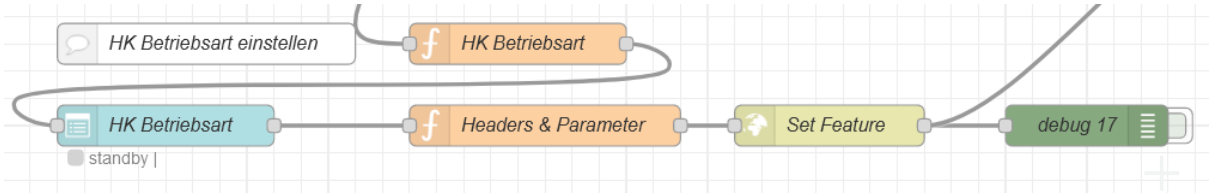
The *http request node* " set feature" is identical for the straight forward example

```
https://api.viessmann.com/iot/v1/features/installations/{{installationID}}/gateways/{{gatewaySerial}}/devices/{{deviceId}}/features/heating.circuits.2.operating . _ programs.normal /commands/setTemperature
```

And here nor the display of the dashboard Selection Operating mode , slider for normal and reduced Temperature . Release the yellow "OK hooks". out of.



Example : operating mode set



Depending on the system, there are several Operating modes : For me these are *off* , *hot water only* , *Time program Heating and hot water* , *permanent reduced* and *permanently normal* .

This you adjust depending on heating circuit How follows a ; in our Example for heating circuit 2:

```
heating.circuits . 2.operating.modes.active /commands/ s etMode
```

Basically _ functions everything like that when setting the temperature . The Function Node HK operating mode is the feedback again from the feature query *Operating mode* . The setting he follows above a *Dashboard Dropdown Node,* the Headers *& Parameters* Node sets the payload to one of the prescribed ones Values “mode: standby”, “mode: dhw ”, “mode: dhwAndHeating ”, “mode: forcedReduced ”, “mode: forcedNormal ”.
The http request has the URL

```
https://api.viessmann.com/iot/v1/features/installations/{{installationID}}/gateways/{{gatewaySerial}}/devices/{{deviceId}}/features/heating.circuits.2.operating . _ modes.active /commands/setMode
```

You can find the associated JSON file for Node-Red here :

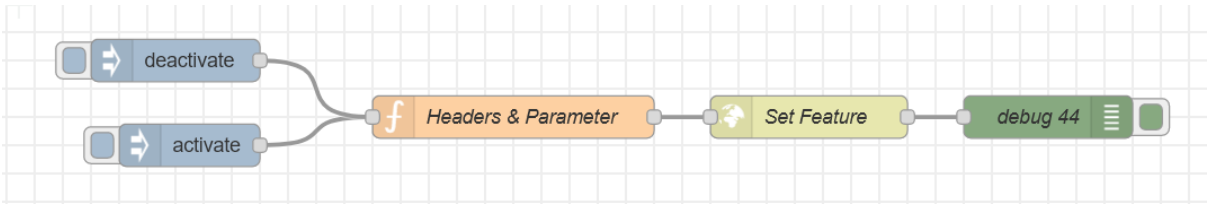

```
[{"id":"354cb6e083ea9537","type":"function","z":"31c57f8640528976","name":"HK Betriebsart","func":"var featureArray = msg.payload.data;\nvar idx = featureArray.findIndex((element) => element.feature === 'heating.circuits.2.operating.modes.active');\nmsg.payload = msg.payload.data[idx].properties.value.value;\nreturn msg;\n","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":420,"y":2560,"wires":[["9ec49bb67331871e"]]},{"id":"9ec49bb67331871e","type":"ui_dropdown","z":"31c57f8640528976","name":"HK Betriebsart","label":"","tooltip":"","place":"Betriebsart","group":"ba68cdf22ac62395","order":1,"width":6,"height":1,"passthru":false,"multiple":false,"options":[{"label":"aus","value":"standby","type":"str"},{"label":"nur WW","value":"dhw","type":"str"},{"label":"Zeitprogr. Heizung & WW","value":"dhwAndHeating","type":"str"},{"label":"dauernd reduziert","value":"forcedReduced","type":"str"},{"label":"dauernd Tag","value":"forcedNormal","type":"str"}],"payload":"","topic":"topic","topicType":"msg","className":"","x":160,"y":2620,"wires":[["8c38cdcc5d584653"]]},{"id":"8c38cdcc5d584653","type":"function","z":"31c57f8640528976","name":"Headers & Parameter","func":"var atoken = flow.get('accessToken')\nvar setTo = msg.payload\n\nmsg.payload = {\n \"mode\": setTo\n}\nmsg.headers = {};\nmsg.headers['content-type'] = 'application/json',\nmsg.headers['Authorization'] = \"Bearer \" + atoken;\n\nmsg.installationID = flow.get('installationID');\nmsg.gatewaySerial = flow.get('gatewaySerial');\nmsg.deviceId = flow.get('deviceID');\n;\n\nreturn msg;\n\n","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":440,"y":2620,"wires":[["3f5d70a0adfe2d5a"]]},{"id":"3f5d70a0adfe2d5a","type":"http request","z":"31c57f8640528976","name":"Set Feature","method":"POST","ret":"obj","paytoqs":"ignore","url":"https://api.viessmann.com/iot/v1/equipment/installations/{{installationID}}/gateways/{{gatewaySerial}}/devices/{{deviceId}}/features/heating.circuits.2.operating.modes.active/commands/setMode","tls":"","persist":false,"proxy":"","insecureHTTPParser":false,"authType":"","senderr":false,"headers":[],"credentials":{},"x":650,"y":2620,"wires":[["ac3e542798c51fa6","47147733bf11ab71"]]},{"id":"ac3e542798c51fa6","type":"debug","z":"31c57f8640528976","name":"debug 17","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"payload","targetType":"msg","statusVal":"","statusType":"auto","x":840,"y":2620,"wires":[]},{"id":"ba68cdf22ac62395","type":"ui_group","name":"Einstellung HK Rustico","tab":"7e5ee24d.2d5ae4","order":5,"disp":true,"width":"6","collapse":false,"className":""},{"id":"7e5ee24d.2d5ae4","type":"ui_tab","name":"Solar/Heizung","icon":"dashboard","order":3,"disabled":false,"hidden":false}]
```


Example : “I want hot water ”

Behind this The app's quick dial function is hidden the execution of the “ Hot water” command once provide ” or _ heating.dhw.oneTimeCharge . This will then the hot water boiler one-time up to chosen temperature heated up , no matter which one operation mode the heating _ located – too if it is set to OFF .

The command is on the one hand something different than the ones above described , on the other hand also not more difficult – you just have to on it come !

I describe here just the principle . Install in yours Application and with Switch Mistake you can do it yourself – now You can do that , right ?



Send the two triggers a string namely either “activate” or “deactivate”. The following function node becomes The URL to be accessed is constructed from this and passed on to the http request node .

```
var atoken = flow.get('accessToken');
var setTo = msg.payload;
var installationID = flow.get('installationID');
var gatewaySerial = flow.get('gatewaySerial');
var deviceId = flow.get('deviceID');

msg.payload = {};
msg.headers = {};
msg.headers['content-type'] = 'application/json',
msg.headers['Authorization'] = "Bearer " + atoken;

msg.url = "https://api.viessmann.com/iot/v1/features/installations/" + installationID + "/gateways/" +
gatewaySerial + "/devices/" + deviceId + "/features/heating.dhw.oneTimeCharge/commands/" + setTo;

return msg;
```

What happens there: The function node sets the payload to one empty set `msg.payload ={}` , that is important because the http request does not transmit a payload receive should . In the penultimate line the URL is put together , ie expanded to include the desired one Activation status .
The function or . the status results itself here so from the URL and not How otherwise out of a parameter in the http request header that is passed becomes .

The *http request node* becomes configured as POST and has nothing in the URL line inside stand .

You send it twice the same status – e.g twice *activate* , then Is there a Error message of the type:

Here is the JSON snippet for import

[{"id":"549e96f8b1dcf2b1","type":"http request","z":"31c57f8640528976","name":"Set Feature","method":"POST","ret":"obj","paytoqs":"ignore","url":"","tls":"","persist":false,"proxy":"","insecureHTTPParser":false,"authType":"","senderr":false,"headers":[],"x":650,"y":2880,"wires":[["ed98324b1f418d0b"]]},{"id":"b1ab78c225f82b3b","type":"function","z":"31c57f8640528976","name":"Headers & Parameter","func":"var atoken = flow.get('accessToken');\nvar setTo = msg.payload;\nvar installationID = flow.get('installationID');\nvar gatewaySerial = flow.get('gatewaySerial');\nvar deviceId = flow.get('deviceID');\n\nmsg.payload = {};\nmsg.headers = {};\nmsg.headers['content-type'] = 'application/json',\nmsg.headers['Authorization'] = \"Bearer \" + atoken;\nmsg.url = \"https://api.viessmann.com/iot/v1/features/installations/\" + installationID + \"/gateways/\" + gatewaySerial + \"/devices/\" + deviceId + \"/features/heating.dhw.oneTimeCharge/commands/\" + setTo;\n\nreturn msg;\n\n","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":440,"y":2880,"wires":[["549e96f8b1dcf2b1"]]},{"id":"ed98324b1f418d0b","type":"debug","z":"31c57f8640528976","name":"debug 44","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"false","statusVal":"","statusType":"auto","x":820,"y":2880,"wires":[]},{"id":"0b018d39b29952e0","type":"inject","z":"31c57f8640528976","name":"","props":[{"p":"payload"},{"p":"topic","vt":"str"}],"repeat":"","crontab":"","once":false,"onceDelay":0.1,"topic":"","payload":"activate","payloadType":"str","x":210,"y":2900,"wires":[["b1ab78c225f82b3b"]]},{"id":"78d7db200a51816d","type":"inject","z":"31c57f8640528976","name":"","props":[{"p":"payload"},{"p":"topic","vt":"str"}],"repeat":"","crontab":"","once":false,"onceDelay":0.1,"topic":"","payload":"deactivate","payloadType":"str","x":200,"y":2840,"wires":[["b1ab78c225f82b3b"]]},{"id":"22529d0c5cc35dea","type":"comment","z":"31c57f8640528976","name":"dhw One Time charge","info":"","x":220,"y":2800,"wires":[]}]

That's it.

Comments , error messages and a small Thank you very much would me happy make .

Post navigation

← Viessmann API and Node-Red – Part 5 Viessmann API and Node-Red – Part 3 →

4 thoughts on “Viessmann API and Node Red – Part 4”

1. Jürgen Grett

   February 21 , 2024 at 7:52 p.m

   Hello Chris,
   Running everything like that desired .
   Thanks for the great , understandable and detailed Directions .
   Many greetings Jürgen

   Answer

2. thilo

   March 8, 2023 at 3:38 p.m

   Hello, how can I use “ speed dials ” like for the Example “I want Activate hot water ? That would one great Addition to yours great series that I 'm talking about now Heater in mine NodeRed installation successful integrated have . Thank you for your work!
   Many greetings
   Thilo

   Answer

   1. Chris Krzikalla

      March 8, 2023 at 6:19 p.m

      Hi,
      glad me if I you _ help could . I puzzle , experiment alone and write all this yourself . I don’t have a “team” . Occasionally Michael Hanna from Viessmann helps me when I have questions have .
      ~~Schnellwahlen kannst du dir selbst zusammenbauen, indem du z.B. für~~

~~“ich möchte einmalig Warmwasser” einen bzw. mehrere Nodes hernimmst: Der erste setzt die Zieltemperatur auf 60°C oder so, den zweiten lässt du alle 90 Sekunden (empfohlenes Intervall) die Wassertemperatur messen. Wenn Zieltemperatur erreicht, stellt der darauffolgende Node (aktiviert durch einen Switch Node) die Zieltemperatur auf den “Standby Wert” zurück. Alternativ kannst du das auch in zwei Function Nodes packen.~~
~~Im Prinzip macht die App bzw. die Viessmann Logik dahinter auch nichts Anderes. Aber, Danke für den Tipp. Ich werde bei nächster Gelegenheit was darüber schreiben.~~
What I have above written have is that is not quite wrong , but there is actually a parameter in the API which I am unique Hot water generate can :
`heating.dhw.oneTimeCharge`
In the Viessmann documentation says : *Shows whether one time charge of dhw is active. Also provides the commands to activate/deactivate it*
is included in the basic package contain . The instruction in addition finds itself now at the end of this updated one article .
Have fun still .
Chris

Answer

1. thilo

   March 20, 2023 at 10:26 p.m

   Hi Chris, I wanted to me again for the solution “I would like “Hot water ” thank you . I really liked that helped ! Works really great. I hope you do continue …
   Many greetings
   Thilo

   Answer

Write a comment

I am happy me about praise and criticism.
If you have problems with this one presented You have instructions and you don't further If you come :
Please describe the problem or error message (s) if possible Exactly describe , including at what point (e.g. in which node or command) and under which circumstances the error

occurs .
You 're welcome too write an email . The address can be found in the imprint .
I give myself a lot trouble , I think readers to help further . The more specific you are , the more easier and faster can I try to help .
Your e-mail address becomes not published .
" Required fields are with * marked

comment *

Surname *

Email *

website

My name, email address and website in this browser for the next Commentary save .

Kommentar abschicken

News

Viessman API / Node Red

- Current Viessmann API problem
- overview
- Access to the API
- Query and visualize data
- Settings change
- InfluxDB installation
    - InfluxDB : Problems during installation and update – solution
- InfluxDB and charts
- Node Red Dashboard Tips & Tricks
    - other tips & tricks
- Solar thermal energy : yield calculate

Networking

- Reverse SSH Tunnel – step by step
- Reverse SSH tunnel – administer remote devices
- Own IP address find out

- Dynamic IP address with on-board resources
- Hanging router automatically reboot
- SSH port address change
- WLAN in headless operation configure
- SSH at new Raspbian distributions disabled
- More stable Operation at weak WiFi

Internet of Things

- ShellyPlus actuators (2nd generation) with MQTT and Node-Red
- Geofencing experiments with the Locative App
  Part 1: PHP
- Geofencing experiments with the Locative App
  Part 2: Node-Red
- ESP8266 (Wemos D1 MINI) sensor with MQTT
- Update does Mosquito broken!
- Clear flash memory
- ESP8266: WiFi password etc. without Program save
- Dehumidify the basement with MQTT and Node Red
- Sell Sensor board for D1 Mini and DHT11/22 for 5 euros
- Build a WiFi- enabled sensor with ESP8266 & PHP
- Air pressure measure – but correct !
- More stable Operation at weak WiFi
- 433MHz switching sockets remote control
- IoT heating control 1 (LCD panel)
- IoT heating control 2 (Web Interface)
- Hanging router automatically reboot

python

- 433MHz switching sockets remote control
- Second update: Read and process weather report in JSON format
- Image Manipulation – with PIL image files change
- More stable Operation at weak WiFi
- Shebang! Or how to omit Python
- Simple Development environment
- Moon phase and illumination calculate

Raspbian

- Reverse SSH Tunnel – step by step
- InfluxDB : Problems during installation and update – solution
- Reverse SSH Tunnel – administer remote routers
- Raspbian file system automatically check and repair
- SD cards wear avoid

- I2C, SPI etc. works not after update more
- Raspberry Pi Camera Module – Red spot in the center of the image
- SSH port address change
- Add users to a group
- Own IP address find out
- WLAN in headless operation configure
- SSH at new Raspbian distributions disabled

Tips & Tricks

- InfluxDB : Problems during installation and update – solution
- Raspbian file system automatically check and repair
- More stable Operation at weak WiFi
- Clear flash memory
- GPIO 14 and 15 only with Caution enjoy
- SD cards wear avoid
- Air pressure measure – but correct !
- WLAN in headless operation configure
- Reverse SSH Tunnel – administer remote routers
- SSH at new Raspbian distributions disabled
- Simple Development environment
- The raspberry annoying ...
- Add users to a group
- Own IP address find out

Last Posts

- Current Viessmann API problem
- Solar – Pylontech battery (C series) via console wake
- ShellyPlus actuators (2nd generation) with MQTT and Node-Red
- Viessmann API and Node-Red – Part 1
- Viessmann API and Node-Red – Part 2
- Viessmann API and Node-Red – Part 3
- Viessmann API and Node Red – Part 4

Tags

- 433MHz

- Absolute humidity

- Arduino

- call
- Barometric Elevation formula
- BME280
- BMP085
- BMP180
- Brokers
- Connect clients itself not
- Darksky
- DHT22
- ESP8266
- Exception 28
- Flash Memory

- Influx version 1.8
- Dehumidifier
- Cellular
- Moon phases
- Mosquito 2.0.0
- MQTT
- MQTTfx
- Node Red
- Notepad++
- PHP
- Program
- python

- Raspberry Pi
- more relative Air pressure
- Reverse SSH
- routers
- Security
- sunrise
- sunset
- start
- tunnel
- UMTS
- Update

- Viessmann API
- Visualize
- Weather Underground
- Web service
- Weather report
- WiFi
- WIRELESS INTERNET ACCESS

Latest Posts

- Current Viessmann API problem
- Solar – Pylontech battery (C series) via console wake
- ShellyPlus actuators (2nd generation) with MQTT and Node-Red
- Viessmann API and Node-Red – Part 1
- Viessmann API and Node-Red – Part 2

Latest Comments

- Jürgen Graf on Viessmann API and Node Red – Part 4
- Chris Hofstaedter on Reverse SSH Tunnel – step by step
- Pierre on Reverse SSH Tunnel – step by step
- Chris Hofstaedter on Viessmann API and Node-Red – Part 2
- Jürgen Graf on Viessmann API and Node-Red – Part 2

Categories

- 433MHz

- Generally
- Arduino
- D1 Mini
- ESP8266/ESP32
- Firmware
- Geofencing
- GPIO
- InfluxDB
- IoT
- JSON
- camera
- mobile
- MQTT
- network
- Node Red
- php
- python
- Raspberry Pi
- Raspbian
- Shelly
- Solar
- Tips & Tricks
- Viessmann API

Search after: