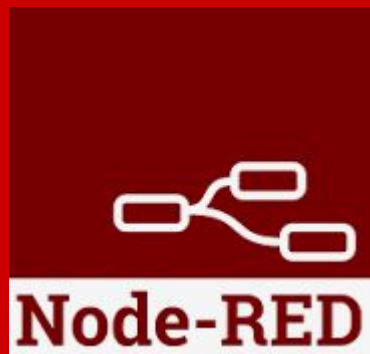

Node-RED & MQTT

For Dummies



De eerste stappen

Webinar Domoticz Facebook groep 2022 v1.1

Node-RED Simpele Flows en MQTT/Domoticz integratie

In dit gedeelte van de webinar behandelen we de eerste stappen in Node-Red en MQTT.

Een aantal simpele basis/template flows welke dan zelf verder uitgebreid/doorgebouwd kunnen worden.

Node-RED Simpele Flows en MQTT/Domoticz integratie

Om verder te kunnen gaan verwachten we dat je een werkende installatie hebt van Node-Red en MQTT .

MQTT beide hebt geconfigureerd in zowel Node-Red als in Domoticz.

(zie vorige presentatie).

MQTT in vogelvlucht

- MQTT is een lichtgewicht transport protocol met een publish en subscribe principe en verzorgt de boodschappen tussen de verschillende devices.
- Er bestaan 1 of meerdere Brokers en meerdere clients
- Clients communiceren alleen met een Broker, nooit onderling.
- Een client subscribed aan een Broker door middel van een Topic.
- Met het Topic filtert de Broker de boodschappen voor de desbetreffende client.

Voorbeeld topic:

myhome/beganegrond/huiskamer/temp

MQTT in vogelvlucht

- Een topic kan ook een WildCard bevatten zoals een + of een # b.v. myhome/beganegrond/huiskamer/#
- De standaard Domoticz Topics zijn :
domoticz/out en domoticz/in
- MQTT heeft allerlei mechanismen om te definiëren mocht er een device off-line gaan of on-line komen of de connectie wegvallen. Dit valt buiten het doel van deze presentatie.

Een duidelijke en gedetailleerde uitleg kun je hier vinden:

<https://www.hivemq.com/blog/mqtt-essentials-part-1-introducing-mqtt/>

Node-Red in vogelvlucht

- Node-Red is een flow-based visual programming tool
- De NR flow kan in een browser gemaakt worden.
- NR draait onderliggend op Node.js en maakt JavaScripts welke opgeslagen worden in een json file.
- NR kan op allerlei machines geïnstalleerd worden maar bestaat bijvoorbeeld ook in de Cloud.
- NR heeft een aantal standaard nodes beschikbaar maar heeft een enorme bibliotheek aan nodes met uiteenlopende toepassingen welke met een simpel commando geladen en gebruikt kunnen worden. Mocht dit niet toereikend zijn kan altijd direct in JavaScript verder gegaan worden.

Node-Red in vogelvlucht

- Node-Red draait als een service en op het moment dat een flow is opgeslagen, is deze altijd beschikbaar.
- NR is een flow-based, dat wil zeggen dat het nooit vanzelf iets gaat “ondernemen”
- Een startpunt kan zijn een handmatige of time trigger of een status verandering of waarde gemeld via MQTT bijvoorbeeld vanuit Domoticz.
- De output kan een berekening zijn of een message naar Domoticz maar ook een grafiek of een meter (gauge) naar het dashboard.
- Flows kunnen eenvoudig geïmporteerd of geëxporteerd en dus gedeeld worden.

Voor meer info ga naar <http://www.nodered.org>

Node-Red in vogelvlucht

Als de Node-RED als een service draaid :

- **node-red-start** - Dit start de Node-RED service en laat de logoutput zien.
Door Ctrl-C of het window te sluiten zal de service niet stoppen en in de achtergrond door blijven lopen.
- **node-red-stop** - dit stopt de Node-RED service
- **node-red-restart** - dit herstart de Node-RED service (dus stop en start)
- **node-red-log** - dit laat de log output van de Node-Red service zien

Note:

Mocht het gebeuren dat je een palette geïnstalleerd hebt maar deze niet wil werken en alle flows laat vastlopen, Delete dan zo veel mogelijk referenties naar deze nieuwe palette en doe een node-red-restart. 9 van de 10 keer gaat Node-Red weer werken. Een nieuwe installatie is meestal niet nodig.

Door een nieuwe installatie blijven de flows “gewoon” behouden en lost dat ook niet op. Alleen door dan de flows ook te verwijderen krijg je een echte clean install.

Open de Node-red poort niet naar het internet....Node-Red is niet erg goed met security.

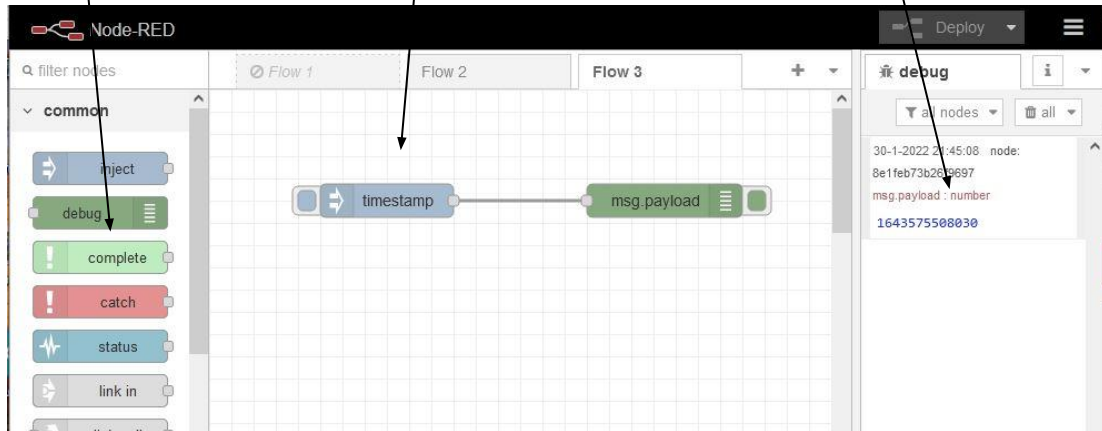
Node-Red user interface

Node Palette

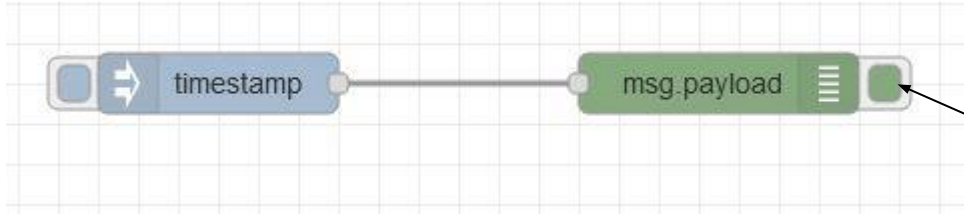
Flow Workspace

Sidebar

Help/debug/config window



Node-Red de eerste Flow



Sleep de Inject node en de Debug node in de flow. Debug on/off

Verbind beide zoals hierboven en klik op Deploy.

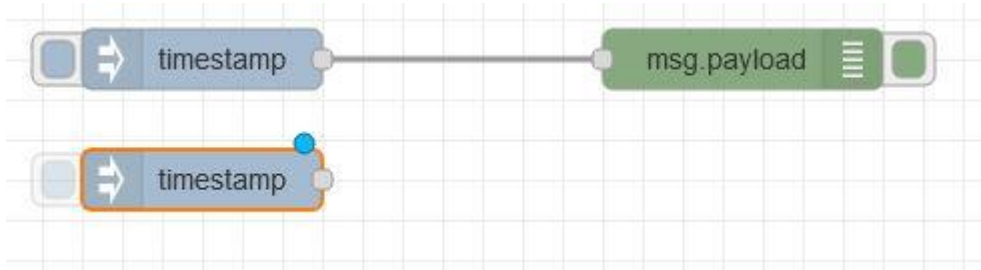
Met het groene vakje rechts van de debug kun je deze snel aan of uit zetten.

Als je nu op het blauwe vakje klikt zal de flow geactiveerd worden en zie je in het Debug window de tijd in e-poch notatie verschijnen.

Het debug window wordt zichtbaar als je op de kleine “Bug” klikt. Met de vuilnisbak vlak daaronder kiep je de Output tekst in de prullenbak.



Node-Red de eerste Flows

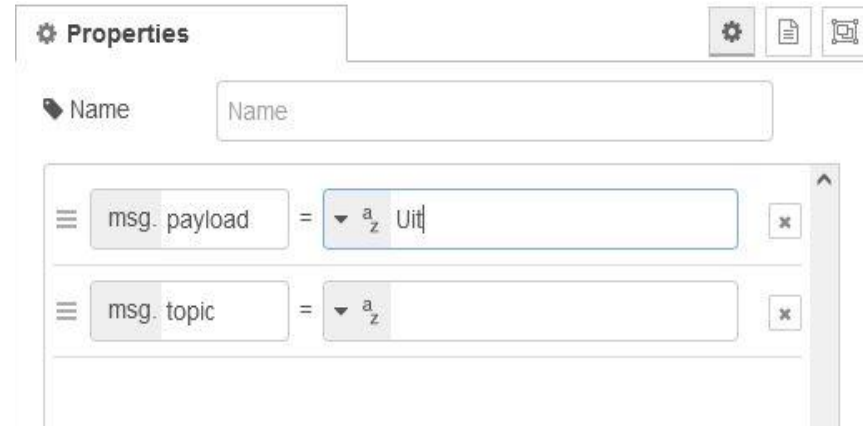


Sleep een tweede Inject node in de Flow en dubbelklik op het icoon.

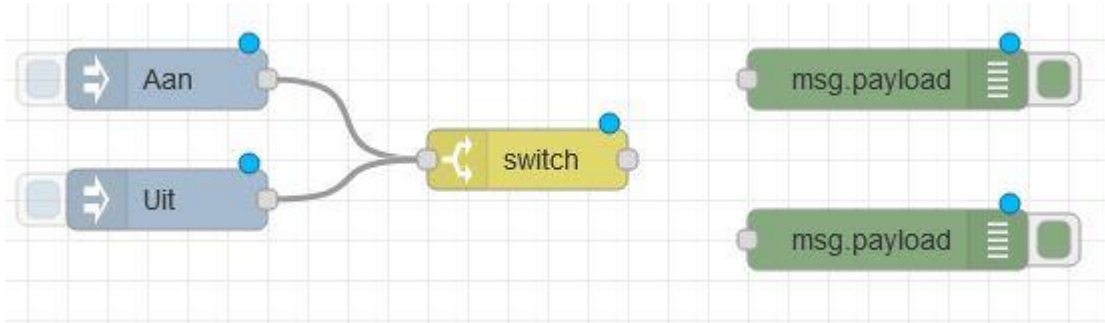
Klik op de dropdown bij Timestamp en selecteer String.

Zet in het vak bij a-z het woordje “Uit”

Doe hetzelfde met de andere timestamp
Maar plaats nu het woordje “Aan”



Node-Red de eerste Flows



Sleep de Switch node op de flow en verbind als hierboven.

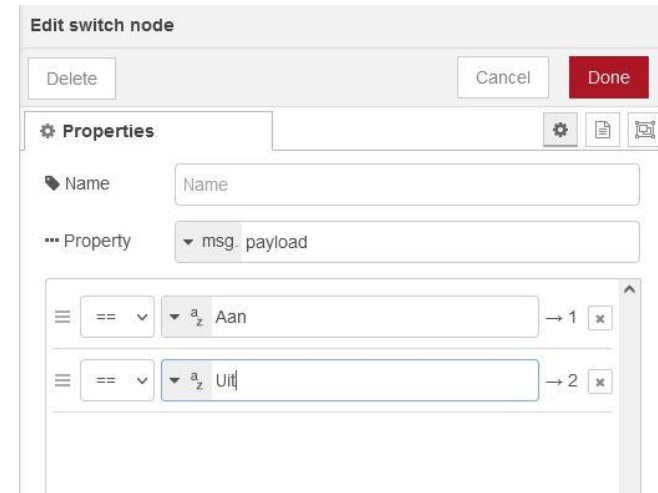
Dubbel klik vervolgens op de Switch node.

Vul nu de woordjes “Aan” en “Uit” in. Let op

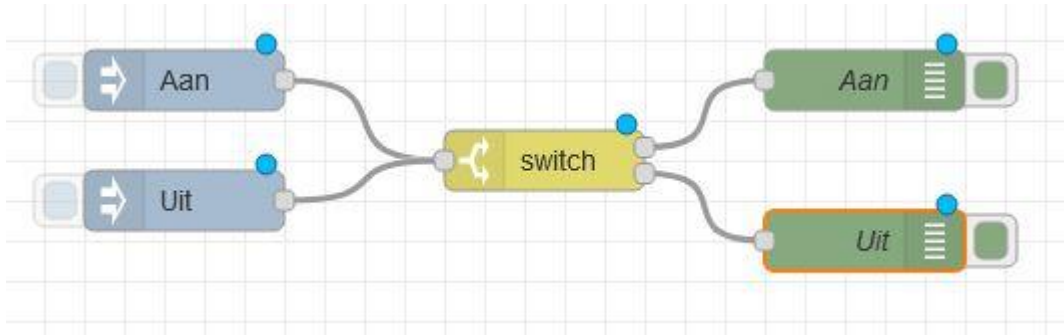
Hoofdletter gevoelig!

Het tweede vak wordt gekregen door links onder op Add te klikken.

Klik Done.



Node-Red de eerste Flows



Verbind de Debug nodes als in het plaatje.
Dubbel klik de debug node en vul bij Naam
Het woordje “Aan” in en in de tweede node het
Woordje “Uit”.

Klik op Done en klik op Deploy om de flow
Actief te maken.

Edit debug node

Delete Cancel Done

⚙ Properties

Output msg. payload

To ☒ debug window

☐ system console

☐ node status (32 characters)

Name Aan

Node-Red de eerste Flows

Klik op de blauwe trigger node “Aan”. Rechts in de debug verschijnt nu de output van de Switch en het woordje “Aan”. Hetzelfde natuurlijk voor “Uit”

Debug node naam

Debug node Output

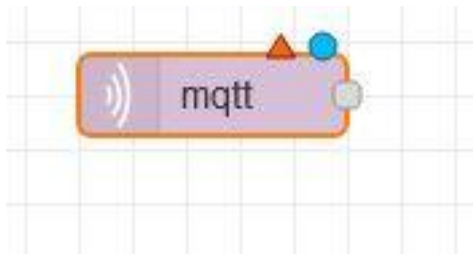
Kopieer hier eventueel de Flow :

<https://github.com/freijn/SimpleNode-RedFlows/blob/main/AanUitflow.json>



Node-RED MQTT client configuratie

Sleep de MQTT-in node op een nieuwe flow, dubbel click de node en achter Server op het potlood.

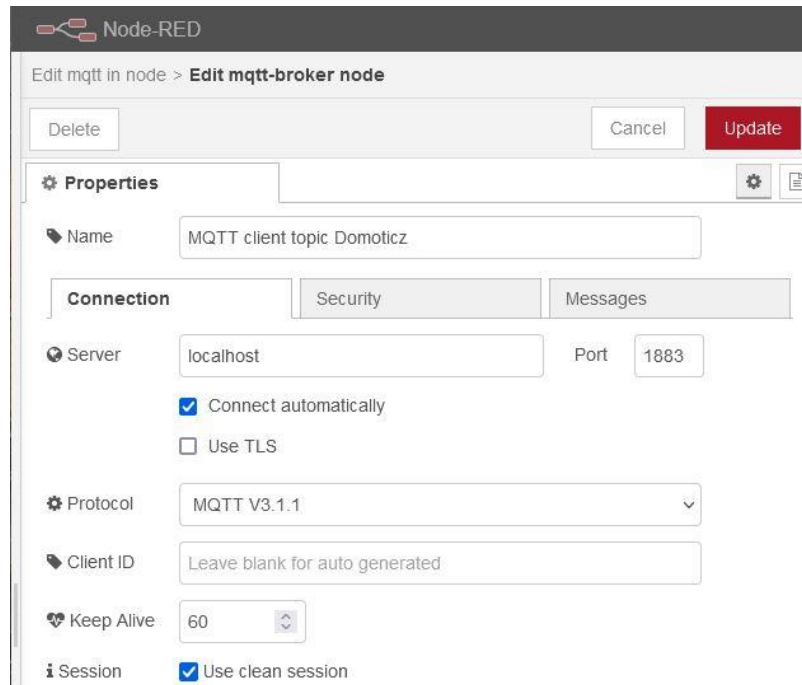


Naam: MQTT client topic Domoticz

Server: localhost

Port: 1883

Klik op add/Update

A screenshot of the Node-RED MQTT client configuration panel. The panel is titled 'Edit mqtt in node > Edit mqtt-broker node'. It has a 'Delete' button, a 'Cancel' button, and an 'Update' button. The 'Properties' tab is active, showing the 'Name' field set to 'MQTT client topic Domoticz'. The 'Connection' tab is also visible, showing the 'Server' field set to 'localhost', the 'Port' field set to '1883', and the 'Connect automatically' checkbox checked. The 'Protocol' dropdown is set to 'MQTT V3.1.1'. The 'Client ID' field is set to 'Leave blank for auto generated'. The 'Keep Alive' field is set to '60'. The 'Session' section shows the 'Use clean session' checkbox checked.

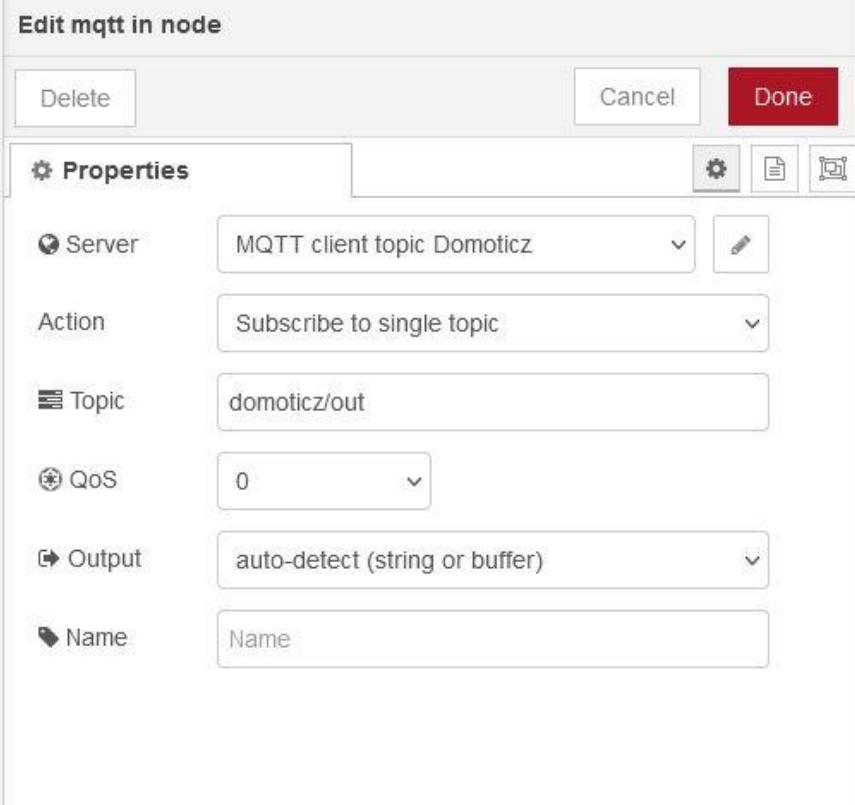
Node-RED MQTT client configuratie

Vult het Topic in:

Topic : domoticz/out

QoS: 0

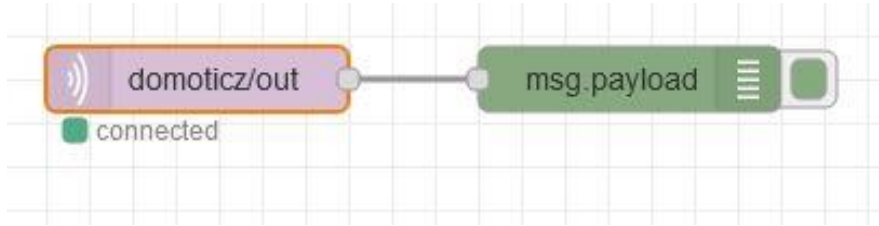
Klik 'Done'



The screenshot shows the 'Edit mqtt in node' configuration window in Node-RED. At the top, there are three buttons: 'Delete', 'Cancel', and 'Done'. Below these is a 'Properties' tab with a settings icon. The configuration fields are as follows:

Property	Value
Server	MQTT client topic Domoticz
Action	Subscribe to single topic
Topic	domoticz/out
QoS	0
Output	auto-detect (string or buffer)
Name	Name

Node-Red ↔ MQTT



Verbind de debug node als in het plaatje en klik op 'deploy'.
Als alles goed is gegaan komt er onder de node "connected" te staan.

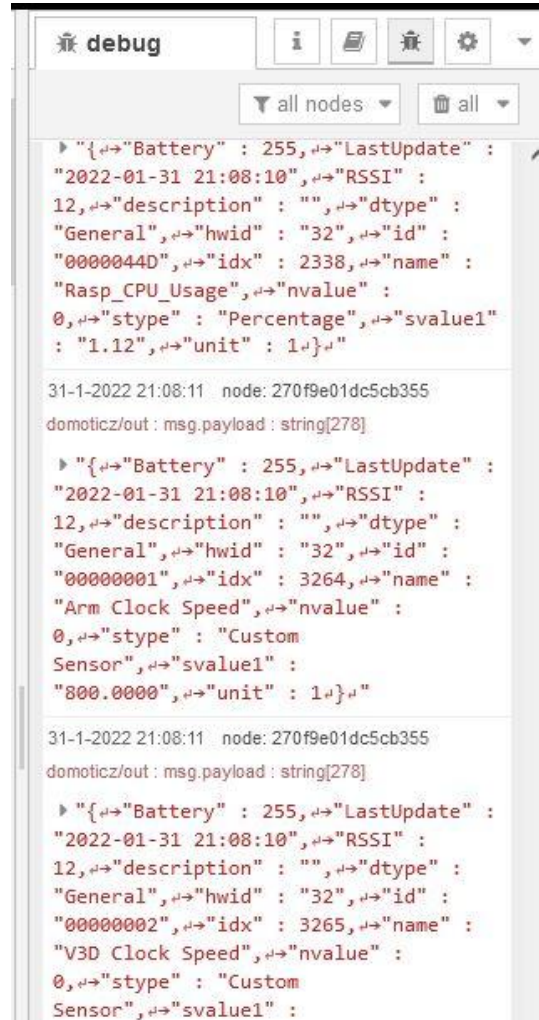
In het debug window verschijnen nu de boodschappen die door Domoticz verzonden worden met het topic "domoticz/out".

Node-Red ↔ MQTT

Debug output window

De text die je ziet is een aaneen geregen string gescheiden door komma's.

Nu zou je met string functies de verschillende elementen kunnen scheiden maar je kunt de string ook door de Json node heen halen waardoor alle elementen in een object gezet worden.



```
debug
all nodes
all

▶ {"Battery": 255, "LastUpdate": "2022-01-31 21:08:10", "RSSI": 12, "description": "", "dtype": "General", "hwid": "32", "id": "0000044D", "idx": 2338, "name": "Rasp_CPU_Usage", "nvalue": 0, "stype": "Percentage", "svalue1": "1.12", "unit": 1}

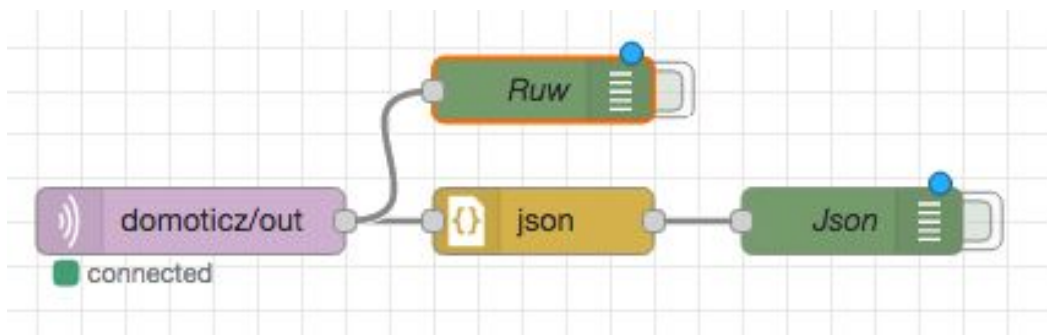
31-1-2022 21:08:11 node: 270f9e01dc5cb355
domoticz/out : msg.payload : string[278]

▶ {"Battery": 255, "LastUpdate": "2022-01-31 21:08:10", "RSSI": 12, "description": "", "dtype": "General", "hwid": "32", "id": "00000001", "idx": 3264, "name": "Arm Clock Speed", "nvalue": 0, "stype": "Custom Sensor", "svalue1": "800.0000", "unit": 1}

31-1-2022 21:08:11 node: 270f9e01dc5cb355
domoticz/out : msg.payload : string[278]

▶ {"Battery": 255, "LastUpdate": "2022-01-31 21:08:10", "RSSI": 12, "description": "", "dtype": "General", "hwid": "32", "id": "00000002", "idx": 3265, "name": "V3D Clock Speed", "nvalue": 0, "stype": "Custom Sensor", "svalue1": }
```

Node-Red ↔ MQTT



```
1-2-2022 16:09:30 node: Ruw
domoticz/out : msg.payload : string[237]

> {"Battery" : 255, "RSSI" : 12, "description" : "", "dtype" : "General", "hwid" : "18", "id" : "84028", "idx" : 2028, "name" : "WaterMeter", "nvalue" : 0, "stype" : "Counter Incremental", "svalue1" : "962959.0", "unit" : 1}
```

1-2-2022 16:09:30 node: Json

Ruwe text

Output na Json

```
1-2-2022 16:09:30 node: Json
domoticz/out : msg.payload : Object

> { Battery: 255, RSSI: 12,
  description: "", dtype: "General",
  hwid: "18" ... }
```

Node-Red ↔ MQTT

```
▼object
  Battery: 255
  RSSI: 12
  description: ""
  dtype: "General"
  hwid: "18"
  id: "84028"
  idx: 2028
  name: "WaterMeter"
  nvalue: 0
  stype: "Counter Incremental"
  svalue1: "962959.0"
  unit: 1
```

De
uitgeklapte
Json Text

domoticz/out : msg.payload : Object

```
▼object
  Battery: 255
  RSSI: 12
  description: ""
  dtype: "General"
  hwid: "18"
  id: "84028"
  idx: 2028
  name: "WaterMeter"
  nvalue: 0
  stype: "Counter Incremental"
  svalue1: "962959.0"
  unit: 1
```

Copy path

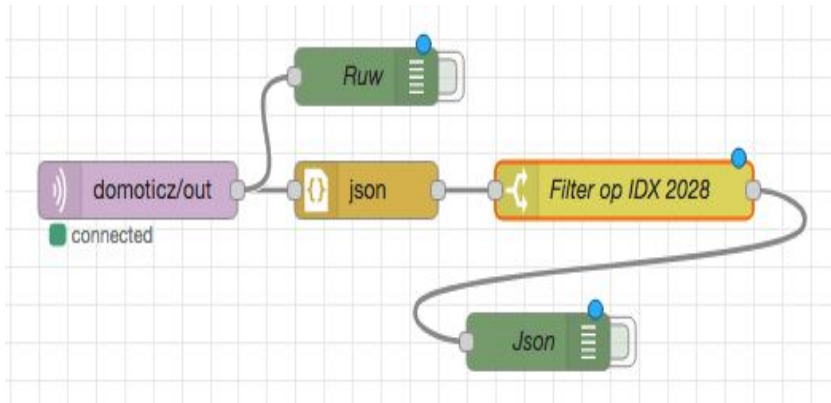
1-2-2022 16:09:32 node: Ruw

domoticz/out : msg.payload : string[217]

Door op dit
symbool te klikken
kan het path naar
de Json
gekopieerd
worden.

Node-Red ↔ MQTT

Dit path kan dan gebruikt worden om te filteren in bijvoorbeeld de switch node.
Hiermee kunnen we filteren op alleen de boodschappen van sensor met IDX= 2028



Edit switch node

Delete Cancel Done

Properties

Name: Name

Property: msg.payload.idx

Filter: `== 2028` → 1

checking all rules

☐ recreate message sequences

Enabled

debug

all nodes

```
: "General", "hwid" : "18", "id" : "84028", "idx" : 2028, "name" : "WaterMeter", "nvalue" : 0, "stype" : "Counter Incremental", "svalue1" : "962959.0", "unit" : 1}
```

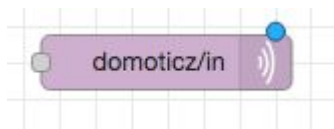
1-2-2022 16:09:30 node: Json
domoticz/out : msg.payload : Object

object

Battery: 255
RSSI: 12
description: ""
dtype: "General"
hwid: "18"
id: "84028"
idx: 2028
name: "WaterMeter"
nvalue: 0
stype: "Counter Incremental"
svalue1: "962959.0"
unit: 1

1-2-2022 16:09:32 node: Ruw
domoticz/out : msg.payload : string[217]

Node-Red ↔ MQTT



De reactie op een waarde in de flow kan een schakelaar moeten bedienen.
Met de Node MQTT-out waar we als topic "domoticz/in" in zetten kunnen we MQTT boodschappen weer terug sturen naar Domoticz. Domoticz subscribed (leest) alleen topics als "domoticz/in" alle andere komen niet aan in Domoticz.

Edit mqtt out node

Delete

Cancel

Done

⚙ Properties

⚙

📄

🔖

🌐 Server

127.0.0.1:1883

▼

✎

📄 Topic

domoticz/in

⚙ QoS

▼

🔄 Retain

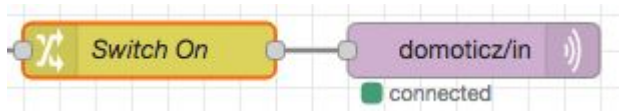
▼

🔖 Name

Name

Tip: Leave topic, qos or retain blank if you want to set them via msg properties.

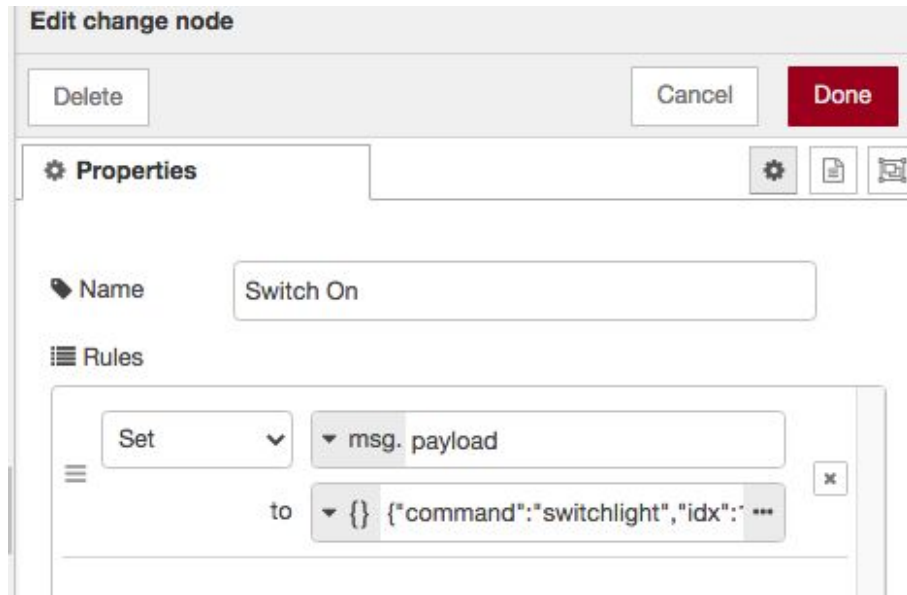
Node-Red ↔ MQTT



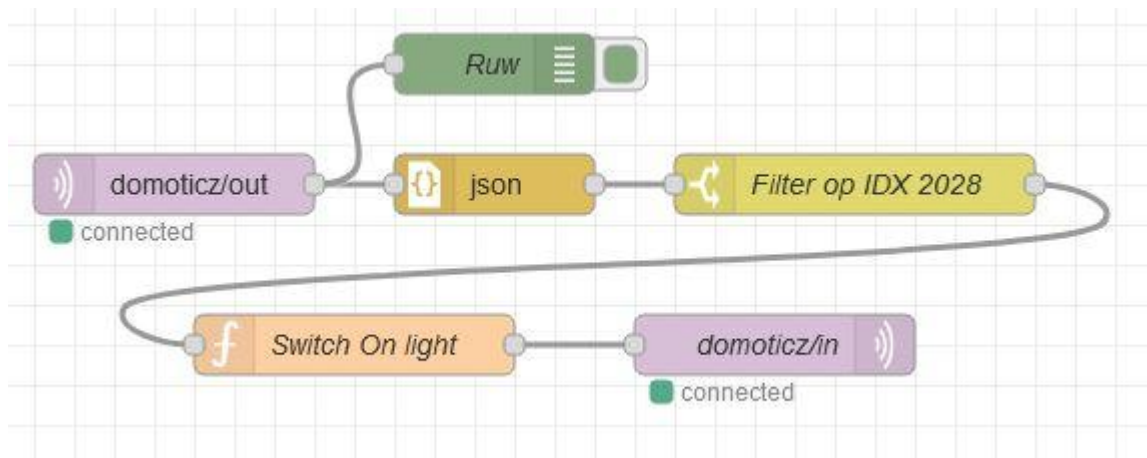
```
{"command": "switchlight",  
  "idx": 118,  
  "switchcmd": "On"}
```

Commando's worden in een strict format verwacht door Domoticz. Hierboven de payload inhoud om een switch met idx=118 aan te zetten. Deze kun je in een change node plaatsen en versturen via de MQTT-out naar Domoticz.

Meer commando's en formats vind je hier:
<https://www.domoticz.com/wiki/MQTT>



Node-Red ↔ MQTT



Een flow met een filter op de IDX en een actie richting Domoticz komt er dan zo uit te zien.

Kopieer hier de Switch flow:

<https://github.com/freijn/SimpleNode-RedFlows/blob/main/SwitchAanopMQTT.json>

Node-Red msg.payload

Een Node-Red flow werkt door middel van boodschappen door te geven tussen de verschillende nodes.

De boodschappen zijn Javascript objecten en kunnen allerlei attributen krijgen.

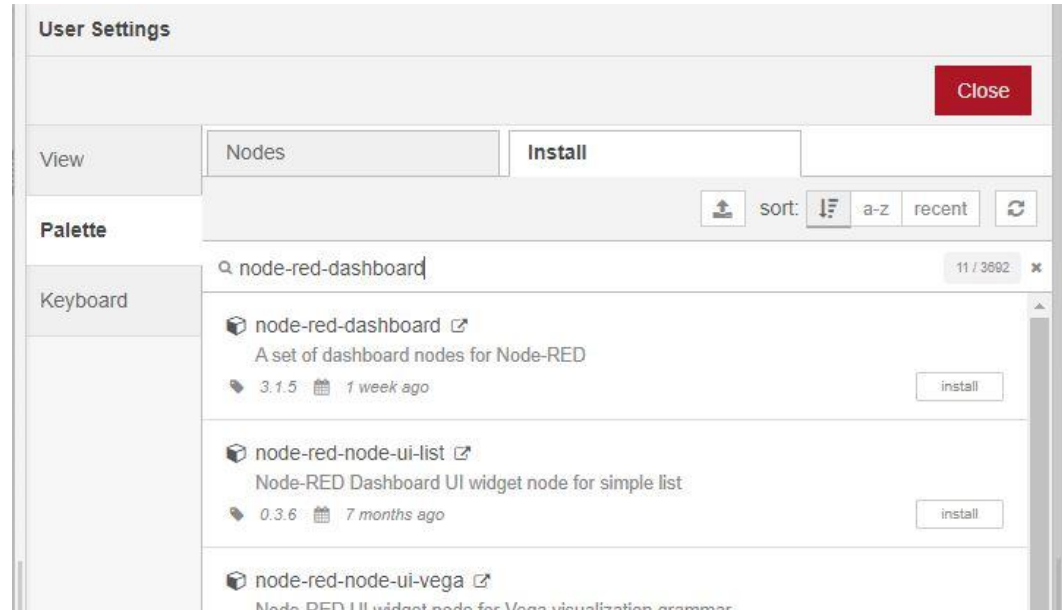
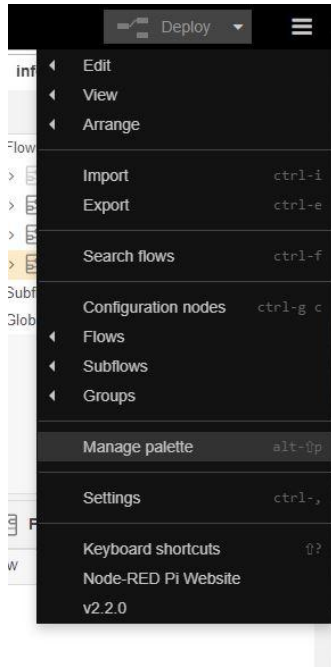
De standaard boodschap object in Node-Red is `msg.payload`. Deze kan de inhoud van bijvoorbeeld een binnengekomen MQTT boodschap bevatten. Zoals gezien in de vorige slides kun je waarden hier uit filteren. Als je daarna met een change node of een function node de waarde van `msg.payload` overschrijft kunnen de volgende nodes de inhoud dus niet meer uitlezen omdat de originele waarde overschreven is.

Als je dus iets wilt aanpassen in een bestaande `msg.payload` moet je deze payload uitlezen, aanpassen en weer terug schrijven.

Je kunt ook (of beter) een eigen attribuut aan de payload 'hangen' en daar de waarde in schrijven. `msg.payload.temperatuur` of `msg.payload.elektra` is zijn dus mogelijkheden.

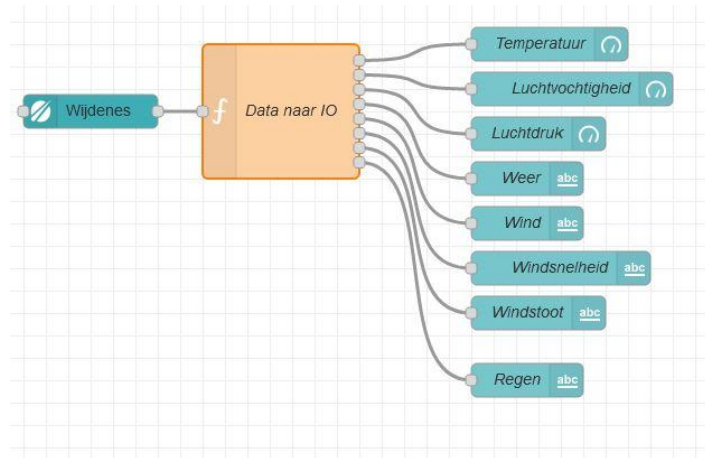
Een Array kun je bijvoorbeeld op deze manier uitlezen `msg.payload.phone[1].type`

Node-Red Palette install



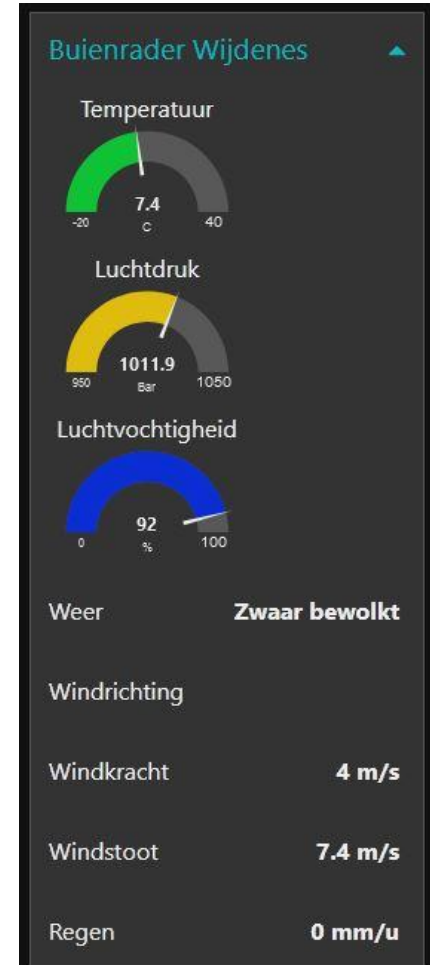
Open rechts “Manage palette” en selecteer de tab “Install” In het zoekvenster typ je “node-red-dashboard” en klik op install. Installeer ook de palette “node-red-contrib-buienradar”

Node-Red UI interface



We gaan nu bovenstaande flow importeren welke, als we die een deploy geven, een grafische voorstelling van de buienradar node zal maken . Uiteraard kun je zelf kiezen welk weerstation het meest in de buurt is van jou locatie. Dit doe je in de meest linkse node.

De Node-Red UI interface vind je op :
<http://192.168.1.152:1880/ui>



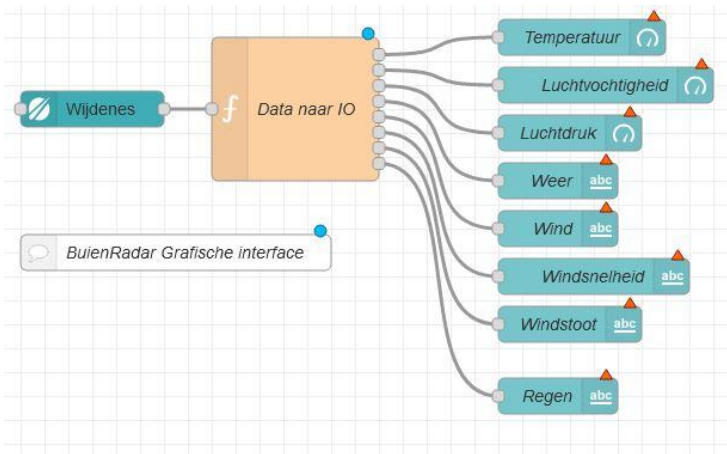
Node-Red UI interface

Kopieer hier de Buienradar flow:

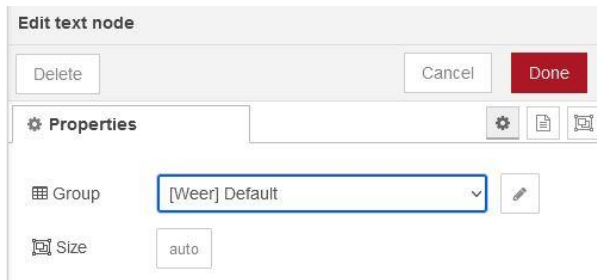
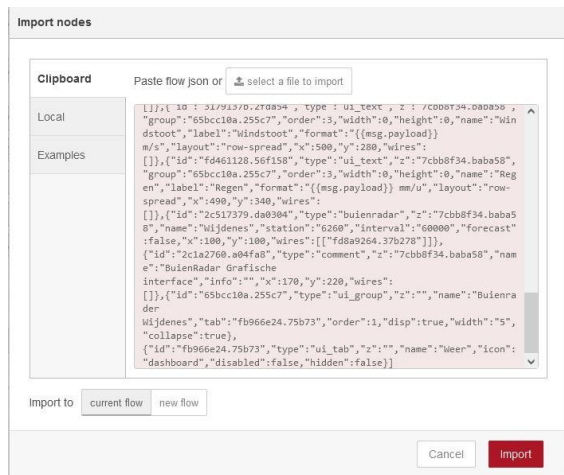
<https://github.com/freijn/SimpleNode-RedFlows/blob/main/buienradarflow.json>

Klik op de “raw” button om de inhoud te copieren.

Ga nu naar Node-Red en klik rechtsboven op de hamburger (3 streepjes). Selecteer “import” Plak de inhoud van het klembord in het venster. Klik op import.



Als alles goed gegaan is heb je nu deze flow geïmporteerd. Klik 1 voor 1 op de icoontjes met een rode driehoek en selecteer de [Het weer] groep. Klik op done en deploy. Voila het weer in de UI interface.



Node-Red Extra Flows

Extra Flows te gebruiken als template of probleem oplossing :

Covid19 Tellers:

<https://github.com/freijn/SimpleNode-RedFlows/blob/main/Covid19flow.json>

DigiClock:

<https://github.com/freijn/SimpleNode-RedFlows/blob/main/DigiClockflow.json>

GoogleHomeMini:

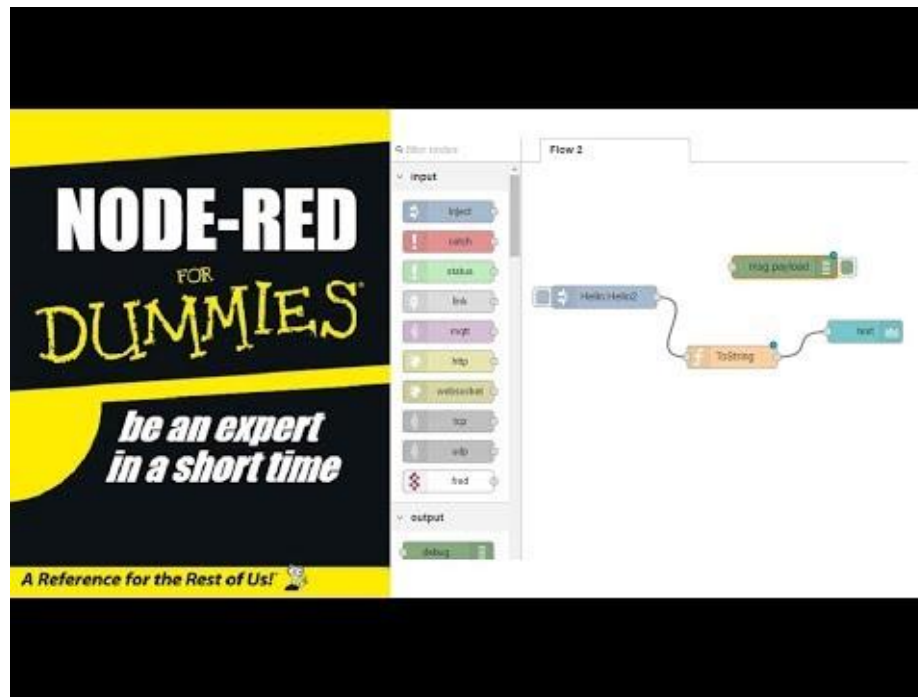
<https://github.com/freijn/SimpleNode-RedFlows/blob/main/GoogleHomeMiniNotifierflow.json>

Sensor to UI :

<https://github.com/freijn/SimpleNode-RedFlows/blob/main/SensorToUIflow.json>

OctoPrint with Telegram messages

<https://github.com/freijn/SimpleNode-RedFlows/blob/main/OctoPTelegram.json>



Veel programmer plezier met Node-Red en Domoticz