

How to connect Volvo on Call to Domoticz

Table of Contents

1 Introduction.....	2
2 Volvo on Call configuration of Domoticz.....	2
3 Installation of the Volvo on Call script.....	4
4 Configuration of the Volvo on Call script.....	4
5 Node-Red flow VoC to Domoticz.....	6
6 Node-Red flow Domoticz to VoC.....	25
7 Temperature and Location.....	30
8 Monitoring of VoC script and MQTT server.....	37
9 Known issues.....	41
10 Credits.....	41

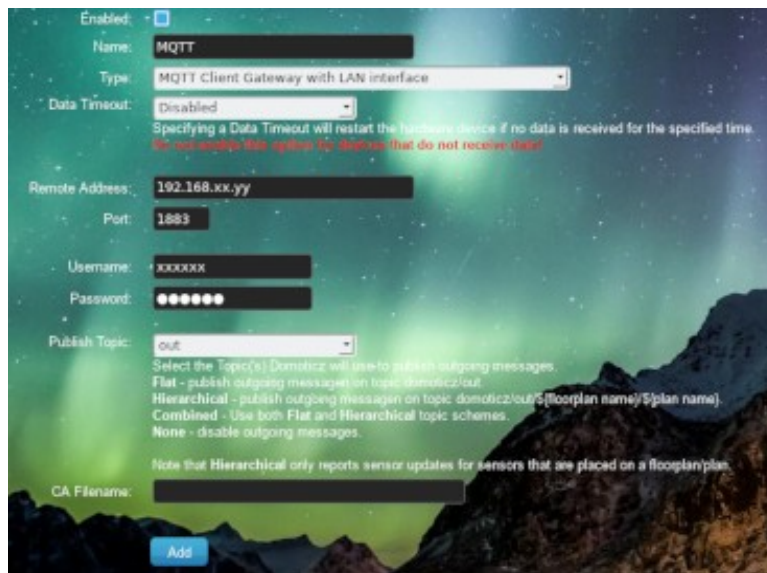
1 Introduction

It is assumed that you have installed the following programs and that they are functioning correctly:

- MQTT broker. It is preferred to use Mosquitto.
To download it, see: <https://mosquitto.org/>
- Node Red.
To download it, see: <https://nodered.org/>

2 Volvo on Call configuration of Domoticz

In Domoticz install in “Hardware” MQTT Client Gateway with LAN interface

The image shows a web-based configuration form for an MQTT Client Gateway in Domoticz. The form is set against a background of a night sky with aurora borealis. Fields include: 'Enabled' (checked), 'Name' (MQTT), 'Type' (MQTT Client Gateway with LAN interface), 'Data Timeout' (Disabled), 'Remote Address' (192.168.xx.yy), 'Port' (1883), 'Username' (xxxxxx), 'Password' (masked with dots), 'Publish Topic' (out), and 'CA Filename'. A blue 'Add' button is at the bottom. A note explains the 'Publish Topic' options: Flat, Hierarchical, Combined, and None.

Configure the IP address of the MQTT broker (server). If Domoticz and the MQTT server are installed on the same hardware you can fill in “localhost” (without “ ”). It is recommended to leave the port as default (1883). If you have secured your MQTT broker with a user name and a password (recommended) fill in these in the corresponding fields. Leave “Publish Type” as “out”.

The Volvo on Call script will publish various sensors, both binary and analogue. Beside that, we want to send some commands to our Volvo. Therefore we will need one or more switches.

In order to create these sensors and switches, create in “Hardware” a Dummy (Does nothing, use for virtual switches only) and add the virtual sensors.

Each user has to check, which sensors and commands are supported for his/her Volvo.

To do so, go to the command line and issue the following command: *“voc print”*.

The result will be that all all available sensors and functions are listed.

We have to create several virtual sensors. For the binary sensors, which will present 2 values (ON/OFF, Open/Close, Safe/Unsafe) we have the option to create an “Alert” sensor or a simple

“Text” sensor. The analogue sensors, except one, are “Custom” sensors, with the correct index on the y- axis. The most important difference between the “Alert” sensor and the “Text” sensor is that the “Alert” sensor is able to send notifications and that it makes use of different colours, to attract attention. The user can decide for themselves which function requires an “Alert” or a “Text” sensor.

In this document we will use the “Alert” sensor for the windows and the doors. The other binary sensors are “Text” sensors.

So we have to create the following virtual sensors:

- 4 “Alert” sensors for the front left window, the rear left window, the front right window and the rear right window.
- 6 “Alert” sensors for the front left door, the rear left door, the front right door, the rear right door, the hood and the tailgate.
- 4 “Text” sensors for the front left tyre pressure, the rear left tyre pressure, the front right tyre pressure and the rear right tyre pressure.
- 8 “Text” sensors for the following functions: brake fluid, washer fluid, service warning, bulb failures, latest trip, car locked, heater(if applicable) and engine running.

The next sensors to configure are the other (analogue) sensors, as follows:

- Custom sensor for “Fuel Amount”, with axis label L
- Percentage sensor for “Fuel Amount Level”
- Custom sensor for “Fuel Consumption”, with axis label L/100 km
- Custom sensor for “Average Speed”, with axis label km/h
- Custom sensor for “Distance to Empty”, with axis label km
- Custom sensor for “Odometer”, with axis label km

If your car provides more than these analogue sensors, create the corresponding sensors accordingly.

Go to “Setup” and then to “Devices”. Make a note of the Idx number of the newly created devices, as we will need these later.

3 Installation of the Volvo on Call script

As the Volvo on Call script requires Python 3.6 or higher, we may have to update python.
Follow the guideline:

<https://www.scivision.dev/compile-install-python-beta-raspberry-pi/>

OR

<https://gist.github.com/SeppPenner/6a5a30ebc8f79936fa136c524417761d>

Verify always: `python -V`

Install the Volvo on Call python script from molobrakos in any folder you like, preferably your home directory (/home/pi). It's recommended that this folder is added to your path.

Go to your "home" directory and issue the command:

"git clone <https://github.com/molobrakos/volvooncall>"

The Volvo on Call script will be installed in your "home" directory (/home/pi).

4 Configuration of the Volvo on Call script

Insert your credentials, like username (email address) and password into: .voc.conf.

The configuration file is a hidden file in your "home" directory, so mind the dot.

The credentials are equal to the one you use in your app.

Configuration file in \$HOME/.voc.conf:

username: <username>

password: <password>

Mind the <space>, between the semicolon and the user name (email address) and password.

Now it is time to test.

Give the following command:

```
pi@rpi3plus:~ $ voc --help
```

You will get:

Retrieve information from VOC

Usage:

`voc (-h | --help)`

`voc --version`

`voc [-v|-vv] [options] list`

`voc [-v|-vv] [options] status`

```
voc [-v|-vv] [options] trips [--pretty|--json|--csv]
voc [-v|-vv] [options] owntracks
voc [-v|-vv] [options] print [<attribute>]
voc [-v|-vv] [options] (lock | unlock)
voc [-v|-vv] [options] heater (start | stop)
voc [-v|-vv] [options] engine (start | stop)
voc [-v|-vv] [options] honk_and_blink
voc [-v|-vv] [options] call <method>
voc [-v|-vv] [options] mqtt
voc [-v|-vv] [options] dashboard
```

Options:

```
-u <username> VOC username
-p <password> VOC password
-r <region> VOC region (na, cn, etc.)
-s <url> VOC service URL
-i <vin> Vehicle VIN or registration number
-g Geolocate position
--owntracks_key=<key> Owntracks encryption password
-l <interval> Polling interval (seconds) [default: 300]
-h --help Show this message
--immutable Read only mode
-v,-vv Increase verbosity
-d More debugging
--scandinavian_miles Report using Scandinavian miles instead of km ISO unit
--utc Print timestamps in UTC (+00:00) instead of local time
--version Show version
```

So if this works your fine.

Test further with *voc dashboard* and you will find a lot of information of your car.

To see everything that might be supported for your car issue the command *voc print*.

If the VoC script and Mosquitto run on different Rpi's and you want to publish all the topics with *voc mqtt* to that Mosquitto MQTT server, you have to add the following line to *.voc.conf*:

```
mqtt_url: mqtt://<username of mqtt broker>:<password of mqtt broker>@192.168.xxx.yyy:1883
```

(if you use the default port and without<>)

Mind the <space>, between the semicolon and *mqtt://....*

The username of the mqtt broker and the password of mqtt broker is only needed, if you have secured your MQTT broker with a user name and password, which is recommended.

Check with your client, that *voc mqtt* publishes all sensors under the following topic:

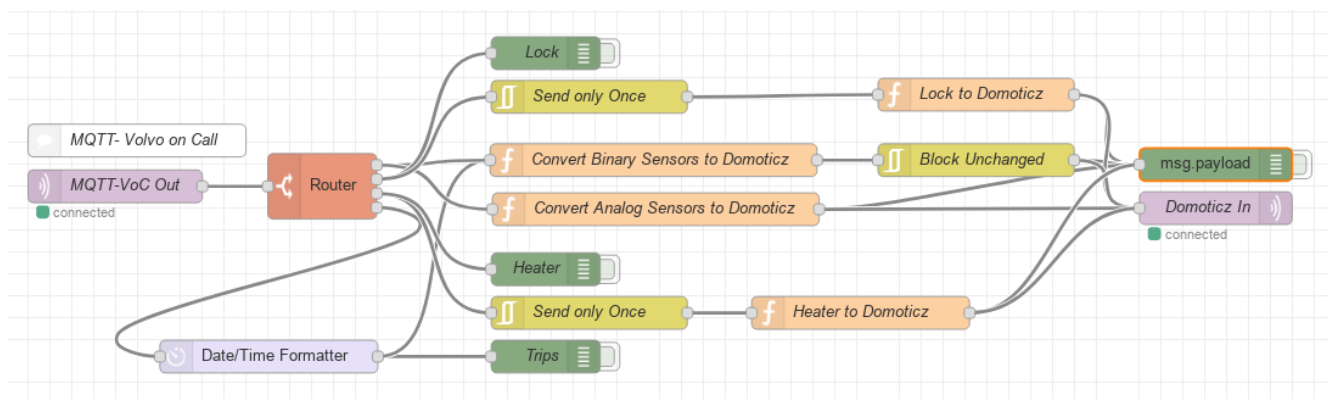
```
volvo/abc123/#
```

(abc123 is your license plate).

If you see your Volvo sensors published in your client, you are ready to implement the flows in Node Red.

5 Node-Red flow VoC to Domoticz

In the picture below, you will see the total flow.



All nodes used, are core Node-Red nodes, except the node, called “Router” and the node, called “Date/Time Formatter”.

The node “Router” has to be installed from, either the “Palette” of Node Red or directly from the command line. In “Palette” search for a node called: “node-red-contrib-routing”, and install this node. For more information, see: <https://flows.nodered.org/node/node-red-contrib-routing>.

The node “Date/Time Formatter” has to be installed as well.

Search for: “node-red-contrib-moment” and install this node.

For more information, see: <https://flows.nodered.org/node/node-red-contrib-moment>

The first node in the flow is a MQTT input node.

The configuration of the node is as follows:

Edit mqtt in node

Delete Cancel Done

Properties

Server RPI1_ MQTT_Broker

Topic volvo/abc123/+/+/state

QoS 0

Output auto-detect (string or buffer)

Name MQTT-VoC Out

The name of the Server is up to the user and can be anything, but will be used later, as well.

The VoC python script publishes many topics to the configured MQTT server. Many are intended to be used with Home Assistant, in order to create the various devices. These topics will not be used in Node Red and Domoticz. We are only interested in the various statuses and therefore the Topic will be: "volvo/abc123/+ /+ /state".

Note 1: Do not use a leading forward slash, such as "/volvo".

Note 2: Replace "abc123" with your license plate number (on all occurrences in the flow).

The Quality of Service (QoS) is up to the user, but can be left default.

The Output, you can leave default as .well.

For the name of the nodes, it is recommended to use a name, that represents the function.

Next we have to configure the MQTT broker. To do so, press the pencil, right of the Server name. We start with the "Connection" tab.

First configure the IP address of the Server. If Node-Red and the MQTT server are installed on the same hardware you may use "localhost". All other properties you can leave as indicated. Finally you have to press: "Update".

The screenshot shows the 'Edit mqtt in node > Edit mqtt-broker node' dialog. At the top are 'Delete', 'Cancel', and 'Update' buttons. Below is a 'Properties' section with a gear icon and a dropdown menu. The 'Name' field is 'RPI1_MQTT_Broker'. There are three tabs: 'Connection' (selected), 'Security', and 'Messages'. In the 'Connection' tab, the 'Server' field is '192.168.100.50' and the 'Port' field is '1883'. There is a checkbox for 'Enable secure (SSL/TLS) connection' which is unchecked. The 'Client ID' field is 'Leave blank for auto generated'. There is a 'Keep alive time (s)' field with '60' and a 'Use clean session' checkbox which is checked. At the bottom, there is a 'Use legacy MQTT 3.1 support' checkbox which is checked.

Edit mqtt in node > Edit mqtt-broker node

Delete Cancel Update

⚙ Properties

🔑 Name RPI1_MQTT_Broker

Connection Security Messages

🌐 Server 192.168.100.50 Port 1883

☐ Enable secure (SSL/TLS) connection

🔑 Client ID Leave blank for auto generated

⌚ Keep alive time (s) 60 ☒ Use clean session

☒ Use legacy MQTT 3.1 support

The next tab is the Security tab.

The screenshot shows the 'Edit mqtt in node > Edit mqtt-broker node' window. At the top, there are three buttons: 'Delete', 'Cancel', and 'Update'. Below these is a 'Properties' section with a gear icon and a dropdown menu. The 'Name' field is set to 'RPI1_MQTT_Broker'. Below the name field are three tabs: 'Connection', 'Security', and 'Messages'. The 'Security' tab is selected. Under the 'Security' tab, there are two fields: 'Username' with the value 'username' and 'Password' with a masked password '.....'.

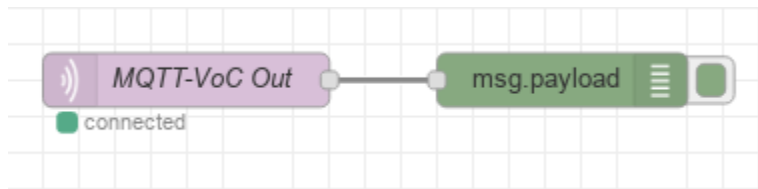
If you have secured your MQTT broker with, at least, with a user name and a password (recommended) you should fill in your chosen Username and Password.

Press "Update" again.

The Messages tab you can leave as default.

The screenshot shows the 'Edit mqtt in node > Edit mqtt-broker node' window. At the top, there are three buttons: 'Delete', 'Cancel', and 'Update'. Below these is a 'Properties' section with a gear icon and a dropdown menu. The 'Name' field is set to 'RPI1_MQTT_Broker'. Below the name field are three tabs: 'Connection', 'Security', and 'Messages'. The 'Messages' tab is selected. Under the 'Messages' tab, there is a section titled 'Message sent on connection (birth message)' with a dropdown arrow. Below this section are four fields: 'Topic' with the value 'Leave blank to disable birth message', 'Retain' with the value 'false', 'Payload' with the value 'Payload', and 'QoS' with the value '0'. Below these fields are two more sections: 'Message sent before disconnecting (close message)' and 'Message sent on an unexpected disconnection (will message)'. Both sections have a dropdown arrow.

If everything is configured correctly and you press Deploy in the right top corner, you should see that the MQTT input node connects to the MQTT server. If you connect a “debug” node directly to its output, you will see the result in the right debug pane.



Check carefully that all the topics, you need, are listed in the debug pane and that you created all virtual sensors for it.

debug

all nodes

02/11/2019, 13:18:23 node: 253a5899.d2ce6

volvo, lock/lock/state : msg.payload : string[4]

"lock"

02/11/2019, 13:18:24 node: 253a5899.d2ce6

volvo, switch/heater/state : msg.payload : string[3]

"off"

02/11/2019, 13:18:24 node: 253a5899.d2ce6

volvo/, g/sensor/odometer/state : msg.payload : string[4]

"5039"

02/11/2019, 13:18:24 node: 253a5899.d2ce6

volvo, sensor/trip_meter1/state : msg.payload : string[4]

"5039"

02/11/2019, 13:18:24 node: 253a5899.d2ce6

volvo, /sensor/trip_meter2/state : msg.payload : string[4]

"5039"

02/11/2019, 13:18:24 node: 253a5899.d2ce6

volvo/ 'sensor/fuel_amount/state : msg.payload : string[2]

"34"

02/11/2019, 13:18:24 node: 253a5899.d2ce6

volvo, sensor/fuel_amount_level/state : msg.payload : string[2]

"54"

The output of the MQTT input node is connected to the “Router Node”. Any (green) debug node is optional and can be removed safely.

The task of the “Router node” is to split different topics to different outputs, because some topics require extra and/or a different “treatment”.

Edit routing node

Delete Cancel Done

⚙️ **Properties** ⚙️ 📄 🖨️

📁 Name Router

≡	volvo/abc123/*path	→ 1	✕
≡	volvo/abc123/binary_sensor/is_locked/state	→ 2	✕
≡	volvo/abc123/switch/heater/state	→ 3	✕
≡	volvo/abc123/sensor/trips/state	→ 4	✕

The first path routes everything, except the three below, to output 1
So insert: volvo/abc123/*path (mind the asterisk).

As we will insert later a push button, to lock and unlock the car, we will route the lock information to output 2. So insert: volvo/abc123/binary_sensor/is_locked/state to the second output.

The same applies for the heater (if applicable) and we will route this to output 3.
So insert: volvo/abc123/switch/heater/state to the third output.

As we want to modify our date and time, before we send it further, we will route the latest trip information to output 4. So insert: volvo/abc123/sensor/trips/state to output 4.

If we have more functions available, e.g. remote start engine, it might be necessary to create additional outputs and insert their corresponding topics.

The first output of the “Router” node is connected to 2 “Function” nodes. In principal, it can be 1 function node, but in order to avoid that the Domoticz log is quickly filled with the same message for every binary sensor (Alarm or Text), only a message will be sent, if the message is different from the previous message. This saves a lot of log entries in Domoticz.

In principal this can also be done with the analogue sensor, but a few sensors, such as “Fuel

Consumption” and “Average Speed” will not change much. Especially if the car makes more kilometers or miles, the impact on the fuel consumption per kilometer or average speed becomes less and less.

The contents of the first function node, called “Convert Binary Sensors to Domoticz” is as follows:

```
// Function node for Binary Sensors
// Declare variables
var number_plate = "abc123";

// Binary Sensors which will give alert

if (msg.topic == "volvo/" + number_plate + "/binary_sensor/doors_hood_open/state"){
  if (msg.payload == "close") {
    msg.payload = {"command":"udevice","idx":204,"nvalue":1,"svalue":"Closed"};
  } else {
    msg.payload = {"command":"udevice","idx":204,"nvalue":4,"svalue":"Open"};
  }
  return msg;
}

if (msg.topic == "volvo/" + number_plate + "/binary_sensor/doors_front_left_door_open/state"){
  if (msg.payload == "close") {
    msg.payload = {"command":"udevice","idx":200,"nvalue":1,"svalue":"Closed"};
  } else {
    msg.payload = {"command":"udevice","idx":200,"nvalue":4,"svalue":"Open"};
  }
  return msg;
}

if (msg.topic == "volvo/" + number_plate + "/binary_sensor/doors_front_right_door_open/state"){
  if (msg.payload == "close") {
    msg.payload = {"command":"udevice","idx":203,"nvalue":1,"svalue":"Closed"};
```

```
} else {
msg.payload = {"command":"udevice","idx":203,"nvalue":4,"svalue":"Open"};
}
return msg;
}

if (msg.topic == "volvo/" + number_plate + "/binary_sensor/doors_rear_left_door_open/state"){
if (msg.payload == "close") {
msg.payload = {"command":"udevice","idx":201,"nvalue":1,"svalue":"Closed"};
} else {
msg.payload = {"command":"udevice","idx":201,"nvalue":4,"svalue":"Open"};
}
return msg;
}

if (msg.topic == "volvo/" + number_plate + "/binary_sensor/doors_rear_right_door_open/state"){
if (msg.payload == "close") {
msg.payload = {"command":"udevice","idx":202,"nvalue":1,"svalue":"Closed"};
} else {
msg.payload = {"command":"udevice","idx":202,"nvalue":4,"svalue":"Open"};
}
return msg;
}

if (msg.topic == "volvo/" + number_plate + "/binary_sensor/doors_tailgate_open/state"){
if (msg.payload == "close") {
msg.payload = {"command":"udevice","idx":205,"nvalue":1,"svalue":"Closed"};
} else {
msg.payload = {"command":"udevice","idx":205,"nvalue":4,"svalue":"Open"};
}
return msg;
}
```

```
}

if (msg.topic == "volvo/" + number_plate + "/binary_sensor/windows_front_left_window_open/state"){
if (msg.payload == "close") {
msg.payload = {"command":"udevice","idx":196,"nvalue":1,"svalue":"Closed"};
} else {
msg.payload = {"command":"udevice","idx":196,"nvalue":4,"svalue":"Open"};
}
return msg;
}

if (msg.topic == "volvo/" + number_plate + "/binary_sensor/windows_front_right_window_open/state"){
if (msg.payload == "close") {
msg.payload = {"command":"udevice","idx":199,"nvalue":1,"svalue":"Closed"};
} else {
msg.payload = {"command":"udevice","idx":199,"nvalue":4,"svalue":"Open"};
}
return msg;
}

if (msg.topic == "volvo/" + number_plate + "/binary_sensor/windows_rear_left_window_open/state"){
if (msg.payload == "close") {
msg.payload = {"command":"udevice","idx":197,"nvalue":1,"svalue":"Closed"};
} else {
msg.payload = {"command":"udevice","idx":197,"nvalue":4,"svalue":"Open"};
}
return msg;
}

if (msg.topic == "volvo/" + number_plate + "/binary_sensor/windows_rear_right_window_open/state"){
if (msg.payload == "close") {
```

```
msg.payload = {"command":"udevice","idx":198,"nvalue":1,"svalue":"Closed"}; Convert Binary Sensors to Domoticz
```

```
} else {
```

```
msg.payload = {"command":"udevice","idx":198,"nvalue":4,"svalue":"Open"};
```

```
}
```

```
return msg;
```

```
}
```

```
// Binary Sensors which will give text
```

```
if (msg.topic == "volvo/" + number_plate +  
"/binary_sensor/tyre_pressure_front_left_tyre_pressure/state"){
```

```
if (msg.payload == "safe") {
```

```
msg.payload = {"command":"udevice","idx":206,"nvalue":0,"svalue":"Safe"};
```

```
} else {
```

```
msg.payload = {"command":"udevice","idx":206,"nvalue":0,"svalue":"Unsafe"};
```

```
}
```

```
return msg;
```

```
}
```

```
if (msg.topic == "volvo/" + number_plate +  
"/binary_sensor/tyre_pressure_front_right_tyre_pressure/state"){
```

```
if (msg.payload == "safe") {
```

```
msg.payload = {"command":"udevice","idx":207,"nvalue":0,"svalue":"Safe"};
```

```
} else {
```

```
msg.payload = {"command":"udevice","idx":207,"nvalue":0,"svalue":"Unsafe"};
```

```
}
```

```
return msg;
```

```
}
```

```
if (msg.topic == "volvo/" + number_plate +  
"/binary_sensor/tyre_pressure_rear_left_tyre_pressure/state"){
```

```
if (msg.payload == "safe") {
```

```
msg.payload = {"command":"udevice","idx":208,"nvalue":0,"svalue":"Safe"};
} else {
msg.payload = {"command":"udevice","idx":208,"nvalue":0,"svalue":"Unsafe"};
}
return msg;
}

if (msg.topic == "volvo/" + number_plate +
"/binary_sensor/tyre_pressure_rear_right_tyre_pressure/state"){
if (msg.payload == "safe") {
msg.payload = {"command":"udevice","idx":209,"nvalue":0,"svalue":"Safe"};
} else {
msg.payload = {"command":"udevice","idx":209,"nvalue":0,"svalue":"Unsafe"};
}
return msg;
}

if (msg.topic == "volvo/" + number_plate + "/binary_sensor/washer_fluid_level/state"){
if (msg.payload == "safe") {
msg.payload = {"command":"udevice","idx":218,"nvalue":0,"svalue":"Safe"};
} else {
msg.payload = {"command":"udevice","idx":218,"nvalue":0,"svalue":"Unsafe"};
}
return msg;
}

if (msg.topic == "volvo/" + number_plate + "/binary_sensor/brake_fluid/state"){
if (msg.payload == "safe") {
msg.payload = {"command":"udevice","idx":217,"nvalue":0,"svalue":"Safe"};
} else {
msg.payload = {"command":"udevice","idx":217,"nvalue":0,"svalue":"Unsafe"};
}
}
```

```
return msg;
}

if (msg.topic == "volvo/" + number_plate + "/binary_sensor/service_warning_status/state"){
if (msg.payload == "safe") {
msg.payload = {"command":"udevice","idx":224,"nvalue":0,"svalue":"Safe"};
} else {
msg.payload = {"command":"udevice","idx":224,"nvalue":0,"svalue":"Unsafe"};
}
return msg;
}

if (msg.topic == "volvo/" + number_plate + "/binary_sensor/bulb_failures/state"){
if (msg.payload == "safe") {
msg.payload = {"command":"udevice","idx":226,"nvalue":0,"svalue":"Safe"};
} else {
msg.payload = {"command":"udevice","idx":226,"nvalue":0,"svalue":"Unsafe"};
}
return msg;
}

// Binary Sensors not in use

//if (msg.topic == "volvo/" + number_plate + "/binary_sensor/any_door_open/state"){
//if (msg.payload == "close") {
//msg.payload = {"command":"udevice","idx":xxx,"nvalue":1,"svalue":"Closed"};
//} else {
//msg.payload = {"command":"udevice","idx":xxx,"nvalue":4,"svalue":"Open"};
//}
//return msg;
//}
```



```
//if (msg.topic == "volvo/" + number_plate + "/binary_sensor/any_window_open/state"){  
//if (msg.payload == "close") {  
//msg.payload = {"command":"udevice","idx":yyy,"nvalue":1,"svalue":"Closed"};  
//} else {  
//msg.payload = {"command":"udevice","idx":yyy,"nvalue":4,"svalue":"Open"};  
//}  
//return msg;  
//}
```

```
if (msg.topic == "volvo/" + number_plate + "/binary_sensor/is_engine_running/state"){  
if (msg.payload == "off") {  
msg.payload = {"command":"udevice","idx":230,"nvalue":0,"svalue":"Off"};  
} else {  
msg.payload = {"command":"udevice","idx":230,"nvalue":0,"svalue":"On"};  
}  
return msg;  
}
```

```
if (msg.topic == "volvo/" + number_plate + "/lock/lock/state"){  
if (msg.payload == "lock") {  
msg.payload = {"command":"udevice","idx":228,"nvalue":0,"svalue":"Locked"};  
} else {  
msg.payload = {"command":"udevice","idx":228,"nvalue":0,"svalue":"Unlocked"};  
}  
return msg;  
}
```

```
if (msg.topic == "volvo/" + number_plate + "/switch/heater/state"){  
if (msg.payload == "off") {  
msg.payload = {"command":"udevice","idx":229,"nvalue":0,"svalue":"Off"};  
}
```

```

} else {
msg.payload = {"command":"udevice","idx":229,"nvalue":0,"svalue":"On"};
}
return msg;
}

if (msg.topic == "volvo/" + number_plate + "/sensor/trips/state"){
msg.payload = {"command":"udevice","idx":227,"nvalue":0,"svalue":(msg.payload)};
return msg;
}

```

The user has to replace the value of idx with its own idx numbers and the var number_plate (abc123) for its own license plate.

For the Binary Sensors which will give alert, the “nvalue” represents the colour (0 = grey, 1 = green, 2 = yellow, 3 = orange, 4 = red). The “svalue” represents the text to display. You can translate it to your own language, if desired.

For the Binary Sensors which will give text, the “nvalue” is always 0. The “svalue” represents the text to display. You can translate it to your own language, if desired.

Two binary sensors are not in use (any_door_open and any_window_open). Feel free to use them, by removing //. They will behave as an “Alert” sensor.

The contents of the second function node, called “Convert Analog Sensors to Domoticz” is as follows:

```

// Function node for Analog Sensors
// Declare variables
var number_plate = "abc123";

// Sensors Analogue values

if (msg.topic == "volvo/" + number_plate + "/sensor/odometer/state"){
msg.payload = {"command":"udevice","idx":222,"nvalue":0,"svalue":(msg.payload)};
return msg;
}

if (msg.topic == "volvo/" + number_plate + "/sensor/fuel_amount/state"){
msg.payload = {"command":"udevice","idx":212,"nvalue":0,"svalue":(msg.payload)};

```

```
return msg;
}

if (msg.topic == "volvo/" + number_plate + "/sensor/fuel_amount_level/state"){
msg.payload = {"command":"udevice","idx":213,"nvalue":0,"svalue":(msg.payload)};
return msg;
}

if (msg.topic == "volvo/" + number_plate + "/sensor/average_fuel_consumption/state"){
msg.payload = {"command":"udevice","idx":214,"nvalue":0,"svalue":(msg.payload)};
return msg;
}

if (msg.topic == "volvo/" + number_plate + "/sensor/distance_to_empty/state"){
msg.payload = {"command":"udevice","idx":216,"nvalue":0,"svalue":(msg.payload)};
return msg;
}

if (msg.topic == "volvo/" + number_plate + "/sensor/average_speed/state"){
msg.payload = {"command":"udevice","idx":215,"nvalue":0,"svalue":(msg.payload)};
return msg;
}

// Sensors not in use

//if (msg.topic == "volvo/" + number_plate + "/sensor/trip_meter1/state"){
//msg.payload = {"command":"udevice","idx":aaa,"nvalue":0,"svalue":(msg.payload)};
//return msg;
//}

//if (msg.topic == "volvo/" + number_plate + "/sensor/trip_meter2/state"){
```

```
//msg.payload = {"command":"udevice","idx":bbb,"nvalue":0,"svalue":(msg.payload)};;
//return msg;
//}
```

The user has to replace the value of idx with its own idx numbers and the var number_plate (abc123) for its own license plate.

Two analogue sensors are not in use (trip_meter1 and trip_meter2). Feel free to use them, by removing //.

The second output of the “Router” node is connected to a RBE (Report by Exception), called “Send only Once”. This node blocks identical payloads and it avoids that a loop will exist, because later on we will add a push button for “Lock/Unlock”.

The configuration is the default configuration and we don't have to change anything.

The output of this node has to be connected to another function node.

The contents is as follows:

```
if (msg.payload == "on") {
msg.payload = {"command":"udevice","idx":210,"nvalue":1,"svalue":"0"};
} else {
msg.payload = {"command":"udevice","idx":210,"nvalue":0,"svalue":"0"};
}
return msg;
```

The third output of the “Router” node is also connected to a RBE (Report by Exception), called “Send only Once”, identical to the previous one. This node blocks also identical payloads and it avoids that a loop will exist, because later on we will add a push button for the heater “On/Off”.

The output of this node has to be connected to another function node.
The contents is as follows:

```
if (msg.payload == "off") {  
msg.payload = {"command":"udevice","idx":211,"nvalue":0,"svalue":"0"};  
} else {  
msg.payload = {"command":"udevice","idx":211,"nvalue":1,"svalue":"0"};  
}  
return msg;
```

Also here the user has to replace the value of idx with its own idx numbers.

The fourth output of the “Router” node is connected to a node, called Date/Time Formatter and is actually a “moment” node.

The output of the “Router” node presents the date and time in the following format:

```
"2019-10-25 11:57:14+02:00"
```

If this is not what you want, you can convert it easily, as follows:

Edit moment node

Delete
Cancel
Done

⚙ Properties

⚙
📄
🖨

← Input from

▼ msg. payload

Input Timezone

Europe/Amsterdam

Adjustment

+ ▼
0
Days ▼

Output Format

DD-MM-YYYY HH:mm:ss

🇺🇸 Locale

en_US

Output Tz

Europe/Amsterdam

→ Output to

▼ msg. payload

📄 Topic

Topic

🔖 Name

Name

See the info sidebar for formatting details.
Use locale and format to change string output.
See the info sidebar for several warnings about inputting strings.

The output of this “Date/Time Formatter” node has to be connected to the input of the “Convert Binary Sensors to Domoticz” node.

The output of the “Convert Binary Sensors to Domoticz” is connected to another RBE (Report by Exception), called “Block Unchanged”, in order to prevent, that the Domoticz log is filled with identical messages quickly.

The output of this RBE node is, together with the outputs of the three other function nodes connected to a MQTT output node, called “Domoticz In”.

The configuration of this node is almost identical to the MQTT input node, except for the topic.

The Topic has to be: domoticz/in.

All other properties can be left as default.

If everything has been configured successfully all the sensors will appear in Domoticz under the “Utility” tab. Other sensors can be added easily, if you split it into binary and analogue sensors and copy a code block in either the “Convert Binary Sensors to Domoticz” node or “Convert Analog Sensors to Domoticz” node.

The final result, you see on the next page.



6 Node-Red flow Domoticz to VoC

Before we start, we should test the functionality from the command line.

So go to the command line on the device, where the VoC script is running and issue the following commands, if your car supports it and check that the commands are carried out:

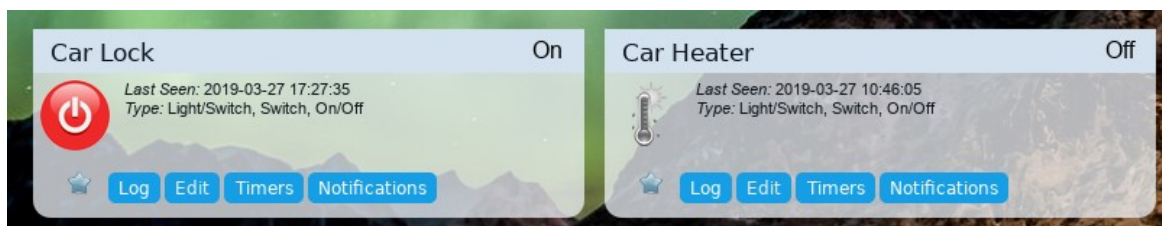
- `$ voc lock`
- `$ voc unlock`
- `$ voc heater on`
- `$ voc heater off`
- `$ voc engine on`
- `$ voc engine off`

Note: The VoC help file indicate start and stop. If the commands on/off do not work, try start/stop instead.

We have to create several virtual switches.

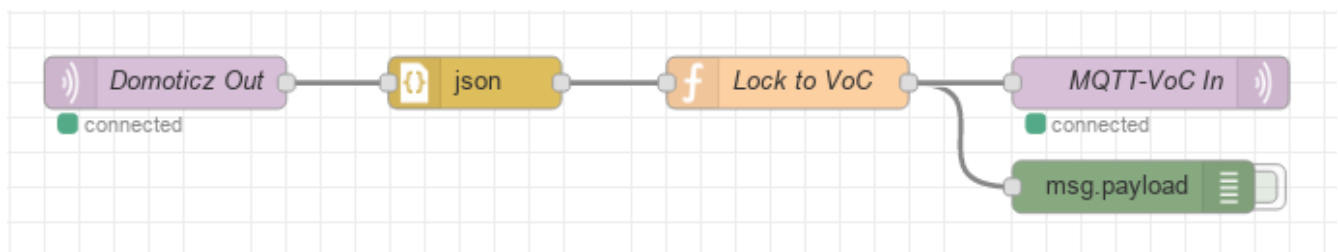
In order to create these switches, if not done already, create in “Hardware” a Dummy (Does nothing, use for virtual switches only) and add the virtual switches.

Each user has to check, which commands are supported for his/her Volvo.



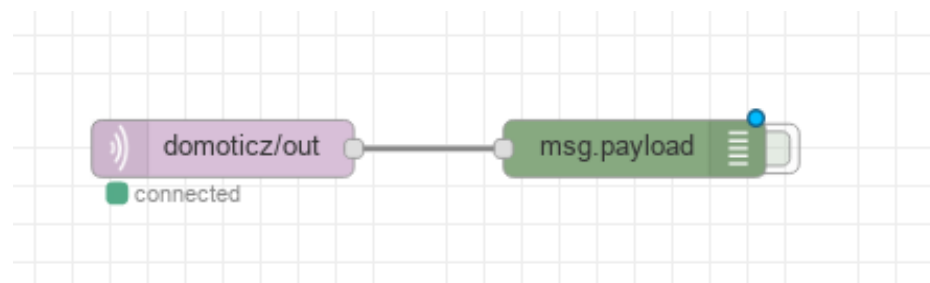
Go to “Setup” and then to “Devices”. Make a note of the Idx number of the newly created switches, as we will need these later.

In the picture below, you will see the total flow.



Every event in Domoticz is published to domoticz/out. So if you press on one of the newly created switched a message is published.

If you connect a debug node to the output of “Domoticz Out” you will see a lot of messages published. We have to filter the right one and therefore we need the Idx number of the switch.



all nodes



01/09/2019, 18:41:27 node: 1231abb9.5da8f4

domoticz/out : msg.payload : string[231]

```
▶ "{ "Battery" : 255, "RSSI" :  
12, "description" : "", "dtype"  
: "General", "id" : "00000001",  
"idx" : 18, "name" : "Voltage  
L1", "nvalue" : 0, "stype" :  
"Voltage", "svalue1" :  
"232.000", "unit" : 1 }"
```

01/09/2019, 18:41:27 node: 1231abb9.5da8f4

domoticz/out : msg.payload : string[231]

```
▶ "{ "Battery" : 255, "RSSI" :  
12, "description" : "", "dtype"  
: "General", "id" : "00000002",  
"idx" : 19, "name" : "Voltage  
L2", "nvalue" : 0, "stype" :  
"Voltage", "svalue1" :  
"233.000", "unit" : 1 }"
```

01/09/2019, 18:41:27 node: 1231abb9.5da8f4

domoticz/out : msg.payload : string[231]

```
▶ "{ "Battery" : 255, "RSSI" :  
12, "description" : "", "dtype"  
: "General", "id" : "00000003",  
"idx" : 20, "name" : "Voltage  
L3", "nvalue" : 0, "stype" :  
"Voltage", "svalue1" :  
"232.000", "unit" : 1 }"
```

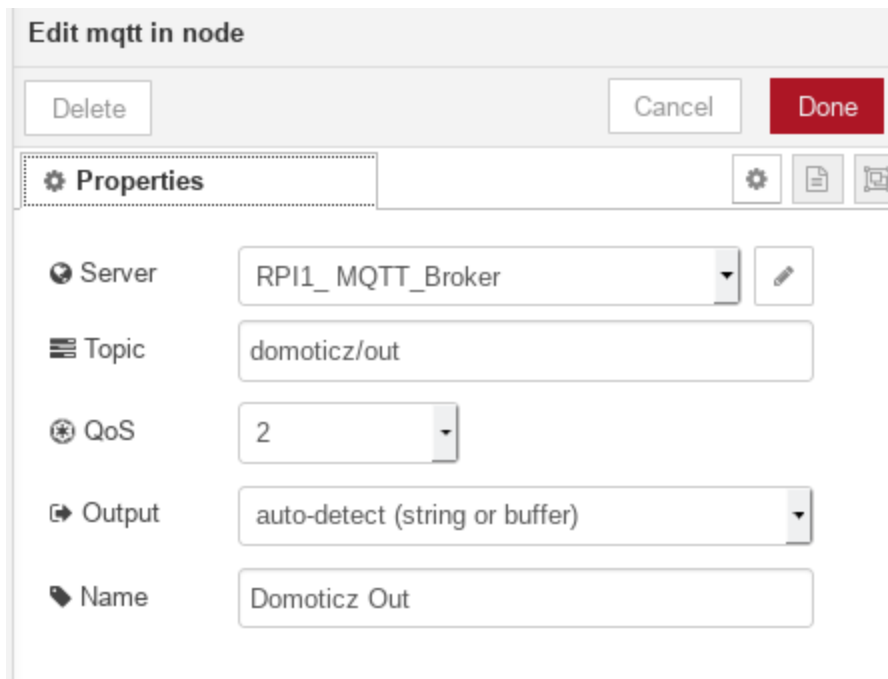
01/09/2019, 18:41:27 node: 1231abb9.5da8f4

domoticz/out : msg.payload : string[238]

```
▶ "{ "Battery" : 255, "RSSI" :
```

The first node for this flow is a MQTT input node, called “Domoticz Out”

The configuration is identical to the previous used MQTT nodes, except the Topic.



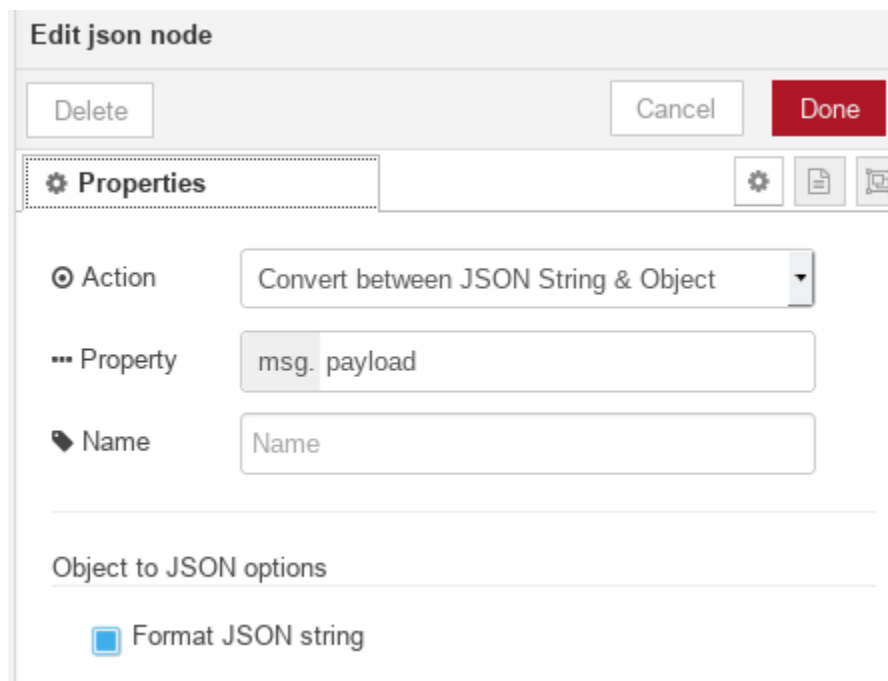
The screenshot shows the 'Edit mqtt in node' configuration window. At the top, there are three buttons: 'Delete', 'Cancel', and 'Done'. Below these is a 'Properties' section with a gear icon and three sub-panels: 'Server', 'Topic', and 'QoS'. The 'Server' dropdown is set to 'RPI1_ MQTT_Broker'. The 'Topic' text field contains 'domoticz/out'. The 'QoS' dropdown is set to '2'. Below these is an 'Output' dropdown set to 'auto-detect (string or buffer)'. At the bottom, the 'Name' text field contains 'Domoticz Out'.

The Topic has to be configured as: domoticz/out

As we can see the output is a JavaScript string, but we like to see a JSON message.

Therefore the second node is a JSON function node, which converts between a JSON string and an object.

The configuration is as follows:



The screenshot shows the 'Edit json node' configuration window. At the top, there are three buttons: 'Delete', 'Cancel', and 'Done'. Below these is a 'Properties' section with a gear icon and three sub-panels: 'Action', 'Property', and 'Name'. The 'Action' dropdown is set to 'Convert between JSON String & Object'. The 'Property' text field contains 'msg. payload'. The 'Name' text field contains 'Name'. Below these is a section titled 'Object to JSON options' with a checkbox labeled 'Format JSON string' which is currently unchecked.

The output of the JSON function node is connected to the input of a function node.

This function node takes care for selecting the right idx number of the switch and converts the messages from Domoticz to a message suitable to be handled by the VoC python script.

The contents is as follows:

```
// Declare variables
var number_plate = "abc123";

// Lock
if (msg.payload.idx == "210") {
    msg.topic = "volvo/" + number_plate + "/lock/lock/cmd";
    if (msg.payload.nvalue == 1) {
        msg.payload = "lock";
    }else{
        msg.payload = "unlock";
    }
}
return msg;
}

//Heater
//Not available anymore
//if (msg.payload.idx == "211") {
//    msg.topic = "volvo/" + number_plate + "/switch/heater/cmd";
//    if (msg.payload.nvalue == 1) {
//        msg.payload = "on";
//    }else{
//        msg.payload = "off";
//    }
//}
//return msg;
//}
```

As my new car has no heater available any more, this section is commented. If your car is capable to start the heater remotely, simply uncomment this section. If your car is capable to start the engine remotely, simply copy a section. Replace the idx numbers with your numbers and the var number_plate (abc123) for your own license plate.

The output of the function node “Lock to VoC” is connected to the input of a MQTT output node, called MQTT-VoC In.

The configuration is identical as the previous MQTT nodes, except for the Topic.

Edit mqtt out node

Delete

Cancel

Done

Properties

Server

RPI1_MQTT_Broker

Topic

Topic

QoS

2

Retain

Name

MQTT-VoC In

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

The Topic should be left empty, as we have set already the topic in the previous function node to volvo/abc123/lock/lock/cmd or volvo/abc123/switch/heater/cmd.

In case you have added a switch for remote engine start/stop you can test it yourself, remembering that the VoC python script has subscribed to the command with /cmd.

It is now time to test the functionality.

7 Temperature and Location

In the Volvo on Call app you will see on the "Home" page "Weather". It indicates only the temperature, either in degrees Celsius or Fahrenheit. This depends on the configuration in App Settings. It presents the Temperature **at** the car. The temperature sensor of the car is not transmitted to the Volvo on Call server and so it is not published by the VoC python script to the MQTT server. What you see in your app is not the temperature, measured by the temperature sensor of your car, but the temperature on the location of the car.

Volvo has an agreement with HERE Technologies, <https://www.here.com/>, as many other car manufacturers. Also your maps are delivered by HERE.

You can use any other weather services, but in this example we will use HERE, as well. You cannot use the co-ordinates of your "Home", as configured under "Settings and Location" in Domoticz, as your car is moving and not always at home. Sometimes, I see differences between the weather information, delivered by different weather services. By using HERE, as the weather service, the results in Domoticz are identical with the results in your VoC app.

In order to get the weather data from HERE, you have to make an account.

Go to: <https://account.here.com/> and create your developer account.

You will receive an email from HERE and some guidelines to access your, so called, app_id and app_code. You will need these in order to have access to the HERE Api's.

You will receive 250.000 request/month for free. To get the weather, every 5 minutes and that is the default time interval between two subscribes to the MQTT server, you need approx 9000 requests. This is enough for our application.

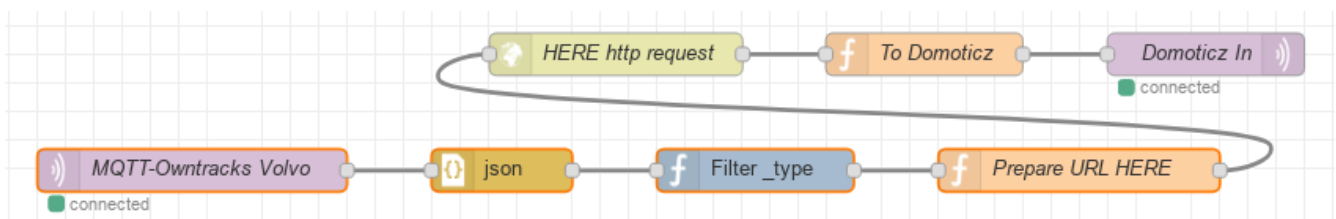
As we will request the weather for the co-ordinates of the car, we have more weather data available than the temperature only. So, if you wish, you can configure extra sensors, such as humidity or pressure at the car. For now, we will only create a temperature sensor.

We have to create another virtual sensor.

In order to create this sensor, if not done already, create in "Hardware" a Dummy (Does nothing, use for virtual switches only) and add the virtual Temperature sensor.

Next, go to "Setup" and then to "Devices". Make a note of the Idx number of the newly created sensor, as we will need this later.

In the picture below you will see the total flow.



The first node for this flow is a MQTT input node, called "MQTT-Owntracks Volvo"

The configuration of the node is as follows:

Edit mqtt in node

Delete
Cancel
Done

Properties

Server
RPI1_ MQTT_Broker

Topic
owntracks/volvo/abc123

QoS
2

Output
auto-detect (string or buffer)

Name
MQTT-Owntracks Volvo

The Topic has to be configured as: owntracks/volvo/abc123, where abc123 is your license number. The topic, that the VoC python script publishes to, is different than the previous topics. It is compatible with the Owntracks format. So, if you have installed the Owntracks app on your phone, you will see the data in your Owntracks app under the name “volvo”. The received data is fully Owntracks compatible.

```
{ "_type": "location", "tid": "volvo", "t": "p", "lat": 52.12345678912345, "lon": 5.123456789012345, "acc": 1, "tst": 1572809685, "now": 1572887506, "tst_iso": "2019-11-03T19:34:45+00:00" }
```

The next node is a JSON function node, which converts between a JSON string and an object. As an alternative, you can also use the “Owntracks” node, which has to be installed from the palette. The configuration is as follows:

Edit json node

Delete
Cancel
Done

Properties

Action
Convert between JSON String & Object

Property
msg. payload

Name
Name

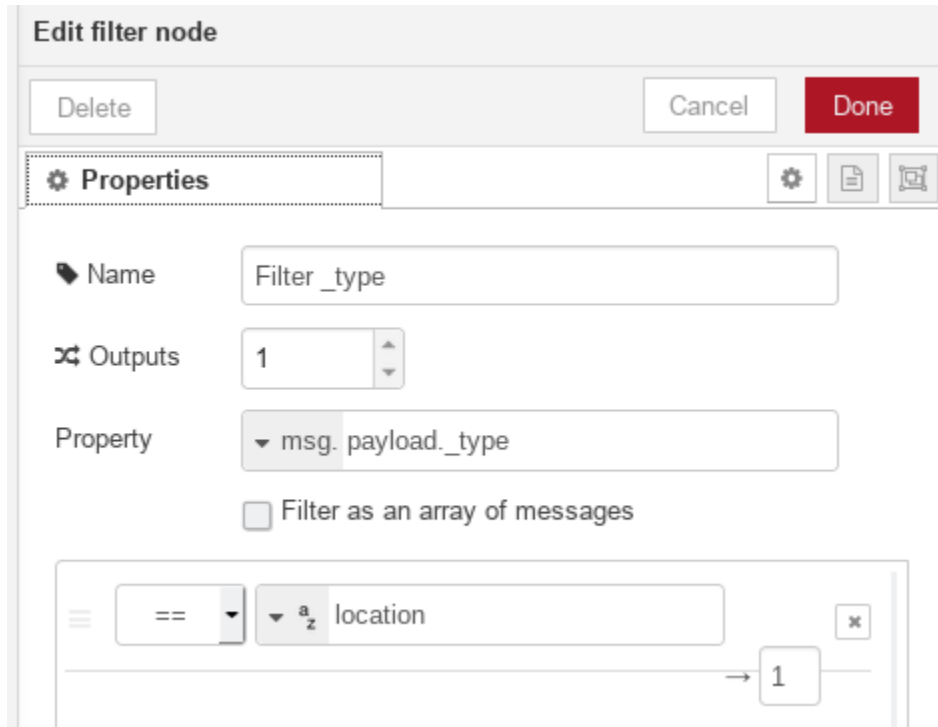
Object to JSON options
☐ Format JSON string

This will give the following output, which you can see, if you connect: a debug node:

```
object
_type: "location"
tid: "volvo"
t: "p"
lat: 52.12345678912345
lon: 5.123456789012345
acc: 1
tst: 1572809685
now: 1572887808
tst_iso: "2019-11-03T19:34:45+00:00"
```

Because Owntracks is able to send more data, which will have a different `_type`, we will add a filter node, to be sure that only `_type` equals "location" is transferred to the next node. This node is not a core node and must be installed from the palette. Search for "node-red-contrib-filter" and install it.

The configuration is as follows:



As we will receive the location data from the VoC python script and we have to add information about the `app_id` and `app_code` from HERE and additional product information, we will construct the request in a function node.

The contents of this function node is as follows:

```
var url = "https://weather.api.here.com/weather/1.0/report.json";  
var prod = "observation";  
var app_id = "Your app_id here";  
var app_code = "Your app_code here";  
var newUrl = { url: url + "?product=" + prod + "&latitude=" + msg.payload.lat + "&longitude=" +  
msg.payload.lon + "&oneobservation=true&app_id=" + app_id + "&app_code=" + app_code};  
return newUrl;
```

The output of the function node is connected to the input of a “http request” node from the “Function” section. This node, called “HERE http request”, will do a http request to the HERE weather service.

As we have already prepared the URL in the function node, we can leave the field URL blank. So the configuration is as follows:

The screenshot shows the 'Edit http request node' dialog box. At the top, there are 'Delete', 'Cancel', and 'Done' buttons. Below them is a 'Properties' tab with icons for settings, a file, and a preview. The configuration fields are as follows:

- Method:** A dropdown menu set to 'GET'.
- URL:** A text input field containing 'http://'.
- Append msg.payload as query string parameters:** An unchecked checkbox.
- Enable secure (SSL/TLS) connection:** An unchecked checkbox.
- Use authentication:** An unchecked checkbox.
- Use proxy:** An unchecked checkbox.
- Return:** A dropdown menu set to 'a parsed JSON object'.
- Name:** A text input field containing 'HERE http request'.

At the bottom, there is a yellow tip box that reads: 'Tip: If the JSON parse fails the fetched string is returned as-is.'

If you look to the received data in a debug node, you will see that we have received complete weather information from HERE for the cars location. We will only use the received temperature, but if you want to use more, you can create additional sensors and add these in the next function node.

The contents of the function node called “To Domoticz” is as follows:

```
msg.payload =  
{ "command": "udevice", "idx": 231, "nvalue": 0, "svalue": parseFloat(msg.payload.observations.location[0].  
observation[0].temperature).toFixed(0)};  
return msg;
```

The user has to change the idx number to the index number of the recently created virtual “Temperature” sensor.

The temperature is converted to an integer value, using `parseFloat.toFixed(0)`. You can change the value 0, between `()`, to your desired value. The output of the function node is connected to a MQTT input node, called “Domoticz In”.

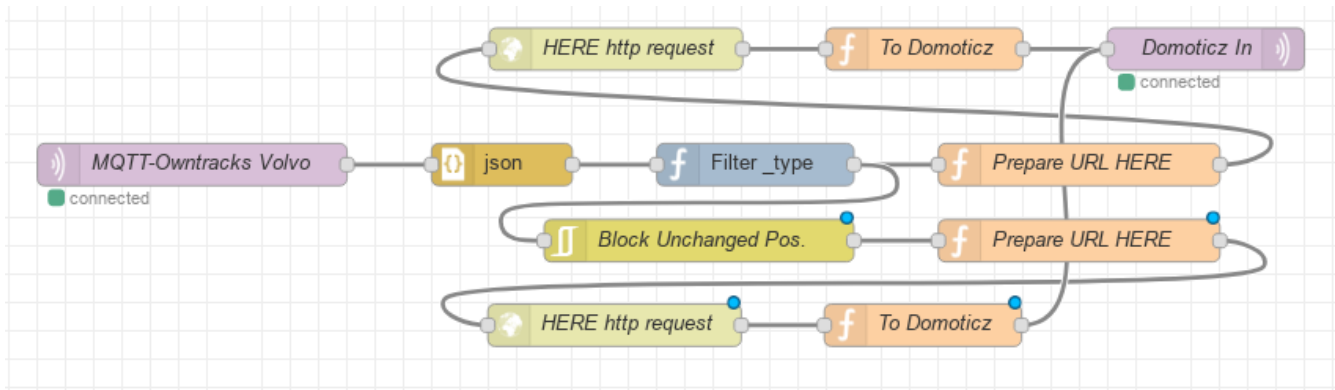
The configuration is as follows:

This is the same “Domoticz In” node, as used before.

You will find your “Temperature at Car” sensor in Domoticz under the “Temperature” tab.

As we use the lat/lon co-ordinates to retrieve the temperature, it is a small step to use these co-ordinates to retrieve the address as well.

So create a virtual “Text” sensor. Make a note of the Idx number of the newly created sensor, as we will need this later.



As it is useless to retrieve addresses, unlike the temperature, every 5 minutes, while the car is in a parked position, we will connect a RBE node to the “Filter” node, called “Filter_type”.

The configuration of the RBE node, called “Block Unchanged Pos.” is as follows:

Edit rbe node

Delete
Cancel
Done

Properties

Mode
block unless value changes

Property
msg.payload.tst

Name
Block Unchanged Pos.

As long as the location (lat/lon) is unchanged, we will not request a new address.

We can use any geo location service to translate lat/lon co-ordinates to addresses, but as we have already a HERE account, it is the easiest way to use HERE as well.

The request is different than used for the weather request, so we have to create a function node, with the following contents:

```
var url = "https://reverse.geocoder.api.here.com/6.2/reversegeocode.json";
var mode = "retrieveAddresses";
var app_id = "Your app_id here";
var app_code = "Your app_code here";
var newUrl = { url: url + "?prox=" + msg.payload.lat + "%2C" + msg.payload.lon + "%2C" + "10" +
"&mode=" + mode + "&maxresults=1&gen=9" + "&app_id=" + app_id + "&app_code=" + app_code};
return newUrl;
```

The output of this function node is connected to the input of a “http request” node from the “Function” section. This node, called “HERE http request” is identical to the previous one and will make a request to the HERE reverse geo-coder service.

As we have already prepared the URL in the function node, we can leave the field URL blank.

If you look to the received data in a debug node, you will see that we have received the address from HERE for the cars location.

The contents of the function node called “To Domoticz” is as follows:

```
msg.payload = {"command":"udevice","idx":247,"nvalue":0,"svalue":  
(msg.payload.Response.View[0].Result[0].Location.Address.Label)};  
return msg;
```

The user has to change the idx number to the index number of the recently created virtual Text sensor.

The output of the function node is connected to the MQTT input node, called “Domoticz In”.

8 Monitoring of VoC script and MQTT server

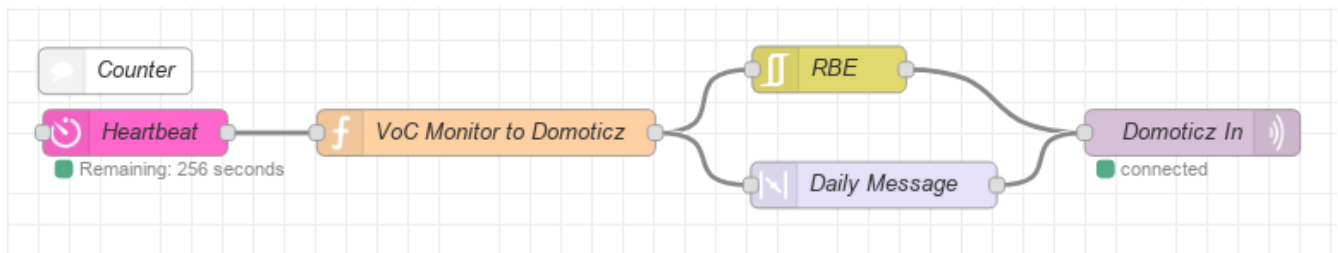
In the case, that the Volvo on Call python script runs on a different machine than the MQTT server and that Domoticz on its turn runs on another machine, it is a good idea to monitor the communication between the VoC script, MQTT server and Node Red. In order to do so, you have to create a virtual sensor, in this case an “Alert” sensor, as you can send notifications from this type of sensor. The sensor will check that every 5 minutes a topic is published to the MQTT server and that Node Red is able to subscribe to that topic. If you have select another publishing interval to the MQTT server, you should change this time. It is assumed that, if you are able to subscribe to one topic, every 5 minutes, the VoC python script, the MQTT server and Node Red is running.

In order to create this “Alert” sensor, if not done already, create in “Hardware” a Dummy (Does nothing, use for virtual switches only) and add the virtual Alert sensor.

Next, go to “Setup” and then to “Devices”. Make a note of the Idx number of the newly created sensor, as we will need this later.

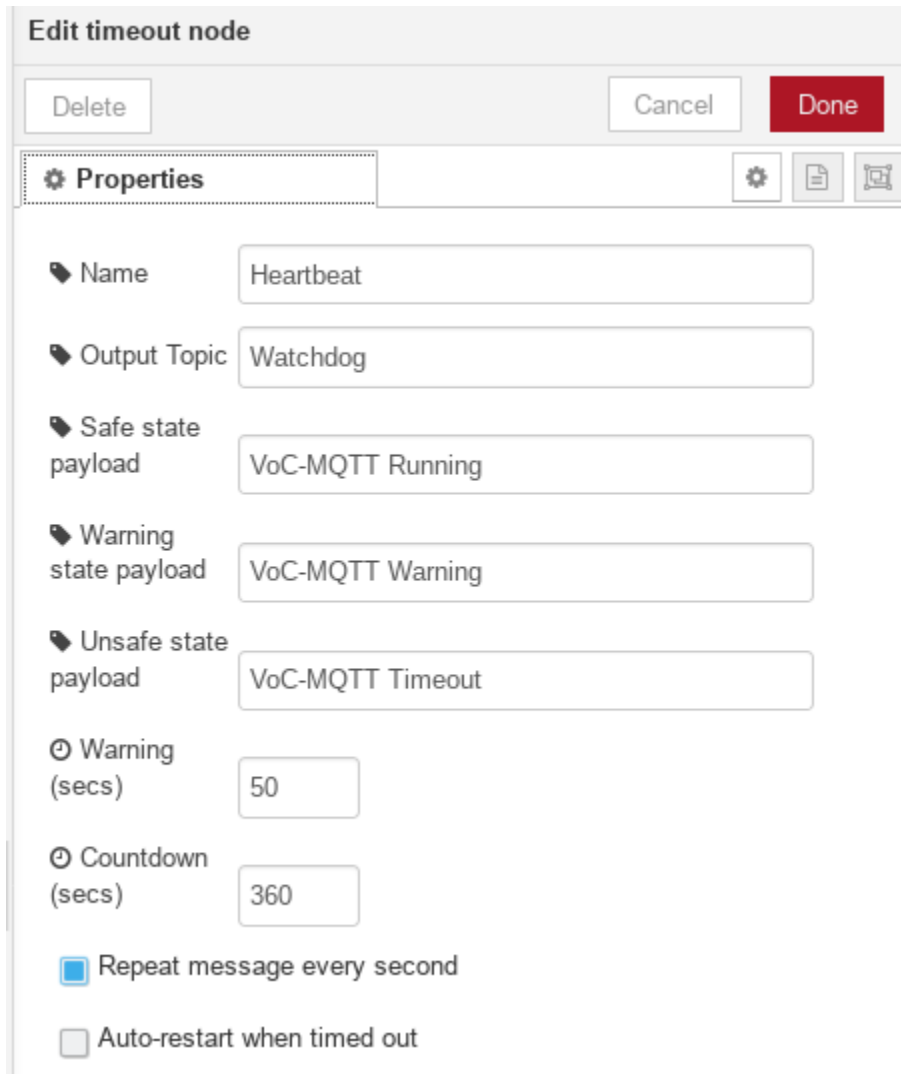
Also, if your VoC script, MQTT server, Node Red and Domoticz are running on the same hardware, you might want to configure some kind of monitoring.

In the picture below you will see the total flow.



The first node is not a core node in Node Red. You should install “node-red-contrib-timeout” from the palette.

The configuration is as follows:



Edit timeout node

Delete Cancel Done

Properties

Name Heartbeat

Output Topic Watchdog

Safe state payload VoC-MQTT Running

Warning state payload VoC-MQTT Warning

Unsafe state payload VoC-MQTT Timeout

Warning (secs) 50

Countdown (secs) 360

☒ Repeat message every second

☐ Auto-restart when timed out

The Name is up to the user. Optionally you can configure an Output Topic.

The text for “Safe state payload”, “Warning state payload” and “Unsafe state payload” is up to the user, as well, but this text is presented in the Domoticz “Alert” sensor.

You can change it, e.g. to your local language, but you have to change the next “Function” node too.

In this case the counter counts down from 360 seconds to 60 seconds. This represents the 5 minute publishing time of the VoC script. In addition 10 seconds has been added, so if the timer has not got any trigger since 370 seconds, it will enter the “Warning” state. If another 50 seconds has expired, it will enter the “Unsafe” state, which is considered as a time-out. The output of this timeout node, called “Heartbeat” is connected to the input of a “Function” node.

The contents of the function node called “VoC Monitor to Domoticz” is as follows:

```

switch (msg.payload) {

  case "VoC-MQTT Running": // delay since last received data: 0 -310 seconds
    msg.payload = {"command":"udevice","idx":287,"nvalue":1,"svalue":(msg.payload)};
    break;

  case "VoC-MQTT Warning": //delay since last received data: 310 - 360 seconds
    msg.payload = {"command":"udevice","idx":287,"nvalue":2,"svalue":(msg.payload)};
    break;

  case "VoC-MQTT Timeout": //delay since last received data: > 360 seconds
    msg.payload = {"command":"udevice","idx":287,"nvalue":4,"svalue":(msg.payload)};
    break;

}
return msg;

```

The user has to change the idx number to the index number of the recently created virtual Alert sensor. The “nvalue” represent the colour of the “Alert” sensor, 1 for green, 2 for yellow and 4 for red. If you have changed the text in the timeout mode, do not forget to change it in the “Function” node, as well (The text after case, between “”).

The output of the function node is connected to two nodes, a RBE (Report by Exception) and a “Throttle” node.

The last node is not a core node in Node Red. You should install “node-red-contrib-throttle” from the palette.

The function of this pair of parallel nodes is as follows:

To avoid, that the log in Domoticz is filled up quickly with messages every 5 minutes, indicating “VoC-MQTT Running”, we have inserted the RBE node.

The configuration of the RBE node is, as before, default.

However if you like to see, that this monitoring is still active, a “Throttle” node has been added, so that at least once a day 1 message is send to Domoticz. This reduces the log significantly (1 message, instead of 288 messages per day)

The configuration is as follows:

Edit throttle node

Delete Cancel Done

Properties ⚙️ 📄 🖨️

🔍 Name

⚙️ Throttle

🕒 Time Limit

☐ Locked by default?

Select the “Throttle” to “by time” and the “Time Limit” to 24 hours.

In this case you will receive 1 message per day, unless the status changes.

The output of the two nodes are connected to the MQTT input node, called “Domoticz In”.

If all desired sensors are configured, you may want to make a “Floor plan”, in this case a “Car plan” in order to see all sensors in one view.



9 Known issues

Only if the car is in a parked position, the car is communicating with the Volvo on Call server. Because of safety and privacy issues, according to Volvo, the car will not transmit its position, while it is moving. The Volvo on Call help desk is able to locate a cars position, e.g. in case of car theft, but this function is not available to others.

This means that both the temperature en the location is not available, while the car is moving. So Domoticz will display the temperature at the last known location, as well as the last address and it will update again, as soon as the car has been parked again.

It has been noted that the gps co-ordinates are not always correctly updated and still indicate the cars previous position.

This will have effect on the temperature, the time of the last trip and the address, both in Domoticz, but also in the VoC app.

The honk and blink function does not work, although it seemed to be supported in the VoC python script. If the correct syntax is known, it will be implemented.

Theft alarm is not implemented. If you run `voc print`, you will see that the function is available, but it is unsupported. It can be easily added, but it requires testing, without alarming the Volvo on Call service desk.

10 Credits

Erik Eriksson (molobrakos), who provided the Volvo on Call Python script.

<https://github.com/molobrakos/volvooncall>



FireWizard
11 November 2019