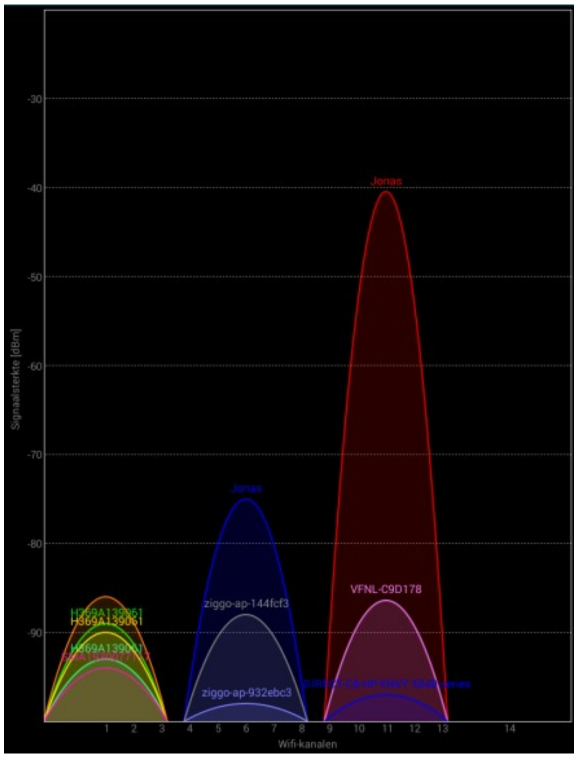
(Howto) Monitor Wireless Access Points

1. Introduction

This document describes the monitoring of Wireless Access Points, makes it available as a MQTT message and publish it to Domoticz.

Today almost every premises has its own wireless network with one or more access points. Unfortunately in my case, it happens from time to time that the wireless signal drops completely. The only way to restore the wireless access point is to reset it or to give it a power cycle. If this happens some wireless devices loose their connection completely. In order to restore the situation a notification is essential. Sending a “ping request” does not work, as the access point’s wired interface is still online.
So, we have to monitor the wireless signal from the access points.

In my case I have 2 access points in the 2.4 GHz band and 1 acess point in the 5 GHz band.



In the picture, we see the two access points on channel 6 and channel 11. The signal on channel 11 tops at -40 dBm (and even higher) as I measured it on a distance of approx 20 cm. The second access point tops at about -75 dBm and is at the complete other end of the house.
The SSID of the wireless network is “Jonas”

You also see neighbouring networks and we are not interested in those networks and they do not peak over -80 dBm.

However in big cities, you will find more neighbouring networks. This might require multiple access points in order to achieve an acceptable quality level.

2. Required equipment

As said, we have to monitor the wireless signal, so we need a device, which is able to receive Wifi signals.
As I run my Domoticz on a Raspberry Pi and I also use Raspberry Pi’s for several other applications, the hardware choice in this case was a Raspberry Pi 3B+. This device is not heavely loaded with all kind of software and I use it only for some scripts. Also a Raspberry Pi Zero W would be an affordable choice.

However, if you use a device with only wireless ethernet (wlan0), such as a Raspberry Pi Zero, realize that in case the wireless access point, to which the device has connected fails, you will loose all information of all monitored access points, as no communication is possible.

For software we only need the Raspberry Pi OS and in addition the Mosquitto-clients package.

sudo apt-get install mosquitto-clients

3. Collect data

To collect the required information we have the option of two available tools for wireless ethernet, *iwlist* or *iw*.

Iwlist is used to display some additional information from a wireless network interface that is not displayed by iwconfig. Iw is the successor of iwlist and offers more possibilities.
In this case we will use iwlist as the extra information from iw is not necessarry.

If we issue the command: *sudo iwlist wlan0 scan*, where wlan0 is the wireless interface, we will get for each detected wireless network the following information:

```
Cell 02 -  Address: C0:56:27:0F:B7:77
           Channel:44
           Frequency:5.22 GHz (Channel 44)
           Quality=70/70  Signal level=-20 dBm
           Encryption key:on
           ESSID:"Jonas5"
           Bit Rates:6 Mb/s; 9 Mb/s; 12 Mb/s; 18 Mb/s; 24 Mb/s
                     36 Mb/s; 48 Mb/s; 54 Mb/s
           Mode:Master
           Extra:tsf=0000000000000000
           Extra: Last beacon: 10ms ago
```

If we issue the command: *sudo iw wlan0 scan*, where wlan0 is the wireless interface, we will receive a lot more information.

Although iwlist is the older command and may become obsolete sooner or later, we will use iwlist, as this is more than sufficient.

As we are only interested in our own network and not in neighbouring networks we will filter on our SSID (or ESSID).

Issue the command: *sudo iwlist wlan0 scan | grep ESSID:\"Jonas\" -B3*

The command has to be executed by root and therefore “sudo” is used.
wlan0 is the name of the interface and we will scan for all access points with the ESSID “Jonas”. The -B3 option displays the 3 lines before the match, so before the line
ESSID: “Jonas”.

As I have 2 access points with the ESSID “Jonas”. We will see the following:

```
        Frequency:2.462 GHz (Channel 11)
        Quality=70/70  Signal level=-10 dBm
        Encryption key:on
        ESSID:"Jonas"
--
        Frequency:2.437 GHz (Channel 6)
        Quality=52/70  Signal level=-58 dBm
        Encryption key:on
        ESSID:"Jonas"
```

As I have also an access point in the 5GHz band with the SSID “Jonas5”, I repeat the command with ESSID “Jonas5”.

```
sudo iwlist wlan0 scan | grep ESSID:\"Jonas5\" -B3
```

This will result in:

```
        Frequency:5.22 GHz (Channel 44)
        Quality=70/70  Signal level=-20 dBm
        Encryption key:on
        ESSID:"Jonas5"
```

4. Publish data to MQTT

As Domoticz and Node Red and Mosquitto MQTT server, all run on different hardware with different IP addresses we need to transfer the message with MQTT to the Mosquitto server. Therefore we need the mosquitto-clients package.

If not already done so, install this package with:

```
sudo apt-get install mosquitto-clients
```

We have to extend the previous command with a command to send the data to the MQTT server. Here comes the power of Linux. It is possible to “pipe” the output of the previous command (stdout) into the publishing command of mosquitto (stdin).

```
sudo iwlist wlan0 scan | grep ESSID:\"Jonas\" -B3|sudo mosquitto_pub -L
mqtt://username:password@192.168.10.51:1883/rpi3plus/wifi -s
```

The command consists of:
- the previous command to collect the data
- the | (pipe) symbol
- sudo mosquitto_pub (mqtt publishing command with root rights)
- -L option, specifying user, password, hostname, port and topic at once as a URL.
  The username:password is optional and only needed, if you have secured your MQTT server with a username and a password.
  The URL must be in the form: mqtt(s)://[username[:password]@]host[:port]/topic.
  The topic, you want to publish to, is up to the user and in this case rpi3plus/wifi
- -s option, which sends a message, read from stdin, sending the entire content as a single message.

As we have 2 commands, 1 for the access point on the 2.4 GHZ band and 1 for the access point on the 5 GHz band, we have to execute also the other command.

```
sudo iwlist wlan0 scan | grep ESSID:\"Jonas5\" -B3|sudo mosquitto_pub -L
mqtt://username:password@192.168.10.51:1883/rpi3plus/wifi -s
```

For easy use of both commands, we will combine them in a simple bash script, which will be called wifi_monitor.sh:

```
#!/bin/bash
sudo iwlist wlan0 scan | grep ESSID:\"Jonas\" -B3|sudo mosquitto_pub
-L mqtt://username:password@192.168.10.51:1883/rpi3plus/wifi -s
sleep 1
sudo iwlist wlan0 scan | grep ESSID:\"Jonas5\" -B3|sudo mosquitto_pub
-L mqtt://username:password@192.168.10.51:1883/rpi3plus/wifi -s
```

Both commands are seperated with a short pause of 1 second.

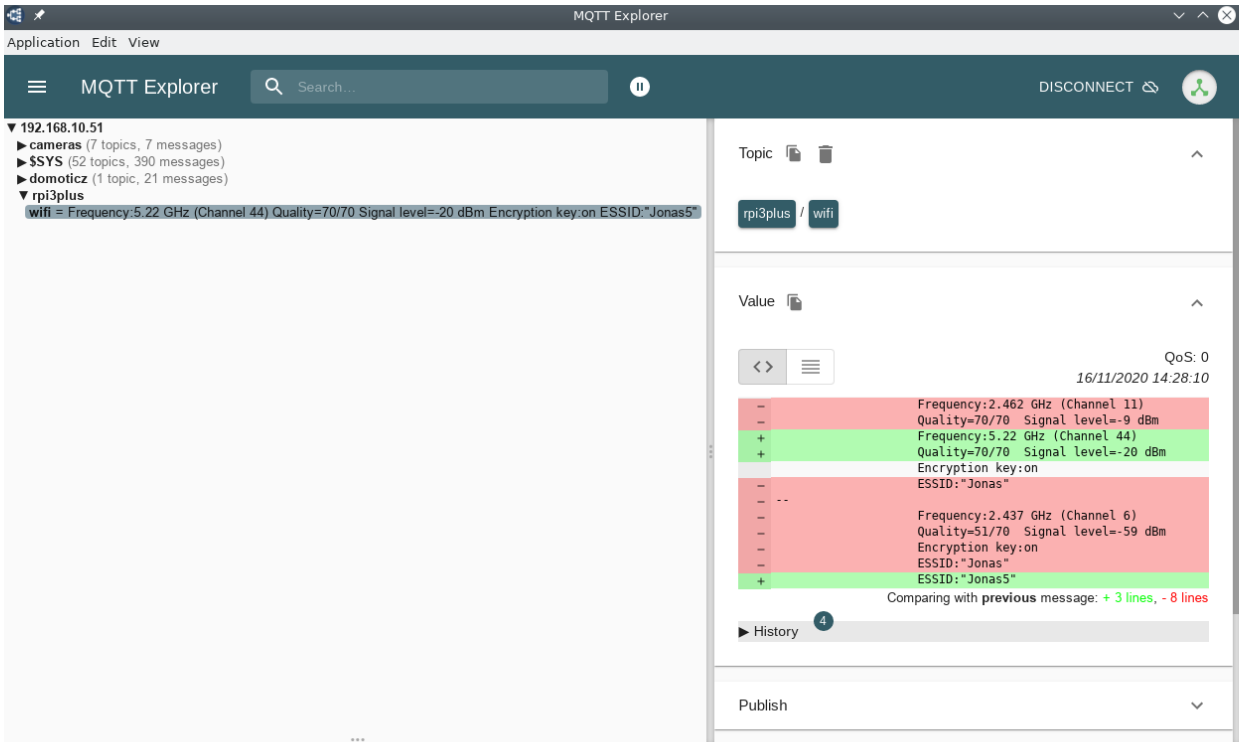
Take care that this script has root rights and have executable rights.

This script will be run every minute, using a cronjob.

Execute: `sudo crontab -e`

Enter the following: `* * * * * /bin/bash /home/pi/wifi_monitor.sh > /dev/null`

Check that the data is published to the MQTT server with e.g. MQTT Explorer.



```
▼ 192.168.10.51
  ▶ cameras (7 topics, 7 messages)
  ▶ $SYS (52 topics, 390 messages)
  ▶ domoticz (1 topic, 21 messages)
  ▼ rpi3plus
    wifi = Frequency:5.22 GHz (Channel 44) Quality=70/70 Signal level=-20 dBm Encryption key:on ESSID:"Jonas5"
```

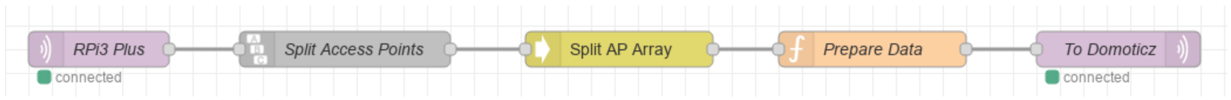
```
QoS: 0
16/11/2020 14:28:10
-                     Frequency:2.462 GHz (Channel 11)
-                     Quality=70/70  Signal level=-9 dBm
+                     Frequency:5.22 GHz (Channel 44)
+                     Quality=70/70  Signal level=-20 dBm
                      Encryption key:on
-                     ESSID:"Jonas"
-  --
-                     Frequency:2.437 GHz (Channel 6)
-                     Quality=51/70  Signal level=-59 dBm
-                     Encryption key:on
-                     ESSID:"Jonas"
+                     ESSID:"Jonas5"
Comparing with previous message: + 3 lines, - 8 lines
▶ History 4
```

You can subscribe to this topic and create a script with any scripting language.

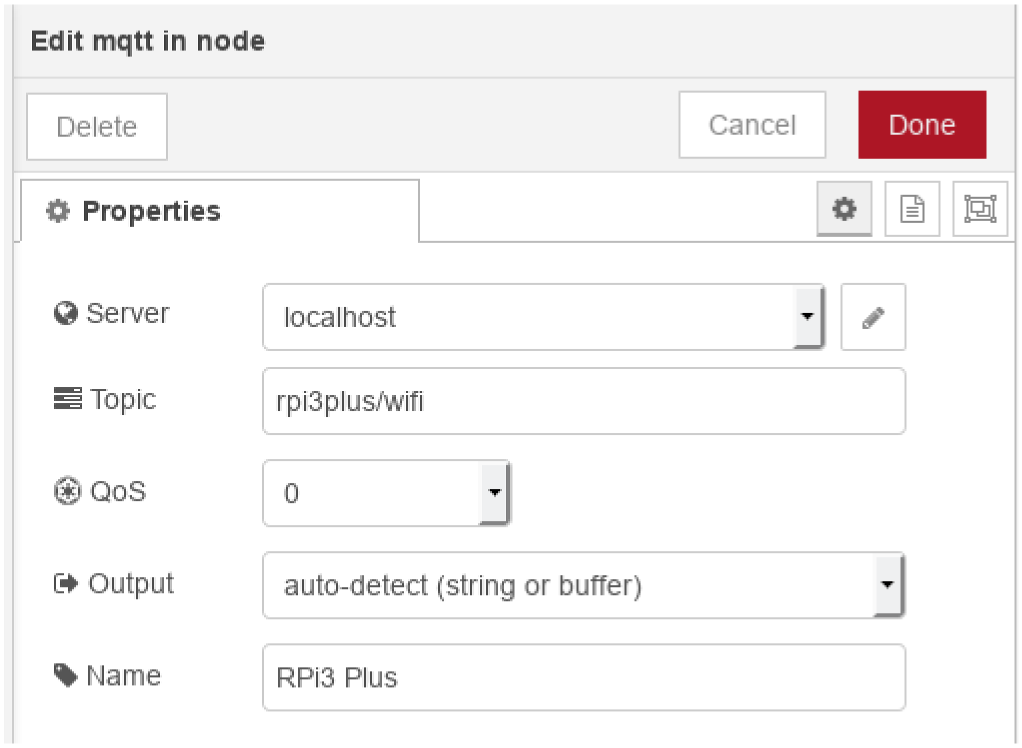
5. Node Red flow

Node Red is used to subscribe to this topic, parse the data and prepare it in order to send it to Domoticz with MQTT.

The complete flow is shown below.



The first node is a default MQTT input node.



Server: localhost

Topic: rpi3plus/wifi

QoS: 0

Output: auto-detect (string or buffer)

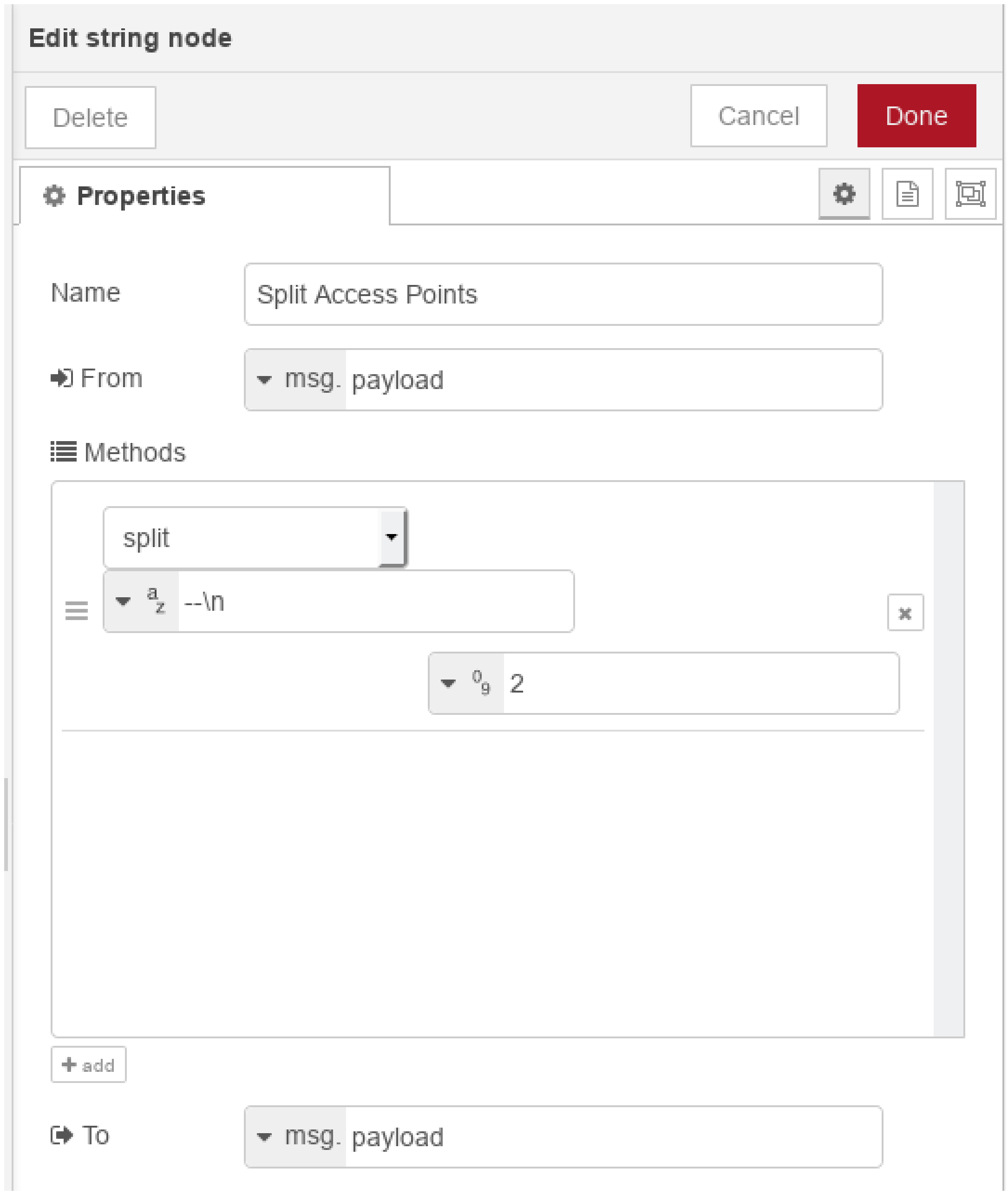
Name: RPi3 Plus

In this case the MQTT server (Mosquitto) is installed on the same device as Node Red and therefore you can use ‘localhost’ as server address. For the ‘Topic’ you fill in the topic you have selected in the previous step, so in this case ‘rpi3plus/wifi’.
The rest you can leave, as shown in the picture. The ‘Name’ is free to choose.

The next node is not a default node in Node Red, but you have to install it, using the command line or using the Palette in the menu, which is recommended.

Go to the Menu (in the rigjt upper corner) and select ‘Manage palette’. Next go to ‘Install’ and search for ‘*node-red-contrib-string*’. Install this node.

As we receive 2 messages, one with the data of 2 access points on the 2.4 GHz band and one with the data of 1 access points on the 5 GHz band, we want to split the data of the two access points on the 2.4 GHz band.
This string node has a lot of functionality, but we will use it only to split the string with data of two access points into an array, with 2 elements, one for each access point.


Edit string node

Name: Split Access Points

From: msg. payload

Methods

split

--\n

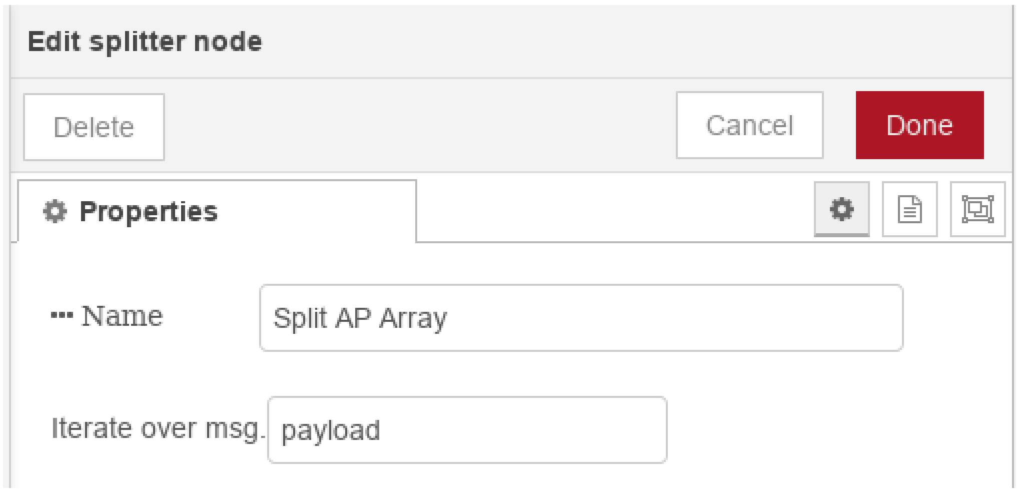
2

To: msg. payload

As we can see the data of the two access points are seperated by two dashes and a linefeed and we will use this as the string to split the data into an array with 2 elements.
If you have more that 2 access points, you have to adjust the number of splits accordingly.

The next node is a splitter node, which is also installed from the palette, as it is also a non-default node. Follow the same procedure as before and search for '*node-red-contrib-splitter*'. Install this node.

The final result will be that we will now have 3 separate messages, one for each access point in the 2.4 GHZ band and 1 for the access point in the 5 GHZ band.



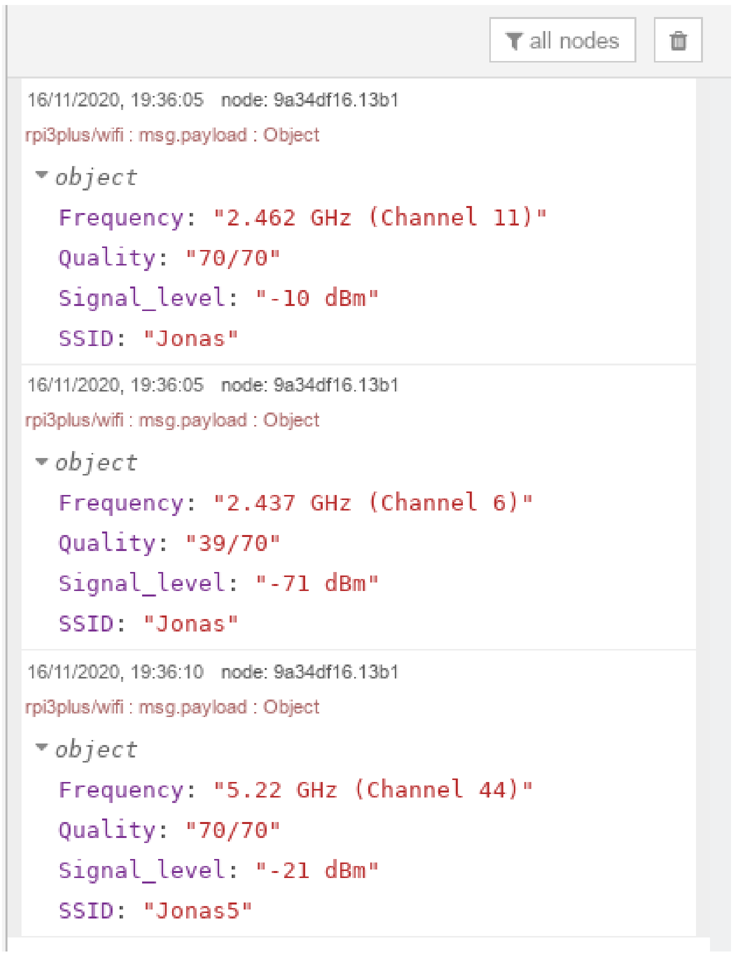
Name: Split AP Array

Iterate over msg. payload

The next node in the 'Function' node, which prepares the data for Domoticz.

This 'Function' node consists of several parts.

- Declaration of variables. (This is not strictly necessarry, but it avoids very long lines.)
- Filter for SSID, so that only in case of the two SSID's the next part is executed.
- Parsing of the payload, in order to create a nicely formatted object.



```
16/11/2020, 19:36:05   node: 9a34df16.13b1
rpi3plus/wifi : msg.payload : Object
▼object
  Frequency: "2.462 GHz (Channel 11)"
  Quality: "70/70"
  Signal_level: "-10 dBm"
  SSID: "Jonas"

16/11/2020, 19:36:05   node: 9a34df16.13b1
rpi3plus/wifi : msg.payload : Object
▼object
  Frequency: "2.437 GHz (Channel 6)"
  Quality: "39/70"
  Signal_level: "-71 dBm"
  SSID: "Jonas"

16/11/2020, 19:36:10   node: 9a34df16.13b1
rpi3plus/wifi : msg.payload : Object
▼object
  Frequency: "5.22 GHz (Channel 44)"
  Quality: "70/70"
  Signal_level: "-21 dBm"
  SSID: "Jonas5"
```

- Function to calculate the ‘Alert’ level, based on the received ‘Signal level’ in dBm.
- Setting of Domoticz idx, alert_level and alert_text in order to avoid very long lines.
- Setting output command to Domoticz.

Total contents of the ‘Function’node is as follows:

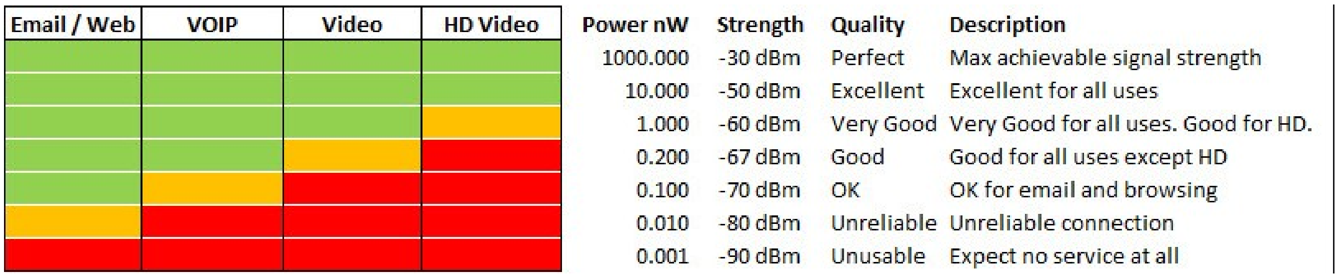
```
var alert_level;
var alert_text = {};
var idx;

if ((msg.payload.substring(msg.payload.length - 7, msg.payload.length - 2) === "Jonas")||
    (msg.payload.substring(msg.payload.length - 8, msg.payload.length - 2) === "Jonas5"))
{
    msg.payload = {
        Frequency : msg.payload.substring(msg.payload.search("Frequency") + 10,msg.payload.search("\n")),
        Quality : msg.payload.substring(msg.payload.search("Quality") + 8,msg.payload.search("Quality")+13),
        Signal_level : msg.payload.substring(msg.payload.search("Signal level") + 13,msg.payload.search("dBm")+3),
        SSID : msg.payload.substring(msg.payload.search("ESSID") + 7,msg.payload.length - 2)
   }
}

// If not used with Domoticz, remove everything below this line (except last "return msg;") and insert your own code.

// Function to convert Signal level to Alert level

function dBm2level(x) {

if (x > -30) {
  return 0;
} else if (x > -50) {
  return 1;
} else if (x > -60) {
  return 2;
} else if (x > -70) {
  return 3;
} else {
  return 4;
}
}

if ((msg.payload.SSID === "Jonas") && (msg.payload.Frequency === "2.437 GHz (Channel 6)")) { idx = 477 }
else if ((msg.payload.SSID === "Jonas") && (msg.payload.Frequency === "2.462 GHz (Channel 11)")) { idx = 476 }
else if ((msg.payload.SSID === "Jonas5") && (msg.payload.Frequency === "5.22 GHz (Channel 44)")) { idx = 478 }

alert_level = dBm2level(parseInt((msg.payload.Signal_level).substring(0,3)));
alert_text = "SSID: " + msg.payload.SSID + "\n" + "Frequency: " + msg.payload.Frequency + "\n" + "Signal level: " +
msg.payload.Signal_level + "; " + "Quality: " + msg.payload.Quality;

msg.payload = {"command":"udevice", "idx":idx, "nvalue": alert_level, "svalue":alert_text};

return msg;
```

Note 1

If this ‘Function’ node has to be used, with other home automation systems, such as OpenHAB or Home Assistant, delete everything below the line with the comment, except ‘return msg;’
The output is shown on the previous page.

Note 2.

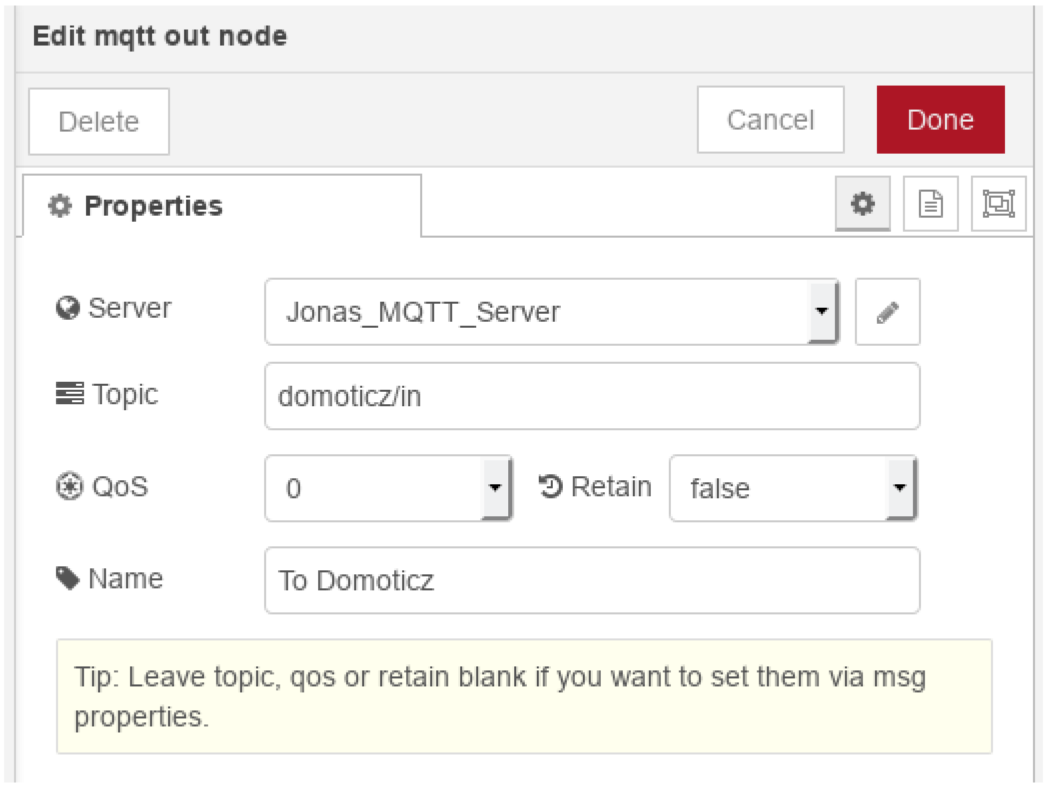
The alert level of Domoticz is based on the following Signal levels:



Email / Web	VOIP	Video	HD Video	Power nW	Strength	Quality	Description
🟩	🟩	🟩	🟩	1000.000	-30 dBm	Perfect	Max achievable signal strength
🟩	🟩	🟩	🟩	10.000	-50 dBm	Excellent	Excellent for all uses
🟩	🟩	🟩	🟧	1.000	-60 dBm	Very Good	Very Good for all uses. Good for HD.
🟩	🟩	🟧	🟥	0.200	-67 dBm	Good	Good for all uses except HD
🟩	🟧	🟥	🟥	0.100	-70 dBm	OK	OK for email and browsing
🟧	🟥	🟥	🟥	0.010	-80 dBm	Unreliable	Unreliable connection
🟥	🟥	🟥	🟥	0.001	-90 dBm	Unusable	Expect no service at all

Alert level 0 (grey)	> -30dBm
Alert level 1 (green)	> -50dBm
Alert level 2 (yellow)	> -60dBm
Alert level 3 (orange)	> -70dBm
Alert level 4 (red)	<= -70dBm

The last node is de MQTT Output node, which publish the data to Domoticz.



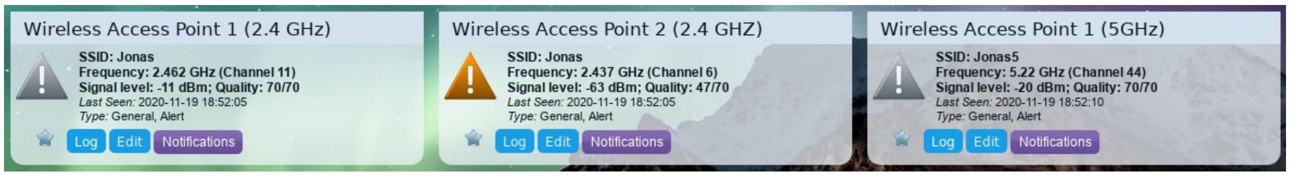
Server: Jonas_MQTT_Server
Topic: domoticz/in
QoS: 0
Retain: false
Name: To Domoticz

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

Publish the message to the MQTT server, to which Domoticz has subscribed to and publishes to. The Topic has to be ‘domoticz/in’. The ‘Name’ may be anything.

6. Domoticz

In order to get the required data into Domoticz, the user has to create virtual sensors for each access point. In this example, we will create 3 ‘Alert’ sensors.
Make a note of the Domoticz IDX numbers and insert these in the ‘Function’ node (in the lines below the function, which defines the alert levels)



Wireless Access Point 1 (2.4 GHz)
SSID: Jonas
Frequency: 2.462 GHz (Channel 11)
Signal level: -11 dBm; Quality: 70/70
Last Seen: 2020-11-19 18:52:05
Type: General, Alert

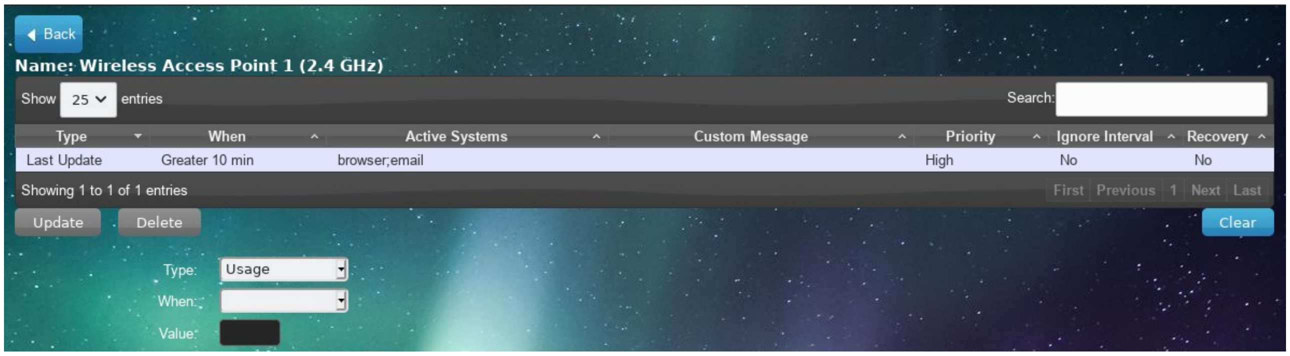
Wireless Access Point 2 (2.4 GHZ)
SSID: Jonas
Frequency: 2.437 GHz (Channel 6)
Signal level: -63 dBm; Quality: 47/70
Last Seen: 2020-11-19 18:52:05
Type: General, Alert

Wireless Access Point 1 (5GHz)
SSID: Jonas5
Frequency: 5.22 GHz (Channel 44)
Signal level: -20 dBm; Quality: 70/70
Last Seen: 2020-11-19 18:52:10
Type: General, Alert

7. Set Domoticz notification

For each sensor we will configure ‘Notifications’. Click on the ‘Notifications’button.

In this case a time-out after the last update of the sensor of 10 minutes has been selected. This has proven to be a reliable and sensible value, that does not give unwanted notifications and is fast enough to restore the wifi signal



Name: Wireless Access Point 1 (2.4 GHz)

Show 25 entries Search:

Type	When	Active Systems	Custom Message	Priority	Ignore Interval	Recovery
Last Update	Greater 10 min	browser;email		High	No	No

Showing 1 to 1 of 1 entries

Type: Usage
When:
Value:

Happy Wifi

November 2020

Annex

Please find the complete flow below:

```
[{"id":"cfe7e699.36cc9","type":"tab","label":"Wifi Monitor","disabled":false,"info":""},
{"id":"1fb062b6.c3465d","type":"mqtt in","z":"cfe7e699.36cc9","name":"RPi3
Plus","topic":"rpi3plus/wifi","qos":"0","datatype":"auto","broker":"f9f13036.e28b58","x":100,"y":140,"wires":
[["856c0cdd.45b2d"]]},{"id":"856c0cdd.45b2d","type":"string","z":"cfe7e699.36cc9","name":"Split Access
Points","methods":[{"name":"split","params":[{"type":"str","value":"--\\n"},
{"type":"num","value":"2"}]}],"prop":"payload","propout":"payload","object":"msg","objectout":"msg","x":310,"y":140,"wir
es":[["52d533ea.dc064c"]]},{"id":"52d533ea.dc064c","type":"splitter","z":"cfe7e699.36cc9","name":"Split AP
Array","property":"payload","x":544,"y":140,"wires":[["62515176.7d39"]]},
{"id":"62515176.7d39","type":"function","z":"cfe7e699.36cc9","name":"Prepare Data","func":"var alert_level;\nvar
alert_text = {};\nvar idx;\n\nif ((msg.payload.substring(msg.payload.length - 7, msg.payload.length - 2)
=== \"Jonas\")||\n    (msg.payload.substring(msg.payload.length - 8, msg.payload.length - 2) === \"Jonas5\"))\n{\n
msg.payload = {\n        Frequency : msg.payload.substring(msg.payload.search(\"Frequency\") +
10,msg.payload.search(\"\\n\")),\n        Quality : msg.payload.substring(msg.payload.search(\"Quality\") +
8,msg.payload.search(\"Quality\")+13),\n        Signal_level : msg.payload.substring(msg.payload.search(\"Signal
level\") + 13,msg.payload.search(\"dBm\")+3),\n        SSID : msg.payload.substring(msg.payload.search(\"ESSID\") +
7,msg.payload.length - 2)\n    }\n}\n\n// If not used with Domoticz, remove everything below this line (except
last \"return msg;\") and insert your own code.\n\n// Function to convert Signal level to Alert level\n\nfunction
dBm2level(x) {\n\nif (x > -30) {\n  return 0;\n} else if (x > -50) {\n  return 1;\n} else if (x > -60) {\n  return 2;\n} else if (x
> -70) {\n  return 3;\n} else {\n  return 4;\n}\n}\n\nif ((msg.payload.SSID === \"Jonas\") && (msg.payload.Frequency
=== \"2.437 GHz (Channel 6)\")) { idx = 477 }\nelse if ((msg.payload.SSID === \"Jonas\") &&
(msg.payload.Frequency === \"2.462 GHz (Channel 11)\")) { idx = 476 }\nelse if ((msg.payload.SSID === \"Jonas5\")
&& (msg.payload.Frequency === \"5.22 GHz (Channel 44)\")) { idx = 478 }\n\nalert_level =
dBm2level(parseInt((msg.payload.Signal_level).substring(0,3)));\nalert_text = \"SSID: \" + msg.payload.SSID + \"\\n\"
+ \"Frequency: \" + msg.payload.Frequency + \"\\n\" + \"Signal level: \" + msg.payload.Signal_level + \"; \"
+ \"Quality: \" + msg.payload.Quality;\n\nmsg.payload = {\"command\":\"udevice\", \"idx\":idx, \"nvalue\":
alert_level, \"svalue\":alert_text};\n\nreturn msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","x":760,"y":140,"wires":
[["ad6d367.9e50448"]]},{"id":"ad6d367.9e50448","type":"mqtt out","z":"cfe7e699.36cc9","name":"To
Domoticz","topic":"domoticz/in","qos":"0","retain":"false","broker":"8591549f.77809","x":970,"y":140,"wires":[]},
{"id":"f9f13036.e28b58","type":"mqtt-
broker","name":"localhost","broker":"127.0.0.1","port":"1883","clientid":"","usetls":false,"compatmode":true,"keepalive":
"60","cleansession":true,"birthTopic":"","birthQos":"0","birthRetain":"false","birthPayload":"","closeTopic":"","closeQos":
"0","closePayload":"","willTopic":"","willQos":"0","willPayload":""},{"id":"8591549f.77809","type":"mqtt-
broker","name":"Jonas_MQTT_Server","broker":"192.168.10.24","port":"1883","clientid":"","usetls":false,"compatmode
":true,"keepalive":"60","cleansession":true,"birthTopic":"","birthQos":"0","birthPayload":"","closeTopic":"","closeQos":"0"
,"closePayload":"","willTopic":"","willQos":"0","willPayload":""}]
```