Node-RED Smart Home Control

Contents

Before you can use this service with Alexa or Google Home you need to:

1. Create an account via https://red.cb-net.co.uk/new-user.

2. Verify your account, using the email sent following account creation

3. Link your Amazon and/ or Google account with the Node-RED Smart Home Control API

4. Define one or more devices.

5. Install required Node-RED Nodes

6. Setup Node-RED flows using your devices.

You may also need to consider whether deploying a local MQTT service is required (to act as a hub for your devices), if so follow the instructions under *Install Local MQTT Service* to get up and running.

A Raspberry Pi is the ideal "work horse" for both Node-RED and MQTT server workloads (within reason!).

Tip: If you get stuck, don't forget to review the *troubleshooting section*.

Note: Looking to migrate from another service? See *Migrating from Another Service*

Account Linking

1.1 Link your Amazon Account

Note: There are Alexa restrictions based on region/ locale, Amazon publish and update their region-specific restrictions here.

To link your Amazon account:

1. Open Alexa App on your mobile device

2. Browse to "Skills and Games"

3. Search for "Node-RED Smart Home Control"

4. Link your account!

1.2 Link your Google Account

Warning: If your locale is **not** English you currently have to request extension of the service to your Google account. Please contact node-red@cb-net.co.uk for access. Once granted access you may need to browse to https://console.actions.google.com/u/0/project/node-red-208520/simulatorcreate?isDeepLink to accept Google Terms and Conditions to use the service.

To link your Google account:

1. Open the Google Home App

2. Hit the '+' icon in the top left of the App
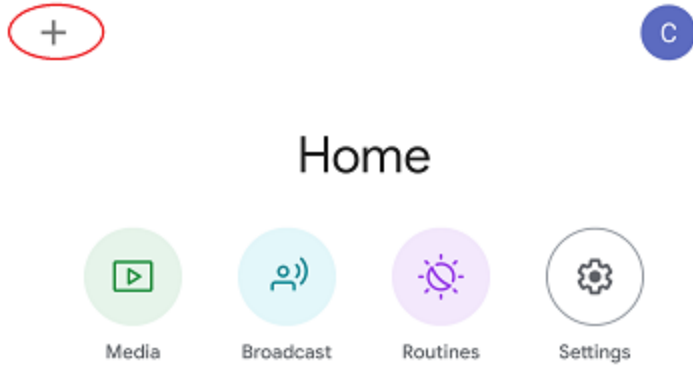
3. Go to "Set up device"
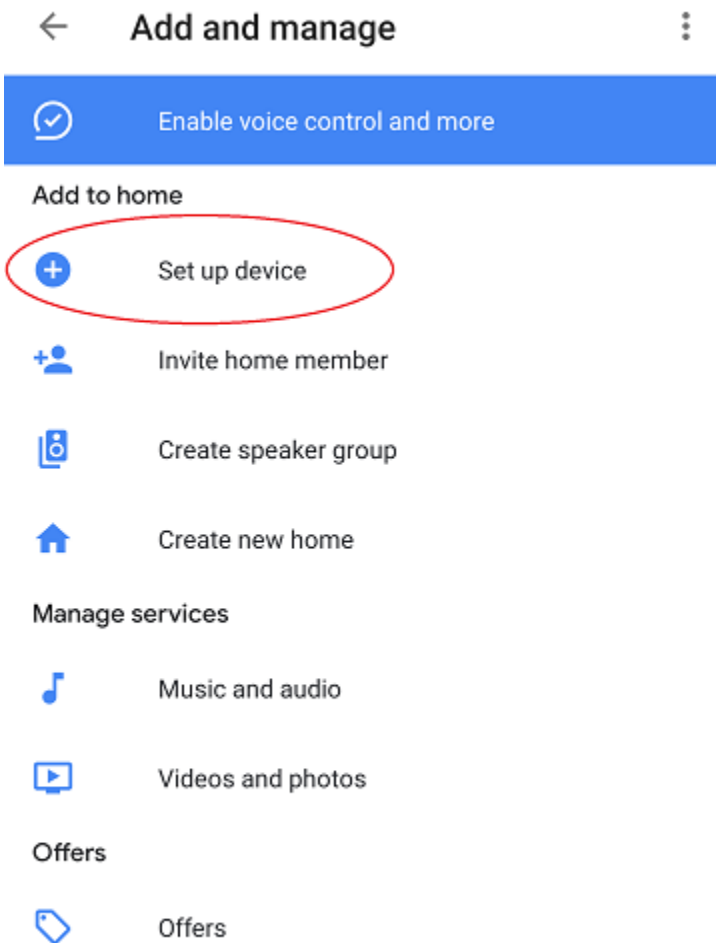
4. Under "Works with Google" hit "Have something already setup?"

5. For English locale users select "Node-RED Smart Home Control" otherwise, for non-English locale users, select "[test] Node-RED Smart Home Control" and complete the account linking process

Connect with Google

Select flag to translate:

To use this service you need to:

- Create an account.
- Define one or more devices.
- Install the associated Node-RED nodes.
- Setup Node-RED flows using your devices.

Please sign-in using your Node-RED Smart Home Control account.

By linking your account you are giving Google permission to control your devices.

Username

Note username is case sensitive.

Password

Authorise

Note: Not all device types and traits are supported by Google Home, the device creation wizard highlights which capabilities/ traits are supported. To remove the need to define Google or Alexa-specific devices the API itself will automatically only expose a devices' supported capabilities or "traits" to Google Home. You can see a comparison between the two services here.

Install Local MQTT Service

Tip: If you're looking to use MQTT-connected devices, running firmware such as Tasmota, you're going to need a local MQTT service to act as a "hub." The instructions below outline how to install Mosquitto and configure it to act as an **internal** bridge for your devices. You must ensure that the MQTT server you deploy is accessible from the network where your IoT/ MQTT enabled devices reside.

Warning: If you're only using HTTP-controlled, or other non-MQTT devices then you can skip this step.

First, install Docker CE using the commands/ process outlined *here*. If you're using a Raspberry Pi you can follow these instructions to get up and running.

Now prepare configuration/ persistent storage for Mosquitto container:

```
sudo mkdir -p /var/docker/mosquitto/config/conf.d
sudo mkdir -p /var/docker/mosquitto/data
sudo mkdir -p /var/docker/mosquitto/log
```

Create the required configuration file:

```
sudo vi /var/docker/mosquitto/config/mosquitto.conf
```

File contents should be as below:

```
pid_file /var/run/mosquitto.pid

# Configure ports
port 1883

# Block anonymous access
allow_anonymous false
```

```
# Configure persistence for retained messages
persistence true
persistence_location /mosquitto/data/

# Configure Logging
log_timestamp_format %Y-%m-%dT%H:%M:%S
log_dest file /mosquitto/log/mosquitto.log
log_dest stdout
log_type all

# Configure file-based access
password_file /mosquitto/config/pwfile

# Add /mosquitto/config/conf.d to includes
include_dir /mosquitto/config/conf.d
```

Ensure Mosquitto related file/ directory ownership is correct and create the Docker container:

```
sudo chown -R 1883:1883 /var/docker/mosquitto/config
sudo chown -R 1883:1883 /var/docker/mosquitto/data
sudo chown -R 1883:1883 /var/docker/mosquitto/log

sudo docker create --name mosquitto \
-p 1883:1883 \
-v /var/docker/mosquitto/config:/mosquitto/config \
-v /var/docker/mosquitto/data:/mosquitto/data \
-v /var/docker/mosquitto/log:/mosquitto/log \
--restart=always \
--log-opt max-size=10m \
--log-opt max-file=5 \
eclipse-mosquitto
```

Start the Mosquitto MQTT server:

```
sudo docker start mosquitto
```

Now create users, on a **per-device** basis (that way if any single device is compromised the impact will be minimised):

```
sudo docker exec -it mosquitto_passwd -b /mosquitto/config/pwfile 'username'
↪'password '
```

Tip: If you are using Tasmota, the usernames and passwords you define in the step above will be what you enter in
the device MQTT configuration, as outlined here: https://github.com/arendst/Tasmota/wiki/MQTT

Install Node-RED

If you don't already have Node-RED running in your environment I'd highly recommend using the Docker images available here: https://hub.docker.com/r/nodered/node-red

Install Docker CE using the commands/ process outlined *here*. If you're using a Raspberry Pi you can follow these instructions to get up and running.

Create the Node-RED Docker container using the following commands:

```
# Create Docker volume to enable persistent data/ config
sudo docker volume create nodered-data

# Create Node-RED Docker container
sudo docker create \
-p 1880:1880
--name="nodered" \
-v nodered-data:/data \
-e TZ=Europe/London \
--restart=always \
--log-opt max-size=10m \
--log-opt max-file=5 \
nodered/node-red
```

Start Node-RED:

```
sudo docker start nodered
```

You now have Node-RED running in your environment, browse to http://<hostname_or_IP>:1880 in order to install Nodes and configure your flows.

CHAPTER 4

Install Node-RED Nodes

Install the Node-RED Nodes by either:
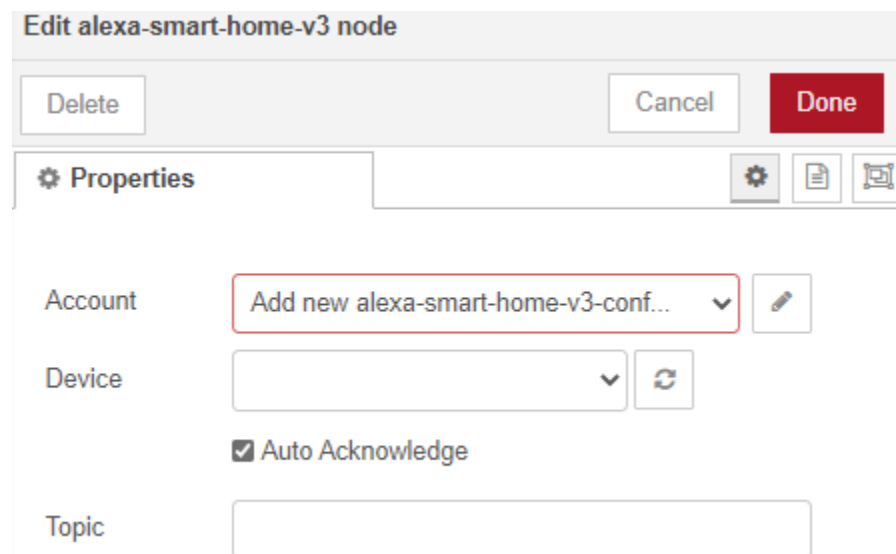
- Using the palette look for *node-red-contrib-alexa-smart-home*
- Using the command: *npm i node-red-contrib-alexa-smart-home*

Node-RED Configuration

> **Warning:** You must verify your account (via email sent on account creation) in order for the Node-RED nodes to connect.

Once Node-RED nodes are installed you'll need to configure your account settings, drag a new "alexa-smart-home-v3" node onto the canvas and double click it.

Click the edit button to define a new account:

Enter your Node-RED Smart Home Control username and password and click 'Add' to save the account details.

Edit alexa-smart-home-v3 node > **Add new alexa-smart-home-v3-conf config node**

Cancel   Add

⚙ **Properties**   ⚙ 📄

Username

Password   ⦿

MQTT
Hostname   mq-red.cb-net.co.uk

Web API
Hostname   red.cb-net.co.uk

You can now start to build flows using the concept and example flows in this documentation for inspiration.

Tip: Unless you are hosting your own instance of the API, you can leave the default "MQTT Hostname" and "Web API Hostname" fields as-is.

Warning: You only need to define your account configuration once, re-use this configuration across all of your flow.

Concept Flows

6.1 Start Simple

If you are planning to use voice control **only**, and you are not concerned about state visibility in the Alexa/ Google Home Apps, you only need:

- An "alexa-smart-home-v3" node (set to *Auto Acknowledge*)

- A receiving node for commands, such as MQTT out/ publishing that enables you to interact with the device itself

Note: Any device you chose to use this simple flow with must be configured with "Report State" **disabled**. See *Add State* if you want to benefit from state information in your Smart Assistant application(s).

You may also require a standard Node-RED function node (with your own code) to "format" command output appropriately for your chosen endpoint - examples include HTTP request, MQTT out, Yamaha AVR nodes that will likely require a specific msg format.

This basic flow is a great starting point for first-time users. You can then progress to extend the flow to enable state updates, out-of-band state updates or to perform other functions as outlined in later examples.

Warning: You should only include a single "alexa-smart-home-v3" and single "alexa-smart-home-v3-state" node per device.

6.2 Add State

Tip: Not all capabilities support state, see *here*.

Now you have basic voice commands working, let's add state updates to your flow.

If you only plan on interacting with the device using the Alexa or Google Home app, or voice assistants you can simply take state from the "alexa-smart-home-v3" node and feed it straight into the "alexa-smart-home-v3-state" node.
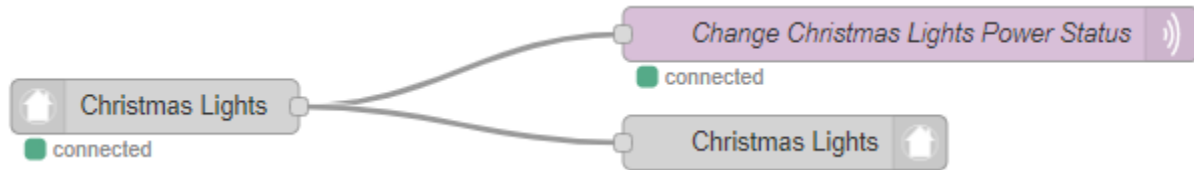
If, however, you will physically interact with the device, or it has a timer function or there are any other means for you to change its state, you will need to ensure you are sending "out of band" state updates (where the changes in state have not come from activity within the service itself) to the Node-RED Smart Home Control service.
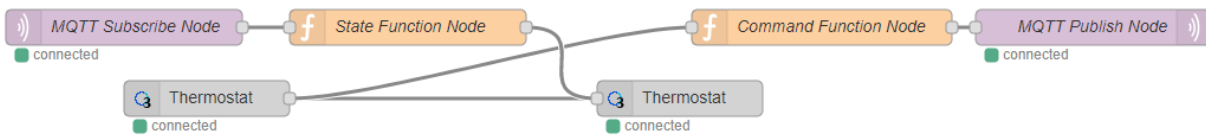
In the example above you can see a function node that has been created to intercept MQTT messages for the device and "translate" them to the required format to send back to Node-RED Smart Home Control. Example function code, for a wi-fi light switch running Tasmota firmware is shown below:

```
var tokens = msg.topic.split("/");
var device = tokens[1];
var element = tokens[2]
var state = msg.payload;

// MQTT POWER State
if (element == 'POWER') {
    return { "payload" : { "state" : { "power" : state } }, "acknowledge" : true };
}
```

Warning: When both an "alexa-smart-home-v3" and "alexa-smart-home-v3-state" node are used in a flow you must ensure that these nodes are configured for the same device.

6.3 Auto Acknowledge

By default, when you add an "alexa-smart-home-v3" node to a flow it is configured for "Auto Acknowledge," this means that a response is sent back to Node-RED Smart Home Control confirming that the command has been received, and it is **assumed** that the command was successful. This may not be desirable, depending upon the criticality of the command you have issued.

It is possible to disable "Auto Acknowledge" and use your own logic to establish whether the command was successful, before setting *msg.acknowledge* to *true* or *false* and sending the message to a *alexa-smart-home-v3-resp* node. Note that you must send the **original** message, as output from the "alexa-smart-home-v3" node, modified to include msg.acknowledge.
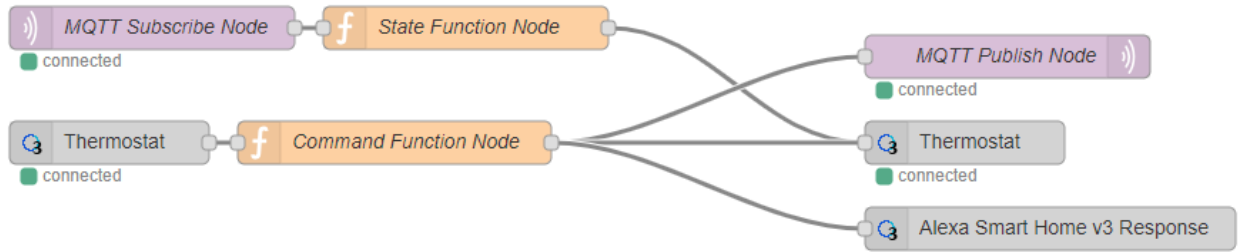
> **Warning:** This is the most advanced flow type, the majority of scenarios do not warrant/ require this level of complexity - it's just available should you want it!

Migrating from Another Service

The Node-RED nodes from other services such as https://alexa-node-red.bm.hardill.me.uk/ service and this API can co-exist, but your existing flows will need to be modified if you want them to use the "v3" service/ devices.

You are able to test new flows, using the the nodes associated with this API alongside another service, prior to moving your devices.

A typical migration path would look like:

1. Follow initial setup instructions, as-per *Getting Started*

2. Redefine your devices via https://red.cb-net.co.uk/devices - you'll need different names if co-existing with another service

3. Review Default Node Outputs - namely response nodes change to msg.acknowledge from msg.payload plus the addition on State nodes with the required function nodes to handle input from outside of the pre-defined nodes

4. Replace legacy/ V2 Nodes with nodes associated with new nodes, removing devices from the v2 service and the Alexa App

Note: These services do not share any data, therefore you must create a new account on the v3 service/ define your devices.

Example Flows

This section of the documentation includes example flows, you're welcome to submit further flows either via a pull request on GitHub, via Slack or by contacting node-red@cb-net.co.uk.

8.1 Input Controller

This flow shows a simple Input Controller example for a Yamaha MusicCast AV Receiver (using Yamaha AVR Nodes as endpoint).

Note: There is no state supported by the InputController capability.
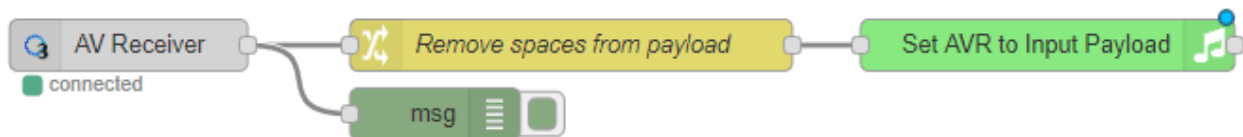
Flow code (copy and paste into Node-RED):

```
[{"id":"b2e5a71f.ce6948","type":"alexa-smart-home-v3","z":"affa0b92.8813b8","conf":
"bfd0fcf4.bc90e","device":"8","acknowledge":true,"name":"AV Receiver","topic":"","x
":130,"y":100,"wires":[["b1cb891c.e1d0a8"]]},{"id":"78b9e3f4.f9401c","type":"AVR-
Yamaha-put","z":"affa0b92.8813b8","device":"d15a41e7.73539","name":"Set AVR to
Input Payload","topic":"Main_Zone.Input.Input_Sel","payload":"","x":670,"y":100,
"wires":[[]]},{"id":"b1cb891c.e1d0a8","type":"change","z":"affa0b92.8813b8","name":
"Remove spaces from payload","rules":[{"t":"change","p":"payload","pt":"msg","from":
" ","fromt":"str","to":"","tot":"str"}],"action":"","property":"","from":"","to":"",
"reg":false,"x":390,"y":100,"wires":[["78b9e3f4.f9401c"]]},{"id":"d15a41e7.73539",
"type":"avr-yamaha","z":"","name":"Lounge AV Receiver","address":"0.0.0.0","port":"
","debug":false}]
```

Tip: For this example to work you must install/ use the Yamaha AVR Node-RED nodes.

8.2 Playback Controller

This flow shows a simple Playback Controller example for a Kodi RPC endpoint (used as a Plex client).

Note: There is no state supported by the PlaybackController capability.
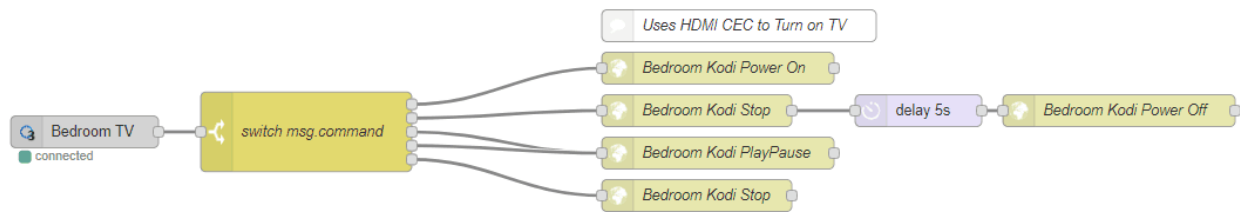
Flow code (copy and paste into Node-RED):

```
[{"id":"519d45a2.45356c","type":"http request","z":"affa0b92.8813b8","name":"Bedroom
↪Kodi Power On","method":"GET","ret":"txt","url":"http://192.169.1.100:8080/jsonrpc?
↪request={\"jsonrpc\":\"2.0\",\"method\":\"Addons.ExecuteAddon\",\"params\":{\
↪"addonid\":\"script.json-cec\",\"params\":{\"command\":\"activate\"}},\"id\":1}",
↪"tls":"","x":750,"y":660,"wires":[[]]},{"id":"f2ec1012.29ae2","type":"http request",
↪"z":"affa0b92.8813b8","name":"Bedroom Kodi Power Off","method":"GET","ret":"txt",
↪"url":"http://192.168.1.100:8080/jsonrpc?request={\"jsonrpc\":\"2.0\",\"method\":\
↪"Addons.ExecuteAddon\",\"params\":{\"addonid\":\"script.json-cec\",\"params\":{\
↪"command\":\"standby\"}},\"id\":1}","tls":"","x":1130,"y":700,"wires":[[]]},{"id":
↪"f99e0eb9.00291","type":"http request","z":"affa0b92.8813b8","name":"Bedroom Kodi
↪Stop","method":"GET","ret":"txt","url":"http://192.168.1.100:8080/jsonrpc?request={\
↪"jsonrpc\":\"2.0\",\"method\":\"Player.Stop\",\"params\":{\"playerid\":1},\"id\":1}
↪","tls":"","x":730,"y":700,"wires":[["eb1ffd69.c955b"]]},{"id":"eb1ffd69.c955b",
↪"type":"delay","z":"affa0b92.8813b8","name":"","pauseType":"delay","timeout":"5",
↪"timeoutUnits":"seconds","rate":"1","nbRateUnits":"1","rateUnits":"second",
↪"randomFirst":"1","randomLast":"5","randomUnits":"seconds","drop":false,"x":940,"y
↪":700,"wires":[["f2ec1012.29ae2"]]},{"id":"7379f6e7.023b28","type":"comment","z":
↪"affa0b92.8813b8","name":"Uses HDMI CEC to Turn on TV","info":"","x":770,"y":620,
↪"wires":[]},{"id":"a917b54d.a91138","type":"switch","z":"affa0b92.8813b8","name":
↪"switch msg.command","property":"command","propertyType":"msg","rules":[{"t":"eq","v
↪":"TurnOn","vt":"str"},{"t":"eq","v":"TurnOff","vt":"str"},{"t":"eq","v":"Pause","vt
↪":"str"},{"t":"eq","v":"Play","vt":"str"},{"t":"eq","v":"Stop","vt":"str"}],
↪"checkall":"true","repair":false,"outputs":5,"x":360,"y":720,"wires":[["519d45a2.
↪45356c"],["f99e0eb9.00291"],["393cc3e.680103c"],["393cc3e.680103c"],["42be5f30.30ed5
↪"]]},{"id":"133b1547.a2447b","type":"alexa-smart-home-v3","z":"affa0b92.8813b8",
↪"conf":"bfd0fcf4.bc90e","device":"10","acknowledge":true,"name":"Bedroom TV","topic
↪":"","x":150,"y":720,"wires":[["a917b54d.a91138"]]},{"id":"393cc3e.680103c","type":
↪"http request","z":"affa0b92.8813b8","name":"Bedroom Kodi PlayPause","method":"GET",
↪"ret":"txt","url":"http://192.168.1.100:8080/jsonrpc?request={\"jsonrpc\":\"2.0\",\
↪"method\":\"Player.PlayPause\",\"params\":{\"playerid\":1},\"id\":1}","tls":"","x
↪":750,"y":740,"wires":[[]]},{"id":"42be5f30.30ed5","type":"http request","z":
↪"affa0b92.8813b8","name":"Bedroom Kodi Stop","method":"GET","ret":"txt","url":
↪"http://192.168.1.100:8080/jsonrpc?request={\"jsonrpc\":\"2.0\",\"method\":\"Player.
↪Stop\",\"params\":{\"playerid\":1},\"id\":1}","tls":"","x":730,"y":780,"wires":[[]]}
↪]
```


Tip: For this example to work you need to install the Kodi json-cec add-on, and setup fixed/ reserved IP addresses for your Kodi devices.

8.3 Motion Sensor

Note: Google Home does not currently support motion sensors, and as a result you cannot use these devices as triggers to perform other actions.

Use the flow below to send motion sensor updates to Amazon/ Alexa - useful if you want to be able to perform actions that may not be achievable locally via NBode-RED or MQTT (for example getting Alexa to speak or interact with other Alexa-connected smart home devices and services).
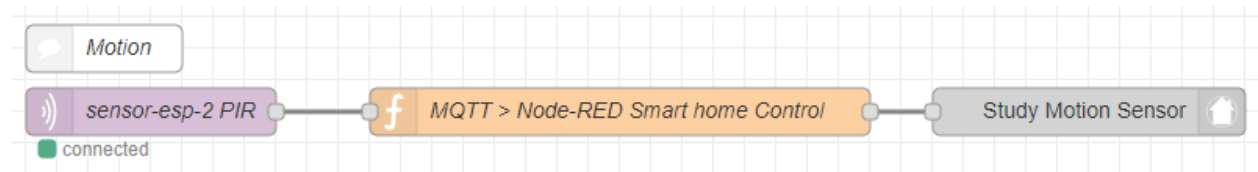
Function code needed to submit the state updates to the Node-RED Smart Home Control service, and in-turn Amazon:

```
// Motion State
if (msg.payload == "ON") {
    return { "payload" : { "state" : { "motion" : "DETECTED" } }, "acknowledge" :␣
→true };
}
else if (msg.payload == "OFF") {
    return { "payload" : { "state" : { "motion" : "NOT_DETECTED" } }, "acknowledge" :␣
→true };
}
```

Tip: In the majority of cases it will be more performant and more reliable to use your local Node-RED instance or MQTT server perform actions based upon motion sensor state changes.

Default Node Outputs

Note that outputs are consistent across Alexa and Google Home issued commands, this is intentional in order to eliminate the need to re-engineer flows/ create complex logic to manage the different command directives.

9.1 General Capabilities/ Traits

9.1.1 Percentage Control

Adjust percentage command output, used when reducing/ increasing percentage (either by a specific amount or stating "increase"/ "decrease"):

```
msg : {
    topic: ""
    name: "Test Fan"
    _messageId: "ffa95808-dc09-4c50-a242-d166acb05d1b"
    _endpointId: "104"
    _confId: "bfd0fcf4.bc90e"
    command: "AdjustPercentage"
    extraInfo: object
    payload: 25
    acknowledge: true
    _msgid: "68eadf30.4f1a4"
}
```

Tip: msg.payload will be a +/- numerical value to be used in adjustment

Set percentage command output, used when specifying a percentage, such as 25%:

```
msg : {
    topic: ""
    name: "Test Fan"
```

```
    _messageId: "6851dbbf-e826-41f9-89ee-7cd4c9699a17"
    _endpointId: "104"
    _confId: "bfd0fcf4.bc90e"
    command: "SetPercentage"
    extraInfo: object
    payload: 25
    acknowledge: true
    _msgid: "a9433270.f9ea8"
}
```

Tip: msg.payload will be a numerical value for specific percentage

9.1.2 Power Control

Set percentage command output, for "turn on" commands:

```
msg : {
    topic: ""
    name: "Study Light"
    _messageId: "0791c4b3-c874-4192-a5ac-4c4b643c36ab"
    _endpointId: "17"
    _confId: "bfd0fcf4.bc90e"
    command: "TurnOn"
    extraInfo: object
    payload: "ON"
    acknowledge: true
    _msgid: "c94c43fa.41d31"
}
```

Tip: msg.payload will be a string, either "ON" or "OFF"

9.1.3 Scene Control

On scene activation:

```
msg : {
    topic: ""
    name: "Movie Night"
    _messageId: "3b6f7aa1-38c3-45a4-a94d-96e488c6d5ad"
    _endpointId: "7"
    _confId: "bfd0fcf4.bc90e"
    command: "Activate"
    extraInfo: object
    payload: "ON"
    acknowledge: true
    _msgid: "c3f50a98.9e0b08"
}
```

Tip: msg.payload will be string, and will always be "ON"

9.2 Light-Specific Capabilities/ Traits

9.2.1 Brightness Control

Adjust Brightness command output, used when reducing/ increasing brightness (either by a specific amount or stating increase/ decrease):

```
msg : {
    topic: ""
    name: "Bedroom Light"
    _messageId: "8cbe1407-34f1-4eef-97c9-007b4b4edcfd"
    _endpointId: "29"
    _confId: "bfd0fcf4.bc90e"
    command: "AdjustBrightness"
    extraInfo: object
    payload: -25
    acknowledge: true
    _msgid: "87891d99.acdbb"
}
```

Tip: msg.payload will be a +/- numerical value to be used in adjustment

---

Set brightness command output, used when specifying a percentage, such as 80%:

```
msg : {
    topic: ""
    name: "Bedroom Light"
    _messageId: "9c289ee2-fd71-4222-ad55-8a894f70b319"
    _endpointId: "29"
    _confId: "bfd0fcf4.bc90e"
    command: "SetBrightness"
    extraInfo: object
    payload: 80
    acknowledge: true
    _msgid: "c484148c.0aa918"
}
```

Tip: msg.payload will be a numerical value for specific percentage

---

9.2.2 Color Control

Set colour command output, used when specifying a colour, such as green:

```
msg : {
    topic: ""
    name: "Test Smartlight"
```

```
    _messageId: "245ae0ea-40cb-4a44-8618-fdea822de1bf"
    _endpointId: "99"
    _confId: "bfd0fcf4.bc90e"
    command: "SetColor"
    extraInfo: object
    payload: {
        "hue": 350.5,
        "saturation": 0.7138,
        "brightness": 0.6524
        }
    acknowledge: true
    _msgid: "334fa7b2.f8d148"
}
```

Tip: msg.payload will be a JSON object containing hue, saturation and brightness values

9.2.3 Color Temperature Control

Set color temperature command output, used when specifying values either by name, or numerical value in Kelvin:

- warm || warmwhite: 2200

- incandescent || soft white: 2700

- white: 4000

- daylight || daylight white:5500

- cool || cool white: 7000

```
msg : {
    topic: ""
    name: "Bedroom Light"
    _messageId: "d506edb8-29a4-4009-9882-b17fe18e982d"
    _endpointId: "99"
    _confId: "bfd0fcf4.bc90e"
    command: "SetColorTemperature"
    extraInfo: object
    payload: 2200
    acknowledge: true
    _msgid: "47f1c84f.65f138"
}
```

Tip: msg.payload will a numerical value, representing colour temperature in Kelvin

9.3 Lock-Specific Capabilities/ Traits

Lock/ unlock command output:

```
msg : {
    topic: ""
    name: "Door Lock"
    _messageId: "5a15c0c4-1e05-4ca6-bf40-fca4393c2ec4"
    _endpointId: "128"
    _confId: "bfd0fcf4.bc90e"
    command: "Lock"
    extraInfo: object
    payload: "Lock"
    acknowledge: true
    _msgid: "7ce7f0e3.e96bd"
}
```

Tip: msg.payload will be a string, either "Lock" or "Unlock"

9.4 Media-Specific Capabilities/ Traits

9.4.1 Channel Control

Change channel command output, used when specifying a channel number, such as 101:

```
msg : {
    topic: ""
    name: "Lounge TV"
    _messageId: "01843371-f3e1-429c-9a68-199b77ffe577"
    _endpointId: "11"
    _confId: "bfd0fcf4.bc90e"
    command: "ChangeChannel"
    extraInfo: object
    payload: "101"
    acknowledge: true
    _msgid: "bd3268f0.742d98"
}
```

Tip: msg.payload will be a numerical value, representing the specific channel number

---

Command output, used when specifying a channel number, such as "BBC 1":

```
msg : {
    topic: ""
    name: "Lounge TV"
    _messageId: "c3f8fb2d-5882-491f-b0ce-7aa79eaad2fe"
    _endpointId: "11"
    _confId: "bfd0fcf4.bc90e"
    command: "ChangeChannel"
    extraInfo: object
    payload: "BBC 1"
    acknowledge: true
    _msgid: "db9cc171.e30de"
}
```

Tip: msg.payload will be a string, representing the name of the channel requested

---

Warning: Channel names are only supported by Alexa, you can only use channel numbers when using this capability/ trait with Google Assistant.

9.4.2 Input Control

Select input command output, used when specifying an input such as "HDMI 2":

```
msg : {
    topic: ""
    name: "Lounge TV"
    _messageId: "4e12b3dd-c5a0-457a-ad8b-db1799e10398"
    _endpointId: "11"
    _confId: "bfd0fcf4.bc90e"
    command: "SelectInput"
    extraInfo: object
    payload: "HDMI 2"
    acknowledge: true
    _msgid: "74f61e13.34871"
}
```

Tip: msg.payload will be a string, representing the requested input. Supported input names: HDMI1, HDMI2, HDMI3, HDMI4, phono, audio1, audio2 and "chromecast"

9.4.3 Playback Control

For playback control, msg.command changes, based upon the requested action (i.e. Play, Pause etc):

```
msg : {
    topic: ""
    name: "Lounge TV"
    _messageId: "f4379dcb-f431-4662-afdc-dc0452d313a0"
    _endpointId: "11"
    _confId: "bfd0fcf4.bc90e"
    command: "Play"
    extraInfo: object
    acknowledge: true
    _msgid: "fda4a47c.e79c08"
}
```

Tip: msg.payload will be a string, supported commands: Play, Pause, Stop, Fast Forward, Rewind, Next, Previous, Start Over

---

9.4.4 Volume Control

Tip: There are two speaker device types, a "Step Speaker" which is a "dumb" speaker that has no state and a "Speaker" which can return state (in terms of volume level).

Adjust volume command:

```
msg : {
    topic: ""
    name: "Test Speaker"
    _messageId: "77c8161c-8935-446a-9087-2ee0b9b90cdc"
    _endpointId: "98"
    _confId: "bfd0fcf4.bc90e"
    command: "AdjustVolume"
    extraInfo: object
    payload: 10
    acknowledge: true
    _msgid: "9f95ad7e.c2574"
}
```

Tip: msg.payload will be a +/- numerical value, if no value specified message msg.payload will be +/- 10

Warning: For "Step Speaker" devices, msg.payload will always be +/- 10.

Set volume command, used to set to specific value/ percentage:

```
msg : {
    topic: ""
    name: "Lounge TV"
    _messageId: "0bfd0aac-8dd1-4c8c-a341-9cfb14fa06d6"
    _endpointId: "11"
    _confId: "bfd0fcf4.bc90e"
    command: "SetVolume"
    extraInfo: object
    payload: 50
    acknowledge: true
    _msgid: "aa31e847.2da6e8"
}
```

Tip: msg.payload will be a +/- numerical value for specific percentage

Warning: "Step Speaker" volume cannot be set to a specific number.

Mute command:

```
msg : {
    topic: ""
    name: "Lounge TV"
    _messageId: "7fd278b4-1e9f-4195-9dc9-40e378a5f24b"
    _endpointId: "11"
```

(continues on next page)

```
    _confId: "bfd0fcf4.bc90e"
    command: "SetMute"
    extraInfo: object
    payload: "ON"
    acknowledge: true
    _msgid: "8fcd1348.907e1"
}
```

Tip: msg.payload will be a string, either "ON" or "OFF"

9.5 Thermostat-Specific Capabilities/ Traits

9.5.1 Adjust Temperature

Adjust the temperature through "lower," "raise," "turn up the heat" etc. commands:

```
msg : {
    topic: ""
    name: "Thermostat"
    _messageId: "3b618e03-f112-4e54-a291-62953467a1f3"
    _endpointId: "91"
    _confId: "bfd0fcf4.bc90e"
    command: "AdjustTargetTemperature"
    extraInfo: object
    payload: 1
    temperatureScale: "CELSIUS"
    acknowledge: true
    _msgid: "26950952.9183b6"
}
```

Tip: msg.payload will be +/- 1, the number to adjust the thermostat set point by

---

9.5.2 Set Target Temperature

Set target temperature:

```
msg : {
    topic: ""
    name: "Thermostat"
    _messageId: "67ebfd1b-dd16-4681-afb3-e0d0f3152865"
    _endpointId: "91"
    _confId: "bfd0fcf4.bc90e"
    command: "SetTargetTemperature"
    extraInfo: object
    payload: 22
    temperatureScale: "CELSIUS"
    acknowledge: true
    _msgid: "b8afdc95.b06fe"
}
```

Tip: msg.payload will be a numerical value, representing desired/ target temperature

---

9.5.3 Set Thermostat Mode

Available modes will depend upon device configuration within the Node-RED Smart Home Control service, as well as the physical device capabilities:

```
msg : {
    topic: ""
    name: "Thermostat"
    _messageId: "7f5b0559-f015-4e75-9443-3feac8fe6ac5"
    _endpointId: "91"
    _confId: "bfd0fcf4.bc90e"
    command: "SetThermostatMode"
    extraInfo: object
    payload: "OFF"
    acknowledge: true
    _msgid: "6a879991.5d6d38"
}
```

Tip: msg.payload will be a string, API supported modes: Auto, Eco, Heat, Cool, On, Off (support varies by smart assistant platform)

State Reporting

10.1 Capabilities that Support State

Not all capabilities/ traits support state updates, the table below illustrates those that do:

Capability	Alexa App	Google Home
BrightnessController	YES	YES
ChannelController	NO	N/A
ColorController	YES	YES*
ColorTemperatureController	YES	YES*
ContactSensor	YES	N/A
InputController	NO	N/A
LockController	YES	N/A
MotionSensor	YES	N/A
PercentageController	YES	N/A
PlaybackController	NO	N/A
PowerController	YES	YES
RangeController	YES	YES
SceneController	NO	NO
Speaker	YES	N/A
StepSpeaker	NO	N/A
TemperatureSensor	YES	N/A
ThermostatController	YES	YES

Note: Google Home support for capabilities varies by mobile platform (i.e. iOS vs Android).

10.2 Expected State Payload

State payload format is *very specific* - as a minimum you must include msg.acknowledge set to `true` and a state element update that is relevant for the device. For example, a device that has a PowerController capability can have it's state set if the following is passed to the "alexa-smart-home-v3-state" node:

```
msg : {
    "acknowledge":true,
    "payload" : {
        "state" : {
            "power": "ON"
        }
    }
}
```

> **Warning:** If you disable "Auto Acknowledge" you must set msg.acknowledge to true later in the flow, otherwise any command and state update will be dropped.

10.3 State Payload Reference

Where "‖" is listed this implies "OR" - do not include this in your state responses, they will be dropped:

```
msg {
    acknowledge: true,
    payload {
        state {
            "brightness": 0-100,
            "colorbrightness": 0-1,
            "colorHue": 0-360,
            "colorSaturation": 0-1,
            "colorTemperature": 0-10000,
            "contact": "DETECTED" || "NOT_DETECTED"
            "input": string,
            "lock": "LOCKED" || "UNLOCKED",
            "motion": "DETECTED" || "NOT_DETECTED"
            "percentage": number,
            "percentageDelta": number,
            "playback": "Play",
            "power": "ON" || "OFF",
            "range":   number (1-100 for blinds/ awnings, 1-10 for other devices),
            "temperature": number,
            "thermostatMode": "HEAT" || "COOL",
            "thermostatSetPoint" : number,
            "targetSetpointDelta": number,
            "volume": number,
            "volumeDelta": number
        }
    }
}
```

Troubleshooting

When something isn't working as expected you have a few things you can check.

11.1 Add a Node-RED Debug Node

Add a debug node after any "alexa-smart-home-v3" node, you can then verify that the command output is being received when you issue a voice command, or interact with a device using the Alexa/ Google Home applications. Ensure you configure the debug node to "output complete msg object."
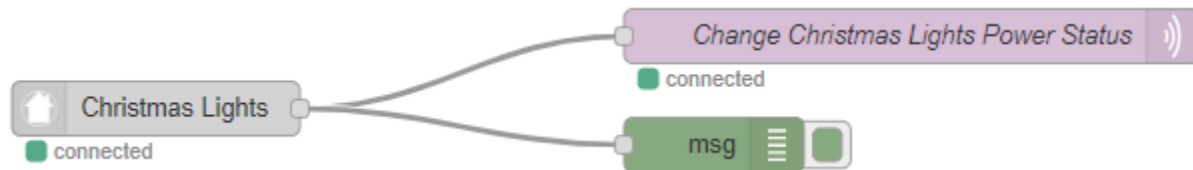
You should see output as described *here*.

11.2 Review the Node-RED Debug Console

Your next port of call is the built-in Node-RED debug console, available in the web-interface.

Node-RED Smart Home Control will send messages to your individual Node-RED instance if you send an incorrect state update/ an update that is in the wrong format. You will also be warned if your account is subject to *Throttling?*


```
1/10/2020, 4:01:43 PM   node: Christmas Lights
msg : string[131]

"Christmas Lights state node:
msg.payload.state not valid, check
data types (numbers are not strings
etc.) / format of state element"
```

11.3 Check Local MQTT Service

Tip: This is environment-specific. Most users will have a local MQTT service they have setup to act as a hub for their assisted/ smart homes.

If your environment contains a local MQTT server, such as Mosquitto, please ensure this is up and running - if down then device control will likely fail.

11.4 Check MQTT Messages

If you're not seeing any errors in the Node-RED debug console you can use "mosquitto_sub" to check for account-specific MQTT messages. This will enable you to confirm that the Node-RED Smart Home Control API is receiving your commands, at that they are available to your Node-RED Instance:

```
mosquitto_sub -h mq-red.cb-net.co.uk -t '#' -v -u <bridge_username> -P '<bridge_
→password>' --capath /etc/ssl/certs --id test-<bridge_username> -p 8883
```

If, after issuing a command via the Alexa or Google Home applications or, after using a voice command you see no output you should:

- Reset your password via the My Account page - it may be your Web API and MQTT account passwords are no longer synchronised.

Note: You'll only see messages for your account, the service uses Access Control Lists (ACLs) to filter MQTT messages.

11.5 Review Node-RED Console Log

A more "involved" approach is to look at the Node-RED console logs. The service related Nodes/ contrib output significant information to the console log. Include any output here, and from the commands/ views above if you end up raising an issue on GuitHub.

For Docker-deployed instances, this is as simple as executing the command (container name dependant):

```
sudo docker logs -f <container_name>
```

11.6 Re-link Your Account

If you are still struggling to get the service working it is definitely worth un-linking/ disabling the service via your smart assistant application and re-linking/ enabling the service.

Some long-term users of the service have been linked with a **development-only** edition of the service which can expire after 90 days of development inactivity. Since the majority of development work occurs in the API/ back-end it is highly likely that this can be a re-occurring issue.

The production version of the service is a "permanent" fixture.

Tip: Before re-linking your account it is worth clearing your account "tokens" - use the "Delete Tokens" button found on the "My Account" page.

11.7 Still Stuck?

Check out the GitHub repository for this project where you can raise questions, bugs and feature requests.

There is also a new Slack Workspace where you discuss issues with other users.

Warning: Node-RED Smart Home Control is an open source, free to use service. There is no warranty or support, implied or otherwise and the creators and contributors of this service and/ or related website are not responsible for any issues arising from it's use, including loss or damage relating to equipment, property, injury or life. You consume this service at your own risk.

11.8 Throttling?

Yes, throttling. There is an AWS Lambda function that supports this service/ any Amazon Alexa interactions. In order to limit potential costs and ensure a good service experience for users across Node-RED Smart Home Control, a rate limiter is in-place for:

- Viewing state in the Alexa Application

In day-to-day usage you are extremely unlikely to be throttled, however during testing you may trigger the rate limit against your account/ a specific device.

Note: The current rate limit is 100 requests, per device, per hour. If you exceed the defined limit you will be unable to request state data on the specific device for one hour. Commands are currently unaffected by this limit. This is subject to change at any time, without warning.

Deploy Your Own

> **Warning:** This is for advanced users/ scenarios only, with the free to use *hosted instance.* you can be up and running in just a few minutes.

12.1 Pre-Requisites

To deploy your own instance you are going to need:

1. A cloud-based Linux server (this guide assumes Ubuntu server)

2. An AWS account where you can run Lambda instances

3. An email service that supports programmatic sending/ receiving of email

4. A registered domain

5. A CloudFlare account, configured to perform DNS for your registered domain

You will require two DNS host names/ A records to be defined for the API and MQTT service: 1. Web interface/ API - where you/ your users will login and define their devices 2. MQTT service

These should be separate A records to enable caching/ security functionality via CloudFlare - you cannot route MQTT traffic through the CloudFlare security platform.

> **Tip:** You can of course choose to run your environment differently, if you will have to workout how to modify the setup instructions accordingly.

12.2 Define Service Accounts

You need to define three user accounts/ passwords:

1. MongoDB admin account

2. MongoDB account for the API to connect to the database

3. Superuser account for the API to connect with to the MQTT server/ your admin account for the Web API

Define these as environment variables to make container setup easier:

```
export MONGO_ADMIN=<username>
export MONGO_PASSWORD=<password>
export MQTT_USER=<username>
export MQTT_PASSWORD=<password>
export WEB_USER=<username>
export WEB_PASSWORD=<password>
```

These will also be copied into a .env file later in the deployment process.

> **Warning:** Once the API is setup you should clear your shell history.

12.3 Install Docker CE

For Ubuntu 18.04 follow this Digital Ocean guide.

Summarised version:

```
sudo apt install apt-transport-https ca-certificates curl software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
→bionic stable"
sudo apt update
sudo apt install docker-ce
```

> **Note:** These instructions are specifically designed for use on Ubuntu 18.04.

12.4 Create Docker Network

To isolate our application from other Docker workloads we will create a dedicated Docker network:

```
sudo docker network create nr-alexav3
```

12.5 MongoDB Container/ Account Creation

Docker image is used for mongo, with auth enabled.

API-required user accounts are created automatically via docker-entrypoint-initdb.d script, use the following commands to setup the MongoDB database (modifying the environment variables to suit):

```
sudo mkdir -p /var/docker/mongodb/docker-entrypoint-initdb.d
sudo mkdir -p /var/docker/mongodb/etc
sudo mkdir -p /var/docker/mongodb/data
cd /var/docker/mongodb/docker-entrypoint-initdb.d

sudo wget -O mongodb-accounts.sh https://gist.github.com/coldfire84/
↪93ae246f145ef09da682ee3a8e297ac8/raw/7b66fc4c4821703b85902c85b9e9a31dc875b066/
↪mongodb-accounts.sh
sudo chmod +x mongodb-accounts.sh

sudo sed -i "s|<mongo-admin-user>|$MONGO_ADMIN|g" mongodb-accounts.sh
sudo sed -i "s|<mongo-admin-password>|$MONGO_PASSWORD|g" mongodb-accounts.sh
sudo sed -i "s|<web-app-user>|$WEB_USER|g" mongodb-accounts.sh
sudo sed -i "s|<web-app-password>|$WEB_PASSWORD|g" mongodb-accounts.sh
sudo sed -i "s|<mqtt-user>|$MQTT_USER|g" mongodb-accounts.sh
sudo sed -i "s|<mqtt-password>|$MQTT_PASSWORD|g" mongodb-accounts.sh

sudo docker create \
--name mongodb -p 27017:27017 \
--network nr-alexav3 \
-e MONGO_INITDB_ROOT_USERNAME=$MONGO_ADMIN \
-e MONGO_INITDB_ROOT_PASSWORD=$MONGO_PASSWORD \
-v /var/docker/mongodb/docker-entrypoint-initdb.d/:/docker-entrypoint-initdb.d/ \
-v /var/docker/mongodb/etc/:/etc/mongo/ \
-v /var/docker/mongodb/data/:/data/db/ \
-v /var/docker/backup:/backup/ \
--log-opt max-size=100m \
--log-opt max-file=5 \
mongo

sudo docker start mongodb
```

On first launch the init script should run, creating all of the required MongoDB users, as outlined above.

The credentials defined under WEB_USER/ WEB_PASSWORD are your superuser account, required for setting up OAuth in the Web Service.

12.6 Certificates

We will use the same SSL certificate to protect the NodeJS and MQTT services. Ensure that, before running these commands, your hosting solution has HTTPS connectivity enabled.

We'll use certbot to request a free certificate for the Web App, and its integration with CloudFlare.

First, install certbot:

```
sudo add-apt-repository ppa:certbot/certbot
sudo apt-get update
sudo apt-get install python3-certbot-dns-cloudflare
```

Create cloudflare.ini file under /home/username/.secrets/cloudflare.ini:

```
# Cloudflare API credentials used by Certbot
dns_cloudflare_email = <cloudflare email address>
dns_cloudflare_api_key = <cloudflare API key>
```

Request your certificates:

```
sudo certbot certonly \
--agree-tos \
--renew-by-default \
--dns-cloudflare \
--dns-cloudflare-credentials <path to cloudflare.ini> \
--dns-cloudflare-propagation-seconds 60 \
-d <fqdn of web API> \
--email <your email address>

sudo certbot certonly \
--agree-tos \
--renew-by-default \
--dns-cloudflare \
--dns-cloudflare-credentials <path to cloudflare.ini> \
--dns-cloudflare-propagation-seconds 60 \
-d <fqdn of MQTT> \
--email <your email address>
```

Renewals will be handled automatically by certbot, but we will need to configure a script to run on renewal that sends a SIGHUP to NGINX and a restart to mosquitto. We have to restart Mosquitto as it will not reload the TLS certificate on SIGHUP, see here:

```
sudo vi /etc/letsencrypt/renewal-hooks/deploy/reload-containers.sh
```

Now paste the following contents into this script:

```
#!/bin/bash
docker kill --signal=HUP nginx
docker restart mosquitto
Finally, make this script executable:

sudo chmod +x /etc/letsencrypt/renewal-hooks/deploy/reload-containers.sh
```

12.7 Mosquitto Container

A custom mosquitto/ mosquitto-auth-plug container is used in this deployment:

```
sudo mkdir -p /var/docker/mosquitto/config/conf.d
sudo mkdir -p /var/docker/mosquitto/data
sudo mkdir -p /var/docker/mosquitto/log
sudo chown -R 1883:1883 /var/docker/mosquitto/config
sudo chown -R 1883:1883 /var/docker/mosquitto/data
sudo chown -R 1883:1883 /var/docker/mosquitto/log

cd /var/docker/mosquitto/config
sudo wget -O mosquitto.conf https://gist.githubusercontent.com/coldfire84/
→9f497c131d80763f5bd8408762581fe6/raw/e656ca5ace3a4183dfa6f7bcbcb8acb9c16c0438/
→mosquitto.conf

cd /var/docker/mosquitto/config/conf.d/
sudo wget -O node-red-alexa-smart-home-v3.conf https://gist.github.com/coldfire84/
→51eb34808e2066f866e6cc26fe481fc0/raw/88b69fd7392612d4be968501747c138e54391fe4/node-
→red-alexa-smart-home-v3.conf
```

```
export MQTT_DNS_HOSTNAME=<IP/ hostname used for SSL Certs>
export MONGO_SERVER=<mongodb container name>
export MQTT_USER=<username>
export MQTT_PASSWORD=<password>

sudo sed -i "s/<mongo-server>/$MONGO_SERVER/g" node-red-alexa-smart-home-v3.conf
sudo sed -i "s/<user>/$MQTT_USER/g" node-red-alexa-smart-home-v3.conf
sudo sed -i "s/<password>/$MQTT_PASSWORD/g" node-red-alexa-smart-home-v3.conf
sudo sed -i "s/<dns-hostname>/$MQTT_DNS_HOSTNAME/g" node-red-alexa-smart-home-v3.conf
sudo sed -i "s|/usr/local/src|/usr/local/lib|g" node-red-alexa-smart-home-v3.conf
```

Then start the container:

```
sudo docker create --name mosquitto \
--network nr-alexav3 \
-p 1883:1883 \
-p 8883:8883 \
-v /etc/letsencrypt:/etc/letsencrypt \
-v /var/docker/mosquitto/config:/mosquitto/config \
-v /var/docker/mosquitto/data:/mosquitto/data \
-v /var/docker/mosquitto/log:/mosquitto/log \
--restart=always \
--log-opt max-size=10m \
--log-opt max-file=5 \
coldfire84/mosquitto-auth:development
```

Note: A custom container is used as it includes the mosquitto-auth-plug

12.8 Redis Container

Create the required Redis server container:

```
sudo mkdir -p /var/docker/redis/data
sudo docker create --name redis \
--network nr-alexav3 \
-v /var/docker/redis/data:/data \
--restart always \
--log-opt max-size=10m \
--log-opt max-file=5 \
redis
```

Note: Redis is used by express-limiter

12.9 NodeJS WebApp Container

Now it's time to build/ deploy the Web API itself.

12.9.1 Create Google Home Graph JWT

If you planning on using Google Home integration you need to setup an account and obtain the associated JWT to send state reports to the Home Graph API:

```
sudo mkdir -p /var/docker/red
sudo vi /var/docker/red/.ghomejwt
# Copy contents from downloaded JWT, supplied by Google
sudo chmod 600 /var/docker/red/.ghomejwt
```

Tip: More information on this process here.

12.9.2 Build/ Create NodeJS Docker Container

It is currently recommended to use source to build your container:

```
cd ~
rm -rf nodejs-webapp
mkdir nodejs-webapp
cd nodejs-webapp/
git clone --single-branch -b development https://github.com/coldfire84/node-red-alexa-
→home-skill-v3-web.git .
sudo docker build -t red:0.11 -f Dockerfile .

sudo docker create --name red \
--network nr-alexav3 \
-p 3000:3000 \
-v /etc/letsencrypt:/etc/letsencrypt \
-v /var/docker/red/credentials:/root/.aws/credentials \
-v /var/docker/red/.env:/usr/src/app/.env \
-v /var/docker/red/.ghomejwt:/usr/src/app/ghomejwt.json \
--restart always \
--log-opt max-size=100m \
--log-opt max-file=5 \
red:0.11

sudo docker start red
sudo docker logs -f red
```

12.9.3 Create .env File

Copy the supplied template .env.template to a secure folder on your Docker host, i.e:

```
sudo mkdir -p /var/docker/red
sudo vi /var/docker/red/.env
# Copy contents from template and populate accordingly
sudo chmod 600 /var/docker/red/.env
```

12.10 Nginx

Create the NGINX container using the following commands:

```
sudo mkdir -p /var/docker/nginx/conf.d
sudo mkdir -p /var/docker/nginx/stream_conf.d
sudo mkdir -p /var/docker/nginx/includes
sudo mkdir -p /var/docker/nginx/www

export WEB_HOSTNAME=<external FQDN of web app>
export MQTT_DNS_HOSTNAME=<external FDQN of MQTT service>

# Get Config Files
sudo wget -O /var/docker/nginx/conf.d/default.conf https://gist.github.com/coldfire84/
↪47f90bb19a91f218717e0b7632040970/raw/65bb04af575ab637fa279faef03444f2525793db/
↪default.conf

sudo wget -O /var/docker/nginx/includes/header.conf https://gist.github.com/
↪coldfire84/47f90bb19a91f218717e0b7632040970/raw/
↪65bb04af575ab637fa279faef03444f2525793db/header.conf

sudo wget -O /var/docker/nginx/includes/letsencrypt.conf https://gist.github.com/
↪coldfire84/47f90bb19a91f218717e0b7632040970/raw/
↪65bb04af575ab637fa279faef03444f2525793db/letsencrypt.conf

sudo wget -O /var/docker/nginx/conf.d/nr-alexav3.cb-net.co.uk.conf https://gist.
↪githubusercontent.com/coldfire84/47f90bb19a91f218717e0b7632040970/raw/
↪e38df9035789676bdf13093af0ef1a7c657176af/nr-alexav3.cb-net.co.uk.conf

sudo wget -O /var/docker/nginx/includes/restrictions.conf https://gist.github.com/
↪coldfire84/47f90bb19a91f218717e0b7632040970/raw/
↪65bb04af575ab637fa279faef03444f2525793db/restrictions.conf

sudo wget -O /var/docker/nginx/includes/ssl-params.conf https://gist.github.com/
↪coldfire84/47f90bb19a91f218717e0b7632040970/raw/
↪65bb04af575ab637fa279faef03444f2525793db/ssl-params.conf

sudo wget -O /var/docker/nginx/conf.d/mq-alexav3.cb-net.co.uk.conf https://gist.
↪github.com/coldfire84/47f90bb19a91f218717e0b7632040970/raw/
↪c234985e379a08c7836282b7efaff8669368dc41/mq-alexav3.cb-net.co.uk.conf

sudo sed -i "s/<web-dns-name>/$WEB_HOSTNAME/g" /var/docker/nginx/conf.d/nr-alexav3.cb-
↪net.co.uk.conf
sudo sed -i "s/<web-dns-name>/$WEB_HOSTNAME/g" /var/docker/nginx/conf.d/mq-alexav3.cb-
↪net.co.uk.conf
sudo sed -i "s/<mq-dns-name>/$MQTT_DNS_HOSTNAME/g" /var/docker/nginx/conf.d/mq-
↪alexav3.cb-net.co.uk.conf

if [ ! -f /etc/letsencrypt/dhparams.pem ]; then
    sudo openssl dhparam -out /etc/letsencrypt/dhparams.pem 2048
fi

sudo docker create --network nr-alexav3 --name nginx -p 80:80 -p 443:443 \
-v /var/docker/nginx/conf.d/:/etc/nginx/conf.d/ \
-v /var/docker/nginx/stream_conf.d/:/etc/nginx/stream_conf.d/ \
-v /etc/letsencrypt:/etc/nginx/ssl/ \
-v /var/docker/nginx/includes:/etc/nginx/includes/ \
-v /var/docker/nginx/www/:/var/www \
--restart always \
--log-opt max-size=100m \
--log-opt max-file=5 \
nginx
```

12.11 Dynamic DNS

Depending on how/ where you deploy you may suffer from "ephemeral" IP addresses that changes on every power off/on of your cloud server(i.e. on Google Cloud Platform). You can pay for a Static IP address, or use ddclient to update CloudFlare or similar services:

```
mkdir -p /var/docker/ddclient/config

docker create \
--name=ddclient \
-v /var/docker/ddclient/config:/config \
linuxserver/ddclient

sudo vi /var/docker/ddclient/config/ddclient.conf

##
## Cloudflare (cloudflare.com)
##
daemon=300
verbose=yes
debug=yes
use=web, web=ipinfo.io/ip
ssl=yes
protocol=cloudflare
login=<cloudflare username>
password=<cloudflare global API key>
zone=<DNS zone>
<FQDN of web service>, <FQDN of MQTT service>
```

12.12 Create AWS Lambda Function

Create a new AWS Lambda function in the following regions:

```
* eu-west-1 (for European users)
* us-east-1 (for US East-coast)
* us-west-1 (for APAC users)
```

Tip: If your users are localised to a specific region you can avoid deploying Lambda functions in all three locations, however if they are not you must deploy Lambda functions as outlined above.

Upload node-red-alexa-home-skill-v3-lambda.zip from the lambda repo.

Set options as below:

```
* Runtime: Node.js 10.x
* Handler: index.handler
* From the top right of the Lambda console, copy the "ARN", i.e. arn:aws:lambda:eu-
→west-1:<number>:function:node-red-alexa-smart-home-v3-lambda - you will need this␣
→for the Alexa skill definition.
```

Finally, define an environment variable:

```
* WEB_API_HOSTNAME : set this to your web API hostname as defined in your .env file,␣
↪i.e. "red.cb-net.co.uk"
```

12.13 Create Alexa Skill

Under Build | Account Linking set:

- Authorization URI: *https://<hostname>/auth/start*
- Access Token URI: *https://<hostname>/auth/exchange*
- Client ID: is generated by system automatically on creating a new service via *https://<hostname>/admin/services* (client id starts at 1, is auto incremented)
- Gather redirect URLs from Alexa Skill config, enter with comma separation, i.e.
- Client Secret: manually generated numerical (yes, numerical only) i.e. 6600218816767991872626
- Client Authentication Scheme: Credentials in request body
- Scopes: access_devices and create_devices
- Domain List: <hostname used to publish web service>

Under Build | Permissions:

- Enable Send Alexa Events

Tip: Make note of the Alexa Client Id and Alexa Client Secret

---

Use the Client Id/ Client Secret in your .env file:

- ALEXA_CLIENTID=<skill send events client id>
- ALEXA_CLIENTSECRET=<skill send events client secret>

Note: Send Alexa Events enable the skill to send "out of band" state updates that are then reflected in the Alea App/ through voice queries.

12.14 Configure Web Service OAuth

To configure OAuth / enable account linking between Amazon and the skill:

1. Browse to *https://<hostname>/login*
2. login to the Web Service using the credentials supplied in launching the Web App container via MQTT_USER and MQTT_PASSWORD
3. Browse to *https://<hostname>/admin/services*, create a new service using the same numerical secret above
4. Domain list is comma separated, for example: *layla.amazon.com,pitangui.amazon.com,alexa.amazon.co.jp*

Tip: Ensure the domain list is comma separated with **no** spaces.

12.15 Firewall Configuration

External ports/ communication is all secured by either HTTPS or MQTT/TLS, as a result you will need to configure your external firewall as follows:

- Internet > TCP port 443 : HTTPS

- Internet > TCP port 8883 : MQTTS

Before executing these commands you need to confirm the subnet in use by the new Docker network you created. Use this command to confirm the subnet:

```
sudo docker network inspect nr-alexav3 | grep Subnet
```

The following commands will configure UFW and Docker - **be sure to change '172.18.0.0/16' to match your subnet**:

```
sudo apt-get install ufw

# Set Default Rules
sudo ufw default allow outgoing
sudo ufw default deny incoming
# Allow Management
sudo ufw allow 22
# Allow HTTP/HTTPS, we auto-rediect from HTTP>HTTPS
sudo ufw allow 443
sudo ufw allow 80
sudo ufw allow 8883
# Allow internal Docker network traffic for Redis, MQTT, MongoDB and NodeJS
sudo ufw allow from 172.18.0.0/16 to any port 3000 proto tcp
sudo ufw allow from 172.18.0.0/16 to any port 1883 proto tcp
sudo ufw allow from 172.18.0.0/16 to any port 27017 proto tcp
sudo ufw allow from 172.18.0.0/16 to any port 6397 proto tcp

# Ensure Docker/ UFW inter-op (without this UFW rules are bypassed)
sudo echo "{
\"iptables\": false
}" > /etc/docker/daemon.json
sudo sed -i -e 's/DEFAULT_FORWARD_POLICY="DROP"/DEFAULT_FORWARD_POLICY="ACCEPT"/g' /
→etc/default/ufw
sudo ufw reload
# Use ifconfig/ sudo docker networks ls to find the network id, it will start "br-"
sudo iptables -t nat -A POSTROUTING ! -o br-<network id> -s 172.18.0.0/16 -j␣
→MASQUERADE
sudo apt-get install iptables-persistent netfilter-persistent
# Save existing rules!
sudo docker restart
```

Additionally you can configure fail2ban to provide brute-force protection on your server following the instructions here.

12.16 Configure AWS Cloudwatch Logging

First, create the required Identity/ Group via the AWS IAM console:

1. Add a user: node-red-logger

2. Add a group: grp-node-red-log

---

3. Assign 'AmazonAPIGatewayPushToCloudWatchLogs' managed policy to the group.

4. Generate and Save API Key/ Secret

Now create a file that you can pass-through to docker container as /root/.aws/credentials - I use /var/docker/red/credentials in the command-line example for the container.

This file should contain:

```
[default]
aws_access_key_id = <YOUR_ACCESS_KEY_ID>
aws_secret_access_key = <YOUR_SECRET_ACCESS_KEY>
```

12.17 MongoDB Backups

Everything else is immutable, so our only real concern here is Mongodb backups.

1. Create a new S3 bucket, i.e: s3-node-red-alexa (capture access token and secret access token)

2. Create a new AWS Identity to use for access to the s3 bucket, i.e: id-backup-node-red-alexa, ensure you capture the access and secret access key.

3. Create a new Policy and attach to the new identity:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:ListBucket",
                "s3:PutObjectAcl"
            ],
            "Resource": "arn:aws:s3:::<s3-bucket-name>/*"
        }
    ]
}
```

4. Install aws cli on the host using the command: sudo snap install aws-cli –classic

5. Configure aws cli using the command: aws configure entering the access and secret access key

6. Create a new script under: ~/scripts/s3-backup-mongodb.sh:

```
#!/bin/bash

# Variables
###########################
CONTAINER="mongodb"
DATETIME=$(date +"%Y_%m_%d")
BACKUP_PATH="/var/docker/backup"
LOCAL_BACKUP_THRESHOD="7"
DROPBOX_BACKUP_THRESHOLD="28"
# Container paths to backup

# Script
```

```
############################
echo "Backing up conatiner: $CONTAINER"
echo "Using backup path for tgz storage: $BACKUP_PATH"
echo "Local backup threshold: $LOCAL_BACKUP_THRESHOLD"
echo "Remote backup threshold: $DROPBOX_BACKUP_THRESHOD"


# Perform Container Backup to tgz

# Perform Backup
CONTAINER_UPPER=$(echo $CONTAINER | awk '{print toupper($0)}')
PATH_REPLACE=$(echo $i | sed -e 's/\//-/g')
FILENAME="$DATETIME-$CONTAINER_UPPER$PATH_REPLACE.tgz"

# Use mongodump to backup database
mkdir -p /var/docker/backup/$CONTAINER_$DATETIME
docker exec -e CONTAINER=$CONTAINER -e DATETIME=$DATETIME -it mongodb mongodump --
↪host $CONTAINER:27017 --username <username> --authenticationDatabase admin --
↪password <password> --out /backup/$CONTAINER_$DATETIME
# Archive backup
tar -cvzf /var/docker/backup/$FILENAME /var/docker/backup/$CONTAINER_$DATETIME
# Remove backup files
echo "Will remove folder: /var/docker/backup/$CONTAINER_$DATETIME/"
rm -rf /var/docker/backup/$CONTAINER_$DATETIME/

# Check for backup in expected backup path
BACKUP_FILE="$BACKUP_PATH/$FILENAME"
if [[ ! -f $BACKUP_FILE ]]; then
    echo "ERROR Backup file NOT found: $BACKUP_PATH/$FILENAME"
    exit 1;
else
    echo "SUCCESS Backup file found: $BACKUP_PATH/$FILENAME"
fi

# Upload Backup to AWS S3
aws s3 cp $BACKUP_PATH/$FILENAME s3://<s3-bucket-name>/$FILENAME

# Cleanup LOCAL backup files older than Now - $LOCAL_BACKUP_THRESHOD days
THRESHOLD=$(date +"%Y_%m_%d" -d "-$LOCAL_BACKUP_THRESHOD days");
for i in $BACKUP_PATH/*$PATH_REPLACE.tgz
do
    IFS='/' read -ra arrfilepath <<< "$i";
    IFS='-' read -ra arrfilename <<< "${arrfilepath[-1]}";
    if [[ ${arrfilename[0]} < $THRESHOLD ]]; then
        rm $i;
        echo "INFO Deleted aged backup: $i"
    fi
done
```

Edit root crontab using the command sudo crontab -e, adding the following line (this will trigger a weekly backup at
22:45 every Saturday):

```
45 22 * * 6 /bin/bash <path to script>/s3-backup-mongodb.sh > <path to script>/backup-
↪mongodb.log
```

Tip: Adjust the frequency of backups to suit your RPO.

Introduction

With 1500+ users, and 4500+ defined devices, available in 12 Amazon Alexa markets and English locale for Google Assistant (other markets/ locales to follow), Node-RED Smart Home Control enables you to quickly bring voice-control to your Node-RED flows, using Amazon Alexa and/ or Google Assistant.

You can support the ongoing development and hosting costs of this service via PayPal or alternatively through the GitHub Sponsors programme.

> **Warning:** Node-RED Smart Home Control is an open source, free to use service. There is no warranty or support, implied or otherwise and the creators and contributors of this service and/ or related website are not responsible for any issues arising from it's use, including loss or damage relating to equipment, property, injury or life. You consume this service at your own risk.

13.1 Key Features

- Amazon Alexa and Google Assistant support, either enabled individually or in parallel.

- Support for a large array of device types including blinds, smart plugs, lights, thermostats (see more here).

- Supports "out of band" state updates (from physical or other automated device interactions) whilst providing real-time visibility of device state across Smart Assistant applications.

13.2 Regional Restrictions

> **Tip:** There are Alexa restrictions based on region/ locale, Amazon publish and update their region-specific restrictions here.

13.3 Supported Capabilities

Alexa Interface	Google Trait
Brightness Controller	Brightness
Channel Controller	Experimental (number only)
Color Controller	ColorSetting
Color Temperature Controller	ColorSetting
Contact Sensor	None
Input Controller	None
Lock Controller	Experimental*
Motion Sensor	None
Playback Controller	Experimental
Percentage Controller	None
Power Controller	OnOff
Range Controller	OpenClose (Support for Blinds/ Awning only)
Scene Controller	Scene
Speaker Controller	Experimental
Step Speaker Controller	None
Temperature Sensor	None
Thermostat Control (Single Point)	Temperature Setting