NeoHub JSON commands

NeoHub JSON *Commands Description*

Revision:	2.5
Document:	
Date:	03 February 2016

1. Overview

This document describes the NeoHub JSON command description developed for the Heatmiser NeoHub board it is intended for Application developers who wish to communicate with the Neohub system directly.

Connection to the Hub is made on port 4242 using a TCP connection.

For current status information use the “INFO” command

Contents

1,1, Revision History.

V 2.0 Initial release
V 2.1 Zigbee channel change added
V 2.3 Neoplug added

Note to Reader

This document contains references to cooling commands, As of 15th April 2015 these commands have not yet been implemented on the hardware.

2.1 Data for each device is stored as a array called a DCB the json commands access these arrays and report the data back, any error relating to a DCB access should be taken as a failure to read the data from the target device.

2.2 Each device can be set up as a thermostat or a time clock. The Info command will return timeclock = true for time clocks.

2.3 For reading or setting on/off switching times use set_timeclock or read_timeclock for timeclocks

2.4 For reading or setting comfort levels use set_comfort_levels and read_comfort_levels for thermostats.

2. JSON commands

2.1. Syntax description

```
<device(s)> is either a device, an array of devices and/or group
names or a group name so "device1" or
["device1","device2","ourgroup"] or "ourgroup"
<group> is always a group name like "ourgroup"
anything inside <> ending with name is the name as a string "name"
<temp> can be an integer or a floating point value
```

So the basic command structure is like this

when there are no arguments

{"cmd":0}

these will not check the values at all, so {"cmd":[1,2,3,4,{"fsdfdsf":3}]} would work as well

when there is one argument
{"cmd":<arg>}

when there are more

{"cmd":[<args>]}

where devices is always one argument, for one device or group it can be just the device zone name or device number or group name, if it's more they'll need to be an array (to stay the one devices argument)

so

{"cmd":[1,"kitchen"]} for one device in a multiple argument command

{"cmd":[1,["kitchen", "bathroom"]]} for multiple argument multiple devices

such commands will only check if there are enough arguments (and the correct contents)

if above the one argument is the devices it would be {"cmd":"kitchen"} or
{"cmd":["kitchen":"bathroom"]}, single device arrays are also allowed {"cmd":["kitchen"]}

2.2. Stat related commands

```
AWAY_OFF
{"AWAY_OFF":<device(s)>}

Possible results
{"result":"away off"}
{"error":"Could not complete away off"}
{"error":"Invalid argument to AWAY_OFF, should be a valid device or
array of valid devices"}

AWAY_ON
{"AWAY_ON":<device(s)>}

Possible results
{"result":"away on"}
{"error":"Could not complete away on"}
{"error":"Invalid argument to AWAY_OFF, should be a valid device or
array of valid devices"}

BOOST_OFF
{"BOOST_OFF":[{"hours":0,"minutes":10},<devices>]}

Possible results
{"result":"boost off"}"
{"error":"BOOST_OFF failed"}
{"error":"BOOST_OFF arguments not in an array"}
{"error":"Invalid first argument to BOOST_OFF, should be object"}
{"error":"Invalid second argument to BOOST_OFF, should be a valid
device or array of valid devices"}

BOOST_ON
{"BOOST_OFF":[{"hours":0,"minutes":10},<devices>]]}

Possible results
{"result":"boost on"}
{"error":"BOOST_ON failed"}
{"error":"BOOST_ON arguments not in an array"}
{"error":"Invalid first argument to BOOST_ON, should be object"}
{"error":"Invalid second argument to BOOST_ON, should be a valid
device or array of valid devices"}


COOL
{"COOL":<device(s)>}

Possible results
{"result":"Cool mode"}
{"error":"Could not complete Cool mode"}
{"error":"Invalid argument to COOL, should be a valid device or
array of valid devices"}

COOL_DEFAULT
{"COOL_DEFAULT":[<temp>, <device(s)>]}

Possible results
```

```
{"result":"default cooling temperature was set"}
{"error":"failed to set default cooling temperature"}
{"error":"COOL_DEFAULT arguments not in an array"}
{"error":"Invalid first argument to COOL_DEFAULT, should be
integer"}
{"error":"Invalid second argument to COOL_DEFAULT, should be a valid
device or array of valid devices"}

FROST_OFF
{"FROST_OFF":<device(s)>}

Possible results
{"result":"frost off"}
{"error":"Could not complete frost off"}
{"error":"Invalid argument to FROST_OFF, should be a valid device or
array of valid devices"}

FROST_ON
{"FROST_ON":<device(s)>}

Possible results
{"result":"frost on"}
{"error":"Could not complete frost on"}
{"error":"Invalid argument to FROST_ON, should be a valid device or
array of valid devices"}


HEAT
{"HEAT":<device(s)>}

Possible results
{"result":"Heat mode"}
{"error":"Could not complete HEAT"}
{"error":"Invalid argument to HEAT, should be a valid device or
array of valid devices"}


LOCK
{"LOCK":[[<pin1>,<pin2>,<pin3>,<pin4>], <device(s)>]}

Possible results
{"result":"locked"}
{"error":"lock failed1"} (setting pin failed)
{"error":"lock failed2"} (pin was set but locking failed)
{"error":"LOCK arguments not in an array"}
{"error":"LOCK first (pin) argument not in an array"}
{"error":"Invalid argument to LOCK, should be array of 4 integers
(0-9)"}
{"error":"Invalid second argument to LOCK, should be a valid device
or array of valid devices"}
```

```
UNLOCK
{"UNLOCK":<device(s)>}

Possible results
{"result":"unlocked"}
{"error":"unlock failed"}
{"error":"Invalid argument to UNLOCK, should be a valid device or
array of valid devices"}

SET_COOL_TEMP
{"SET_COOL_TEMP":[<temp>, <device(s)>]};

Possible results
{"result":"temperature was set"}
{"error":"setting temperature failed"}
{"error":"SET_COOL_TEMP arguments not in an array"}
{"error":"Invalid first argument to SET_COOL_TEMP, should be
integer"}
{"error":"Invalid second argument to SET_COOL_TEMP, should be a
valid device or array of valid devices"}


SET_DELAY
{"SET_DELAY":[<delay>, <device(s)>]}
delay is an integer

Possible results
{"result":"delay was set"}
{"error":"set delay failed"}
{"error":"SET_DELAY arguments not in an array"}
{"error":"Invalid first argument to SET_DELAY, should be integer"}
{"error":"Invalid second argument to SET_DELAY, should be a valid
device or array of valid devices"}


SET_DIFF
{"SET_DIFF":[<switching differential>, <device(s)>]}

Possible results
{"result":"switching differential was set"}
{"error":"switching differential failed"}
{"error":"SET_DIFF arguments not in an array"}
{"error":"Invalid first argument to SET_DIFF, should be integer"}
{"error":"Invalid second argument to SET_DIFF, should be a valid
device or array of valid devices"}

SET_FLOOR
{"SET_FLOOR":[<temp>, <device(s)>]}

Possible results
{"result":"floor limit temperature was set"}
{"error":"setting floor limit temperature failed"}
{"error":"SET_FLOOR arguments not in an array"}
{"error":"Invalid first argument to SET_FLOOR, should be integer"}
{"error":"Invalid second argument to SET_FLOOR, should be a valid
device or array of valid devices"}
```

```
SET_FROST
{"SET_FROST":[<temp>, <device(s)>]}

Possible results
{"result":"temperature was set"}
{"error":"set frost failed"}
{"error":"SET_FROST arguments not in an array"}
{"error":"Invalid first argument to SET_FROST, should be integer"}
{"error":"Invalid second argument to SET_FROST, should be a valid
device or array of valid devices"}


SET_PREHEAT
{"SET_PREHEAT":[<temp>, <device(s)>]}

Possible results
{"result":"max preheat was set"}
{"error":"setting max preheat failed"}
{"error":"SET_PREHEAT arguments not in an array"}
{"error":"Invalid first argument to SET_PREHEAT, should be integer"}
{"error":"Invalid second argument to SET_PREHEAT, should be a valid
device or array of valid devices"}


SET_TEMP
{"SET_TEMP":[<temp>, <device(s)>]}

Possible results
{"result":"temperature was set"}
{"error":"setting temperature failed"}
{"error":"SET_TEMP arguments not in an array"}
{"error":"Invalid first argument to SET_TEMP, should be integer or
float"}
{"error":"Invalid second argument to SET_TEMP, should be a valid
device or array of valid devices"}


READ_COMFORT_LEVELS
{"READ_COMFORT_LEVELS":<device(s)>}

Possible results
an object with keys named after the requested devices, the values
will be objects with weekday names as keys, of which the value will
be yet another object with keys "wake", "leave", "return" and
"sleep", the values there will be an array with the time and the
temperature for that period, example below:

{"kitchen":{"friday":{"leave":["09:00",16],"return":["16:00",21],"sl
eep":["22:00",16],"wake":["07:00",21]},"monday":{"leave":["02:00",9]
,"return":["22:00",20],"sleep":["23:45",8],"wake":["01:00",10]},"sat
urday":{"leave":["22:00",16],"return":["24:00",21],"sleep":["24:00",
16],"wake":["09:00",21]},"sunday":{"leave":["02:00",9],"return":["22
:00",20],"sleep":["23:45",8],"wake":["01:00",10]},"thursday":{"leave
":["09:00",16],"return":["16:00",21],"sleep":["22:00",16],"wake":["0
7:00",21]},"tuesday":{"leave":["09:00",16],"return":["16:00",21],"sl
eep":["22:00",16],"wake":["07:00",21]},"wednesday":{"leave":["09:00"
,16],"return":["16:00",21],"sleep":["22:00",16],"wake":["07:00",21]}
}}
{"error":"Could not read dcb"}
{"error":"Invalid device in READ_COMFORT_LEVELS"}
```

SET_COMFORT_LEVELS

```
{"SET_COMFORT_LEVELS":[<levels>, <device(s)>]}
```

<levels> is an object with comfort levels as above in READ_COMFORT_LEVELS

```
{"SET_COMFORT_LEVELS":[{"friday":{"leave":["09:00",16],"return":["16
:00",21],"sleep":["22:00",16],"wake":["07:00",21]},"monday":{"leave"
:["02:00",9],"return":["22:00",20],"sleep":["23:45",8],"wake":["01:0
0",10]},"saturday":{"leave":["22:00",16],"return":["24:00",21],"slee
p":["24:00",16],"wake":["09:00",21]},"sunday":{"leave":["02:00",9],"
return":["22:00",20],"sleep":["23:45",8],"wake":["01:00",10]},"thurs
day":{"leave":["09:00",16],"return":["16:00",21],"sleep":["22:00",16
],"wake":["07:00",21]},"tuesday":{"leave":["09:00",16],"return":["16
:00",21],"sleep":["22:00",16],"wake":["07:00",21]},"wednesday":{"lea
ve":["09:00",16],"return":["16:00",21],"sleep":["22:00",16],"wake":[
"07:00",21]}}, ["bedrooms", "kitchen"]]}
```

Possible results

```
{"result":"comfort levels set"}
{"error":"setting comfort levels failed"}
{"error":"SET_COMFORT_LEVELS arguments not in an array"}
{"error":"Invalid first argument to SET_COMFORT_LEVELS, should be
object"}
{"error":"Invalid first argument to SET_COMFORT_LEVELS, missing
<field>""}
```

where <field> can be "wake", "leave", "return" or "sleep"

```
{"error":"nothing was done (no dcbs or valid devices?)"}
{"error":"Invalid second argument to SET_COMFORT_LEVELS, should be a
valid device or array of valid devices"}
```

READ_TIMECLOCK

```
{"READ_TIMECLOCK":<device(s)>}
```

Possible results

an object with keys named after the requested devices, the values will be objects with weekday names as keys, of which the value will be yet another object with keys "time1", "time2", "time3" and "time4", the values there will be an array with the start and end time for that period:

```
{"kitchen":{"friday":{"time1":["07:00","09:00"],"time2":["16:00","20
:00"],"time3":["24:00","00:00"],"time4":["24:00","00:00"]},"monday":
{"time1":["01:00","02:00"],"time2":["03:00","04:00"],"time3":["05:00
","06:01"],"time4":["12:34","15:59"]},"saturday":{"time1":["07:00","
09:00"],"time2":["16:00","20:00"],"time3":["24:00","00:00"],"time4":
["24:00","00:00"]},"sunday":{"time1":["01:00","02:00"],"time2":["03:
00","04:00"],"time3":["05:00","06:01"],"time4":["12:34","15:59"]},"t
hursday":{"time1":["07:00","09:00"],"time2":["16:00","20:00"],"time3
":["24:00","00:00"],"time4":["24:00","00:00"]},"tuesday":{"time1":["
01:00","02:00"],"time2":["03:00","04:00"],"time3":["05:00","06:01"],
"time4":["12:34","15:59"]},"wednesday":{"time1":["01:00","02:00"],"t
ime2":["03:00","04:00"],"time3":["05:00","06:01"],"time4":["12:34","
15:59"]}}}
{"error":"Could not read dcb"}
{"error":"Invalid device in READ_TIMECLOCK"}
{"error":"Invalid argument to READ_TIMECLOCK, should be a valid
device or array of valid devices"}

SET_TIMECLOCK
{"SET_TIMECLOCK":[<levels>, <device(s)>]}
For <levels> see READ_TIMECLOCK result
{"SET_TIMECLOCK":[{"friday":{"time1":["07:00","09:00"],"time2":["16:
00","20:00"],"time3":["24:00","00:00"],"time4":["24:00","00:00"]},"m
onday":{"time1":["01:00","02:00"],"time2":["03:00","04:00"],"time3":
["05:00","06:01"],"time4":["12:34","15:59"]},"saturday":{"time1":["0
7:00","09:00"],"time2":["16:00","20:00"],"time3":["24:00","00:00"],"
time4":["24:00","00:00"]},"sunday":{"time1":["01:00","02:00"],"time2
":["03:00","04:00"],"time3":["05:00","06:01"],"time4":["12:34","15:5
9"]},"thursday":{"time1":["07:00","09:00"],"time2":["16:00","20:00"]
,"time3":["24:00","00:00"],"time4":["24:00","00:00"]},"tuesday":{"ti
me1":["01:00","02:00"],"time2":["03:00","04:00"],"time3":["05:00","0
6:01"],"time4":["12:34","15:59"]},"wednesday":{"time1":["01:00","02:
00"],"time2":["03:00","04:00"],"time3":["05:00","06:01"],"time4":["1
2:34","15:59"]}},"kitchen"]}

Possible results
{"result":"time clocks were set"}
{"error":"setting time clocks failed"}
{"error":"SET_TIMECLOCK arguments not in an array"}
{"error":"Invalid first argument to SET_TIMECLOCK, should be
object"}
{"error":"Invalid second argument to SET_TIMECLOCK, should be a
valid device or array of valid devices"}


SUMMER_OFF
{"SUMMER_OFF":<device(s)>}
Possible results
{"result":"summer off"}
{"error":"Could not complete SUMMER_OFF"}
{"error":"Invalid argument to SUMMER_OFF, should be a valid device
or array of valid devices"}


SUMMER_ON
{"SUMMER_ON":<device(s)>}

Possible results
{"result":"summer on"}
{"error":"Could not complete SUMMER_ON"}
{"error":"Invalid argument to SUMMER_ON, should be a valid device or
array of valid devices"}
```

```
USER_LIMIT
{"USER_LIMIT":[<int>, <device(s)>]}

Possible results
{"result":"user limit set"}
{"error":"setting user limit failed"}
{"error":"USER_LIMIT arguments not in an array"}
{"error":"Invalid first argument to USER_LIMIT, should be integer"}
{"error":"Invalid second argument to USER_LIMIT, should be a valid
device or array of valid devices"}


VIEW_ROC
{"VIEW_ROC":<device(s)>}

Possible results
{<id>:<number>, etc}
{"error":"Invalid argument to VIEW_ROC, should be a valid device or
array of valid devices"}


TIMER_ON
{"TIMER_ON":<device(s)>}
Possible results
{"result":"time clock overide on"}
{"error":"Could not complete timer on"}
{"error":"Invalid argument to TIMER_ON, should be a valid device or
array of valid devices"}

TIMER_OFF
{"TIMER_OFF":<device(s)>}
Possible results
{"result":"timers off"}
{"error":"Could not complete timer off"}
{"error":"Invalid argument to TIMER_OFF, should be a valid device or
array of valid devices"}


TIMER_HOLD_ON
{"TIMER_HOLD_ON":[<minutes>, <device(s)>]}
Minutes is a number, 0 cancels

Possible results
{"result":"timer hold on"}
{"error":"TIMER_HOLD_ON failed"}
{\"error\":\"TIMER_HOLD_ON arguments not in an array\"}
{\"error\":\"Invalid first argument to TIMER_HOLD_ON, should be
integer (minutes)\"}
{\"error\":\"Invalid second argument to TIMER_HOLD_ON, should be a
valid device or array of valid devices\"}

TIMER_HOLD_OFF
{"TIMER_HOLD_OFF":[<minutes>, <device(s)>]}
Minutes is a number, 0 cancels

Possible results
```

```
{\"result\":\"timer hold off\"}
{\"error\":\"TIMER_HOLD_OFF failed\"}
{\"error\":\"TIMER_HOLD_OFF arguments not in an array\"}
{\"error\":\"Invalid second argument to TIMER_HOLD_OFF, should be a
valid device or array of valid devices\"}
```

2.3. NeoHub related commands

2.3.1. Hold related

```
HOLD
{"HOLD":[{"temp":<number>, "id":<string>,"hours":<number>,
"minutes":<number>}, <device(s)>]}

Possible results
{"result":"temperature on hold"}
{"error":"hold failed"}
{"error":"HOLD arguments not in an array"}
{"error":"Invalid first argument to HOLD, should be object"}
{"error":"no temperature (temp field) in HOLD"}
{"error":"no id (id field) in HOLD"}
{"error":"Invalid second argument to HOLD, should be a valid device
or array of valid devices"}


CANCEL_HGROUP
{"CANCEL_HGROUP":<group>}

Possible results
{"result":"hold group cancelled"}
{"error":"cancel hold group failed"}
{"error":"Invalid argument to CANCEL_HGROUP, should be string"}
{"error":"Unknown hold group in CANCEL_HGROUP"}


CANCEL_HOLD_ALL
{"CANCEL_HOLD_ALL":0}

Possible results
{"result":"all holds canceled"}
{"result":"no known holds"}
{"error":"cancel hold group failed"}


GET_HOLD
{"GET_HOLD":0}

Possible results
{"<devicename>":<holdtime>, "<devicename2>":<holdtime2>, <etc>}
```

2.3.2. Holiday related

```
HOLIDAY
{"HOLIDAY":["HHMMSSDDMMYYYY","HHMMSSDDMMYYYY"]};


Possible results
{"result":"holiday set"}
{"error":"HOLIDAY arguments not in an array"}
{"error":"Invalid first or second argument to HOLIDAY, should be a
string [HHMMSSDDMMYYYY]"}
{"error":"Invalid third argument to HOLIDAY, should be a valid
device or array of valid devices"}
CANCEL_HOLIDAY
{"CANCEL_HOLIDAY":0}

Possible results
{"result":"holiday cancelled"}

GET_HOLIDAY
{"GET_HOLIDAY":0}

Possible results
{"start":"<starttime>", "end":"<endtime>", "ids":["<devicename1>",
"<devicename2>", etc]}
{"start":0,"end":0,"ids":[]} no holiday is set in this case
```

2.3.3. Profile related

```
STORE_PROFILE
{"STORE_PROFILE":<profile ob>}
the profile ob should have the following fields:
name : the profile's name
group : <device(s)> the group of devices the profile applies to
info : an object with levels as in SET_COMFORT_LEVELS

Possible result
{"result":"profile created"}
{"error":"Argument to STORE_PROFILE should be an object"}
{"error":"object given to STORE_PROFILE should contain a field
called "name" (profile name)"}
{"error":"object given to STORE_PROFILE should contain a field
called "info" (profile info)"}
{"error":"object given to STORE_PROFILE should contain a field
called "group" (profile group)"}


CLEAR_PROFILE
{"CLEAR_PROFILE":<profile name>}

Possible results
{"result":"profile removed"}
{"error":"Argument to CLEAR_PROFILE should be a string (profile
name)"}
```

```
GET_PROFILE
{"GET_PROFILE":<profile name>}

Possible results
{"group":<device(s)>,"info":<levels>,"name":<profile name>}
{"error":"Argument to GET_PROFILE should be an existing profile
(profile name)"}

GET_PROFILES
{"GET_PROFILES":0}

Possible results
{}
{<profile name>:<profile>, etc} <profile> as in the GET_PROFILE
command


RUN_PROFILE
{"RUN_PROFILE":<profile name>}

Possible results
{"result":"profile was run"}
{"error":"profile failed"}
{"error":"Argument to RUN_PROFILE should be a string (profile
name)"}
{"error":"Argument to RUN_PROFILE should be an exisiting profile
(profile name)"}
{"error":"nothing to do"}
```

2.3.4. Group related

```
CREATE_GROUP
{"CREATE_GROUP":[[<devices>], <name>]}

Possible results
{"result":"group created"}
{"error":"Argument to CREATE_GROUP should be an array"}
{"error":"array for CREATE_GROUP should be size 2
[devices,groupname]"}
{"error":"first argument to CREATE_GROUP should be an array of
devices"}
{"error":"second argument to CREATE_GROUP should be a string (group
name)"}

DELETE_GROUP
{"DELETE_GROUP":<group>}

Possible results
{"result":"group removed"}
{"error":"Argument to DELETE_GROUP should be a string"}
```

```
GET_GROUPS
{"GET_GROUPS":0}

Possible results
{"<groupname1>":["<devicename1>", "<devicename2>", <etc>],
"<groupname2>":[<members>], <etc>}
```

2.3.5. Zones related

```
ZONE_TITLE
{"ZONE_TITLE":[<oldname>, <newname>]}

Possible results
{"result":"zone renamed"}
{"error":"Argument to ZONE_TITLE should be an array"}
{"error":"array for ZONE_TITLE should be size 2 [oldname,newname]"}
{"error":"first argument to ZONE_TITLE should be a device"}
{"error":"second argument to ZONE_TITLE should be a string (new
device name)"}

GET_ZONES
{"GET_ZONES":0}

Possible results
{<id>:<number>,<id>:<number>,etc} the numbers are NeoHub internal
references and can be ignored, they will work as alternative for the
device names
{}

PERMIT_JOIN (adds a zone to the system).
seconds go up to 254, 255 indicated allowing joining forever, this
would be a bad idea
{"PERMIT_JOIN":[<seconds>, <name>]} <name> is the name of the device
about to be added to the NeoHub
Or to join a zigbee repeater to the network
{"PERMIT_JOIN":["repeater", <seconds>]}
This form MUST be used for repeaters as they will not communicate
with the NeoHub like thermostats.

Possible results
{"result":"network allows joining"}
{"error":"Arguments to PERMIT_JOIN should be in an array"}
{"error":"PERMIT_JOIN takes two arguments"}
{"error":"First argument to PERMIT_JOIN should be an integer"}
{"error":"Second argument to PERMIT_JOIN should be a string (new
device name)"}
{"error":"Second argument to PERMIT_JOIN repeater should be an
integer"}


REMOVE_ZONE
{"REMOVE_ZONE":<zone>}

Possible results
{"result":"zone removed"}
```

```
{"error":"Invalid argument to REMOVE_ZONE, should be a valid device
or array of valid devices"}

REMOVE_REPEATER
{"REMOVE_REPEATER":<repeater>}

Possible results
{"result":"Repeater <repeaternode> removed from network"}
{"error":"Invalid argument to REMOVE_ZONE, should be a valid
repeater name"}

Example
{"REMOVE_REPEATER":"repeaternode1234"}
{"result":"Repeater repeaternode1234 removed from network"}
```

2.3.6. rest

```
ENGINEERS_DATA
{"ENGINEERS_DATA":0}

Possible results
{<devicename>:{"CURRENT TIME HOURS":<num>, "CURRENT TIME
MINUTES":<num>, "CURRENT DAY":<num>, "SWITCHING DIFFERENTIAL":<num>,
"FROST TEMPERATURE":<num>, "OUTPUT DELAY":<num>, "USER LIMIT
(UP/DOWN LIMIT )":<num>, "FLOOR LIMIT TEMPERATURE":<num>, "MAX
PREHEAT":<num>, "COOLING SET TEMPERATURE":<num>, "COOLING DEAD
BAND":<num>, "RESERVED":<num>, "FAN SPEED":<num>, "FAIL SAFE
ENABLED":<num>, "RADIATORS OR UNDERFLOOR":<num>,"RATE OF
CHANGE":<num>, "TARGET DEVICE BOARD":<num>, "TARGET
ZONE":<num>},<devicename2>:<data>, <etc>}
{}

FIRMWARE
{"FIRMWARE":0}

Possible results
{"firmware version":"<version>"}

GET_HOURSRUN
{"GET_HOURSRUN":<device(s)>}

Possible results
{"day:1":{"box":0,"other":0},"day:2":{"box":0,"other":0},"day:3":{"b
ox":0,"other":0},"day:4":{"box":0,"other":0},"day:5":{"box":0,"other
":0},"day:6":{"box":0,"other":0},"day:7":{"box":0,"other":0},"today"
:{"box":3,"other":3}}
{"error":"Invalid argument to GET_HOURSRUN, should be a valid device
or array of valid devices"}
```

```
INFO
{"INFO":0}

Possible results
{} or if there are devices info returns an array of objects (one for
each stat) as the value of the "devices" field, each object contains
the following fields:


"AWAY"
"COOLING"
"COOLING_ENABLED"
"COOLING_TEMPERATURE_IN_WHOLE_DEGREES"
"COOL_INP"
"COUNT_DOWN_TIME"
"CRADLE_PAIRED_TO_REMOTE_SENSOR"
"CRADLE_PAIRED_TO_STAT"
"CURRENT_FLOOR_TEMPERATURE"
"CURRENT_SET_TEMPERATURE
"CURRENT_TEMPERATURE
"DEMAND"
"DEVICE_TYPE"
"ENABLE_BOILER"
"ENABLE_COOLING"
"ENABLE_PUMP"
"ENABLE_VALVE"
"ENABLE_ZONE"
"FAILSAFE_STATE"
"FAIL_SAFE_ENABLED"
"FLOOR_LIMIT"
"FULL/PARTIAL_LOCK_AVAILABLE"
"HEAT/COOL_MODE"
"HEATING"
"HOLD_TEMPERATURE"
"HOLD_TIME"
"HOLIDAY"
"HOLIDAY_DAYS"
"HUMIDITY”
"LOCK"
"LOCK_PIN_NUMBER"
"LOW_BATTERY"
"MAX_TEMPERATURE"
"MIN_TEMPERATURE"
"MODULATION_LEVEL"
"NEXT_ON_TIME"
"OFFLINE"
"OUPUT_DELAY"
"OUTPUT_DELAY"
"PREHEAT"
"PREHEAT_TIME"
"PROGRAM_MODE"
"PUMP_DELAY"
"RADIATORS_OR_UNDERFLOOR"
"SENSOR_SELECTION"
"SET_COUNTDOWN_TIME"
"STANDBY"                                        ,
"STAT_MODE"
"MANUAL_OFF"
"THERMOSTAT"
"TEMPERATURE_FORMAT"
"TEMP_HOLD"
"TIMECLOCK_MODE"
"TIMER"
"TIME_CLOCK_OVERIDE_BIT"
"VERSION_NUMBER"
"WRITE_COUNT"
"ZONE_1PAIRED_TO_MULTILINK"
"ZONE_1_OR_2"
"ZONE_2_PAIRED_TO_MULTILINK"
```

```
GET_TEMPLOG
{"GET_TEMPLOG":<device(s)>}

Possible results
{"day:1":{<id>:[<96 temperatures for that day>], etc},"day:2":<same
as day1>, "today":<todays values>}
{"error":"Invalid argument to GET_TEMPLOG, should be a valid device
or array of valid devices"}
```

STATISTICS
{"STATISTICS":0}

For Neohub version 122 this function is not working. We will fix it asap

Possible results
{"houserecords":{"highestmonthtemp":"25.3","highestweektemp":"25.3","highestyeartemp":"25.3","lowestmonthtemp":"0.0","lowestweektemp":"25.3","lowestyeartemp":"0.0"}

highest and lowest temperatures seen by the NeoHub in the last week/month/year

,"preheatrecords":{"monthrun":"kitchen","weekrun":"kitchen","yearrun":"kitchen"},

device with the longest seen preheat sequence in the last week/month/year.

"preheatstarts":
{"kitchen":{"12345677":12345677,"12345700":12345700}},

times per stat when preheat started (in the last 7 days), would have looked likethe numbers are in seconds since 1 jan 1970 (unix time stamp).

"roomrecords":{"kitchen":{"highestmonthtemp":"25.3","highestweektemp":"25.3","highestyeartemp":"25.3","lowestmonthtemp":"0.0","lowestweektemp":"25.3","lowestyeartemp":"0.0"}},

temperature high/low records per zone.

"runrecords":{"monthrun":"kitchen","weekrun":"kitchen","yearrun":"kitchen"}

device with the longest heating run in the last month/week/year

}

SET_FORMAT
{"SET_FORMAT":<format>}
format is a string, "NONPROGRAMMABLE", "7DAY", "5DAY/2DAY", or "24HOURSFIXED"

Possible results
{"result":"Format was set"}
{"error":"setting format failed"}
{"error":"Invalid argument to SET_FORMAT, should be string"}
{"error":"Invalid argument to SET_FORMAT, unknown format (7DAY|5DAY/2DAY|24HOURSFIXED|NONPROGRAMMABLE"}

```
SET_TEMP_FORMAT
{"SET_TEMP_FORMAT":<format>}
format is "C" or "F"

Possible results
{"result":"Temperature format set to F"}
{"result":"Temperature format set to C"}
{"error":"Invalid argument to SET_TEMP_FORMAT, should be "C" or "F"
"}
{"error":"Could not complete setting all stats to F"}
{"error":"Could not complete setting all stats to C"}
```

2.4. <u>other</u>

```
SET_DATE
{"SET_DATE":[<year>, <month>, <day>]}
all are integers

Possible results
{"result":"Date is set"}
{"error":"SET_DATE arguments not in an array"}
{"error":"Invalid argument to SET_DATE, should be array [day, month,
year]"}


SET_TIME
{"SET_TIME":[<hours>,<minutes>]}

Possible results
{"result":"time set"}
{"error":"setting time failed"}
{"error":"SET_TIME arguments not in an array"}
{"error":"Invalid argument to SET_TIME, should be array [hours,
minutes]"}

TIME_ZONE
{"TIME_ZONE":<timezone offset>}
offset is an integer or a float

possible results
{"result":"timezone set"}
{"error":"Argument to TIME_ZONE should be an integer or float"}

DST_ON
{"DST_ON":0}
Automatic dst changing on

Possible results
{"result":"dst on"}

DST_OFF
{"DST_ON":0}
Automatic dst changing off
Possible results
{"result":"dst off"}
```

```
MANUAL_DST
{"MANUAL_DST":<number>}
Valid values are 0 and 1

Possible results
{"result":"Updated time"}
{"error":"argument should be an integer"}

NTP_ON
{"NTP_ON":0}
Turn on network time protocol client

Possible results
{"result":"ntp client started"}

NTP_OFF
{"NTP_OFF":0}
Turn off network time protocol client

Possible results
{"result":"ntp client stopped"}


READ_DCB 100
{"READ_DCB":100} for reading DCB100 used to recover status
information from the NeoHub

Possible results
{"error":"Could not read dcb"}
{"error":"READ_DCB arguments not in an array or 100"}
{"error":"Invalid first argument to READ_DCB, should be integer"}
{"error":"Invalid second argument to READ_DCB, should be a valid
device or array of valid devices"}
or if the argument was 100 an object with fields
"CORF" celcius or fahrenheit
"DATE"
"DSTAUTO"  Automatic switching of DST {daylight savings time)
"DSTON"    The current time is DST
"NTP"      The internet clock is active/inactive
"PROGFORMAT" the number means:
0 "NONPROGRAMMABLE"
1 "24HOURSFIXED"
2 "5DAY/2DAY"
3 "ILLEGAL"
4 "7DAY"
"TIME"
"TIMEZONE"

example:
{"CORF":"C","DATE":"Aug 22","DSTAUTO":true,"DSTON":true"Firmware
version":15,"NTP":"Running","NTPD":0,"PROGFORMAT":4,"TIME":"15:57:54
","TIMEZONE":0.0}
```

Set channel number

Used to change the zigbee channel to avoid conflicts with other devices.

{'SET_CHANNEL':25}

The only acceptable channels are 11, 14, 15, 19, 20, 24, 25

If successful the hub will respond, if the command fails no response will be seen.

The current channel number can now be viewed in DCB100

We recommend that all integrators check the command has been successfully received.

2.5. NeoPlug

Note1
Neoplug is seen primarily as a timeclock by the system with a few additional commands for manual switching so use timeclock commands for programming

Note2
To override the neoplug for a period of time use TIMER_HOLD_ON and TIMER HOLD OFF
To turn the output on or off use TIMER_ON or TIMER_OFF

```
MANUAL_ON
{"MANUAL_ON":<devices>}
Turns on NeoPlug manual mode. This is the opposite of automatic and
disables the time clock,effectively turning the neoplug into a
switch.

Possible results
{"result":"manual on"}
{"error":"Could not complete manual on"}
{"error":"Invalid argument to MANUAL_ON, should be a valid device or
array of valid devices"}

MANUAL_OFF
{"MANUAL_OFF":<devices>}
Turns off NeoPlug manual mode.

Possible results
{"result":"manual off"}
{"error":"Could not complete manual off"}
{"error":"Invalid argument to MANUAL_OFF, should be a valid device
or array of valid devices"}
```