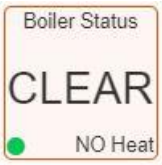
MaxAir Technical – Custom Message Sensors

'Message Sensors' provide the ability to display external text information on a Home Screen sensor tile.



For example, it is possible to display external status information captured from an interface to the boiler. Due to the format of the 'sensors' table, this information is passed as a numeric code, which must be converted to the message to be displayed on the tile'

There are four areas on the tile that accept data:

1. The Tile Name.
2. The centre text area.
3. The lower left status icon colour.
4. The lower right text area.

The same technique could be used for displaying data from any other external sources.

Implementation

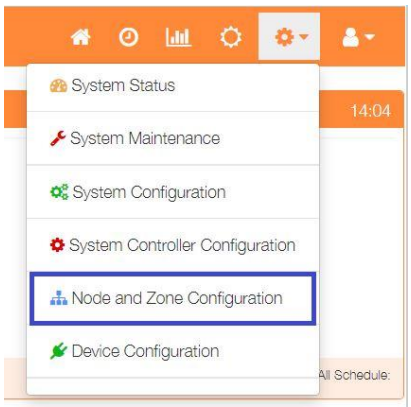
MaxAir

1. A 'Dummy' node will be created.
2. A 'Message Sensor' device will be created and allocated to the 'Dummy' node.
3. Mapping information will be created to place the required information on the sensor tile.

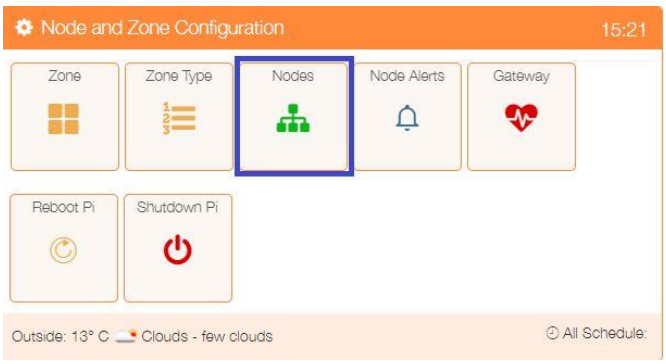
External System

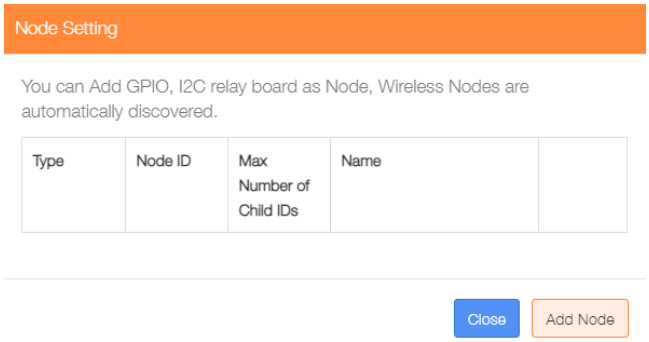
1. The external system will be able to access the MaxAir database from its Python script.
2. The required data in the form of a message code will be captured and used to add an entry to the MaxAir 'sensors' table columns 'current_val_1' and 'current_val_2', using the 'Dummy' node IDs created above.

MaxAir Configuration



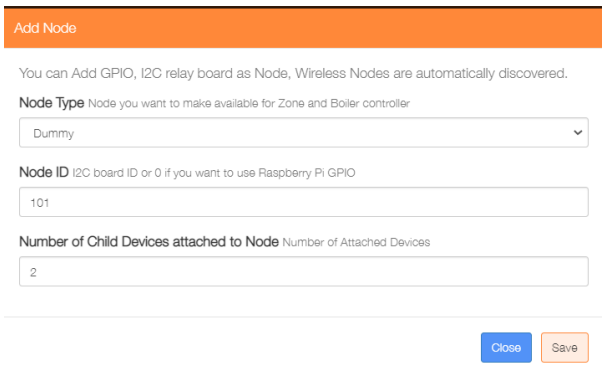
Select 'Node and Zone Configuration' from the Settings dropdown list, then click the 'Sensors' button.

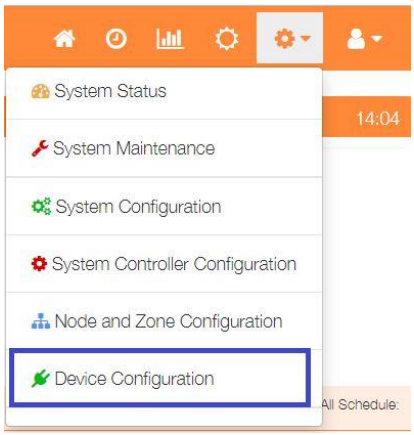




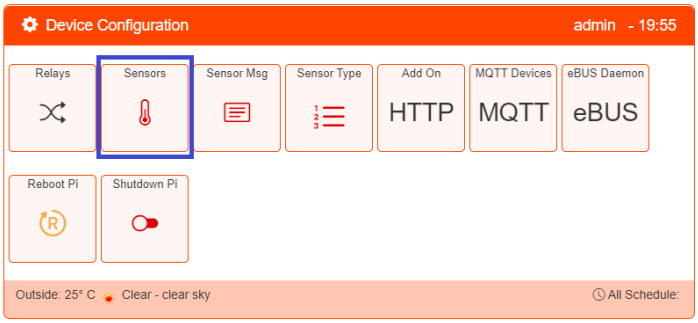
Click on ‘Add Node’.

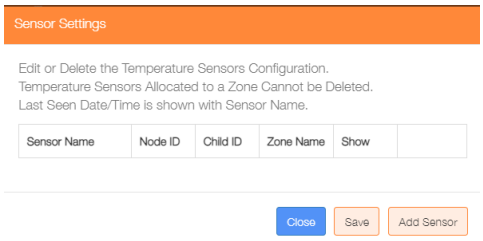
Add a ‘Dummy’ node type, the ‘Node ID’ can be any value not currently in use, and for this example the ‘Number of Child Devices attached to Node’ will be 1.





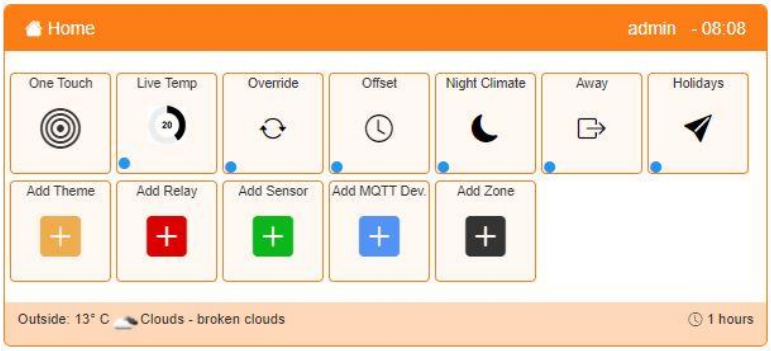
Select ‘Device Configuration’ from the Settings dropdown list, then click the ‘Sensors’ button.

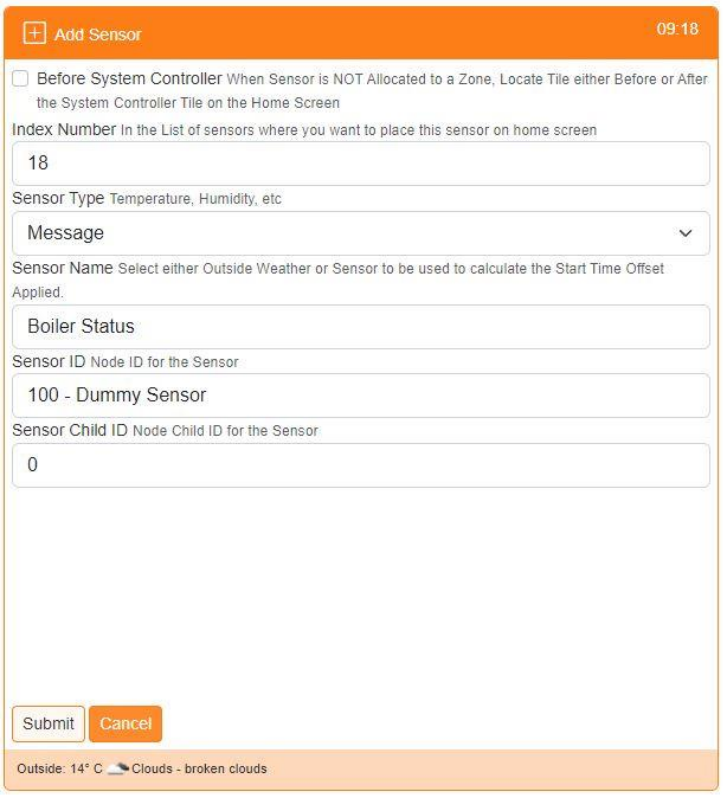




Click on the ‘Add Sensor’ button to configure the first sensor

An alternative method to go directly to the Add Sensor dialogue, is from the Home screen click on the ‘One Touch’ button then select the ‘Add Sensor’ menu item.





Show before or after the system controller on the Home screen

Used to order where on the Home screen the sensor is displayed

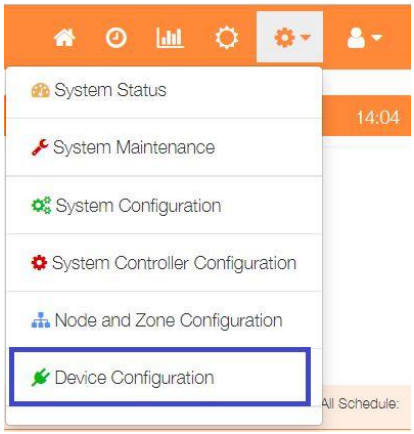
Select 'Message' type

Provide a name for this sensor device

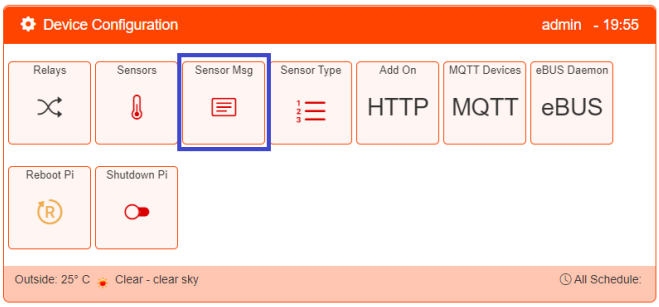
Select the Sensor ID from the dropdown list of available Nodes

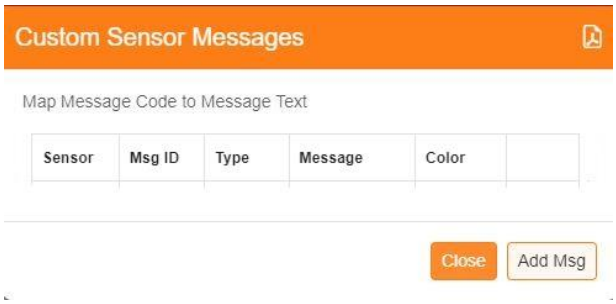
Choose the Child ID from the dropdown list, for nodes with only 1 sensor, this will be 0

Click on 'Submit' to add the device.



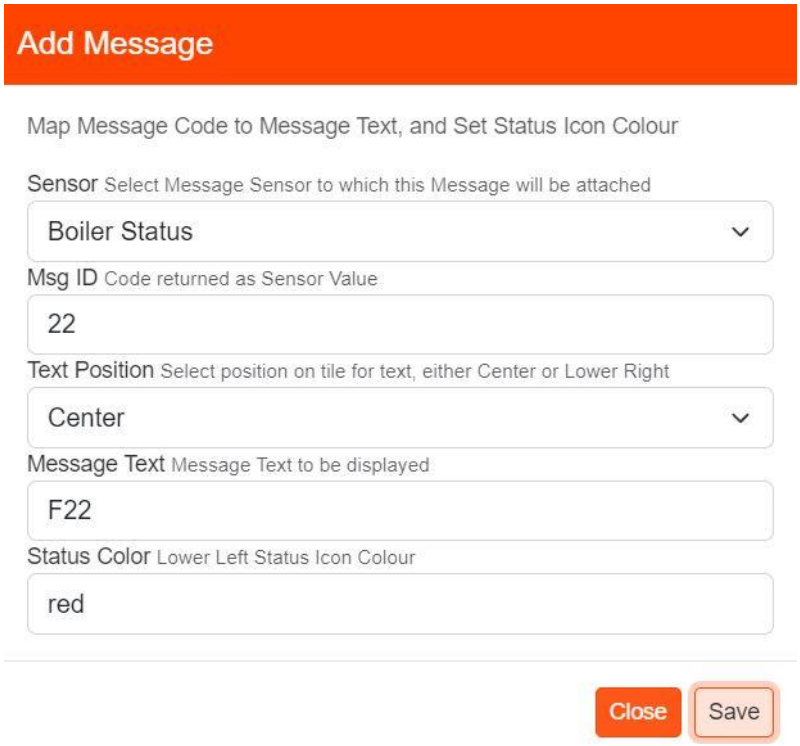
Select 'Device Configuration' from the Settings dropdown list, then click the 'Sensors Msg' button.





To start building the message mapping, click on the 'Add Msg' button.

For a centre message and associated status icon color:



Select the Message Sensor

Add the Message numeric code

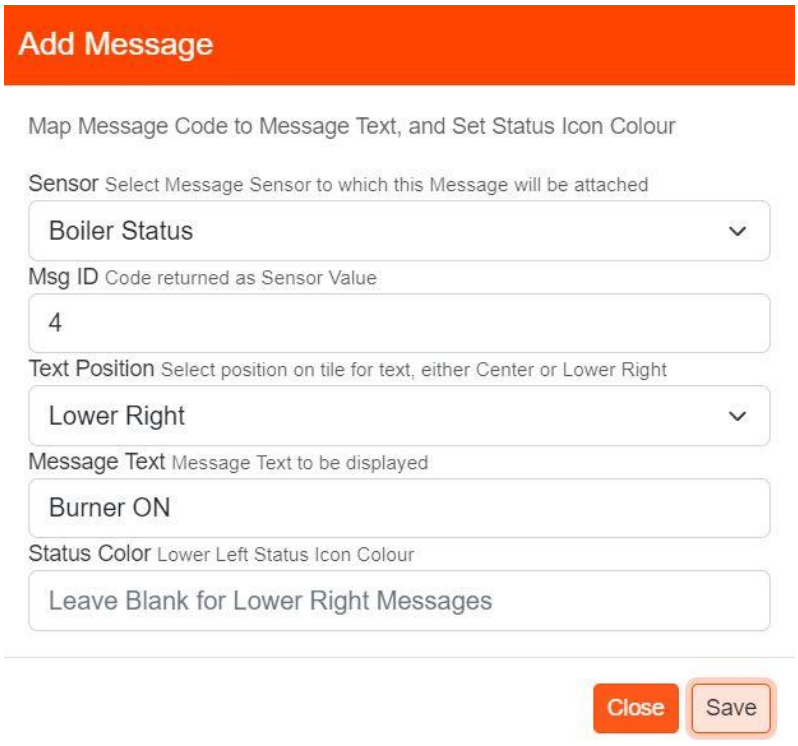
Select message position

Enter the text to be displayed

Set the associated status icon colour (HEX color codes can be used)

Click ‘Save’ when completed.

For a lower right message:



Select the Message Sensor

Add the Message numeric code

Select ‘1’ for lower right message

Enter the text to be displayed

Leave blank

Click ‘Save’ when completed.

Example Python Script to Update the MaxAir Database

```
#!/usr/bin/env python
import time, datetime, MySQLdb
from configparser import ConfigParser

#### Initialise the database access varables ####
config = ConfigParser()
config.read('/var/www/st_inc/db_config.ini')
servername = config.get('db', 'hostname')
username = config.get('db', 'dbusername')
password = config.get('db', 'dbpassword')
dbname = config.get('db', 'dbname')

#### Initialise the database connection ####
con = mdb.connect(dbhost, dbuser, dbpass, dbname)
cur = con.cursor()

def update_maxair_sensors (conn, node_id, sensor_id, val_1, val_2, msg_in, msg_in_val) :
    cnx = conn.cursor()
    # get 'current_val_1
    cnx.execute("SELECT * FROM `sensors` WHERE `id` = (%s) LIMIT 1;",
    (sensor_id,))
    result = cnx.fetchone()
    sensor_to_index = dict(
        (d[0], i) for i, d in enumerate(cnx.description)
    )
    sensor_name = result[sensor_to_index["name"]]
    sensor_child_id = int(result[sensor_to_index["sensor_child_id"]])
    current_val_1 = float(result[sensor_to_index["current_val_1"]])
    current_val_2 = float(result[sensor_to_index["current_val_2"]])
    graph_num = int(result[sensor_to_index["graph_num"]])
    timeout = int(result[sensor_to_index["timeout"]])
    resolution = int(result[sensor_to_index["resolution"]])
    if val_1 != current_val_1 or val_2 != current_val_2 :
        # update 'current_val_1' and 'current_val_2'
        try :
            query = ("UPDATE `sensors` SET `current_val_1` = " + str(val_1) + ", `current_val_2` = " + str(val_2) + " WHERE `id`
= " + str(sensor_id) + ";")
            cnx.execute(query)
            conn.commit()
        except mdb.Error as e:
            print("DB Error %d: %s" % (e.args[0], e.args[1]))
            print(traceback.format_exc())
            logging.error(e)
            logging.info(traceback.format_exc())
            conn.close()
            print(infomsg)
            sys.exit(1)
        # update node last seen time
        try :
            query = ("UPDATE `nodes` SET `sync` = 0, `last_seen` = '" + str(datetime.now()) + "' WHERE `node_id` = '" +
str(node_id) + "';")
            cnx.execute(query)
            conn.commit()
        except mdb.Error as e:
            print("DB Error %d: %s" % (e.args[0], e.args[1]))
            print(traceback.format_exc())
            logging.error(e)
```

```
            logging.info(traceback.format_exc())
            conn.close()
            print(infomsg)
            sys.exit(1)

# check if a 'Boiler Flow' sensor exists in the database
boiler_flow_sensor = False
cur.execute("SELECT * FROM sensors WHERE name = 'Boiler Flow' LIMIT 1;")
result = cur.fetchone()
if cur.rowcount > 0 :
    sensor_to_index = dict(
        (d[0], i) for i, d in enumerate(cur.description)
    )
    boiler_flow_id = int(result[sensor_to_index["id"]])
    boiler_flow_sensor_id = int(result[sensor_to_index["sensor_id"]])
    if int(result[sensor_to_index["message_in"]]) == 1 :
        boiler_flow_msg_in = True
    else :
        boiler_flow_msg_in = False
    cur.execute('SELECT node_id FROM nodes WHERE id = (%s)', (boiler_flow_sensor_id, ))
    result = cur.fetchone()
    if cur.rowcount > 0 :
        node_to_index = dict(
            (d[0], i) for i, d in enumerate(cur.description)
        )
        boiler_flow_node_id = int(result[node_to_index["node_id"]])
        boiler_flow_sensor = True

Loop reading status from boiler and send to MaxAir
while True:
    # get the current flow temperature
    if boiler_flow_sensor :
        response = ems_read('flowtemp')
        if "Error" not in response:
            flowtemp = float(response.rstrip())
            flow_temp_error = False
        else :
            flow_temp_error = True
        update_maxair_sensors(con, boiler_flow_node_id, boiler_flow_id, flowtemp, 0, boiler_flow_msg_in, flowtemp)
        print(bc.dtm + datetime.now().strftime("%Y-%m-%d %H:%M:%S") + bc.ENDC + " - Flow Temp               - " +
str(flowtemp))
        if flow_temp_error :
            log_txt = log_txt  + 'BOILER FLOW TEMP 0C*\n'
        else :
            log_txt = log_txt  + 'BOILER FLOW TEMP ' + str(flowtemp) + 'C\n'      pass

    time.sleep(1)
```