ioBroker.ems-esp Adapter Documentation – August 2023

ioBroker.ems-esp Adapter - Introduction

The adapter supports an interface to the Bosch Group's heating systems with EMS or EMS+ bus.
(Buderus/Junkers/Netfit, etc.) The adapter can be used for both gateways.

- with the original LAN interfaces of the Bosch Group heaters (IP-inside, km200, km100, km50, MB-LAN2 etc), The newer Bosch interfaces (e.g. MX200 etc) no longer support local LAN access and are not supported by the adapter.

- as well as with the EMS bus gateway with ESP32 chip. (https://github.com/emsesp/EMS-ESP32). The EMS Bus Gateway can be ordered from BBQKees: BBQKees
  Electronics – EMS bus to Home Automation interfaces (bbqkees-electronics.nl)

The ems-esp gateway is a small box that is connected to the service port or directly to the EMS bus of the heating system / heat pump and which then establishes the connection between the heating system and the home automation system via WLAN/LAN and MQTT/WEB-API.

This EMS-BUS gateway can connect to a large number of older heating systems without an Internet connection with Homeassistant. Together with the software developers of the EMS-ESP firmware, the WEB-API was adapted so that this ems-esp ioBroker adapter can be integrated seamlessly.

API-Calls

The adapter can read and write data on both gateways via WEB API to control all heating components. It can be used either for the original gateways of the Bosch Group or the ems-esp gateway or both in parallel.

The WEB-API communication to the km200 gateway is encrypted, the communication to the EMS-ESP gateway is not.
Write operations are ensured at the ems-esp gateway by a generated access token.

	ioBroker Adapter			
regular polling	< ----- web API ---------- > LAN en- / decrypted	**km200 Gateway**	< -------- >	**And**
regular polling	< ----- web API /V3 ------- > LAN / WLAN	**ems-esp Gateway**	< -------- >	**M** **BUS**
Energy polling & calculation additional functions	< ----- Adapter Logic ------ >	**ems-esp/km200**	< -------- >	**S**

Differences between original Bosch gateways (km200) vs. Ems-Esp gateway

The original Bosch gateways ensure that the heating system can be operated via the Internet via the corresponding apps of the Bosch Group. The range of functions is very limited. Communication takes place via the Bosch cloud. Essentially, the switching times, holiday times and the temperature specifications can be set. A few plant data are readable.

A documented API does not exist at Bosch. Communication with the original gateways is encrypted – even on the local LAN (see below). The km200 gateway does not support any control or system parameters in the LAN. The API accesses must be made separately for each data field. This means that the reading process (polling) is slow and the polling cycle is limited by the adapter to a minimum of 90 seconds.

The Ems-ESP Gateway is a separate piece of hardware that directly accesses the Telegram traffic in the EMS bus. Since these telegrams are not documented, it has been a lot of detective work to decipher and implement them for the different heating systems. Many heating parameters and setting options are supported, but other user settings are not (yet) available. A variety of different (even older) heating systems without an internet gateway and brands are supported.

The data is read out very fast, so even short polling cycles of 15 seconds are possible.

Adapter Sequence Logic

In the adapter you can select whether a KM200 and/or an EMS-ESP gateway should be read out. If both are to be present, it is recommended to display the EMS-ESP data structure in the KM200 logic.

In addition, the "Parameters" tab can be used to select whether statistics are to be created or whether the boiler efficiency is to be determined as a function of the temperatures. (Condensing effect). In addition, there is a heating demand control (see below)

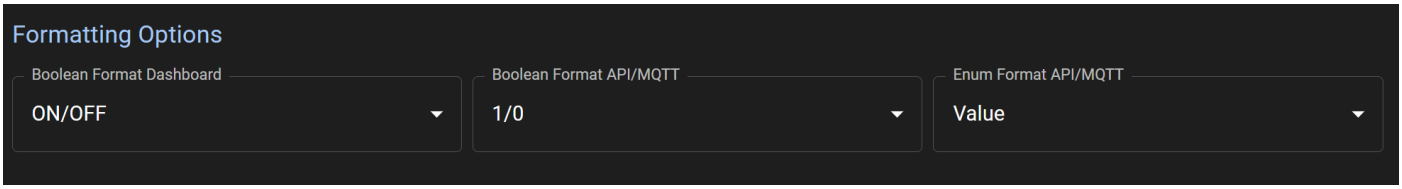
When the adapter is started, all previous states can be deleted (see Parameters). This is only recommended if data structures are changed.

After starting, all details about the states are read once via API and the corresponding ioBroker objects are created. This is followed by a regular polling cycle to read the current values and update the states.

All other processing is carried out independently in separate processing cycles.

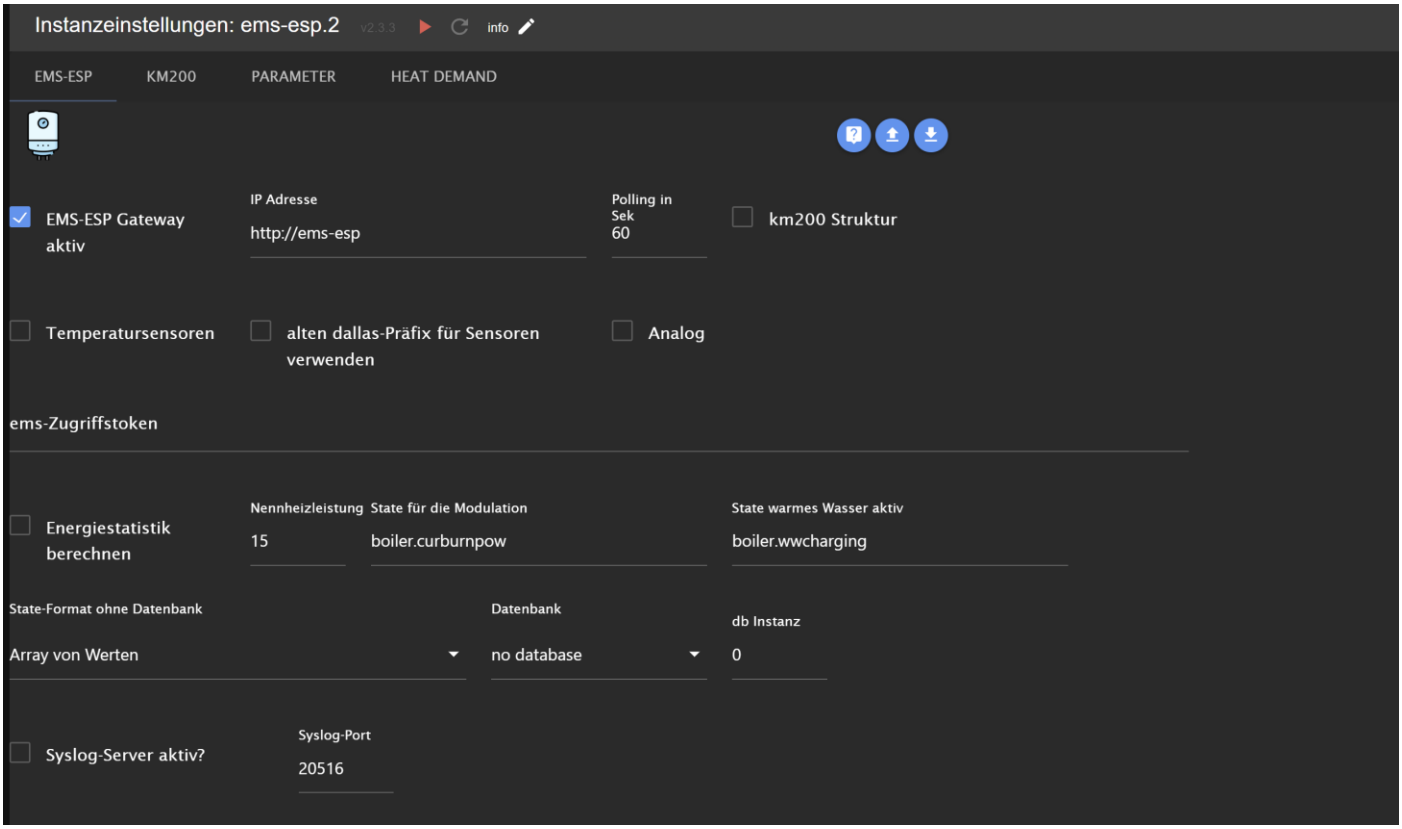
EMS-ESP settings:

In the EMS-ESP Gateway, the "Formatting Options" for the API interface must be set under Settings.
Formatting options for the Boolean format must be 1/0, and for the Enum format, Index or Value.



In order for the ioBroker adapter to be able to write values via API, this must be allowed in the settings. First of all, I recommend „Bypass Access Token authorization on API calls" for beginners. and later generate the token for access protection and enter it in the adapter configuration.

ioBroker Instance Settings:



EMS-ESP settings:

The km structure check box either uses the km200-like device structure for EMS-ESP data fields, or retains the original EMS-ESP device view: boiler, thermostat, mixer, etc. Otherwise, the

IP address of the gateway, the polling time and the access token must be entered. In addition, the reading of connected Dallas and analog sensors can be activated.

NEW: The creation of energy consumption statistics can now be activated (see chapter Energy).

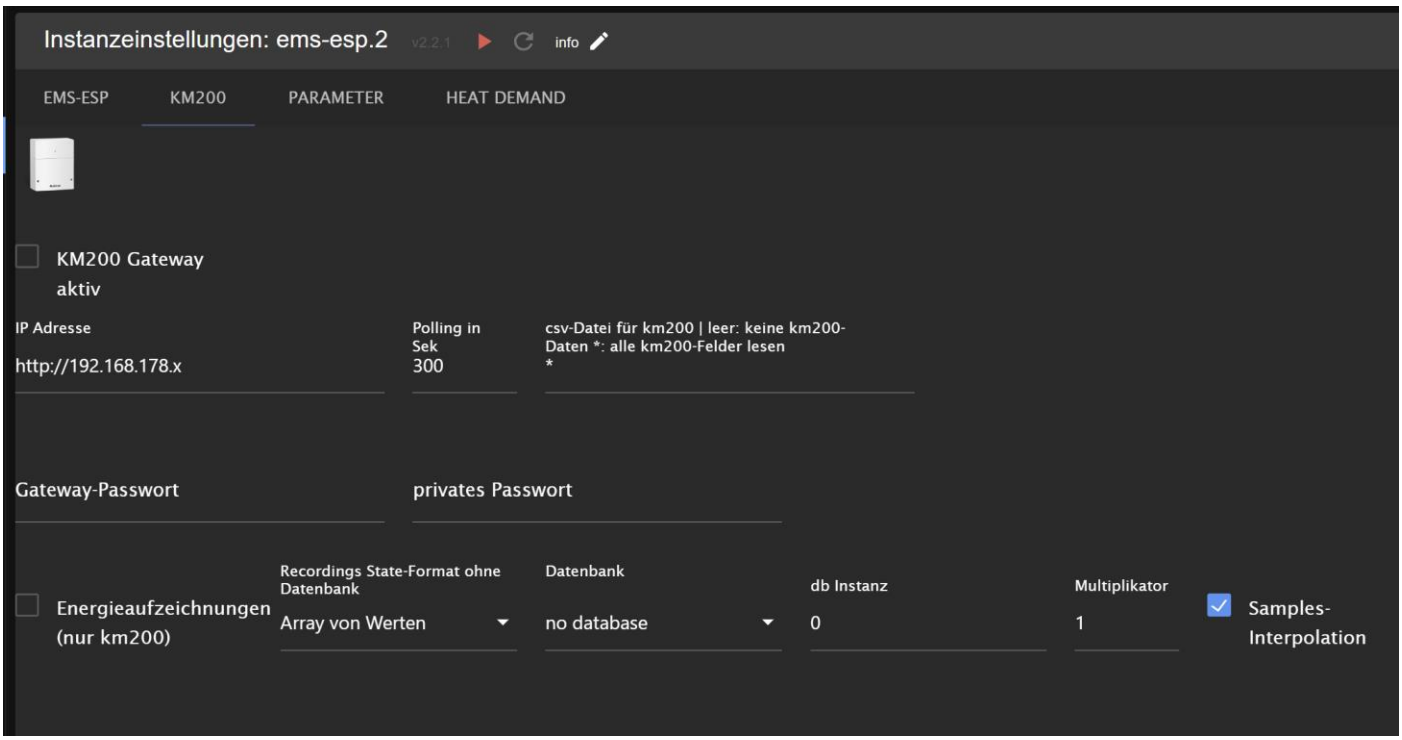
NEW: For evaluations of the bus telegrams, a syslog server can be activated in the adapter. This presupposes that this function is activated in the EMS-ESP settings and that the IP address and the syslog port are configured there. (See chapter Syslog Server).

Settings KM200

The Web API calls to/from the km200 gateway are encrypted. Two passwords are required for encryption/decryption:

- The gateway password on a label on the gateway in the form: xxxx-xxxx-xxxx-xxxx (case sensitive and enter the hyphens)
- The private password is the one set with the Buderus **MyDevice** app!
- (**don't use myBuderus** or similar cloud apps!)

ioBroker Instance Settings:



The IP address of the gateway, the polling time and the two passwords must be entered.

At the 1st adapter start, it is recommended to select all km200 data fields with a "*". The adapter then creates a km200.csv file in the .. directory. /iobroker-data/ems-esp/{instance}.

This file can be used the next time the adapter instance is started. Unneeded rows (fields) can be deleted to reduce the number of km200 fields to be read. (Make a copy and rename file).

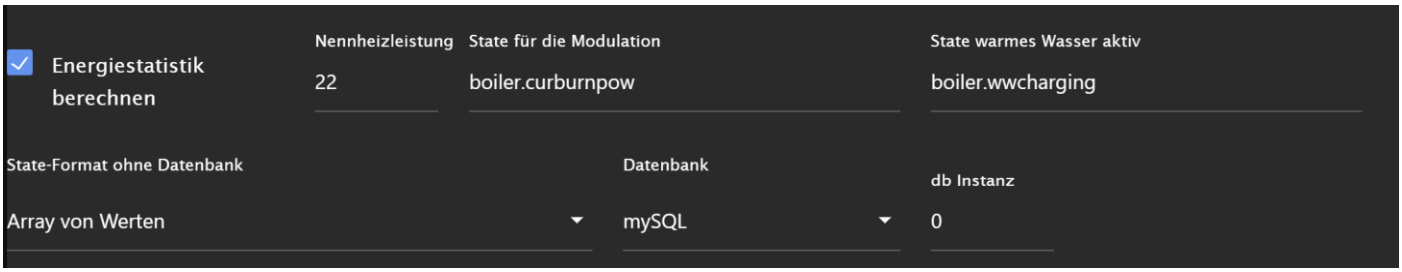
The km200 web API requires querying each value with its own http-get command. This can be about 150 individual queries even for systems with 2 heating circuits. A single query cycle takes a correspondingly long time (15-40 seconds).

- KM200 polling is also a parameter (default 300 seconds) and the minimum adjustable value is 90 seconds. (see duration of the polling cycle)
- km200-recordings (energy consumption and temperature statistics) are updated hourly.

Energy consumption statistics for EMS-ESP

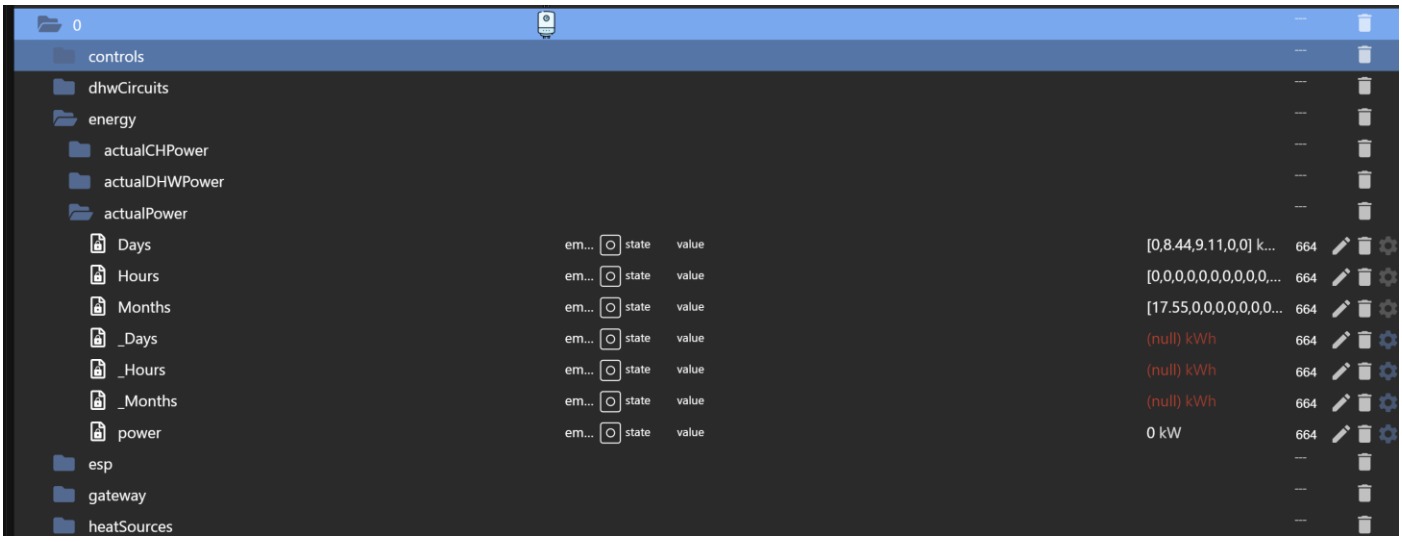
The EMS-ESP Gateway does not calculate the consumption values in the firmware. This calculation is now implemented in the ioBroker adapter. Past values are not readable. For the EMS-ESP gateway, values are determined and updated every 15 seconds.
The energy statistics can be activated in the instance and require an active database instance. (History, mySQL, InfluxDB). For InfluxDB V1, the retention policy must be set to at least 170 weeks. (Change retention policy globally for ioBroker duration 170w;)

EMS-ESP:



The nominal heating capacity of the boiler and the state names for the current modulation and for WW-Active must be entered. On this basis, the current burner, heating (CH) and hot water (DHW) output are then determined.

In the object structure, three substructures are then created under energy: actualCHPower (heating), actualDHWPower (hot water) and actualPower (total).



The power states are updated every 15 seconds and then the energy consumption values are determined every 10 minutes. The consumption values can then be displayed graphically with e.g. Flot.

- hourly values : _Hours
- the daily values: _Days
- the monthly values: _Months

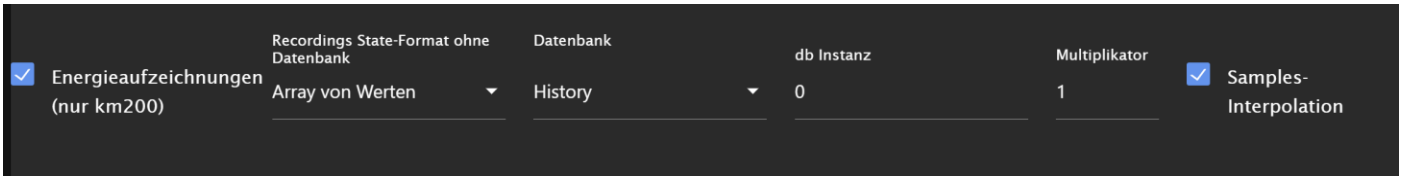
saved/updated with database direct access. (That's why zero). The JSON array values are updated depending on the selection of the format (current values first):

- hourly values: Hours
- the daily values: Days
- the monthly values: Months

Energy Consumption Statistics for KM200 (Recordings)

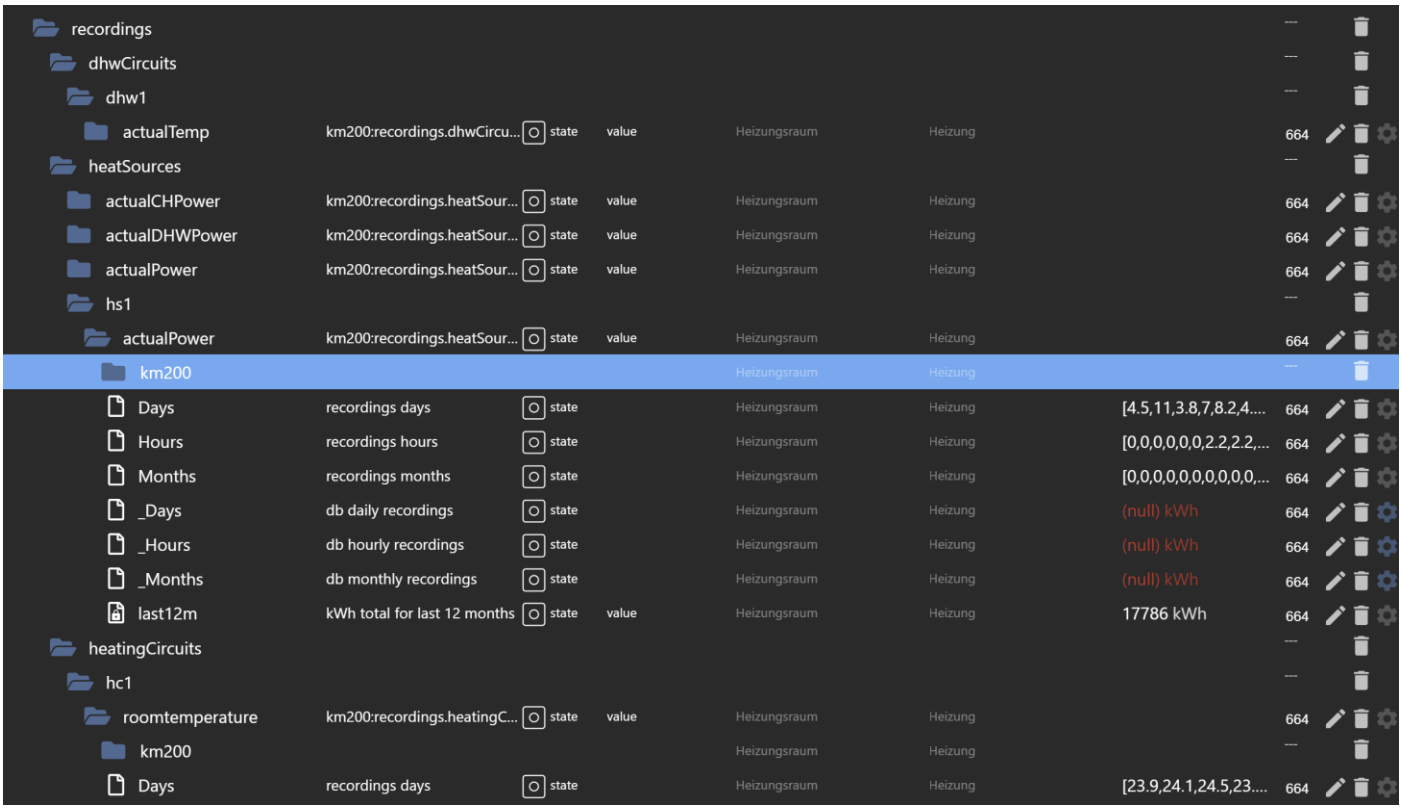
Most modern heating systems have an IP-Inside or KMxxx gateway and support energy and temperature statistics. These are calculated in the gateway. As a rule, consumption and temperature values of the last 12 months can be called up. The Bosch gateways calculate the consumption values by means of "samples" every 60 seconds. Hourly, daily and monthly values can be read. The adapter reads these values every hour.

KM200:



The checkbox "Energy Records" must be activated and the database instance (History, mySQL or InfluxDB) must be defined. The History, SQL, or InfluxDB adapter must be installed and active to use this option. For InfluxDB V1, the retention policy must be set to at least 170 weeks. (Change retention policy globally for ioBroker duration 170w;)

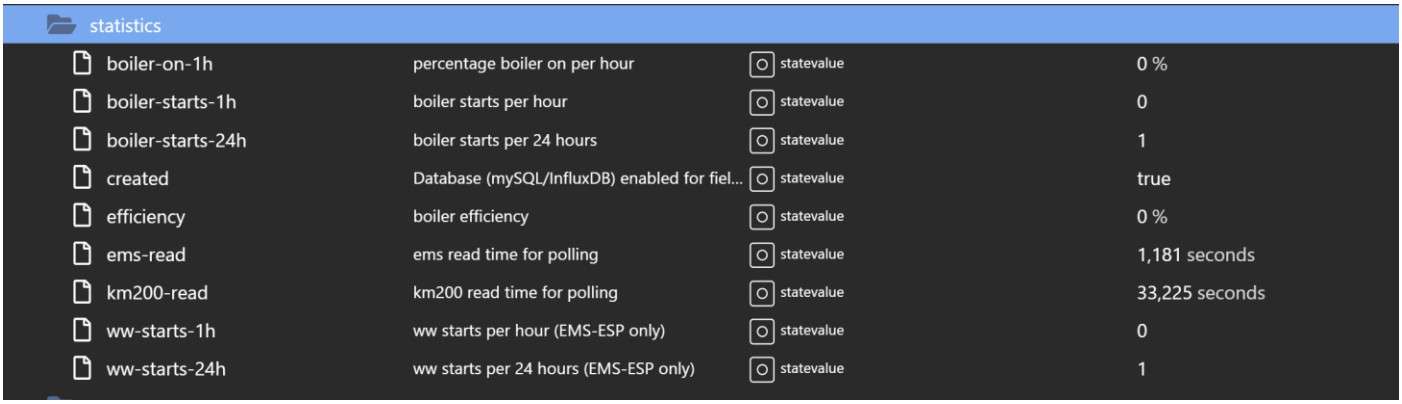
A multiplier can be entered if values are not available in kWh. There are months when samples are missing. The cloud apps then upscale the values (e.g. if 10% of the values are missing, then the value is increased by 10%). This can be enabled in the adapter so that the values match the app.



The object structure is specified by the Bosch API. There are consumption values and temperatures. As with EMS-ESP, it is divided into hours, days and months as arrays of values and in the states _Hours, _Days and _Months directly as database entries for graphical representation.
IMPORTANT: For databases, the data is written directly to the "_" states using SQL commands. The value (zero) is then displayed under Objects. (That's right !!). See previous chapter.

Statistics

Burner statistics can be activated and show:



- boiler-on-1h: What percentage (0-100%) was the boiler active during the last hour
- boiler-starts-1h and boiler-starts-24h: number of boiler starts in the period (1 / 24 hours)
- created: Indicator that the statistics structure has been created
- efficiency: current boiler efficiency when activated (condensing value for gas and oil boilers)
- ems-read: The query cycle processing time for EMS-ESP gateway reads
- km200-read: ... analogue for KM200
- WW-STARTS-1H and WW-STARTS-24H (only with active EMS-ESP Gateway) – Boiler starts for WW preparation

An active database instance (see above) is required to calculate the statistics.

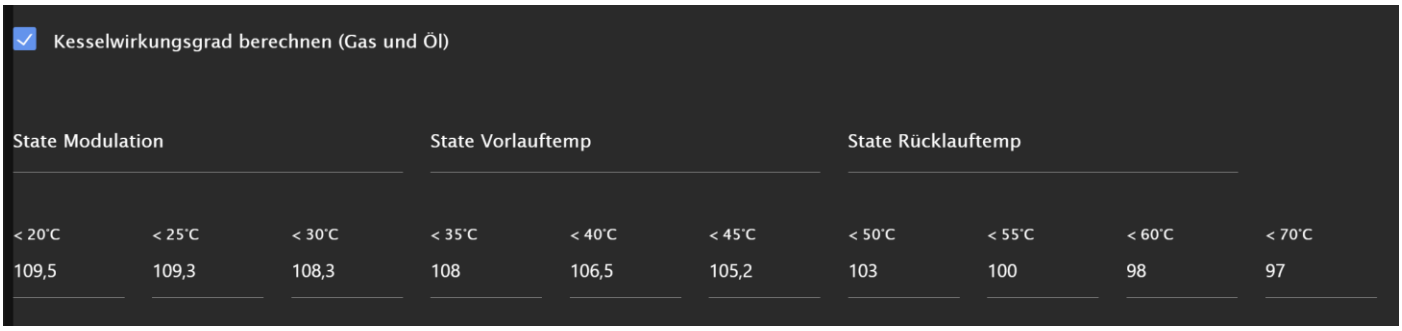
Condensing value benefits – burner efficiency

Boiler efficiency can be calculated if the parameters are filled in.
(gas and oil boilers only)

Efficiency (condensing value) is calculated based on the average boiler temperature: (boiler temperature + return temperature) / 2.

Check the data sheet of your boiler to adjust the efficiency table accordingly.

The states for modulation, flow and return temperature must be entered.



Changes in the State Structure

Whenever a new EMS-ESP firmware adds new data fields and/or changes data field names, they are processed during the adapter run.

However, obsolete data fields are not automatically deleted by the adapter. It is possible to rebuild the state structure by deleting states when the adapter is restarted (states with history / DB entries are retained and may have to be deleted manually).

After an Ems-Esp firmware update, it is recommended to restart the adapter by deleting the fields.
However, this can also be made up for later.

Heat demand control in the ems-esp adapter

In the current version of the ems-esp adapter, the heating is controlled depending on a calculated heat demand.

There is a separate configuration page "Heat demand" in which there are 2 input lists:(New entries with the + symbol)

In the first block, the following entries are defined for each room (name of your choice):

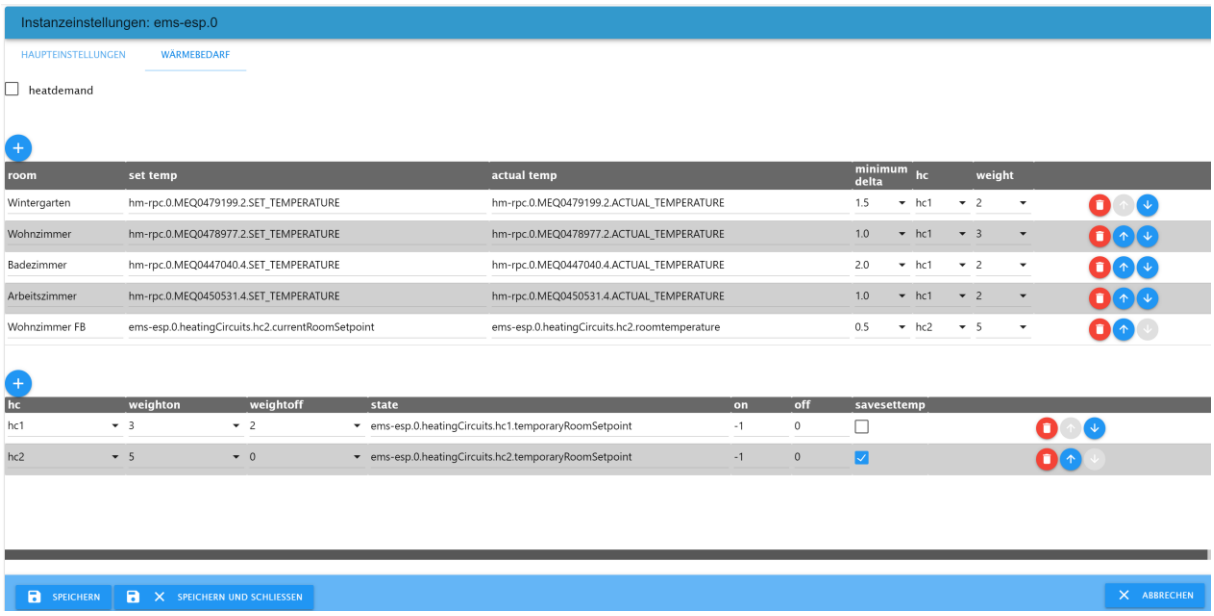
- *Settemp*: State for the setpoint temperature of the radiator / room
- *Actual temp*: State for the actual temperature of the room
- *Minimum delta* – Difference between settemp – actualtemp from the heating demand exists:Example: Target 21° - Is 20→° delta 1°. If delta >= minimum → delta then heating demand.Minimum delta = 0 means heating demand if the current temperature is equal to or less than the target temperature.
- *Hc* : Assignment to the heating circuit (hc1 ... hc4)
- *Weight*: Weighting of the radiator / room (What is the heating capacity of the radiator or underfloor heating?).

In the second block, for each heating circuit, the following are determined:

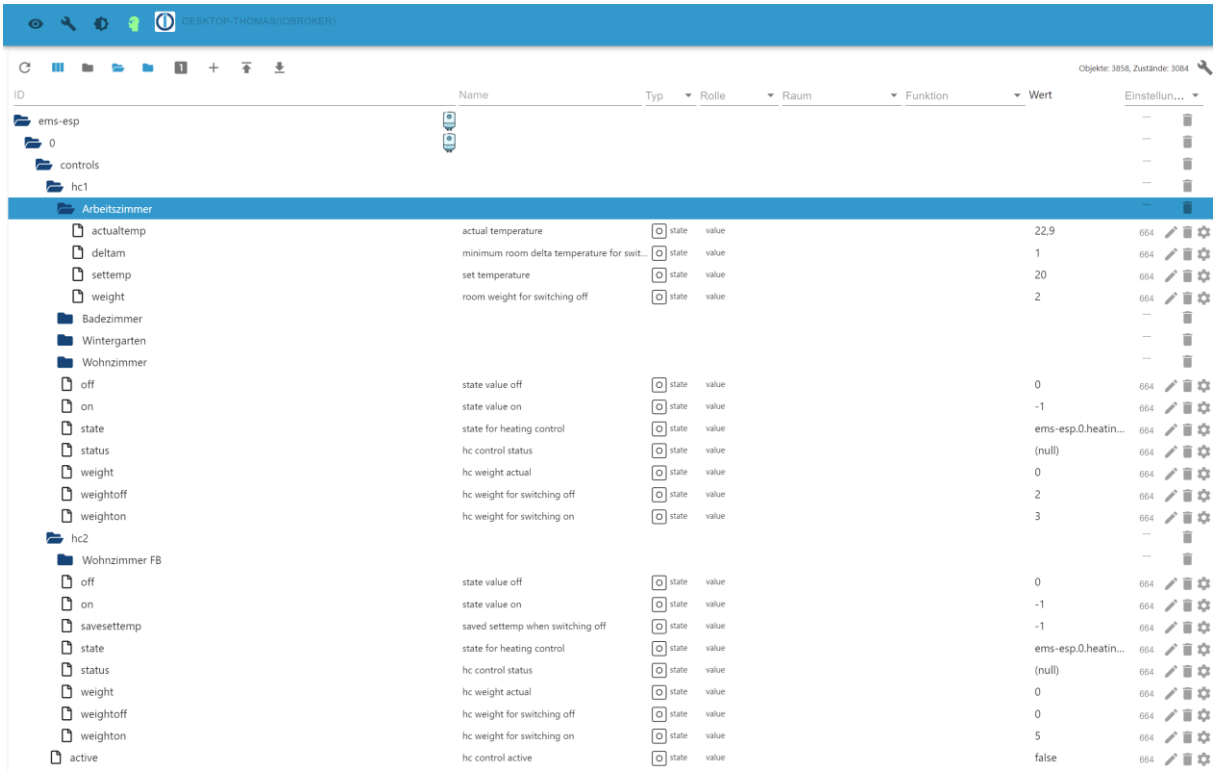
- *Weighton*: **HK at the** sum of the weighting values of the heating circuits >= weighton
- *Weightoff*: **HK off** at sum of the weighting values of the heating circuits <= weightoff
- *State*: state to be switched
- *On*: Value of the state for HK.
- *Off*: Value of the state for HK.
- *Savetemp*: When switched on, the current setpoint is stored and this value is taken as a reference when the heating circuit is switched off. This is necessary because when the underfloor heating circuit is switched off, the setpoint is set to 0.

The *heatdemand* switch switches the automatic heat demand control on or off when the adapter starts.

Here is an example configuration with Homematic thermostats (hc1) and km200 mixer-controlled underfloor heating circuit (hc2).



In the object structure of the ems adapter, the following object states are then created under controls after the instance has been started :



The last state *active is preset with the value of* heatdemand when the adapter is started and controls whether the heat demand-dependent control is active (true) or inactive (false). The value can then be set, for example, via VIS. For example, when the adapter is started, the control is initially inactive without a set value *heatdemand* and can later be set active in VIS.

It is important to choose the state to be switched wisely. It would be possible, for example, to switch the heating circuit off or on over summer / winter mode (e.g. km200: heatingCircuits.hc1.suWiSwitchMode)
This has the disadvantage that in the event of adapter stop or network problems (km200 not available), the heating circuit may remain permanently off or on and must be manually reset on the thermostat. According to Murphy's Law, this usually happens during vacation / absence ....

I therefore prefer the "temporary setpoint" (e.g. heatingCircuits.hc1.temporaryRoomSetpoint).
With my RC310, these temporary adjustment options are available for each heating circuit.

This state has the advantage that changes in values are only temporary until the next switching time of the heating program. The value "0" switches off the HK, the value "-1" back on to automatic mode. (But it would also be possible to set a fixed temperature: e.g. "21" degrees.)

I take the automatic mode, so that an automatic heating circuit shutdown continues to work after reaching the outside temperature threshold of the heating circuit, despite the active heat demand control.

The weighting of the radiators, the switch-on threshold and the "*minimum delta" of the* heating circuit should be determined in such a way that the heat output of the heat generator can be absorbed for a sufficiently long time with the smallest modulation. With a *"minimum delta*" of 1°, my Homematic thermostats ensure that the valves are open. If all rooms / thermostats are weighted equally (e.g. with 1) and the switch-on threshold of the heating circuit is also 1, then every heating requirement of a room will turn on the burner.