

ioBroker.ems-esp Adapter Documentation - August 2023

ioBroker.ems-esp Adapter - Introduction	2
API-Calls	2
Differences Original Bosch Gateways (km200) vs. Ems-Esp Gateway	3
Adapter flow logic	3
Settings EMS ESP:	4
Settings KM200	5
Energy consumption statistics for EMS ESP	6
Energy consumption statistics for KM200 (Recordings)	7
The statistics	8
Condensing benefit – burner efficiency	8
Changes in the states structure	9
Heat requirement control in the ems-esp adapter	10
Syslog Server	12

ioBroker.ems-esp Adapter - Introduction

The adapter supports an interface to the heating systems of the Bosch group with EMS or EMS+ bus. (Buderus/Junkers/Netfit etc.) The adapter can be both

- 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 \(bbqkees-electronics.nl\)](https://bbqkees-electronics.nl)

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

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, that to the EMS-ESP gateway is not. Write operations are secured with the ems-esp gateway by a generated access token.

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

Differences 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 from the Bosch group. The range of functions is very limited. Communication takes place via the Bosch cloud. Essentially, the switching times, vacation times and the temperature specifications can be set. A few system data are readable.

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

The Ems-ESP Gateway is a separate piece of hardware that directly accesses the telegram traffic on the EMS bus. Since these telegrams are not documented, it took a lot of detective work to decode and implement them for the different heating systems. Many heating parameters and setting options are supported, but other user settings are not (yet) available.

Many different (even older) heating systems without an Internet gateway and brands are supported.

The data is read out very quickly, which means that short polling cycles of 15 seconds are also possible.

Adapter flow logic

In the adapter you can select whether a KM200 and/or an EMS-ESP gateway should be read.

If both are to be present, it is advisable to display the EMS-ESP data structure in the KM200 logic.

In addition, the "Parameters" tab can be used to select whether statistics should be created or whether the boiler efficiency should be determined depending on the temperatures. (Condensing effect).

In addition, there is a heating demand control (see below)

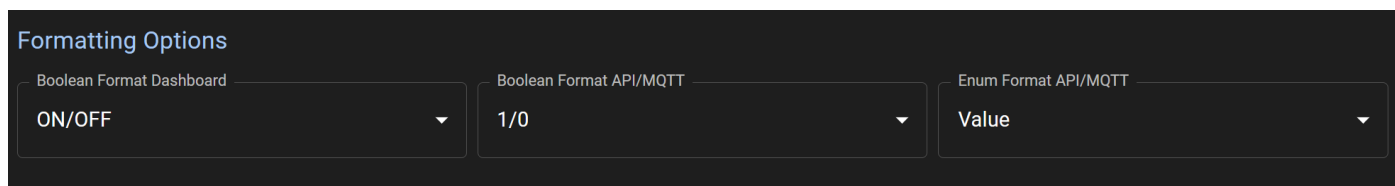
When starting the adapter, all previous states can be deleted (see parameters). This is only recommended if data structures are changed.

After the start, all details about the states are read once via API and the corresponding ioBroker objects are generated. The current values are then read and the states are updated in a regular polling cycle.

All other processing takes place independently of this in separate processing cycles.

Settings EMS ESP:

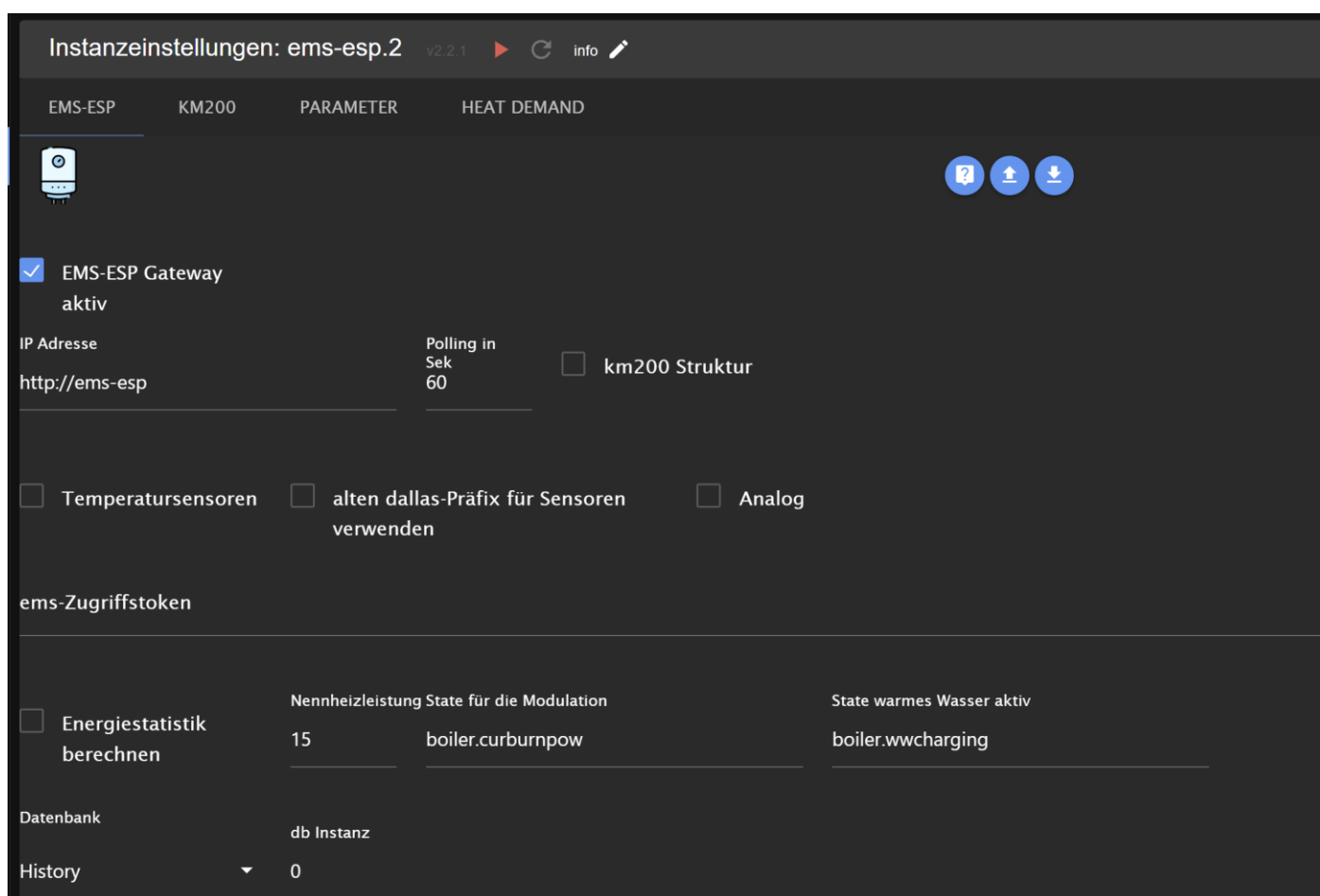
In the EMS-ESP Gateway, the "Formatting Options" for the API interface must be set under Settings. Formatting Options for Boolean format must be 1/0 and for Enum format Index or Value.



The screenshot shows the 'Formatting Options' panel with three dropdown menus. The first dropdown, 'Boolean Format Dashboard', is set to 'ON/OFF'. The second dropdown, 'Boolean Format API/MQTT', is set to '1/0'. The third dropdown, 'Enum Format API/MQTT', is set to 'Value'.

So that the ioBroker adapter can also write values via API, this must be allowed in the settings. I recommend beginners first, **Bypass Access Token authorization on API calls**, and later to generate the token for access protection and to enter it in the adapter configuration.

ioBroker instance settings:



The screenshot shows the 'Instanzeinstellungen: ems-esp.2' settings page. The 'EMS-ESP' tab is selected. The 'EMS-ESP Gateway' is checked and 'aktiv'. The 'IP Adresse' is 'http://ems-esp'. The 'Polling in Sek' is '60'. The 'km200 Struktur' checkbox is unchecked. There are checkboxes for 'Temperatursensoren', 'alten dallas-Präfix für Sensoren verwenden', and 'Analog'. The 'ems-Zugriffstoken' section is empty. There are checkboxes for 'Energiestatistik berechnen', 'Nennheizleistung State für die Modulation' (set to '15'), and 'State warmes Wasser aktiv' (set to 'boiler.wwcharging'). The 'Datenbank' section has a 'db Instanz' set to '0'. The 'History' dropdown is set to '0'.

EMS ESP settings:

The km structure checkbox either uses the km200-like device structure for ems-esp data fields or keeps 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 analogue sensors can be activated.

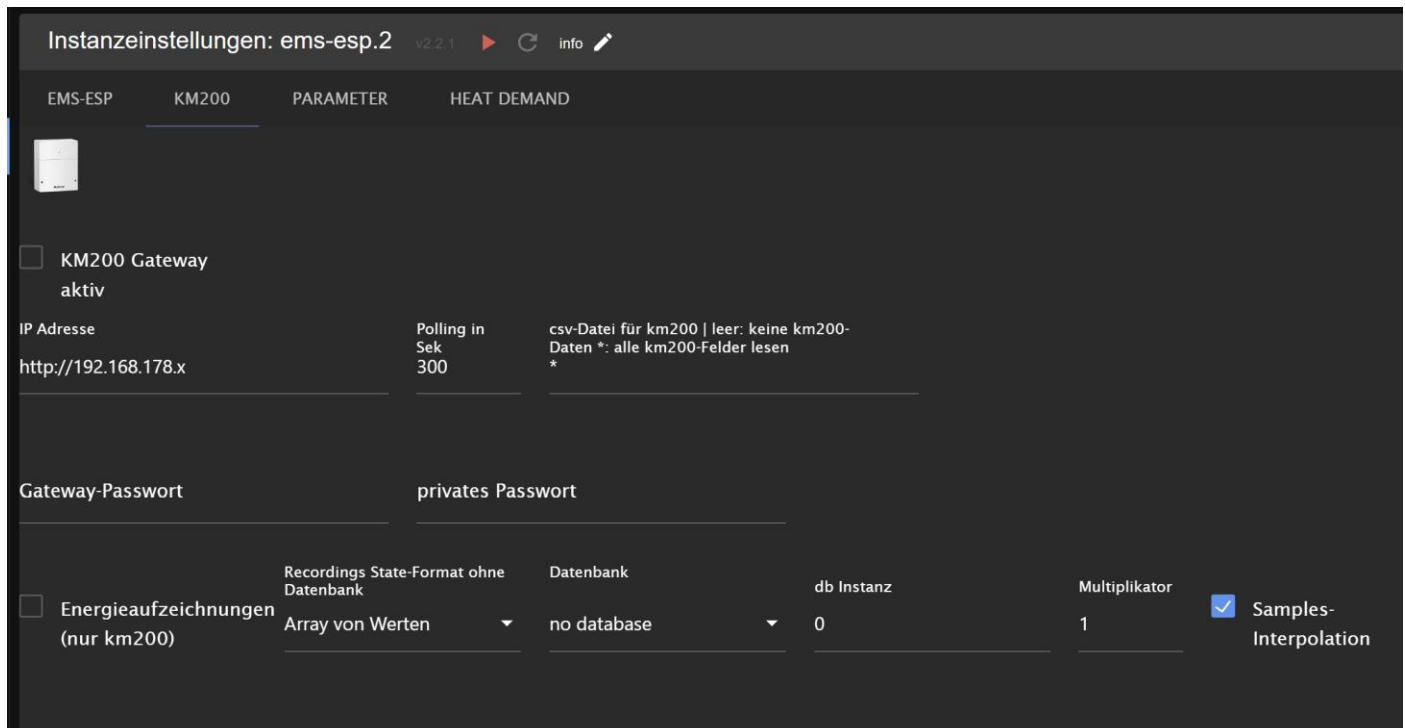
NEW: The creation of energy consumption statistics can now be activated (see chapter Energy).

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 (note upper and lower case and enter the hyphens)
- The private password is the one associated with the Buderus **MyDevice** App!
- **(not myBuderus or use similar cloud apps!)**

ioBroker instance settings:



The screenshot shows the 'Instanzeinstellungen: ems-esp.2' interface with the 'KM200' tab selected. The interface includes a header with 'v2.2.1', a play button, a refresh button, and an 'info' icon. Below the header are tabs for 'EMS-ESP', 'KM200', 'PARAMETER', and 'HEAT DEMAND'. The 'KM200' tab contains a 'KM200 Gateway' section with a checkbox labeled 'aktiv'. Below this is the 'IP Adresse' field with the value 'http://192.168.178.x'. To the right is the 'Polling in Sek' field with the value '300'. Further right is a text field for the 'csv-Datei für km200' with the value 'leer: keine km200-Daten *: alle km200-Felder lesen'. Below these fields are two password fields: 'Gateway-Passwort' and 'privates Passwort'. At the bottom, there is a section for 'Energieaufzeichnungen (nur km200)' with a checkbox. To its right are several dropdown menus: 'Recordings State-Format ohne Datenbank' (set to 'Array von Werten'), 'Datenbank' (set to 'no database'), 'db Instanz' (set to '0'), and 'Multiplikator' (set to '1'). On the far right of this section is a checkbox for 'Samples-Interpolation' which is checked.

Enter the IP address of the gateway, the polling time and both passwords.

When starting the adapter for the first time, it is recommended to select all km200 data fields with a "*". The adapter then creates a km200.csv file in the ../iobroker-data/ems-esp/{instance} directory.

This file can be used the next time the adapter instance is started. Lines (fields) that are not required can be deleted in order to reduce the number of km200 fields to be read out. (Create a copy and rename the file).

The km200 Web API requires querying each individual value with its own http-get command. In systems with 2 heating circuits, this can be around 150 individual queries. A single query cycle then takes a correspondingly long time (15-40 seconds).

- KM200 polling is also a parameter (default 300 seconds) and the minimum value that can be set is 90 seconds. (see above 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 unreadable. Values are determined and updated every 15 seconds for the EMS-ESP Gateway.

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 screenshot shows the configuration interface for the EMS-ESP Gateway. It features several input fields and dropdown menus. The 'Energiestatistik berechnen' checkbox is checked. The 'Nennheizleistung' field is set to 22. The 'State für die Modulation' field is set to boiler.curburnpow. The 'State warmes Wasser aktiv' field is set to boiler.wwcharging. Below these, there are sections for 'State-Format ohne Datenbank' (set to Array von Werten) and 'Datenbank' (set to MySQL). The 'db Instanz' field is set to 0.

The nominal heat output of the boiler must be entered and the state names for the current modulation and for DHW active. The current burner, heating (CH) and WW performance (DHW) are then determined on this basis.

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

The screenshot shows the ioBroker object tree. The 'energy' object is expanded, showing three sub-objects: 'actualCHPower', 'actualDHWPowPower', and 'actualPower'. Each of these sub-objects has a 'state' property with a 'value' field. The 'actualPower' object also has a 'power' property with a 'value' field. The 'actualPower' object has a 'value' field with a value of 0 kW. The 'actualCHPower' object has a 'value' field with a value of [0,8.44,9.11,0,0] k... and a 'power' property with a value of (null) kWh. The 'actualDHWPowPower' object has a 'value' field with a value of [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0] k... and a 'power' property with a value of (null) kWh. The 'actualPower' object has a 'value' field with a value of 0 kW.

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 under _Hours
- the daily values under _Days
- the monthly values under _Months

saved / updated with database direct access. (Therefore zero). The JSON array values are updated depending on the format selected (current values first):

- hourly values under Hours
- the daily values under Days
- the monthly values under 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 for the last 12 months can be called up. The Bosch gateways calculate the consumption values using "samples" every 60 seconds. Hourly, daily and monthly values can be read out. The adapter reads these values every hour.

KM200:

<input checked="" type="checkbox"/> Energieaufzeichnungen (nur km200)	Recordings State-Format ohne Datenbank Array von Werten	Datenbank History	db Instanz 0	Multiplikator 1	<input checked="" type="checkbox"/> Samples- Interpolation
--	---	----------------------	-----------------	--------------------	---

The "Energy recordings" checkbox must be activated and the database instance (History, mySQL or InfluxDB) must be defined. The History, SQL or InfluxDB adapter must be installed and running 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 where samples are missing. The cloud apps then extrapolate the values (e.g. if 10% of the values are missing, the value is increased by 10%). This can be activated in the adapter so that the values match the app.










recordings									
dhwCircuits									
dhw1									
actualTemp	km200:recordings.dhwCircu...	<input type="radio"/> state	value	Heizungsraum	Heizung		664		
heatSources									
actualCHPower	km200:recordings.heatSour...	<input type="radio"/> state	value	Heizungsraum	Heizung		664		
actualDHWPower	km200:recordings.heatSour...	<input type="radio"/> state	value	Heizungsraum	Heizung		664		
actualPower	km200:recordings.heatSour...	<input type="radio"/> state	value	Heizungsraum	Heizung		664		
hs1									
actualPower	km200:recordings.heatSour...	<input type="radio"/> state	value	Heizungsraum	Heizung		664		
km200				Heizungsraum	Heizung				
Days	recordings days	<input type="radio"/> state		Heizungsraum	Heizung	[4.5,11,3.8,7.8,2.4....	664		
Hours	recordings hours	<input type="radio"/> state		Heizungsraum	Heizung	[0,0,0,0,0,0,2,2,2,...	664		
Months	recordings months	<input type="radio"/> state		Heizungsraum	Heizung	[0,0,0,0,0,0,0,0,0,...	664		
_Days	db daily recordings	<input type="radio"/> state		Heizungsraum	Heizung	(null) kWh	664		
_Hours	db hourly recordings	<input type="radio"/> state		Heizungsraum	Heizung	(null) kWh	664		
_Months	db monthly recordings	<input type="radio"/> state		Heizungsraum	Heizung	(null) kWh	664		
last12m	kWh total for last 12 months	<input type="radio"/> state	value	Heizungsraum	Heizung	17786 kWh	664		
heatingCircuits									
hc1									
roomtemperature	km200:recordings.heatingC...	<input type="radio"/> state	value	Heizungsraum	Heizung		664		
km200				Heizungsraum	Heizung				
Days	recordings days	<input type="radio"/> state		Heizungsraum	Heizung	[23.9,24.1,24.5,23....	664		

The object structure is specified by the Bosch API. There are consumption values and temperatures. As with the EMS-ESP, divided into hours (Hours), days (Days) and months (Months) as arrays of values and in the states _Hours, _Days and _Months directly as database entries for graphic display.

IMPORTANT: In the case of 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.

The statistics

Burner statistics can be activated and show:

statistics			
 boiler-on-1h	percentage boiler on per hour	<input type="checkbox"/> statevalue	0 %
 boiler-starts-1h	boiler starts per hour	<input type="checkbox"/> statevalue	0
 boiler-starts-24h	boiler starts per 24 hours	<input type="checkbox"/> statevalue	1
 created	Database (mySQL/InfluxDB) enabled for fiel...	<input type="checkbox"/> statevalue	true
 efficiency	boiler efficiency	<input type="checkbox"/> statevalue	0 %
 ems-read	ems read time for polling	<input type="checkbox"/> statevalue	1,181 seconds
 km200-read	km200 read time for polling	<input type="checkbox"/> statevalue	33,225 seconds
 ww-starts-1h	ww starts per hour (EMS-ESP only)	<input type="checkbox"/> statevalue	0
 ww-starts-24h	ww starts per 24 hours (EMS-ESP only)	<input type="checkbox"/> statevalue	1

- 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 was 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: ... similar for KM200
- ww-starts-1h and ww-starts-24h (only with active EMS-ESP Gateway) - boiler starts for DHW preparation

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

Condensing benefit – burner efficiency

The boiler efficiency can be calculated when the parameters are filled.
(only gas and oil boilers)

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

Consult your boiler data sheet to adjust the efficiency table accordingly.

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

<input checked="" type="checkbox"/> Kesselwirkungsgrad berechnen (Gas und Öl)									
State Modulation			State Vorlauftemp			State Rücklauftemp			
< 20°C	< 25°C	< 30°C	< 35°C	< 40°C	< 45°C	< 50°C	< 55°C	< 60°C	< 70°C
109,5	109,3	108,3	108	106,5	105,2	103	100	98	97

Changes in the states 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 from the adapter. It is possible to rebuild the status structure by deleting statuses when restarting the adapter (statuses with history / DB entries are retained and may have to be deleted manually).

After an Ems-Esp firmware update, we recommend restarting the adapter and deleting the fields. However, this can also be made up for later.

Heat requirement control in the ems-esp adapter

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

There is a separate configuration page "Heat requirement" in which 2 entry lists exist:

(New entries with the + symbol)

In the first block, the following entries are defined for each room (name freely selectable):

- *September*: State for the target temperature of the radiator / room
- *Actual temp*: State for the actual temperature of the room
- *Low delta* – Difference between settemp – actualtemp from when there is a heating requirement:
Example: target 21° - actual 20° delta 1°. If delta >= minimum delta then heating demand.
Minimum Delta = 0 means there is a heating requirement if the current temperature is equal to or lower than the target temperature.
- *Hc* : Allocation to the heating circuit (hc1 ... hc4)
- *Weight*: Weighting of the radiator / room (What is the heat output of the radiator or underfloor heating?).

In the second block, the following is specified for each heating circuit:

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

The *deskheatdemand* Switches the automatic heat requirement 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).

Instanzeinstellungen: ems-esp.0

HAUPT-EINSTELLUNGEN WÄRMEBEDARF

☐ heatdemand

+

room	set temp	actual temp	minimum delta	hc	weight	
Wintergarten	hm-rpc.0.MEQ0479199.2.SET_TEMPERATURE	hm-rpc.0.MEQ0479199.2.ACTUAL_TEMPERATURE	1.5	hc1	2	<input type="button" value="R"/> <input type="button" value="U"/> <input type="button" value="D"/>
Wohnzimmer	hm-rpc.0.MEQ0478977.2.SET_TEMPERATURE	hm-rpc.0.MEQ0478977.2.ACTUAL_TEMPERATURE	1.0	hc1	3	<input type="button" value="R"/> <input type="button" value="U"/> <input type="button" value="D"/>
Badzimmer	hm-rpc.0.MEQ0447040.4.SET_TEMPERATURE	hm-rpc.0.MEQ0447040.4.ACTUAL_TEMPERATURE	2.0	hc1	2	<input type="button" value="R"/> <input type="button" value="U"/> <input type="button" value="D"/>
Arbeitszimmer	hm-rpc.0.MEQ0450531.4.SET_TEMPERATURE	hm-rpc.0.MEQ0450531.4.ACTUAL_TEMPERATURE	1.0	hc1	2	<input type="button" value="R"/> <input type="button" value="U"/> <input type="button" value="D"/>
Wohnzimmer FB	ems-esp.0.heatingCircuits.hc2.currentRoomSetpoint	ems-esp.0.heatingCircuits.hc2.roomtemperature	0.5	hc2	5	<input type="button" value="R"/> <input type="button" value="U"/> <input type="button" value="D"/>

+

hc	weighton	weightoff	state	on	off	savesettemp	
hc1	3	2	ems-esp.0.heatingCircuits.hc1.temporaryRoomSetpoint	-1	0	<input type="checkbox"/>	<input type="button" value="R"/> <input type="button" value="U"/> <input type="button" value="D"/>
hc2	5	0	ems-esp.0.heatingCircuits.hc2.temporaryRoomSetpoint	-1	0	<input checked="" type="checkbox"/>	<input type="button" value="R"/> <input type="button" value="U"/> <input type="button" value="D"/>

SPEICHERN X SPEICHERN UND SCHLIESSEN X ABRECHNEN

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

Page10

Documentation ioBroker.ems-esp Adapter © tp1de 2023

DESKTOP-THOMASUDBROCKEN

Objekte: 1858, Zustände: 3084

ID	Name	Typ	Rolle	Raum	Funktion	Wert	Einstellun...
ems-esp							---
0							---
controls							---
hc1							---
Arbeitszimmer							---
actualtemp	actual temperature	state	value			22.9	664
deltam	minimum room delta temperature for swit...	state	value			1	664
settemp	set temperature	state	value			20	664
weight	room weight for switching off	state	value			2	664
Badezimmer							---
Wintergarten							---
Wohnzimmer							---
off	state value off	state	value			0	664
on	state value on	state	value			-1	664
state	state for heating control	state	value			ems-esp.0.heatin...	664
status	hc control status	state	value			(null)	664
weight	hc weight actual	state	value			0	664
weightoff	hc weight for switching off	state	value			2	664
weighton	hc weight for switching on	state	value			3	664
hc2							---
Wohnzimmer FB							---
off	state value off	state	value			0	664
on	state value on	state	value			-1	664
savesettemp	saved settemp when switching off	state	value			-1	664
state	state for heating control	state	value			ems-esp.0.heatin...	664
status	hc control status	state	value			(null)	664
weight	hc weight actual	state	value			0	664
weightoff	hc weight for switching off	state	value			0	664
weighton	hc weight for switching on	state	value			5	664
active	hc control active	state	value			false	664

The last state *active* is at adapter startup with the value of *heatdemand* preassigned and controls whether the heat demand-dependent regulation is active (true) or inactive (false). The value can then be set via VIS, for example. At adapter start is e.g. without set value *heatdemand* the regulation is initially inactive and can later be activated in VIS.

It is important to choose the state to be switched carefully. For example, it would be possible to switch the heating circuit on and off via summer/winter mode (e.g. km200:heatingCircuits.hc1.suWiSwitchMode)

This has the disadvantage that in the event of an adapter stop or network problems (km200 not reachable), the heating circuit may remain permanently switched off or on and must be reset manually 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 setting options are available for each heating circuit.

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

I use automatic mode so that an automatic heating circuit switch-off continues to function after the outside temperature threshold of the heating circuit has been reached, despite the active heat requirement control.

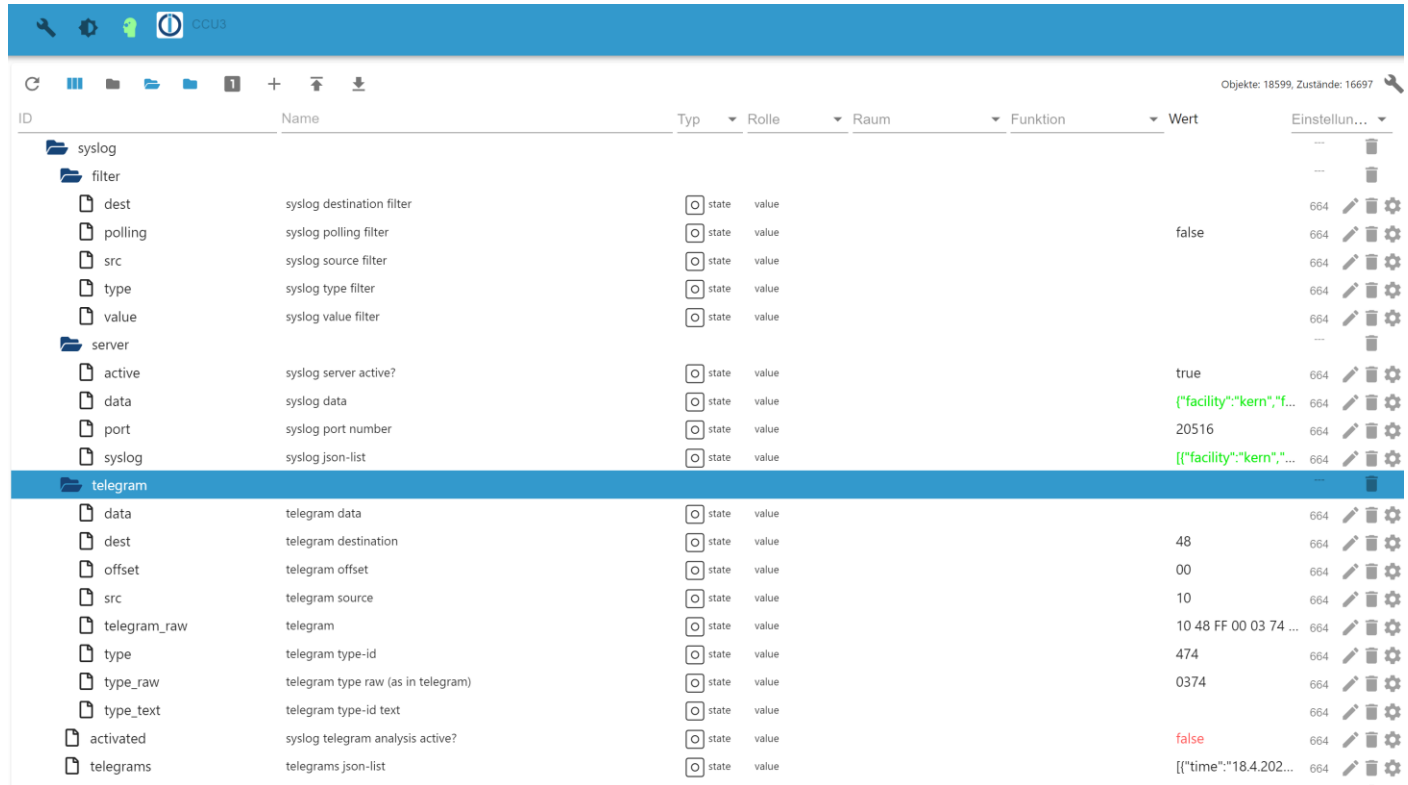
The weighting of the radiators, the switch-on threshold and the "*minimum delta*" of the heating circuit should be set in such a way that the heat output of the heat generator can be absorbed for a sufficiently long time with the smallest modulation.

At a "*minimum delta*" of 1° ensures that the valves are open with my Homematic thermostats. 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 in a room will switch on the burner.

Functions of the built-in syslog server

The syslog server runs - when switched on - permanently with and analyzes 100% of the telegrams on the EMS bus of the heating system. Please note that this is quite CPU intensive. With my Raspberry PI4 with 4 GB but no problem. On a PI3, however, I would possibly leave that at that.

The syslog program part creates the following object structure:



ID	Name	Typ	Rolle	Raum	Funktion	Wert	Einstellun...
syslog							---
filter							---
dest	syslog destination filter	<input type="checkbox"/>	state	value			664
polling	syslog polling filter	<input type="checkbox"/>	state	value		false	664
src	syslog source filter	<input type="checkbox"/>	state	value			664
type	syslog type filter	<input type="checkbox"/>	state	value			664
value	syslog value filter	<input type="checkbox"/>	state	value			664
server							---
active	syslog server active?	<input type="checkbox"/>	state	value		true	664
data	syslog data	<input type="checkbox"/>	state	value		["facility":"kern","f...	664
port	syslog port number	<input type="checkbox"/>	state	value		20516	664
syslog	syslog json-list	<input type="checkbox"/>	state	value		[["facility":"kern","...	664
telegram							---
data	telegram data	<input type="checkbox"/>	state	value			664
dest	telegram destination	<input type="checkbox"/>	state	value		48	664
offset	telegram offset	<input type="checkbox"/>	state	value		00	664
src	telegram source	<input type="checkbox"/>	state	value		10	664
telegram_raw	telegram	<input type="checkbox"/>	state	value		10 48 FF 00 03 74 ...	664
type	telegram type-id	<input type="checkbox"/>	state	value		474	664
type_raw	telegram type raw (as in telegram)	<input type="checkbox"/>	state	value		0374	664
type_text	telegram type-id text	<input type="checkbox"/>	state	value			664
activated	syslog telegram analysis active?	<input type="checkbox"/>	state	value		false	664
telegrams	telegrams json-list	<input type="checkbox"/>	state	value		[["time":"18.4.202...	664

Under "server" the incoming syslog data stream is stored.

Under "telegram" the telegram is split into the components:

- src - source
- dest – destination
- offset – from telegram
- type – telegram-type (as defined in the ems-esp project)
- type_raw: telegram type 1:1 from Telegram (why different to type ???)
- data – data part
- telegram_raw (entire telegram)
- type text – text from non-hex telegrams (conversion type in text)

Under "filter" filters for telegram analysis can be set for source, destination, type, value and if polling requests should be displayed.

It is then easy to build a simple VIS view for display. Example:

active ☐

src filter

08: Boiler

dest filter

ohne

polling ☐

type filter

Port: 20516

telegram: 08 00 FF 00 07 E4 00 00 00 30 48 00 00 00 00 00 AC

src: 08 dest: 00 type raw: 07E4 type: 8E4 offset: 00

data: 00 00 00 30 48 00 00 00 00 00

16.5.2022, 17:52:35 08 00 FF 00 07 E4 00 00 00 30 48 00 00 00 00 00 AC

16.5.2022, 17:52:35 08 00 E9 00 34 01 E2 01 E2 00 00 00 00 4B 3C 00 00 00 01 41 2F 00 07 D1 00 00 00 34 00 34 36

16.5.2022, 17:52:35 08 00 E3 00 04 00 00 00 00 00 00 00 00 00 00 01 B6 00 1E 50 00 00 00 00 00 AB

16.5.2022, 17:52:35 08 00 E4 1B 01 B1 80 00 7F FF 80 00 25

16.5.2022, 17:52:34 08 00 E4 00 10 20 30 48 00 CB 0F 01 B6 00 00 00 00 00 00 00 00 00 01 A2 00 01 10 00 01 B6 80 00 CE

16.5.2022, 17:52:34 Start

VIS example view can be imported from : <https://github.com/tp1de/ioBroker.ems-esp/blob/main/vis/telegrams.txt>
vis-inventwo adapter is needed to be installed.

Here is the example with scr filter 08 (boiler). If active is set to *true*, then the telegrams are displayed. As soon as active is set to *false*, this state is "frozen".