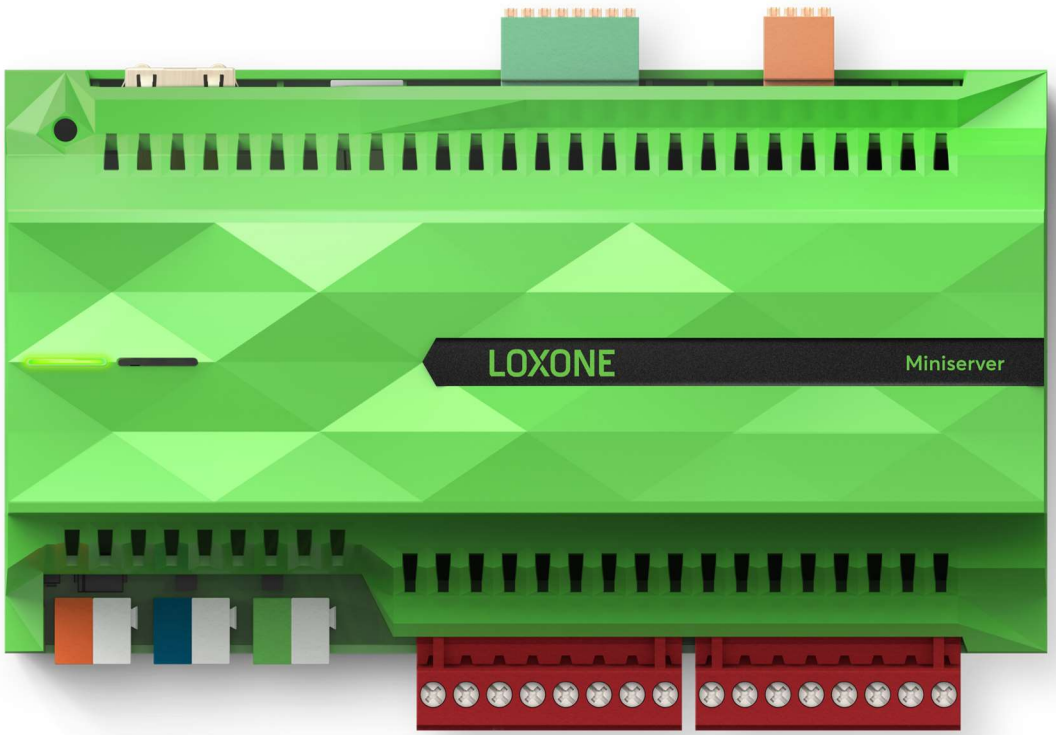
LOXONE

STRUCTURE FILE

31.3.2026



V17.0

The aim of this document is to give you a fundamental understanding on how the static structure of a Miniservers Configuration is represented. At Loxone we call this representation “Structure-File” and it is available as “LoxAPP3.json”.

In order to create a UI for remote controlling a Smart-Home different infos are required. There is the static structure which changes due to modifications in the configuration itself. On the other side there are the states that change over time due to the permanently changing environment (temperature, movement, ..) or actions taken.

While “Communicating with the Miniserver” did go into detail on how the dynamic states are structured and communicated, this document is going to focus on the fixed structure.

Table of contents

General Info

A Smart Home is a set of various sensors and actuators that are linked together by our Miniserver. Rooms and Categories are used to group these sensors and actuators (which we'll be calling controls from now on). And besides those controls and the rooms and categories which they belong to, there is some other global and external information, like the weather-server or info about the Miniserver itself. These different types of information lay out the basic structure of this file and in the next chapters, we're going to go into detail on each one of these.

lastModified

Since the Structure-File can grow rather large, it would not be a good idea to download a completely new version each time the UI is built up. So once you download the Structure-File, make sure that you cache it and save the "lastModified" attribute value. Every time you reconnect you can use the command "jdev/sps/LoxAPPversion3" to compare whether or not the Structure you've cached is still up to date.

msInfo

The msInfo area contains static information on the Miniserver and it's configuration. While some of these are pretty self explanatory, the need for others might be unclear at first.

- serialNr
  - serial number of this Miniserver
- msName
  - Name of the Miniserver as specified in the configuration document
- projectName
  - Name of the configuration document used for this Miniserver.
- localUrl
  - IP & Port that are used to connect to this Miniserver inside its local network.
- remoteUrl
  - Url/IP & Port using which the Miniserver is globally reachable.
- hostname
  - Miniserver hostname including domain
  - Only when Miniserver is set to dhcp
  - Since V13.3
- tempUnit
  - Gives info what temperature unit is used by this Miniserver.
    - 0 = °C
    - 1 = °F
- currency
  - a string containing the currency symbol to use (€, $, ...)
- squareMeasureligh
  - the unit of area (for rooms)
- location

    - The address of this Miniserver, where it is located. This info is also used for calculating the sunrise & sunset as well as for the cloud-weather.
- latitude, longitude and altitude
    - Since 12.16.14, resolved coordinates and altitude based on the location provided.
- heatPeriodStart
    - **DEPRECATED**
        - Replaced by the command jdev/sps/calendargetheatperiod
    - Month and day when the heating period starts each year, used by the intelligent room-controller.
- heatPeriodEnd
    - **DEPRECATED**
        - Replaced by the command jdev/sps/calendargetheatperiod
    - End of the heating period.
- coolPeriodStart
    - **DEPRECATED**
        - Replaced by the command jdev/sps/calendargetcoolperiod
    - Start of the cooling period.
- coolPeriodEnd
    - **DEPRECATED**
        - Replaced by the command jdev/sps/calendargetcoolperiod
    - End of the cooling period.
- catTitle
    - Top level name for all the categories, maybe there is a different kind of grouping.
- roomTitle
    - Some configurations handle the location grouping differently, they might not want to call it “Room” but “Zone” or alike.
- miniserverType
    - 0 → Miniserver (Gen 1)
    - 1 → Miniserver Go (Gen 1)
    - 2 → Miniserver (Gen 2)
    - 3 → Miniserver Go (Gen 2)
    - 4 → Miniserver Compact
- sortByRating
    - Indicates whether or not the controls are to be sorted based on their rating, which was specified in the config using the stars (0-5).
- currentUser
    - name
    - uuid - used e.g. to identify the user in user-management
    - isAdmin
    - changePassword
        - if the user is allowed to change the password via WebService
    - userRights

- List of permissions available for this user.
- Some of these may already be present in the JWT returned during auth
- Others set here can be requested explicitly using getJWT
- See “Communicating with the Miniserver” on more details (Permissions & Tokens section)

globalStates

This section lists all states that affect not only a single control but the whole Miniserver. The UUIDs here can be used to lookup the corresponding state values that arrive with all the other controls state updates.

- sunrise
  - seconds since midnight, time when the sun will rise at the Miniservers location.
- sunset
  - minutes since midnight, time when the sun will go down.
- favColorSequences
  - An array of color sequences that have been marked as favorites in any light controller v2.
- favColors
  - An array of colors that have been marked as favorites in any light controller v2
- notifications
  - Push Notifications are also sent to open WebSocket connections (if the user is allowed & registered for these notifications)
  - {

```
 “uid”: String,                 // unique message id
 “ts”: Number,                  // unix timestamp
 “type”: 10,                    // type, 10 = normal message
 “title”: String,               // title
 “message”: String,             // message
 “data”: {                      // additional data
  “lvl”: Number,                // level, 1 = Info, 2 = Error, 3 = SystemError
  [“uuid”: String]              // optional uuid (from Control, eg. Alarm)
 }
}
```

- miniserverTime
  - the current date, time & UTC-offset of the Miniserver. (e.g. “2017-07-03 13:01:36 +02:00:00”)
  - Updates only when the time is changed (e.g. shift to/from daylight saving time or manual time change).
- liveSearch
  - Json with current Information about device learning via app
- modifications
  - structural changes made via API (not via Loxone Config) are published as text events using this UUID.

operatingModes

Operating Modes are used by the Miniserver to respond to time-based events, like a weekday, weekend, vacation or alike. Mostly this is used in daytimers and the intelligent roomcontrollers.

rooms

In this section, there's a list of all the rooms that are used to group the controls in the configuration.

- uuid
  - Unique identifier for this room on this Miniserver
- name
- image
  - Icon for this Room
- defaultRating
  - Based on this number, the rooms are sorted in the UI (depending on the sortByRating-Attribute in msInfo

cats

Just like the rooms that group controls based on their location, categories group them logically.

- type
  - Categories can be given a type, which provide semantic info on what the controls in this category are for.
    - lights
    - indoortemperature
    - shading
    - media
- color
  - categories can be given a color on the UI
- colorShade
  - Since V17.0
  - Shade of the UI color which can be used for drawing the icon with a second color
  - Optional (currently only used for the light color)

weatherServer

If a Cloud Weather is configured, this section is added to the Structure-File.

states

There are only two states listed here, actual and forecast. Along with the state-updates of the other controls, the weather-updates are also delivered. See the corresponding section in the "Communicating with the Miniserver" document for details on how to parse the Weather-State-Events.

- actual
  - the current weather data for the moment
- forecast
  - the weather data for the future (for the next 96 hours)

format

This is a list on what (C-Style) formats to use for the different states.

- relativeHumidity
- temperature
- windSpeed
- precipitation
- barometricPressure

weatherTypeTexts

Each forecast and the actual weather situation has a type that is visualized differently. This section gives the user friendly texts for each of this weather situations.

weatherFieldTypes

available since Miniserver 8. Returns the possible weather field types. This types are the same as in the Loxone Config (e. g. Temperature)

times

available since Miniserver 8. Returns a list of possible time fields. This types are the same as in the Loxone Config (e. g. Minutes until sunset).

caller

available since Miniserver 8. Returns a list of configured caller service in the Loxone Config.

autopilot

available since Miniserver 8. This section is used for the autopilot generator configuration. An autopilot is a rule which can be created on the app side and is executed on the Miniserver as soon as all events of the rule are matched. The API isn't publicly available.

mediaServer

This section is present as soon as one or more Loxone Music Servers (or Castatunes Servers) are in use on this Miniserver. They are generically referred to as "mediaServer" and each one is identified by a UUID.

- type

    - 2 = Loxone Audioserver / Miniserver Compact (see subtype)
    - 1 = Loxone Music Server
    - 0 = Casatunes
- subtype (only available for type 2)
    - 0 (or missing) = Loxone Audioserver
    - 1 = Miniserver Compact
- host
    - IP and port used for communicating with the server
- MAC
    - Mac address of server
    - Available since 11.1
- localIP
    - Local resolved IP address
    - Only for Audioserver/Miniserver Compact

Loxone Audioserver

Din-Rail mounted Loxone Audioserver with a more powerful API. Clients will connect directly to the Audioserver, just like with the Loxone Music Server. When remote, a proxy on the Loxone Miniserver is used.

Miniserver Compact

The Miniserver Compact contains the audio service of the Loxone Audioserver, while the API remains the same, it is reachable on the same host as the Miniserver. For external connection, there is still a proxy to be used, as remote-connect will only make the HTTPS/WSS port reachable, not the port for the audio-web-socket.

Loxone Music Server

The Loxone Music Server provides a powerful API to clients that are connected directly to it, here are some examples:

- adding/removing and browsing external services like Spotify or Google Music
- browsing the music stored directly on the Music Server
- creating playlists
- browsing web-radio-stations
- modifying per zone favorites

This API is not covered in this documentation and is not publicly available as of now.

messageCenter

available since Miniserver 10.0. This section is used for the Systemstatus, the ultimative trouble guide for your Smart Home.

States

- changed

- Unixtimestamp when the the Systemstate entries have been modified the last time

Controls

Controls are a term that covers both actuators and sensors, simple in- or outputs and complex block-controls like an intelligent roomcontroller.

Mandatory fields

- name
- type
  - this attribute identifies what kind of control this is (Jalousie, Daytimer, ...)
  - an empty string as type indicates a control that should not be visualized.
- uuidAction
  - unique identifier for this control
- defaultRating
  - just like rooms and categories, controls can also be rated
- isSecured
  - whether or not the visualisation password has to be entered for this control

Optional fields

The optional fields might differ between the various types of controls as well as between different controls of the same type. More detailed information can be found in the documentation for the different types of controls.

- room
  - uuid of the room this control belongs to
- cat
  - uuid of the category this control belongs to
- states
  - list of state uuids for the control
- securedDetails
  - indicates that there is sensitive information available. for details see Secured Details
- details
  - visualisation details, like the format or jLockable
- statistic
  - if a control is scheduled for recording its values to a statistic, this attributes contains all info necessary.
- restrictions
  - available since 11.0
  - bitmap providing info on restrictions for this block.
    - 0 = referenced only (internal)

                - if this bit is set, it means that such a control must only be shown e.g. inside the system-scheme block, but not in a rooms list of controls.
            - 1 = read only (internal)
                - Not allowed to send any commands only visualizes values
            - 2..3 = reserved
            - 4 = referenced only (external)
                - just like bit #0, but for external connections.
            - 5 = read only (external)
                - same as bit #1
- hasControlNotes
    - available since 11.0
    - This flag indicates that there are notes/helptexts assigned to this control.
    - The webservice jdev/sps/io/{controlUUID}/controlnotes returns these notes in plaintext.
    - the string has a maximum length of 500 characters
- preset
    - available since 11.3
    - Only added if the control uses a preset
    - uuid
        - uuid of preset
        - to be used in commands like '/dev/sps/resettodefaultall/[preset uuid]'
    - name
        - Name of the preset that can be used for display
- links
    - available since 11.3
    - uuid array of linked controls
    - the order of the uuids is the order defined in Loxone config

Locking and Unlocking Controls

Available since version 11.3.2.11

All controls can potentially be (un)locked via API calls and provide their current lock state via a status. Controls that are locked can't be controlled via API or logic inputs, just like when the Reset-Input is active.

Which controls can be (un)locked via API?

Whether or not a control supports locking via API calls is defined in a controls details section, via the boolean attribute "jLockable".

How to know a control is locked and why?

Whether or not a control is locked is provided by the "jLocked" status via Status Updates. This is a text-status that, if locked, provides a JSON containing the following attributes:

- locked
  - 0 -> not locked
  - 1 -> locked by visu
  - 2 -> locked by logic
- reason
  - Text describing the reason of the lock.

Please note that if the status is an empty string it has to be assumed that the control is not locked. It may be used so that less data is transmitted.

How to (un)lock a control via API?

- Locking via Json Get request
  - Only available for admins!
  - /jdev/sps/io/{controlUuid}/lockcontrol/{0,1,true,false}/{uriEncodedReason}
    - Reason is optional
- Unlocking via Get Request
  - Only available for admins!
  - Can be used alternatively to post request with 'lock' attribute set to false
  - /jdev/sps/io/{controlUuid}/unlockcontrol

Statistic

Each control that has statistics enabled, will provide the following infos in its statistic attribute:

- frequency
  - how often is the statistic written
  - interval (8-11)
    - new as of Version 11.0
    - Will periodically write down the current value
    - When a value remains unchanged, no value will be written.
  - average (2-6)
    - deprecated as of Version 12.0
  - Possible values
    - 0 = none
    - 1 = every change
    - 2 = average per minute
    - 3 = average per 5 minutes
    - 4 = average per 10 minutes
    - 5 = average per 30 minutes
    - 6 = average per hour
    - 7 = <Reserved>
    - 8 = Interval, 5 Minutes

        - 9 = Interval, 10 Minutes
        - 10 = Interval, 30 Minutes
        - 11 = Interval, 60 Minutes
        - 12 = Interval, 15 Minutes
- outputs
    - an array of outputs for whom statistic data is recorded
    - each output has the following attributes
        - id = Index, what datapoint-row is used for this output (0-6)
        - name
        - format
            - format specifier for analog values
        - uuid
        - visuType
            - 0 = line chart
            - 1 = digital
            - 2 = bar chart

Example

The statistic infos for a line chart showing the hourly current CO2 sensor values in PPM.

```
{
 "frequency": 10,
 "outputs": [
  {
   "id": 0,
   "name": "CO2 Sensor Tree",
   "format": "%.0fppm",
   "uuid": "12917710-0034-43af-ffff11707f41df48",
   "visuType": 0
  }
 ]
}
```

Commands

- jdev/sps/getstatsdate
    - date of statistics.json file
    - Will get set after requesting the file at least once
- statistics.json
    - contains info of all stored statistics
    - Also available via HTTP-Request: {ip}/stats/statistics.json
    - Request valid via Websocket
    - Via HTTP: stats/statistics.json

- statistics.json/{controlUUID}
  - filtered statistic file
  - available since 6.1.10.16
  - Also available via HTTP-Request: {ip}/stats/statistics.json/{controlUUID}
- statisticdata.xml/{controlUUID}/{date}
  - XML file containing the statistics
  - {date} = "YYYYMM" or "YYYYMMDD"
  - Also available via HTTP-Request: {ip}/stats/statisticdata.xml/{controlUUID}/{date}
- binstatisticdata/{controlUUID}/{date}
  - returns a binary "stream", format below
  - {date} = "YYYYMM" or "YYYYMMDD"

BinaryFormat

Via websocket binstatisticdata can be downloaded. Here's how it is structured.

- ts: Uint32 (4 Bytes)
  - seconds since 1.1.2009 in local Miniserver-time
- values Float64 (8 Bytes)
  - amount of values can be found in the LoxAPP3.json for each Control (property "statistic", length of "outputs" array)
  - if multiple outputs are available (eg. meter) the order is the same as in outputs array of the control statistic details

StatisticV2

With the introduction of the energy flow monitor, a new way of handling statistics has been introduced. Meters may also support this new statistic handling.

If a control supports the new handling, the "statisticV2" property must be set in the control, containing the structure below.

Structure

- groups
  - values with equal frequency are grouped together, e.g. the actual storage power is in one group and the total stored/used energy is in a second group.
  - id
    - the id of a group, used for the getStatistics request
  - mode
    - the recording mode of the statistics
    - 0 = none
    - 1 = every change (max 1 per minute)
    - 7 = every change
    - 8 = Current value recorded every 5 minutes
    - 9 = Current value recorded every 10 minutes
    - 10 = Current value recorded every 30 minutes

        - 11 = Current value recorded every 60 minutes
        - 12 = Current value recorded every 15 Minutes
    - accumulated
        - if present & true, this indicates that this is an accumulated meter value, such as energy produced by a solar panel. This means the values are incremented over time and to display a sensible graph, the diff between those values is to be shown.
    - dataPoints
        - array of dataPoints that are part of this statistic group - see section dataPoints for more information.
        - the order of the datapoints in this array represents the order they will be returned when requested via getStatistics without filtering by outputs.

dataPoint

- title → user friendly name to show in a graph
- format → the format to use when showing the value of the datapoint
- output → the name of the output/state used for recording the values.

Example

The statistics entry of a grid meter keeping track of the consumption/production exported to or consumed.

```
{
  "groups":[
          {
          "id":"1",
          "mode":1,
    "activeSince":19989888, //Unix UTC Timestamp
          "dataPoints": [
                              {
                              "title": "Current",
                              "format": "%.3 kW",
                              "output": "actual",
                              }
                    ]
          },
          {
          "id":"2",
          "mode":2,
          "accumulated": true
          "dataPoints": [
```

```
                        {
                        "title": "Grid export",
                        "format": "%.3 kWh",
                        "output": "totalNeg"
                },
                {
                        "title": "Grid Import",
                        "format": "%.3 kWh",
                        "output": "total",
                 }

                ]
        },...
]
}
```

Commands

Unlike with the commands of the previous statistic handling, where the data could only be transferred via a websocket connection, these commands will now work using HTTP-Get requests.

Acquiring info about statistic data

Get info about statistics of a control.
Request:
jdev/sps/getStatisticInfo/{controlUuid}

Value field of answer:
[{"id":2,"activeSince":1661990400},{"id":1,"activeSince":1661990400}]

- id
  - Statistic group id
- activeSince
  - unixUtc-Timestamp, specifies since what date statistics are available

Acquiring raw statistic data

- Request
  - dev/sps/getStatistic/{controlUuid}/**raw**/{fromUnixUtc}/{toUnixUtc}/{dataPointUnit}/{groupId}/{outputName, e.g. actual}
    - returns the statistic data as recorded on the Miniserver during the specified timespan (including from & until).
    - {fromUnixUtc}
      - from what point in time onwards the statistic is requested.

        - unix timestamp in UTC, not local device or Miniserver time.
        - e.g. start of a day
            - 2022.09.01 00:00:00 in Austria (GMT+2) is 1661983200
            - 2022.09.01 00:00:00 in US (PA) (GMT-4) is 1661961600
    - {untilUnixUtc}
        - end time of statistic, still included in dataset.
        - e.g. end of a day
            - 2022.09.01 23:59:59 in Austria (GMT+2) is 1662069599
            - 2022.09.01 23:59:59 in US (PA) (GMT-4) is 1662047999
    - {dataPointUnit}
        - all = all available data points
        - hour = a new datapoint for each hour
            - e.g. sum of 00:00:00 until 00:59:59
        - day = a new datapoint for each day
            - e.g. sum of 00:00:00 until 23:59:59
        - month = a new datapoint for each month
            - e.g sum of 1.2.2022 00:00:00 until 28.2.2022 23:59:59
        - year = a new datapoint for each year
    - {groupId}
        - Which statistic group is requested
    - {outputName}
        - optional - not required, if not provided, all outputs of the group are returned
        - the name of the output for which the statistic is requested for
- Result
    - Binary content, see Binary Result

Acquiring preprocessed statistic data

- Request
    - dev/sps/getStatistic/{controlUuid}/**diff**/{fromUnixUtc}/{untilUnixUtc}/{dataPointUnit}/{groupId}/{outputName, e.g. total}
        - returns a statistic datapoint for each {dataPointUnit}, where it sums up the difference between the recorded values between {fromUnixUtc} and {untilUtcUnix}
        - {fromUnixUtc}
            - from what point in time onwards the statistic is requested.
            - unix timestamp in UTC, not local device or Miniserver time.
            - e.g. start of a day
                - 2022.09.01 00:00:00 in Austria (GMT+2) is 1661983200
                - 2022.09.01 00:00:00 in US (PA) (GMT-4) is 1661961600
        - {untilUnixUtc}
            - end time of statistic, still included in dataset.
            - e.g. end of a day

        - 2022.09.01 23:59:59 in Austria (GMT+2) is 1662069599
        - 2022.09.01 23:59:59 in US (PA) (GMT-4) is 1662047999
    - {dataPointUnit}
        - all = all available data points
        - hour = a new datapoint for each hour
            - e.g. sum of 00:00:00 until 00:59:59
        - day = a new datapoint for each day
            - e.g. sum of 00:00:00 until 23:59:59
        - month = a new datapoint for each month
            - e.g sum of 1.2.2022 00:00:00 until 28.2.2022 23:59:59
        - year = a new datapoint for each year
    - {groupId}
        - Which statistic group is requested
    - {outputName}
        - optional - not required, if not provided, all outputs of the group are returned
        - the name of the output for which the statistic is requested for
- Result
    - Binary content, see Binary Result

Examples

- request the hourly energy production of a production meter (where >0 means production) from a Miniserver located in Austria on the 1.9.2022
    - Request*:*
        - *getStatistic/**diff**/1661983200/1662069599/hour/1/total*
    - Response:
        - Binary array of entries that contain the following information:
        - [unixUtcTimeStamp][hourlyValueDiff]
            - *1661983200;0 = 0 kWh von 00:00:00 bis einschließlich 00:59:59*
            - *1661986800;0 = 0 kWh von 01:00:00 bis einschließlich 01:59:59*
            - *…*
            - *1662004800;6,5 = 6,5 kWh von 12:00:00 bis einschließlich 12:59:59*
            - *…*
            - *1662066000;0 = 0 kWh von 23:00:00 bis einschließlich 23:59:59*
- request the energy export power levels of a grid meter (where <0 means export, >0 means import) of a Miniserver located in Austria in August 2022
    - Request:
        - *getStatistic/**raw**/1661983200/1662069599/all/1/actual*
    - *Response:*
        - Binary *array of entries with the following information*
        - *[unixUtcTimeStamp][rawValue]*

- *1661983200;3,2 = at 00:00:00 3,2 kW were consumed from the grid*
- *1661983262;2,9 = at 00:01:02 2,9 kW were consumed from the grid*
- *….*
- *1662004833;-6,5 = at 12:00:33 6,5 kW were exported to the grid*
- *…*

- energy stored into a battery daily throughout august 2022 in Austria (GMT+2)
  - Request:
    - *getStatistic/**diff**/1659304800/1661983199/day/1/totalNeg*
  - Response:
    - Binary array of entries that contain the following information:
    - [unixUtcTimeStamp][dailyValueDiff]
      - *1659304800;6,5*
        - *6,5k Wh stored on 2022.08.01 from 00:00:00 to 23:59:59*
      - *….*
      - *1661896800;12,1*
        - *12,1 kWh stored, 2022.08.31 from 00:00:00 to 23:59:59*

Binary Result

Via websocket binstatisticdata can be downloaded. Here's how it is structured.

- ts Uint32 (4 Bytes)
  - unix timestamp in UTC
- values Float64 (8 Bytes)
  - amount of values per datapoint depends on the number of outputs requested
  - if multiple outputs have been requested, the values are listed in the order of the request.

Secured Details

If the flag "securedDetails" is set, this indicates that a control has sensitive information available, such as credentials for accessing an Intercoms video stream. This sensitive information needs to be requested using a separate, encrypted request.

Commands

- jdev/sps/io/{controlUUID}/securedDetails
  - Will return all securedDetails of the control

Control History

Available since version 14.5.?.?

Some control blocks support tracking a history, providing insight on why the block is currently operating in the way it does. E.g. why did the shading just close?

Blocks supporting the control block history provide a flag “hasHistory: true” within their details property. If this is missing or false, the block does not support it.

Commands

- jdev/sps/io/{controlUUID}/gethistory
    - Will return a JSON array containing the history entries.

Data Structure

- Array of JSON objects, each object contains the following properties
    - ts - unix-timestamp of the entry
    - what - String indicating the action that has been performed, e.g. “Lights off”
    - trigger - String indicating the reason as to why it has been performed, e.g. “Input off”.
    - impacts - array of strings describing what has happened due to the trigger, may be empty or missing if the trigger had no direct impact.
    - triggerType - see Trigger Types section for more information on this.
    - triggerUuid - optional, depending on the type, a uuid may give away e.g. the user that did perform the action, which can be used to link to the user management do adopt permissions.

Trigger Types

Trigger types provide a non-user friendly string identifying the cause of the action.

- user - user interacted with the block using the app.
- control - another control did cause the change of behavior (e.g. via input or central block)
- logic - the control itself
- automaticRule - an automatic designer rule did trigger it
- scene - scene did trigger it.
- centralGw - trigger came from a central block on another MS in a client gw system. trigger may be empty, if the name couldn’t be looked up by the MS
- device - a device connected to a block did trigger it.
- generic - may be something identifiable by the triggerUuid (e.g. a memory flag) but may not be visualized

Control Types

In the following sections you will find details on the different types for controls in our LoxAPP3.json. It will not provide a detailed documentation on how and what these controls are being used for. If you lack this info, please see our online documentation.

It will provide info on what commands these controls support and what states they will provide.

AalEmergency

Covered Config Items

- Emergency Alarm

States

- status
  - 0: running, normal operation, waiting for emergency button press
  - 1: alarm triggered
  - 2: reset input in config asserted, control is shut down
  - 3: app has temporarily disabled control
- disableEndTime
  - end time (unix) when control will start to operate again.
- resetActive
  - text state with the active reset input (if control is in reset)

Commands

- trigger
  - trigger an alarm from the app
- quit
  - quit an active alarm
- disable/<timespan>
  - disable the control for the given time in seconds. Set to 0 to start control again if it is disabled.

AalSmartAlarm

Covered Config Items

- AAL Smart Alarm

States

- alarmLevel
  - state of alarm
    - 0 = no alarm
    - 1 = immediate alarm
    - 2 = delayed alarm
- alarmCause
  - A string representing the last cause for an alarm

- isLocked
  - Reset active, inputs will be ignored and therefore no alarms will be executed
- isLeaveActive
  - Leave input is set, no alarms will be executed
- disableEndTime
  - End time for the control to be disabled

Commands

- confirm
  - Confirm pending alarm
- disable/{seconds}
  - Disable control for a certain period of time, no alarms will be executed
  - disable/0 will reenable the Smart Alarm
- startDrill
  - Execute test alarm

Alarm

Covered Config Items

- ∉ Burglar alarm

Details

- alert
  - Not used
- presenceConnected
  - TRUE if a presence detector is connected to DisMv

States

- armed
  - If the AlarmControl is armed
- nextLevel
  - the ID of the next alarm level
    - 1 = Silent
    - 2 = Acustic
    - 3 = Optical
    - 4 = Internal
    - 5 = External
    - 6 = Remote
- nextLevelDelay (**DEPRECATED** as of Version 13.0, use nextLevelAt)
  - The delay of the next level in seconds, this can be specified with the parameters D1 - D6 in Loxone Config. This increments every second...

- nextLevelAt
  - unix Timestamp when the next alarm level goes off
  - initial value = 0
- nextLevelDelayTotal
  - The total delay of the next level in seconds, this can be specified with the parameters D1 - D6 in Loxone Config.
- level
  - The ID of the current alarm level
    - 1 = Silent
    - 2 = Acustic
    - 3 = Optical
    - 4 = Internal
    - 5 = External
    - 6 = Remote
- armedDelay (**DEPRECATED** as of Version 13.0, use armedAt)
  - The delay of the alarm control being armed
- armedAt
  - unix timestamp when the alarm is armed
  - initial value = 0
- armedDelayTotal
  - The total delay of the alarm control being armed
- sensors (**DEPRECATED**, handled by subcontrol)
  - A string of sensors separated by a pipe ("|")
- disabledMove
  - If the movement is disabled or not
- startTime
  - timestamp when alarm started

Commands

- on
  - Arms the AlarmControl
- on/{number}
  - number can be 0 or 1
    - 0 = arm without movement
    - 1 = arm with movement
  - available since Miniserver 7.4.4.14
- delayedon
  - Arms the AlarmControl with the given delay (Parameter "Da")
- delayedon/{number}
  - number can be 0 or 1
    - 0 = delayed arm without movement
    - 1 = delayed arm with movement
  - available since Miniserver 7.4.4.14

- off
  - Disarms the AlarmControl
- quit
  - Acknowledge the alarm (quit the alarm)
- dismv/{number}
  - number can be 0 or 1
    - 0 = disable movement
    - 1 = enable movement

AlarmChain

Covered Config Items

- Alarm Sequence

States

- activeAlarmType:
  - Bitmap
    - 0: Inactive
    - 1: Acknowledged
      - user has quitted the alarm
    - 2: Alarm
      - Equals Input "A" of the AlarmSequence block
    - 4: Urgent Alarm
      - Equals Input "AU" of the AlarmSequence block
    - 8: EMS Alarm
      - Equals Input "AEs" of the AlarmSequence block
- nextAlarmLevelAt:
  - Seconds since 2009 when the next level will be escalated to
  - null if there is no next level
- activeAlarmText:
  - Json array with zero to two elements, describing the currently active alarm sequence. Can be an empty string at startup
- nextAlarmText:
  - The text describing the next alarm Sequence
  - empty if there is no next alarm
- iterationCount:
  - Counts how many iterations have already been made

Commands

- quit
  - Acknowledge the alarm (quit the alarm)

Subcontrol

- Tracker

AlarmClock

Operating mode 0, 1 and 2 are prioritized!

Covered Config Items

- ∉ Alarm clock

Details

- hasNightLight
  - If the control has a Touch Nightlight
  - Available since Miniserver 9.3
  - As of Version 13.0 this flag will always be true, even without a Touch Nightlight Air connected to the block, a default entry exists.
- brightInactiveConnected
  - Is the connector for the nightlight brightness inactive connected
  - Available since Miniserver 10.0
- brightActiveConnected
  - Is the connector for the nightlight brightness active connected
  - Available since Miniserver 10.0
- snoozeDurationConnected
- wakeAlarmSounds
  - Array of all available sounds
  - [
    - {
      - “id”: 0
      - “name”: “SOUND_NAME”
    - }
    - ...
  -]
  - Available since Miniserver 10.3
- wakeAlarmSoundConnected
  - true if the parameter input S is defined by logic
  - Available since Miniserver 10.3
- wakeAlarmVolumeConnected
  - true if the parameter input Sv is defined by logic
  - Available since Miniserver 10.3
- wakeAlarmSlopingConnected
  - true if the parameter input Sr is defined by logic
  - Available since Miniserver 10.3

States

- isEnabled
  - If the AlarmClock is enabled
- isAlarmActive
  - If an entry is ringing
- confirmationNeeded
  - If the User needs to confirm the entry
- entryList
  - Object with all the entries
    - {
      - "entryID":
        - {
          - "name": "AlarmClock1",
          - "isActive": true,
          - "alarmTime": 29940,
          - "modes": [0,1,2,3,5],
          - "nightLight: false,
          - "daily": false
        - }
    - }
    - nightLight and daily are available since Miniserver 9.3
- wakeAlarmSoundSettings
  - The settings as a JSON containing the following properties
    - sound
      - The ID of the sound found in "wakeAlarmSounds" control details
    - volume
      - The numerical volume of the wake alarm sound
    - isSloping
      - If sloping is enabled
  - Available since Miniserver 10.3
- currentEntry
  - The "entryID" of the current entry
  - -1 if there is no current entry
- nextEntry
  - The "entryID" of the next entry
  - -1 if there is no next entry
- nextEntryMode
  - Represents operating modes 3 - 9 from our structure file
- ringingTime
  - countdown in seconds how long the alarmClock will be ringing until it's going to snooze again
- ringDuration
  - The duration the AlarmClock is ringing
- prepareDuration
  - The preparation time in seconds

- snoozeTime
  - Seconds until snoozing ends
- snoozeDuration
  - Duration of snoozing
- nextEntryTime
  - Date of next entry in seconds since 1.1.2009
  - available since Miniserver 8.1
- deviceState
  - 0 = No Touch Nightlight is connected
  - 1 = Touch Nightlight is offline
  - 2 = Touch Nightlight is online
  - available since Miniserver 9.3
- deviceSettings
  - json object that is empty when no nightlight is used
  - available since Miniserver 10.0
  - {
  - “beepUsed”:true, // BOOLEAN if the Buzzer on the nightlight is used
  - “brightInactive”:0, // value - brightness of display when inactive
  - “brightActive:100 // value - brightness of display when active
  - }

Commands

- snooze
  - snoozes the current active entry
- dismiss
  - dismisses the current active entry
- entryList/put/{entryID}/{name}/{alarmTime}/{isActive}/{modes|daily}
  - Creates an entry or overrides it if the entryID is the same
  - entryID
    - The ID of the entry (Existing IDs will be overridden)
  - name
    - The name of the entry
  - alarmTime
    - Alarmtime in seconds since midnight
  - isActive
    - If the entry should be activated per default
  - modes
    - an array of mode IDs (operating modes found in our structure file)
    - **Note: Only for entries without the “nightLight” property**
      - Example: “1,3,5,6,7” -> “Holidays, Mondays, Wednesday, Thursday, Friday”
  - daily
    - Integer value

        - **Note: Only for entries with the "nightLight" property**
            - 0 = Ringing only once
            - 1 = Ringing daily
- entryList/delete/{entryID}
    - Deletes an entry with the same entryID
    - entryID
        - The ID of the entry
- setPrepDuration/{number}
    - Sets the prepare duration
        - number = The Prepare duration in seconds
- setRingDuration/{number}
    - Sets the ringing duration
        - number = The Ringing duration in seconds
- setSnoozeDuration/{number}
    - Sets the snoozing duration
        - number = The snoozing duration in seconds
        - Minimum = 60
- setBeepOn/{1 / 0}
    - Set if the buzzer on the nightlight should be used
- setBrightnessInactive/{0-100}
    - Set the display brightness of the nightlight when inactive
- setBrightnessActive/{0-100}
    - Set the display brightness of the nightlight when active

- setWakeAlarmSound/{number}

    - Number = Sound ID from details.wakeAlarmSounds

- setWakeAlarmVolume/{number}

- setWakeAlarmSlopingOn/{0 or 1}

Application

Covered Config Items

- ∉ Application

Details

- url: the defined url to the application
- image
- iconColor
    - Color of the Icon
    - Since 13.1?

V17.0

AudioZone

Covered Config Items

- ∉ Music Server Zone

Details

- server
  - the UUID of the Loxone Music Server this zone belongs to. See section on Loxone Music Server for details.
- playerid
  - the ID used to identify this zone within the Loxone Music Server
- clientType
  - 0 = physically connected
  - 1 = UPNP
- parameterConnections
  - bitmask which parameters are controlled via logic
    - ■
  - the index is in the order of the parameters of the control

States

- serverState
  - -3 = unknown/invalid zone
  - -2 = not reachable
  - -1 = unknown
  - 0 = offline
  - 1 = initializing (booting, trying to reach it)
  - 2 = online
- playState
  - -1 = unknown
  - 0 = stopped
  - 1 = paused (only Casatunes)
  - 2 = playing
- clientState
  - only used for UPNP clients!
  - 0 = offline
  - 1 = initializing (booting, trying to reach it)
  - 2 = online
- power
  - whether or not the client power is active
- volume
  - current volume

- maxVolume
  - zones can be assigned a maximum volume
- volumeStep
  - how large a single volume step is (important for button-control)
- shuffle
  - shuffle/0 = off
  - shuffle/1 = on
- sourceList
  - JSON containing all zone-favorites.
  - e.g.: { "getroomfavs_result": [{"id": 3, "type":4, "totalitems":6, "start":0, "items": [{ "slot": 1, "name": "Led Zeppelin" }, { "slot": 8, "name": "Stüberl" }, { "slot": 7, "name": "Dein Mix der Woche" }, { "slot": 2, "name": "07 Interlude 2.mp3" }, { "slot": 5, "name": "Johnny Cash" }, { "slot": 3, "name": "Große Liste" }] }], "command":"audio/cfg/getroomfavs/3/0/10" }
- repeat
  - -1 = unknown
  - 0 = off
  - 1 = repeat all
  - 2 = -not used-
  - 3 = repeat current item
- songName
- duration
  - how long the whole track is, -1 if not known (stream)
- progress
  - current position in the track, will be updated every 10 seconds
- album
- artist
- station
- genre
- cover
  - path to an image representing the current item.
- source
  - current selected source identifier (integer)
  - available since Miniserver 7.4.4.14, Music Server 1.1.4.14
- queueIndex
  - current song-index in audioqueue
  - available since Miniserver 10.3.10.5
- enableAirPlay
  - Airplay is enabled on the Musicserver
  - available since Miniserver 10.3.11.5
- enableSpotifyConnect
  - SpotifyConnect is enabled on the Musicserver
  - available since Miniserver 10.3.11.5
- alarmVolume

    - current volume for Alarm-Events
    - available since Miniserver 10.3.11.5
- bellVolume
    - current volume for Alarm-Events
    - available since Miniserver 10.3.11.5
- buzzerVolume
    - current volume for Buzzer-Events
    - available since Miniserver 10.3.11.5
- ttsVolume
    - current volume for TTS-Events
    - available since Miniserver 10.3.11.5
- defaultVolume
    - current default-volume
    - available since Miniserver 10.3.11.5
- maxVolume
    - current max-volume
    - available since Miniserver 10.3.11.5
- equalizerSettings
    - equalizersettings for this Zone
    - comma-separated list of equalizer-values
    - available since Miniserver 10.3.11.5
- mastervolume
    - mastervolume for grouped zones
    - available since Miniserver 10.5.3.24

Commands

- volume/{newVolume}
- volstep/{step}
    - increases or decreases the current volume by a step. E.g.” volstep/3” or “volstep/-3”
    - available since Miniserver 8.0
- prev
    - previous track
- next
    - next track
- play
    - (urns the client on if needed
- pause
- progress/{seconds}
- shuffle
    - toggles the shuffle state on/off
- repeat/{repeatState}
    - 0 = off

    - 1 = repeat list
    - 3 = repeat track
- on
    - turns the client on and starts playing right away
- off
    - turns the client off
- svpower/on
    - will wake the Music Server from standby
- svpower/off
    - will send the Music Server into standby
- source/{sourceNumber}
    - 1-8, starts playing the corresponding zone-favorite as specified by the app.
    - as of now, the list of zone-favorites cannot be obtained from the Miniserver via this API
- enablespotifyconnect/{1/0}
    - enables spotify Connect on the Musicserver
- enableairplay/{1/0}
    - enables Airplay on the Musicserver
- alarmvolume/{0-100}
    - set minimum-volume for Alarm-Events
- bellvolume/{0-100}
    - set minimum-volume for Bell-Events
- buzzervolume/{0-100}
    - set minimum-volume for Buzzer-Events
- ttsvolume/{0-100}
    - set minumum-volume for TTS-Output
- defaultvolume/{0-100}
    - set default-volume
- equalizersettings/{string}
    - set equalizer-settings for this zone
    - equalizersettings/0.0,12.0,7.0,0.0,0.0,8.0,0.0,0.0,0.0,2.0
- mastervolume/{0-100}
    - sets mastervolume for grouped zones

AudioZoneV2

Covered Config Items

- Music Server Zone Gen. 2

Details

- server
    - the UUID of the Loxone Music Server this zone belongs to. See section on Loxone Music Server for details.
- playerid

V17.0

    - the ID used to identify this zone within the Loxone Music Server
- clientType
    - 0 = physically connected
    - 1 = UPNP
- parameterConnections
    - bitmask which parameters are controlled via logic
    - the index is in the order of the parameters of the control
    - Parameter-Bits
        - Bit 1 = Bluetooth
        - Bit 2 = Default Volume
        - Bit 4 = Alarm Volumes (Alarm + Firealarm)
        - Bit 5 = Bell
        - Bit 6 = Buzzer
        - Bit 7 = TTS
- validOutputs
    - FALSE when no Audio output is connected and player is therefore not working
- groupObjects
    - Array of objects containing uuid of linked music player (only when in a group)
- groupName
    - Name of group (only when in a group)
- volumeModes
    - Bitmask of all used Volumemodes
        - 0: Audioserver
        - 1: External Absolute Value
        - 2: External up/down pulses
            - volup/voldown commands are to be sent instead of absolute values or volume step changes.
    - Available since 12.2
- disableFeature
    - Bitmask for disabled features
        - Bit 0: Spotify
        - Bit 1: Airplay
        - Bit 2: Bluetooth
        - Bit 3: Wlan
        - Bit 4: equalizer
- speakerConfig
    - Bitmask for information about connected Hardware
        - Value 0: no speakers
        - Bit 1: Stereo Extension(s)
        - Bit 2: Master Speakers
        - Bit 3: Audioserver Channel
        - Bit 9: Subwoofer

        - Bit 10: Client Speaker
        - Bit 11: Any Speaker
        - Bit 12: Bluetooth is supported
- playerGroup
    - Uuid of player group
    - Since V15.0
- presence
    - Bitmask that tells if presence is allowed on the control
        - Bit0: Presence is allowed by control-input DisP
        - Bit1: Presence is allowed by App-Setting
        - Bit2: Presence is used on block (Input is connected)
    - Presence on the control is allowed when both bits are active
    - Since V15.3

States

- ∉ serverState
    - -5 = server is rebooting
    - -2 = not reachable
    - -1 = unknown
    - 0 = offline
    - 1 = initializing (booting, trying to reach it)
    - 2 = online
- ∉ playState
    - -1 = unknown
    - 0 = stopped
    - 1 = paused
    - 2 = playing
- ∉ clientState
    - -5 = rebooting
    - -4 = updating
    - -2 = not reachable
    - 0 = offline
    - 1 = initializing (booting, trying to reach it)
    - 2 = online
- ∉ power
    - whether or not the client power is active
- ∉ volumeStep
    - how large a single volume step is (important for button-control)
- ∉ defaultVolume
- ∉ volume
- ∉ alarmVolume
- ∉ bellVolume

V17.0

- ∉ buzzerVolume
- ∉ ttsVolume
- ∉ maxVolume
    - Since V16.1
- ∉ presenceFrom
    - Presence simulation start time
- ∉ presenceTo
    - Presence simulation end time
- ∉ isLocked
    - Status of Reset Input

- ∉ bluetooth
    - Info if Bluetooth parameter is on/off
    - Since V15.0

Commands

- ∉ volUp
- ∉ volDown
- ∉ volume/{value}
- ∉ tts/{text}/{volume}
    - text = The Text that should be spoken
    - volume = optional
- ∉ playZoneFav/{id}
- ∉ prev
    - previous track
- ∉ next
    - next track
- ∉ play
    - (turns the client on if needed
- ∉ Pause
- ∉ bluetooth/[1/0]
    - ∉ Enable or disable bluetooth
    - ∉ Will return an error if parameter is set by logic
    - ∉ Since V15.0
- ∉ resetbluetoothpairings
    - Resets all bluetooth pairings
    - Since V15.0
- ∉ presence/{on,off}
    - Enable/disable presence functionality

CarCharger

Covered Config Items

- ∉ Wallbox Gen. 1

Details

- chargerType
  - -1 = no external charger, block only
  - 0 = KEBA
  - 1 = BMW
- limitAllowed
  - whether or not the charging limit input is used. If false, limitMode 2 (Automatic) will default to limitMode 0 (maxLImit)
  - available since Miniserver 8.0

States

- status (0=Offline, 1=Initializing, 2=Online)
- charging (0,1)
- connected (0,1)
- chargingFinished (0,1)
- power (kW)
- energySession (kWh)
- limitMode (0=Off, 1 = Manual, 2= Automatic)
- currentLimit (kW)
- minLimit (kW)
- maxLimit (kW)
- chargeDuration (Secs)
- showLoadManagement (0,1)
  - Standalone, Keba
    - always 1
  - BMW
    - "Loadmanagement with Miniserver" must be enabled on the BMW Wallbox

Commands

- charge/on (start charging)
- charge/off (stop/pause charging)
- limitMode/{mode}
  - 0 = maxLimit
  - 1 = manually with App
  - 2 = Automatic (Input (All) is used)
- limit/{limit}
  - charging limit between minLimit and maxLimit (kW)
  - changes only take every 15min affect (BMW only)

BMW Wallbox specific

Status:

- profiles (String, "|"-separated)
  - profiles must be set up on the BMW Wallbox
  - separate statistics (energy) are supported for each profile
- currentProfile (0,1)

Commands:

- profile/{profile}
  - switches the profile (0,1)

Central Objects

available since Miniserver 9.0

Covered Config Items

- ∉ CentralAlarm
- ∉ CentralAudioZone
- ∉ CentralGate
- ∉ CentralJalousie
- ∉ CentralLightController
- ∉ CentralWindow
- ∉ CentralPresence

Details

- controls - an array of JSON-Objects
  - uuid - uuid of the control
  - id - ID for selectedControls Command

Commands

- Command for selective control of objects
  - selectedcontrols/{ids - separated with comma}/{command}
  - If a selected control requires a visu password, it needs to be provided when sending the command to the Central Object.
  - If a control isn't accessible to a certain user, it also can't be controlled it via the Central Object.
  - for possible values for {command}, please see the concrete types below.
- CentralAlarm
  - on, off, quit, delayedon
- CentralAudio
  - play, pause, volup, voldown
- CentralGate
  - open, close, stop
- CentralJalousie
  - FullUp, FullDown, shade, auto, NoAuto, stop

- CentralLightController
  - on, reset
  - setMoods/{uuidLightController1}:{moodId1},{uuidLightController2}:{moodId2},...
    - sets the mood of light controller with uuidLightController1 to moodId1, uuidLightController2 to moodId2 and so on
- CentralWindow
  - toggle, fullopen, fullclose, moveToPosition, open/[on/off], close/[on/off], slightlyOpen, protection, stop

ClimateController

available since Miniserver 10.0

Covered Config Items

- ∉ ClimateController

Details

- capabilities
  - Defined, what outputs are used
  - Possible values:
    - 0 = None
    - 1 = Only heating
    - 2 = Only cooling
    - 3 = Heating and cooling
- connectedParameters
  - Bitmask which parameters are controlled by logic and may not be adapted by commands
  - Bit 0: Mode (autoMode/)
  - Bit 13: Heating Boundary (setHeatingBoundary/)
  - Bit 14: Cooling Boundary (setCoolingBoundary/)
- connectedInputs
  - Bit 0: Outdoor Temperature
  - Bit 1: Boost
  - Bit 2: Off
  - Bit 3: Additional Heating
  - Bit 4: Fan
  - Bit 5: Confirm filter change
  - Bit 6: Excess Cool
  - Bit 7: Excess Heat
  - Bit 8: Manual Heating

Commands

- resetMaintenance
  - Resets the maintenance counter
- setServiceMode/{active}
  - Activates or deactivates the service mode
  - active:
    - Please check “serviceMode”
- ventilation/{active}
  - Activates the ventilator
- autoMode/{mode}
  - Activated the automatic mode
  - mode
    - -1 = Off
    - 0 = Heating and cooling
    - 1 = heating
    - 2 = cooling
- setHeatingBoundary/{temp}

- setCoolingBoundary/{temp}

States

- controls
  - List of controls (IRCv2) added to the ClimateController
  - Possible values:
    - uuid = The uuidAction of the represented IRCv2
    - demand = The demand of the represented IRCv2
      - -1 = Cooling
      - 0 = None
      - 1 = Heating
  - Example:
    - [
      - {
        - uuid: "",
        - demand: -1, 0, 1
      - }
    -]
- currentMode
  - The current active mode
  - Possible values:
    - 0 = No requirement
    - 1 = Heating
    - 2 = Cooling
    - 3 = Heating boost
    - 4 = Cooling boost
    - 5 = Service mode
    - 6 = External Heater
- autoMode
  - -1 = Off
  - 0 = Heating and cooling
  - 1 = Heating
  - 2 = Cooling
- currentAutomatic
  - The current active automatic mode
  - Possible values:
    - 0 = Automatic like conditions
    - 1 = Automatik like average temperature
- temperatureBoundaryInfo
  - Information about the temperature boundaries
  - Possible values
    - 0 = Not enough data
    - 1 = Ok
    - 2 = No data at all
- heatingTempBoundary
  - Temperature boundary for heating
- coolingTempBoundary
  - Temperature boundary for cooling

- actualOutdoorTemp
  - The outdoor temperature
  - -1000 = No temperature available
- averageOutdoorTemp
  - the calcualted average-temperature
  - -1000 = no 48h average available yet
- overwriteReason
  - How the control is overwritten
  - Possible values:
    - 0 = Automatic
    - 1 = Boost
    - 2 = External Heater
    - 3 = Stop
    - 4 = Custom Info
- infoText
  - The name of the control connected to the currently active overwrite input
  - Possible values:
    - "..." = The name of the connected control
- serviceMode
  - What service mode setting is currently active
  - Possible values:
    - 0 = Off
    - 1 = Standby (Heating & Cooling OFF)
    - 2 = Heating On
    - 3 = Cooling On
    - 4 = Fan On
- nextMaintenance
  - Unix timestamp when the next maintenance must occur
- ventilation
  - State of Ventilation-Output
- excessEnergy
  - Bitmask State of EH / EC-Input:
  - Bit 0: EH is Active
  - Bit 1: EC is Active
- standbyReason
  - Reason why control is not heating or cooling
  - 0: not in standby
  - 1: no request from RoomControllers
  - 2: demand below threshold
  - 3: mode prevented by outdoor temperature

ClimateControllerUS

available since Miniserver 12.1

Covered Config Items

- ∉ HVAC Controller

Details

- capabilities
  - Defined, what outputs are used
  - Bitmask
    - 1 Bit = Heating
    - 2 Bit = Cooling
    - 3 Bit = Emergency Heat
    - 4 Bit = Fan (Since 15.1)
    - 5 Bit = Switch On-Threshold (Since 15.1)
    - 6 Bit = Servicemode (Since 15.1)
- type
  - 0 = Furnace (Fossil Fuel)
  - 1 = Heatpump (Fossil Fuel)
  - 2 = Heatpump (Electric)
- ctype
  - Since 15.1
  - Which control type is used for this visualization
  - Climate Controller, HVAC Controller,...
  - Used to provide proper filter-functionality in automatic designer
- connectedOutputs
  - Bitmask
  - 1 Bit = Heat Stage 1
  - 2 Bit = Heat Stage 2
  - 3 Bit = Compressor
  - 4 Bit = Cool Stage 2
  - 5 Bit = Heat Emergency
  - 6 Bit = Reversing Valve
  - 7 Bit = Fan
  - 8 Bit = Humidifer
- connectedInputs
  - Bitmask
  - 1 Bit = Mode
  - 2 Bit = Outside temperature
  - 3 Bit = Boost
  - 4 Bit = Stop
  - 5 Bit = Emergency
  - 6 Bit = Actual Humidity
- connectedParameters
  - Bitmask

- 1 Bit = Minimum off time
- 2 Bit = On Threshold
- 3 Bit = Fan overrun time
- 4 Bit = Time for second stage
- 5 Bit = Delta temp for second stage
- 6 Bit = Minimum temperature for cooling
- 7 Bit = Maximum temperature for heating
- 8 Bit = Humidity Target

Commands

- ventilation/{active}
  - Only when not defined via logic (connectedInputs)
  - 0 / false = automatic ventilation based on demand
  - 1 / true = activates the ventilator and opens all the vents in the rooms
- startVentilationTimer/{secondsSince2009Until}
  - Stop with 0
  - -1 for always on
- setMode/{mode}
  - Activated the automatic mode
  - Only when not defined via logic (connectedInputs)
  - mode
    - 0 = Off
    - 1 = Heating and cooling
    - 2 = heating
    - 3 = cooling
- startmodetimer/{mode}/{until}/{modeend}
  - When wanting to set a mode without a timer please use setMode command
  - mode = mode that should be set during the time
  - until = seconds since 2009 until the timer should run
    - Setting this value to zero stops an existing timer
    - -1 activates the mode endlessly
  - modeend = mode that should be set after the timer elapses or timer is stopped
- useEmergency/{1/0}
  - Only when not defined via logic (connectedInputs)
  - If activated emergency heat is used for heating
- startEmergencyTimer/{secondsSince2009Until}
  - Activate emergency input until the given time
  - Stop with 0
  - -1 for always on
- setMinimumTempCooling/{temp}
- setMaximumTempHeating/{temp}
- setServiceMode/{active}
  - ∉ Since 15.1
  - ∉ Activates or deactivates the service mode
  - ∉ active:

- ∉ Please check “serviceMode”

States

- mode
  - 0 = Off
  - 1 = Heating and cooling
  - 2 = Heating
  - 3 = Cooling
- controls
  - List of controls (IRCv2) added to the ClimateController
  - Possible values:
    - uuid = The uuidAction of the represented IRCv2
    - demand = The demand of the represented IRCv2
      - -1 = Cooling
      - 0 = None
      - 1 = Heating
  - Example:
    - [
      - {
        - uuid: "",
        - demand: -1, 0, 1
      - }
    -]
- currentStatus
  - The current active status/mode
  - Bitmask:
    - 1 Bit = Heating first stage active
    - 2 Bit = Heating second stage active
    - 3 Bit = Heating emergency active
    - 4 Bit = Cooling first stage active
    - 5 Bit = Cooling second stage active
    - 6 Bit = Switching mode to cool
    - 7 Bit = Switching mode to heat
    - 8 Bit = Switching delayed by reversing valve
    - 9 Bit = Switching delayed by fan overrun time
    - 10 Bit = Switching delayed by minimum off time
    - 11 Bit = Need heat but too hot outside
    - 12 Bit = Need cool but too cold outside
    - 13 Bit = Cooling demand but too less
    - 14 Bit = Heating demand but too less
- fanTimerUntil
  - Seconds since 2009 until the fan timer is running
  - -1 if the fan runs until manually put back to auto
  - 0 when the fan is on auto

- modeTimerUntil
  - Seconds since 2009 until the mode override is running
  - -1 if the mode override is active without an end specified.
  - 0 when no mode timer is active
- emergencyTimerUntil
  - Seconds since 2009 until the emergency heating is used
  - -1 for always until deactivated
  - 0 when the emergency heat is not used
- fan
  - Current status of fan output
- emergencyOverride
  - State of emergency input or value set via app when nothing is connected to input
- humidity
  - Current humidity in %
- actualOutdoorTemp
  - The outdoor temperature
  - -1000 = No temperature available
- minimumTempCooling
  - Minimum temperature to be able to cool
- maximumTempHeating
  - Minimum temperature to be able to heat
- protectionTemp
  - Optional, only available for HVAC USA (details:ctype:510)
  - If given, the value defines the lower bound for minimumTempCooling
- threshold
  - Minimum demand in percent (0-100) to start heating or cooling
- demandCool
  - Cooling demand of all room controllers in percent (0-100)
- demandHeat
  - Heating demand of all room controllers in percent (0-100)
- ExcessEnergy
  - Since V15.1
  - Bitmask State of EH / EC-Input:
  - Bit 1: EH is Active
  - Bit 2: EC is Active
- stage
  - Currently active heating/cooling stage
  - 0 - Off
  - 1 - First stage
  - 2 - Second stage

- serviceMode
  - What service mode setting is currently active
  - Possible values:

        - 0 = Off
        - 1 = Standby (Heating & Cooling OFF)
        - 2 = Heating On
        - 3 = Cooling On
        - 4 = Fan On
- outdoorTempInfo
    - Information about the used outdoor temperature
    - Possible values
        - 0 = Not enough data
        - 1 = Ok
        - 2 = No data at all
    - Since V15.1
- outdoorTempMode
    - Since V15.1
    - Info about which outdoor temperature is used
    - 0 = not used
    - 1 = average of past 48h
    - 2 = system variable Expected outdoor temperature 48h
    - 3 = current outdoor temperature

ColorPickerV2

This control type only appears as subcontrol of the LightControllerV2.

Details

- ∉ TWMin
  - Lowest color-temperature supported
  - e.g. 2700[K]
  - Avaliable since 15.0(default: 2700)
- ∉ TWMax
  - Highest color-temperature supported
  - e.g. 8500[K]
  - Avaliable since 15.0(default: 6500)
- ∉ pickerType
  - Color-picker type
  - Rgb/Lumitech
    - Support color temperature + RGB
  - TunableWhite
    - Support only color temperature

States

- ∉ color
  - The color as a string in the text property of the returned JSON
  - hsv(0,100,100) for RGB Values
  - temp(100,4483) for color temperatures (brightness, kelvin)
- ∉ sequence
  - an JSON-Object containing the description of a sequence
    - colors
      - An array of colors that will be iterated while being active
      - z.B.: [“hsv(0,100,100)”, ”hsv(100,100,100)”, “hsv(200,100,100)”]
    - interval
      - seconds how long it takes until one color changes to the next color.
      - between 60s and 3600
    - type
      - Available since version 13.0
      - type of sequence
      - 0: rgb
      - 2: daylight
    - mode
      - available for daylight-sequence (since version 13.0)
      - 4 = Daylight-sequence for direct lighting
      - 5 = Daylight-sequence for indirect lighting

- ∉ sequenceColorIdx
  - -1 if the sequence isn't active
  - otherwise it's the index of the active target color in the colors array of the current sequence.

Commands

- setFav/{favColorIdx}/{color}
  - Sets the favorite color
    - favColorIdx
      - The index of the color in the favoriteColors array
      - A new entry is created if the value already exists
      - Value is updated if it already exists
    - color
      - Either an HSV or an lumitech color string
        - hsv(0,100,100) for RGB Values
        - temp(100,4483) for brightness/colortemp
      - favorite color is deleted if color is empty
- setFavSequence/{sequenceIdx}/{duration_seconds}/{colro_n...}
  - sequenceIdx
    - Index of Sequence array in GlobalStates
  - duration_seconds
    - duration in seconds
  - color_n / Maximal 6
    - Colors separated by /
- setSequence/{duration_seconds}/{color_n...}/{startIdx}
  - duration_second
    - duration in seconds
  - color_n/ Maximal 6
    - Color separated by /
  - startIdx
    - Start index of the sequence
    - -1 if the position in the sequence should not be modified.
- setBrightness/{value}
  - updates the sequencers brightness.
- hsv({hue},{saturation},{value})
  - sets a certain color, using HSV representation
- temp({brightness},{temperature})
  - sets a light with a certain color. (warm white, cold white)
- daylight({brightness})
  - Available since version 13.0
- daylighttype({type})
  - Set direct/indirect daylight type for this channel
  - Available since version 15.1

- Types: 4(direct), 5(indirect)

Details

- ∉ pickerType
  - Rgb for RGB Pickers
  - Lumitech for Lumitech Pickers
  - TunableWhite: no RGB support, only light-temperature

ColorPicker

This control type only appears as subcontrol of the LightController.

Details

- ∉ pickerType
  - Rgb for RGB Pickers
  - Lumitech for Lumitech Pickers

States

- ∉ color
  - The color as a string in the text property of the returned JSON
  - hsv(0,100,100) for RGB Values
  - lumitech(100,4483) for Lumitech Values
- ∉ favorites
  - The favorites colors in the text property of the returned JSON
  - An array of either hsv or lumitech colors

Commands

- on
  - Enables the ColorPicker
- off
  - Disables the ColorPicker
- hsv(hue, sat, val)
  - set a new HSV color
- lumitech(brigthness,kelvin)
  - set a new color temperature, only supported for Lumitech Pickers
- setfav/{favIndex}/{color}
  - favIndex
    - The index of the favorit (1 - 4)
  - color
    - Either an HSV or an lumitech color string
      - hsv(0,100,100) for RGB Values
      - lumitech(100,4483) for Lumitech Values

Daytimer

Covered Config Items

- ∉ Schedule

Details

- analog
  - indicates if the daytimer has an analog or digital output
- text (digital only)
  - on
    - the text if the value is 1
  - off
    - the text if the value is 0
- format (analog only)
  - the format of the value

States

- modeList
  - All available modes in a proprietary list format
  - 0:mode=0;name=\"Feiertag\",1:mode=1;name=\"Urlaub\"
- mode
  - current operating mode of the daytimer
- override
  - the remaining time of the time
- value
  - current value, 0 or 1 digital and a value for analog
- entriesAndDefaultValue
  - daytimer event with entries
  - a default value (used only for analog)
- resetActive
  - stays active as long as the reset input of the daytimer is active.
  - Since Config 9.0
- needsActivation
  - Only available if the control needs to be activated

Commands

- pulse
  - activates the new value if an entry needs activation
- default
  - changes the default value in the analog daytimer
  - e.g.: default/8
- startOverride

    - starts the timer with a new value
    - e.g.: startOverride/{value}/{howLongInSecs}
- stopOverride
    - stops the timer
- set
    - change entries of the daytimer,
    - set/{numberOfEntries}/{entry}/{entry}/...,
    - {entry} = {mode};{fromMin};{toMin};{needsActivation};{valueOfEntry}
        - valueOfEntry will always be "0" in digital daytimers, or left out. Digital daytimers outputs are "On" as long as an entry exists.
        - from and to are to be given as minutes since midnight.
- modeslist
    - operating modes list sorted by the priority, on the end all weekdays from 3-9
    - 7,1,2,3,4,5,6,7,8,9

Intelligent Room Controller Daytimer v2

available since Miniserver 10.0

The daytimer used in the intelligent Room Controller v2 is an analog daytimer where the entries values identify the target temperature (identified by the temperature id, e.g. Comfort Temperature, Building Protection). This control inherits states and details from the Daytimer.

Manual Temperature Range in Schedule

New Since Config xx.xx.xx

Until now, the daytimer-entry values 0, 1 or 2 did indicate which temperature mode is active.

In order to support custom temperatures, entry values can now carry a custom temperature which also defines in which mode the IRC should operate to reach it (heating and cooling, heating only, cooling only). The following steps go into detail on how to identify those entries & extract the contained information:

UINT32 nVal = (UINT32) dVal

BYTE cMode = nVal & 0xFF → erstes Byte ist der Modus

INT16 nTemp = (int16)(nVal >> 8) → nächste 2 Byte: Target * 10

Example, 22,5°Celsius with heating only.

1. Cast the entries value to an uint32 (it is transferred as double)
    a. Double: 57.648,0
    b. UInt32: 57.648
    c. Binary: 0000 0000 0000 0000 1110 0001 0011 0000
2. If the 5th bit is set, the value contains a custom temperature and if heating and or cooling is allowed.

   a. Binary: 0011 0000
3. Heating or Cooling Mode must defined with Bit´s 6-7
   a. Bit 6 set = Heating allowed (Binary: 0011 0000)
   b. Bit 7 set = Cooling allowed (Binary: 0101 0000)
   c. Both bits set = Heating and Cooling allowed. (Binary: 0111 0000)
4. Omit the first 8 bits and extract the next 16bits.
   a. Binary: 0000 0000 1110 0001
5. These 16 Bits represent the target-temperature in INT16-Format with an Factor 10
   a. Binary 0000 0000 1110 0001
   b. INT16-Value: 225
   c. Target-Temperature: 22.5°C

Commands

- set
  - set the calendar entries
- modeslist
  - operating modes list

Intelligent Room Controller Daytimer

The daytimers used in the intelligent Room Controller are analog daytimers where the entries values identify the target temperature (identified by the temperature id, e.g. Comfort Temperature, Empty House). This control inherits states and details from the Daytimer.

Commands

- setc
  - change entries of the cooling daytimer,
- set
  - for heating daytimer
- modeslistc
  - operating modes list for the cooling daytimer, modeslist for heating daytimer

Pool Daytimer

A pool daytimer is an analog daytimer where the analog value identifies what cycle (e.g. Backwash, Filter) is used while the entry is active. This control inherits states and details from the Daytimer.

Dimmer

Covered Config Items

- ∉ Dimmer

States

- ∉ position
  - The current position for the dimmer
- ∉ min
  - The current min value
- ∉ max
  - The current max value
- ∉ step
  - The current step value

Commands

- on
  - Sets the Dimmer to the last known position
- off
  - Disables the dimmer, sets position to 0
- {pos}
  - The position of the Dimmer
  - If position if over max, max will be set and if position is over min, min will be set

EnergyManager

Covered Config Items

- ∉ Energy Manager

Details

- loads
  - Json Array containing an object for each used load
  - {"id": 0, "name": "Name of load", “minOnTime”:10}
  - minOnTime - in seconds (when zero no trigger available)

States

- currentPower
  - value of Input AIp
- currentBat
  - value of Input AIb
- loads
  - Json TextEvent
  - Array containing an object for each used load
  - {"id":0,"isActive":false,"isPreparing":false,"activeUntil":0, “isWaitingforActivation”: false, “isPermanentOn”: false}
    - id - identifier/priority of the load (starting at 0)

- isActive - Output for the load is currently active
- isPreparing - Output is active due to preparing/activation time
- activeUntil - Seconds Since 2009 until when the output is active (0 when active without end)
- isWaitingforActivation - Needs manual activation so that it can activate
- isPermanentOn - True when load is permanentely activated by input or via app

Commands

- trigger/{id}
  - Start an output for minimum time or start activation
- turnOn/{id}
  - Permanent On for load
- turnOff/{id}
  - Turn load off
  - Overrides input until next change on input

EnergyManager2

Covered Config Items

- ∉ Energy Manager Gen. 2

Details

- MinSocLocked
  - Boolean
  - Parameter MinSoc has something connect and thus can not be changed
- MaxSpwrLocked
  - Boolean
  - Parameter MaxSpwr has something connect and thus can not be changed
- HasSsoc
  - Ssoc (Storage state of charge) is connected and valid
- HasSpwr
  - Spwr (Storage power) is connected and valid

States

- Gpwr
  - value of Input Gpwr (Grid Power) [kW]
- Spwr
  - value of Input Spwr (Storage Power) [kW]
- Ppwr
  - value of Input Ppwr (Production Power) [kW]

- Ssoc
  - value of Input Ssoc (Storage State of Charge) [%]
- MinSoc
  - value of Parameter MinSoc(Minimum Storage State of Charge) [%]
- MaxSpwr
  - value of Parameter MaxSpwr (Max Storage Power) [kW]
- loads
  - Json TextEvent
  - Array containing an object for each used load
    - prio-priority of the load (starting at 0)
    - id - output index (starting at 0) - used for automatic designer actions
    - name – The name of the load
    - uuid - Unique identifier used for webservices
    - icon - path to icon. Request the icon with [miniserver address]/[icon]
    - pwr - current actual used power
      - The actual used power, this will equal the provided power if the Manager does not know the state
    - ppwr – provided power, the power which is provided by the Energy Manager
      - Since V15.1
    - hasActual – Boolean, True when the Manager knows the status of the load (e.g. digital status, or actual power)
      - False when status input of load is not connected and manager can only assume that load is active when power is provided
      - Since V15.1
    - active - Output for the load is currently active
    - activatedManually - activated manually (input or app)
    - deactivatedManually - deactivated manually (input or app)
    - minimumActiveUntil - UTC Seconds Since 2009 until when the load is active at minimum (when load has a minimum runtime)
    - activeDueToDailyRuntime - Load was activated to reach minimum daily runtime
    -

Commands

- manage
  - Do a Check of the inputs right now without waiting for any timeouts
- setMinSoc/{value}
  - Set MinSoc Parameter
  - Will only work when input is not locked (see details)
- setMaxSpwr/{value}
  - Set MaxSpwr Parameter
  - Will only work when input is not locked (see details)

V17.0

- [uuid load]/activate
  - Activate load until midnight
- [uuid load]/deactivate
  - Deactivate load until midnight
- [uuid load]/automatic
  - Set load back to automatic
  - Since V13.1.11.2
- [uuid load]/edit/...
  - Same commands and returns as in expert mode
  - [uuid load]/edit/load
    - Get edit json in the same format as expert mode
    - excluding priority setting
  - [uuid load]/edit/cancel
    - Cancel editing
  - [uuid load]/edit/verify/[setting-id]/[setting-value]
  - [uuid load]/edit/refresh
    - Refresh edit timeout
  - [uuid load]/edit/save
    - Apply changed settings
- order/[uuidPrio1],[uuidPrio2],...
- getStorageSettingDescriptions
  - json-Array containing the title, description and unit of the energy storage settings (MinSoc, MaxSpwr)

EnergyFlowMonitor

Covered Config Items

- Energy Flow Monitor

Details

- nodes
  - array of nodes to be shown in this monitor, see separate section below for more infos
- actualFormat
  - the format to use for displaying actual values, e.g. “%.3f kW” for the currently consumed power.
- totalFormat
  - used for both total & totalNeg values.
  - the format to use for displaying accumulated values, e.g. “%.3f kWh” for showing the total energy consumption of a day/hour/...
- storageFormat
  - Unit for Storage
- rest
  - Show Rest of root node

- [restName]
  - Rest name of root node
- [restIcon]
  - Icon path of root node rest
  - e.g. "IconsFilled/power-grid.svg"

Node

A node provides information about a meter or group that is to be shown in the energy flow monitor. Nodes can be built up in a tree hierarchy, where one node could contain other nodes. rest

Only meters supporting the new statistic handling are supported to be used in this block.

- Properties (properties in [brackets] are optional)
  - uuid → identifies this node in the EFM, used in getNodeState-Command
  - nodeType → identifies the type of node, the following values exist
    - Grid
      - "actual" < 0 = currently exporting to grid
        - accumulated value available via "totalNeg"
      - "actual" > 0 = currently consuming from the grid
        - accumulated value available via "total"
    - Storage
      - "actual" < 0 = storage is being charged/filled
        - accumulated value available via "totalNeg"
      - "actual" > 0 = storage is being used (discharged, relieved)
        - accumulated value available via "total"
    - Production
      - "actual" < 0 = not producing, but consuming
        - accumulated value available via "totalNeg"
        - e.g. solar panels that heat up to melt snow/ice
      - "actual" > 0 = producing, delivering
        - accumulated value available via "total"
    - Load
      - "actual" < 0 = not consuming, but providing
        - e.g. a battery or solar energy is located behind this meter - invalid configuration?
        - accumulated value available via "totalNeg"
      - "actual" > 0 = consuming
        - accumulated value available via "total"
    - Wallbox
      - same as consumption
    - Group
  - title → name to show on the node.

    - icon → the source path of the icon to use for this node
    - [actualEfmState]
        - name of the state-property of the EFM representing the live value for this node.
        - only available for root-level nodes, child nodes need to use the getNodeState-command with their uuid to request state-values.
    - [ctrlUuid]
        - uuid of the control represented by this node. Used to request statistics or the states.
        - If the ctrlUuid is missing, it means that the statistics for this node need to be collected from its child nodes.
    - [nodes]
        - List of child-nodes represented by this node.
    - [rest]
        - Show the rest
    - [restName]
        - Rest name of root node
    - [restIcon]
        - Icon path of root node reset

States

- Ppwr
    - Current production power (unit as defined in the settings of the block)
- Gpwr
    - Current grid power (unit as defined in the settings of the block)
- Spwr
    - Current storage power (unit as defined in the settings of the block)
- Pre
    - Price export per kWh
- Pri
    - Price import per kWh
- CO2
    - CO2 Factor (Kg/kWh)
- actual0 ... actualN
    - for each node that the EFM itself delivers the live state, there is such a state.
    - returns the actual value, e.g. the current production power.
    - the “actualEfmState” property of the nodes provide info on what state-property delivers the state for that node.

Commands

- getNodeValue/{viewType}/{nodeUuidList}
    - returns a JSON object, where the nodeUuid is the key and the value is a JSON with the output values for the node requested

---

        - { “node1uuid”: { “total”: 5.6, “totalNeg”: “2.1” }, “node2uuid”: {“total”: 1,2}}
        - {“node1uuid”: {“actual”: 1}, “node2uuid”: {“actual”: -1, storage: 50}}
    - this command is used to avoid live status updates of all nodes for all viewTypes, as this would cause unnecessary data transfers.
    - {nodeUuidList}
        - List of nodeUuids separated by a semicolon “;”
    - {viewType} can either be
        - actual → e.g. the current consumption power
        - day → e.g. the energy consumed on the current day
        - week → e.g. the energy consumed in the current week
        - month → e.g. the energy consumption of the current month
        - year → e.g. the energy production of the current year
        - lifetime → total/lifetime
- get/{viewType}
    - returns a JSON object, with data values which where calculated by the EFM
        - Actual request contains Gpwr, Ppwr, Spwr, Cpwr (Power of loads), Rest
        - Day, week, month,.. contain Export, Import, Production, Charge, Discharge, Consumption and Income
    - {viewType} can either be
        - see getNodeValue

Statistics

For this control, the new statistic handling has been implemented, see StatisticV2

Fronius

Covered Config Items

- ∉ Energy Monitor

States

- prodCurr
    - kW current production power
- prodCurrDay
    - kWh energy production all over the current day
- prodCurrMonth
- prodCurrYear
- prodTotal
    - kWh energy production since setting up
- consCurr

    - kW current consumption power
- consCurrDay
    - kWh energy consumed throughout the current day
- gridCurr
    - kW current grid consumption/delivery power
    - if negative, power is being delivered to the grid
    - available since Miniserver 8.1
- batteryCurr
    - kW current battery charging/usage power.
    - if negative, the battery is charging
    - available since Miniserver 8.1
- stateOfCharge
    - 0-100, represents the charging state of the battery. 100 = fully charged.
    - available since Miniserver 8.1
- earningsDay
    - how much money was earned by either consuming the produced power yourself instead of consuming it from the grid, or by exporting unused produced power to the grid. Depends on priceDelivery and priceConsumption
- earningsMonth
- earningsYear
- earningsTotal
- priceDelivery
    - Price per unit when exporting to the grid
- priceConsumption
    - Price per Unit while consuming from the grid
- co2Factor
    - How much co2 does it take to produce one kWh, used to compute CO2 savings
- generatorType
    - 0 = Fronius
        - data supplied by a Fronius generator
    - 1 = Inputs
        - data supplied by the block inputs
    - 2 = Kostal
        - data supplied by a Kostal devices
- mode
    - The mode gives info on what data sources are available on this energy monitor. As there are several combinations, a bitmask is used to determine whether or not certain sources are available.
        - ProducedPower: 0x01
        - Cons -> GridPower: 0x02
        - Batt -> BattPower: 0x04
        - ProdEnergy: 0x08

        - ConsEnergy: 0x16
        - DelEnergy: 0x32
    - Here are some examples for possible resulting values
        - 3 = Production and Consumption available
        - 1 = Production only
        - 2 = Consumption only
        - 0 = No data available
        - 7 = Production, Consumption and Battery available
        - ..
- online
    - 0 = online
    - 1 = offline

Gate

Covered Config Items

- ∉ Gate

Details

- animation
    - 0 = Garage Door
    - 1 = Single Gate opening to the left
    - 2 = Single Gate opening to the right
    - 3 = Gate opening to both sides
    - 4 = Folding door opening to the left
    - 5 = Folding door opening to the right

States

- ∉ position
    - the position from 1 = open and 0 = closed
- ∉ active
    - -1 = close
    - 0 = not moving
    - 1 = open
- ∉ preventOpen
    - 0 = not preventing opening of door
    - 1 = preventing opening of door
- ∉ preventClose
    - 0 = not preventing closing of door
    - 1 = preventing closing of door

Commands

- open
  - Opens the Gate
- close
  - Closes the Gate
- stop
  - Stops a moving gate
- forceOpen
  - Stops the gate when it is moving and then opens it
  - Since V12.1
- forceClose
  - Stops the gate when it is moving and then closes it
  - Since V12.1
- PartiallyOpen
  - Moves the gate to the defined partially open position
  - Since 14.2

- getState
  - Fetches the gate current state [direction, position]
  - Since 16.2

Heatmixer

Covered Config Items

- ∉ Mixing Valve Controller

States

- tempTarget
  - temperature the controller currently aims for
- tempActual
  - actual temperature reported by the sensor attached to the input

Hourcounter

Covered Config Items

- ∉ Maintenance counter

States

- total

    - total number of seconds the counter has been active so far
- remaining
    - how many seconds left until the next maintenance is required
    - 0 if required or overdue
- lastActivation
    - the timestamp (in seconds) when the counter was activated the last time.
    - updated on each new activation
- overdue
    - 0 if not overdue, otherwise maintenance is required
- maintenanceInterval
    - seconds until the next maintenance
- stateUnit
    - desired output unit on the UI (state values remain in seconds!)
        - 0 = seconds
        - 1 = minutes
        - 2 = hours
        - 3 = days
- active
    - 0/1, whether or not the counter is currently active
- overdueSince
    - seconds since the maintainanceInterval was exceeded
    - 0 maintenance is not required yet.

Commands

- reset
    - will cause a reset of the following values
        - remaining to maintenanceInterval
        - overdue to 0
        - overdueSince to 0
- resetAll
    - like reset, but also sets
        - total to 0
        - lastActivation to 0

InfoOnlyAnalog

Covered Config Items

∉ Virtual state

Details

∉ format

- the format of the value

States

- value
  - the current value of the virtual state

InfoOnlyDigital

Covered Config Items

- Virtual state

Details

- text
  - on
    - on text if the value is 1
  - off
    - off text if the value is 0
- image
  - on
    - uuid of the “on”-image
  - onColor
    - Color of the image. Fill the On image with this color if possible
    - Since 13.1
  - off
    - uuid of the “off”-image
  - offColor
    - Color of the image. Fill the Off image with this color if possible
    - Since 13.1
- color
  - on
    - text color if the value is 1
  - off
    - text color if the value is 0

States

- active
  - the current value of the virtual state

InfoOnlyText

Covered Config Items

- ∉ Virtual State

Details

- format (like %f)

States

- text
  - Text Event with the text

Intelligent Room Controller v2

available since Miniserver 10.0

Covered Config Items

- ∉ Intelligent room controller V2

Details

- **format**
  - defines Temperature-Format (°C or °F)
- **timerModes**
  - Defined the available timers
  - A manual mode is also available (id = 3), but not available in this structure
  - Structure description
    - name = Name of the mode
    - id = ID of the mode, defined by the Miniserver (is used in **states.activeMode**)
- **connectedInputs**
  - Bitmap; defines which temperature-settings are locked
  - Available Bits
    - Bit 1 Comfort-Temperature Heating
    - Bit 2 Comfort-Temperature Cooling
    - Bit 3 Comfort Temperature Heat+Cooling
    - Bit 4 Allowed Comfort-Tolerance
    - Bit 5 Lower absent Temperature
    - Bit 6 Upper absent Temperature
    - Bit 7 Allowe deviation absent
    - Bit 8 Shading temperature heating
    - Bit 9 Shading temperature cooling
    - Bit 10 FFrostprotect Temperature
    - Bit 11 HeatProtect Temperature
    - Bit 12 Mode input

        - Bit 13 CO2-Level
        - Bit 14 Indoor Humidity
- possibleCapabilities
    - Bitmask of possible capabilities of the room controller
        - 1 Bit = Active Heat possible
        - 2 Bit = Active Cooling possible
        - 3 Bit = Shading possible
        - 4 Bit = ventilation available
        - 5 Bit = reserved
        - 6 Bit = Passive Cooling (set with shading capability)
        -
    - Depending on the status of the HVAC/ClimateControllers the capabilities may be different during runtime (see state capabilities)

- SingleComfortTemperature
    - Since Miniserver 15.1
    - If true, the miniserver uses a fixed comfort temperature instead of separate heating/cooling temperature
- sources
    - Array containing an object for each source
    - Each object having a 'name' and 'uuid' attribute
- linkedAcControls
    - Array of UUIDs of the AC Controls linked to this IRC
    - Since Miniserver 15.1
- linkedFancoils
    - useFancoil [bool] - if it is linked to IRC
    - fanspeedSteps [int] - max number of Fan speed steps
    - Since 16.3.x.x

@States

- activeMode
    - The active mode
    - Available modes:
        - 0 = Economy
        - 1 = Comfort temperature
        - 2 = Building protection
        - 3 = Manual
        - 4 = Off
        - @TODO-combined mode can also be set here
- operatingMode
    - Available modes
        - 0 = Automatic, heating and cooling allowed
        - 1 = Automatic, only heating allowed

        - 2 = Automatic, only cooling allowed
        - 3 = Manual, heating and cooling allowed
        - 4 = Manual, only heating allowed
        - 5 = Manual, only cooling allowed
        - -1 = Off
- overrideEntries
    - Shows an array of the current active override or an empty array if no active override exists
    - Structure
        - start = Seconds since 2009
        - end = 0 if currently active or seconds since 2009
        - reason
            - 0 = None
            - 1 = Someone is present *-> Comfort mode is active*
            - 2 = Window open *-> Eco+ mode is active*
            - 3 = Comfort override
            - 4 = Eco override
            - 5 = Eco+ override
            - 6 = Prepare State Heat Up
            - 7 = Prepare State Cool Down
            - 8 = Overriden by source (source needs demand)
            - 9 = Cooling Forbidden
            - 10 = Heating Forbidden
            - 11 = Manual
            - 12 = No Cooling Available
            - 13 = No Heating Available
            - 14 = Low request
            - 15 = Pause timer
            - 16 = Different mode
            - 17 = Load shedding
            - 18 = Locked
            - 19 = Device offline
            - 20 = Different mode than central
        - level
            - -1 = none (for timers)
            - 0 = central block (Climate Controller, AC Central, ...)
            - 1 = executing block (AC Unit, Fancoil Unit, ...)
        - severity
            - -1 = none
            - 0 = info
            - 3 = error
        - source

                - the name of the source, e.g: The name of the connected input of the block or null it not available
            - isTimer
                - Indicates, that the timer has been started via the app
                - true | false
- prepareState
    - Possible Values
        - -1 = Cooling down
        - 0 = No Action
        - 1 = Heating up
- overrideReason
    - PossibleValues
        - Please refer to **overrideEntries.reason**
- tempActual
    - The current Temperature
- tempTarget
    - The current target Temperature
- comfortTemperature
    - Comfort temperature for heating
    - When using single comfort temperature: comfort temperature for heating and cooling (since 15.1)
- comfortTemperatureCool
    - Comfort temperature for cooling
    - Available since Miniserver version 12.1
- comfortTolerance
    - Since 15.1
    - The allowed comfort tolerance (±)
- absentMinOffset
    - Literally the minimal temperature offset of the economy mode
    - When using single comfort temperature: Allowed deviation in Eco-Mode (±) (since 15.1)
    -
- absentMaxOffset
    - Literally the maximal temperature offset of the economy mode
- frostProtectTemperature
    - Literally the minimal temperature of the Building protection
- heatProtectTemperature
    - Literally the maximal temperature of the Building protection
- comfortTemperatureOffset
    - The offset of the comfort temperature, if adopted. this is temporary and will reset itself once the next scheduled comfort window ends.
- openWindow

    - state of open Window
- excessEnergyTempOffset
    - available since miniserver 11.02
    - shows relative offset to planned target temperature forced by heating or cooling source (KlimaController,...)

- temperatureBoundaryInfo
    - Information about the temperature boundaries
    - Possible values
        - 0 = Not enough data
        - 1 = Ok
        - 2 = No data at all
- actualOutdoorTemp
    - The outdoor temperature
    - -1000 = No temperature available
- averageOutdoorTemp
    - the calcualted average-temperature
    - -1000 = no 48h average available yet
- currentMode
    - shows the currently active Mode
    - values identical to "operatingMode"
    - as a difference to the state "operatingMode", this state gives information about the actual mode in heating+cooling modes
- capabilities
    - State because Climate/HVAC Control may not provide cooling or heating due to current outdoor temperature
    - Bitmask of current capabilities of the room controller
        - 1 Bit = Active Heat possible
        - 2 Bit = Active Cooling possible
        - 3 Bit = Shading possible
        - 4 Bit = ventilation available
- shadingCoolTemp
    - Temperature when shading is started in mode cooling
- shadingHeatTemp
    - Temperature when shading is started in mode heating
- shadingOut
    - Status of shading output

- co2
    - Since 15.1
    - Current co2 level
- humidityActual
    - Since 15.1
    - Current indoor humidity
- fan

    - Fanspeed mode to Ac Unit (since 16.2.x.x)
- ventMode
    - Airflow mode to Ac Unit (since 16.2.x.x)
- co2Target
    - CO2 Target from Fan-based blocks [ppm] (since 16.3.x.x)

Commands

- override/{modeId}/[{until}]/[{temp}]
    - Starts a override timer
    - modeId
        - The requested modeId of the timer
    - [until]
        - Seconds since 2009 when the timer should end
    - [temp]
        - Only needed if we specifically want to set the temperature (manual mode)
- stopOverride
    - Stops a currently running override timer
- setComfortTemperature/{temp}
    - temp
        - The comfort temperature for heating
    - When using a single comfort temperature, this webservice sets the the absolute comfort temperature (since 15.1)
- setComfortTemperatureCool/{temp}
    - temp
        - The comfort temperature for cooling
    - Available since Miniserver version 12.1
- setshadingtemperaturecool/{temp}
    - Temperature when shading is started in mode cooling
    - Available since Miniserver version 12.1
- setshadingtemperatureheat/{temp}
    - Temperature when shading is started in mode heating
    - Available since Miniserver version 12.1
- setComfortTolerance/{tolerance}
    - Since 15.1
    - tolerance
        - The allowed deviation between current temperature and comfort temperature before switching mode
            - Min = 0.5, max = 3.0
- setAbsentMinTemperature/{offset}
    - offset
        - The minimum temperature in Eco-Mode

    - When using single comfort temperature: Set Allowed deviation in Eco-Mode (±) (since 15.1)
- setAbsentMaxTemperature/{offset}
    - offset
        - The maximum temperature in Eco-Mode
- setManualTemperature/{temp}
    - temp
        - The Manual Target-Temperature
- setOperatingMode/{opMode}
    - opMode
        - Please check **states.operatingMode**
- setComfortModeTemp/{temp}
    - Temporary change the comfort mode temperature. This adoption wil be reset once the next scheduled comfort-mode ends.
    - temp
        - Temperature described as an numerical value
    - *How to calculate the temp:*
        - *states.comfortTemperatur + states.comfortTemperatureOffset + step(0.5)*
        - *states.comfortTemperatur + states.comfortTemperatureOffset - step(0.5)*
- set/
    - Set calendar-Entries
- modeslist/
    - Set new Modeslist for Daytimer
- setCoolingBoundary/
    - sets the upper temperature-boundary for “only cooling allowed”
- setHeatingBoundary/
    - sets the lower Temperature-boundary for “only heating allowed
- activatePresenceSchedule
    - Available since 12.0

- setFan/{mode}
    - Sends the fan speed to be set on the connected Ac Unit Controller
        - Available since 16.2.x.x
- setAirDir/{mode}
    - Sends the airflow to be set on the connected Ac Unit Controller
        - Available since 16.2.x.x
- increasetemperature/{amount}
    - Increases target temperature by the set amount (when no amount is given default: 0.5 is used)
- decreasetemperature/{amount}
    - Decreases target temperature by the set amount (when no amount is given default: 0.5 is used)

Sub-Controls

- A daytimer to define when which temperature is to be used

Intelligent Room Controller

Covered Config Items

- ∉ Intelligent room controller

Info

Please note that the Miniserver will convert the temperatures to the unit specified on the Miniserver. This means that if Fahrenheit is set in Loxone Config, the temperatures are transmitted in Fahrenheit.

Details

- restrictedToMode
  - 0
    - Visualize heating and cooling
  - 1
    - Visualize cooling only
  - 2
    - Visualize heating only
- heatPeriodStart, heatPeriodEnd
  - Provided if this room controller is using a custom heating period. Returns the month and day when the heating period of this IRoomController will start and end. Missing if the global heating period is used.
  - Modified in Config 8.3
- coolPeriodStart, coolPeriodEnd
  - the same as heatPeriodStart/heatPeriodEnd but for the cooling period.
- temperatures
  - id's of the temperature modes to identify values from the states, e.g.
  - temperature-ids
    - 0 = Economy
    - 1 = Comfort Heating
    - 2 = Comfort Cooling
    - 3 = Empty House
    - 4 = Heat Protection
    - 5 = Increased Heat
    - 6 = Party
    - 7 = Manual
  - isAbsolute
    - is the value depending of comfort heating/cooling or an absolute value

States

- ∉ tempTarget
  - the current target temperature
- ∉ tempActual
  - the current temperature
- ∉ error
  - could be a big difference between target and actual temperature, the actual temperature is bigger than the heat protection temperature or the actual temperature is lower than the empty house temperature
- ∉ mode
  - information about the mode of the IRoomController
  - modes:
    - 0 = Automatic
      - cooling or heating depending on the heating/cooling period
      - only returned if neither a heating nor a cooling period is active
    - 1 = Automatic (currently heating),
    - 2 = Automatic (currently cooling),
    - 3 = Automatic heating,
    - 4 = Automatic cooling,
    - 5 = manual heating,
    - 6 = manual cooling
- ∉ serviceMode
  - the current service mode index
  - serviceModes:
    - 0 = off
    - 1 = heating and cooling off
    - 2 = heating on cooling of
    - 3 = heating off cooling on
    - 4 = heating and cooling on
- ∉ currHeatTempIx
  - the current heating temperature index of the temperatures
- ∉ currCoolTempIx
  - the current cooling temperature index of the temperatures
- ∉ override
  - the remaining time of the timer
- ∉ openWindow
  - if the window is currently opened
- ∉ overrideTotal
  - the total time with which the timer was started
- ∉ manualMode
  - if the user overrides with manual intervention
  - modes:
    - 0 = off

        - 1 = comfort overriding
        - 2 = economy overriding
        - 3 = timer overriding (through app)
        - 4 = movement/presence
- ∉ temperatures
    - an array of temperatures, index is the same as the id of the existing temperatures in the details object
- ∉ stop
    - While this state is on, all outputs of the room controller will remain off, regardless of the temperatures. The rest of the room controller will respond as usual.
    - available since Miniserver 9.0

Commands

- mode
    - in which mode the IRoomController should work (0-6)
        - Automatic (currently cooling or heating, nr 1 & 2) are not manually selectable, they are chosen depending on the heating/cooling period. Use 3 & 4 instead.
- service
    - to activate the service mode with an id from 0 - 4
- starttimer
    - starts the timer with a temperature id and remaining seconds
- stoptimer
    - stops the timer
- settemp
    - changes the value of an temperature with a temperature id and the new value

Sub-Controls

- ∉ 2 Intelligent room controller daytimer for heating and cooling

Intercom

Covered Config Items

- ∉ Door Controller

Details

- deviceType
    - 0 = Custom/Unknown
    - 1 = Loxone Intercom

    - 2 = Loxone Intercom XL
    - available since Miniserver 8.0
- videoInfo
    - An empty object for legacy reasons
- audioInfo
    - An empty object for legacy reasons
- The content of videoInfo and audioInfo have been moved to securedDetails in Version 8.1
- lastBellEventImages
    - true if the Miniserver does store images for the last bell event entries.
    - false or missing if it does not.
    - available since Miniserver 8.3
- showBellImage
    - Bool
    - false by default
    - true → the app shows the bell image when ringing
    - false → the app automatically starts the live stream
    - Since version 12.4.1.14

Secured Details

- videoInfo
    - alertImage
        - an (optional) path to a still image (jpg), used to save pictures for "lastBellEvents"
    - streamUrl
        - The path to either a mjpg stream or jpg images
        - the streamUrl might contain "remoteConnect" or "cloudDNS" instead of a hostname or ip address. If so, they need to be replaced as follows:
            - cloudDNS
                - Replace it with the resolved IP of the Miniserver
            - remoteConnect
                - Replace the host and the port with the current hostname & port of the Miniserver.
                - https://{adoptedip}.{snr}.dyndns.loxonecloud.com:{port}
                - The Miniserver will forward the video from the camera.
                - https is mandatory in this case
    - user
    - pass
- audioInfo
    - host
        - the host portion for making a sip-call
    - user

        - the (optional) user portion for making an sip-call
    - pass
        - Loxone Intercoms only - the password to authenticate the call
        - available since Miniserver 8.0

States

- bell
    - 0 = not ringing.
    - 1 = ringing.
- lastBellEvents
    - Text containing the timestamps for each bell-activity that wasn't answered.
    - YYYYMMDDHHMMSS, separated by a pipe
    - e.g.: "20151001074904|20151001075008|20151008171552"
    - recorded images (if available) can be accessed via "camimage/{*uuidAction*}/{*timestamp*}"
- version
    - Loxone Intercoms only - text containing the currently installed firmware versions.
- lastBellTimestamp
    - Timestamp during ringing when ringing started
    - Format: YYYYMMDDHHMMSS,
    - Same timestamp as bell image
    - From Version 12.2.9.14

Commands

- answer
    - Will deactivate the bell.

Sub-Controls

- 0-3 Outputs of type Pushbutton
    - since 8.0 outputs can handle a 'pulse' command. It was ignored in earlier versions.

IntercomV2

Covered Config Items

- Intercom

Details

- deviceType

    - 0 = Custom/Unknown
    - 1 = Intercom
- serialNo:
    - The SerialNumber of the device
- deviceUuid
    - UUID of the device object
- deviceName
    - Name of the device object
- optionsFramerate
    - Array with possible framerate options. Use the id in the setvideosettings webservices
    - e.g.:

      ```
      {
      "id": 5,
      "name": "5 fps"
      }
      ```

- optionsResolution
    - Array with possible resolution options. Use the id in the setvideosettings webservices
    - e.g.:

      ```
      {
      "id": 720,
      "name": "720p"
      }
      ```

- trustAddress
    - Is set when the intercom device originates from a trust member
    - This address must be used to connect to the intercom instead of the connected Miniserver
    - Since V12.4
- trustMember
    - Is set when the intercom device originates from a trust member
    - Trust member serial from which the intercom device originates
    - Since V12.4

States

- ∉ bell
    - 0 = not ringing.
    - 1 = ringing.
- ∉ address
    - The resolved IP address of the device
- ∉ answers
    - Array of Answers
- ∉ muted
    - Bool - Represents the Qb output of the Control Block

V17.0

- ∉ deviceState
  - StateUnknown = 0
  - StateOk = 1
  - StateRebooting = 2
  - StateInitializing = 3
- ∉ video Settings Intern
  - 4 Byte Value divided into 2 pairs
  - Each Pair is a setting
  - Byte 0 & 1: Framerate
  - Byte 2 & 3: Resolution
- ∉ videoSettingsExtern
  - 4 Byte Value divided into 2 pairs
  - Each Pair is a setting
  - Byte 0 & 1: Framerate
  - Byte 2 & 3: Resolution

Commands

- answer
  - Will deactivate the bell.
- playTts/{idx}
  - {idx}
    - The index of the answer to play
- mute/{0/1}
  - Mute/unmute the Control Block
- setAnswers/{answer0}/{answer1}/...
  - Sets all answers
  - No empty answer in the middle allowed!
- setallvideosettings/{framerate internal}/{resolution internal}/{framerate external}/{resolution external}
- setvideosettings/{internal 0/1}/{framerate}/{resolution}
- setframerate/{internal 0/1}/{framerate}
- setresolution/{internal 0/1}/{resolution}
- setnumberbellimages/{number bell images}
- getnumberbellimages

Sub-Controls

- ∉ 0-N Outputs of type Pushbutton

Irrigation

Covered Config Items

- ∉ Irrigation

Details

States

- ∉ rainActive
- ∉ expectedPrecipitation
- ∉ maxExpectedPrecipitation
    - Parameter maximum precipitation
    - When expected precipitation exceeds this value irrigation will not start when triggered by automatic rule or logic
- ∉ currentZone
    - Current active zone
    - -1 - Off
    - 0 - Zone 1
    - 1 - Zone 2
    - ...
    - 8 - All active
- ∉ zones
    - json array containing an object for each used zone
    - [{"id":0,"name":"Name of zone","duration":600}]
        - id - index/id of zone (starting at zero)
        - name - set name of zone
        - duration - ON duration in seconds
        - setByLogic - TRUE when duration is set by logic
- ∉ rainTime
    - Total seconds it was raining in the last 24 hours

Commands

- startForce
    - Start irrigation ignoring expected rain and past rain amount
- start
    - Start irrigation only when it has not rained enough and will not rain enough
- stop
    - Stops running irrigation
    - Irrigation is running when currentZone is not -1
- setDuration/{zone ID}={Duration in seconds}
    - Set duration of one zone
- setDurations/{zone ID 1}={Duration}/{zone ID 2}={Duration}/...
    - Set duration of multiple zones
- select/{zone ID}
    - Activate a zone manually
    - Activates the zone until stopped
    - 0 - Deactivate all
    - 9 - Activate all

Sub-Controls

- ∉ Tracker

Jalousie

Covered Config Items

- ∉ Blinds
- ∉ Automatic blinds
- ∉ Automatic blinds integrated
- ∉ Roof Window Shading

Details

- isAutomatic
  - If this is an Jalousie with Automatic (automatic blinds, automatic blinds integrated)
- animation
  - The animation type of the JalousieControl
    - 0 = Blinds
    - 1 = Shutters
    - 2 = Curtain both sides
    - 3 = Not supported
    - 4 = Curtain Left
    - 5 = Curtain Right
- availableModes
  - ConfigMode, not documented.
- type
  - subtype of control
    - 348: AutoJalousie
    - 502: Roof Shade

States

- up
  - Jalousie is moving up
- down
  - Jalousie is moving down
- position
  - The position of the Jalousie, a number from 0 to 1
    - Jalousie upper position = 0
    - Jalousie lower position = 1
- shadePosition
  - The orientation of the slats (important only for animation 0 = blinds), a number from 0 to 1

        - horizontal slats = 0
        - vertical slats (fully shaded) = 1
- safetyActive
    - Only used by ones with Automatic, this represents the safety shutdown
- autoAllowed
    - Only used by ones with Automatic
- autoActive
    - Only used by ones with Automatic
- locked
    - Only used by ones with Automatic, this represents the output QI in Loxone Config. Overrules the safetyActive, as not even for moving to the safety position is allowed.
- hasEndposition
    - ConfigMode, not documented.
- mode
    - ConfigMode, not documented.
- learningStep
    - ConfigMode, not documented.
- infoText
    - informs e.g. on what caused the locked state, or what did cause the safety to become active.
    - available since Miniserver 9.0
- deviceState
    - State of used device (0 - no device, 1 - offline, 2 - online)
    - available since Miniserver 11.3
- targetPosition
    - Blind target position (0 = upper position, 1 = lower position)
    - available since Miniserver 11.3
- targetPositionLamelle
    - Slats target position (0 = horizontal, 1 = vertical)
    - available since Miniserver 11.3
- adjustingEndPos
    - Contains the Bitmask when Adjusting the End-Position is active
    - Bits are described in Command “endPosAdjustment”

Commands

- up
    - Sends an Up command to the Jalousie, as long as no UpOff or other direction command has been sent the Jalousie will go UP
- UpOff
    - Sends an UpOff command to the Jalousie, this will stop the up motion of the Jalousie
- down

    - Sends an Down command to the Jalousie, as long as no DownOff or other direction command has been sent the Jalousie will go Down
- DownOff
    - Sends an DownOff command to the Jalousie, this will stop the down motion of the Jalousie
- FullUp
    - Triggers a full up motion
- FullDown
    - Triggers a full down motion
- shade
    - Shades the Jalousie to the perfect position
- auto
    - Enables the Automatic of the Jalousie (if the control supports it)
- NoAuto
    - This disables the Automatic mode for the Jalousie (if the control supports it)
- manualPosition/{targetPosition}
    - when received, the jalousie will move to the targetPosition provided
    - targetPosition
        - 0 = fully open
        - 100 = fully closed
- manualLamelle/{targetPosition}
    - when received, the slats will rotate to this position
        - 0 = horizontal
        - 100 = vertical (user facing edge higher than window facing edge)
- manualPosBlind/{targetPosition}/{targetLamellePosition}
    - combination of manualPosition and manualLamelle
- stop
    - immediately stops the position, remains in current position
- setAdjustingEndPos/{0/1}
    - Used to activate end position adjustment mode. Activating the mode is required to be able to use “endPosAdjustment”
    - Requires expert mode permission.
        - 0 = off, both the up and down output turn off and the shading is back to normal operation
        - 1 = on, the command “endPosAdjustment” can be used to activate the up and down outputs.
    - The adjust mode has to be kept alive, by resending it every 5000ms
        - Otherwise, the adjust mode will be disabled and the outputs will turn off 15000ms (15s) after the last command.
- endPosAdjustment/{outputBitmask}.
    - Only allowed, when adjustingEndPos is active, which in turn can be activated using the “setAdjustingEndPos”-Command
    - Bits
        - 0x00 → Up & Down off.

        - 0x01 → Down output on
        - 0x02 → Up output on
    - Active output commands must be repeated each 200ms, otherwise, the outputs will turn off after 600ms (safety precaution)
    - Examples
        - endPosAdjust/0 → both outputs are off
        - endPosAdjust/1 → up output is on
        - endPosAdjust/2 → down output is on
        - endPosAdjust/3 → up and down outputs are on

NFC Code Touch

Covered Config Items

- NFC Code Touch

Details

- accessOutputs
    - ⊄ object with output names, q1-q6
    - ⊄ if output isn't used, it isn't listed
- place: optional field with the installation place string if the object has one
    - ⊄ available since 10.2
- twoFactorAuth: optional field, true if device requires two authentifications for access
    - ⊄ available since 10.3
- keyPadAuthType: authentication type int
        - available since v 13.2
        - Deprecated since v 15.0
    - ⊄ 0 = 2FA
    - ⊄ 1 = Code or Nfc
    - ⊄ 2 = Nfc
    - ⊄ 3 = Code

States

- historyDate
    - ⊄ unix timestamp in milliseconds from the latest history entry
    - ⊄ is 0 if no history exists
    - ⊄ reload the history if this updates
- codeDate
    - ⊄ unix timestamp in milliseconds from the latest change of some code
    - ⊄ the codes has changed, reload them
- deviceState
    - ⊄ available since 10.2

- ⊄ bitmap: determines if the linked device has the capability to learn nfc tags.
  - bit 0: offline
  - bit 1: dummy
  - bit 2: nfc unavailable (battery powered air device)
- nfcLearnResult
  - ⊄ available since 10.2
  - ⊄ json array with nfc objects:
  - ⊄ nfc structure:
    - name - name of nfc tag, or 0
    - id - Tag Id, always given.

      In case of error:

      - 00 00 00 00 00 00 00 00 E8 read error
      - 00 00 00 00 00 00 00 00 EE authentication error
      - 00 00 00 00 00 00 00 00 EF init error
      - 00 00 00 00 00 00 00 00 init error (for backwards compability)
    - userUuid- uuid of the user this is linked to, or 0
    - deviceUuid - uuid of a nfc code touch this is linked to, or 0

- keyPadAuthType
  - ⊄ Available since V15.0
  - ⊄ 0 = 2FA
  - ⊄ 1 = Code or Nfc
  - ⊄ 2 = Nfc
  - ⊄ 3 = Code
  - ⊄ 4 = OCPP

Commands

- output/{outputNr}
  - outputNr can be 1-6
  - sends an impuls to the specific output
  - Code 423
    - user is not permitted at the moment
- history
  - returns JSON array with history entry objects
  - entry structure:
    - ts
      - unix timestamp in utc
    - output
      - 1-6 for standard operation
      - 1-9999 with additional preselection enabled
      - 0 for first factor of two factor authentication

        - available since 10.3
      - -1 for denied access
    - type
      - int, 0-11
      - 0 = code (can be associated with one user)
      - 1 = code (with name)
      - 2 = code (ambiguous, used from multiple users)
      - 3 = nfc (can be associated with one user)
      - 4 = nfc (with name)
      - 5 = nfc (ambiguous, used from multiple users)
      - 6 = via app
      - 7 = denied code
      - 8 = denied NFC Tag
      - 9 = denied app access attempt
      - 10 = allowed external (available since V15.1)
      - 11 = denied external (available since V15.1)
      - 12 = keycode disabled
      - 13 = nfc disabled
    - user
      - string, optional, name of user (if type is 0, 3, 6, 10, 11)
      - if user of trust member, trust member name
    - description
      - string optional
        - if type 1: name of code
        - if type 3 or 4: name of tag
    - extMedium
      - Only for types 10 and 11
      - optional, e.g. “Card” or “OCPP-ID” or “NFC Tag”
    - extAuthority
      - Only for types 10 and 11
      - e.g. “Server”, “Charging Station Management”
- codes
  - JSON array with Code objects from this keypad
  - Code structure:
    - uuid
      - string
    - name
      - string
    - isActive
      - boolean, if code is currently active
    - isEmpty
      - boolean, if code is not yet given
    - type
      - int, 0-5

                - 0 = permanent code
                - 1 = one-time code
                - 2 = time-dependent code
                - 3 = valid until code
                    - available since v 13.2
                - 4 = valid from code
                    - available since v 13.2
                - 5 = deactivated code
                    - available since v 13.2
            - outputs
                - permitted outputs this code can be used for
                - bitmask
                    - 1 = q1, 2 = q2, 3 = q1+q2, 4 = q3, 5 = q1+q3, 6 = q2+q3...
            - standardOutput
                - int, 1-6
            - timeFrom
                - optional, if type is 2
                - unix timestamp in utc
            - timeTo
                - optional, if type is 2
                - unix timestamp in utc
- code/create/{name}/{code}/{type}/{outputs}/{standardOutput}
    - creates a new code with the following properties:
    - name
        - string, URIComponent encoded
    - code
        - string (numeric)
    - type
        - int, 0-2 (see Code structure above)
    - outputs
        - bitmask (see Code structure above)
    - standardOutput
        - int, 1-6
- code/create/{name}/{code}/{type}/{outputs}/{standardOutput}/{timeFrom}/{timeTo}
    - same as previous command but for codes with type 2:
    - timeFrom
        - unix timestamp in utc
    - timeTo
        - unix timestamp in utc
- code/update/{uuid}/{isActive}/{name}/{code}/{type}/{outputs}/{standardOutput}
    - updates the code with the following properties:
    - uuid
        - string, code uuid

    - isActive
        - true/false, if the code should be activated or deactivated
    - code
        - code or -1 if code does not change
- code/update/{uuid}/{isActive}/{name}/{code}/{type}/{outputs}/{standardOutput}/{timeFrom}/{timeTo}
    - updates the code with the following properties:
    - timeFrom
    - timeTo
- code/activate/{uuid}
    - activates code
- code/deactivate/{uuid}
    - deactivates code
- code/delete/{uuid}
    - deletes the code
- nfc/startlearn
    - starts a nfc learn session. Returns 423 if device is not capable of executing the command (device offline, not configured or battery powered). Returns 500 in every other error case (User not admin, user has no visu pwd...)
    - available since 10.2
- nfc/stoplearn
    - stops a nfc learn session
    - available since 10.2

LightController

Covered Config Items

- ∉ Lighting controller
- ∉ Hotel lighting controller

Details

- movementScene
    - The scene number that is assigned as the movement scene

States

- ∉ activeScene
    - The current active scene number
- ∉ sceneList
    - Returns a JSON-Object in the following format
    - uuid -> the UUID of the LightControl
    - uuidIcon -> This is only used by the Status Control
    - text -> This is an array of Scenes separated by a “,”
        - "1=\"Szene1\",2=\"Szene2\",7=\"Szene7\""

Commands

- off
  - Enables lightscene 0 (All off)
- on
  - Enables lightscene 9 (All on)
- {sceneNumber}
  - This will activate the scene
- {sceneNumber}/learn/{sceneName}
  - This learns the current output values to the given scene and overrides it if the scene already exists. Also used to rename a scene or create a new one.
- {sceneNumber}/delete
  - This deletes the given scene
- plus
  - Changes to the next scene
  - available since Miniserver 8.1
- minus
  - Changes to the previous scene
  - available since Miniserver 8.1

LightControllerV2

Covered Config Items

- ∉ Lighting Controller

Details

- masterValue
  - the UUID of the master brightness in the subcontrols
  - will always be present.
- masterColor
  - the UUID of the master rgb in the subcontrols
  - will only be present if there are subcontrols of the type colorPicker

States

- ∉ activeMoods
  - A list of mood ids that are currently active (JSON-Array)
- ∉ moodList
  - Returns an array with JSON-Objects each containing the attributes listed below. The position in the array reflects the list order.
  - name
    - The userfriendly name for this mood.
  - shortName
    - Short name (used e.g. for Touch Flex display)

        - Since version 12.4.2.24
    - id
        - An ID that uniquely identifies this mood (e.g. inside activeMoods) (e.g. ID1, ID2,..)
    - t5
        - whether or not this mood can be controlled with a t5 input
    - static
        - If a mood is marked as static it cannot be deleted or modified in any way. But it can be moved within and between favorite and additional lists.
    - used
        - Bitmask that tells if the mood is used for a specific purpose in the logic. If it's not used, it can be removed without affecting the logic on the Miniserver.
            - 0: not used
            - 1: this mood is activated by a movement event
            - 2: assigned to a T5/1-n input and there is logic or a touch connected to
            - 4: this mood is activated by the Alarm Clock
            - 8: assigned to a P/1-n input and there is logic or a presence/movement detector connected to it
                - Supported from config V12.xx on
- ∉ favoriteMoods
    - [ID1,ID12,ID4,ID5]
    -
- ∉ additionalMoods
    - [ID2,ID3,ID6,ID7,ID8,ID9,ID10,ID11,ID13]
- ∉ circuitNames
    - Available since version 11.3.1.26
    - This is used for dynamic names of the subcontrols
    - Name of light circuits in a key value json
- ∉ daylightConfig
    - Available since version 13.0
    - from →minutes since midnight
    - until → minutes since midnight
    - mode
        - 0 → sunrise / sunset
        - 1 → custom time
    - type
        - Key-value json if daylight-lighting-type per light circuit:
        - Values: 4: direct, 5: indirect

        - Contains only circuits which supports daylight controls

- ∉ presence
    - Bitmask that tells if presence is allowed on the control
        - Bit0: Presence is allowed by control-input DisP
        - Bit1: Presence is allowed by App-Setting
        - Bit2: Presence is used on control (any presence or movement input connected)
    - Presence on the control is allowed when both bits are active

Commands

- changeTo/{moodId}
    - This will activate the mood, other moods will be deactivated
    - The moodID 0 is always “off”
- addMood/{moodId}
    - adds the provided mood and mixes it with the other moods that are active
- removeMood/{moodId}
    - deactivates the provided mood, leaves the other moods untouched.
- moveFavoriteMood/{moodId}/{newIdx}
    - moves a mood within the favorite mood list
    - moodId
        - Id of the mode
    - newIdx
        - New Index of the mood
- moveAdditionalMood/{moodId}/{newIdx}
    - Moves a mood within the additional mood list
    - moodId
        - Id of the mode
    - newIdx
        - New Index of the mood
- moveMood/{moodId}/{newIdx}
    - Moves a mood within the mood list
    - moodId
        - id of the mode
    - newIdx
        - New Index of the mood
- addToFavoriteMood/{moodId}
    - Moves a mood to the favorite mood (last index)
    - moodId
        - Id of the mode

- removeFromFavoriteMood/{moodId}
    - Moves a mood from the favorite mood list to the additional mood list
    - moodId
        - Id of the mode
- learn/{moodId}/{moodName}
    - This learns the current output values to the given mood and overrides it if the mood already exists. Also used to rename a mood or create a new one.
    - moodIds < 8 are linked to the appropriate T5 Input
- delete/{moodId}
    - This deletes the given mood.
- plus
    - Changes to the next mood
- minus
    - Changes to the previous mood
- setmoodid/{currentID}/{newID}
    - changes the id of an existing mood
    - if a mood with newID exists, moodIDs are exchanged
- setmoodshortname/{moodId}/{new name}
    - Set the short name of a mood
    - since version 12.4.2.24
- setmoodname/{moodId}/{new name}
    - Set the name of a mood
- setcircuitnames/{json key/value pairs}
    - Available since Miniserver version 11.3.1.26
    - Json Key must be subcontrol IDs, for example
        - {"AI1":"Name circuit 1", "uuidaction/AI2":"Name circuit 2"}
- setdaylightconfig/{json}
    - available since Version 13.0
    - from →minutes since midnight
    - until → minutes since midnight
    - mode
        - 0 → sunrise / sunset
        - 1 → custom time

- presence/{on,off}

    - Enable/disable presence functionality

Subcontrols

- There is one subcontrol per light circuit attached to this block and, if available, Master-Brightness and/or Master-Color.
- Types of subcontrols
    - Dimmer:
        - Master-Brightness, if available

        - Light-Circuits of any dimmer type and suitable smart actuators
    - ColorPickerV2:
        - Master-Color, if available
        - Light-Circuits of type RGB, Lumitech and suitable smart actuators
    - Switch:
        - Light-Circuits of type switch
- **Commands**
    - setName/[new Name]
        - Available since Miniserver version 11.3.1.26
        - Set name of light-circuit, if setting names of more light circuits please use 'setcircuitnames' command

LightsceneRGB

Covered Config Items

- ∉ RGB lighting controller

Details

- sceneList
    - An array of scene names

States

- ∉ activeScene
    - The current active scene number
- ∉ color
    - A string of the current color
        - hsv(0, 100, 100)

Commands

- off
    - Enables lightscene 0 (All off)
- on
    - All on
- {sceneNumber}
    - this will activate the scene
- {sceneNumber}/learn
    - this will override the selected scene with the new selected color If this scene does not exist no new scene will be created!

LoadManager

Available Since 12.1.5.25

Covered Config Items

- Load Manager (Last Manager)

Details

- loads
  - Json array containing an object for each load:
  - { "id": Index of the load, "name":"Name of load", "power": Power of the load in kW, "hasStatus": boolean if the status in the status bitmask is valid and should be displayed }
  - idx=0 is the first load that will be locked
- mode
  - 0 = Overload Manager, 1 = Peak Manager, 2 = Peak Overload Manager
  - Since V15.3

States

- currentPower
  - Current power in kW (rounded to two decimal places) in 'Overload' mode
  - Average power in kW (rounded to two decimal places) in 'Peak' mode
- peakOverloadPower
  - Current power in kW (rounded to two decimal places) in 'Peak Overload' mode
  - Since V15.3
- maxTp
  - Maximum technical power in kW (rounded to two decimal places) from the block parameter in 'Peak Overload' mode
  - Since V15.3
- maxPower
  - Maximum power in kW (rounded to two decimal places)
- availablePower
  - Remaining free Power in kW (rounded to two decimal places)
- maxPowerExceeded
  - TRUE when maximum power is reached
- lockedLoads
  - Bitmask of each locked load
  - 1 -> locked
  - First Bit -> First Load
- statusLoads
  - Bitmask of each active load
  - 1 -> active
  - First Bit -> First Load
  - Only display status if 'hasStatus' attribute of load is true

MailBox

Available Since 10.5.1.14

Covered Config Items

- Paketsafe

Details

- deviceType
  - 0 -> no device
  - 1 -> Air device
  - 2 -> Tree device

States

- notificationsDisabledInput
  - State of the notifications disabled input
- packetReceived
  - State if a packet has been received
- mailReceived
  - State if mail has been received
- disableEndTime
  - UTC timestamp until the notifications are disabled in seconds since 2009

Subcontrol

- Tracker

Commands

- confirmPacket
  - Confirm receive of a packet
- confirmMail
  - Confirm receive of mail
- disableNotifications/[seconds]
  - Disable the notifications for x seconds
  - 0 seconds for cancelling timer

Meter

Covered Config Items

- ∉ Utility meter

Details

- actualFormat
  - format specifier for the value of “actual”
- ∉ totalFormat
  - format specifier for the value of “total”
- ∉ [type]
  - available since 13.01
  - values:
    - unidirectional = default, if missing
    - bidirectional
    - storage
    - legacy (old meter objects)
- ∉ [totalFormatNeg]
  - unavailable if unidirectional
  - available since 13.01
- ∉ [storageFormat]
  - only available for storage types
  - available since 13.01
- ∉ [storageMax]
  - available since 13.01
  - this equals 100% filled storage
- ∉ [displayType]
  - available since 13.01
  - how the app should represent this meter (a container, a battery, ...)
  - “regular” or missing → regular meter UI.
  - Values:
    - 0: regular
- ∉ [powerName]
  - Display Name of ‘Power or flow’

States

- actual
- total
- [totalNeg]
  - available since 13.01
  - only available if type is not unidirectional
- [storage]
  - available since 13.01
  - only available if type is storage
- [totalDay]
  - Meter reading today
- [totalWeek]
  - Meter reading this week
- [totalMonth]

V17.0

    - Meter reading this month
- [totalYear]
    - Meter reading this year
- [totalNegDay]
    - Meter reading today
    - only available if type is not unidirectional
- [totalNegWeek]
    - Meter reading this week
    - only available if type is not unidirectional
- [totalNegMonth]
    - Meter reading this month
    - only available if type is not unidirectional
- [totalNegYear]
    - Meter reading this year
    - only available if type is not unidirectional

Commands

- reset
    - Resets all values to 0

Statistics

Since 13.01 instances of this control may support the new StatisticV2 handling. Check the JSON property “statisticV2” vs “statistic” to check what type of statistic is supported.

MsShortcut

Covered Config Items

- ∉ Miniserver Shortcut

Details

- ∉ trust
    - Boolean – True when its a Trust Link
    - Get the address in this case from the Trust Link API
- ∉ serialNr
    - Serial number of the Miniserver
- ∉ localUrl
    - Local address of the Miniserver
    - Only when not a Trust Miniserver
- ∉ remoteUrl
    - Remote address of the Miniserver
    - Only when not a Trust Miniserver

V17.0

PoolController

Covered Config Items

- ∉ Pool Controller

Info

The PoolControl only works with an Aquastar Air Valve.

Details

- valveType
  - The types of the valve
    - 0 = No Valve
    - 1 = Aquastar
  - hasEco
    - If something is connected to the ECO input
  - hasTargetTemp
    - Is input T connected
  - hasActualTemp
    - Is input AI connected
  - hasWaterLevel
    - Is input WI connected
  - waterLevelUnit
    - The unit for the waterlevel
  - hasCoverPosition
    - Is input CP connected
  - hasCover
    - Is output Qco and Qcc connected
  - swimmingMachineType
    - Is dependent on output AQsm
      - 0 = no swimming machine
      - 1 = digital value
      - 2 = analog value
  - hasCustom1
    - Is input AI1 connected
  - hasCustom2
    - Is input AI2 connected
  - customName1
    - Name for value of input AI1
  - customName2
    - Name for value of input AI2
  - customUnit1

        - Unit for value of AI1
    - customUnit2
        - Unit for value of AI2
    - filterBounds
        - Min and max values for mode “Filter” in seconds
    - backwashBounds
        - Min and Max values for mode “backwash” in seconds
    - rinseBounds
        - Min and Max values for mode “rinse” in seconds
    - circularBounds
        - Min and Max values for mode “circulate” in seconds
    - drainBounds
        - Min and Max values for mode “drain” in seconds
    - hasHeating
        - Is output Qh connected
    - hasCooling
        - Is output Qc connected

States

- currentOpMode
    - current operating mode
        - 0 = Out of order
            - while the reset input is active
        - 1 = Automatic
        - 2 = Servicemode
- currentTempMode
    - current temperature mode
        - 0 = off
        - 1 = Automatic
        - 2 = Automatic heating
        - 3 = Automatic cooling
        - 4 = manual heating
        - 5 = manual cooling
- tempActual
    - The actual water temperature
- tempModeCycleActive
    - If the temperature regulation cycle is active
- tempTarget
    - The target temperature
- waterLevel
    - The actual water level
- custom1
    - The value of AI1

- custom2
  - The value of AI2
- heatingApproved
  - If heating is approved, only used if hasHeating is true
- coolingApproved
  - If cooling is approved, only used if hasCooling is true
- ecoActive
  - If the eco mode is active
- swimmingMachine
  - This value can either be digital or analog, please refer to "swimmingMachineType" in the control details
- coverPosition
  - Analog value of the cover position
    - 0.0 = open
    - 1.0 = closed
- coverOpening
  - If the cover is opening right now
- coverClosing
  - If the cover is closing right now
- currentCycle
  - Current active cycle
    - 0 = No cycle is active
    - 1 = Filter
    - 2 = Flushing
    - 3 = Circulate
    - 4 = Drain
- remainingTime
  - Remaining time of the active cycle in seconds
- valvePosition
  - The current position of the valve
    - -1 = Valve moves
    - 0 = Filter
    - 1 = Backwash
    - 2 = Clearwash
    - 3 = Circulate
    - 4 = Closed
    - 5 = Drain
      - 2-way ball valve is activated
      - Valve moves to position drain
      - Activates pump for a given time
      - After given time valve will be deactivated, but the 2-way ball valve and the valve stays at the drain position
    - 6 = Relieve
      - This is used to relieve the valve in the winter

- pump
  - If the pump is active
- drainValve
  - If the drainvalve is opened. There might be a separate drainValve attached that needs to be opened besides setting the valvePosition to drain.
- delayTime
  - The time of the delay
    - This must be in the range of “delayBounds”
- filterTime
  - The time in seconds the mode “Filter” will be active
    - This must be in the range of “filterBounds”
- backwashTime
  - The time in seconds the mode “BackwashTime” will be active
    - This must be in the range of “backwashBounds”
- rinseTime
  - The time in seconds the mode “Rinse” will be active
    - This must be in the range of “rinseBounds”
- circulateTime
  - The time in seconds the mode “Circulate” will be active
    - This must be in the range of “circulateBounds”
- drainTime
  - The time in seconds the mode “Drain” will be active
- error
  - Error status of the PoolController (available since Miniserver 8.0)
    - 0 = No error is or was present
    - 1 = An error was present
    - 2 = An error is currently present
    - 3 = Device is offline
- cycleAbortable
  - State that shows if the current cycle can be aborted
  - available since 10.2

Commands

- coverClose
  - Closes the cover if one is connected
- coverOpen
  - Opens the cover if one is connected
- operatingMode/{opMode}
  - Activates the given operating mode
    - 0 = Out of order
    - 1 = Automatic
    - 2 = Servicemode
- tempMode/{tempMode}

    - Activates the given temperature mode
        - 0 = off
        - 1 = Automatic
        - 2 = Automatic heating
        - 3 = Automatic cooling
        - 4 = manual heating
        - 5 = manual cooling
- eco/{state}
    - state
        - 0 = off
        - 1 = on
- targetTemp/{temperatur}
    - Sets the given target temperature
- swimmingMachine/{value}
    - value can be either analog (0.0 - 1.0) or digital. Please refer to "swimmingMachineType"
- startCycle/{cycleID}/{seconds1}/{seconds2}
    - starts the given cycle with the given seconds
        - cycleID"
            - 1 = Filter
            - 2 = Flushing
            - 3 = Circulate
            - 4 = Drain
        - seconds1 = The duration the cycle will be active
            - If no seconds are given the values from the states (filterTime, circleTime,..) will be used
- filter
    - Short for "startCycle/1"
- backwash
    - short for "startCycle/2"
- circulate
    - short for "startCycle/3"
- drain
    - short for "startCycle/4"
- valvePos/{position}
    - Sets the valve position
        - position
            - 0 = Filter
            - 1 = Backwash
            - 2 = Clearwash
            - 3 = Circulate
            - 4 = Closed
            - 5 = Drain
        - Position 6 = Relieve-position cannot be set using a command

- pump/{state}
    - Activates or deactivates the pump
    - state
        - 0 = off
        - 1 = on
- drainValve/{state}
    - Opens or closes the drainvalve
    - state
        - 0 = open
        - 1 = close
- reset
    - Pulse for reset
    - reset/1 will activate “Out of order”
- disable/{state}
    - Disables or enables childlock
    - state
        - 0 = off
        - 1 = on
- delayTime/{time}
    - Sets the delaytime in seconds
    - NOTE: This value must be between “delayBounds”
- filterTime/{time}
    - Sets the filtertime in seconds
    - NOTE: This value must be between “filterBounds”
- backwashTime/{time}
    - Sets the backwashtime in seconds
    - NOTE: This value must be between “backwashBounds”
- rinseTime/{time}
    - Sets the rinseTime in seconds
    - NOTE: This value must be between “rinseBounds”
- circulateTime/{time}
    - Sets the circulateTime in seconds
    - NOTE: This value must be between “circulateBounds”
- drainTime/{time}
    - Sets the drainTime in seconds
    - NOTE: This value must be between “drainTime”
- approveHeating/{value}
    - Approves or disapproves heating
    - value
        - 1 = approve
        - 0 = disapprove
- approveCooling/{value}
    - Approves or disapproves heating
    - value

        - 1 = approve
        - 0 = disapprove
- skipDelay
    - This cancels the delay
- ackError
    - Acknowledges the current error (available since Miniserver 8.0)

The poolcontroller uses a daytimer as subcontrol for defining the schedule.

Pushbutton

Covered Config Items

- ∉ Virtual Input (Pushbutton)
- ∉ Push Button
- ∉ Scene

Details

- type
    - 511 - scene
    - 71 - virtual input visualized as push button
    - Missing = Push button block

States

- active
    - the current state of the pushbutton

Commands

- pulse
    - if the button was only tapped for a very short time, use pulse instead of on and off
- on
    - when the button is hit but not released immediately
- off
    - to deactivate the button after an on-command

PVProductionForecast

Covered Config Items

- PV Production Forecast

Details

- periodByLogic
  - True if period parameter is set by logic
- hasPeriodAutomation
  - True if something is connected to period output
- hasTodayAutomation
  - True if something is connected to today output
- hasTomorrowAutomation
  - True if something is connected to tomorrow output
- hasReadyAutomation
  - True if something is connected to ready output
- maxPower
  - PV's theoretical maximum power setting [kWp]

States

- inputperiod
  - Value of period parameter [h]
- period
  - Predicted production for given period of time [kWh]
- today
  - Predicted production for today [kWh]
- tomorrow
  - Predicted production for tomorrow [kWh]
- after
  - Predicted production for the day after tomorrow [kWh]
- ready
  - False if block locked by off input, or on errors. E.g. fetching data, Weather Subscription expired, etc.
- timestamp
  - Timestamp of predictions [Seconds since 2009]
- cycleFrom
  - Timestamp of the start of the current prediction hour [Seconds since 2009]
- cycleUntil
  - Timestamp of the end of the current prediction hour [Seconds since 2009]
- errorInfo
  - Bitmask of eventual errors: xy where x=WeatherServiceError and y=InternetError

Commands

- getForecasts
  - Current predictions as JSON array (24 per day, for each hour):
    {

"today":
[
{"t":1756072800,"v":0},
{"t":1756076400,"v":0},
... ,
{"t":1756155600,"v":0}
],
"tomorrow":
[
... ,
],
"after":
[
... ,
]
}

- t – Timestamp [Unix]
- v – Value [kWh]

Radio

Covered Config Items

- ∉ Radio buttons (8x)
- ∉ Radio buttons (16x)

Details

- allOff
  - the name shown when no output is selected
  - Empty string if selecting neither one of the outputs is not an option
- ∉ outputs
  - set of output names with their ID as key. (1 - 16)
  - there might be missing IDs (e.g.: 1,2,5,8 is a valid set of IDs)

States

- activeOutput
  - ID of the currently active output.
  - 0 means no output is selected, show “All Off” (if available)

Commands

- reset
  - Deselects the currently selected ID, changes activeOutput to 0
- *ID*

    - Simply sending an ID will activate the corresponding output & change activeOutput
    - Send '1' for the first output
    - 0 cannot be selected directly, only via 'reset'
- next
    - Select the next output. Respects the 'Sk0' parameter
    - Since 13.3.1.10
- prev
    - Select the previous output. Respects the 'Sk0' parameter
    - Since 13.3.1.10

PresenceDetector

Details

- ∉ maxEntries
    - Maximum number of Tracker-Entries
- ∉ parameterConnections
    - bitmask of parameters controlled by logic.
    - the index is in the order of the parameters of the control
    - Value 1 (Bit0): parameter "time" is locked

States

- ∉ activeSince
    - timestamp in seconds since 2009
- ∉ active
    - presence state
- ∉ locked
    - locked state
- ∉ infoText
    - reason why the presence detector is locked
- ∉ time
    - current overrun time (TH)

Commands

- {value}
    - value for the virtual input
    - between min and max
- time/{value]
    - set overrun time
- presence/{value]
    - set presence on for the given duration

- deactivate/{value]
    - turns presence off for the given duration

Subcontrols

- trackerSubcontrol

PulseAt

Covered Config Items

- ∉ Pulse At

Details

- pulseDurationLocked
    - true wenn Pulse time input is set by logic

States

- isActive
    - If the Output on Q is 1
- startTime
    - seconds since midnight
- oneTimePulseDate
    - If a fix date for a one time pulse is set secondsSince2009
- pulseDuration
    - Pulse duration in seconds
- type
    - Current used type

Commands

- setTime/{time}
- setOneTimePulse/{date}
    - Value=0 sets the pulse recurring every day
- setPulseDuration/{duration}
- pulse
- setType/{type-id}
    - Set the pulse type to one of the types available in the control details

Remote

Covered Config Items

- ∉ Media controller

Details

- favoritePad
  - 0 = D-Pad should be initially visible
  - 1 = Number-Pad should be initially visible
- modeList
  - Object with the modes (usedButtons are currently not used!)
    - { "1": { "name": "Mode One", "usedButtons": []}}

States

- ∉ timeout
  - The timeout in milliseconds
- ∉ mode
  - The key for the current mode (The key is for the modeList)
  - 0 means "no mode selected" = all off.
- ∉ active
  - Will be true if the Miniserver is sending the commands for switching the modes or power on
  - since Config 8.0

Commands

- mode/{modeID}
  - This enables the mode with the given ID
  - cannot select mode "0" - use reset for that.
- on
  - This enables the AQp (Power) output
- off
  - This disables the AQp (Power) output
- mute
  - Represents the Mute Button
- play
  - Represents the Play Button
- pause
  - Represents the Pause Button
- stop
  - Represents the Stop Button
- rewind
  - Represents the Rewind Button
- previous
  - Represents the Previous Button
- next

    - Represents the Next Button
- forward
    - Represents the Forward Button
- menu
    - Represents the Menu Button
- info
    - Represents the Info Button
- exit
    - Represents the Exit Button
- guide
    - Represents the Guide Button
- volplus
    - Represents the press of the Volume Plus Button
- volminus
    - Represents the press of the Volume Minus Button
- volplusoff
    - Represents the release of the Volume Plus Button
- volminusoff
    - Represents the release of the Volume Minus Button
- prgplus
    - Represents the press of the Program Plus Button
- prgminus
    - Represents the press of the Program Minus Button
- prgplusoff
    - Represents the release of the Program Plus Button
- prgminusoff
    - Represents the release of the Program Minus Button
- return
    - Represents the Return Button
- btnred
    - Represents the Red Button
- btnblue
    - Represents the Blue Button
- btnyellow
    - Represents the Yellow Button
- btngreen
    - Represents the Green Button
- dirok
    - Represents the D-Pad OK Button
- dirup
    - Represents the press of the D-Pad Up Button
- dirupoff
    - Represents the release of the D-Pad Up Button
- dirdown

    - Represents the press of the D-Pad Down Button
- dirdownoff
    - Represents the release of the D-Pad Down Button
- dirleft
    - Represents the press of the D-Pad Left Button
- dirleftoff
    - Represents the release of the D-Pad Left Button
- dirright
    - Represents the press of the D-Pad Right Button
- dirrightoff
    - Represents the release of the D-Pad Right Button
- num{x}
    - This sends the number “x”, “x” goes from 0-9
        - Example: “num1” for number 1
- number/{x}
    - This sends the number “x”, “x” can be any positive number
        - Example: “number/18” for number 18
- reset
    - turns off all devices of the current mode & changes to mode 0 (= no mode active)
    - available since Miniserver 8.0

Sauna

Covered Config Items

- ∉ Sauna controller
- ∉ Sauna controller with evaporator

Details

- hasVaporizer
    - determines whether or not it is a full featured sauna with vaporizer
- hasDoorSensor
    - whether or not the value of the door sensor can be visualized (might not be attached)

States

- active
    - whether or not the sauna is active (!= power!)
- power
    - is it currently heating up
- tempActual

    - the actual temperature inside the sauna
- tempBench
    - the actual temperature provided by the bench sensor
- tempTarget
    - the current target temperature
- fan
    - is the fan on
    - indicates the airing phase (if drying is on too)
    - the airing phase will stop after the airing-time configured on the block is reached.
- drying
    - indicates that the “drying phase” is on
    - if the fan is on too, it’s called the “airing phase”
    - the drying will stop once the targetTemp is reached
- doorClosed
    - active if door is closed, only to be used if hasDoorSensor is true
- presence
    - forwards the state of the presence input of the block
- error
    - digital indicator for a sauna error
- saunaError
    - indicates what error has occurred and why the sauna has shut down.
    - 0 = no error
    - 1 = too hot
- timer
    - seconds left of the sauna timer
- timerTotal
    - total number of seconds of the sauna timer
- lessWater (evaporator only)
    - becomes active if the evaporator runs out of “water”
- humidityActual (evaporator only)
    - actual humidity inside the sauna
- humidityTarget (evaporator only)
    - target humidity inside the sauna
- mode (evaporator only)
    - when an evaporator is present, different sauna modes can be used, these are identified by a modeNr
    - 0 = Off
    - 1 = Finnish manual operation
    - 2 = Humidity manual operation
    - 3 = Finnish automatic operation (80°C)
    - 4 = Herbal automatic sauna (45°C, 50%)
    - 5 = Soft steam bath automatic (50°C, 50%)
    - 6 = Warm air bath automatic (45°C, 45%)

Commands

- on
  - turns the sauna on
- off
  - turns the sauna off right away (no airing/drying phase)
- fanoff
  - turns the fan off
- fanon
  - turns the fan on, only works if sauna is active
- temp/{target}
  - set the target temperature (for manual mode)
- humidity/{target}
  - set the target humidity (hasVaporizer only)
- mode/{modeNr}
  - see mode state description for details
- pulse
  - changes between the sauna activity-states
    - off -> on
    - on -> drying
    - drying -> airing
    - airing -> off
- starttimer
  - starts the sand timer, will count down from timerTotal
- ontemp/{target}
  - turns the sauna on and sets target temperature

Sequential

Covered Config Items

- Sequential Controller

Details

- sequences
  - JSON array containing sequence objects.
  - [{ id: 4, name: “Gustav” }, {id: 8, name: “Karl}”]

States

- activeSequence
  - id of the currently active sequence
  - 0 = no sequence active.

Commands

- triggerSequence/{sequenceId}
  - activates the sequence with the provided id.
  - id 0 means that any currently active sequence will be stopped.

Slider

Covered Config Items

- ∉ Virtual Input (Slider)

Details

- ∉ format
  - the format of the value
- ∉ min
  - the minimum value
- ∉ max
  - the maximum value
- ∉ step
  - the step to the next value of the slider when pressing "-" or "+"

States

- value
  - the current value of the slider
- error
  - indicates an invalid value of the slider

Commands

- {number}
  - value for the slider
  - between min and max

SmokeAlarm

Covered Config Items

- Fire/water alarm

Details

- hasAcousticAlarm
  - returns true if the smoke alarm control has an acoustic alarm configured

- availableAlarms
    - Bitmask showing what is being monitored (may be a combination)
        - 0x01: Smoke
        - 0x02: Water
        - 0x04: Temperature
        - 0x08: Arc Fault (electrical wiring), available since Miniserver 9.3

States

- nextLevel
    - the ID of the next alarm level
        - 1 = Silent
        - 2 = Acustic
        - 3 = Optical
        - 4 = Internal
        - 5 = External
        - 6 = Remote
- nextLevelDelay
    - The delay of the next level in seconds, this can be specified with the parameters D1 - D6 in Loxone Config. This increments every second...
- nextLevelDelayTotal
    - The total delay of the next level in seconds, this can be specified with the parameters D1 - D6 in Loxone Config.
- level
    - The ID of the current alarm level
        - 1 = Pre Alarm
        - 2 = Main Alarm
- sensors (**DEPRECATED**, handled by subcontrol)
    - A string of sensors separated by a pipe ("|")
- acousticAlarm
    - The state of the acoustic alarm 0 for not active and 1 for active
    - This only can be 1 if something is connected to the "Qh" Output in Loxone Config and the main alarm is active
- testAlarm
    - 0 if testalarm is not active and 1 if it is active
- alarmCause
    - Bitmask for Alarm-Causes (may be a combination)
        - 0x01: Smoke
        - 0x02: Water
        - 0x04: Temperature
        - 0x08: Arc Fault (electrical wiring)
- startTime
    - timestamp when alarm started
- timeServiceMode

    - Remaining seconds how long the service mode remains active
    - 99999 - infinite
- areAlarmSignalsOff
    - State if all alarm signals are disabled (by confirming alarm)
        - available since 10.3

Commands

- mute
    - mutes the sirene
- confirm
    - Acknowledge the alarm
- servicemode/{number}
    - 0 = Off
    - 1 = duration is infinite (timeServiceMode state will be set to 99999)
    - >1 Time in seconds until the service mode stops
- startDrill
    - Start a test alarm (available since 10.3)

SolarPumpController

Covered Config Items

∉ Thermal Solar Controller

Details

- buffers
    - List of used buffers
    - eg. [
        - {
            - “name”: “Buffer 1”
        - },
        - {
            - “name”: “Buffer2”
        - }
    -]

States

- bufferTemp{n}
    - n: number from 0 to 4
    - Temperature of buffer n
- bufferState{n}
    - n: number from 0 to 4
    - State of buffer n

    - Possible Values
        - 0 = Waiting, Buffer is waiting to be heated
        - 1 = Heating, Buffer is heating
        - 2 = Cooling, Buffer is cooling
        - 3 = OK, Buffer is heated to its temperature
- logicOverride
    - If the control is overwritten by logic
- collectorTemp
    - Temperature of the collector
- heatOverload
    - If heat overload is reached

SpotPriceOptimizer

Covered Config Items

- ∉ Spot Price Optimizer

Details

- format
    - Unit format e.g. “%.3f€/kWh”
- demandByLogic
    - if true demand parameter is set by logic
- periodByLogic
    - if true period parameter is set by logic
- minRuntimeByLogic
    - If true minimum runtime is set by logic
- hasAutomation
    - If something is connected to output O

States

- active
    - If the output is currently active
- current
    - Current Price/Value
- demand
    - Value of Demand Parameter
- period
    - Value of period Parameter
- minRuntime
    - Value of minimum runtime parameter, which defines the minimum continuous duration for which the output must remain active once turned on.
- manualMax
    - Is the price above this value it's always very high

- forecastsTimestamp
  - Timestamp of the forecasts - when it changes request the forecasts again with ‘getforecasts’
- cycleFrom
  - Seconds since 2009 UTC
  - Timestamp from what time the current cycle started (rounded to hours)
- cycleUntil
  - Seconds since 2009 UTC
  - Timestamp until what time the current cycle is running (rounded to hours)

Commands

- cancel
  - Cancel active cycle
- start/[demand in hours]/[period in hours]
  - Start a new cycle with the given parameters
  - When demand is set by logic the parameter can be set to 0
- getForecasts
  - Current available forecast objects as json array


```
{
"u": "1a173fef-02bf-223c-ffff504f94a00067",
"v": 24.795,
"s": 1668510000,
"e": 1668513600,
"planned": true
},
```


  - u - unique id
  - v - value
  - s - Start Unix UTC Timestamp
  - e - End Unix UTC Timestamp
  - planned
    - optional
    - true when the block has planned to activate the output at this timestamp

- getCheapestForecasts/[demand in hours]/[period in hours] (Loxone Config 16.1.11.5)
  - Finds the cheapest forecast objects for given cycle and demand (in hours) as json array. Note: if we are in an active cycle, cheapest forecast objects are taken from the planned function block forecasts


```
{
"u": "1a173fef-02bf-223c-ffff504f94a00067",
```

"v": 24.795,
"s": 1668510000,
"e": 1668513600
},

- u - unique id
- v - value
- s - Start Unix UTC Timestamp
- e - End Unix UTC Timestamp

- getNextDayCheckTime
  - Returns the saved time in the miniserver (seconds since midnight) when data for the selected market is available (Loxone Config 16.3.2.11)

StatusMonitor

Covered Config Items

Status Monitor

Details

- ∉ inputs
  - ⊄ An array of objects containing information of the inputs monitored by this block.
  - ⊄ The index of the object inside this array will correspond to the position of the state for this object in the *inputStates* array.
  - ⊄ An object contains a name for the input, the uuid of the room it is in and the installation place
    - uuid the uuid of the control which may be visualized to navigate there
      - Fallback when control cannot be serialized
    - name the name of input
    - room the uuid of the room
    - installPlace the installation place of the input
      - New since V12.0, previously this was part of the name.

- ∉ status
  - ⊄ An object containing information about states. This object includes sub-objects, each representing a specific state. Subobject naming index starts 0.
  - ⊄ States without an status value will not appear in this array.

- ⊄ A state has a name, uuid of the state itself, output index, priority and color for the text
  - name text of the status
  - prio priority of the state (starting at 0)
  - id output index of the state (starting at 0)
  - color color of the status text
  - uuid the uuid of the state itself

States

- numState0 – numState9, numDef
  - ⊄ The number of inputs wiht the corresponding state.
  - ⊄ The sum of the values from all these states is equal to the number of inputs monitored including integrated status monitors
  - ⊄ Inputs that do not have a matching state value will be counted towards numCdef
  - ⊄ The identifier "id" from Details->states corresponds with these states.
    - e.g. id = 0 → numState0
    - e.g. id = 10 → numDef
- inputStates
  - ⊄ Will return a string containing the states of each of the inputs by this block including integrated status monitors.
  - ⊄ The individual states are separated by a comma. The position of the state in this string corresponds to the position of the objects inside the *inputs* array under *details.*
  - ⊄ Each state is a integer value that is used to identify the current status the input belongs to (starting at 0)
    - e.g. 1 → status with 'id' = 1
  - ⊄ Integrated status monitors have the value of the highest priority status.
  - ⊄ Integrated unconfigured status monitors have value 255.

SteakThermo

Covered Config Items

- ∉ Touch & Grill
  - Sensors are counted from left (yellow) to right (green)

Details

- deviceType
  - 0: No device connected
  - 1: Touch & Grill Air with two sensors

- isTouchProtectConnected
  - 0: No logic connected to the input DisT
  - 1: Logic connected to the input DisT
- isBrightnessConnected
  - 0: No logic connected to the parameter B
  - 1: Logic connected to the parameter B

States

- currentTemperatures
  - TextEvent can be interpreted as JSON
  - [
    - 82,
    - 148
  -]
  - Temperatures of the sensors, from left to right
- temperatureYellow
  - Temperature of the yellow sensor
- temperatureGreen
  - Temperature of the green sensor
- sensorInfo
  - TextEvent text can be interpreted as JSON
  - [
    - {
      - “name”: ”Brisket”,
      - “connected”: true,
      - “target”: 85
    - },
    - {
      - “name”: ”Meat”,
      - “connected”: true,
      - “target”: 100
    - }
  -]
  - name:
    - Name of the sensor, can be set by the user
  - connected
    - If the sensor is connected
  - target
    - defined target temperature of the sensor
  - Index in the Array
    - 0 = Left (yellow)
    - 1 = Right (green)
- targetYellow

    - Target temperature of the yellow sensor
- targetGreen
    - Target temperature of the green sensor
- sensorAlarms
    - TextEvent can be interpreted as JSON
    - [
        - {
            - “text”: “Target temperature reached”,
            - “time”: {secondsSince2009},
            - “ringing”: true
        - },
        - {
            - “text”: “Target temperature reached”,
            - “time”: {secondsSince2009},
            - “ringing”: false
        - }
    -]
    - text:
        - Description of the sensor alarm
    - time:
        - When the alarm has been triggered in seconds since 2009
    - ringing:
        - If the device is beeping or not
- yellowAlarmActive
    - If the yellow sensor does have an active alarm
- greenAlarmActive
    - If the green sensor does have an active alarm
- activeAlarmText
    - TextEvent the text property is the text of the currently active Alarm
    - Text is set if Qa is active and logic is connected Qa
    - If no alarm is active this value will be an empty string
- timerRemaining
    - Remaining time in seconds of the timer
- timerInfo
    - TextEvent, can be interpreted as JSON
    - {
        - “active”: true,
        - “duration”: 300
    - }
    - active:
        - If the timer is running
    - duration:
        - Time in Seconds
- timerAlarm

    - TextEvent, can be interpreted as JSON
    - {
        - “text”: “Timer started”,
        - “time”: {secondsSince2009},
        - “ringing”: true
    - }
    - text:
        - Description of the timer alarm
    - time:
        - When the alarm has been triggered in seconds since 2009
    - ringing:
        - If the device is beeping or not
- timerAlarmActive
    - If a timer alarm is active
- deviceState
    - The current state of the device
    - 0 = Running
    - 1 = Offline
    - 2 = Turned off
    - 3 = Error
    - 4 = Device Asleep
- displayAlwaysOnBat
    - If the display should stay on while the device is running on battery
- displayAlwaysOnDc
    - If the display should stay on while the device is connected to power
- availableControls
    - TextEvent, can be interpreted as JSON
    - [
        - {
            - “name”: “Kitchen”,
            - “uuid”: “0f38633e-0073-1a75-ffffed93b67b4c69”
        - },
        - {
            - “name”: “Garden”,
            - “uuid”: “0b734138-032f-0284-ffff403fb0c34b9e”
        - }
    -]
    - Defines the available controls the device has been assigned to
- activeControl
    - Index of the availableControls state
- isActive
    - If this particular control is the active control
- touchProtection
    - If the touch protection of the device is active

- displayBrightness
    - Brightness of the display, available since 10.3
    - Values from 0 - 100 (%)
- powerMode
    - The powerMode of the device, available since 10.2.1.16
    - 1 = Device is DC powered
    - 2 = Device is battery powered
- batteryStateOfCharge
    - The battery state of charge in percentage, available since 10.2.1.16
    - Values from 0 - 100 (%)

Commands

- quitAlarm
    - Quits all ongoing alarms
- setSensor/{sensorIndex}/{targetTemp}/{sensorName}
    - sensorIndex
        - The index of the sensor
            - 0 = Left (yellow)
            - 1 = Right (green)
    - targetTemp
        - The target temperature of the sensor
    - sensorName
        - The name for the sensor
- setDisplayAlwaysOnBat/{on}
    - on
        - 0 = Disabled
        - 1 = Enabled
- setDisplayAlwaysOnDc/{on}
    - on
        - 0 = Disabled
        - 1 = Enabled
- setActive/{index}
    - Sets the given control as active
    - index
        - The index of the control in the availableControls state
- setThisActive
    - Sets this control as the active control
- setTimerDuration/{duration}
    - Sets the timer duration
    - duration
        - The duration in seconds
- startTimer

    - Starts the timer with the defined duration
- stopTimer
    - Stops the timer
- setTouchProtection/{on}
    - Enables or disables the touch protection
    - on
        - 0 = Disabled
        - 1 = Enabled
- setDisplayBrightness/{brightness}
    - Sets the brightness of the display, available since 10.3
    - brightness
        - Values from 0 to 100 (%)

Switch

Covered Config Items

- ∉ Virtual Input (Switch)
- ∉ Push button

States

- active
    - the current state of the switch
- lockedOn
    - active when switch cant be deactivated because of a constant on by logic

Commands

- on
    - activates the switch
- off
    - deactivates the switch

SystemScheme

Available since 11.0

Covered Config Items

- ∉ SystemScheme

Details

- imagePath

V17.0

    - Path to the image to be shown in this block (Path → uuid from the control)
- imageVersion
    - Version that can be used to detect image change
- mainControl
    - Uuid of control that should be displayed on overview
- schemeSize
    - how much space will the scheme require (the image might be larger/smaller and needs to be scaled)
    - Object, containing width and height attribute
- controlReferences
    - which controls are being referenced and how/where to show them
    - each object within the array contains the following attributes:
        - uuidAction
            - the uuid of the referenced control. The control might not be available within the structure file due to permission restrictions.
        - pos
            - position which the displayed control needs to be centered to
            - Object, containing the x and y position.

TextState

Covered Config Items

- ∉ Status

States

- textAndIcon
    - TextEvent with text and icon
- iconAndColor
    - Text state with json containing icon and icon color
    - { "icon":"Icons/xxxx-xxx-xx.svg", "color": “#45864A” }
    - Since 13.1

TextInput

Covered Config Items

- ∉ Virtual text input

States

- text
    - TextEvent with text

Commands

- {text}
  - new value of text input

TimedSwitch

Covered Config Items

- ∉ Stairwell light switch
- ∉ Comfort/Multifunction switch

Details

- isStairwayLs
  - true = “Stairwell light switch”
  - false = “Comfort/Multifunction switch”

States

- ∉ deactivationDelayTotal
  - seconds, how long the output will be active if the timer is used.
- ∉ deactivationDelay
  - countdown until the output is deactivated.
    - 0 = the output is turned off
    - -1 = the output is permanently on
    - otherwise it will count down from deactivationDelayTotal

Commands

- on
  - Permanently activates the TimedSwitch, deactivationDelay will change to -1
- off
  - Turns off the TimedSwitch, deactivationDelay will change to 0
- pulse
  - deactivationDelay = 0
    - Will start the countdown, from deactivationDelayTotal to 0
  - isStairwayLs = true
    - deactivationDelay = -1
      - No effect, will remain permanently on.
    - deactivationDelay > 0
      - Restarts the countdown
  - isStairwayLs = false
    - turns it off. (from countdown or permanent on state)

Tracker

Covered Config Items

- ∉ Tracker

Details

- maxEntries
  - maximal count of entries returned by the miniserver

States

- entries
  - entries returned from the miniserver as string
  - entries are separated by a pipe symbol "|"
  - Hex character "\x14" indicates a new line

UpDownLeftRight digital

Covered Config Items

- ∉ Virtual Input (Left-right buttons)
- ∉ Virtual Input (up-down buttons)

Commands

- UpOn
  - activates the up/right output
- UpOff
  - deactivates the up/right output
- PulseUp
  - since Config 8.0
  - impuls on up/right output
- DownOn
  - activates the down/left output
- DownOff
  - deactivates the down/left output
- PulseDown
  - since Config 8.0
  - impuls on down/left output

UpDownLeftRight analog

Covered Config Items

- ∉ Virtual Input (Left-Right buttons)
- ∉ Virtual Input (Up-Down buttons)

Details

- ∉ format
  - the format of the value
- ∉ min
  - the minimum value
- ∉ max
  - the maximum value
- ∉ step
  - the step to the next value of the virtual input when pressing up/down/left/right

States

- value
  - the current value of the virtual input
- error
  - indicates an invalid value of the virtual input

Commands

- {value}
  - value for the virtual input
  - between min and max

ValueSelector

Covered Config Items

- ∉ Push-button +/-
- ∉ Push-button +

Details

- ∉ increaseOnly
  - indicates if the button has only an “+”-input
- ∉ format
  - the format of the value

States

- ∉ min
  - the minimum value
- ∉ max
  - the maximum value
- ∉ step
  - the step to the next value of the virtual input when pressing up/down/left/right
- ∉ value
  - the current value of the virtual input

Commands

- {value}
  - value for the virtual input
  - between min and max

Ventilation

Covered Config Items

- ∉ Ventilation

Details

- ∉ hasPresence
  - indicates that the “Mv” input is connected to logic
- ∉ hasAbsenceMin
  - indicates that the parameter “V” is connected to logic
- ∉ hasAbsenceMax
  - indicates that the parameter “Vi” is connected to logic
- ∉ hasPresenceMin
  - indicates that the parameter “VP” is connected to logic
- ∉ hasPresenceMax
  - indicates that the parameter “VPi is connected to logic
- ∉ hasHumidityMax
  - indicates that the parameter “Hmax” is connected to logic
- ∉ hasIndorHumidity
  - indicates that the “Hi” input is connected to logic
- ∉ modes
  - All available modes
  - [
    - {
      - name: “Heat exchanger”

- id: 2
- },
- {
- name: “Exhaust”,
- id: 1
- },
- ...

○]

∉ timerProfiles

○ Available timer profiles

○ [

- {
- name: “Resting”,
- useCase: “”,
- interval: 3600,
- speed:
  - {
    - value: 100,
    - enabled: true
  - }
- modes:
  - [
    - 0,
    - 1,
    - 4,
  -],
- defaultMode: 1
- },
- ...

○]

○ [useCase]

- Optional description of the timer profile, string will be empty if no useCase is given

○ interval

- Time in seconds this timer should be running

○ speed

- Object with following properties
  - value
    - Speed in % of the ventilator
  - enabled
    - If the current timer profile is allowed to change the speed

○ modes

- Array of available mode ids for this particular timer profile

        - These mode ids refer to the ids of the mode object in the modes detail
    - defaultMode
        - defines the per default selected mode for the timerProfile
    - In addition manual timers are available, the manual timer must be handled by the app itself, these are not provided by the Miniserver
- type
    - Defines the type of the control. This will be used to differentiate between different blocks to be able to use different UI or functions
        - 0: Generic
        - 1: Leaf

States

- ventReason
    - The ID for ventilation reason
        - 0: Basic ventilation
        - 1: Increased ventilation
        - 2: Reserved for future use
        - 3: Reserved for future use
        - 4: Stop
            - Block input “St”
        - 5: Window opened
            - Block input “Iw”
        - 6: Turbo
            - Block input “Tb”
        - 7: Manual
            - Overwritten via timer
        - 8: Exhaust
            - Block input “Ex”
        - 9: Fall-asleep-mode
            - Block input “Sl”
        - 10: Frostprotection
            - Intake Temperature < TF
            - available since 10.2
- temperatureSupport
    - Whether or not temperature support is active
    - -1: Cooling
    - 0: No Temperature support is active
    - 1: Heating
- activeTimerProfile
    - Index of the current active timer profile
    - -1: Manual
    - -2: No timer active

- -3: Someone is changing settings
- ∉ stoppedBy
  - The name of the connected logic if the stop (“St”) input is active
- ∉ overwriteUntil
  - Unixtimestamp
  - This state is 0 if no timer is active
- ∉ controlInfo
  - TextEvent can be interpreted as JSON
  - {
    - level: 1,
    - title: “Error”,
    - desc: “Ventilation left offline”,
    - link: “https://www.loxone.com”,
    - action:
      - {
        - name: “Acknowledge”,
        - cmd: “cancelAlarm”
      - }
  - }
  - level
    - Defines the state of the control
    - 0: Ok
      - Won’t be displayed in the visualisation
    - 1: Error
      - Red colored
    - 2: Warning
      - Orange colored
    - 3: Info
      - Gray colored
  - [title]
    - Optional: The level will be used instead, if no title is defined
  - desc
    - Describes the current state
  - [link]
    - Optional: link to further information, e.g Online documentation
  - [action]
    - Optional: Defines an action object to resolve the current state
      - name: Name of the action
      - cmd: Command that resolves the current state
- ∉ speed
  - Value in % representing the speed of the ventilation
- ∉ mode
  - Defines the id of the current active mode defined in the details modes object

- ∉ presenceMin
  - Minimal procentual value of the ventilation if someone is present
- ∉ presenceMax
  - Maximal procentual value of the ventilation if someone is present
- ∉ absenceMin
  - Minimal procentual value of the ventilation if no one is present
- ∉ absenceMax
  - Maximal procentual value of the ventilation if no one is present
- ∉ humidityIndoor
  - Value of the indoor humidity sensor
- ∉ presence
  - If presence is active
- ∉ frostTemp
  - Temperature when the frost protection gets active
- ∉ humidityMax
  - The max humidity set (in %)
- ∉ airQualityMax
  - The max air quality set (in ppm)
- ∉ airQualityIndoor
  - The current air quality in ppm)
- ∉ temperatureIndoor
  - The current indoor temperature
- ∉ temperatureOutdoor
  - The current outdoor temperature
- ∉ temperatureTarget
  - The current target temperature

Commands

- setTimer/{interval}/{speed}/{modeId}/{timerProfileIdx}
  - interval
    - Time in seconds
  - speed
    - Ventilation speed
    - 0 - 100
  - modeId
    - Id of the mode defined in the details modes object
  - timerProfileIdx
    - Array index of the timer
    - -1 for the Manual mode
- setTimer/0
  - stops any currently running timer, returns to automatic mode
- setAbsenceMin/{value}
  - Sets the minimal ventilation intensity if no one is present

- setAbsenceMax/{value}
  - Sets the maximal ventilation intensity if no one is present
- setPresenceMin/{value}
  - Sets the minimal ventilation intensity if someone is present
- setPresenceMax/{value}
  - Sets the maximal ventilation intensity if someone is present
- ackFilterChange
  - Acknowledges the “filter change” message

Webpage

Covered Config Items

- ∉ Webpage

Details

- url
  - The defined low resolution URL (This will be the Internal URL if you are connected internally and the external URL if you are connected externally)
- urlHd
  - The defined high resolution URL (This will be the Internal URL if you are connected internally and the external URL if you are connected from externally)
- image
- iconColor
  - Color of the Icon
  - Since 13.1

Window

Covered Config Items

- Window

States

- position
  - window position
  - 0..fully closed
  - 1..fully opened
  - 0 < pos < 1...something in between
- direction
  - -1.. window is getting closed
  - 0.. window is not moving
  - 1.. window is getting opened
- lockedReason

    - Name of the reason why the control is blocked. It is the name of the input connected with “WP”
- targetPosition
    - Window target position (0 = fully closed - 1 = fully opened)
    - available since Miniserver 11.3

- type
    - Window Type for Visualisation
        - Since V15.3
    - 0... Tilt Top
    - 1... Tilt Bottom
    - 2... Tilt Left
    - 3... Tilt Right
    - 4... Sliding Left
    - 5... Sliding Right

Commands

- open/on, open/off
    - start/stop opening window in jog mode
- close/on, close/off
    - start/stop closing window in jog mode
- fullopen
    - open window completely
- fullclose
    - close window completely
- moveToPosition/...
    - starts movement to a defined position
    - 0 = fully closed
    - 100 = fully open
- slightlyOpen/
    - move to partially open position
- stop
    - Stop movement
    - available since Miniserver 11.3

WindowMonitor

Covered Config Items

Window- and Door Monitor

Details

- ∉ windows
    - ⊄ An array of objects containing information on a window or a door monitored by this block.

- ⊄ The index of the object inside this array will correspond to the position of the state for this door or window in the windowStates array.
- ⊄ An object contains a name for the door or window, the uuid of the room it is in and the installation place
  - uuid the uuid of the control which may be visualized to navigate there
    - Fallback when control cannot be serialized
  - name the name of the door/window
  - room the uuid of the room
  - installPlace the installation place of the window/door
    - New since V12.0, previously this was part of the name.

States

- windowStates
  - ⊄ Will return a string containing the states of each of the windows and doors monitored by this block.
  - ⊄ The individual states are separated by a comma. The position of the state in this string corresponds to the position of the object this state is for inside the windows-array in the details.
  - ⊄ Each state is a integer value that represents a bitmask where the individual bits correspond to the following states:
    - none → state unknown / sensor offline
    - 1 → closed
    - 2 → tilted
    - 4 → open
    - 8 → locked
    - 16 → unlocked
- numOpen, numClosed, numTilted, numOffline, numLocked, numUnlocked
  - ⊄ The number of windows & doors in the corresponding states.
  - ⊄ The sum of the values from all these states is equal to the number of windows & doors monitored.
  - ⊄ The windows/doors with two states will always be counted to the “worst” state.
    - e.g.: A lockable door is unlocked and closed. It will be counted to numUnlocked and not to numClosed.

PowerUnit

Covered Config Items

- ∉ Power Supply & Backup

V17.0

Details

- ∉ Outputs
  - ⊄ An array of the configured names for each Output / fuse
  - ⊄ provided name with output number if used.
  - ⊄ Empty strings if unused (=no name specified)
  - ⊄ output numbers starting with 1, not 0!
  - ⊄ e.g. ["East (1)", "", "West (3)", "North (4)", "South (5)", "Outside (6)", ""]
- ∉ supplyMode
  - ⊄ configured supply-mode
  - ⊄ 0: use as power-supply and backup
  - ⊄ 1: use as power-supply only
- ∉ meterRefs
  - ⊄ maps between meter-id and subcontrol
  - ⊄ id 0: total energy meter
  - ⊄ id 1-7: meters for CP1-CP7

States

- outputPower
  - ⊄ Total output power of the Device (kW)
- CP1 - CP7
  - ⊄ Power of Outputs 1 - 7 (kW)
- fuse
  - ⊄ Bitmask for all fallen Fuses
  - ⊄ Bit 1: Fuse for Output 1,...
- supplyTimeRemaining
  - ⊄ Unix-Timestamp: end of supply time in Backupmode
  - ⊄ 0 if unknown / not applicable
- batteryStateOfCharge
  - ⊄ SOC of attached Battery 0 - 100 (%)
- deviceInfo
  - ⊄ Bitmask for Additional Device-States
  - ⊄ Bit 1: Backup-Mode active
  - ⊄ Bit 2: Overcurrent detected
  - ⊄ Bit 3: Battery Missing
  - ⊄ Bit 4: Battery Empty
  - ⊄ Bit 5: Battery-Test active
  - ⊄ Bit 6: Battery-Test finished
  - ⊄ Bit 7: Battery needs service
  - ⊄ Bit 8: Battery defective
  - ⊄ Bit 9: Battery Test OK
  - ⊄ Bit 10: Device is overheating

Subcontrol

- Tracker

Wallbox2

Covered Config Items

- ∉ Wallbox Gen. 2

Details

- Details for Meter
  - see <Meter>
- min
  - Minimum allowed charging Power
- max
  - Maximum allowed charging Power
- modes
  - configurable charging modes
  - JSON-array
  - fields per object:
    - id: connector-id
    - name: configured name, empty: unconfigured
    - value: configured charging limit [kWh]
    - setByLogic: limit is defined by logic
    - price: Price per kWh
      - Since V14.5
- connectedInputs
  - Bitmask of inputs which are used by logic or referenced device
    - Bit 0: Allow charging (Ec)
    - Bit 1: Vehicle Connected (Vc)
    - Bit 2: Power connected
    - Bit 3: Meter Reading connected
    - Bit 4: Charge Active Information
- manager
  - Uuid of Wallbox Energy Manager
  - Since V14.5
- ocpp
  - Optional, object when used with OCPP Plugin, missing if not.
  - “name” – Name of the OCPP Plugin
  - Since 15.1

States

- Details for Meter
  - see <Meter>
  - using actual + total

V17.0

- connected
  - vehicle is connected
- enabled
  - charging is allowed / enabled
- active
  - vehicle is currently charging
- loadshed
  - loadshedding is active
- phaseSwitching
  - charging is paused as phase switching is active, will resume when done
- mode
  - limitation mode <int see Modes>
  - additional manual mode:
    - identified by id 99
    - cannot be renamed, but the limit can be changed
- limit
  - limitation value [kW]
- modeUnplug
  - limitation mode after unplugging vehicle
  - cannot be the manual mode.
  - 0 = keep previous mode
- session:
  - JSON-object for current charging session
  - fields
    - connect: unix timestamp of vehicle connection
    - disconnect: unix timestamp if vehicle disconnection
    - start: unix timestamp of last enabling of charge
    - energy: total energy charged in this session
    - power: current charging power
    - user: userID for current charging session
    - price: Price of the session
      - since V14.5
      - Since 15.1, -1337 = no price identifier
    - extAuthority
      - optional, available since 15.2
      - e.g. Name of OCPP Plugin that approved a charging session
- priority
  - Flag if requested priority (When used in Wallbox Energy Manager)
  - since V14.5
- pricePerHour
  - Price per connected hour
  - since V14.5
  - Since 15.1, -1337 = no price identifier
- pricePerkWh
  - Current price per kWh
  - since V14.5
  - Since 15.1, -1337 = no price identifier
- [limitedBy]

V17.0

    - Limitation type, see limitation flags on Wallbox Manager
    - Only when managed by Wallbox Manager
    - since V14.5

- temporaryLimitation
    - Target charging power is temporary limited due to Regional Regulations, e.g. blocked or ramping up after power loss
    - values:
        - 0: no temporary limitation
        - 1: blocked
        - 2: reduced limitation
        - 3: above limit
    - Since V16.1

Subcontrol

- Tracker - see <Tracker>
    - History of last charging sessions
    - Each line is a JSON-Object of a single session
    - Fields: see <session> state

Commands

- allow/off
    - disable charging
- allow/on
    - enable charging
- setmode/<value>
    - set limitation mode
    - allowed range: 1-5 + 99 (manual)
    - When manual mode is selected (99), it will reuse the previous charging limit.
- modeUnplug/<value>
    - set charging mode when disconnecting vehicle
    - 0 = keep current Mode
    - 1-5: switch to charging mode 1-5
    - Must not be 99 (Manual Mode)
- updateMode/<id>/<limitValue>/<name>/<price>
    - <name> -> the name, but uriEncoded
    - <price> -> optional, since V14.5
- manualLimit/<value>
    - sets the manual charging limit
    - if not already in manual mode, it will activate the manual mode (99)

WallboxManager

Covered Config Items

- Wallbox Manager

Details

- nodes
    - array of nodes to be shown in this manager, see separate section below for more infos

A node provides information about a wallbox or strand that is to be shown in the Wallbox Manager. Nodes can be built up in a tree hierarchy, where one node could contain other nodes. A node can either be a Wallbox itself, or a collection of nodes.

- Properties (properties in [brackets] are optional)
    - uuid → identifies this node in the WBEM, used in getNodeState-Command
    - nodeType → identifies the type of node, the following values exist
        - Wallbox
        - Group
    - title → name to show on the node.
    - icon → the source path of the icon to use for this node
    - [fuse]
        - Fuse power when the node is a group
    - [actualState]
        - name of the state-property of the WBM representing the live value for this node.
        - only available for root-level nodes, child nodes need to use the getNodeState-command with their uuid to request state-values.
    - [limitedByState]
        - name of the satte property of the WBM representing the limitedby value for this node
        - only available for root-level nodes, child nodes need to use the getNodeState-command with their uuid to request state-values.
    - [ctrlUuid]
        - uuid of the control represented by this node. Used to request statistics or the states.
        - If the ctrlUuid is missing, it means that the statistics for this node need to be collected from its child nodes.
    - [nodes]
        - List of child-nodes represented by this node.

States

- priceEco
- pricePrio

- priceHour
- Peco
- Pmax
- Cp
    - Current Power, power actually used by the wallboxes
- Ap
    - Assigned Power, power assigned to the wallboxes to be used.
- actual0 … actualN
    - for each top node of the Manager there is a state of the current assigned power
    - returns the assigned power (not the actually used power)
    - the “actualState” property of the nodes provide info on what state-property delivers the state for that node.
- limitedBy0 … limitedByN
    - for each top node of the Manager there is a state of the current limited by bitmask
    - returns the limitedByMask (see limitedBy below for bits)
    - the “limitedByState” property of the nodes provide info on what state-property delivers the state for that node.

Commands

- getNodeValue/{nodeUuidList}
    - {nodeUuidList}
    - List of nodeUuids separated by a semicolon “;” or comma ‘,’
        - returns a JSON object, where the nodeUuid is the key and the value is a JSON with the output values for the node requested
            - { “node1uuid”: { “fuse”: 1}, “node2uuid”: {“assigned”: 2,”used”:1.5}}
            - {“node1uuid”: {“assignedl”: 0, limitedBy: 1}}
        - [fuse]
            - when node is a strand the maximum fuse power
        - assigned
            - Assigned power
        - used
            - actual used power
            - Combined usedpower when its a strand
        - [limitedBy]
            - When there is a limit on a wallbox
            - Reason why there is a charging limit
            - Bitmask
            - Bits:
                - none = 0x00
                    - No limit
                - eco = 0x01

        - Peco
    - strand = 0x02
        - This strand
    - parentStrand = 0x04
        - Maximum of a parent (must not be the direct parent)
    - Pmax = 0x08
        - General Pmax
- [demand]
    - Boolean flag that is set when Wallbox has a demand (connected and charging activated
- [priority]
    - Boolean flag when Wallbox has a priority demand

ACControl

Covered Config Items

- ∉ AC Control

Details

- type
    - Configured AC Type
- modes
    - Json-array with used modes
- airflow
    - Json-array with used airflow-modes
- fanspeed
    - Json-array with used fanspeeds
    - since 15.1
        - Off mode
- connectedInputs
    - Bitmask of Connected Inputs
    - Bit1: toggle (1 << 0 - 0b0001)
    - Bit2: On (1 << 1 - 0b0010)
    - Bit3: Off (1 << 2 - 0b0100)
    - Bit4: ϑt (1 << 3 - 0b1000)
    - Bit5: ϑc (...)
    - Bit6: Mode
    - Bit7: Fan
    - Bit8: ADir
    - Bit9: Door/Window contact
    - Bit10: Pause Timer
    - Bit11: Load Shedding
    - Bit12: Reset to Default

    - Bit13: Silent Mode
- connectedParameters
    - Bitmask of connected Parameters
    - Bit2: externallyControlled
- externalControlled
    - since 15.1
    - true if AC is externally controlled by an IRC
- UuidExternal
    - since 15.1
    - Optional;Uuid of connected IRC if configured
- UuidClimate
    - since 15.1
    - Optional; uuid of configured AC Climate Controller if configured

States

- status
    - Power-Status(On/Off)
- mode
    - Active mode (1-5)
- fan
    - Active fanspeed(1-6)
- defaultFan
    - Fan speed set when control is powered off
    - since 15.1
- ventMode
    - Active airflow mode (1-7)
- operatingmodes
    - Json-array with configured mode names
- fanspeeds
    - Json-array with configured fanspeed names
- airflows
    - Json-array with configured mode names
- pauseUntil
    - When unequal 0 a sleep timer is active
    - Equal -1 when sleep input is constant on
    - UTC Unix-Seconds Timestamp
- pauseReason
    - Bitmask reason why the AC is paused (Off)
    - When equal 0 block is not paused or Off intentionally (only set when desired state is active)
    - pause_timer = 0x01
        - Paused by timer
    - pause_doorWindow = 0x02,
        - Paused by opened window/door
    - pause_loadShed = 0x04
        - Paused by load shedding
    - pause_silentMode = 0x08
        - Since V15.2

        - Paused by Silent Mode
    - pause_prohibitedHeat = 0x10
        - Since V15.1
        - Paused because desired heating mode is prohibited by Ac Climate Controller
    - pause_prohibitedCool= 0x20
        - Since V15.1
        - Paused because desired cooling mode is prohibited by Ac Climate Controller
        -
    - pause_AccDiffmode = 0x40
        - Since V15.1
        - The Ac-Climate controller (outdoor unit) is currently in a different heating/cooling mode
    - pause_IrcDiffmode = 0x80
        - Since V15.6
        - The connected Intelligent Roomcontroller is currently in a different heating/cooling mode
- pauseTime
    - Value of pause time parameter (Smt)
    - Seconds
- temperature
    - Current Room Temperature
- ⊄ targetTemperature
    - Target Temperature for the AC
- ⊄ silentMode
    - Since V15.1
- ⊄ Override
    - JSON -serialized status of override-timer (see command startoverride)
    - Available JSON-keys
        - ⊄ mode: active mode
        - ⊄ fanspeed: active fanspeed
        - ⊄ airflow: active airflow (vent-mode)
        - ⊄ target: target-temperature
        - ⊄ until:unix timestamp of end of this override timer
        - ⊄ value 0: no override timer is active
- ⊄ settings
    - Since V15.2
    - Settings json which can also be retrieved via "/getsettings"
- minTemp
    - Since V15.3
    - minimal possible value for target temperature
- maxTemp
    - Since V15.3
    - maximum possible value for target temperature

Commands

- toggle
    - Toggle on/off
- on

V17.0

    - Set power on
- off
    - Set power off
- setTarget/<temp>
    - Set target temperature
- setMode/<mode>
    - Set operating mode
- setFan/<mode>
    - Set fan speed mode
- setAirDir/<mode>
    - Set airflow mode
- setmodename/<ID>/<name>
    - Update name for operating mode
- setairflowname/<ID>/<name>
    - Update airflow name
- setfanspeedname/<ID>/<name>
    - Update fanspeed-name
- setoverride/0
    - since V15.1
    - stops overridetimer
- setoverride/<duration>/<mode>/<fanspeed>/<airflow>/<targettemp>
    - since V15.1
    - starts a override-timer with given parameters
    - duration: in seconds
    - other parameters: if 0, current setting is preserved
- startpause/<duration>
    - Start pause timer with the given duration
    - duration: in seconds
- cancelpause
    - Stop pause timer
- setdefaultairflow/<ID>
    - Update default airflow setting
    - Since V15.1
    - 0: Retain Last Value
    - 1: Auto
    - ...
- setdefaultfanspeed/<ID>
    - Update default fanspeed setting
    - Since V15.1
    - 0: Retain Last value
    - 1: Auto
    - ...
- setdefaulttargettemp/<temp>
    - Update default target temperature setting
    - Since V15.1
    - -1: Retain Last Value
- setsilentmode/<ID>
    - Update silent mode setting
    - Since V15.1
    - 0: Off

V17.0

    - 1: Auto
    - 2: ...
- SetAutomaticModeFanSpeed/<ID>
    - Since V15.1
    - Fan Speed Options
- GetSettings
    - Formats the settings in JSON
    - Since V15.1
    - Example:
      {"SilentMode":2,"DefaultAirflow":0,"DefaultFanspeed":0,"DefaultTargetTemp":-1.000}
    - SilentMode: Default Value for Silent Mode
        - 0 Off
    - AutomaticModeFanspeed: Fanspeed which is set when AC is controlled by IRC
        - Fan Speed Options
    - DefaultAirflow: Default Value for Airflow
        - -1: Retain Last Value
    - DefaultFanspeed: Default Value for Fanspeed
        - -1: Retain Last Value
    - DefaultTargetTemp: Default Value for target temperature
        - -1: Retain Last Value

Revision History

17.0

- Intelligent Room Controller
  - Support for fancoils
  - Info about sources
  - Changing of target temperatures
- Presence
  - New API commands

15.3

- LoadManager
  - - added “mode” in details which specifies which working mode is used
  - - added “currentpower2” state used in “Peak Overload” mode for current power
- Window - Added types

15.1

- Wallbox2 - added “ocpp” in details when OCPP is used.
  - When available, prices with value “-1337.0” will be hidden in App
  - Name property is used to specify exter
  - Affected: pricePerkWh, pricePerHour, session-price
- AC Control - Added Default Values, Silent Mode, IRC Integration

15.0

- NFC Code Touch – new state “keyPadAuthType” for authentification parameter
- Audio – bluetooth states and commands

14.7

- MsShortcut Support
- Door and Windowsmonitor – Support for nesting of controls
- Status Monitor Support

14.5

- Wallbox Manager Support
- Wallbox - New states for prices and Manager support

14.4

- Climate Controller - Option to turn it off
- Control History Support
- Statistic Option 15 minute interval

14.2

- Intelligent Room Controller - fixed setpoints support added

---

V17.0

- ACControl - Pause Timer, Load Shedding added
- Garage Door - Partially Open Support added
- Spot PriceOptimizer

14.1

- ACControl
- WallboxV2 support introduced
- Statistics Download via HTTP support added

14.0

- AudioZoneV2 - added volume state
- ClimateController - connected inputs/parameters info added
- AudioZoneV2 - connected parameter info added
- msInfo - now provides info on MS hostname
- EFM - Added custom names/icons for rest (untracked)
- Radiobuttons - prev+next commands

13.2

- NFC Code Touch - KeypadAuthTypes added + additional code types

13.1

- Energy Flow Monitor
- StatisticV2
- Meter
  - Adopted properties to represent new meter types
- EnergyManager2

13.0

- Power Supply and Backup
- Alarm
  - nextLevelAt with unix-Timestamp, nextLevelDelay deprecated
  - armedAt with unix-Timestamp, armedDelay and armedDelayTotal deprecated
- Audio Zone
  - volumeModes - provides info on external volume control (none, external absolute value, external up/down pulses)
- Climate Controller
  - standbyReason - provides info on why it is not be heating or cooling
- ColorPickerV2
  - Support for daylight sequences
  - Support for tunable white only
- IRCv2
  - New possible capability for passive cooling
- Intercom

V17.0

    - showBellImage detail specifying whether to show the image or stream when bell is rang
- IntercomV2
    - Support for trust.
- NFC Code Touch
    - Attribute provides insight on whether a code has been provided or not.
- LightControllerV2
    - supports short names for lighting moods
    - support for human centered lighting.
- Pushbutton
    - provides insight on which type of pushbutton it represents (e.g. scene)

12.2

- Sequential Controller support
- Support for new Intercom (IntercomV2)

12.1

- New US HVAC block
- New features in IRCv2
    - Additional cooling comfort temperature
    - Presence-dependent daytimer entries
- Jalousie - Support for end position adjustment (expert mode only)
- New Load Manager block
- API for Energy Manager
- New Irrigation block
- API for Pulse At
- Geo-Location provided in msInfo (latitude, longitude and altitude)

12.0

- Reset controls to predefined presets
- Locking and Unlocking Controls via API
- Controls may now have other controls linked to them
- LightControlV2 - circuit names may now be modified via API
- IRCv2 - added support for temperature limits
- ClimateController - informs on excess energy available