



EZcontrol[©] XS1

Kommunikationsschnittstelle

Benutzerbefehlssatz (Auszug)

HTTP / JSON

Protokoll Ver. 15

(ab FW 2.0.0.1544)

EZcontrol® XS1 Dokumentation der Benutzer Programmierschnittstelle v15 - Ausgabe 1

Alle in dieser Bedienungsanleitung genannten Marken und Firmennamen sind eingetragene Warenzeichen der entsprechenden Firmen. Die Nennung von Firmen- und Markennamen sowie Produktbezeichnungen hat lediglich beschreibenden Charakter und dient zur Identifizierung der genannten Geräte. Ihre Nennung in dieser Bedienungsanleitung erfolgt in Anerkennung sämtlicher Rechte ihrer jeweiligen Eigentümer. Alle Warenzeichen und Schutzrechte werden anerkannt.

Für Fehler technischer oder drucktechnischer Art in dieser Bedienungsanleitung und deren Folgen übernehmen wir keine Haftung. Änderungen, die dem technischen Fortschritt dienen, können ohne Vorankündigung vorgenommen werden.

Rose + Herleth GbR
Wendenweg 17
13595 Berlin
Tel. +49 (0)30 369 91554
Fax +49 (0)30 362 83064
www.ezcontrol.de

Inhaltsverzeichnis

Einleitung	4
Beispiel: Abfrage eines Aktor-Zustands	5
Sicherheitshinweise	6
Befehlstypen Kurzerklärung	8
Allgemeine Begrenzungen	9
Fehlermeldungen	10
Pakettypen	11
get_protocol_info – Protokollinformation auslesen	11
get_config_info – Geräteinformationen auslesen	12
get_date_time – Datum und Zeit auslesen	14
get_list_actuators – Liste aller Aktoren auslesen	16
get_list_sensors – Liste aller Sensoren auslesen	18
get_list_timers – Liste aller Timer auslesen	20
get_state_actuator – Aktor auslesen	21
set_state_actuator – Aktor Zustand setzen	22
get_state_sensor – Sensor auslesen	25
get_state_sensor – Auslesen des aktuellen Wertes und alter, auf der Speicherkarte gespeicherter Werte	26
get_state_sensor – Auslesen des aktuellen Wertes und der auf der MMC Karte gespeicherten Statistikdaten	28
get_state_sensor – Download gespeicherter Sensordaten in CSV Format	30
set_state_sensor – Sensorwert setzen	31
subscribe – Aktor/Sensor Abonnement	33
History / Changelog	35

Einleitung

Bei der Programmierschnittstelle des EZcontrol XS1 handelt es sich um eine Kombination von HTTP GET Requests (HTTP URLs) und Antworten in Textform, die das Datenformat *JSON* (*JavaScript Object Notation*) benutzen.

Das Datenformat JSON wurde gewählt, da nur dieses eine Kommunikation über Domaingrenzen hinweg ermöglicht.

Dies ist notwendig, damit eine JavaScript/AJAX Anwendung lokal auf dem jeweiligen Client liegen kann („lokale“ Domain) und trotzdem mit dem EZcontrol XS1 (entfernte Domain) kommunizieren kann.

Lange Ladezeiten der Anwendung über die eventuell schmalbandige Internetverbindung entfallen dadurch und auch die Datenmenge wird klein gehalten.

Des Weiteren lassen sich mehrere XS1 Geräte in verschiedenen Domains steuern, ohne die Anwendung neu laden zu müssen.

Die Sicherheitseinstellungen der Browser lassen in der Regel eine andere Art der Kommunikation, wie zum Beispiel XML, über Domaingrenzen hinweg nicht zu.

Kompatibilität

Cross-Domain JSON wird unterstützt und wurde von uns getestet mit folgenden Webbrowsern:

- Internet Explorer 6, 7, 8 (Active Scripting muss erlaubt werden)
- Mozilla Firefox 3.0.x, 3.5.x, 3.6.x
- Mozilla Firefox 2.0.0.x (nur mit Plugin <http://crypto.stanford.edu/jsonrequest/>)
- Opera 9.x, 10.10
- Google Chrome 4.x (Windows/Linux/Mac OS X), 5.0.375.55 (Linux)
- SRWare Iron 0.2.152.0
- Safari 3.1.2 (525.21) (Windows)
- Safari 4.0.3, 4.0.5 (Mac)

Beispiel: Abfrage eines Aktor-Zustands

Hier ein einfaches Beispiel:

Anfrage

[http://192.168.1.242/control?
callback=cname&cmd=get_state_actuator&number=1](http://192.168.1.242/control?callback=cname&cmd=get_state_actuator&number=1)

Auf dem EZcontrol XS1 mit der IP 192.168.1.242 wird die „virtuelle“ Seite *control* mit den entsprechenden Parametern aufgerufen.

Hierbei bedeutet:

callback : Der Variablen *callback* ist ein frei wählbarer Name (**max. Länge 19 Zeichen**) zuweisbar. Dieser spielt nur eine Rolle bei der Verwendung einer Callbackfunktion in JavaScript. **Wird dieser Parameter weggelassen, bzw. nicht zugewiesen, wird kein JSON Antwortpaket geschickt.** Dies kann zur Einbettung von Schaltlinks in HTML Seiten Sinn machen, bei denen keine Antwort erwünscht ist (wie bei EZcontrol T-10 Schaltlinks).

cmd : Command. Bestimmt den Befehl, den das XS1 ausführen soll.

Antwort

Die Antwort wird als Text mit dem Mimetype *application/json* ausgeliefert.

Dieser Typ wird wahrscheinlich mit keiner Anwendung verknüpft sein, es empfiehlt sich diesen Mimetype einfach mit einem Texteditor zu verknüpfen oder als Datei abspeichern:

```
cname({
  "version": 15,
  "type": "get_state_actuator",
  "actuator": {
    "number": 1,
    "name": "Lampe",
    "type": "switch",
    "value": 0.0,
    "unit": "%",
    "utime": 1224135475,
    "date": {
      "weekday": "th",
      "day": 16,
      "month": 10,
      "year": 2008
    },
    "time": {
      "hour": 6,
      "min": 37,
```

```
        "sec": 55
      }
    }
  })
```

Falls die Verarbeitung des Paketes mit JavaScript erfolgt, kann nun die Callbackfunktion beim Erhalt des Paketes automatisch ausgeführt werden. Im Beispiel hat diese den Namen *cname*.

JSON ist strukturell ähnlich aufgebaut wie XML und es existieren Implementierungen schon in diversen Programmiersprachen. Nähere Informationen zur JSON Syntax und Struktur finden Sie z.B. bei Wikipedia und <http://www.json.org>.

Hinweis

Einige Webbrowser cachen ggf. die Antworten, d.h. wenn mehrmals die gleiche Anfrage (URL) gesendet wird, muss der Callbackfunktionsname variiert oder ein frei wählbarer Parameter angehängt werden, damit die Anfrage auch wirklich gesendet wird.

Sicherheitshinweise

Neben denen in der Bedienungsanleitung+Beiblatt genannten Sicherheitshinweisen zum Betrieb des Gerätes sind folgende speziell bei der Programmierung zu beachten:

- Wie bei allen Funkschaltssysteme im 868-868,6 MHz Bereich ist nur eine durchschnittliche maximale zeitliche Sendedauer von 1% in diesem Bereich rechtlich zulässig. Dies hat der Nutzer bei der Nutzung des Gerätes, d.h. auch durch Programmierungen im Gerät und Ansteuerung von außen, sicherzustellen. Dies gilt entsprechend mit einem Maximum von 10% Sendedauer im 433 MHz Bereich.
- Zyklische/automatisierte Schreibvorgänge von Konfigurationsdaten (Aktor-, Sensor-, Timer-, Script-, Raum-Einstellungen) und Uhrzeiteinstellung dürfen nicht vorgenommen werden, da diese langfristig den internen Flashspeicher abnutzen und das Gerät zerstört wird. Dies betrifft natürlich nicht das Setzen oder Auslesen von Aktor- und Sensorzuständen, **der veröffentlichte Benutzerbefehlssatz in diesem Dokument darf uneingeschränkt genutzt werden**. Bei normaler Benutzung ist eine sehr lange Lebensdauer zu erwarten, da das Gerät u.a. über eine interne Verteilung der Schreibvorgänge verfügt (Wear-Leveling). Die vom Chiphersteller garantierte Erhaltungszeit der Konfigurationsdaten liegt bei einem Minimum von 20 Jahren.

JavaScript Programmierhinweise

Zugriff auf JSON Datenelemente per JavaScript

Wenn auf Elemente zugegriffen wird, bei denen der JSON Elementnamen, die gleiche Bezeichnung hat, wie ein reservierter JavaScript Name, z.B. *function*, so ist an Stelle der Syntax

```
arg.timer.actuator.function
```

folgender zu verwenden:

```
arg.timer.actuator['function']
```

Bitte beachten Sie unsere XS JavaScript Programmierbeispiele unter *Support/XS1* bzw. auf der XS1 Produktbeschreibungsseite unter *Programmierbeispiele*. Diese werden wir noch erweitern.

Serialisierung von HTTP Anfragen

Eine HTTP Anfrage sollte erst an das XS1 geschickt werden, wenn die letzte Anfrage abgearbeitet wurde, d.h. die Anwendung die JSON Antwort des vorherigen Befehls erhalten hat.

Sollte dies nicht geschehen, kann die Anzahl der gleichzeitigen TCP Verbindung stark anwachsen. Da die maximale Verbindungsanzahl des XS1 begrenzt ist, kann dies dazuführen, dass das Gerät die Anfragen nicht mehr verarbeiten kann.

JavaScript Programmierbeispiele zur Serialisierung werden wir noch veröffentlichen.

Info:

Bei *set_state_actuator* erfolgt die Bestätigung sofort, solange die Sendebefehlswarteschlange nicht gefüllt ist. D.h. erst wird nicht erst gewartet, die der Funkbefehl gesendet wurde.

Befehlstypen Kurzerklärung

Allgemeine Befehle

[get_protocol_info](#)

Version des Kommunikationsprotokolls abfragen

[get_config_info](#)

Allgemein Gerätestatusinformationen (nicht änderbar)

Gesamtliste eines Objekttyps auslesen

Kurze Übersichtsliste über alle gespeicherten Objekte liefern die folgenden Befehle.

[get_list_actuators](#)

Liste aller Aktoren

[get_list_sensors](#)

Liste aller Sensoren

[get_list_timers](#)

Liste aller Timer

Aktuellen Schaltzustand / Sensorwert auslesen

[get_state_actuator](#)

Aktuellen Zustand eines Aktors auslesen oder auf der Speicherkarte gespeicherte Werte auslesen

[get_state_sensor](#)

Aktuellen Zustand eines Sensors auslesen oder auf der Speicherkarte gespeicherte Werte auslesen

Aktuellen Schaltzustand / Sensorwert setzen

Das Setzen des Zustandes eines Sensors ist nur bei virtuellen Sensoren (System *virtual*) möglich, die keine Daten von empfangenen Sensoren darstellen.

Dieses soll ermöglichen Messwerte/Zustände von Drittprogrammen in das XS1 einspeisen zu können und diese dort in Scripten zu verwenden oder auf der Speicherkarte zu protokollieren.

[set_state_actuator](#)

Zustand eines Aktors setzen

[set_state_sensor](#)

Zustand eines Sensors setzen (nur System *virtual*)

Allgemeine Begrenzungen

Maximale Länge aller Namen (name=) :

19 Buchstaben (intern 20, inklusive Null-Begrenzer)

Maximale Anzahl der jeweiligen Objekttypen (number=):

SYSTEMS	15
ACTUATORS	64
SENSORS	64
TIMERS	128
SCRIPTS	32

(**Firmwareabhängig**, kann durch den Befehl `get_config_info` abgefragt werden)

Die **Numerierung** der Objekte und auch Aktor Unterfunktionen erfolgt generell **von 1 an**.

D.h. z.B. Aktor 1 bis 64 sind adressierbar.

Fehlermeldungen

```
01 invalid command
02 cmd type missing
03 number/name not found
04 duplicate name
05 invalid system
06 invalid function
07 invalid date/time
08 object not found
09 type not virtual
10 syntax error
11 error time range
12 protocol version mismatch
```

Beispiel:

```
cname({
    "type": "void",
    "error": "01"
})
```

Pakettypen

get_protocol_info – Protokollinformation auslesen

Anfrage (Client -> XS1):

http://192.168.1.242/control?callback=cname&cmd=get_protocol_info

Antwort (XS1 -> Client):

```
cname({  
  "version": 15,  
  "type": "get_protocol_info"  
})
```

Die Versionsnummer ist eine positive 16 Bit Ganzzahl.

Den Versionsstand dieser Protokolldokumentation können Sie der Kopfzeile oben entnehmen.

get_config_info – Geräteinformationen auslesen

Alle nicht über die Programmierschnittstelle veränderbaren Gerätedaten.

Anfrage (Client -> XS1):

http://192.168.1.242/control?callback=cname&cmd=get_config_info

Antwort (XS1 -> Client):

```
cname({
  "version": 15,
  "type": "get_config_info",
  "info": {
    "devicename": "xs1",
    "hardware": "1.2.0.0",
    "bootloader": "1.0.0.2",
    "firmware": "2.5.0.1688",
    "systems": 21,
    "maxactuators": 64,
    "maxsensors": 64,
    "maxtimers": 128,
    "maxscripts": 32,
    "maxrooms": 64,
    "uptime": 1543660,
    "features": [ "A", "B", "C", "D"],
    "mac": "00:AA:BB:CC:DD:EE",
    "autoip": "off",
    "current": {
      "ip": "192.168.1.242",
      "netmask": "255.255.255.0",
      "gateway": "192.168.1.1",
      "dns": "192.168.1.1"
    },
    "saved": {
      "ip": "192.168.1.242",
      "netmask": "255.255.255.0",
      "gateway": "192.168.1.1",
      "dns": "192.168.1.1"
    }
  }
})
```

devicename	Hostname (genutzt u.a. als DHCP Hostname)
------------	---

hardware, bootloader, firmware	Versionsnummern
systems	Anzahl der unterstützten Funkschaltsysteme
max*	Maximale Anzahl der abspeicherbaren Objekte
uptime	Einschaltdauer seit letztem Gerätestart in Sekunden (neu in v15)
features	Liste der freigeschalteten Features/Optionen (neu in v15)
mac	MAC Adresse
autoip	DHCP an/aus
current *	Aktuelle IP Einstellungen (wenn DHCP an, dann die bezogene IP)
saved *	Abgespeicherte statische IP Daten, nur wirksam, wenn DHCP ausgeschaltet ist

get_date_time – Datum und Zeit auslesen

Die interne Batterie gestützte Echtzeit Uhr (RTC) arbeitet mit UTC / GMT Zeit.

Anfrage (Client -> XS1):

http://192.168.1.242/control?callback=cname&cmd=get_date_time

Antwort (XS1 -> Client):

```
cname({
  "version": 15,
  "type": "get_date_time",
  "utime": 1275385978,
  "date": {
    "weekday": "tu",
    "day": 1,
    "month": 6,
    "year": 2010
  },
  "time": {
    "hour": 11,
    "min": 52,
    "sec": 58
  },
  "utc_offset": 60,
  "dstmode": "on",
  "dst": "on",
  "ntp": "on",
  "ntp_server": "ntp.ezcontrol.de",
  "ntp_state": "syncd"
})
```

utime	Zeit in Unix Zeit (Sekunden seit 01.01.1970) (UTC Zeit)
date	aktuelles Datum (lokale Zeit)
time	aktuelle Zeit (lokale Zeit)
utc_offset	Zeitunterschied zur UTC/GMT Zeit (Zeitzone) in Minuten (Deutschland: 60)
dstmode	automatische Sommer-/Winterzeitumschaltung AN/AUS (zeigt nicht den aktuellen Zustand Sommer-/Winter an)

ntp	Automatische Zeitsynchronisierung per NTP AN/AUS
ntp_server	Name oder IP Adresse des NTP Servers
ntp_state	Aktueller Zustand der NTP Synchronisierung, mögliche Zustände: <i>synced, error, start, lookup, resolving, resolved, requesting, retry, unknown_host, timeout</i>

get_list Actuators - Liste aller Aktoren auslesen

Jeder gespeicherte Aktor beinhaltet vier Funktionen, die jeweils aus der Art der Schaltfunktion (*type*) und einer Bezeichnung (*dsc*, Description) bestehen.

Anfrage (Client -> XS1):

http://192.168.1.242/control?callback=cname&cmd=get_list_actuators

Antwort (XS1 -> Client):

```
cname({
  "version": 15,
  "type": "get_list_actuators",
  "actuator": [
    {
      "name": "Schalter",
      "type": "switch",
      "value": 0.0,
      "utime": 0,
      "unit": "%",
      "function": [
        {
          "type": "on",
          "dsc": "ON"
        },
        {
          "type": "off",
          "dsc": "OFF"
        },
        {
          "type": "disabled",
          "dsc": ""
        },
        {
          "type": "disabled",
          "dsc": ""
        }
      ]
    }
  ],
},
```

...

```
    {
      "name": "actuator_64",
      "type": "disabled",
      "value": 0.0,
      "utime": 1238164275,
      "unit": "%",
      "function": [
        {
          "type": "disabled",
          "dsc": ""
        },
        {
          "type": "disabled",
          "dsc": ""
        },
        {
          "type": "disabled",
          "dsc": ""
        },
        {
          "type": "disabled",
          "dsc": ""
        }
      ]
    }
  ]
})
```

get_list_sensors - Liste aller Sensoren auslesen

Anfrage (Client -> XS1):

http://192.168.1.242/control?callback=cname&cmd=get_list_sensors

Antwort (XS1 -> Client):

```
cname({
  "version": 15,
  "type": "get_list_sensors",
  "sensor": [
    {
      "name": "Aussentemperatur",
      "type": "temperature",
      "value": 10.0,
      "utime": 1238164313,
      "unit": "°C"
    },
    {
      "name": "Aussenfeuchte",
      "type": "hygrometer",
      "value": 42.5,
      "utime": 1238164313,
      "unit": "%"
    },
    ...
    {
      "name": "Sensor_64",
      "type": "disabled",
      "value": 0.0,
      "utime": 0,
      "unit": ""
    }
  ]
})
```

name	Sensorname
type	Typ des Sensors Insbesondere wichtig für Mehrfachsensoren (z.B. Thermo./Hygro./Baro.) mit der gleichen Adresse

value	Aktueller Wert
utime	Timestamp, Zeitpunkt des letzten Empfangs (UTC)
unit	Maßeinheit (UTF-8 kodiert)

get_list_timers - Liste aller Timer auslesen

Anfrage (Client -> XS1):

http://192.168.1.242/control?callback=cname&cmd=get_list_timers

Antwort (XS1 -> Client):

```
cname({
  "version": 15,
  "type": "get_list_timers",
  "timer": [
    {
      "name": "Schalter_ein",
      "type": "time",
      "next": 1238251680
    },
    {
      "name": "Timer_2",
      "type": "disabled",
      "next": 0
    },
    ...
    {
      "name": "Timer_128",
      "type": "disabled",
      "next": 0
    }
  ]
})
```

name	Timername
type	Timertype (<i>disabled, time, sunrise, sunset</i>)
next	Nächstes Schaltereignis (in 32 Bit Unix Zeit, UTC)

get_state_actuator - Aktor auslesen

Auslesen des aktuellen Wertes

Anfrage (Client -> XS1):

[http://192.168.1.242/control?
callback=cname&cmd=get_state_actuator&number=1](http://192.168.1.242/control?callback=cname&cmd=get_state_actuator&number=1)

Antwort (XS1 -> Client):

```
cname({
  "version": 15,
  "type": "get_state_actuator",
  "actuator": {
    "number": 1,
    "name": "Lampe",
    "type": "switch",
    "value": 0.0,
    "unit": "%",
    "utime": 1224135475,
    "date": {
      "weekday": "th",
      "day": 16,
      "month": 10,
      "year": 2008
    },
    "time": {
      "hour": 6,
      "min": 37,
      "sec": 55
    }
  }
})
```

Der Timestamp gibt den Zeitpunkt des letzten Schaltens per XS1 bei unidirektionalen Aktoren an, bzw. den aktuellen Aktorstatus bei bidirektionalen.

utime	Zeit in Unix Zeit (Sekunden seit 01.01.1970) bei letztem Empfang (UTC Zeit)
date	Datum bei letztem Empfang (lokale Zeit)
time	Zeit bei letztem Empfang (lokale Zeit)

set_state_actuator - Aktor Zustand setzen

PRESET MODE (Abgespeicherte Funktion eines abgespeicherten Aktors auslösen)

Es wird die Nummer des Aktors und die Nummer der Aktorfunktion (1..4) angegeben.

Anfrage (Client -> XS1):

[http://192.168.1.242/control?
callback=cname&cmd=set_state_actuator&number=1&function=1](http://192.168.1.242/control?callback=cname&cmd=set_state_actuator&number=1&function=1)

Antwort (XS1 -> Client):

```
cname({
  "version": 15,
  "type": "set_state_actuator",
  "actuator": {
    "number": 1,
    "name": "Lampe",
    "type": "switch",
    "value": 100.0,
    "unit": "%",
    "utime": 1224135475,
    "date": {
      "weekday": "th",
      "day": 16,
      "month": 10,
      "year": 2008
    },
    "time": {
      "hour": 6,
      "min": 37,
      "sec": 55
    }
  }
})
```

utime	Zeit in Unix Zeit (Sekunden seit 01.01.1970) bei letztem Empfang (UTC Zeit)
date	Datum bei letztem Empfang (lokale Zeit)
time	Zeit bei letztem Empfang (lokale Zeit)

set_state_actuator - DIRECT MODE (Wert eines abgespeicherten Aktors direkt setzen)

Es wird die Nummer des Aktors und der zu setzende Wert (value) angegeben.

Anfrage (Client -> XS1):

http://192.168.1.242/control?callback=cname&cmd=set_state_actuator&number=1&value=50

Antwort (XS1 -> Client):

```
cname({
  "version": 15,
  "type": "set_state_actuator",
  "actuator": {
    "number": 1,
    "name": "Lampe",
    "type": "dimmer",
    "value": 50.0,
    "unit": "%",
    "utime": 1224135475,
    "date": {
      "weekday": "th",
      "day": 16,
      "month": 10,
      "year": 2008
    },
    "time": {
      "hour": 6,
      "min": 37,
      "sec": 55
    }
  }
})
```

utime	Zeit in Unix Zeit (Sekunden seit 01.01.1970) bei letztem Empfang (UTC Zeit)
date	Datum bei letztem Empfang (lokale Zeit)
time	Zeit bei letztem Empfang (lokale Zeit)

set_state_actuator - EZcontrol T-10 kompatibler „PRESET MODE“ zum Aktorschalten

Beispielaufruf:

<http://192.168.1.242/preset?switch=1&value=on>

Schaltet den im EZcontrol XS1 oder T-10 auf Speicherplatz 1 konfigurierten Schalter ein.

Erforderliche Parameter:

Parametername	Wertebereich EZcontrol T-10	Wertebereich EZcontrol XS1	
<i>switch</i>	1..16	1..64	Aktorspeicherplatznummer
<i>value</i>	on/off ab FW 2.25 zusätzlich: 0..100	on/off oder: 0...100	Schaltwert T-10: bei „on“ der gespeicherte Wert gesendet XS1: bei „on“ wird 100 % (an) gesendet

get_state_sensor - Sensor auslesen

Anfrage (Client -> XS1):

[http://192.168.1.242/control?
callback=cname&cmd=get_state_sensor&number=1](http://192.168.1.242/control?callback=cname&cmd=get_state_sensor&number=1)

Antwort (XS1 -> Client):

```
cname({
  "version": 15,
  "type": "get_state_sensor",
  "sensor": {
    "number": 1,
    "name": "AussentempWS433",
    "type": "temperature",
    "value": 3.9,
    "state": [],
    "unit": "°C",
    "utime": 1268312872,
    "date": {
      "weekday": "th",
      "day": 11,
      "month": 3,
      "year": 2010
    },
    "time": {
      "hour": 14,
      "min": 7,
      "sec": 52
    }
  }
})
```

status	Optionale Meldungen des Sensors („Battery empty“ etc.)
utime	Zeit in Unix Zeit (Sekunden seit 01.01.1970) bei letztem Empfang (UTC Zeit)
date	Datum bei letztem Empfang (lokale Zeit)
time	Zeit bei letztem Empfang (lokale Zeit)

get_state_sensor – Auslesen des aktuellen Wertes und alter, auf der Speicherkarte gespeicherter Werte

Anfrage (Client -> XS1):

[http://192.168.1.242/control?
callback=cname&cmd=GET_STATE_SENSOR&number=4&sutime=1224108000&eutime=1224151199](http://192.168.1.242/control?callback=cname&cmd=GET_STATE_SENSOR&number=4&sutime=1224108000&eutime=1224151199)

sutime	start utime Zeitpunkt in der Vergangenheit ab dem die aufgezeichneten Werte abgefragt werden sollen, Zeit in Unix Zeit (Sekunden seit 01.01.1970) (UTC Zeit)
eutime	end utime Zeitpunkt bis zu dem die Werte ausgeliefert werden sollen (exklusiv der letzten Sekunde) Zeit in Unix Zeit (Sekunden seit 01.01.1970) (UTC Zeit)

Alternativ ist die Abfrage auch mit detaillierter Zeitangabe in lokaler Zeit möglich:

sday, smonth, syear, shour smin, ssec,	Startzeit Zeitpunkt bis zu dem die Werte ausgeliefert werden sollen (exklusiv der letzten Sekunde) (lokale Zeit)
eday, emonth, eyear, ehour emin, esec,	Endzeitzeit Zeitpunkt bis zu dem die Werte ausgeliefert werden sollen (exklusiv der letzten Sekunde) (lokale Zeit)

Antwort (XS1 -> Client):

```

cname({
  "version": 15,
  "type": "get_state_sensor",
  "sensor": {
    "number": 4,
    "name": "Aussentemperatur",
    "type": "temperature",
    "value": 9.7,
    "status": [ ],
    "unit": "°C",
    "utime": 1224174070,
    "date": {
      "weekday": "th",
      "day": 16,
      "month": 10,
      "year": 2008
    },
    "time": {
      "hour": 17,
      "min": 21,
      "sec": 10
    }
  },

  "data": [
    { "utime": 1224117198,"value": 9.1 },
    { "utime": 1224117375,"value": 9.7 },
    { "utime": 1224117552,"value": 9.8 },
    { "utime": 1224117729,"value": 9.3 },
    ...
    { "utime": 1224150304,"value": 18.5 },
    { "utime": 1224150481,"value": 18.8 },
    { "utime": 1224150658,"value": 17.4 },
    { "utime": 1224150835,"value": 17.0 },
    ( "utime": 1224151189,"value": 13.2 }
  ]
})

```

Das Antwortpaket entspricht dem Antwortpaket bei der Abfrage des aktuellen Status, hinzukommen die jeweils für die Vergangenheit gespeicherten Daten im `data` Block.

data utime	Zeit des Empfangs in Unix Zeit (Sekunden seit 01.01.1970) (UTC Zeit)
data value	Empfangener Sensorwert

get_state_sensor – Auslesen des aktuellen Wertes und der auf der MMC Karte gespeicherten Statistikdaten

Das Gerät berechnet zu jeder vollen Stunde das Minimum (min), Maximum (max) und den Durchschnittswert eines aufgezeichneten Sensors/Aktors. Diese können mit den gleichen Zeitangaben der normalen Datenabfrage abgerufen werden.

Erklärungen zu den sonstigen Parametern finden Sie bei *get_state_sensor – Sensor auslesen*, Seite 25.

Anfrage (Client -> XS1):

http://192.168.1.242/control?callback=cname&cmd=get_state_sensor&number=1&sutime=1238122800&eutime=1238140800&statistics

Antwort (XS1 -> Client):

```
cname({
  "version": 15,
  "type": "get_state_sensor",
  "sensor": {
    "number": 1,
    "name": "Aussentemperatur",
    "type": "temperature",
    "value": 9.0,
    "state": [ ],
    "unit": "°C",
    "utime": 1238168561,
    "date": {
      "weekday": "fr",
      "day": 27,
      "month": 3,
      "year": 2009
    },
  },
  "time": {
    "hour": 16,
    "min": 42,
    "sec": 41
  }
})
```

```
    },
    "statistics": [
    {
        "utime": 1238122800,
        "avg": 6.6,
        "min": 6.5,
        "max": 6.8
    },
    {
        "utime": 1238126400,
        "avg": 6.6,
        "min": 6.5,
        "max": 6.8
    },
    {
        "utime": 1238130000,
        "avg": 6.9,
        "min": 6.8,
        "max": 7.0
    },
    {
        "utime": 1238133600,
        "avg": 7.6,
        "min": 7.0,
        "max": 8.8
    },
    {
        "utime": 1238137200,
        "avg": 9.7,
        "min": 8.6,
        "max": 11.5
    },
    {
        "utime": 1238140800,
        "avg": 10.3,
        "min": 9.8,
        "max": 11.0
    }
    ]
}
})
```

get_state_sensor – Download gespeicherter Sensordaten in CSV Format

Hier findet zusätzlich der Parameter `&format=csv` Verwendung.

Erklärungen zu den sonstigen Parametern finden Sie bei

`get_state_sensor` – Sensor auslesen, Seite 25.

Anfrage (Client -> XS1):

[http://192.168.1.242/control?
callback=cname&cmd=GET_STATE_SENSOR&number=4&sutime=1224108000&eutime
=1224151199&format=csv](http://192.168.1.242/control?callback=cname&cmd=GET_STATE_SENSOR&number=4&sutime=1224108000&eutime=1224151199&format=csv)

Antwort (XS1 -> Client):

Die Antwort erfolgt mit Mimetype *application/octet-stream*:

```
1271427086;16.4.2010;16:11:26;+0200;S;30;Stromzaehler;pwr_consump;0,0  
1271427386;16.4.2010;16:16:26;+0200;S;30;Stromzaehler;pwr_consump;24,0  
1271427686;16.4.2010;16:21:26;+0200;S;30;Stromzaehler;pwr_consump;288,0  
1271427986;16.4.2010;16:26:26;+0200;S;30;Stromzaehler;pwr_consump;276,0
```

Datensatzaufbau:

Zeit in UTC, Datum, Zeit, UTC Offset, Typ, Nummer, Name, Aktor-/Sensortyp, Wert

Typ: S = Sensor, A = Aktor

set_state_sensor - Sensorwert setzen

Nur in Ausnahmefällen bei „virtuellen“ Sensoren sinnvoll. Es können so z.B. auch externe Datenquellen auf der Speicherkarte mitgeloggt werden oder in Skriptentscheidungen miteinbezogen werden.

Das **Sensorsystem** muss vom Type **virtual** sein, damit die Werte gesetzt werden dürfen. (XS1 interne Skripte dürfen alle Sensoren setzen.)

Anfrage (Client -> XS1):

[http://192.168.1.242/control?
callback=cname&cmd=set_state_sensor&number=8&value=5.00](http://192.168.1.242/control?callback=cname&cmd=set_state_sensor&number=8&value=5.00)

Antwort (XS1 -> Client):

```
cname({
  "version": 15,
  "type": "set_state_sensor",
  "sensor": {
    "number": 8,
    "name": "Sensor_8",
    "type": "temperature",
    "value": 5.00,
    "unit": "°C",
    "timestamp": {
      "utime": 1224174070,
      "date": {
        "weekday": "th",
        "day": 16,
        "month": 10,
        "year": 2008
      },
      "time": {
        "hour": 17,
        "min": 21,
        "sec": 10
      }
    }
  },
},
})
```

utime	Zeit in Unix Zeit (Sekunden seit 01.01.1970) bei letztem Empfang, bzw. gesendeten Befehls (UTC Zeit)
date	Datum bei letztem Empfangs, bzw. gesendeten Befehls (lokale Zeit)
time	Zeit bei letztem Empfangs, bzw. gesendeten Befehls (lokale Zeit)

subscribe - Aktor/Sensor Abonnement

Startet ein Abonnement von allen Aktor / Sensor Ereignissen.
Das XS1 behält die aktuelle Verbindung offen und liefert bei einem Schallereignis, bzw. empfangenen Sensorwert eine Zeile mit Zeitstempel und Wert aus.

Es können verschiedene Ausgabeformate gewählt werden:

HTXT	Human readable Text, lesbarer Klartext (Tab separiert) für direkte Darstellung im Webbrowser
TXT	Daten mit Trennung durch Leerzeichen
CSV	Daten mit Trennung durch Komma
TSV	Daten mit Trennung durch Tab
JSON	Daten im JSON Format

Die Übertragung der Daten bei den Formaten HTXT / TXT / CSV / TSV erfolgt mit dem Mimetype **text/plain; charset=UTF-8** und im **Chunked Transfermode** (HTTP/1.1 , RFC 2616), dies sorgt dafür, dass die empfangenen Daten unmittelbar „live“ im Browser dargestellt werden und die Verbindung permanent offengehalten wird.

Anfrage (Client -> XS1):

<http://192.168.1.242/control?callback=cname&cmd=subscribe&format=htxt>

Antwort (XS1 -> Client):

Format HTXT (Human readable Text):

```
Fri, 17 Oct 2009 10:18:40 Luftdruck          barometer      997.0    hPa
Fri, 17 Oct 2009 10:18:40 Innenfeuchte    hygrometer     30.2     %
Fri, 17 Oct 2009 10:18:40 Innentemperatur temperature    25.1     °C
Fri, 17 Oct 2009 10:18:43 FS20_FB          remotecontrol  0.0
Fri, 17 Oct 2009 10:18:56 Aussenfeuchte  hygrometer     76.8     %
Fri, 17 Oct 2009 10:18:56 Aussentemperatur temperature     10.1     °C
```

Zeitangabe erfolgt in lokaler Zeit.

Format TXT / CSV / TSV:

```
UNIX_Zeit(UTC) Jahr Monat Tag Wochentag Stunde Minute Sekunde Zeitzone Art Nummer Name Wert
1224247936 2009 10 17 Fri 13 52 16 +100 S 7 FS20_FB remotecontrol 0.0
1224248041 2009 10 17 Fri 13 54 1 +100 S 5 Luftdruck barometer 999.0
1224248041 2009 10 17 Fri 13 54 1 +100 S 4 Innenfeuchte hygrometer 27.6
1224248041 2009 10 17 Fri 13 54 1 +100 S 2 Innentemperatur temperature 25.1
```

Art (des Objekts):

A	Aktor
S	Sensor

History / Changelog

Protokollversion 15

- "version" Element in allen Antwortpaketen mit Angabe der Protokollversion
- get_config_info Paket:
 - "features" Element: Liste freigeschalteter Features
 - "uptime" Element: vergangener Zeit seit Gerätestart

Protokollversion 14

- Der Änderungen von Protokollversion 12 auf 14 umfassen nur die Befehle zum Time auslesen bzw. Timer setzen. Alle Befehle und Syntax der Protokoll 11+12 Benutzer-Dokumentation bleiben gültig.

Protokollversion 12

- Anpassung von Skriptfehlermeldungen (Nicht Bestandteil dieser Dokumentation)

Protokollversion 11

Erstes Release

Notizen:

