Spezifikation Home Server API

Specification Home Server API

Für den E-Komfortdurchlauferhitzer DSX Touch

For the E-comfort instantaneous water heater DSX Touch

Inhaltsübersicht

1 Vorwort

Der Home Server des DSX Touch schafft die Voraussetzungen, CLAGE Durchlauferhitzer in ein IP-Netzwerk zu integrieren. Hierdurch wird eine Steuerung und Konfiguration der Geräte sowie die Abfrage von Betriebsdaten unabhängig vom Standort ermöglicht.

Der Server implementiert das CLAGE Funkprotokoll (separat spezifiziert) für die direkte Kommunikation mit dem DSX Touch sowie die im weiteren Verlauf beschriebene Applikationsschnittstelle.
Die Integration und Nutzung der beschriebenen Protokolle ist aktuell nur mit einem DSX Touch möglich.
Die CLAGE GmbH arbeitet zurzeit an verschiedenen Neuerungen, die künftig die Kommunikation mit anderen Modellen ermöglichen.

Die Kommunikation mit externen Applikationen wird über eine REST-basierte API mittels HyperText Transfer Protokoll abgewickelt. Vorzugsweise sollte der Zugriff verschlüsselt via HTTPS erfolgen. Der Server stellt diese Funktionalität über den Standard-Port 443 bereit (ohne Verschlüsselung/HTTP: Port 80).
Die CLAGE App Smart Control kommuniziert bei WLAN-Verbindung über die gleiche Schnittstelle und bildet den möglichen Funktionsumfang ab.

Bei Fragen zur Programmierung REST-basierter APIs wenden Sie sich bitte an entsprechende Fach-Foren im Internet. Die CLAGE GmbH stellt in diesem Bereich keinen Kundenservice zur Verfügung.

Antworten des Servers werden, wenn nicht anders erwähnt, generell mit dem Content-Type "`application/json`" als textbasierte JSON-Datenstruktur ausgeliefert. Auf Füllzeichen (Leerzeichen) etc. wird in der Regel verzichtet, um die Datenmenge zu reduzieren. Für eine bessere Lesbarkeit und Übersicht, werden die in diesem Dokument gezeigten beispielhaften Ausgaben jedoch formatiert dargestellt.

Alle durch die Nutzung der API vorgenommenen Eingriffe in das Gerät obliegen der vollen Verantwortung des Nutzers. Die CLAGE GmbH lehnt jegliche Schadensersatzansprüche aufgrund der Nutzung dieser Schnittstelle durch nicht von der CLAGE GmbH bereitgestellte Software ab.

2 Kommunikation mit dem Home Server

2.1 Einrichtung im Netzwerk

Der Home Server kann unter einer beliebigen IP-Adresse (derzeit nur IP v4) erreichbar sein, Standard-URI: `https://192.168.204.204`

Eine einfache Erkennung von Home Servern im LAN erfolgt über "DNS Service Discovery" (DNS-SD):

Verfügbare Server melden sich mit Service-Type "`_clage-hs._tcp`", IP-Adresse und dem genutzten Port (Standard 443). An dem über dieses Verfahren bekanntgegebenen Port wird stets HTTPS erwartet.

Weitere Infos zu DNS-SD: http://www.dns-sd.org

2.2 Erkennung der unterstützten Services

Der Server gibt unter seiner Adresse die unterstützten Services bzw. Ressourcen und die verwendete API-Version bekannt. Eine Anwendung muss in der Lage sein, daraus das Datenformat für die Kommunikation und weitere, ggf. versteckte oder noch nicht existente Ressourcen für die weitere Verwendung abzuleiten.

Beispiel einer Antwort:

```
{
      "version": "1.3",
      "error": 0,
      "time": 0,
      "knx": false,
      "services": [
          {"deviceList":"/devices"},
          {"deviceStatus":"/devices/status"},
          {"deviceSetpoint":"/devices/setpoint"},
          {"deviceLogs":"/devices/logs"},
      ],
      "server": {
          "id": "0123456789AB",
          "name": "Home Server 1",
          "channel": 106,
          "address": 100
      }
}
```

Jeder Antwort des Servers enthält mindestens die folgenden Daten:

- `version` ➔ API-Version des Servers
- `error` ➔ Fehlerstatus des Servers, 0 = kein Fehler
- `time` ➔ Systemzeit des Servers (Unixtime) in UTC

Optionale Daten:

- `knx` ➔ KNX-Client „CLAGE HS-K“ vorhanden, wenn „true“
- `server` ➔ Informationen über den Server

Interne Daten (nicht zur produktiven Verwendung bestimmt):

- `total` ➔ z.B. Anzahl der Datensätze einer angefragten Liste
- `cached` ➔ Die Anfrage wurde aus dem RAM des Servers bedient, wenn „true“ oder eine direkte Geräte-Kommunikation fand statt, wenn „false“.
- `success` ➔ Die Anfrage wurde erfolgreich bearbeitet, wenn „true“

Bildung des URI für eine Ressource:

z.B. für `deviceStatus` aus dem obigen JSON-Objekt und der Server URI aus 3.1:
`https://192.168.204.204/devices/status`

Hinweis zur Auswertung einer JSON-Response:

Eine Anwendung darf nur die Schlüssel "version", "error", "time", "knx" und das Array/Objekt, das die angeforderten Daten enthält (z.B. "services", "devices"), auswerten.

Die Nutzung der Schlüssel "cached", "success" und "total" wird nicht empfohlen, da diese für interne Zwecke reserviert sind und jederzeit geändert werden könnten.

Ist ein optionaler oder interner Schlüssel nicht vorhanden, darf dies nicht zu einer Fehlfunktion der Anwendung führen.

Unter dem Haupt-URI wird zusätzlich das "server"-Objekt ausgeliefert. Enthalten sind die eindeutige Server-ID (`id`), die Bezeichnung des Servers (`name`), der Funk-Kanal (`channel`) und die Funk-Adresse (`address`). Da für diesen URI keine Authentifizierung notwendig ist, kann eine Applikation bei der Ausführung einer Serversuche (DNS-SD) neue Server von bereits bekannten unterscheiden. Das Server-Objekt ist in jeder Antwort zulässig.

2.3 Authentifizierung

Der Zugriff auf die bereitgestellten Services und Ressourcen ist nur nach entsprechender Authentifikation über einen Benutzernamen und ein Passwort am Home Server möglich. Diese erfolgt bei jedem Request durch "HTTP Basic Authentication" gemäß RFC2617. Voraussetzung ist ein gültiger Account im Home Server für den verwendeten Nutzer. Der Standardnutzer lautet `admin` mit dem Passwort `geheim`.

2.4 Initialisierungsphase

1. Der Client ruft die URI des Home Servers per HTTP GET-Methode auf.
2. Der Client authentifiziert sich bei jedem Request via "HTTP Basic Authentication".
3. Der Home Server sendet ein JSON-Objekt. Dieses enthält die Pfade zu den unterstützten Ressourcen im Objekt `services`.
4. Der Client ruft die Ressource für `deviceList` auf, um alle am Home Server angemeldeten Geräte mit ihrem Grundstatus zu ermitteln.
5. Der Client erkennt neue Geräte und erlaubt deren Ersteinrichtung, wie z.B. deren Raumzuordnung.

2.5 Betriebsphase

- Der Client prüft zyklisch die Ressource `deviceList`, um Status-Änderungen (Sollwert, Status-Flags, Fehler) der Geräte zu erfassen. Minimales Poll-Intervall: 1s. **Vorzugsweise "HTTP Long Polling" verwenden (siehe 3.6), um den Traffic zu reduzieren!**
- Der aktuelle Gerätestatus (ungepuffert) kann über die Ressource `deviceStatus` (siehe 4.2) abgefragt werden. Auch hier ist ein minimales Poll-Intervall von 1s zulässig, um die Anzeige der Leistungsaufnahme und des Wasserdurchflusses ohne merkliche Verzögerungen darstellen zu können. Ein kurzes Poll-Intervall sollte nur verwendet werden, wenn das betreffende Gerät tatsächlich in Betrieb ist (Heizung an, Wasser fließt, siehe auch "5.10 flags") und die Werte für eine Visualisierung in der Anwendung erforderlich sind.
- Der Sollwert eines Gerätes kann jederzeit über die Ressource `deviceSetpoint` gelesen und gesetzt werden (siehe 4.3 und 4.4). Als Antwort wird das Status-Objekt zurückgeliefert. Sehr kurz hintereinander folgende Sollwertänderungen (wie z.B. über Slider generiert) dürfen nicht einzeln übermittelt werden. Erst nach einem angemessenen Timeout (z.B. 2s) ist die Anfrage mit der zuletzt gewählten Solltemperatur zu stellen.

2.6 Vermeidung unnötiger/ zu häufiger Abfragen

Der Home Server erlaubt optional die Verwendung von "HTTP Long Polling" (angelehnt an RFC6202) für die Ressourcen `deviceList` und `deviceLogs`.

Der Server liefert bei einem GET-Request an z.B.
`https://192.168.204.204/devices?lp=1`

erst bei einer Status-Änderung die Antwort an den Client aus. Die Verbindung bleibt demnach so lange geöffnet, bis eine Antwort gesendet worden ist, der Client sie beendet oder ein Timeout auftritt.

Der Wert des Parameters „lp“ muss dem Rückgabewert des globalen Revisionzählers „rev“ der vorherigen Abfrage entsprechen. Initial ist der Wert „1“ zu verwenden.

Für größtmögliche Kompatibilität und Zuverlässigkeit beendet der Server einen Request auch ohne aufgetretene Änderungen nach 30 Sekunden mit einer regulären Anwort. Tritt zuvor ein Ereignis auf, wird die Antwort wie auch im "Short Poll"-Modus als JSON-Objekt ausgeliefert bzw. im Fehlerfall ggf. nur ein entsprechender HTTP-Fehlercode.

Um weitere Änderungen zu erfassen, wiederholt der Client die Anfragen zyklisch (mit aktualisiertem Wert für „lp“), sobald der vorherige Request beantwortet worden ist und wartet jeweils bis zum nächsten Ereignis oder Timeout. Bei einer Änderung von „rev“ obliegt es dem Client, die geänderten Objekte zu ermitteln und auszuwerten. Falls der Server nach 60 Sekunden immer noch keine Antwort ausgeliefert hat, muss der Client die Anfrage selbst beenden. Parallele Long-Polling-Anfragen an dieselbe Ressource sind unzulässig.

3 Spezifikation der Ressourcen

3.1 Abfrage der verfügbaren Geräte

```
GET /devices?<parameter>
```

Antwort:

```
{
      "version": "1.3",
      "error": 0,
      "time": 0,
      "rev": 0,
      "devices": [
            {
                  "id": "1234567890",
                  "busId": 0,
                  "name": "Badezimmer",
                  "rssi": -74,
                  "lqi": 18,
                  "connected": true,
                  "info": {
                        "setpoint": 380,
                        "tLimit": 550,
                        "flags": 0,
                        "error": 0,
                        "access": 65535,
                        "activity": 0,
                        "serverUri": "https://192.168.204.204",
                        "serverCh": 106,
                        "serverAddr": 100
                  }
            }
      ]
}
```

HTTP-Status Codes: 200, 403, 404

Optionale Parameter:

`lp`	„http Long Polling“ verwenden. Der Request schließt erst bei Änderung von `setpoint`, `tLimit`, `flags`, `error` oder nach einem Timeout. Initialwert: „1“, dann auf den Wert von „rev“ der vorherigen Abfrage setzen.
`showBusId`	„true“ oder „1“: nur Geräte mit zugewiesener „busId“ anzeigen
`showErrors`	„true oder „1“: Ausgabe des Arrays „errors“ inkludieren
`showLogs`	„true oder „1“: Ausgabe des Arrays „logs“ inkludieren
`showTotal`	„true oder „1“: statt Ausgabe der letzten Zapfung, Ausgabe der Gesamtsummen (in Verbindung mit „showLogs“)
`showCache`	„true“ oder „1“: auch nicht angemeldete Geräte in der Umgebung anzeigen (s. u.)

alle Geräte listen, die seit Reboot gehört wurden:
bzw. auf den letzten Scan-Request geantwortet haben

```
GET /devices?showCache=true
```

```
[{
      "id": "1234567890",
      "name": "",
      "rssi": -74,
      "lqi": 18,
      "connected": false,
      info: {
            "setpoint": null,
            "flags": null,
            "error": null,
            "access": null,
            "activity": 121414134,
            "serverCh": 106,
            "serverAddr": 101,
            "serverUri": null
      }
}]
```

Unbekannte bzw. noch nicht ermittelte Werte erhalten den Wert `null` oder sind nicht vorhanden.
Bei Geräten, die bereits an diesem Server angemeldet waren, kann hier die Bezeichnung unter „name“ stehen.

Anmeldung

```
PUT /devices/{id}
forcedConnect=true
```

HTTP-Status Codes 200 (erfoglreich angemeldet), 403 (keine Berechtigung), 409 (bereits angemeldet)

Abmeldung

```
DELETE /devices/{id}
```

Gerätesuche starten und Geräteliste aktualisieren

```
POST /devices
autoConnect=false
```

HTTP-Status Code: 202 *ACCEPTED*
Body: wie "`GET /devices`"

Die verfügbaren Geräte melden sich nun innerhalb der folgenden 10 Sekunden beim Server. Daher ist erst nach Ablauf dieser Zeit beim Aufruf von

```
GET /devices?showCache=true
```

die Liste vollständig. Natürlich können auch zwischenzeitlich Abfragen erfolgen, um z.B. den Fortschritt der Gerätesuche zu visualisieren.

Hinweis:
Bei Verwendung von PUT/POST ist die Angabe von "`Content-Type: application/x-www-form-urlencoded`" im HTTP-Header erforderlich.

3.2 Abfrage des Gerätestatus

(Daten werden neu vom Gerät angefordert, falls erforderlich)

```
GET /devices/status/{id}?<parameter>
```

Antwort:


```
{
	"version": "1.3",
	"cached": false,
	"error": 0,
	"time": 0,
	"devices": [
		{
			"id": "1234567890",
			"busId": 0,
			"name": "Badezimmer",
			"rssi": -74,
			"lqi": 18,
			"status": {
				"setpoint": 380,
				"tLimit": 550,
				"tIn": 150,
				"tOut": 380,
				"tP1": 350,
				"tP2": 380,
				"tP3": 420,
				"tP4": 480,
				"flow": 0,
				"flowMax": 254,
				"valvePos": 0,
				"power": 0,
				"powerMax": 0,
				"flags": 0,
				"error": 0
			}
		}
	]
}
```


HTTP-Status Codes: 200, 403, 404, 410

3.3 Abfrage des Sollwertes eines Gerätes

(übrige Daten werden aus dem Cache bedient)

```
GET /devices/setpoint/{id}
```

Antwort:

```
{
          "version": "1.3",
          "cached": true,
          "error": 0,
          "time": 0,
          "devices": [
            {
                  "id": "1234567890",
                  "busId": 0,
                  "name": "Badezimmer",
                  "rssi": -74,
                  "lqi": 18,
                  "status": {STATUS_OBJEKT}
            }
          ]
}
```

HTTP-Status Codes: 200, 403, 404, 410

3.4 Änderung des Sollwertes eines Gerätes

```
PUT /devices/setpoint/1234567890
data=420&cid=1
```

Beispiel:

```
PUT https://192.168.204.204/devices/setpoint/1234567890 HTTP/1.1
Host: 192.168.204.204
Authorization: Basic YWRtaW46Z2VoZWlt
Content-Type: application/x-www-form-urlencoded
Content-Length: 14

data=420&cid=1
```

Alternative für Clients, die keine PUT/POST-Requests unterstützen (intern, bitte nicht verwenden):

```
GET /devices/setpoint/1234567890?_method=PUT&data=420&cid=1
```

Antwort: s.o.

Parameter:

`data`	Sollwert (uint16_t) in 1/10 Grad Celsius

optional:

`_method`	gewünschte HTTP-Methode (PUT, POST); falls der Client diese nicht direkt unterstützt
`cid`	kundenspez. ID (int32_t); wird bis zur nächsten Aktualisierung in allen zukünftigen Log-Einträgen des Gerätes mit der ID `1234567890` vermerkt

HTTP-Status Codes: 200, 403, 404, 410

3.5 Abfrage der Verbrauchswerte (Zapfzyklen)

```
GET /devices/logs/{id}?<parameter>
```

Antwort:

```
{
      "version": "1.3",
      "error": 0,
      "devices": [
            {
                  "id": "1234567890",
                  "busId": 0,
                  "name": "Badezimmer",
                  "rssi": -74,
                  "lqi": 18,
                  "logs": [
                        {
                              "id": 1,
                              "time": 1355266800,
                              "length": 9,
                              "power": 0,
                              "water": 0,
                              "cid": 0
                        }
                  ]
            }
      ]
}
```

Optionale Parameter:

`cid`	Filter für Einträge mit kundenspez. ID
`from_id`	Filter zur Ausgabe der Einträge ab ID
`from_time`	Filter zur Ausgabe der Einträge ab einem Zeitpunkt (Unixtime)

Die Parameter können auch gemeinsam verwendet werden, um die Ergebnismenge weiter einzugrenzen.

Abfrage des letzten Zapfzyklus oder des Gesamtverbrauchs aller Geräte

```
GET /devices/logs?<parameter>
```

Optionale Parameter:

`lp`	„http Long Polling“ verwenden. Request schließt erst bei Änderung oder Timeout. Initialwert: „1“, dann auf den Wert von „rev“ der vorherigen Abfrage setzen.
`showBusId`	„true“ oder „1“: nur Geräte mit zugewiesener „busId“ (>0) anzeigen
`showTotal`	„true oder „1“: statt Ausgabe der letzten Zapfung erfolgt die Ausgabe der Gesamtsummen: power, water, length, time=Zeitstempel letzte Zapfung; etwaige andere Objekte sind zu ignorieren

3.6 Ablage und Abruf von Konfigurationsdateien

Der Client kann mehrere Dateien auf dem Server ablegen. Vorzugsweise sollten die Dateien komprimiert werden (GZIP oder ZIP), da der Speicherplatz limitiert ist. Sofern eine Ermittlung des verfügbaren Speicherplatzes möglich ist, darf das Hochladen nur erfolgen, wenn ausreichend Speicherplatz (zzgl. einer Sicherheitsreserve von mind. 2 MB) vorhanden ist.

Erlaubt sind Ziffern von 0 bis 9, Buchstaben von "a" bis "z" bzw. "A" bis "Z" und die Zeichen "." (Punkt), "_" (Unterstrich) und "-" (Minus). Umlaute oder andere Sonderzeichen dürfen nicht verwendet werden. Es wird zwischen Groß- und Kleinschreibung unterschieden.

Die Länge des Dateinamens ist auf 20 Zeichen begrenzt und kann bis auf die genannten Einschränkungen frei gewählt werden.

Für die Ablage wird das WebDAV-Protokoll verwendet:

```
https://192.168.204.204/files/config/
```

Weiterführende Informationen:

- http://webdav.org
- http://redmine.lighttpd.net/projects/lighttpd/wiki/Docs_ModWebDAV

3.7 Timer hinzufügen und bearbeiten

Am Server angemeldete Geräte können zu vorgegebenen Zeiten automatisch auf eine Wunschtemperatur eingestellt werden.

Der Server kann eine Umschaltung verweigern, wenn z.B.:
- ein Gerät beim Eintreten eines Timer-Events in Benutzung ist
- die letzte Nutzung des Gerätes nicht mindestens x Minuten zurück liegt

```
GET /timers
➔ Ausgabe: alle Timer-Events

GET /timers?deviceId={id}
➔ Ausgabe: alle Timer-Events fuer Geraet {id}

GET /timers/{id}
➔ Ausgabe: Timer-Event {id}
```

Antwort:

```
{
      …

      "timers": [
            {
                  "id": 1234,
                  "enabled": 1,
                  "status": 0,
                  "type": 0,
                  "weekdays": 62,
                  "start": "08:30",
                  "stop": "09:45",
                  "deviceId": "201600069A",
                  "setpoint": 350
            }
      ]
}
```

Hinzufügen:

```
POST /timers
type=0&weekdays=62&start=08:30&stop=09:30&deviceId=201600069A&setpoint=350
```

Ändern:

```
PUT /timers/{id}
weekdays=127
```

Abschalten aller Timer für Gerät {id}:

```
PUT /timers?deviceId={id}
enabled=0
```

Löschen eines Events:

```
DELETE /timers/{id}
```

Löschen aller Timer für Gerät {id}:

```
DELETE /timers?deviceId={id}
```

Hinweis:
Bei Verwendung von PUT/POST ist die Angabe von "`Content-Type: application/x-www-form-urlencoded`" im HTTP-Header erforderlich.

3.8 Interpretation der HTTP Status Codes

Anfragen werden mit einem der folgenden HTTP Status Codes beantwortet.

Genauere Informationen über die mögliche Ursache bei Fehlern (4xx, 5xx), lassen sich dem `error`-Objekt entnehmen, sofern ein JSON-Content mit ausgeliefert wurde (siehe 5.11).

Code	Bezeichnung	Bedeutung im Home Server
200	OK	Anfrage ausgeführt
201	Created	Neue Ressource angelegt
202	Accepted	Anfrage zur weiteren Verarbeitung angenommen
204	No Content	Anfrage ausgeführt, leere Antwort
400	Bad Request	Ressource falsch oder Syntaxfehler
401	Unauthorized	Authentifizierung erforderlich; Zugangsdaten ungültig
403	Forbidden	Berechtigungen nicht ausreichend, Ausführung nicht erlaubt
404	Not Found	Ressource existiert nicht; Gerät ist derzeit nicht erreichbar (ausgeschaltet, Timeout), Gerät nicht vorhanden
405	Method Not Allowed	Methode für die gewünschte Ressource nicht zulässig.
409	Conflict	Zugriffskonflikt beim Ändern/ Erstellen einer Ressource
410	Gone	Gerät nicht verfügbar, war aber zuvor schon einmal angemeldet.
500	Internal Server Error	Unspezifischer Server-Fehler
501	Not Implemented	Methode oder Funktion nicht implementiert.
503	Service Unavailable	Server ausgelastet, Dienst derzeit nicht verfügbar

3.9 Zuweisung einer individuellen Gerätenummer

Jedem angemeldeten Gerät kann eine für den Server einmalige kurze Gerätenummer im Bereich 1 bis 255 (dezimal) vergeben werden. Diese busId identifiziert somit ein Gerät eindeutig und kann ersatzweise für die 10-stellige hexadezimale Gerätekennung genutzt werden. Dies funktioniert allerdings nicht über mehrere Server hinweg. In diesem Fall ist die eindeutige Gerätekennung zu verwenden.

Wird die Gerätenummer nicht (mehr) benötigt, muss die busId auf 0 gesetzt werden. Dies ist auch der Vorgabewert eines neu angemeldeten Gerätes.

Der Client muss sicherstellen, dass je Server keine busId mehrfach vergeben wird. Andernfalls ist bei einem späteren Request nicht vorhersehbar, welches Gerät auf die busId reagieren wird.

Wert ändern

```
PUT /devices/{id}
busId=23
```

Wert löschen

```
PUT /devices/{id}
busId=0
```

Hinweis:
Falls das Gerät noch nicht am Server angemeldet ist, wird hierdurch eine Anmeldung mit gleichzeitiger Zuweisung der Gerätenummer durchgeführt.

3.10 Änderung der Gerätekonfiguration

Einige Betriebsparameter der angemeldeten Geräte lassen sich über die Ressource `deviceSetup` anpassen. Die hierzu vorgesehenen Parameter sind mit „R/W“ gekennzeichnet.

Beispiele:

Durchflussmengenbegrenzung auf AUTO
```
PUT /devices/setup/{id}
flowMax=254
```

Temperaturgrenze bzw. Verbrühschutz auf 55 °C
```
PUT /devices/setup/{id}
scaldProtection=550
```

Lastabwurf ausschalten
```
PUT /devices/setup/{id}
loadShedding=0
```

Signalton am Gerät ausschalten
```
PUT /devices/setup/{id}
sound=0
```

3.11 Vergabe von Server- und Gerätenamen

Für den Server und für die angemeldeten Geräte lassen sich individuelle Bezeichnungen vergeben.

Beispiele:

Gerätenamen ändern
```
PUT /devices/{id}
name=Badezimmer
```

Servernamen ändern
```
PUT /server
name=Home%20Server%201
```

Um eine Bezeichnung zu löschen, kann ein ein Leerstring wie folgt übergeben werden:
`name=%00`

Hinweis:
Bei Verwendung von PUT/POST ist die Angabe von "`Content-Type: application/x-www-form-urlencoded`" im HTTP-Header erforderlich.

3.12 Änderung der Server-Adresse

Jeder Home Server nutzt für die Kommunikation mit den Geräten eine intern festgelegte Adresse zwischen 100 und 199. Werden mehrere Server zusammen in der Nähe eingesetzt, ist es möglich, dass sie werkseitig dieselbe Adresse erhalten haben. Für eine störungsfreie Kommunikation muss allerdings jeder Server eine eindeutige Adresse verwenden. Hierzu müssen alle Server mit derselben Adresse - bis auf den letzten - eine neue, von den anderen Servern unterschiedliche, Adresse zugewiesen bekommen.

Beispiele:

Server-Adresse ändern

```
PUT /server
address=100
```

Server-Adresse zurücksetzen

```
PUT /server
address=-1
```

Die Werte müssen im Bereich von 100 bis 199 liegen. Ungültige Werte führen zu keiner Änderung. Über den Wert -1 lässt sich die Server-Adresse auf den Auslieferungszustand zurücksetzen.

Hinweis:
Bei Verwendung von PUT/POST ist die Angabe von "`Content-Type: application/x-www-form-urlencoded`" im HTTP-Header erforderlich.

3.13 Abfrage der internen Fehlerhistorie eines Gerätes

```
GET /devices/errors/{id}
```

Antwort:

```
{
      "version": "1.3",
      "error": 0,
      "devices": [
            {
                  "id": "1234567890",
                  "busId": 0,
                  "name": "Badezimmer",
                  "rssi": -74,
                  "lqi": 18,
                  "errors": [
                        {
                              "code": 0,
                              "type": 0,
                              "text": "0:SYSTEM OK",
                              "timeL": 12594
                        }
                  ]
            }
      ]
}
```

Hinweis:

Es werden die im abgefragten Gerät intern gespeicherten Systemereignisse zurückgeliefert. Dies sind in der Regel die letzten 10 Statusmeldungen.

Die jeweilige Zeitangabe in `timeL` entspricht dem Wert des internen Betriebsstundenzählers `timerLifetime` zum Zeitpunkt des Ereignisses. Falls vom Gerät unterstützt, kann hier auch das Objekt `time` mit einem Zeitstempel im Unixtime-Format enthalten sein.

4 Spezifikation der Daten-Objekte

Alle Daten, die der Server ausliefert, werden als JSON-Objekt zurückgegeben.

4.1 services (array)

Übersicht der Services und Ressourcen als Array von Objekten gemäß folgender Tabelle:

Key	Datentyp	Default	GET	PUT	POST	DELETE	Beschreibung
`deviceList`	string	/devices	X	X	X	X	Liste der verfügbaren Geräte und An- und Abmeldung
`deviceStatus`	string	/devices/status	X				Gerätestatus
`deviceSetpoint`	string	/devices/setpoint	X	X	X		Sollwert
`deviceSetup`	string	/devices/setup	X	X	X		Gerätekonfiguration
`deviceErrors`	string	/devices/errors	X				Fehlerhistorie; Geräte-intern
`deviceLogs`	string	/devices/logs	X				Verbrauchsdaten
`fileList`	string	/files	X	X	X	X	Ablage für Konfigurationsdateien
`timerList`	string	/timers	X	X	X	X	Timer für automatische Temperatur-Umschaltung

Beispiel:

```
"services": [
      {"deviceList": "/devices"},
      {"deviceStatus": "/devices/status"},
      {"deviceSetpoint": "/devices/setpoint"},
      {"deviceSetup": "/devices/setup"},
      {"deviceLogs": "/devices/logs"},
      {"fileList": "/files"},
      {"timerList": "/timers"}
]
```

Hinweis:

Es dürfen nur Services angesprochen werden, die der Server laut diesem Objekt bereitstellt.

4.2 devices (array)

Die Elemente dieses Arrays können Objekte mit folgendem Inhalt enthalten:

Key	Datentyp	Format/ Einheit	Beispiel	Beschreibung
id	string	*siehe 5.8*	A001010214	Gerätekennung
busId[1]	uint8_t	1 Byte (dez.)	0	individuelle Gerätenummer; eindeutig je Home Server; 0 ➔ keine
syncId[2]	uint8_t	1 Byte (dez.)	0	nicht in API v1.3; reserviert für Synchronisationsgruppe; 0 ➔ keine
name	string		Badezimmer	Bezeichnung des Gerätes
rssi[3]	int8_t	dBm	-76	Funksignalstärke des Gerätes am Server größer ➔ besser
lqi[3]	uint8_t	1 Byte (dez.)	15	Indikator für Verbindungsqualität, kleiner ➔ besser
connected	boolean	*true/false*	true	Gerät an diesem Server angemeldet
info	object	*siehe 5.3*		Geräteinformation
status	object	*siehe 5.4*		Gerätestatus
setup	object	*siehe 5.5*		Gerätekonfiguration
logs	array	*siehe 5.6*		Verbrauchsdaten
errors	array	*siehe 5.7*		Fehlerspeicher
rev	uint8_t	1 Byte (dez.)	2	Revisionszähler, intern

[1] Für jedes Gerät kann zur Identifizierung eine eigene Gerätenummer im Bereich 1 bis 10 vergeben werden. Diese darf pro Home Servers nur einmal vergeben werden. Die busId kann alternativ für die 5-Byte Gerätekennung (auch bezeichnet als: id, uid, Funk-ID) in Abragen verwendet werden.

[2] Bei Geräten einer Synchronisationsgruppe werden die Sollwerte gegenseitig synchronisiert. Falls ein Gerät einer Gruppe den gewünschten Sollwert nicht erreichen kann (Verbrühschutz, Bereichsbegrenzung), wird der Istwert dieses Gerätes als Sollwert der übrigen Gruppenmitglieder verwendet. Ausnahme: Der Istwert ist höher als der gewünschte Sollwert. Bei Mitgliedern der Gruppe 0 erfolgt keine Synchronisierung. Timer werden unverändert ausgeführt und führen entsprechend zur Synchronisierung der gekoppelten Geräte.

[3] Diese Properties werden beim Empfang der Funkpakete aktualisiert und entsprechen daher immer dem letzten Wert vom Stand „activity“. Wurden noch keine Pakete empfangen ist der Wert 255 (lqi) bzw. -255 (rssi).

4.3 info (object)

Information zum aktuellen Betriebsstatus eines Gerätes.

Key	Datentyp	Format/Einheit	Beispiel	Beschreibung
setpoint	uint16_t	1/10 °C	380	Solltemperatur
tLimit	uint16_t	1/10 °C	550	Temperaturgrenze, 0: deaktiviert
tOff	uint16_t	1/10 °C	190	nicht in API v1.3; reserviert für Solltemperatur „Heizung aus“
tMin	uint16_t	1/10 °C	200	nicht in API v1.3; reserviert für kleinstmögliche Solltemperatur
tMax	uint16_t	1/10 °C	600	nicht in API v1.3; reserviert für größtmögliche Solltemperatur
flags	uint8_t	*siehe 5.10*	0	Heizstatus des Gerätes
error	int8_t	*siehe 5.11*	0	Fehlercode des Gerätes/des Servers
access	uint16_t	*siehe 5.12*	16383	Zugriffsrechte des aktiven Kontos
activity	uint32_t	Unixtime	1355266800	Zeitstempel der letzten Funk-Aktivität des Gerätes in UTC
serverUri	string		https://chs.local:443	IP-Adresse/ Hostname des zuständigen Servers, Angabe des Ports möglich
serverCh	uint8_t	1 Byte (dez.)	106	genutzter Funk-Kanal
serverAddr	uint8_t	1 Byte (dez.)	100	genutzte Funk-Adresse

Beispiel:

```
"info": {
      "setpoint": 380,
      "tLimit": 550,
      "flags": 0,
      "error": 0,
      "access": 255,
      "activity": 1355266800,
      "serverURI": "https://192.168.204.204",
      "serverCh": 106,
      "serverAddr": 100
}
```

4.4 status (object)

Gerätestatus

Key	Datentyp	Format/ Einheit	Beispiel	Beschreibung
setpoint	uint16_t	1/10 °C	380	Solltemperatur
tLimit	uint16_t	1/10 °C	550	Temperaturgrenze, 0: deaktiviert
tIn	int16_t	1/10 °C	115	Einlauftemperatur
tOut	int16_t	1/10 °C	379	Auslauftemperatur
tP1	uint16_t	1/10 °C	350	Temperaturspeicher 1
tP2	uint16_t	1/10 °C	380	Temperaturspeicher 2
tP3	uint16_t	1/10 °C	420	Temperaturspeicher 3
tP4	uint16_t	1/10 °C	480	Temperaturspeicher 4
flow	uint8_t	1/10 l/min	25	Wasserfluss
flowMax	uint8_t	1/10 l/min	254	Durchflussmengenbegrenzung 0=AUS, 253=ECO, 254=AUTO
valvePos	uint8_t		0	Stellung des Motorventils
power	uint8_t	*siehe 5.9*	0	Leistungsaufnahme
powerMax	uint8_t		180	Höchstwert der Leistungsaufnahme
flags	uint8_t	*siehe 5.10*	0	Heizstatus des Gerätes
error	int8_t	*siehe 5.11*	0	Fehlercode des Gerätes

Beispiel:

```
"status": {
      "setpoint": 380,
      "tLimit": 550,
      "tIn": 115,
      "tOut": 379,
      "tP1": 350,
      "tP2": 380,
      "tP3": 420,
      "tP4": 480,
      "flow": 25,
      "flowMax": 254,
      "valvePos": 0,
      "power": 0,
      "powerMax": 180,
      "flags": 0,
      "error": 0
}
```

4.5 setup (object)

Gerätekonfiguration

Key	Datentyp	Format/ Einheit	Beispiel	Beschreibung	CX	DX	MX
swVersion	String		1.4.1	Version der Gerätesoftware	X	X	X
serialDevice	String			Seriennummer des Gerätes	X	X	X
serialPowerUnit	String			Seriennummer des Leistungsteils	X	X	X
flowMax	uint8_t	1/10 l/min	254	Durchflussmengenbegrenzung 0/255=aus, 253=ECO, 254=AUTO	-	X	-
loadShedding	uint8_t		0	Lastabwurf; 0=aus	X	X	-
scaldProtection	uint16_t	1/10 °C	420	Verbrühschutztemperatur; 0=aus; entspr. tLimit	X	X	-
sound	uint8_t		0	Signalton; 0=aus	-	X	-
fcpAddr	uint8_t	dez.	80	Adresse	X	X	X
powerCosts	uint8_t		25	Kosten pro kWh (Cent)	X	X	-
powerMax	uint8_t		140	Höchstwert der Leistungsaufnahme	X	X	X
calValue	Integer		2800	interner Kontrollwert	X	X	X
timerPowerOn	uint32_t	s	300	Heizdauer	X	X	X
timerLifetime	uint32_t	s	172800	Gesamtbetriebsdauer	X	X	X
timerStandby	uint32_t	s	2400	Betriebsdauer seit dem letzten Stromausfall	X	X	X
totalPowerConsumption	uint16_t	kWh	0	Gesamtleistungsaufnahme	X	X	X
totalWaterConsumption	uint16_t	Liter	0	Gesamtwassermenge	X	X	X

Beispiel:

```
"setup": {
      "swVersion": "",              R
      "serialDevice": "",           R
      "serialPowerUnit": "",        R
      "flowMax": 254,               R/W
      "loadShedding": 0,            R/W
      "scaldProtection": 420,       R/W
      "sound": 0,                   R/W
      "fcpAddr": 80,                R
      "powerCosts": 0,              R
      "powerMax": 140,              R
      "calValue": 2800,             R
      "timerPowerOn": 300,          R
      "timerLifetime": 172800,      R
      "timerStandby": 2400,         R
      "totalPowerConsumption": 0,   R
      "totalWaterConsumption": 0    R
}
```

4.6 logs (array)

Nutzungsdaten

Key	Datentyp	Format/ Einheit	Beispiel	Beschreibung
id	uint32_t		1	eindeutiger Datensatzindex
time	uint64_t	Unixtime	1355266800	Endzeit der Zapfung in UTC
length	uint32_t	s	10 s	Dauer des Zapfvorgangs
power	uint32_t	1/1 Wh	6 Wh	Energiebedarf
water	uint32_t	1/100 l	0,42 l	genutzte Wassermenge
cid	int32_t		2	kundenspez. ID, die beim Zapfvorgang gesetzt war (über „PUT /devices/setpoint/{id}“)

Beispiel:

```
"logs": [
      {
            "id": 1,
            "time": 1355266800,
            "length": 10,
            "power": 6,
            "water": 42,
            "cid": 2
      }
]
```

Hinweis:

Bei Abfrage von akkumulierten Daten (showTotal=true) sind die Werte von „id“ und „cid“ zu ignorieren, sofern vorhanden.

4.7 errors (array)

Fehlerhistorie

Key	Datentyp	Format/ Einheit	Beispiel	Beschreibung
`code`	int8_t	*siehe 5.11*	0	Fehlercode des Gerätes
`type`	uint8_t	1 Byte (dez.)	0	Fehlerart; 0=kein Fehler; 1=Warnung, 2=Defekt
`text`	String	max. 14 Zeichen		Fehlermeldung, Kurztext
`time`	uint64_t	Unixtime	1355266800	Fehlerzeitpunkt in UTC
`timeL`	uint32_t	s	1200	Wert des Betriebsstundenzählers `timerLifetime` zum Fehlerzeitpunkt

Beispiel:

```
"errors": [
      {
            "code": 0,
            "type": 0,
            "text": "0:SYSTEM OK",
            "time": 1419333267,
      }
]
```

Hinweis:

Der erste Datensatz (Index 0) entspricht immer dem aktuellen Systemzustand. Darauf können weitere Datensätze folgen, die vorherige Fehlerzustände dokumentieren. Der älteste Eintrag steht am Ende des Arrays.

Die Objekte `time` und `timeL` werden in der Regel nicht parallel, sondern in Abhängigkeit der abgefragten Ressource ausgeliefert.

4.8 id (string)

Die eindeutige Gerätekennung besteht aus einer Zeichenkette der Länge 10, die als hexadezimale Repräsentation eines 5-Byte-Integer-Wertes zu sehen ist (MSB links, LSB rechts).

Dieser Wert besteht wiederum aus zwei Teilen, der Typ-ID (uint16_t) und einem Index (uint24_t).

id (5 Byte)				
Byte 4	*Byte 3*	*Byte 2*	*Byte 1*	*Byte 0*
Typ-ID		Index		

Typ-ID (16 Bit)															
15	*14*	*13*	*12*	*11*	*10*	*9*	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Geräteklasse			Varianten-Nr.												

Geräteklasse (3 Bit)			
15	*14*	*13*	*Beschreibung*
0	0	0	D-Serie, nicht steuerbar (z.B. DBX)
0	0	1	D-Serie, steuerbar (z.B. DSX/DEX/DCX)
0	1	0	C-Serie, nicht steuerbar (z.B. CDX)
0	1	1	C-Serie, steuerbar (z.B. CEX/CFX)
1	0	0	M-Serie, nicht steuerbar (z.B. MBX)
1	0	1	M-Serie, steuerbar, (z.B. MCX)

Um Geräte zu erkennen, die keine Änderung der Solltemperatur über ein Bedienfeld erlauben, sondern z.B. eine Sollwertvorgabe durch ein internes Trimmpoti realisieren, gibt Bit 13 der Typ-ID darüber Auskunft. Der Sollwert dieser Geräte darf bzw. kann per Fernsteuerung nicht verändert werden.

Beispiel zur Geräte-Identifizierung:

id = 20160006B1

Typ-ID: 0x2016
Index: 0x0006B1

Ergebnis:

DSX/DEX/DCX, Varianten-Nr. 22 (0x16).

Hinweis:

Die genaue Geräte-Bezeichnung kann über den Schlüssel "Typ-ID" aus einer separaten Tabelle ermittelt werden. Diese Tabelle ist nicht Bestandteil der vorliegenden Dokumentation, kann aber auf Anfrage zur Verfügung gestellt werden.

4.9 power (number)

Enthält die Geräteleistung.

Beispiel:

aktuelle Leistung MCX3 bei power=34:

Powerfaktor:
PF = Gesamtleistung [W] / powerMax
PF = 3500 / 60 = 58

P = PF * power = 58 * 34 = 1972 W = 1,972 kW

Ermittlung der eingestellten Leistung bei DSX:

18kW: powerMax=(18/27)*180=120
21kW: powerMax=(21/27)*180=140
24kW: powerMax=(24/27)*180=160

4.10 flags (number)

Heizstatus des Gerätes

Bit	Bezeichnung	Beschreibung	CX	DX	MX
0	fFlow	kein Wasserfluss	X	X	X
1	reserviert				
2	reserviert				
3	reserviert				
4	reserviert				
5	reserviert				
6	reserviert				
7	reserviert				

Die Heizung ist aktiv, wenn alle Flags gelöscht sind (`flags == 0`).

Hinweis:

Aus Kompatibilitätsgründen bleiben die Bits 1 bis 7 stets 0; somit bedeutet `0x01` „kein Wasserfluss“ und `0x00` „Wasserfluss“. Dieser Wert kann genutzt werden, um die zyklische Abfrage der Ressource `deviceStatus` zu triggern. Hierüber lassen sich die aktuellen Momentanwerte im Heizbetrieb ermitteln.

Für spätere Erweiterungen wird allerdings empfohlen, generell nur auf `Bit0` zu testen und die Werte der anderen Bits zu ignorieren.

4.11 error (number)

Systemstatus/ Fehlercode eines Gerätes.

Positive Werte bedeuten Gerätefehler:

Code	Beschreibung	CX	DX	MX
0	kein Fehler	X	X	X
10	Fehler Bussystem, Bedienfeld defekt?	X	X	X
11	Überspannung	X	X	-
12	Unterspannung	X	X	X
13	Phasenfehler	X	X	X
51	Auslauftemperatur falsch	X	X	X
53	Einlauftemperatur falsch	X	X	-
56	Temperaturfühler am Auslauf defekt	X	X	X
58	Temperaturfühler am Einlauf defekt	X	X	-
59	Temperaturfühler vertauscht	X	-	-
61	Kalibrierwert zu hoch	X	X	-
62	Kalibrierwert zu niedrig	X	X	-
63	Fehler Heizelement	-	-	X
75	Durchfluss zu groß (Luft im System)	X	X	X
76	Auslauftemperatur zu groß (Luft im System)	X	X	X
77	Luftblasen erkannt	X	X	X
80	Initialisierungsfehler Funkmodul	X	X	X
99	Unbekannter Fehler	X	X	X

Negative Werte sind für Fehler des Home Servers reserviert:

Code	Beschreibung
0	kein Fehler
-1	Gerät nicht angemeldet oder nicht (mehr) vorhanden.
-2	reserviert
-3	Timeout, Gerät angemeldet aber antwortet nicht
-4	reserviert
-5	reserviert
-6	reserviert für „busId unbekannt/ nicht zugewiesen“

4.12 access (number)

Über diesen Schlüssel lässt sich ermitteln, ob und in welchem Umfang das jeweilige Gerät über die aktive Authentifizierung ansprechbar ist.

Bit	Bezeichnung	Ressource	freigegebene Methoden	Beschreibung
0	fSetpointRead	deviceSetpoint	GET	Sollwert lesen
1	fSetpointWrite	deviceSetpoint	PUT	Sollwert ändern
2	fStatusRead	deviceStatus	GET	Gerätestatus lesen
3	fStatusWrite	deviceStatus	PUT	Gerätestatus ändern
4	fSetupRead	deviceSetup	GET	Gerätekonfiguration lesen
5	fSetupWrite	deviceSetup	PUT	Gerätekonfiguration ändern
6	fErrorsRead	deviceErrors	GET	Fehlerspeicher lesen
7	fErrorsWrite	deviceErrors	PUT	Fehlerspeicher ändern
8	fLogsRead	deviceLogs	GET	Verbrauchsdaten lesen
9	fLogsWrite	deviceLogs	PUT	Verbrauchsdaten ändern
10	fListRead	deviceList	POST, GET	Geräte suchen, anzeigen
11	fListWrite	deviceList	PUT, DELETE	Geräte an-/abmelden
12	fTimerRead	timerList	GET	Timer lesen
13	fTimerWrite	timerList	POST, PUT, DELETE	Timer erstellen, ändern, löschen
14[4]	fFileRead	fileList	GET	Dateien lesen
15[4]	fFileWrite	fileList	POST, PUT	Dateien erstellen, ändern

Falls bei einer Abfrage zusätzlich auch Objekte enthalten sein könnten, für die das aktuell verwendete Nutzerkonto keine ausreichenden Rechte besitzt, werden diese entweder als `null` zurückgegeben oder sind nicht enthalten.

Besteht für eine gesamte Abfrage keine Berechtigung, wird der HTTP Status Code 403 zurückgeliefert.

Hinweis:
Die Bitbelegung wurde gegenüber API v1.0 geändert und erweitert!

[4] Nicht verwendet. Zugriffsbeschränkung per WebDAV derzeit nicht unterstützt.

4.13 timers (array)

Wiederkehrende zeitgesteuerte Ereignisse zur automatischen Temperatur-Umschaltung.

Key	Datentyp	Format/ Einheit	Beispiel	Beschreibung
id	uint64_t		1234	eindeutiger Datensatzindex
enabled	uint8_t	s.u.	1	Event extern aktiviert (1), deaktiviert (0)
status	uint8_t	s.u.	1	interner Status des Events
Type	uint8_t	s.u.	0	Art des Events
weekdays	uint8_t	-/s/f/d/m/d/m/s s.u.	62	Timer aktiv an folgenden Wochentagen: Mo, Di, Mi, Do, Fr
start	string	hh:mm	08:30	Startzeit des Events
stop	string	hh:mm	09:45	Endzeit des Events
deviceId	string	10 Zeichen (hex.)	A001010214	Ereignis betrifft Gerät mit dieser ID
setpoint	uint16_t	1/10 °C	380	gewünschte Temperatur bei Ereignis

Beispiel:

```
"timers": [
      {
            "id": 1234,
            "enabled": 1,
            "status": 1,
            "type": 0,
            "weekdays": 62,
            "start": "08:30",
            "stop": "09:45",
            "deviceId": "201600069A",
            "setpoint": 350
      }
]
```

status		
Wert	*Bezeichnung*	*Beschreibung*
0	inactive	Timer-Event intern deaktiviert
1	pending	Timer-Event wartend
2	running	Ereignis gerade aktiv
3	cancelled	Ereignis wäre noch aktiv, wurde aber zuvor abgebrochen

type		
Wert	*Bezeichnung*	*Beschreibung*
0	manual	normales Ereignis
1	iTimer	Ereignis wurde aufgrund eines Nutzungsprofils ermittelt und automatisch generiert

weekdays (8 Bit)							
7	*6*	*5*	*4*	*3*	*2*	*1*	*0*
-	Samstag	Freitag	Donnerstag	Mittwoch	Dienstag	Montag	Sonntag

4.14 server (object)

Informationen über den Home Server.

Key	Datentyp	Format/Einheit	Beispiel	Beschreibung
id	string	12 Zeichen (hex.)	0123456789AB	Identifikationsnummer
name	string		Home Server 1	Bezeichnung
channel	uint8_t	1 Byte (dez.)	106	genutzter Funk-Kanal
address	uint8_t	1 Byte (dez.)	100	genutzte Funk-Adresse

Beispiel:

```
"server": {
      "id": "0123456789AB",
      "name": "Home Server 1",
      "channel": 106,
      "address": 100,
      "version": "1.2.2",
      "image": "",
      "revision": "2.1"
}
```

Hinweis:

Zusätzliche bzw. unbekannte Objekte sind zu ignorieren. Sie dürfen nicht zu einer Fehlfunktion der Anwendung führen. Bestandteil der Spezifikation sind ausschließlich die in der Tabelle genannten Objekte.

5 Anwendungsbeispiele

In den folgenden Beispielen wird das Programm „curl“ verwendet, um die Nutzung der API zu demonstrieren.

Voraussetzung ist ein Home Server, der über die IP-Adresse 192.168.204.204 erreicht werden kann. Sollte der Home Server als DHCP-Client im Netzwerk integriert worden sein, ist die entsprechende IP zu verwenden.

5.1 Geräte suchen

Vorhandene Geräte in der Nähe ermitteln:

```
curl -k -u admin:geheim https://192.168.204.204/devices -X POST -d
autoConnect=false
```

Nach 10 Sekunden aktualisierte Geräteliste abfragen:

```
curl -k -u admin:geheim https://192.168.204.204/devices?showCache
```

5.2 Gerät anmelden

Das Gerät „A001FF0034“ explizit am Server anmelden:

```
curl -k -u admin:geheim https://192.168.204.204/devices/A001FF0034 -X PUT -d
forcedConnect=false
```

5.3 Gerät abmelden

Das Gerät „A001FF0034“ explizit am Server abmelden:

```
curl -k -u admin:geheim https://192.168.204.204/devices/A001FF0034 -X DELETE
```

Alternativ kann auch, sofern vergeben, die busId zur Selektierung verwendet werden (Annahme: „busId=1“):
curl -k -u admin:geheim https://192.168.204.204/devices/1 -X DELETE

5.4 Solltemperatur ändern

Die Wunschtemperatur des Gerätes „A001FF0034“ auf 32,0 °C setzen:

```
curl -k -u admin:geheim https://192.168.204.204/devices/setpoint/A001FF0034 -X PUT
-d data=320
```

Alternativ über die busId selektieren:

```
curl -k -u admin:geheim https://192.168.204.204/devices/setpoint/1 -X PUT -d
data=320
```

5.5 Timer ändern

```
curl -k -u admin:geheim https://192.168.204.204/timers?deviceId=A001FF0034 -X PUT
-d enabled=0
```

5.6 Abfrage mit „http Long Polling“

Wenn alle Geräte angemeldet sind, können Änderungen über „http Long Polling“ ohne unnötige Abfragen erfasst werden.

Pseudocode:

```
rev=1;
while (1) {
  json=$(curl -s -k -u admin:geheim
    https://192.168.204.204/devices?
    lp=$rev&showBusId=1&showErrors=1&showLogs=1&showTotal=1
  );
  rev_old=$rev;
  rev=$json.rev;
  if ($rev != $rev_old) {
    print('changed');
  } else {
    print('unchanged');
  }
}
```

Hierbei werden nur Geräte mit registrierter busId (> 0) überwacht, da „showBusId=true“. Nach jeder Zapfung, bei Änderung des Betriebsstatus, bei Systemfehlern oder bei Änderungen des Sollwertes wird der jeweilige Request beendet. Passiert nichts, beantwortet der Server nach 30 Sekunden automatisch den Request.