XSD Public HTTP API

Revision 1.0 - Compatible with legacy XSmart System Trays

HTTP endpoint: `http://[System_IP_Adress]:1337/api/board.`

All API calls described assume the use of the above URL as prefix. JSON is used to represent request/response data.

Note: Do not copy/paste JSON objects in this document for practical use.

Every response from the API server is structured in the following way:

```
{
      "success": true,
      "data": {}
}
```

`success` parameter is a boolean indicating whether request has succeeded.
`data` parameter is a JSON object containing any data appropriate for this response, may be an empty object if there's no data.

If `success` is `false`, there may be additional `error` object in the response that describes the reason why error occurred:

```
{
      "success": false,
      "data": {},
      "error": {
            "code": "ERR_HEATER_TEMP_INVALID",
            "message": "Invalid lower temperature setting"
      }
}
```

Authentication

Most of the API calls require authentication before use. If Control Authentication is enabled in the Web UI, **all** API calls will require authentication. Each API call documented below will have a note on whether authentication is required for it.

```
POST /auth/login
```

JSON parameters:

    username - username used to log in via the web UI
    password - password used to log in via the web UI

Example request:

```
POST http://[boardhost]:1337/api/board/auth/login
{
      "username": "admin",
      "password": "somepassword"
}
```

Response:

```
{
      "username": "admin",
      "admincontrol": true,
      "token": "eyJ0eXAiOiJKV1QiLA0KICJhbGciOiJIUzI1NiJ9"
}
```

`token` received from authentication response must then be used in every API call that requires authentication. Token must be sent in the `Authorization` HTTP header in the following form:

```
      Authorization: Bearer [token]
```

An example HTTP header would be:

```
Authorization: Bearer eyJ0eXAiOiJKV1QiLA0KICJhbGciOiJIUzI1NiJ9
```

In practice token will be a bit longer.

Possible errors:

`401` - this HTTP response code is sent if username or password is incorrect

Token acquired during authentication will be valid for 2 hours. After that, a new token must be obtained by repeating the authentication process.

Board stats

```
GET /stats
```

Authentication: required if Control Authentication is enabled in the UI.

Get current board stats. These include: input current, input voltage, internal temperature, external temperature and fluid level. Temperature values are specified in Fahrenheit.

Example request:

```
    GET http://[boardhost]:1337/api/board/stats
```

Example response:

```
    {
        “success”: true,
        “data”: {
            “current”: 5.84,
            “externaltemp”: 8.21,
            “fluidlevel”: 33,
            “internaltemp”: 239,
            “voltage”: 36.37
        }
    }
```

`fluidlevel` is a number from 0 to 100 indicating the percentage of fluid bottle level.

Load can be calculated from voltage and current values by multiplying them together:

```
    load = current * voltage
```

Due to occasional problems occurring while polling the board for stats, `fluidlevel` parameter can be specified as `null`. If external temperature sensor is not installed, `externaltemp` parameter will also be `null`.

Possible errors:

`401` - unauthorized.

Components

Note: component names are aliased and are different from their standard names. The following aliases are used:

Standard name	Alias
`12VDC_1`	`vdc12_1`
`12VDC_2`	`vdc12_2`
`24VDC_1`	`vdc24_1`
`24VDC_2`	`vdc24_2`
`POE`	`poe`
`PUMP`	`pump`
`NOZZLEHEATER`	`nozzleheater`
`INTERNALHEATER`	`internalheater`
`FAN1AND2`	`fans`
`MOTOR`	`domemotor`
`TURBOFAN`	`turbofan`

```
GET /components/state
```

Authentication: required if Control Authentication is enabled in the UI.

Get the state (on/off) of board components.

Example response:

```
{
     “success”: true,
     “data”: {
          "WATCHDOG_LED": 0,
          "WATCHDOG": 0,
          "vdc12_1": 1,
          "vdc12_2": 1,
          "vdc24_1": 1,
          "vdc24_2": 0,
          "poe": 0,
          "pump": 0,
          "nozzleheater": 0,
          "internalheater": 0,
          "fans": 0,
          "domemotor": 0,
          "turbofan": 0
     }
}
```

`0` - on, `1` - off.

Possible errors:

`401` - unauthorized.

`POST /components/:name/:state`

Authentication: required if Control Authentication is enabled in the UI.

Set the state (on/off) of a single component.

URL parameters:

`name` - component name
`state` - component state, possible values are: `on`, `off`

Example request:

```
http://[boardhost]:1337/api/board/components/vdc24_2/on
```

The above request turns on the `vdc24_2` component.

Example response:

```
{
    “success”: true,
    “data”: {}
}
```

No data is sent back by the server in this case.

Possible errors:

`401` - unauthorized.

Wash/wipe routines:

`GET /routine/intervals`

Authentication: required if Control Authentication is enabled in the UI.

Get interval values for both wash and wipe routines.

Example response:

```
{
      “success”: true,
      “data”: {
            "wash": {
                  "on": false,
                  "interval": 86400000,
                  "lastwash": 1403595000126
            },
            "wipe": {
                  "on": false,
                  "interval": 0,
                  "lastwipe": 1403595000126
            }
      }
}
```

`on` is a boolean indicating whether the ‘wash/wipe every’ setting is on or not
`interval` is the wash/wipe interval in milliseconds
`last[wash/wipe]` is the Unix timestamp in milliseconds specifying last wash/wipe time

If the setting if off, it’s OK that the interval is 0.

`POST /routine/intervals/:routinename`

Authentication: required if Control Authentication is enabled in the UI.

Set interval value for the wash/wipe routine. Also used to enable/disable the routine (see below).

URL parameters:

`routinename` - routine name, possible values are: `wash`, `wipe`

JSON parameters:

`value` - interval value in milliseconds.

Example request:

```
POST http://[boardhost]:1337/api/board/routine/intervals/wash

{
     “value”: 43200000
}
```

The above request will set the wash interval to 12 hours, and thus enable the wash routine if it had been disabled. In the same way, to disable the routine, send `0` in the `value` parameter.

Example response:

```
{
     “success”: true,
     “data”: {}
}
```

Possible errors:

`401` - unauthorized.

```
POST /routine/:routinename
```

Authentication: required if Control Authentication is enabled in the UI.

Trigger wash/wipe routine.

URL parameters:

`routinename` - routine name, possible values are: `wash`, `wipe`.

Example request:

```
POST http://[boardhost]:1337/api/board/routine/wash
```

The above request would trigger the wash routine.

Example response:

```
{
      "success": true,
      "data": {}
}
```

Possible errors:

`401` - unauthorized

May also return `"success": false` response if routine is triggered while the previous trigger is still in progress.

Heater

`GET /xsparams/heater`

Authentication: required if Control Authentication is enabled in the UI.

Get internal heater settings: the mode in which it operates and temperature lower limit.

Example response:
```
    {
        “success”: true,
        “data”: {
            “mode”: “auto”,
            “temp”: “10”
        }
    }
```

`mode` is the mode in which internal heater operates. It’s either `auto` or `off`.
`temp` is the lower temperature limit of the heater represented as a string. Value is in Celsius.

Possible errors:

`401 -` unauthorized.

`POST /xsparams/heater`

Authentication: required if Control Authentication is enabled in the UI.

Set heater mode and lower temperature limit.

JSON parameters:

`mode` - heater mode. Possible values are: `auto`, `off`.
`temp` - heater lower temperature limit. Valid values are: `4-20` (inclusive).

Example response:

```
{
      “success”: true,
      “data”: {}
}
```

Possible errors:

`401 -` unauthorized.

May also return `“success”: false` response with `error` object describing the reason why error occurred.

Startup settings

`GET /xsparams/startup`

Authentication: always required.

Get startup settings for board components. Startup setting indicates whether a component is made `on` or `off` on startup.

Example response:

```
{
  "success": true,
  "data": {
    "12VDC_1": "on",
    "12VDC_2": "on",
    "24VDC_1": "on",
    "24VDC_2": "off",
    "POE": "off",
    "Power Supply Fans": "off",
    "Turbo Fan": "off"
  }
}
```

Possible errors:

`401` - unauthorized.

`POST /xsparams/startup`

Authentication: always required.

Set startup setting for all components.

JSON parameters:

`12VDC_1` - either `on` or `off`
`12VDC_2` - either `on` or `off`

`24VDC_1` - either `on` or `off`
`24VDC_2` - either `on` or `off`
`POE` - either `on` or `off`
`Power Supply Fans` - either `on` or `off`
`Turbo Fan` - either `on` or `off`

Example request:

```
POST http://[boardhost]:1337/api/board/xsparams/startup
{
    "12VDC_1": "on",
    "12VDC_2": "on",
    "24VDC_1": "on",
    "24VDC_2": "off",
    "POE": "off",
    "Power Supply Fans": "off",
    "Turbo Fan": "off"
}
```

Example response:

```
{
    “success”: true,
    “data”: {}
}
```

Possible errors:

`401` - unauthorized

May also return `“success”: false` response with `error` object describing the reason why error occurred.

System settings

Settings correspond to the ones specified in the UI.

```
GET /settings
```

Authentication: always required.

Get all system settings.

Example response:

```
{
  "success": true,
  "data": {
    "network": {
      "dhcp": false,
      "address": "192.168.1.25",
      "mask": "255.255.255.0",
      "gateway": "192.168.1.1",
      "dns1": "8.8.8.8",
      "dns2": "192.168.1.1"
    },
    "system": {
      "name": null,
      "location": null,
      "location_group": null,
      "location_address": null,
      "latitude": null,
      "longitude": null,
      "temp_format": "BOTH"
    },
    "time": {
      "method": "manual",
      "ntp_server": "ntp.hdontap.com",
      "powerup_sync": true,
      "utc_offset_hrs": "+03",
      "utc_offset_mins": "00",
      "sync_interval": "daily",
      "timestring": "2014-09-11T18:59:03Z"
    },
    "controlauth": false,
    "reporting": false
  }
}
```

In `time` settings, if `method` is manual, ntp_server and related settings are irrelevant. They are only considered if `method` is specified as `ntp_server`.

Possible errors:

`401` - unauthorized

`POST /settings/password`

Authentication: always required.

Set system password.

JSON parameters:

`password` - password string (must be a string of 3 or more characters).

Example request:

```
POST http://[boardhost]:1337/api/board/settings/password
{
     “password”: “mynewpassword”
}
```

Example response:

```
{
     “success”: true,
     “data”: {}
}
```

Possible errors:

`401` - unauthorized

May also return `“success”: false` response with `error` object describing the reason why error occurred.

`POST /settings/network`

Authentication: always required.

Set system network settings.

JSON parameters:

- `dhcp` - boolean indicating whether to use DHCP or not
- `address` - IP address
- `mask` - network mask
- `gateway` - network gateway
- `dns1` - first DNS server address
- `dns2` - second DNS server address
- `port` - network port used by web UI

Example request:

```
POST http://[boardhost]:1337/api/board/settings/network
{
  "dhcp": false,
  "address": "192.168.1.25",
  "mask": "255.255.255.0",
  "gateway": "192.168.1.1",
  "dns1": "8.8.8.8",
  "dns2": "192.168.1.1",
  "port": "3333"
}
```

Example response:

```
{
     “success”: true,
     “data”: {}
}
```

Note: network settings take at least a few seconds to change.

Possible errors:

`401` - unauthorized

May also return `“success”: false` response with `error` object describing the reason why error occurred.

`POST /settings/system`

Authentication: always required.

Set system specific parameters.

JSON parameters:

- `name` - system name, can be `null`
- `location` - system location, can be `null`
- `location_group` - system location group, can be `null`
- `location_address` - system address, can be `null`
- `latitude` - system latitude, can be `null`
- `longitude` - system longitude, can be `null`
- `temp_format` - temperature display format (UI specific)

Example request:

```
POST http://[boardhost]:1337/api/board/settings/system
{
  "name": "testSystem",
  "location": "Alaska",
  "location_group": "Alaska Mountains",
  "location_address": null,
  "latitude": null,
  "longitude": null,
  "temp_format": "BOTH"
}
```

Example response:

```
      {
           "success": true,
           "data": {}
      }
```

Possible errors:

`401` - unauthorized

`POST /settings/controlauth`

Authentication: always required.

Enable/disable Control Authentication setting.

JSON parameters:

`controlauth` - boolean indicating whether to enable Control Authentication

Possible errors:

`401` - unauthorized

May also return `"success": false` response with `error` object describing the reason why error occurred.

`POST /settings/time`

Authentication: always required.

Set system time settings.

JSON parameters:

`method` - time specification method, either `manual` or `ntp_server`

`ntp_server` - NTP server for syncing time; ignored if `method` is `manual`

`powerup_sync` - boolean indicating whether to sync time with NTP server on power up; ignored if `method` is `manual`

`utc_offset_hrs` - UTC offset hours, e.g: `+02, -09, 0`; ignored if `method` is `manual`

`utc_offset_mins` - UTC offset minutes, `0-59`; ignored if `method` is `manual`
`sync_interval` - interval of syncing with NTP server; valid values are `daily`, `weekly`, `monthly`; ignored if `method` is `manual`

`timestring` - time value encoded in `YYYY-MM-DDTHH:mm:ss` format; ignored if `method` is `ntp_server`

Example request:

```
POST http://[boardhost]:1337/api/board/settings/time
{
  "method": "manual",
  "ntp_server": "ntp.hdontap.com",
  "powerup_sync": true,
  "utc_offset_hrs": "+03",
  "utc_offset_mins": "00",
  "sync_interval": "daily",
  "timestring": "2014-09-11T19:48:00"
}
```

Possible errors:

`401` - unauthorized

May also return `“success”: false` response with `error` object describing the reason why error occurred.

`POST /settings/reporting`

Authentication: always required.

Enable/disable reporting & alerts.

JSON parameters:

`reporting` - boolean indicating whether to enable or disable reporting

Possible errors:

`401` - unauthorized

May also return `“success”: false` response with `error` object describing the reason why error occurred.