Revision 1.1 - SmartHD v1.0

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

Replace `[System_IP_Adress]` with the IP address or hostname of your SmartHD enclosure. All API calls described below assume the use of the above URL as prefix. JSON is used to represent request/response data.

Note: Be careful if copy/paste JSON objects in this document for practical use - they might contain invalid characters.

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

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

`success` parameter is a boolean indicating whether the 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, containing error `code` and `message` describing the error:

```
{
	"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` - username or password is incorrect.
- `500` - internal error.

Token acquired during authentication will be valid for 24 hours. After that, a new token must be obtained by repeating the authentication process.

Realtime Data

`GET /realtime`

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

Get real time data from the board. This is typically called periodically to have the latest data on hand. Response includes data on tank alarm, tank fill status, scheduled wash/wipe configurations, and system stats.

Stats include: input current, input voltage, internal temperature, external temperature (if applicable), fluid level (if applicable), aux. tank fill pump state (if applicable), internal heater state. Temperature values are specified in Fahrenheit.

See below for more detailed descriptions of the response.

Example request:

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

Example response:

```
{
     "success": true,
     "data": {
          "stats": {
               "current": 5.84,
               "fluidlevel": 33,
               "internaltemp": 239.9,
               "externaltemp": 239,
               "voltage": 36.37,
               "heater_state": "ON",
               "tank_pump_state": "OFF",
          },
          "alarms": {
               "tankAlarm": {
                    "time": 1561466374296,
                    "ack": false
               },
```

```
                    "tankFill": {
                         "inProgress": false,
                         "result:": null
                    }
               },
               "schedules" {
                    "washConfig": {
                         "on": true,
                         "interval": 86400000,
                         "timeout": 13
                         "begintime": 1561466374296,
                         "lastrun": 1561466374296
                    },
                    "wipeConfig": {
                         "on": true,
                         "interval": 0,
                         "timeout": 900000
                         "begintime": 1561466374296,
                         "lastrun": 1561466374296,
                         "motorRunning": true
                    }
               }
          }
     }
```

`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 sent as `null`. If external temperature sensor is not installed, `externaltemp` parameter will also be `null`.

Other parameters:

- `tankAlarm` is an object describing when the alarm was detected (`time` field) and whether it is acknowledged or not (`ack` field).

- `tankFill` is an object describing the current status of the manual tank fill procedure. `inProgress` is set to `true` for as long as the tank fill procedure is running. `result` property describes the result of tank fill procedure as follows:
    - `success` (boolean) indicates whether tank fill completed successfully
    - `error` (string) contains the error code, if any
    - `message` (string) contains the error message, if any
    - `fluidLevelNotMoved` (boolean) indicates whether tank fill failed due to fluid level not advancing within the time window configured by Auto Tank Fill in system settings
    - `pumpFailed` (boolean) indicates whether tank fill failed due to failure of the tank pump
    - `timedOut` (boolean) indicates whether tank fill procedure was stopped by the system due to taking too long
- `washConfig` and `wipeConfig` are identical to what's described in the corresponding API calls below, except for the following:
    - `begintime` is a timestamp indicating when the scheduled wash/wipe procedure began
    - `lastrun` is a timestamp indicating when the last wash/wipe occurred. This is inapplicable for continuous wipe.
    - `wipeConfig.motorRunning` is a boolean indicating the current status of the dome motor

Possible errors:

`401` - unauthorized.
`500` - internal error.

Components

Note: some 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
ACTIVEPOE	apoe
PUMP	pump
TANKPUMP	tankpump
INTERNALHEATER	internalheater
FAN1AND2	fans
MOTOR	domemotor

`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": {
          "vdc12_1": 1,
```

```
            "vdc12_2": 1,
            "vdc24_1": 1,
            "vdc24_2": 0,
            "poe": 0,
            "pump": 0,
            "tankpump": 0,
            "internalheater": 0,
            "fans": 0,
            "domemotor": 0
      }
}
```

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

Possible errors:

`401` - unauthorized.
`500` - internal error.

`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.
- `500` - internal error.

Wash/wipe/tankfill 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,
			"timeout": 33,
			"lastwash": 1403595000126
		},
		"wipe": {
			"on": false,
			"interval": 900000,
			"timeout": 300000,
			"lastwipe": 1403595000126
		}
	}
}
```

- `on` is a boolean indicating whether scheduled wash/wipe is running
- `interval` is the wash/wipe interval in milliseconds. In case of wipe, interval of 0 means that it runs continuously
- `timeout` is a timeout value at which the routine stops:
  - for wash it's a percentage value of the fluid level
  - for wipe it's a time period in milliseconds
- `lastwash/lastwipe` is the Unix timestamp in milliseconds specifying last wash/wipe time

`POST /routine/intervals/:routinename`

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

Configure repeated schedule for wash/wipe routines.

URL parameters:

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

JSON parameters:

- `status` - enable or disable scheduled routine, values are `true` or `false`.
- `interval` - interval value in milliseconds. For wipe, the interval can be set to 0 which means that the dome motor will run continuously until stopped or timed out.
- `timeout` - when to time out (stop repeating the routine): for wash it's percentage value of the fluid level (e.g. `33` - time out when fluid level is at or below ⅓), for wipe it's a time period in milliseconds (e.g. `10800000` - time out after 3 hours).

Example request:

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

{
	"status": true,
	"interval": 43200000,
	"timeout": 13
}
```

The above request will set the wash interval to 12 hours, and thus enable the wash routine if it had been disabled. The timeout is set to 13% of the fluid level, which is roughly fluid level `Reserve`.

Example response:

```
{
	"success": true,
```

```
        "data": {}
}
```

Possible errors:

`401` - unauthorized.

`500` - invalid configuration specified, or fluid level is unknown/too low, or another internal error. Another possible failure reason is trying to change wipe type from scheduled to continuous (or vice versa) if either of them is currently running. You will need to disable the current configuration first.

`POST /routine/:routinename`

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

Trigger `wash`, `wipe` or `tankfill` routine one time.

URL parameters:

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

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

`500` - tank fill already in progress, or another internal error

---

Heater

`GET /xsparams/heater`

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

Get internal heater settings.

Example response:

```
{
    "success": true,
    "data": {
        "mode": "auto",
        "temp": "10",
        "heater_cycle_time": "180",
        "psfans_on_with_heat": "on",
        "psfans_off_delay": "180",
        "itemp_fan_override": {
            "value": "on",
            "tempCelsius": "34"
        }
    }
}
```

All values are specified as strings in this case. See descriptions below.

Possible errors:

`401` – unauthorized.
`500` - failed to read configuration or another internal error.

`POST /xsparams/heater`

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

Set heater settings.

JSON parameters:

`mode` is the mode in which internal heater operates. It's either `auto` or `off`.
`temp` is the lower temperature limit at which the heater will turn on automatically. Value is in degrees Celsius.
`heater_cycle_time` is the duration of the heater cycle when in turns on, in seconds.
`psfans_on_with_heat` is either `on` or `off`. On will force the power supply fans to turn on when heater turns on.
`psfans_off_delay` is the number of seconds the power supply fans will continue to operate after the heater turns off.
`itemp_fan_override` is used to turn on power supply fans when internal temperature exceeds the one specified. `value` is either `on` or `off`, `tempCelsius` is the temperature value.

Example request:

```
POST http://[boardhost]:1337/api/board/xsparams/heater

{
     "mode": "auto",
     "temp": "10",
     "heater_cycle_time": "180",
     "psfans_on_with_heat": "on",
     "psfans_off_delay": "180",
     "itemp_fan_override": {
          "value": "on",
          "tempCelsius": "34"
     }
}
```

(All values are strings).

Example response:

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

Possible errors:

`401` - unauthorized.

`500` - some specified values are invalid or failed to save configuration to disk, or another internal error.

Startup settings

`GET /xsparams/startup`

Authentication: always required.

Get startup settings for board components. Startup setting indicates whether a component is turned 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",
    "ACTIVEPOE": "off"
  }
}
```

Possible errors:

`401` - unauthorized.
`500` - failed to read configuration or another internal error.

`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`
`ACTIVEPOE` - 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
`500` - failed to save configuration or another internal error

System settings

Settings correspond to the ones seen in the web user interface.

```
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"
    },
    "time": {
      "IANAZone": "America/Los_Angeles",
      "countryName": "United States"
    },
    "controlauth": false,
    "firmware_version": "1.0.0",
    "player": {
      "host": "www.example.com",
      "port": "8086",
      "app": "live",
      "streamname": "mystream",
      "autoplay": false,
      "mute": false
    },
    "cid": {
```

```
        "Spraytime": "1400",
        "Wipetime": "1200",
        "DomeMotorStartDelay": "200",
        "DomeMotorStopDelay": "400"
      },
      "auxtank": {
        "Autotankfill": {
          "value": "off",
          "fillLevel": "2",
          "timeoutSec": "60",
          "alarmState": "on"
        },
        "AuxPumpStopFullDelay": "5",
      }
    }
}
```

Possible errors:

`401` - unauthorized
`500` - internal error

`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
`500` - invalid password or another internal error

`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
`500` - invalid network configuration specified or another internal error

---

`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
`500` - invalid parameter given or another internal error

`POST /settings/time`

Authentication: always required.

Set system time settings.

JSON parameters:

`IANAZone` - IANA time zone ID that most closely matches the location of the enclosure

`countryName` - country where enclosure is located. This is only used in the web interface for easier time zone ID filtering. Must be a valid name as seen in the country selection box provided in the web interface.

Example request:

```
POST http://[boardhost]:1337/api/board/settings/time
{
  "IANAZone": "America/Los_Angeles",
  "countryName": "United States"
}
```

Possible errors:

`401` - unauthorized

`500` - invalid IANA time zone or some other internal error

`POST /settings/cid`

Authentication: always required.

Set CID settings.

JSON parameters:

`Spraytime` - the amount of time in milliseconds the fluid is sprayed during the wash function

`DomeMotorStartDelay` - the amount of time in milliseconds after which dome motor (wipe) starts operating after spray function begins

`DomeMotorStopDelay` - the amount of time in milliseconds to delay wipe stop after spraying stops

`Wipetime` - the amount of time in milliseconds to spin the dome during wipe operation

---

All values above must be specified as strings.

Example request:

```
POST http://[boardhost]:1337/api/board/settings/cid
{
  "Spraytime": "1400",
  "DomeMotorStartDelay": "200",
  "DomeMotorStopDelay": "400",
  "Wipetime": "1200",
}
```

Possible errors:

`401` - unauthorized
`500` - failed to save configuration or another internal error

`POST /settings/aux-tank`

Authentication: always required.

Set auxiliary tank settings.

JSON parameters:

`value` - enable or disable auto fill. Valid values are `on` or `off`.
`fillLevel` - fluid level at which to refill the tank. This is specified in thirds of the fluid level. Valid values are `1` and `2` which mean ⅓ or ⅔ thirds of the fluid level.
`timeoutSec` - timeout period within which the fluid reservoir must be filled to the next level, triggering the alarm otherwise (if enabled). Value is in seconds.
`AuxPumpStopFullDelay` - duration in seconds to delay shutting down the aux pump after the tank reads Full.
`alarmState` - turn auto fill alarm on or off. The alarm goes off when the fluid in the reservoir fails to reach the next level within the specified timeout period (see above). Valid values are `on` and `off`.

All parameters above must be sent as strings.

Example request:

```
POST http://[boardhost]:1337/api/board/settings/aux-tank
```

```
{
  "value": "on",
  "fillLevel": "1",
  "timeoutSec": "60",
  "AuxPumpStopFullDelay": "5",
  "alarmState": "on"
}
```

Possible errors:

`401` - unauthorized
`500` - failed to save configuration or another internal error

Tank alarm

`POST /alarm/tank/:action`

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

Clear or acknowledge tank alarm that had been previously detected. Clearing the alarm will re-enable the auto tank fill feature.

URL parameters:

- `:action` is either `clear` or `ack`

Example request to acknowledge the alarm:

`POST http://[boardhost]:1337/api/board/alarm/tank/ack`

Possible errors:

`401` - unauthorized
`500` - invalid alarm action, or failed to clear the alarm, or another internal error