xs1_api_client Documentation

Release 1.0.0

Markus Ressel

Sep 29, 2017

Contents

A python 3.3+ library for accessing actuator and sensor data on the the EZcontrol® XS1 Gateway using its HTTP API.

CHAPTER 1

Build Status

CHAPTER 2

How to use

Installation

```
pip install xs1-api-client
```

Usage

For a basic example have a look at the [example.py] file. If you need more info have a look at the [documentation] which should help.

Basic Example

Create the API Object

The basic way of creating an API object is by providing connection info directly when creating it:

```
from xs1_api_client import api as xs1api
from xs1_api_client import api_constants

# Create an api object with private configuration
api = xs1api.XS1('192.168.2.20', "Username", "Password")
```

This will automatically try to connect to the gateway with the given credentials and retrieve basic gateway information which you can output like this:

```
print("Gateway Hostname: " + api.get_gateway_name())
print("Gateway MAC: " + api.get_gateway_mac())
print("Gateway Hardware Version: " + api.get_gateway_hardware_version())
print("Gateway Bootloader Version: " + api.get_gateway_bootloader_version())
```

```
print("Gateway Firmware Version: " + api.get_gateway_firmware_version())
print("Gateway uptime: " + str(api.get_gateway_uptime()) + " seconds")
```

Now that you have a connection to your gateway we can retrieve its configuration and set or retrieve values of the devices.

Currently there is **no way of setting configuration data** with this library. This means you still have to do all your actuator and sensor configuration using the webinterface of the XS1.

After you have done that you can head over to the next section.

Devices

All devices that you have configured in your XS1 are implemented using the `XS1Device` base class which can be found at `/device/base.py`. This class provides basic functionality for every device like getting the **id**, **name**, **type** and other values.

Retrieve Actuators

To retrieve a list of all actuators that are configured (`type != "disabled"`) use the following call:

```
actuators = api.get_all_actuators()
```

This will return a list of `XS1Actuator` objects which is another base class for all actuators. You can use something like this to print all your actuators:

```
for actuator in actuators:
    print("Actuator " + str(actuator.id()) + ": " + actuator.name() + " (" +␣
↪str(actuator.type()) + ")")
```

Retrieving a single actuator is not yet possible.

Retrieve an Actuator Value

To retrieve the current value of an actuator just call:

```
current_value = actuator.value()
```

Set a new Actuator value

To set a new value to this actuator use:

```
actuator.set_value(100)
```

This will send the required request to the XS1 and set the `new_value` property to your value. Most of the time this value is set instantaneously is in sync with the `value` property. However if this value is different from the standard `value` the XS1 gateway is still trying to update the value on the remote device. For some devices this can take up to a couple of minutes (f.ex. FHT 80B heating).

Updating Actuator Information

Currently there is **no callback** when the value is finally updated so **you have to update the device information manually** if you want to get an update on its current state:

```
actuator.update()
```

After that the usual methods like `actuator.value()` will respond with the updated state.

Executing Actuator Functions

If you have defined function presets for a device you can get a list of all functions using:

```
functions = actuator.get_functions()
```

and print them like this:

```
for function in functions:
    print("Function " + str(function.id()) + " (" + function.type() + "): " +␣
↪function.description())
```

to execute one of the functions use type:

```
function.execute()
```

This will (like set_value) update the device state immediately with the gateways response. Remember though that there can be a delay for sending this value to the actual remote device like mentioned above.

Retrieve a List of Sensors

To retrieve a list of all sensors that are configured (`type != "disabled"`) use the following call:

```
sensors = api.get_all_sensors()
```

This will return a list of `XS1Sensor` objects which is the base class for all sensors.
You can print basic information about them like this:

```
for sensor in sensors:
    print("Sensor " + str(sensor.id()) + ": " + sensor.name() + " (" + str(sensor.
↪value()) + ")")
```

Updating Sensor Information

Just like with actuators there is no automatic updates for sensors either. To get a state update from the XS1 gateway for your sensor object call:

```
sensor.update()
```

After that the complete state of this sensor should be updated.

Disabled Devices

The XS1 allows up to 64 actuator and 64 sensor configurations. These 128 device configurations are accessible via the HTTP API at any time - even when there is nothing configured for a specific device id/number.

To check if a device has been configured in the XS1 web interface call:

```
device.enabled()
```

for both actuators and sensors alike.

CHAPTER 3

Contributing

Github is for social coding: if you want to write code, I encourage contributions through pull requests from forks of this repository. Create Github tickets for bugs and new features and comment on the ones that you are interested in.

CHAPTER 4

License

```
xs1-api-client by Markus Ressel
Copyright (C) 2017  Markus Ressel

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

CHAPTER 5

Content:

API

xs1_api_client package

Subpackages

xs1_api_client.device package

Subpackages

xs1_api_client.device.actuator package

Submodules

xs1_api_client.device.actuator.base module

class xs1_api_client.device.actuator.base.**XS1Actuator**(*state*, *api_interface*)

Bases: *xs1_api_client.device.base.XS1Device*

Represents a basic XS1 Actuator, there may be special variants for some types.

call_function(*function*)

Calls the specified function by id and saves the api response as the new state

Parameters **function** – XS1Function object

get_functions()

Returns a list of functions that can be executed using the call_function() method

set_value(*value*)

Sets a new value for this actuator

Parameters **value** – new value to set

update()
Updates the state of this actuator

class xs1_api_client.device.actuator.base.**XS1Function**(*actuator*, *id*, *type*, *description*)
Bases: object

Represents a function of a XS1Actuator.

description()

Returns a description for this function

execute()
Executes this function and sets the response as the new actuator value

id()

Returns the id of this function (note that this id is only unique for a single actuator!)

type()

Returns the type of this function

xs1_api_client.device.actuator.switch module

class xs1_api_client.device.actuator.switch.**XS1Switch**(*state*, *api_interface*)
Bases: *xs1_api_client.device.actuator.base.XS1Actuator*

Represents a XS1 Switch.

turn_off()
Turns off the switch.

turn_on()
Turns on the switch.

xs1_api_client.device.actuator.thermostat module

class xs1_api_client.device.actuator.thermostat.**XS1Thermostat**(*state*, *api_interface*)
Bases: *xs1_api_client.device.actuator.base.XS1Actuator*

Represents a basic XS1 Actuator, there may be special variants for some types.

set_temperature(*temp*)
Sets the new target temperature of this thermostat

Parameters **temp** – double value

Module contents

xs1_api_client.device.sensor package

Submodules

xs1_api_client.device.sensor.base module

class xs1_api_client.device.sensor.base.**XS1Sensor**(*state*, *api_interface*)
Bases: *xs1_api_client.device.base.XS1Device*

Represents a XS1 Sensor

set_value(*value*)
Sets a value for this sensor This should only be used for debugging purpose! :param value: new value to set

update()
Updates the state of this sensor

Module contents

Submodules

xs1_api_client.device.base module

class xs1_api_client.device.base.**XS1Device**(*state: dict*, *api_interface*)
Bases: object

This is a generic XS1 device, all other objetcs inherit from this.

enabled()

Returns Returns if this device is enabled.

id()

Returns id of this device

last_update()

Returns the time when this device's value was updated last

name()

Returns the name of this device

new_value()
Returns the new value to set for this device. If this value differs from the currrent value the gateway is still trying to update the value on the device. If it does not differ the value has already been set.

Returns the new value to set for this device

set_state(*new_state: dict*)
Sets a new state for this device. If there is an existing state, new and old values will be merged to retain any information that was missing from api responses.

Parameters **new_state** – new representation of this device (api response)

set_value(*value*)
Sets a new value for this device. This method should be implemented by inheriting classes.

Parameters **value** – the new value to set

type()

Returns the type of this device

unit()

Returns the unit that is used for the value

update()
Updates the current value of this device. This method should be implemented by inheriting classes.

value()

Returns the current value of this device

Module contents

Submodules

xs1_api_client.api module

This is the main xs1_api_client api which contains the XS1 object to interact with the gateway.

Example usage can be found in the example.py file

class xs1_api_client.api.**XS1**(*host: str = None*, *user: str = None*, *password: str = None*)
Bases: object

This class is the main api interface that handles all communication with the XS1 gateway.

call_actuator_function(*actuator_id*, *function*) → dict
Executes a function on the specified actuator and sets the response on the passed in actuator.

Parameters

- **actuator_id** – actuator id to execute the function on and set response value
- **function** – id of the function to execute

Returns the api response

get_all_actuators() → [<class 'xs1_api_client.device.actuator.base.XS1Actuator'>]
Requests the list of enabled actuators from the gateway.

Returns a list of XS1Actuator objects

get_all_sensors() → [<class 'xs1_api_client.device.sensor.base.XS1Sensor'>]
Requests the list of enabled sensors from the gateway.

Returns list of XS1Sensor objects

get_gateway_bootloader_version() → str

Returns the bootloader version number of the gateway

get_gateway_firmware_version() → str

Returns the firmware version number of the gateway

get_gateway_hardware_version() → str

Returns the hardware version number of the gateway

get_gateway_mac() → str

Returns the mac address of the gateway

get_gateway_name() → str

Returns the hostname of the gateway

get_gateway_uptime() → str

Returns the uptime of the gateway in seconds

get_protocol_info() → str
Retrieves the protocol version that is used by the gateway

Returns protocol version number

get_state_actuator(*actuator_id*) → dict
Gets the current state of the specified actuator.

Parameters **actuator_id** – actuator id

Returns the api response as a dict

get_state_sensor(*sensor_id*) → dict
Gets the current state of the specified sensor.

Parameters **sensor_id** – sensor id

Returns the api response as a dict

send_request(*command*, **parameters*) → dict
Sends a GET request to the XS1 Gateway and returns the response as a JSON object.

Parameters

- **command** – command parameter for the URL (see api_constants)
- **parameters** – additional parameters needed for the specified command like 'number=3' (without any '&' symbol)

Returns the api response as a json object

set_actuator_value(*actuator_id*, *value*) → dict
Sets a new value for the specified actuator.

Parameters

- **actuator_id** – actuator id to set the new value on
- **value** – the new value to set on the specified actuator

Returns the api response

set_connection_info(*host*, *user*, *password*) → None
Sets private connection info for this XS1 instance. This XS1 instance will also immediately use this connection info.

Parameters

- **host** – host address the gateway can be found at
- **user** – username for authentication
- **password** – password for authentication

static set_global_connection_info(*host*, *user*, *password*) → None
Sets the global connection info. This initialization is valid for all XS1 instances that do not have a specific connection configuration upon instantiation or using the set_connection_info() method. If you want a XS1 instance to use the global info instead of private use the use_global_connection_info() method.

Parameters

- **host** – host address the gateway can be found at
- **user** – username for authentication

- **password** – password for authentication

set_sensor_value(*sensor_id*, *value*) → dict
Sets a new value for the specified sensor. WARNING: Only use this for "virtual" sensors or for debugging!

Parameters

- **sensor_id** – sensor id to set the new value on
- **value** – the new value to set on the specified sensor

Returns the api response

update_config_info() → None
Retrieves gateway specific (and immutable) configuration data

use_global_connection_info() → None
Enables the use of global configuration data

xs1_api_client.api_constants module

XS1 HTTP Web API constants used to create GET request URLs and parse the JSON answer.

xs1_api_client.api_constants.**COMMAND_GET_CONFIG_INFO = 'get_config_info'**
Command to get (final) configuration information about the gateway

xs1_api_client.api_constants.**COMMAND_GET_LIST_ACTUATORS = 'get_list_actuators'**
Command to get a list of all actuators

xs1_api_client.api_constants.**COMMAND_GET_LIST_SENSORS = 'get_list_sensors'**
Command to get a list of all sensors

xs1_api_client.api_constants.**COMMAND_GET_PROTOCOL_INFO = 'get_protocol_info'**
Command to get information about the protocol version used by the gateway

xs1_api_client.api_constants.**COMMAND_GET_STATE_ACTUATOR = 'get_state_actuator'**
Command to get the state of a specific actuator

xs1_api_client.api_constants.**COMMAND_GET_STATE_SENSOR = 'get_state_sensor'**
Command to get the state of a specific sensor

xs1_api_client.api_constants.**COMMAND_SET_STATE_ACTUATOR = 'set_state_actuator'**
Command to set a new value on an actuator

xs1_api_client.api_constants.**COMMAND_SET_STATE_SENSOR = 'set_state_sensor'**
Command to set a new value on a sensor (for debugging)

xs1_api_client.api_constants.**ERROR_CODES = {'09': 'type not virtual', '07': 'invalid date/time', '12': 'protocol vers**
Dictionary with description values for each error code

xs1_api_client.api_constants.**ERROR_CODE_CMD_TYPE_MISSING = '02'**
Error code for 'cmd type missing'

xs1_api_client.api_constants.**ERROR_CODE_DUPLICATE = '04'**
Error code for 'duplicate name'

xs1_api_client.api_constants.**ERROR_CODE_INVALID_COMMAND = '01'**
Error code for 'invalid command'

xs1_api_client.api_constants.**ERROR_CODE_INVALID_DATE_TIME = '07'**
Error code for 'invalid date/time'

xs1_api_client.api_constants.**ERROR_CODE_INVALID_FUNCTION = '06'**
Error code for 'invalid function'

xs1_api_client.api_constants.**ERROR_CODE_INVALID_SYSTEM = '05'**
Error code for 'invalid system'

xs1_api_client.api_constants.**ERROR_CODE_INVALID_TIME_RANGE = '11'**
Error code for 'error time range'

xs1_api_client.api_constants.**ERROR_CODE_NOT_FOUND = '03'**
Error code for 'number/name not found'

xs1_api_client.api_constants.**ERROR_CODE_OBJECT_NOT_FOUND = '08'**
Error code for 'object not found'

xs1_api_client.api_constants.**ERROR_CODE_PROTOCOL_VERSION_MISMATCH = '12'**
Error code for 'protocol version mismatch'

xs1_api_client.api_constants.**ERROR_CODE_SYNTAX_ERROR = '10'**
Error code for 'syntax error'

xs1_api_client.api_constants.**ERROR_CODE_TYPE_NOT_VIRTUAL = '09'**
Error code for 'type not virtual'

xs1_api_client.api_constants.**NODE_ACTUATOR = 'actuator'**
Node with an array of actuators

xs1_api_client.api_constants.**NODE_DEVICE_BOOTLOADER_VERSION = 'bootloader'**
Bootloader version number

xs1_api_client.api_constants.**NODE_DEVICE_FIRMWARE_VERSION = 'firmware'**
Firmware version number

xs1_api_client.api_constants.**NODE_DEVICE_HARDWARE_VERSION = 'hardware'**
Hardware revision

xs1_api_client.api_constants.**NODE_DEVICE_MAC = 'mac'**
MAC address

xs1_api_client.api_constants.**NODE_DEVICE_NAME = 'devicename'**
Hostname

xs1_api_client.api_constants.**NODE_DEVICE_UPTIME = 'uptime'**
Uptime in seconds

xs1_api_client.api_constants.**NODE_ERROR = 'error'**
Node containing the error code

xs1_api_client.api_constants.**NODE_INFO = 'info'**
Node with gateway specific information

xs1_api_client.api_constants.**NODE_PARAM_DESCRIPTION = 'dsc'**
Device description

xs1_api_client.api_constants.**NODE_PARAM_FUNCTION = 'function'**
Array of functions

xs1_api_client.api_constants.**NODE_PARAM_ID = 'id'**
Device id (only unique within actuators/sensors)

xs1_api_client.api_constants.**NODE_PARAM_NAME = 'name'**
Device name

xs1_api_client.api_constants.**NODE_PARAM_NEW_VALUE = 'newvalue'**
New value to set for the device

xs1_api_client.api_constants.**NODE_PARAM_NUMBER = 'number'**
Alternative device id (only unique within actuators/sensors)

xs1_api_client.api_constants.**NODE_PARAM_TYPE = 'type'**
Device type

xs1_api_client.api_constants.**NODE_PARAM_UNIT = 'unit'**
Device value unit

xs1_api_client.api_constants.**NODE_PARAM_UTIME = 'utime'**
Time this device was last updated

xs1_api_client.api_constants.**NODE_PARAM_VALUE = 'value'**
Current device value

xs1_api_client.api_constants.**NODE_SENSOR = 'sensor'**
Node with an array of sensors

xs1_api_client.api_constants.**NODE_VERSION = 'version'**
Node with protocol version info

xs1_api_client.api_constants.**UNIT_BOOLEAN = 'boolean'**
Boolean unit type

xs1_api_client.api_constants.**URL_PARAM_COMMAND = 'cmd='**
command parameter that specifies the method the api is queried with

xs1_api_client.api_constants.**URL_PARAM_FUNCTION = 'function='**
parameter that specifies the function to execute (on an actuator)

xs1_api_client.api_constants.**URL_PARAM_NUMBER = 'number='**
number parameter that specifies the id of an actuator or sensor

xs1_api_client.api_constants.**URL_PARAM_PASSWORD = 'pwd='**
'Password' parameter

xs1_api_client.api_constants.**URL_PARAM_USER = 'user='**
'User' parameter

xs1_api_client.api_constants.**URL_PARAM_VALUE = 'value='**
'value' parameter that specifies the new value to set an actuator (or sensor) to

xs1_api_client.api_constants.**VALUE_DISABLED = 'disabled'**
'Disabled' type

xs1_api_client.test_XS1 module

Module contents

Python Module Index

X

Index

I

L

N

S

T

U

V

X