# JSON / HTTP(s) POST

# NETIO M2M API protocols docs

## Protocol version: JSON Version 2.0

## Short summary

JSON / HTTP(s) protocol is a file-based M2M API protocol, where the NETIO device is a HTTP(s) server and the client downloads or uploads one text file document in the json format to control the NETIO power outputs (230V power sockets or IEC-320 power outlets 110/230V).

- For NETIO 4All, the protocol also includes energy metering values.
- The JSON protocol must be enabled first in the WEB configuration of the respective device. For details, see the "NETIO WEB configuration" chapter.
- This protocol is HTTP(s) based. There is one HTTP(s) port for all M2M API protocols in the NETIO device. Details are shown later in this documentation.
- Username and password to access the file is hidden in the HTML header.
  There can be different username & password for the read and write access.

## Supported devices

- NETIO 4All
- NETIO 4 (Energy metering not supported)
- NETIO 4C (Energy metering not supported)

Note: NETIO 4x means all NETIO 4 devices (NETIO 4 / 4All / 4C)

## Supported devices and firmware

NETIO 4x firmware – 3.0.1 and later

*NOTE: This document provides basic info about the M2M API protocol.*
*Other device functions are described in the product manual.*

## Quick start with json & NETIO

- **READ function - status**

  Read a json file from your NETIO by HTTP(s) **GET**: http(s)://<netioIP>/netio.json
  Example: http://192.168.1.1/netio.json

- **WRITE function - control**

  Upload the following json file by HTTP(s) **POST** to: http(s)://<netioIP>/netio.json
  Example: http://192.168.1.1/netio.json

**json file (command to switch Power output 1 to ON):**

```
{
   "Outputs":[
      {
         "ID":1,
         "Action":1
      }
   ]
}
```

*If the json file & command is accepted, then NETIO returns Status Code “200 OK” and status json file.*

## General NETIO 4x output functions

### Output status – “read” function

- **0** – Power **OFF**
- **1** – Power **ON**

### Output actions – “write” function

- **0 –** Turn **OFF**
- **1 –** Turn **ON**
- **2 –** Short OFF delay (restart)
- **3 –** Short ON delay
- **4 –** Toggle (invert the state)
- **5 –** No change
- **6 –** ignored (the State value will be used to control output)

## Short ON / OFF delay

This command switches a power output On / Off for a defined time. It is useful for example to power-cycle a server with a defined switch-off time, or to switch on a pump for a defined time.

This “short” delay is protected: the power output will remain in the defined state regardless of any other M2M requests received. During this time, the output state can only be changed by pressing the button on the NETIO device and this action cancel M2M short ON/OFF command for the particular output. Other requests to control the particular output are simply ignored and an ERROR logged with reason rejected in a device Log.

The short ON / OFF delay interval can be defined in the device web administration. It is specified in ms (milliseconds) and rounded up to hundreds of miliseconds (0,1s).
This interval can be also defined using some M2M API protocol commands. In that case, it is valid only for a single protocol session (the following short ON / Short OFF command). When the connection is closed or restarted, the interval is reset to the device default value (defined in the web administration for each output).

## Security issues

Do not use default usernames and passwords! Keep your Ethernet and WiFi networks secured.

## Power-Up outputs state

All outputs are Off during the first 25 to 30 seconds after power-up.
After this time, all outputs are set to the selected state:

- **Last Output state**
  After a power outage, the NETIO device sets each power output to the last stored state of this one output. The current state of each power output (socket/power outlet) is internally stored every 8 seconds.

Note: **Function Scheduler** is checked in Power-Up initialization. When enabled, it can affect one or more power output stated based on current time and date.

Custom based **Lua scripts** can affect output stated too.

## HTTP(s) port

There are 2 different HTTP(s) ports:

1) The web administration of the device - HTTP(s).
   Web administration is in the Settings/System (HTTP) or System/Security Settings (HTTPs).

2) Separated HTTP(s) port for the M2M API protocols (XML, JSON, CGI).
   Web administration is in each M2M API protocol settings.

All HTTP(s) protocols (XML / JSON / CGI) share one HTTP(s) port.

| Web Configuration | M2M API | |
|---|---|---|
| HTTP | HTTP | Possible |
| HTTPs | HTTPs | Possible |
| HTTP | HTTPs | Not possible today |
| HTTPs | HTTP | Not possible today |

## Energy metering variables

Since NETIO fw 3.0.0 and later, there are 23 variables available for NETIO energy metering.

Parameters for each power output:

| Variable | Unit | Description |
|---|---|---|
| 4x Current | mA | Instantaneous current for the specific power output |
| 4x PowerFactor | - | Instantaneous Power Factor for the specific power output |
| 4x Load | W | Instantaneous load for the specific power output |
| 4x Energy | Wh | Instantaneous Energy counter value for the specific power output |

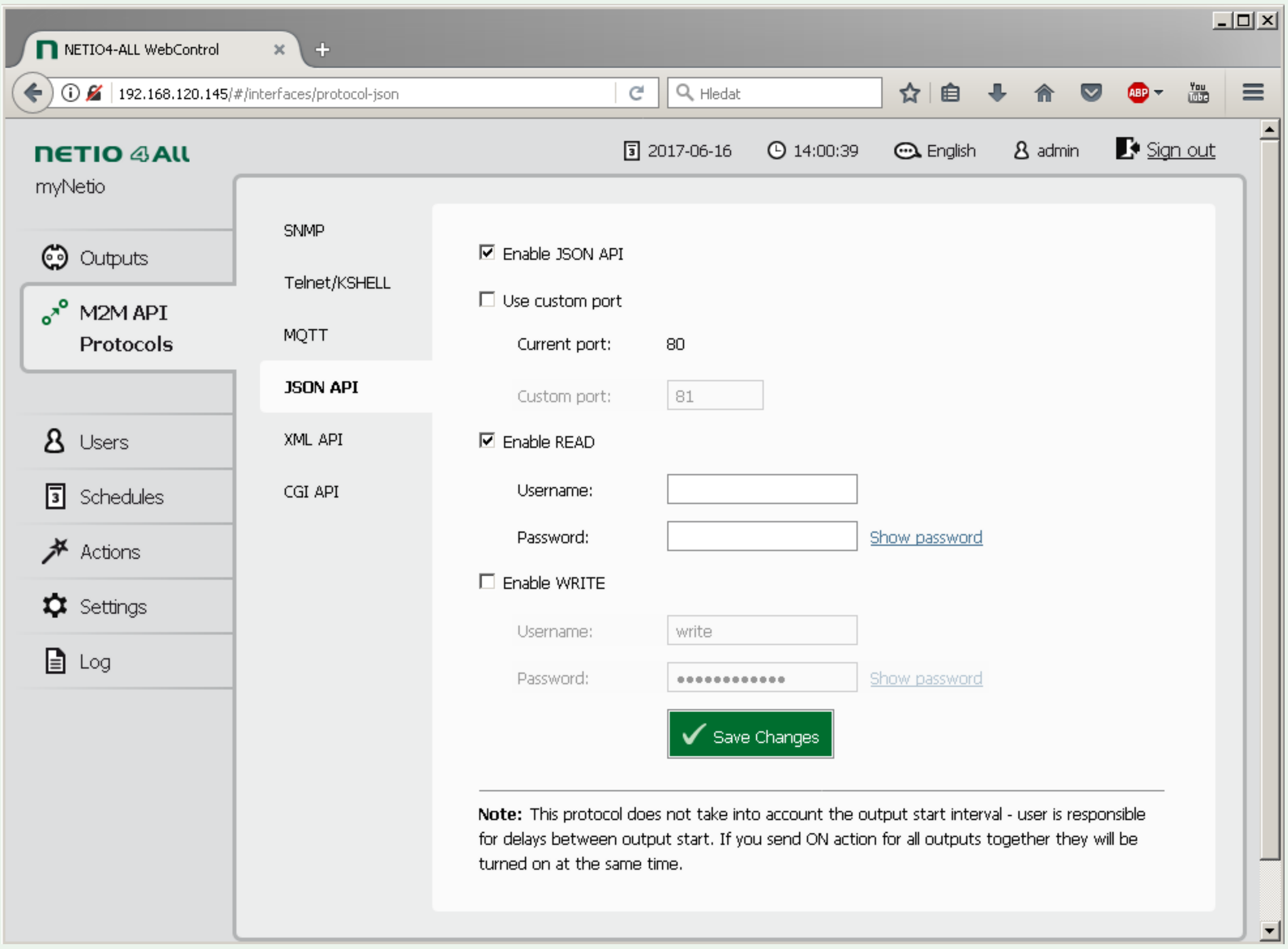
Parameters for the whole NETIO device:

| Variable | Unit | Description |
|---|---|---|
| 1x Voltage | V | Instantaneous voltage |
| 1x Frequency | Hz | Instantaneous frequency |
| 1x TotalCurrent | mA | Instantaneous total current through all power outputs |
| 1x OverallPowerFactor | - | Instantaneous Power Factor – weighted average from all meters |
| 1x TotalLoad | W | Total Load of all power outputs<br>(device's own internal consumption is not included) |
| 1x TotalEnergy | Wh | Instantaneous value of the Total Energy counter |
| 1x EnergyStart | - | Date and time of the last reset of all energy counters |

***Note:*** *Energy metering is not available for all NETIO 4x models.*

## NETIO WEB configuration

M2M API protocols can be enabled and configured only over the web administration – select “M2M API Protocols” in the left-hand side menu and then select the “JSON API” tab.



☑ Enable JSON API
☐ Use custom port
Current port: 80
Custom port: 81
☑ Enable READ
Username:
Password:
☐ Enable WRITE
Username: write
Password: ●●●●●●●●●●●●

**Note:** This protocol does not take into account the output start interval - user is responsible for delays between output start. If you send ON action for all outputs together they will be turned on at the same time.

*Picture 1 –M2M API Protocols / JSON API settings GUI*

- **Enable JSON API** – Enable/disable the M2M API protocol
- **Use custom port** – Check to enable custom port setting
  - **Current port** – Currently used port
  - **Custom port** – Custom port set for XML / JSON / CGI API protocols
- **Enable READ** – Enable READ functionality
  - Username / Password for READ
- **Enable WRITE** – Enable WRITE functionality
  - Username – Username for WRITE
  - Password – Password for WRITE (default: MAC address without colons, lowercase)

## Notes

- **The device webserver is restarted after Saving of JSON API settings.**
- Empty Username and Password means no authentication.
- Credentials are sent in the HTTP header, “Basic authentication” is used.
  The username and password can be also provided in the URL -
  http(s)://username:password@<netioIP>/netio.json

## NETIO JSON protocol structure

JSON standard: RFC4627
JSON Template: 3 Space Tab

### JSON API – READ (status)

**HTTP(s) GET request or HTTP(s) POST request (no file or empty file)**

**GET Request:** **http(s)://<netioIP>/netio.json**

**READ response (status json file):**

```
{
   "Agent": {
      "Model": "NETIO 4All",
      "Version": "3.0.1",
      "JSONVer": "2.0",
      "DeviceName": "myNetio_10",
      "VendorID": 0,
      "OemID": 0,
      "MAC": "24:A4:2C:33:25:E1",
      "Uptime": 110637,
      "Time": "2017-11-03T13:53:38+00:00",
      "NumOutputs": 4
   },
   "GlobalMeasure": {
      "Voltage": 235.8,
      "Frequency": 49.9,
      "TotalCurrent": 20,
      "OverallPowerFactor": 0.22,
      "TotalLoad": 1,
      "TotalEnergy": 965,
      "EnergyStart": "2017-06-23T16:47:53+01:00"
   },
   "Outputs":[
      {
         "ID": 1,
         "Name": "output_1",
         "State": 0,
         "Action": 6,
         "Delay": 5000,
         "Current": 0,
         "PowerFactor": 0,
         "Load": 0,
         "Energy": 192
      },
      {
         "ID": 2,
         "Name": "output_2",
         "State": 0,
         "Delay": 5000,
         "Current": 0,
         "PowerFactor": 0,
         "Load": 0,
         "Energy": 80
      },
      {
         "ID": 3,
         "Name": "output_3",
         "State": 0,
         "Action": 6,
         "Delay": 5000,
         "Current": 0,
         "PowerFactor": 0,
         "Load": 0,
         "Energy": 196
      },
      {
         "ID": 4,
         "Name": "output_4",
         "State": 1,
         "Action": 6,
         "Delay": 5000,
         "Current": 20,
         "PowerFactor": 0.22,
         "Load": 1,
         "Energy": 495
      }
   ]
}
```

***Notes***:
*1. Items/values related to metering (Voltage, Frequency, Current, PowerFactor, Load and Energy, etc.) are available only for the NETIO 4All model.*

*2. Returned status json file contains always "Action" with value "6" for all outputs. This value means "ignore" and works as a placeholder.*

## JSON API – WRITE (control)

**HTTP(s) POST request**

ID - number of output

Outputs can be controlled by two options:

**1. Action**: 0 – off, 1 – on, 2 – short off, 3 – short on, 4 – toggle, 5 – no change, (6 – ignore)

**2. State**: 0 – off, 1 – on   (Action = 6 required)

*Note:* *Action with value other than 6 has priority over the State tag (State value is not reflected in this case). If you wish to use State to control an output then Action must have value 6.*

A json file can be submitted as complete structure (e.g. previously received status json with modified control functions) or partial structure as shown below.

*If the json & command is accepted, then NETIO returns Status Code "200 OK" and status json file.*

**Send command:** **http(s)://<netioIP>/netio.json**

**Switch Power output 1 to ON by Action tag:**

```
{
   "Outputs":[
      {
         "ID":1,
         "Action":1
      }
   ]
}
```

**or (State tag value will not be reflected)**

```
{
   "Outputs":[
      {
         "ID":1,
         "State":0,
         "Action":1
      }
   ]
}
```

**or by State tag (Action tag must have value 6)**

```
{
   "Outputs":[
      {
         "ID":1,
         "State":1,
         "Action":6
      }
   ]
}
```

**Switch Power output 2 to ON for 15 seconds, then switch it OFF.**

```
{
   "Outputs":[
      {
         "ID":2,
         "Action":3,
         "Delay":15000
      }
   ]
}
```

**Command to control more outlets:**
Switch Power output 1 to ON, Toggle Output 2 and Switch Output 4 to ON for 15 seconds:

```
{
   "Outputs":[
      {
         "ID":1,
         "Action":1
      },
      {
         "ID":2,
         "Action":4
      },
      {
         "ID":4,
         "Action":3,
         "Delay":15000
      }
   ]
}
```

## Status codes

| Status codes | Description |
|---|---|
| 200 OK | User authorized and command received |
| 400 Bad Request | Control command syntax error |
| 401 Unauthorized | Invalid Username or Password |
| 403 Forbidden | Read only |
| 500 Internal Server Error | Internal Server Error or Internal Server not fully started yet (e.g. after setting change or restart) |

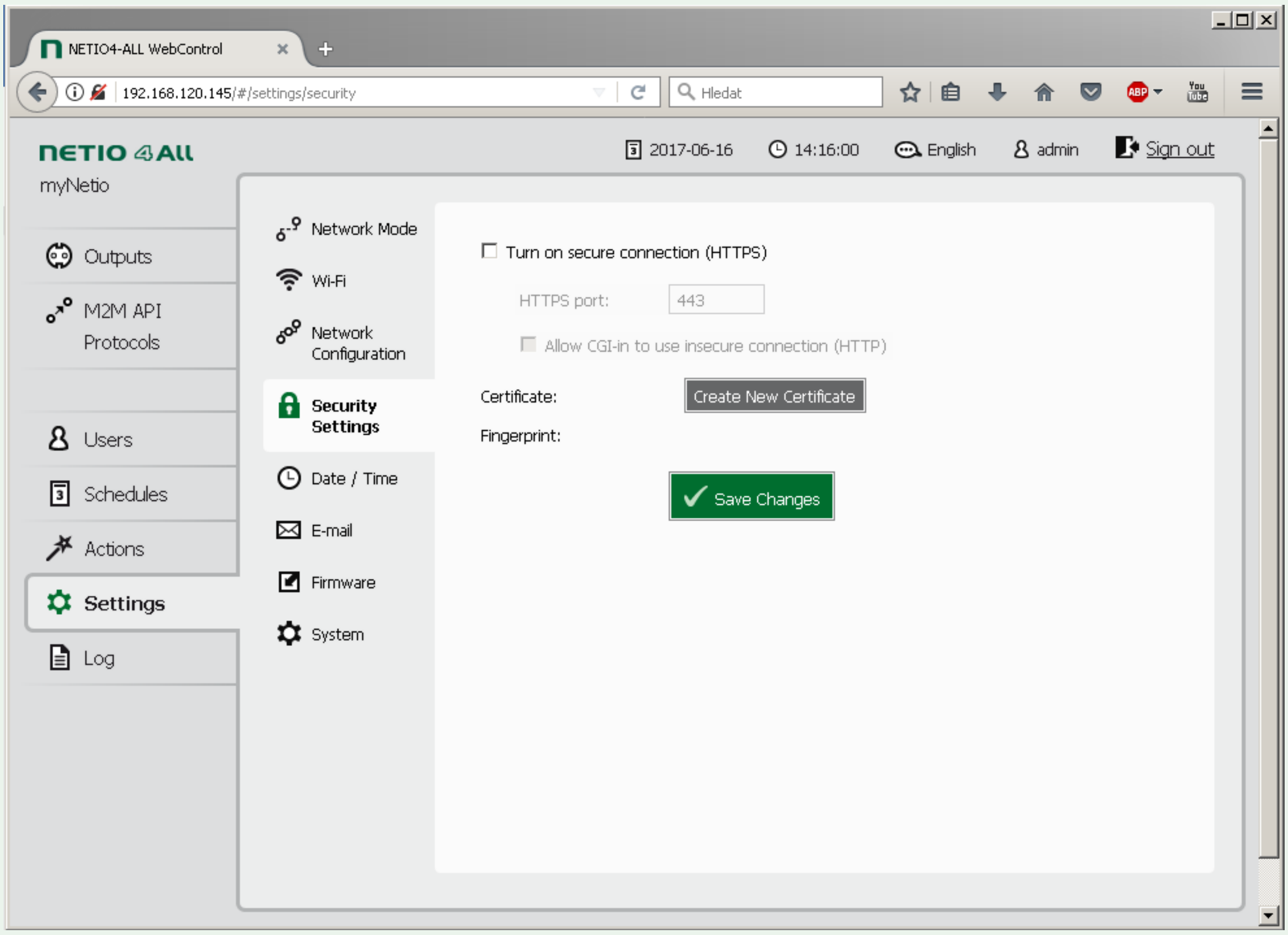
**Response syntax for “OK” state:**
Status json file as described above in chapter “NETIO JSON protocol structure” / READ

**Response syntax for “Error” state:**

```
{
   "result": {
      "error": {
         "code": 200,
         "message": "OK"
      }
   }
}
```

## HTTPs = secure connection

NETIO can use secure connection (HTTPs) for web administration and HTTP-based M2M API Protocols (JSON, XML, CGI). This security feature can be enabled in the web administration: Settings / Security Settings. After checking “Turn on secure connection (HTTPS)” and saving the changes, only secured (HTTPS) communication will be available for web and HTTP-based protocols. The port for HTTP-based M2M API protocols remains the same as already set.


NETIO4-ALL WebControl
192.168.120.145/#/settings/security

NETIO 4ALL myNetio | 2017-06-16 | 14:16:00 | English | admin | Sign out

Outputs, M2M API Protocols, Users, Schedules, Actions, Settings, Log

Network Mode, Wi-Fi, Network Configuration, Security Settings, Date / Time, E-mail, Firmware, System

☐ Turn on secure connection (HTTPS)

HTTPS port: 443

☐ Allow CGI-in to use insecure connection (HTTP)

Certificate: Create New Certificate

Fingerprint:

Save Changes

*Picture 2 –Security settings GUI*

- **Turn on secure connection (HTTPS)** – Enable/disable secure connection for web administration and HTTP-based M2M API protocols (JSON, XML, CGI)
- **HTTPS port** – Port used for the web administration (M2M API protocols use the port set in M2M Protocol settings)
- **Allow CGI-in to use insecure connection (HTTP)** – Enable unsecure connection for CGI-in
- **Create New Certificate** – Generates and immediately installs a new certificate.
- **Certificate** – Info about certificate validity (certificate is generated with one year validity)
- **Fingerprint** – Certificate fingerprint

## Examples

Examples are available in following Application Note (AN) document(s)

- AN21 – JSON HTTP(s) protocol to control NETIO smart power sockets

## NETIO 4 – listing of the json file

*Note: In the NETIO 4 model, there are no metering values available.*

**NETIO 4 – status json file**

```
{"Agent":{"Model":"NETIO
4","Version":"3.0.1","JSONVer":"2.0","DeviceName":"myNetio","VendorID":0,"
OemID":0,"SerialNumber":"","Uptime":739,"Time":"2017-11-
09T13:18:56+01:00","NumOutputs":4},"Outputs":[{"ID":1,"Name":"output_1","S
tate":1,"Action":6,"Delay":5000},{"ID":2,"Name":"output_2","State":1,"Acti
on":6,"Delay":5000},{"ID":3,"Name":"output_3","State":1,"Action":6,"Delay"
:5000},{"ID":4,"Name":"output_4","State":1,"Action":6,"Delay":5000}]}
```

## Notes

- The “Uptime” value is in seconds [s]

## Keywords

NETIO M2M API Protocol JSON

NETIO products

JSON M2M API Protocol documentation

json, NETIO, M2M API, HTTP, HTTPS, outlet, socket, output, control, power sockets, power outlets, json file

## Document history

| Document Revision | Publication Date | Description |
|---|---|---|
| 1.0 | 14.11.2017 | Initial release - JSON Version 2.0, for FW 3.0.1 |
| 1.1 | 7.12.2017 | Documentation optimization - Action 6 |
| 1.2 | 19.12.2017 | Keywords added |