JSON OVER HTTP NETIO M2M API PROTOCOLS

Protocol version: JSON Version 2.6
Document published: 24.02.2026

Short summary

JSON over 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 read out NETIO power outputs.

- All NETIO devices support JSON via HTTP & HTTPs connection
- There is a HTTP connection enable as default with. **HTTPs** is a secure option available from FW 5.0.3. BETA
- The JSON API protocol must be enabled first in the WEB configuration of the respective device. For details, see the “NETIO WEB configuration” chapter.
- 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. Default configuration is read/write for the user=netio password=netio.
- Do not use default usernames and passwords! Keep your networks secured
- By writing (http://netioDeviceIP/netio.json) as URL to browser, the device provide you the latest JSON content in the same structure as the netio.json file.
- Several functionalities i.e. - Temperature, Humidity ...etc. Are available only with a latest FW.

Table of Contents

Supported devices

Standard NETIO devices: All NETIO devices support JSON API.

- NETIO PowerCable REST 101x (Energy metering)
- NETIO PowerCable 2Px
- NETIO PowerCable 2Kx (Energy metering)
- NETIO PowerBOX 3Px
- NETIO PowerBOX 4Kx (Energy metering)
- NETIO PowerDIN 4KZ (Energy metering)
- NETIO PowerDIN ZK3 (Energy metering)
- NETIO PowerDIN ZP3
- NETIO PowerPDU 4Px
- NETIO PowerPDU 4Kx (Energy metering)
- NETIO PowerPDU 8Kx (Energy metering)
- NETIO PowerPDU 8Qx (Energy metering)

*Note: it is a common API specification document through complete portfolio of NETIO devices. In the documentation you could find devices with name as **"PowerPDU 8Kx"**. Here **"X"** represents type of electrical output socket - models **xF** (Type F - schuko), **xS** (IEC-320 C13), **xB** (NEMA 5 -15), **xZ** (terminal block) ...etc. There is not influence on API functionality*

Compatibility Note:

****JSON available from FW 4x 3.0.3*

****"JSONVer": 2.4 support from FW 3.1.3+*

****"JSONVer": 2.6 support from FW 5.0.3+*

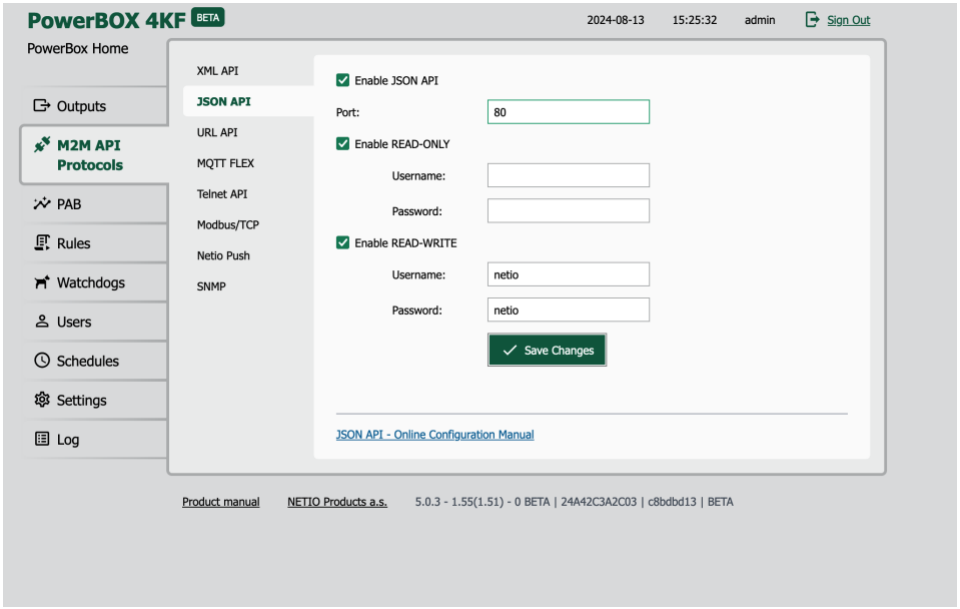
NETIO 4x Linux based devices: networked power sockets with LAN / WiFi connectivity.

- NETIO PowerPDU 4C (End Of Life Dec/2024)
- NETIO 4, 4All (obsolete products)

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

JSON API – 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 - *Enable/disable the M2M API protocol*
- Port - *Custom port set JSON API protocols*
- Enable READ-ONLY - *Enable READ functionality*
- Username/ Password - *Username / Password for READ*
- Enable WRITE - *Enable WRITE functionality*
- Username/ Password - *Username / Password for WRITE*

Notes:

- The device webserver is restarted after saving the 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@< http://192.168.0.7 >/netio.json

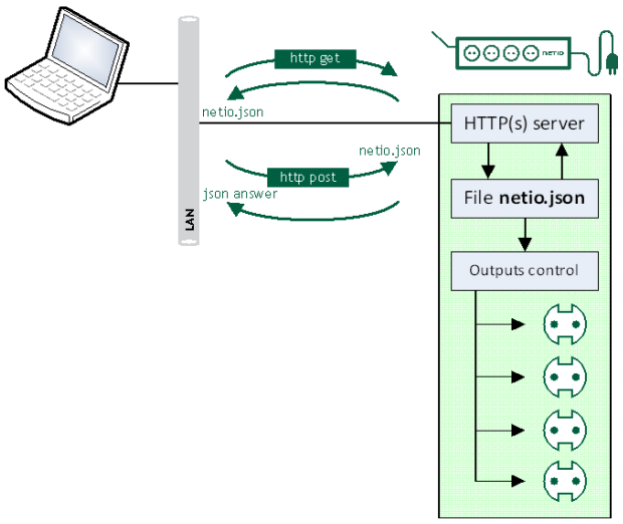
Quick start with JSON & NETIO

- **READ function - status**
  to read out a netio.json file from your NETIO device by HTTP(s) GET:

```
Example:
http(s)://<netioIP>/netio.json
http://192.168.1.1/netio.json
```

- **WRITE function - control**
  Upload the following netio. json file by HTTP(s) POST to NETIO device

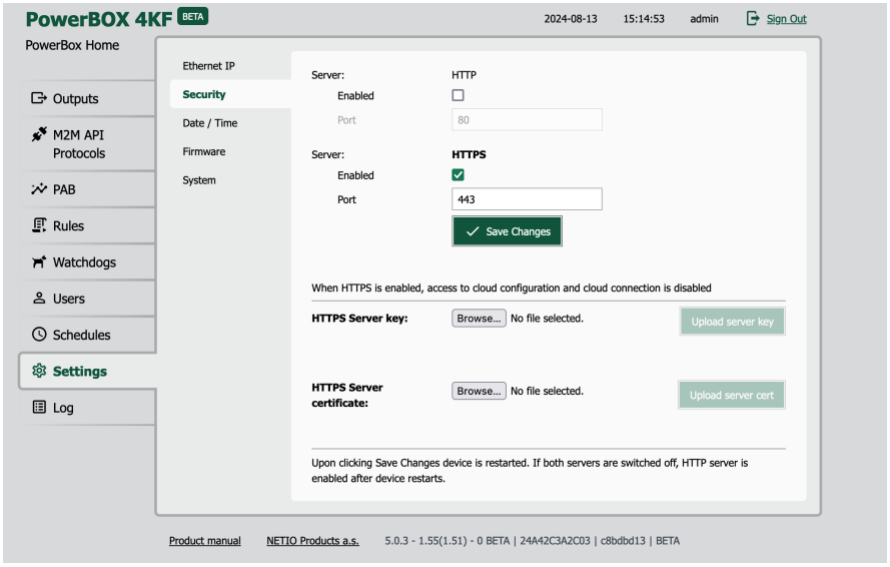
```
Example:
http(s)://<netioIP>/netio.json
http://192.168.1.1/netio.json


Example: netio.json file – command to switch OUTPUT1 ON
{
   "Outputs":[
  {
 "ID":1,
 "Action":1
  }
   ]
}
```



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

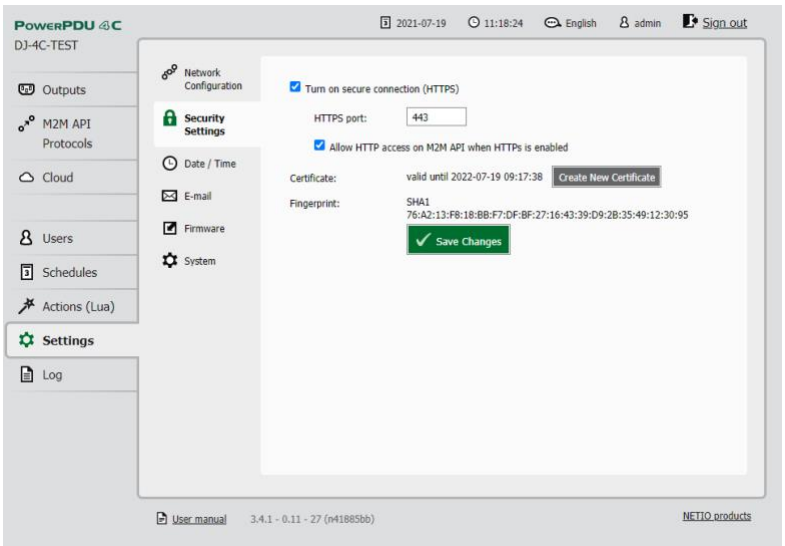
HTTP(s) options

Standard NETIO devices : All NETIO devices support JSON API via HTTPs with FW 5.0.3+ with custom certificates option. More details about HTTPs/



NETIO 4x Linux based devices (NETIO 4C) 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).
   Separated HTTP(s) port for the M2M API protocols (XML, JSON, URL API).
   Web administration is in each M2M API protocol settings.



- All HTTP(s) protocols (XML / JSON / URL-API) share one HTTP(s) port.
- For PowerPDU 4C have to be both ports (for web + URL API) **http** or **https**.
- Custom HTTPS certificate is not supported.

General NETIO output functions description

Output status – “read out” functions

```
"State": 0 – Power Output is OFF
"State": 1 – Power Output is ON
```

Output actions – “write” functions

```
"Action": 0 – Turn OFF Power Output
"Action": 1 – Turn ON Power Output
"Action": 2 – Short OFF delay (restart) Power Output
"Action": 3 – Short ON delay (restart) Power Output
"Action": 4 – Toggle (invert) Power Output State
"Action": 5 – No Change
"Action": 6 – Ignored (return value from reading the tag)
          Actual output value is in ”State” tag (0 / 1).
```

Additional details description for Output actions

https://www.netio-products.com/en/glossary/short-off-action-2
https://www.netio-products.com/en/glossary/short-on-action-3
https://www.netio-products.com/en/glossary/toggle-action-4

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.

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).

NETIO PowerPDU 4C and NETIO 4/4All: The “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.

Power-Up outputs state
All outputs are OFF during the first 5 to 30 seconds after power-up (depending on device model). After this time, all outputs are set to the selected state based on its individual settings:

- **LAST state**
  After a power outage, the NETIO device sets each power output to the last stored state of this one output.
- **ON**
  The output is turned ON.
- **OFF**
  The output stays OFF.

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

NETIO PowerPDU 4C and NETIO 4/4All - Custom based **Lua scripts** can affect output stated too.

Input properties

Input status

```
"State": 0 – "open"/ ON
"State": 1 – "closed"/ OFF
```

Temperature

*** temperature metering function available from FW 5.0.3+; JSON 2.6*

```
"TemperatureC":999 – temperature sensor not connected
"TemperatureF":999 – temperature sensor not connected
"TemperatureC":25 – temperature sensor connected, value 25C
"TemperatureF":78 – temperature sensor connected, value 78F
```

Humidity

*** Humidity metering function available from FW 5.2.0+; JSON 2.6*

```
"Humidity":999 – sensor not connected
"Humidity":25 – sensor connected, value 25[%]
```

S0 counters

Number of S0 impulses / “ON” pulses

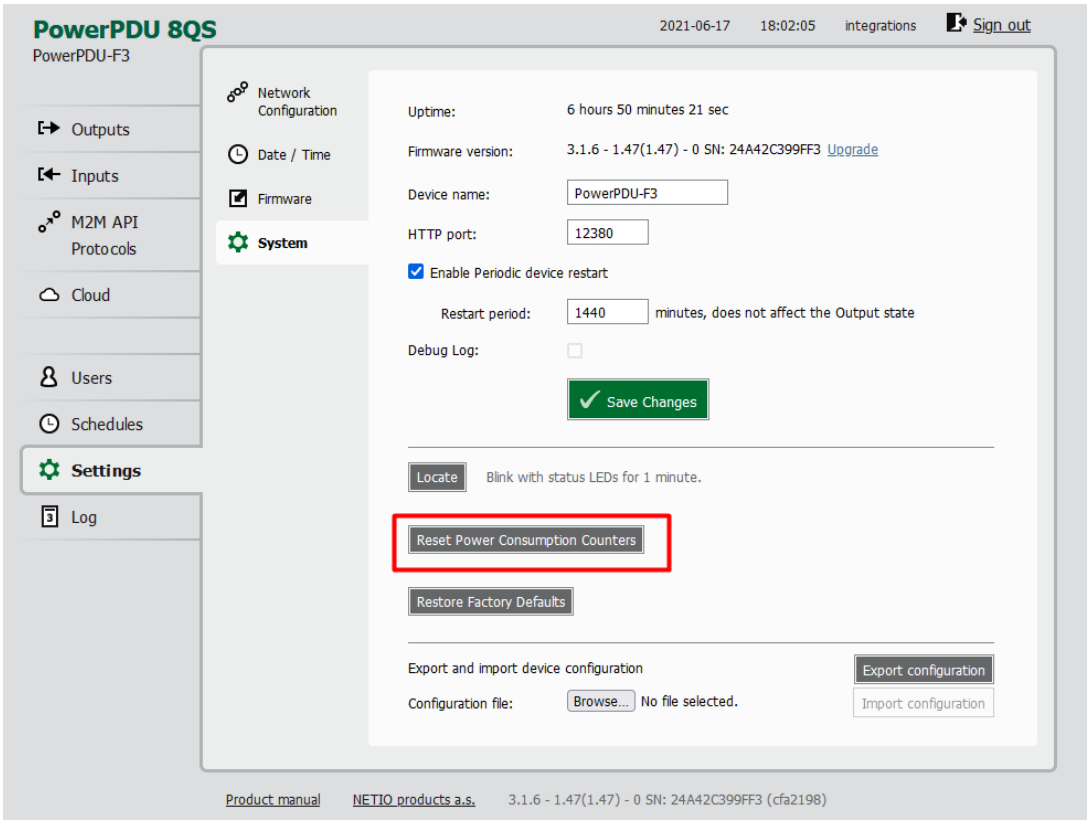
```
"S0Counter":43 – number of counted pulses
```

General NETIO Counter

- Power consumption counters can be reset to 0 manually
- Power consumption NR counters are Not Resettable
- S0 pulse counters on DI (Digital Inputs) can be reset to 0 manually
- All counters are not affected by power outage.

How to reset counter to 0

Click to button in the device settings. It will reset the counters.



Addition NETIO read out properties

PAB – Power Analyses Block

PAB{"Type":"RANGE","Name":"PAB_LOADS_P3","Enabled":true,"In":false},

```
Type: RANGE            - PAB type
Name:                  - PAB Name
Enable: true/false     - PAB enable/ disable
In: true/false         - Inspection result
```

Watchdog – IP watchdog

"Name":"google_pinger","Enabled":true,"Fail":false,"LastStatus":true,"Timestamp":1721043267},
{"Name":"CR01_WDT","Enabled":false,"Fail":false,"LastStatus":false,"Timestamp":0}

```
Name:                  - Watchdog Name
Enable:                - Watchdog enable/ disable
Fail:                  - return value true/false  – in case
LastStatus:            - latest result status true/false
Timestamp:             - Linux timestamp of latest result status
```

Rules

{"Name":" CR16_RULE","Enabled":false,"Result":null},
{"Name":"CR_TEMP","Enabled":true,"Result":true}

```
Name:                  - Rule name
Enable: false/true     - Rule enable/ disable
Result: null/true      - Rule result
```

Addition NETIO write commands

Output name change

```
"Outputs":[{"ID":1,"Name":"New name","Action":5}]
```

Reset energy counter for Output 1

```
"Outputs":[{"ID":1,"resetEnergyCounter":true}]
```

Schedule enable/ disable

```
"Action": 21 – Activate assigned Output Scheduler
"Action": 20 – Deactivate assigned Output Scheduler
```

Energy metering variables

Energy metering is available for:

- NETIO PowerPDU 4C
- PowerCable REST
- PowerCable 2Kx
- PowerBox 4Kx
- PowerDIN 4PZ
- PowerPDU 8Qx
- PowerPDU 8Kx

Parameters for the **whole NETIO device**:

Variable	Unit	Description
Voltage	V	Instantaneous voltage
Frequency	Hz	Instantaneous frequency
Total Current	mA	Instantaneous total current through all power outputs

Total Power Factor	-	Instantaneous True Power Factor – weighted average from all meters
Total Phase	°	Instantaneous True Phase – weighted average from all meters
Total Load	W	Total load (power) of all power outputs (device's own internal consumption is not included)
Total Energy	Wh	Total Counter of consumed Energy (resettable)
Total Energy NR	Wh	Not Resettable counter of total consumed Energy
Total Reverse Energy	Wh	Counter of Reversed (produced) Energy (resettable)
Total Reverse Energy NR	Wh	Not Resettable counter of total Reversed Energy
Energy Start	-	Date and time of the last reset of all energy counters
Overall True Power Factor		*Historical compatibility only, do not use it.*
Overall Phase		*Historical compatibility only, do not use it.*

Parameters for **each power output**:

Variable	Unit	Description
Current	mA	Electric current for the output
Power Factor	-	TPF True Power Factor for the output
Phase	°	Phase for the specific power output
Energy	Wh	Counter of Energy consumed per output (resettable)
Energy NR	Wh	Not Resettable counter of output consumed Energy
Reverse Energy	Wh	Counter of Energy produced per output (resettable)
Reverse Energy NR	Wh	Not Resettable counter of Reversed (produced) Energy
Load	W	Instantaneous load (power) for the specific power output.

NETIO JSON protocol structure

- JSON standard: RFC4627
- JSON Template: 3 Space Tab


```
{
   "Agent": {
"Model": "4PZ",
"DeviceName": "PowerDIN-09-00",
"MAC": "24:A4:2C:3A:5E:55",
"SerialNumber": "24A42C3A5E55",
"JSONVer": "2.6",
"Time": "2024-09-14T20:27:12+0100",
"Uptime": 28104,
"Version": "5.0.3",
"OemID": 400,
"VendorID": 0,
"NumOutputs": 4,
"NumInputs": 2
   },
   "GlobalMeasure": {
"Voltage": 243.07,
"TotalCurrent": 0,
"OverallPowerFactor": 999,
"TotalPowerFactor": 999,
"OverallPhase": 999,
"TotalPhase": 999,
"Frequency": 50.07,
"TotalEnergy": 0,
"TotalReverseEnergy": 0,
"TotalEnergyNR": 202,
"TotalReverseEnergyNR": 0,
"TotalLoad": 0,
"EnergyStart": "2024-08-30T07:23:28+0100"
   },
   "Outputs": [
{
   "ID": 1,
   "Name": "CCC",
   "State": 1,
   "Action": 6,
   "Delay": 5000,
   "Current": 0,
   "PowerFactor": 1,
   "Phase": 0,
   "Energy": 9,
   "ReverseEnergy": 0,
   "EnergyNR": 26,
   "ReverseEnergyNR": 0,
   "Load": 0,
   "EnergyStart": "2024-08-16T16:29:35+0100"
},
{
   "ID": 2,
   "Name": "TEST",
   "State": 1,
   "Action": 6,
   "Delay": 5000,
   "Current": 0,
   "PowerFactor": 1,
   "Phase": 0,
   "Energy": 167,
```

```
  "ReverseEnergy": 0,
  "EnergyNR": 176,
  "ReverseEnergyNR": 0,
  "Load": 0,
  "EnergyStart": "2024-08-16T16:30:08+0100"
},
{
  "ID": 3,
  "Name": "DI1 Trigger",
  "State": 1,
  "Action": 6,
  "Delay": 5000,
  "EnergyStart": "1970-01-01T00:00:00+0100"
},
{
  "ID": 4,
  "Name": "Free Contact 4",
  "State": 1,
  "Action": 6,
  "Delay": 5000,
  "EnergyStart": "1970-01-01T00:00:00+0100"
}
  ],
  "Inputs": [
{
  "ID": 1,
  "Name": "Input 1",
  "State": 1,
  "S0Counter": 2020,
  "TempertureC": 999,
  "TemperatureF": 999,
  "Humidity”: 9990
},
{
  "ID": 2,
  "Name": "Temp_sensor",
  "State": 0,
  "S0Counter": 0,
  "TempertureC": 22.2,
  "TemperatureF": 71.9,
  "Humidity": 30

}
  ],
  "PAB": [
{
  "Type": "RANGE",
  "Name": "CR03_PAB",
  "Enabled": false,
  "In": false
}
  ],
  "Watchdogs": [
{
  "Name": "google_pinger",
  "Enabled": false,
  "Fail": false,
  "LastStatus": false,
  "Timestamp": 0
},
{
  "Name": "CR01_WDT",
```

```
  "Enabled": true,
  "Fail": false,
  "LastStatus": true,
  "Timestamp": 1726345631
}
  ],
  "Rules": [
{
  "Name": "CR01_RULE",
  "Enabled": false,
  "Result": null
},
{
  "Name": "CR02_RULE",
  "Enabled": false,
  "Result": null
},
{
  "Name": "CR03_RULE",
  "Enabled": false,
  "Result": null
},
{
  "Name": "CR13_RULE",
  "Enabled": false,
  "Result": null
}
  ]
}
```


***Notes**:*

1) *Old firmware versions contains the MAC tag only. When both tags available, use the SerialNumber, it's identical with printed label on the device.*

2) *Items/values related to metering (Voltage, Frequency, Current, PowerFactor, Load and Energy, etc.) are available only for the NETIO PowerPDU 4C, PowerCable, PowerDIN 4PZ and NETIO 4All models.*

3) *Returned status **netio.json** file always contains "Action" with value "6" for all outputs. This value means "ignore" and works as a placeholder. Output state 0 / 1 is in the **State** value.*

Values description

Global values:

"Model": "4PZ",	*Model identification*
"DeviceName": "PowerDIN-09-00",	*Device name (user defined on web)*
"MAC": "24:A4:2C:3A:5E:55",	*MAC address of active interface.* *For LAN/Wifi devices only.*
"SerialNumber": "24A42C3A5E55",	*Serial Number of device – preferred identifier* *(identical with label on delivery box).*
"JSONVer": "2.6",	*Protocol version*
"Time": "2024-09-14T20:27:12+0100",	*Date and time of the NETIO device*
"Uptime": 28104,	***[s]** The Uptime value. Goes back to zero after device reset or energy lost*
"Version": "5.0.3",	*Firmware version*
"OemID": 400,	*Manufacturer internal use*
"VendorID": 0,	*Manufacturer internal use*
"NumOutputs": 4,	*Number of outputs*
"NumInputs": 2	*Number of inputs*
"Voltage": 243.07,	***[V] I**nstantaneous voltage*
"TotalCurrent": 0,	***[mA]** Instantaneous total current through all power outputs*
"OverallPowerFactor": 999,	***[°]** Instantaneous PowerFactor weighted average from all meters*
"TotalPowerFactor": 999,	***[°]** Instantaneous PowerFactor weighted average from all meters*
"OverallPhase": 999,	***[°]** Instantaneous Phase weighted average from all meters*
"TotalPhase": 999,	***[°]** Instantaneous Phase weighted average from all meters*
"Frequency": 50.07,	***[Hz]** Instantaneous frequency power outputs*
"TotalEnergy": 0,	***[Wh]** Counter of total consumed Energy (4B)*
"TotalReverseEnergy": 0,	***[Wh]** Counter of total produced Energy (4B)*
"TotalEnergyNR": 202,	***[Wh]** Not resettable counter of total consumed Energy (4B)*
"TotalReverseEnergyNR": 0,	***[Wh]** Not resettable counter of total produced Energy (4B)*
"TotalLoad": 0,	***[W]** Total Power of all power outputs*
EnergyStart": "2024-08-30T07:23:28+0100"	*Date and time of the last reset of all energy counter*

Values for specific output (example values below are for output 1):

`"ID": 1,`	*Output number*
`"Name": "CCC",`	*Output name (user defined on web)*
`"State": 1,`	*Output state*
`"Action": 6,`	*Output action (6 = Ignored value, use State tag)*
`"Delay": 5000,`	***[ms]** Output delay for short On/Off*
`"Current": 0,`	***[mA]** Instantaneous current of the output*
`"PowerFactor": 1,`	***[-]** Instantaneous True Power Factor*
`"Phase": 0,`	***[°]** Instantaneous Phase of the output*
`"Energy": 9,`	***[Wh]** Counter of output consumed Energy (4B)*
`"ReverseEnergy": 0,`	***[Wh]** Counter of output produced Energy (4B)*
`"EnergyNR": 26,`	***[Wh]** Not resettable counter of consumed Energy (4B)*
`"ReverseEnergyNR": 0,`	***[Wh]** Not resettable counter of output produced Energy (4B)*
`"Load": 0,`	***[W]** Total Power of the output*
`"EnergyStart": "2024-08-16T16:29:35+0100"`	*Date and time of the last reset of energy counter*

Values for specific input (example values below are for input 2):

`"ID": 2,`	*Input number*
`"Name": "Temp_sensor",`	*Input name*
`"State": 0,`	*Input state (0 = OFF/"open", 1= ON/"closed")*
`"S0Counter": 0,`	*S0 Counter value (4B)*
`"TempertureC": 22.2,`	*Temperature in Celsius*
`"TemperatureF": 71.9`	*Temperature in Fahrenheit*
`"Humidity": 30`	*Humidity 30 %*

JSON API – WRITE (control)

HTTP(s) POST request

ID - number of output

Outputs can be controlled by two options:

1. Action:

```
"Action": 0 – Turn OFF Power Output
"Action": 1 – Turn ON Power Output
"Action": 2 – Short OFF delay (restart) Power Output
"Action": 3 – Short ON delay (restart) Power Output
"Action": 4 – Toggle (invert) Power Output State
"Action": 5 – No Change
"Action": 6 – Ignored (return value from reading the tag)
          Actual output value is in ”State” tag (0 / 1).
```

2. State:

```
"State": 0 – Power Output is OFF
"State": 1 – Power Output is ON
```

Note: ***Action*** *with other value than 6 has higher priority than the* ***State*** *tag.*
State value is not reflected in case Action = 1 to 5.
If you wish to use ***State*** *tag to control an output,* ***Action = 6 is required***.

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 outputs:

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
  }
   ]
}
```

Command to rename Outpu2 & reset Energy counter

```
{
   "Outputs":[
  {
 "ID":2,
 "Name":"AC Fan 2",
 "ResetEnergyCounter":true,
 "Action":5
  }
 ]
}
```

Command to Turn ON scheduler on Output 1 & Turn OFF scheduler on Output2

```
{
   "Outputs":[
  {
 "ID":1,
 "Action":21
  },
  {
 "ID":2,
 "Action":20
  }

   ]
```

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
404 Not found	JSON API protocol not enabled
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:

```
{
"errors": [
{
"code": 401,
"severity": "fatal",
"message": "Unauthorized"
}
]
}
```

NETIO JSON file examples

NETIO PowerCable 101 – listing of the netio.json file


```
"Agent": {
"Model": "101F",
"DeviceName": "PowerCable-REST",
"MAC": "24:A4:2C:38:E7:5C",
"SerialNumber": "24A42C38E75C",
"JSONVer": "2.6",
"Time": "2024-09-14T21:57:38+0100",
"Uptime": 2260181,
"Version": "5.0.3",
 "OemID": 5,
"VendorID": 0,
"NumOutputs": 1,
"NumInputs": 0 },

"GlobalMeasure": {
"Voltage": 243.32,
"TotalCurrent": 0,
"OverallPowerFactor": 1,
"TotalPowerFactor": 1,
"OverallPhase": 0,
"TotalPhase": 0,
"Frequency": 50.07,
"TotalEnergy": 1936,
"TotalReverseEnergy": 0,
"TotalEnergyNR": 1936,
"TotalReverseEnergyNR": 0,
"TotalLoad": 0,
"EnergyStart": "2021-03-02T14:10:24+0100" },

"Outputs": [ { "ID": 1, "Name": "Power output 1", "State": 0, "Action": 6, "Delay":
5000, "Current": 0, "PowerFactor": 1, "Phase": 0, "Energy": 1936, "ReverseEnergy": 0,
"EnergyNR": 1936, "ReverseEnergyNR": 0, "Load": 0, "EnergyStart": "2021-03-
02T14:10:24+0100" }
```

NETIO PowerBOX 3Px – listing of the netio.json file

Note: In the NETIO PowerBOX 3Px model, there are no metering values available.


```
{

"Agent":{"Model":"3PF","DeviceName":"PowerBOX-
F2","MAC":"24:A4:2C:38:DF:F2","SerialNumber":"24A42C38DFF2","JSONVer":"2.3","Time":"197
0-01-
01T23:05:55+01:00","Uptime":36355,"Version":"2.5.4","OemID":0,"VendorID":0,"NumOutputs"
:3},

"Outputs":[

{"ID":1,"Name":"Power output 1","State":0,"Action":6,"Delay":2020},

{"ID":2,"Name":"Power output 2","State":1,"Action":6,"Delay":2020},

{"ID":3,"Name":"Power output 3","State":1,"Action":6,"Delay":2020}

]}
```

NETIO PowerCable 2KZ – listing of the netio.json file


```
"Agent": {
"Model": "2KZ",
"DeviceName": "PowerCable-2KZ",
"MAC": "24:A4:2C:3A:57:4B",
"SerialNumber": "24A42C3A574B",
"JSONVer": "2.6",
"Time": "2024-09-14T21:48:57+0100",
"Uptime": 137603,
"Version": "5.0.3",
"OemID": 900,
"VendorID": 0,
"NumOutputs": 2,
"NumInputs": 2 },

"GlobalMeasure": {
"Voltage": 247.43,
"TotalCurrent": 0,
"OverallPowerFactor": 999,
"TotalPowerFactor": 999,
"OverallPhase": 999,
"TotalPhase": 999,
"Frequency": 50.05,
"TotalEnergy": 40502149,
"TotalReverseEnergy": 0,
"TotalEnergyNR": 40504222,
"TotalReverseEnergyNR": 0,
"TotalLoad": 0,
"EnergyStart": "2023-02-20T19:02:34+0100" },

"Outputs":
[ { "ID": 1, "Name": "Power output 1", "State": 0, "Action": 6, "Delay": 5000,
"Current": 0, "PowerFactor": 1, "Phase": 0, "Energy": 23160477, "ReverseEnergy": 0,
"EnergyNR": 23162083, "ReverseEnergyNR": 0, "Load": 0, "EnergyStart": "2023-02-
20T19:02:34+0100" },

{ "ID": 2, "Name": "Power output 2", "State": 0, "Action": 6, "Delay": 5000, "Current":
0, "PowerFactor": 1, "Phase": 0, "Energy": 17342139, "ReverseEnergy": 0, "EnergyNR":
17342139, "ReverseEnergyNR": 0, "Load": 0, "EnergyStart": "1970-01-01T00:00:00+0100" }
],

"Inputs":
[{"ID":1,"Name":"Input
1","State":1,"S0Counter":20729,"TemperatureC":999.0,"TemperatureF":999.0,"Humidity":999
0},
{"ID":2,"Name":"Input2","State":0,"S0Counter":0,"TemperatureC":999.0,"TemperatureF":999
.0,"Humidity":9990}
```

NETIO PowerDIN 4PZ – listing of the netio.json file

Note: In the NETIO PowerDIN 4PZ model are just 2 from 4 outputs metered.


```
{
"Agent":{"Model":"4PZ","DeviceName":"powerdin-
4pz","MAC":"24:A4:2C:39:67:17","SerialNumber":"24A42C396717","JSONVer":"2.3","Time":"20
21-03-
24T01:15:32+01:00","Uptime":6020,"Version":"3.0.1","OemID":400,"VendorID":0,"NumOutputs
":4,"NumInputs":2},

"GlobalMeasure":{"Voltage":241,"TotalCurrent":0,"TotalLoad":0,"TotalEnergy":0,"OverallP
owerFactor":0.00,"Frequency":50.07,"Phase":0.00,"EnergyStart":"2020-12-
13T19:34:31+01:00"},

"Outputs":[
{"ID":1,"Name":"Power output
1","State":0,"Action":6,"Delay":2020,"Current":0,"PowerFactor":1.00,"Phase":0.00,"Energ
y":0,"ReverseEnergy":0,"Load":0},
{"ID":2,"Name":"Power output
2","State":1,"Action":6,"Delay":2020,"Current":0,"PowerFactor":1.00,"Phase":0.00,"Energ
y":0,"ReverseEnergy":0,"Load":0},
{"ID":3,"Name":"Free Contact 3","State":0,"Action":6,"Delay":2020},
{"ID":4,"Name":"Free Contact 4","State":1,"Action":6,"Delay":2020}
],

"Inputs":[
{"ID":1,"Name":"Intput 1","State":0,"S0Counter":290,
"TemperatureC":999.0,"TemperatureF":999.0,"Humidity":9990},
{"ID":2,"Name":"Intput 2","State":0,"S0Counter":1,
"TemperatureC":999.0,"TemperatureF":999.0,"Humidity":9990}
]}
```

NETIO PowerPDU 8QS – listing of the netio.json file

Note: In the NETIO PowerPDU 8QS model are metered just 2 channels (Output 1 & global) from total 8 outputs.


```
{
"Agent":{"Model":"8QS","DeviceName":"PowerPDU-
F3","MAC":"24:A4:2C:39:9F:F3","SerialNumber":"24A42C399FF3","JSONVer":"2.4","Time":"202
1-06-
08T13:44:49+01:00","Uptime":76239,"Version":"3.1.6","OemID":600,"VendorID":0,"NumOutput
s":8,"NumInputs":1},
"GlobalMeasure":{"Voltage":240.97,"TotalCurrent":0,"OverallPowerFactor":1.00,"TotalPowe
rFactor":1.00,"OverallPhase":0,"TotalPhase":0,"Frequency":50.09,"TotalEnergy":7,"TotalR
everseEnergy":1,"TotalEnergyNR":7,"TotalReverseEnergyNR":1,"TotalLoad":0,"EnergyStart":
"1970-01-01T00:00:00+01:00"},
"Outputs":[
{"ID":1,"Name":"Power output
1","State":0,"Action":6,"Delay":5000,"Current":0,"PowerFactor":1.00,"Phase":0.00,"Energ
y":7,"ReverseEnergy":0,"EnergyNR":7,"ReverseEnergyNR":0,"Load":0},
{"ID":2,"Name":"Power output 2","State":1,"Action":6,"Delay":5000},
{"ID":3,"Name":"Power output 3","State":0,"Action":6,"Delay":5000},
{"ID":4,"Name":"Power output 4","State":0,"Action":6,"Delay":5000},
{"ID":5,"Name":"Power output 5","State":1,"Action":6,"Delay":5000},
{"ID":6,"Name":"Power output 6","State":1,"Action":6,"Delay":5000},
{"ID":7,"Name":"Power output 7","State":1,"Action":6,"Delay":5000},
{"ID":8,"Name":"Power output 8","State":1,"Action":6,"Delay":5000}
],
"Inputs":[
{"ID":1,"Name":"Input 1","State":1,"S0Counter":0,
"TemperatureC":999.0,"TemperatureF":999.0,"Humidity":9990}
]}
```

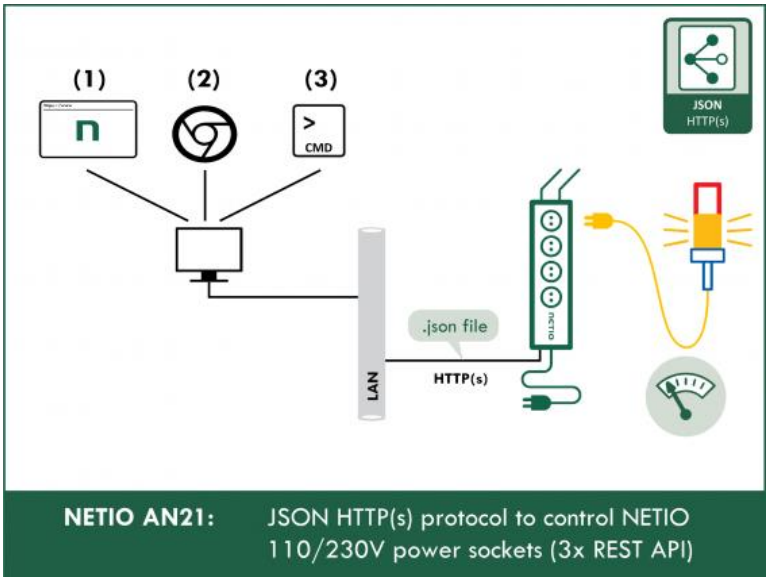
NETIO PowerPDU FK6 – listing of the netio.json file

Note: In the NETIO PowerPDU FK6 model having 6 metered outputs including 2 inputs


```
{
"Agent":{"Model":"FK6","DeviceName":"PowerPDU-
63","MAC":"24:A4:2C:3B:B8:63","SerialNumber":"24A42C3BB863","JSONVer":"2.6","Time":"202
6-03-
02T10:32:26+0100","Uptime":80184,"Version":"5.1.5","OemID":600,"VendorID":0,"NumOutputs
":6,"NumInputs":2},
"GlobalMeasure":{"Voltage":239.16,"TotalCurrent":139,"OverallPowerFactor":999.00,"Total
PowerFactor":999.00,"OverallPhase":999.00,"TotalPhase":999.00,"TotalEnergy":37,"TotalRe
verseEnergy":22,"TotalEnergyNR":37,"TotalReverseEnergyNR":22,"TotalLoad":3,"EnergyStart
":"1970-01-01T00:00:00+0100"},
"Outputs":[
{"ID":1,"Name":"Power output
1","State":1,"Action":6,"Delay":5000,"Current":23,"PowerFactor":0.09,"Phase":277.07,"En
ergy":2,"ReverseEnergy":7,"EnergyNR":2,"ReverseEnergyNR":7,"Load":1,"EnergyStart":"1970
-01-01T00:00:00+0100"},
{"ID":2,"Name":"Power output
2","State":1,"Action":6,"Delay":5000,"Current":23,"PowerFactor":0.11,"Phase":281.10,"En
ergy":10,"ReverseEnergy":0,"EnergyNR":10,"ReverseEnergyNR":0,"Load":1,"EnergyStart":"19
70-01-01T00:00:00+0100"},
{"ID":3,"Name":"Power output
3","State":1,"Action":6,"Delay":5000,"Current":23,"PowerFactor":0.08,"Phase":276.24,"En
ergy":2,"ReverseEnergy":7,"EnergyNR":2,"ReverseEnergyNR":7,"Load":0,"EnergyStart":"1970
-01-01T00:00:00+0100"},
{"ID":4,"Name":"Power output
4","State":1,"Action":6,"Delay":5000,"Current":24,"PowerFactor":0.11,"Phase":279.66,"En
ergy":10,"ReverseEnergy":0,"EnergyNR":10,"ReverseEnergyNR":0,"Load":1,"EnergyStart":"19
70-01-01T00:00:00+0100"},
{"ID":5,"Name":"Power output
5","State":1,"Action":6,"Delay":5000,"Current":23,"PowerFactor":0.10,"Phase":276.06,"En
ergy":2,"ReverseEnergy":8,"EnergyNR":2,"ReverseEnergyNR":8,"Load":1,"EnergyStart":"1970
-01-01T00:00:00+0100"},
{"ID":6,"Name":"Power output
6","State":1,"Action":6,"Delay":5000,"Current":23,"PowerFactor":0.09,"Phase":278.97,"En
ergy":10,"ReverseEnergy":0,"EnergyNR":10,"ReverseEnergyNR":0,"Load":1,"EnergyStart":"19
70-01-01T00:00:00+0100"}
],
"Inputs":[
{"ID":1,"Name":"Input
1","State":0,"S0Counter":0,"TemperatureC":22.8,"TemperatureF":73.0,"Humidity":999},
{"ID":2,"Name":"Input
2","State":0,"S0Counter":1,"TemperatureC":22.8,"TemperatureF":73.0,"Humidity":37}
],
"PAB":[
{"Type":"RANGE","Name":"CR03_PAB","Enabled":false,"In":false}
],
"Watchdogs":[
{"Name":"google_pinger","Enabled":false,"Fail":false,"LastStatus":false,"Timestamp":0},
{"Name":"CR01_WDT","Enabled":false,"Fail":false,"LastStatus":false,"Timestamp":0}
],
"Rules":[
{"Name":"CR01_RULE","Enabled":false,"Result":null},
{"Name":"CR02_RULE","Enabled":false,"Result":null},
{"Name":"CR03_RULE","Enabled":false,"Result":null},
{"Name":"CR13_RULE","Enabled":false,"Result":null}
]}
```

Examples

AN21: JSON HTTP(S) protocol to control NETIO 110/230V power sockets (3x REST API)



The AN21 Application Note shows how to access measurements and control electrical sockets on a NETIO 4x device from third-party applications using the JSON protocol. AN21 demonstrates several different ways to control NETIO power sockets by transferring a netio.json file over http.

The first method uses the “Device HTTP(s) File Upload” tool in the device’s web interface. The second method transfers the JSON file using a Chrome browser extension. The third method uses CURL (command-line tool) to transfer files over http.

>> Read the AN21 on [www.netio-products.com]

Document history

Document Revision	Publication Date	Description
1.0	14.11.2017	Initial release - JSON Version 2.0, for FW 3.0.1 (Netio4x devices)
1.1	7.12.2017	Documentation optimization - Action 6
1.2	19.12.2017	Keywords added
1.3	31.8.2018	Values description updated
1.4	6.9.2018	Infographic added
1.5	16.10.2018	Minor edits
1.6	23.11.2018	Detailed description about the Action tag implemented
1.7	29.1.2020	Added SerialNumber & MAC description for PowerCable/PowerBox/PowerPDU/PowerDIN devices JSON version changed to 2.1
1.8	29.1.2020	Version -> 2.2 (due to inconsistencies with N4, where 2.1 already was)
1.9	19.3.2020	Detail description of difference between MAC and SerialNumber tag was added.
2.0	11.9.2020	JSON Version -> 2.3: Added Inputs, Phase, ReverseEnergy; new supported devices
2.1	24.3.2021	New compatible devices listed
2.2	17.6.2021	JSON Version -> 2.4: Added Reversed energy & DI tags into the JSON structure
2.3	20.7.2021	Minor edits
2.4	13.9.2021	Error codes updated
2.6	20.02.2026	Adding additional commands (temperature, humidity output rename, scheduler, more examples....)