NETIO T1 Sensor

MANUAL

Table of Contents

1. Introduction 3

2. General Characteristic 3

2.1 Technical properties: 3

2.2 Application 3

2.3 Compatibility: 3

2.4 Product description 4

3. Diagram: 5

4. NETIO Web interface 5

5. NETIO Cloud 6

6. Power Analyses Block (PAB) 7

7. Condition & Rules (CR) 8

8. API 9

8.1 M2M API Protocol – XML over HTTP 9

8.2 M2M API Protocol – JSON over HTTP 12

8.3 M2M API Protocol – MQTT Flex 16

1. Introduction

NETIO T1 sensor is plug & play solution for real time temperature monitoring. T1 Sensor is compatible with NETIO products equipped by DI (Digital Input). By adding temperature sensor into NETIO product's family, we would be able to offer our customers additional benefits for PDU usage.

2. General Characteristic

2.1 Technical properties:

- Operating range: -20℃ to +80 ℃
- Accuracy: Up to ±0.4 ℃
- Interface: DI (Digital Input)
- Cable: PVC shielded cable, 3 m
- Probe: Stainless steel, 60 mm, Ø 6 mm
- IP 67 - rating ensure a high level of protection against dust and water according IEC 60529 standard; **device is completely waterproof (level6) and protected against immersion in water up to depth pf 1m/30 min (level)

2.2 Application

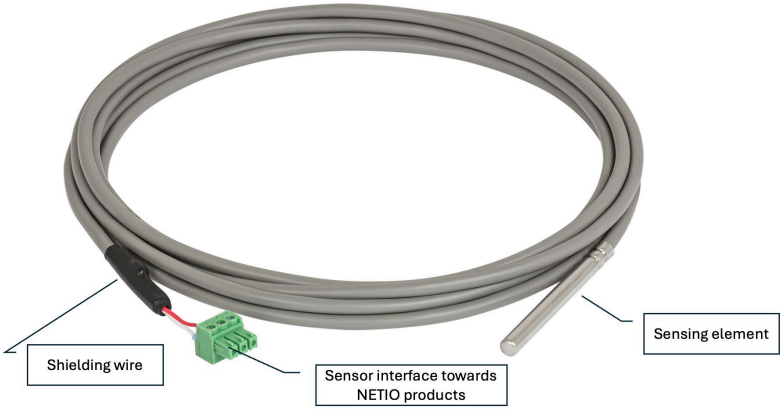
- Thermostat switching power for heating / cooling device
- Indoor/outdoor temperature monitoring (warehouse, production, meeting rooms)
- IT – Temperature in rack, data center or server rooms
- AV – Studios, conference rooms, museums, cinemas, smart home, digital signature

2.3 Compatibility:

- All NETIO devices equipped by DI (Digital Inputs)
- NETIO device must run on FW 5.0.3+
- There can be two NETIO T1 Sensors connected into NETIO devices equipped by two DI. (PowerCable 2Kx; PowerDIN 4PZ)

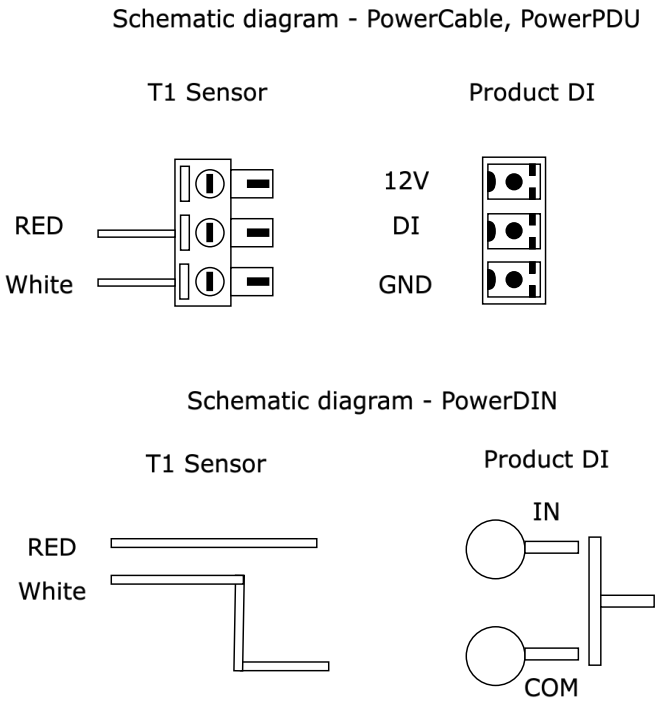
NETIO device	Compatibility with T1 sensor	NETIO device	Compatibility with T1 sensor
PowerCable 2Kx	✓	PowerCable 2Px	✗
PowerPDU 4Kx	✓	PowerCable REST 101x	✗
PowerPDU 8Qx	✓	PowerBox 3Px	✗
PowerPDU 8Kx	✓	PowerBox 4Kx	✗
PowerDIN 4PZ	✓	PowerPDu 4C	✗

2.4 Product description



**Sensor is equipped by shielding wire (under black tape) connected into sensing element.
**typical example shielding usage: – high frequency application, application with EMC noise..etc

3. Diagram:



** Maximum cable length extension up to 20m

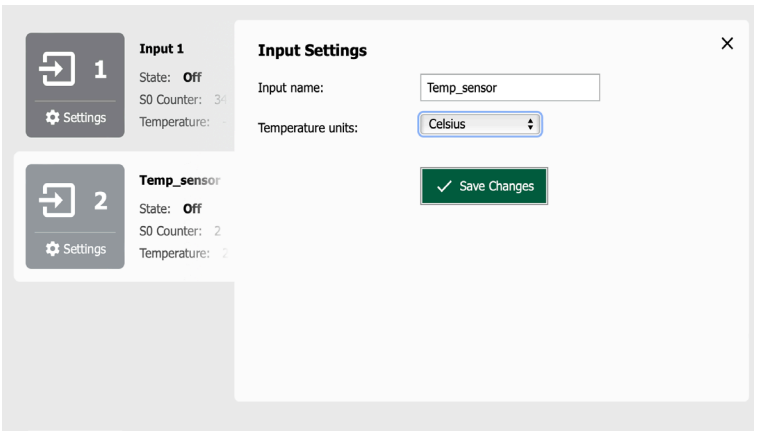
4. NETIO Web interface

Information about metered value is displayed at Input SECTION. The metered unit can be specified by user:

- Celsius - [°C]
- Fahrenheit - [°F]

In case NETIO T1 Sensor is not connected Web interface shows [-]

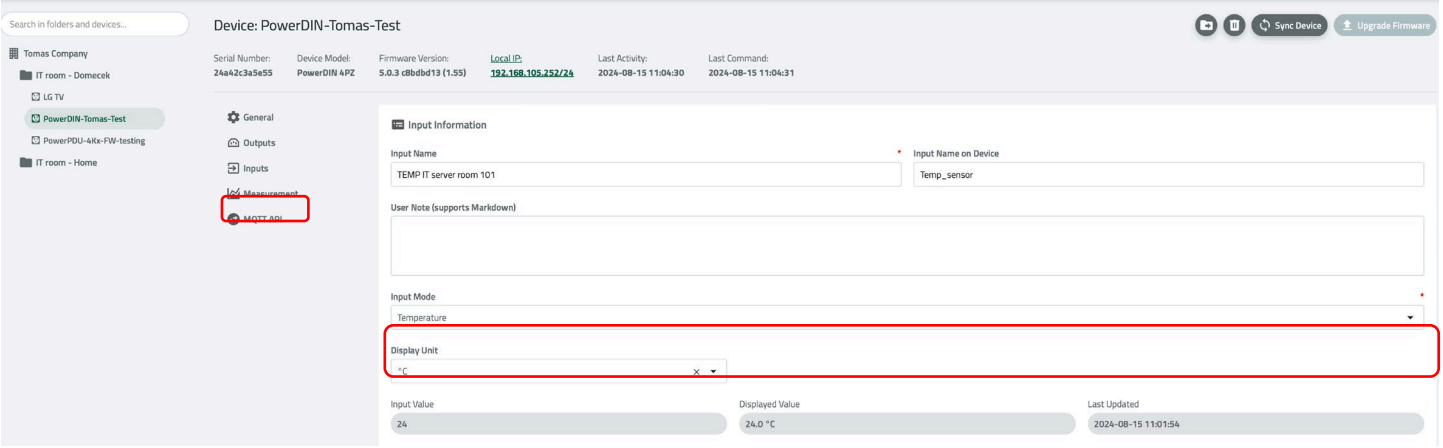
Web interface with actual temperature

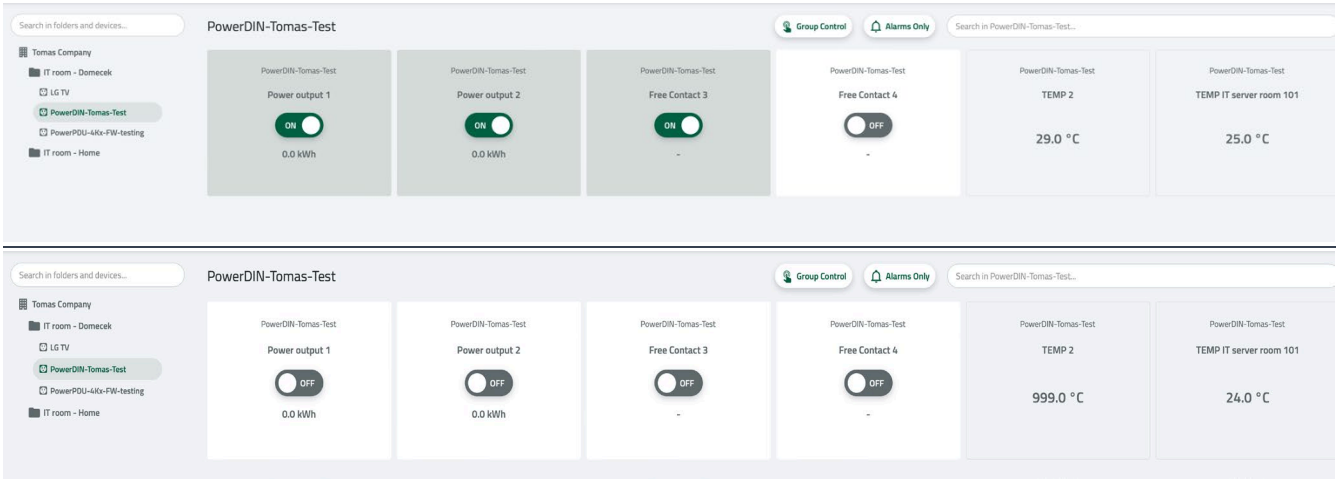


Unit definition

5. NETIO Cloud

Output from temperature metering is available also through NETIO Cloud. Metered unit is displayed at INPUT section. To be able display values at Input/Output section. Please set it up Temperature at Inputs/ Input Mode.





** In case NETIO T1 Sensor is not connected Cloud will interprets as [999]

6. Power Analyses Block (PAB)

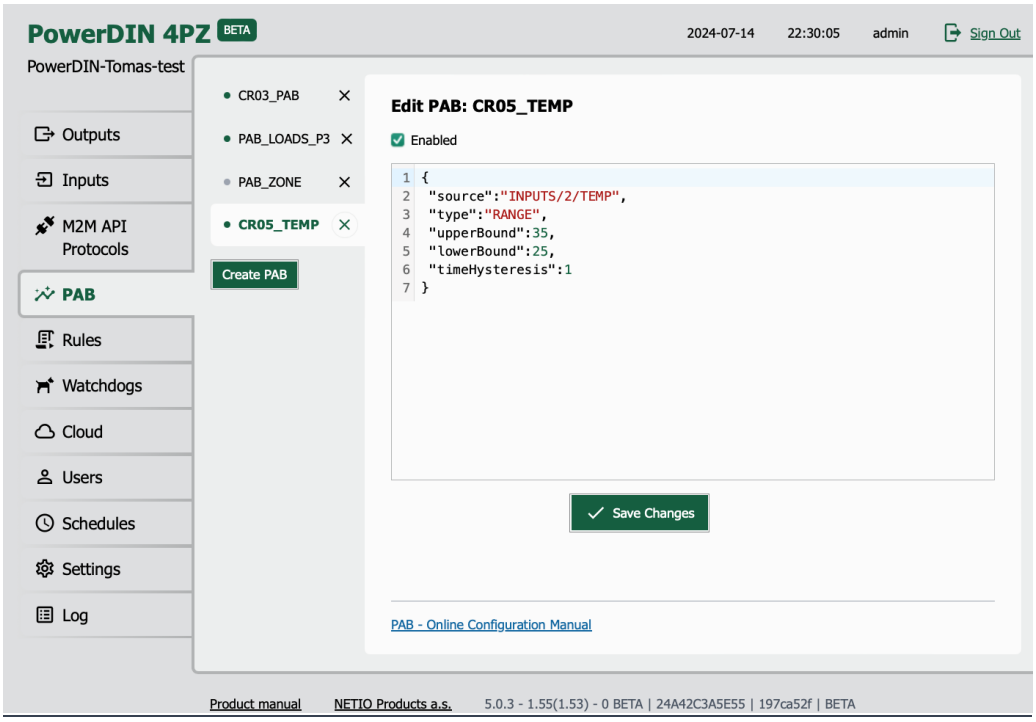
The PAB (Power Analysis Block) is Condition & Rules (C&R) function available for NETIO devices that periodically evaluates sets of conditions. Based on these conditions, subsequent Rules can be executed.

These conditions can consist of one or more of following variables:

- Current [mA],
- Energy [Wh]
- Load [W]
- Power Factor
- S0 pulses
- TEMP [°C], [°F]

Several PAB's can be configured and running simultaneously. Details and examples can be find at our online resource center:

https://wiki.netio-products.com/index.php?title=PAB

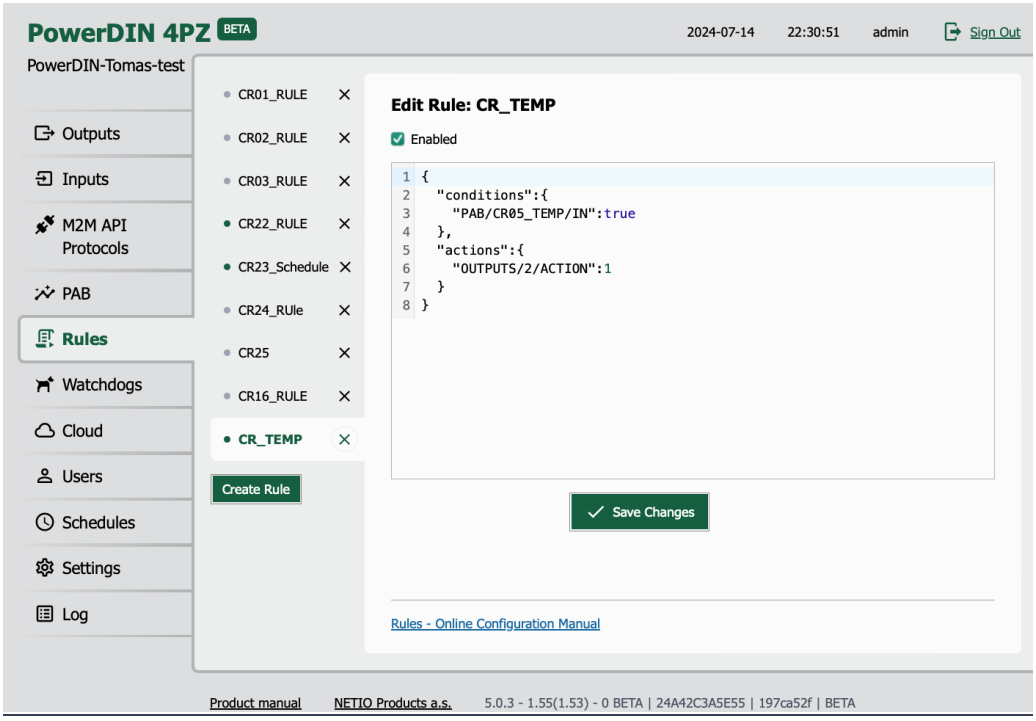


name	The name of PAB configuration.
Enable PAB	Check to enable this PAB
PAB config	PAB configuration.
Save Changes	Saves the changes.
Create PAB	*Opens a dialog to enter the new PAB configuration.*

7. Condition & Rules (CR)

The concept of NETIO Conditions (PAB & WatchDog) & Rules is a set of prepared detections Conditions and Rules that act upon them. They are implemented directly in NETIO PDU devices.

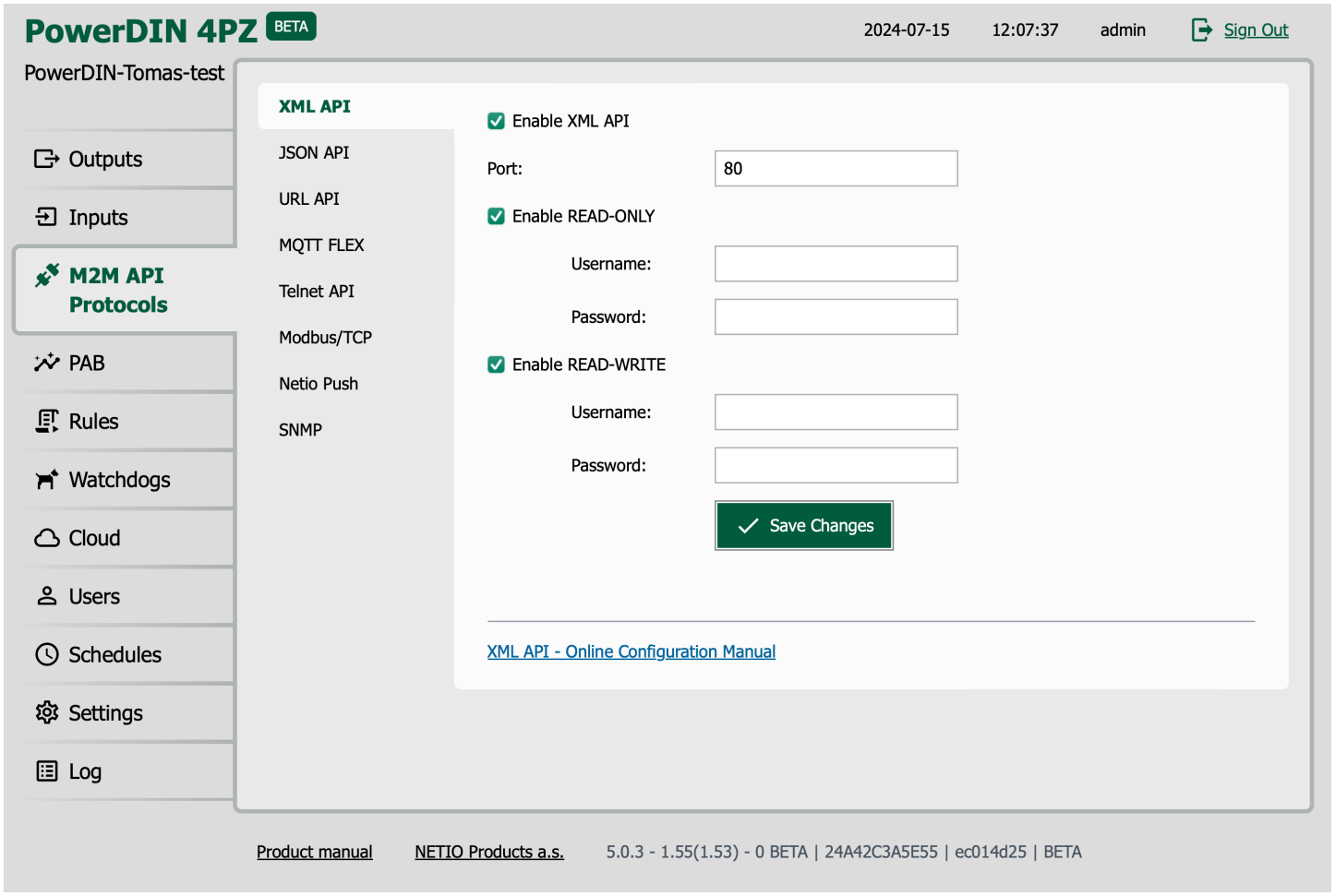
https://wiki.netio-products.com/index.php?title=Rules
https://wiki.netio-products.com/index.php?title=Rules_Complete_Examples



name	The name of CR configuration.
Enable C*R*	Check to enable this CR
C*R* config	CR configuration.
Save Changes	Saves the changes.
Create C*R*	*Opens a dialog to enter the new CR configuration.*

8. API

8.1 M2M API Protocol – XML over HTTP



Enable XML API	Enables M2M XML API functions in the system.
Port	Read-only value. Indicates the port where the device currently listens for M2M XML API commands. This port is the same for all http based M2M API protocols and web administration and can be changed in Settings / System (chapter System).
Enable READ-ONLY	Enables **Read-Only** access via M2M XML API for monitoring. You may also fill in the username and password for this mode.
Enable READ-WRITE	Enables **Read/Write** access for monitoring (reading values) and writing (output control). You may also fill in the username and password for this mode.
Username	Username for the respective access mode (Read-Only/ReadWrite). **Note** – this is unrelated to the username for accessing the NETIO 4x web administration interface. When left empty, the protocol will not require any authentication.
Password	Password for the corresponding username (Read-Only/ReadWrite).
Save Changes	Saves the changes.

As the XML API is enabled, other M2M protocols are disabled except SNMP after clicking **Save changes**.

** Return value – in case of sensor connected

- *<TempertureC>26.3</*
- *<TemperatureF>79.3</*

** Return value - In case sensor is not connected or failed

- *<TemperatureF>999.0</*
- *<TempertureC>999.0</*

Example of the netio.xml file


```
xml version="1.0" encoding="utf-8"?> <set:Root xmlns:set="http://www.netio-products.com/XMLSchema/NETIO.xsd"> <Agent> <Model>4PZ</Model> <DeviceName>PowerDIN-Tomas-test</DeviceName> <OemID>400</OemID> <VendorID>0</VendorID> <Version>5.0.3</Version> <XmlVer>2.4</XmlVer> <SerialNumber>24A42C3A5E55</SerialNumber> <MAC>24:A4:2C:3A:5E:55</MAC> <Uptime>1884</Uptime> <Time>2024-07-15T11:33:53+0100</Time> <NumOutputs>4</NumOutputs> <NumInputs>2</NumInputs> </Agent> <GlobalMeasure> <OverallPowerFactor>999.00</OverallPowerFactor> <TotalPowerFactor>999.00</TotalPowerFactor> <Voltage>235.28</Voltage> <OverallPhase>999.00</OverallPhase> <TotalPhase>999.00</TotalPhase> <Frequency>49.98</Frequency> <TotalLoad>0</TotalLoad> <TotalCurrent>0</TotalCurrent> <TotalEnergy>5</TotalEnergy> <TotalReverseEnergy>0</TotalReverseEnergy> <TotalEnergyNR>8</TotalEnergyNR> <TotalReverseEnergyNR>0</TotalReverseEnergyNR> <EnergyStart>1970-01-01T00:00:00+0100</EnergyStart> </GlobalMeasure> <Outputs> <Output> <ID>1</ID> <Name>Power output 1</Name> <State>0</State> <Action>6</Action> <Delay>3000</Delay> <PowerFactor>1.00</PowerFactor> <Phase>0.00</Phase> <Load>0</Load> <Current>0</Current> <Energy>2</Energy> <ReverseEnergy>0</ReverseEnergy> <EnergyNR>2</EnergyNR> <ReverseEnergyNR>0</ReverseEnergyNR> </Output> <Output> <ID>2</ID> <Name>Power output 2</Name> <State>1</State> <Action>6</Action> <Delay>5000</Delay> <PowerFactor>1.00</PowerFactor> <Phase>0.00</Phase> <Load>0</Load> <Current>0</Current> <Energy>6</Energy> <ReverseEnergy>0</ReverseEnergy> <EnergyNR>6</EnergyNR> <ReverseEnergyNR>0</ReverseEnergyNR> </Output> <Output> <ID>3</ID> <Name>Trigger on DI1</Name> <State>0</State> <Action>6</Action> <Delay>5000</Delay> </Output> <Output> <ID>4</ID> <Name>Free Contact 4</Name> <State>0</State> <Action>6</Action> <Delay>5000</Delay> </Output> </Outputs> <Inputs> <Input> <ID>1</ID> <Name>Input 1</Name> <State>0</State> <S0Counter>34</S0Counter> <TempertureC>999.0</TempertureC> <TemperatureF>999.0</TemperatureF> </Input> <Input> <ID>2</ID> <Name>Temp_sensor</Name> <State>0</State> <S0Counter>2</S0Counter> <TempertureC>26.3</TempertureC> <TemperatureF>79.3</TemperatureF> </Input> </Inputs> <PAB> <PABItem><Type>RANGE</Type> <Name>PAB_LOADS_P3</Name> <Enabled>true</Enabled> <In>false</In> </PABItem> <PABItem><Type>ZONES</Type> <Name>PAB_ZONE</Name> <Enabled>false</Enabled> <Zone>0</Zone> </PABItem> <PABItem><Type>RANGE</Type> <Name>CR05_TEMP</Name> <Enabled>true</Enabled> <In>true</In> </PABItem> </PAB> <Watchdogs><WdtItem><Name>google_pinger</Name> <Enabled>true</Enabled> <Fail>false</Fail> <LastStatus>true</LastStatus> <Timestamp>1721043233</Timestamp> </WdtItem> <WdtItem><Name>CR01_WDT</Name> <Enabled>false</Enabled> <Fail>false</Fail> <LastStatus>false</LastStatus> <Timestamp>0</Timestamp> </WdtItem> </Watchdogs> <Rules><Rule><Name>CR01_RULE</Name> <Enabled>false</Enabled> <Result>null</Result> </Rule> <Rule><Name>CR02_RULE</Name> <Enabled>false</Enabled> <Result>null</Result> </Rule> <Rule><Name>CR03_RULE</Name> <Enabled>false</Enabled> <Result>null</Result> </Rule> <Rule><Name>CR22_RULE</Name> <Enabled>true</Enabled> <Result>null</Result> </Rule> <Rule><Name>CR23_Schedule</Name>
```

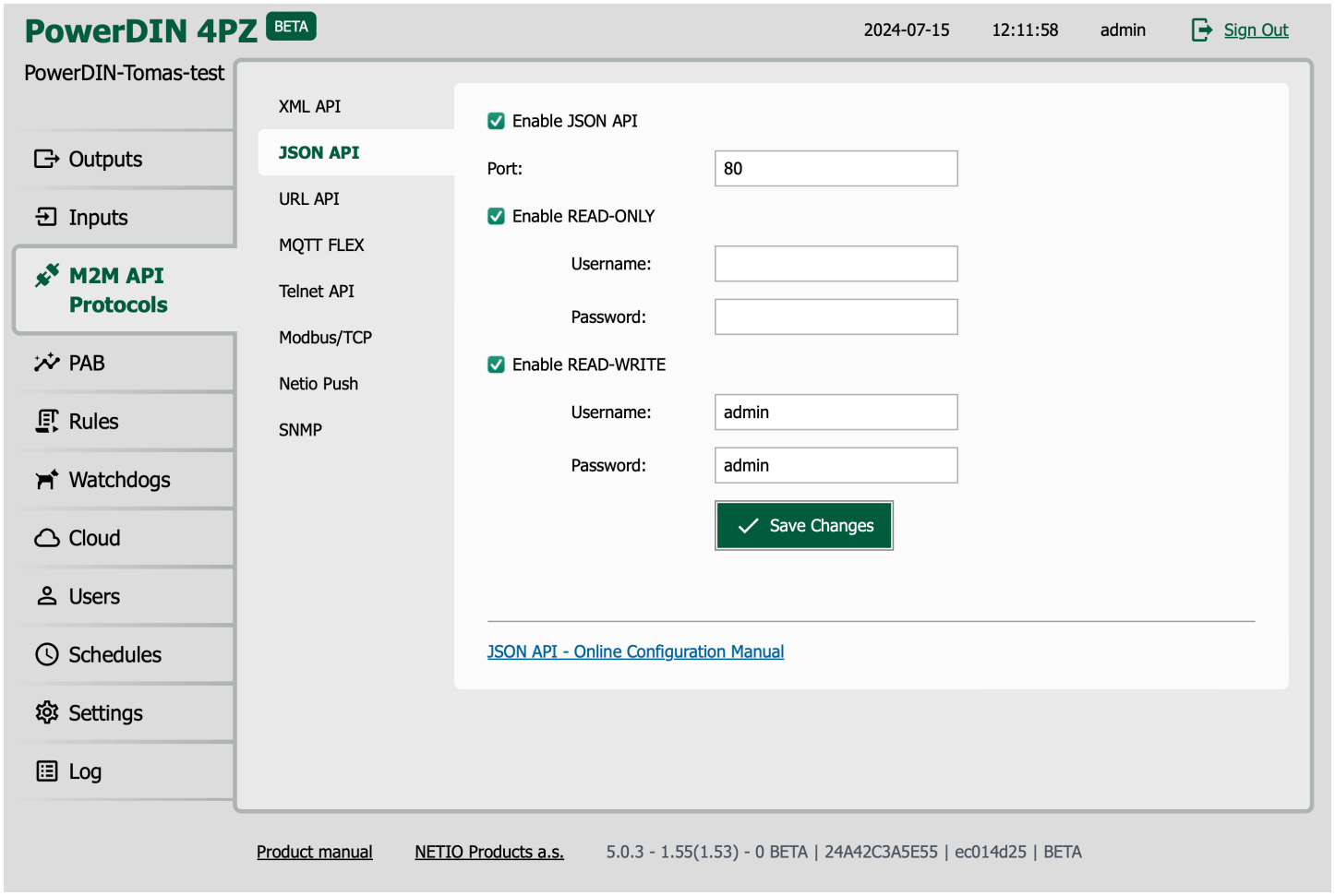
```
<Enabled>true</Enabled> <Result>null</Result> </Rule> <Rule><Name>CR24_RUle</Name>
<Enabled>false</Enabled> <Result>null</Result> </Rule> <Rule><Name>CR25</Name>
<Enabled>false</Enabled> <Result>null</Result> </Rule> <Rule><Name> CR16_RULE</Name>
<Enabled>false</Enabled> <Result>null</Result> </Rule> <Rule><Name>CR_TEMP</Name>
<Enabled>true</Enabled> <Result>true</Result> </Rule> </Rules> </set:Root>
```

For the complete specifications of the M2M XML API protocol, visit the **Support > Download** section of our website and see the following document:
XML - description of NETIO M2M API interface - PDF

For more information and a practical demonstration of using the XML protocol with NETIO smart sockets, see the following Application Note:

AN20 XML HTTP(s) protocol to control NETIO smart power sockets 110/230V

8.2 M2M API Protocol – JSON over HTTP



Enable JSON API	Enables M2M JSON API functions in the system.
Port	Read-only value. Indicates the port where the device currently listens for M2M JSON API commands. This port is the same for all http based M2M API protocols and web administration and can be changed in Settings / System (chapter System).
Enable READ-ONLY	Enables **Read-Only** access via M2M JSON API for monitoring. You may also fill in the username and password for this mode.
Enable READ-WRITE	Enables **Read/Write** access for monitoring and output control. You may also fill in the username and password for this mode.
Username	Username for the respective access mode (Read-Only/ReadWrite). Note – this is unrelated to the username for accessing the NETIO device web administration. When left empty, the protocol will not require any authentication.
Password	Password for the corresponding username (Read-Only/ReadWrite).
Save Changes	Saves the changes.

When the JSON API protocol is enabled, other M2M protocols are disabled except SNMP after clicking **Save changes**.

**return value in case sensor is connected

- *"TempertureC": 26.3,*
- *"TemperatureF": 79.3*

** return value in case sensor is not connected or failed

- *"TempertureC": 999,*
- *"TemperatureF": 999*

Example of the netio.json file


```
Agent": {
   "Model": "4PZ",
   "DeviceName": "PowerDIN-Tomas-test",
   "MAC": "24:A4:2C:3A:5E:55",
   "SerialNumber": "24A42C3A5E55",
   "JSONVer": "2.6",
   "Time": "2024-07-15T11:34:27+0100",
   "Uptime": 1918,
   "Version": "5.0.3",
   "OemID": 400,
   "VendorID": 0,
   "NumOutputs": 4,
   "NumInputs": 2
 },
 "GlobalMeasure": {
   "Voltage": 235.38,
   "TotalCurrent": 0,
   "OverallPowerFactor": 999,
   "TotalPowerFactor": 999,
   "OverallPhase": 999,
   "TotalPhase": 999,
   "Frequency": 50.05,
   "TotalEnergy": 5,
   "TotalReverseEnergy": 0,
   "TotalEnergyNR": 8,
   "TotalReverseEnergyNR": 0,
   "TotalLoad": 0,
   "EnergyStart": "1970-01-01T00:00:00+0100"
 },
 "Outputs": [
  {
    "ID": 1,
    "Name": "Power output 1",
    "State": 0,
```

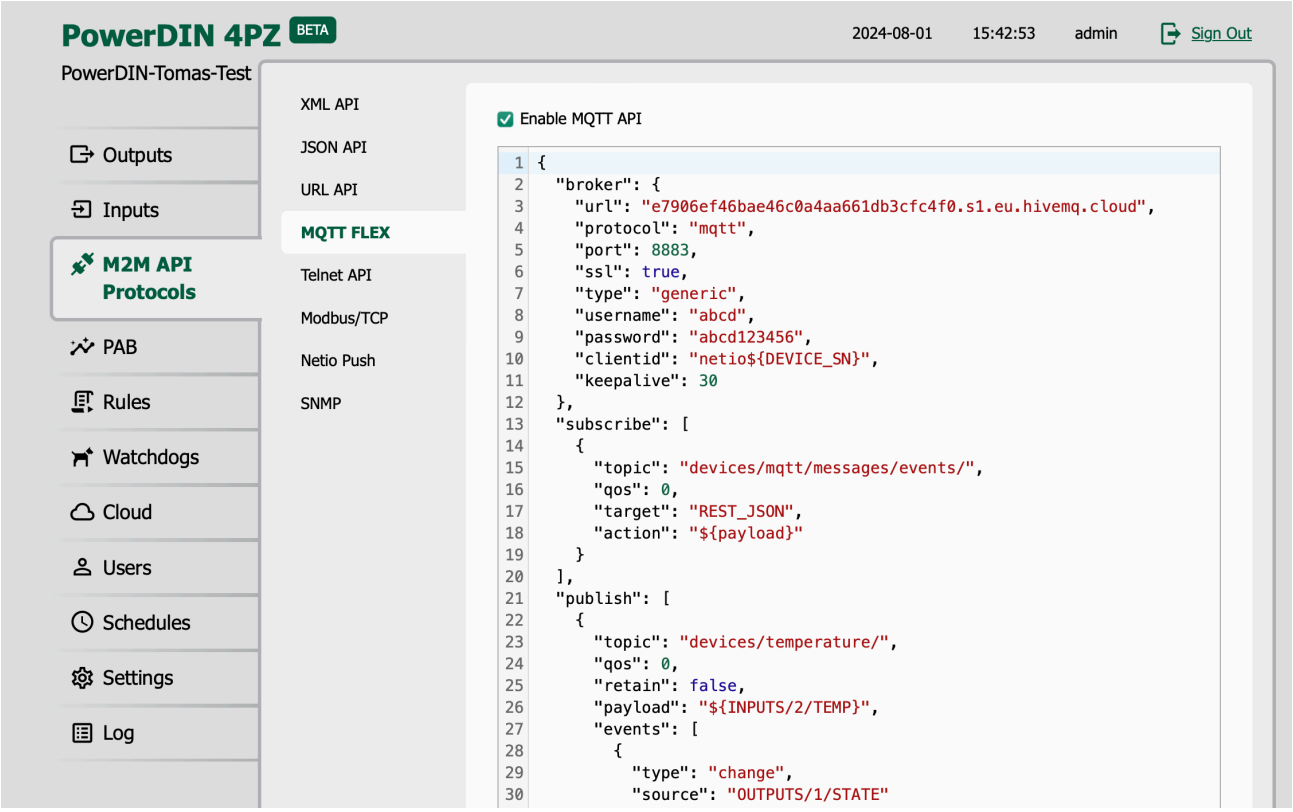
```
      "Action": 6,
      "Delay": 3000,
      "Current": 0,
      "PowerFactor": 1,
      "Phase": 0,
      "Energy": 2,
      "ReverseEnergy": 0,
      "EnergyNR": 2,
      "ReverseEnergyNR": 0,
      "Load": 0,
      "EnergyStart": "1970-01-01T00:00:00+0100"
    },
    {
      "ID": 2,
      "Name": "Power output 2",
      "State": 1,
      "Action": 6,
      "Delay": 5000,
      "Current": 0,
      "PowerFactor": 1,
      "Phase": 0,
      "Energy": 6,
      "ReverseEnergy": 0,
      "EnergyNR": 6,
      "ReverseEnergyNR": 0,
      "Load": 0,
      "EnergyStart": "1970-01-01T00:00:00+0100"
    },
  ],
  "Inputs": [
    {
      "ID": 1,
      "Name": "Input 1",
      "State": 0,
      "S0Counter": 34,
      "TempertureC": 999,
      "TemperatureF": 999
    },
    {
      "ID": 2,
      "Name": "Temp_sensor",
      "State": 0,
      "S0Counter": 2,
      "TempertureC": 26.3,
      "TemperatureF": 79.3
    }
  ],
```

```
    "PAB": [
     {
       "Type": "RANGE",
       "Name": "CR05_TEMP",
       "Enabled": true,
       "In": true
     }
    ],
    "Watchdogs": [
     {
       "Name": "google_pinger",
       "Enabled": true,
       "Fail": false,
       "LastStatus": true,
       "Timestamp": 1721043267
     }
    ],
    "Rules": [

     {
       "Name": "CR_TEMP",
       "Enabled": true,
       "Result": true
     }
    ]

 ] }
```


For more information about the M2M JSON API, visit the **Support > Download** section of our website and see the following document:
JSON - description of NETIO M2M API interface - PDF

For more information and a practical demonstration of using the JSON protocol with NETIO smart sockets, see the following Application Note:
AN21 JSON HTTP(S) protocol to control NETIO 110/230V power sockets (3x REST API)

8.3 M2M API Protocol – MQTT Flex



Enable MQTT-*flex*	Enables MQTT-flex functions in the system.
MQTT-flex Config:	**Text area for entering the MQTT-flex configuration.**
Save Changes	Saves the changes.

When the MQTT-flex is enabled, other M2M protocols are disabled except SNMP after clicking **Save changes**.

NETIO device uses json to define the MQTT-flex structure (MQTT-flex Config). Both subscribe and publish topics can be defined. Publish topics may include actions that initiate a transmission.

MQTT-flex configuration example:

```
{
  "broker": {
    "url":
"e7906ef46bae46c0a4aa661db3cfc4f0.s1.eu.hivemq.cloud",
    "protocol": "mqtt",
    "port": 8883,
    "ssl": true,
    "type": "generic",
    "username": "abcd",
    "password": "abcd123456",
    "clientid": "netio${DEVICE_SN}",
    "keepalive": 30
  },
  "subscribe": [
    {
      "topic": "devices/mqtt/messages/events/",
      "qos": 0,
      "target": "REST_JSON",
      "action": "${payload}"
    }
  ],
  "publish": [
    {
      "topic": "devices/temperature/",
      "qos": 0,
      "retain": false,
      "payload": "${INPUTS/2/TEMP}",
      "events": [
        {
          "type": "change",
          "source": "OUTPUTS/1/STATE"
        },
        {
          "type": "delta",
          "source": "INPUTS/2/TEMP",
          "delta": 1
        },
        {
          "type": "timer",
          "period": 20
        }
```

```
            ]
        }
    ]
}
```

For more information about the M2M MQTT-flex API, visit the **Support > Download** section of our website and see the following document:
MQTT-flex - description of NETIO M2M API interface - PDF

There are wide options for subscribe and publish sections and its possibilities expand over the time.
You will find details and examples at our online resource center:
https://wiki.netio-products.com/index.php?title=MQTT-flex

For more information and a practical demonstration of using the MQTT protocol with NETIO smart sockets, see the following Application Note:
AN40 Getting started with PowerCable MQTT-flex via HiveMQ MQTT broker to mobile App