Dotvision energy

Spoony 1.5 | Manual

- Evolutive Solution
- Load Management
- Secured
- Cloud connected
- Intelligent (automate)
- Small
- Simple
- Wi-Fi Ethernet

energy.dotvision.com

Document versions

Date	Version	Author	Notes
18/10/2021	1.0	Nicolas Leday	First draft – Awaiting verification
22/10/2021	1.1	François Charles	Syntactic/orthographic corrections, addition of figures, corrected title levels
04/11/2021	1.2	Nicolas Leday	Minor corrections Final validation (first Release version)

Table of contents

1 Safety instructions

This product must be installed, configured, and put into service by qualified personnel only. Usual electrical security rules apply and must be followed. In case of incorrect installation or usage, damages to the user or the product can occur.

This product must only be manipulated when no power is present. Risk of electrical shock exists if the product is misused.

This product must be installed, configured and used according to the related documentation. Any use not described in the related documentation must be considered unsafe. If the product is not used according to the cases described in the related documentation, no warranty is applicable.

To avoid product overheating, only use in a well-ventilated area.

Product disassembly and fixing can only be realized by authorized and qualified personnel. In case of doubt regarding the product functioning, please contact DotVision. Incorrect use can damage the product.

2 Product characteristics

Visual characteristics of actual product may differ. Colors and terminals type may differ for older or newer revision of the product.

2.1 Presentation

Spoony by DotVision can be described as an Energy Edge Gateway, a single device holding key functionalities for energy monitoring:

- Single and three-phase energy measurement,
- Network connectivity allowing direct connection to a storage server (cloud-based or on-premises),
- Local data storage,
- Data aggregation thanks to integrated MODBUS client functionality (Option - RS485 and TCP) and TIC demodulator functionality,
- Control-command with 3 integrated open-drain outputs suitable to drive up to three relays.

Spoony integrated software is based on BluePanda operating system by DotVision, providing unprecedented flexibility in terms of configuration and product customization. Contact DotVision for any question related to product customization or new protocols integration.

2.2 Technical specifications

Product name	Spoony
Manufacturer	DotVision
Size	90 x 60 x 53mm
Format	DIN-rail mount module, 53mm width (3 ½ modules). Compatible with Omega type rails
Environmental	IP20
Casing material	ABS
Power supply input range	80-270 VAC, 50-60Hz. 0.2A max
Temperature range	-10°C to 60°C
Humidity	Max 80% non-condensing
Certifications	Standard conformity to European Directives (CE) for low voltage and CEM, ROHS, REACH
Voltage measurement inputs range	400Vpp Line to Neutral, 800Vpp Line to Line Input impedance: ≥ 2MΩ
Current measurement mode	Indirect, through current transformers or Rogowski coils
Current measurement inputs	Current transformers version: ±100mA, Impedance: 5Ω Rogowski coils version: ±250mV, Impedance ≥ 380kΩ
Current measurement range	Depending on current transformers or Rogowski coils choice. Refer to sensor datasheet.
Electrical Network type	Single-phase or three-phase electrical networks, with or without neutral [1].
Measurement Bandwidth (-3dB)	IRMS, power: 14kHz VRMS: 260Hz
Measurements update frequency	1Hz for all measurements
Metering IC	Analog Devices ADE7758
Connection	Power and voltage measurement: screw terminals, 0.2 to 1.5mm² Current measurement: push terminals, 0.1 to 0.75mm² RS485, TIC, relays: screw terminals Network: RJ45
Accuracy	RMS voltage measurement: 1% over 20:1 dynamic range RMS current measurement: depending on current sensor choice. Typical 1% over 20:1 dynamic range for sensors provided by DotVision [2]. Active power/energy measurement: depending on current sensor choice. Typical 1% over 20:1 dynamic range for sensors provided by DotVision [2].

Figure 1 - Technical specifications

Note 1: A neutral connection may still be required for powering the device due to the limited input range of power input.

Note 2: Devices provided by DotVision are always calibrated together with provided current sensors for optimal accuracy. Accuracy is not guaranteed if current sensors are replaced without proper calibration. Consult DotVision if you need to change the current sensors provided with the device to determine if a calibration is required.

2.3 Terminals' description

Terminals' pinout is indicated onto the product. Pinout is available here for convenience:

Figure 2- Main terminals' pinout

N, L: Power supply inputs. 80 to 270 VAC, 50-60Hz

L1, L2, L3, N': Voltage measurement inputs. Refer to next chapter for details

Ix+ / Ix-: Current measurement inputs (x = 1, 2 or 3)

Figure 3- Auxiliary terminals' pinout

A, B, C: Open-drain relay outputs

+: +12V outputs

T1, T2: TIC demodulator input

2.4 Device front panel

2.4.1 Light indicators

Device front panel includes three LED indicators which give information on the device status.

These LEDs are labelled as follow:

Power - Green

Network - Orange

Error – Red

The following list summarizes possible states for the LEDs:

Blinking: the LED is ON for 500ms and OFF for 500ms,

Flashing: the LED is OFF for 2000ms and briefly ON for 80ms,

On: the LED is continuously ON,

Off: the LED is continuously OFF,

X: the LED state is undefined.

The related device status is given in the table below:

POWER	NETWORK	ERROR	Device Status
Blinking	X	X	The device is initializing. If network connection is not configured on the device (Data logger mode), the device will stay in this mode
Flashing	X	X	The device is running and has established a network connectivity since its startup
X	Off	X	Network connection is down
X	On	X	Network connection is up. If DHCP is activated, the device obtained a valid IP lease
X	X	Blinking	MicroSD card is not properly inserted, or device configuration is invalid

Figure 4- LEDs indicators signification

2.4.2 Front panel, microSD card and reset button access

Device's front panel can be lifted from the bottom side to access microSD card and the reset button, as shown in the figure below:

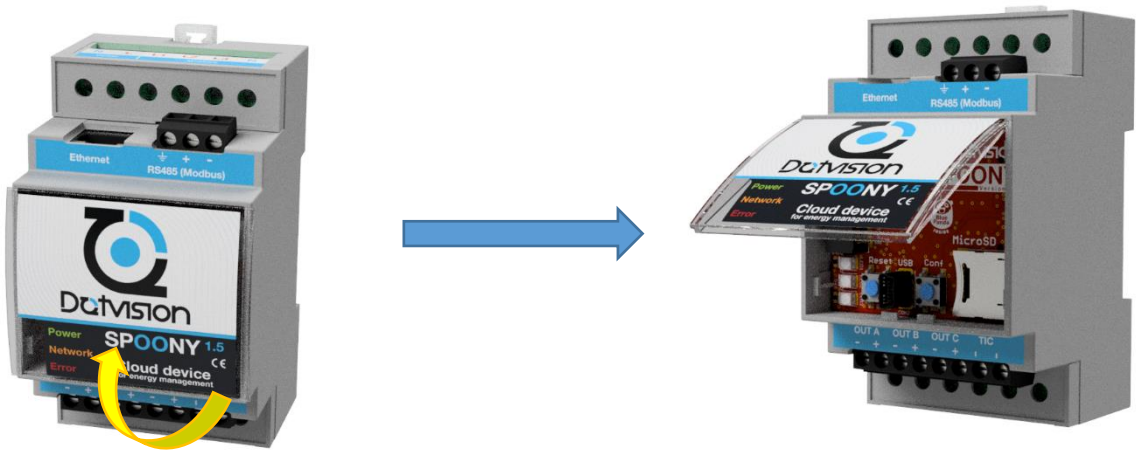

Figure 5- Front panel opening

The following figure shows front panel's key elements:

Figure 6- Front panel view

1 – Micro SD card holder. MicroSD card is extracted by sliding the connector cap in the direction indicated by the arrow.

2 – Reset button. Press this button to restart the device. Reset is required after a firmware update for example (see chapter 4.12).

3 Installation

3.1 Electrical Wiring

3.1.1 Generalities

Installation must be performed by authorized and qualified personnel only.

All location-specific electrical wiring requirements and recommendations should be followed depending on the site of installation.

Suitable protection equipment should be installed on powering and voltage measurement rails (see next sections for details).

Always follow local regulations and recommendations when working with electrical appliances.

DotVision recommends use of a flat screwdriver of 2.5mm width for all terminals.

Warning: current measurement terminals are push type, do not try to screw these terminals. However, the same flat screwdriver can be used to push onto those terminals.

3.1.2 Powering

Device must be powered from an 80 to 270VAC voltage source. Power supply is typically provided from a phase + neutral circuit.

Cabling should be accomplished with 0.5 to 0.75mm² cable. DotVision recommends use of multi-strand flexible cable with ferules on each end.

Power line must be protected according to selected cable section. **DotVision recommends use of 0.2 to 2A protection of preferred type (circuit breaker or fuse).**

Device is equipped with its own onboard fuse. This fuse acts as a security in case of overvoltage and is not replaceable by the end user. Contact DotVision if you think that the device fuse should be replaced.

Following diagram shows how Spoony should be connected to its power supply:

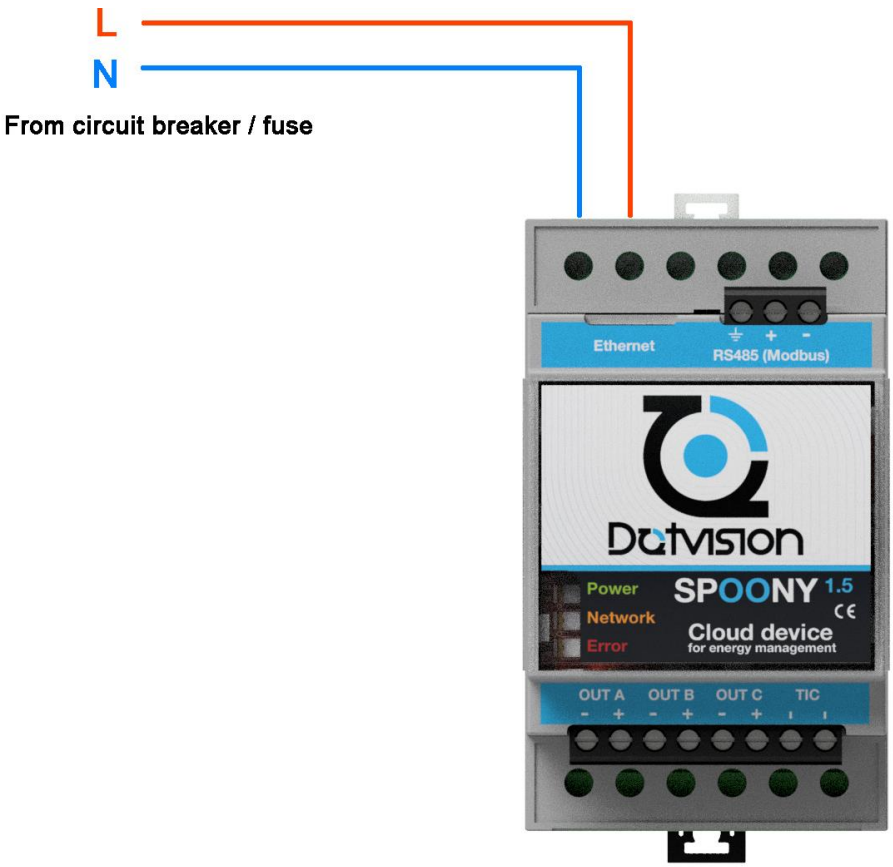

Figure 7- Power connection

3.1.3 Single-phase power measurement

Device is suitable to measure up to three single-phase equipment.

This section covers the case of single-phase installation only. If you want to measure 3 single-phase equipment on a three-phase network, please follow next section (three-phase power measurement) recommendations regarding voltage and current channels identification.

When used in a single-phase electrical network, all voltage reference inputs (L1, L2 and L3) should be tied together, i.e., shorted.

In most cases, L1, L2 and L3 should be connected to L which allows the use of a single protection equipment to protect both power and measurement inputs.

N' must be connected to neutral. In most case, N' should also be tied to N.

Specific cases may require running separate line and neutral pairs to the device for powering and measurement, for example if measured line may be disconnected in normal use. It is not recommended to power the device from an intermittent power supply to avoid data loss. This specific case is not covered in this document.

Following figure shows typical connection of power and voltage measurement inputs in single-phase electrical networks:

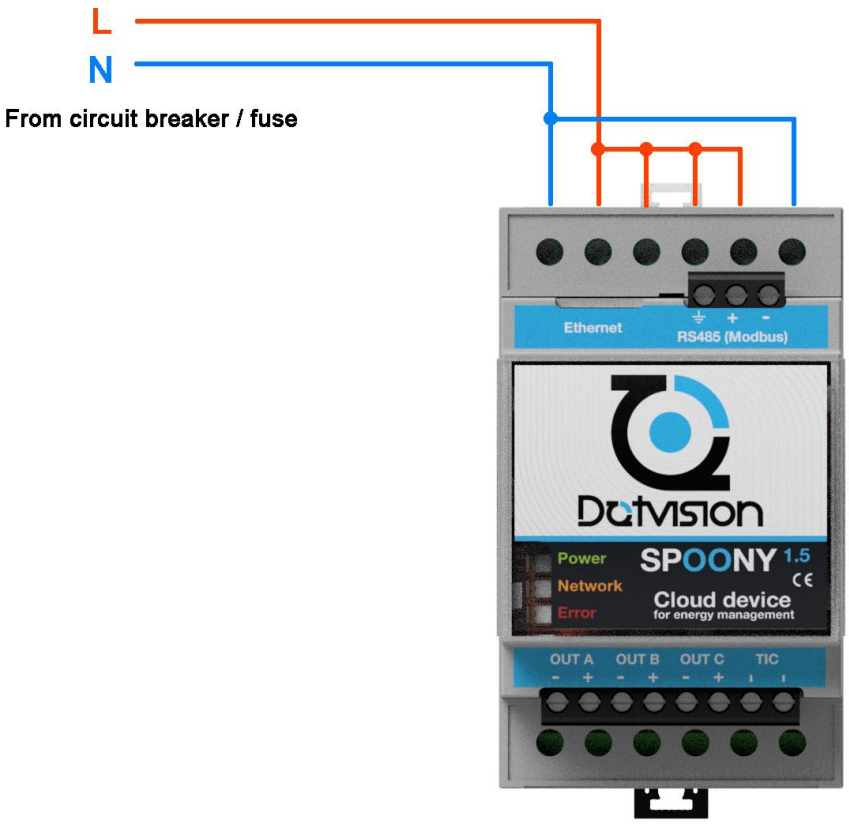

Figure 8- Power and voltage measurement connection in single-phase electrical network

DotVision recommends the use of two-cable ferrules to form the bridges between L, L1, L2, L3 and N, N' as shown in the picture bellow:

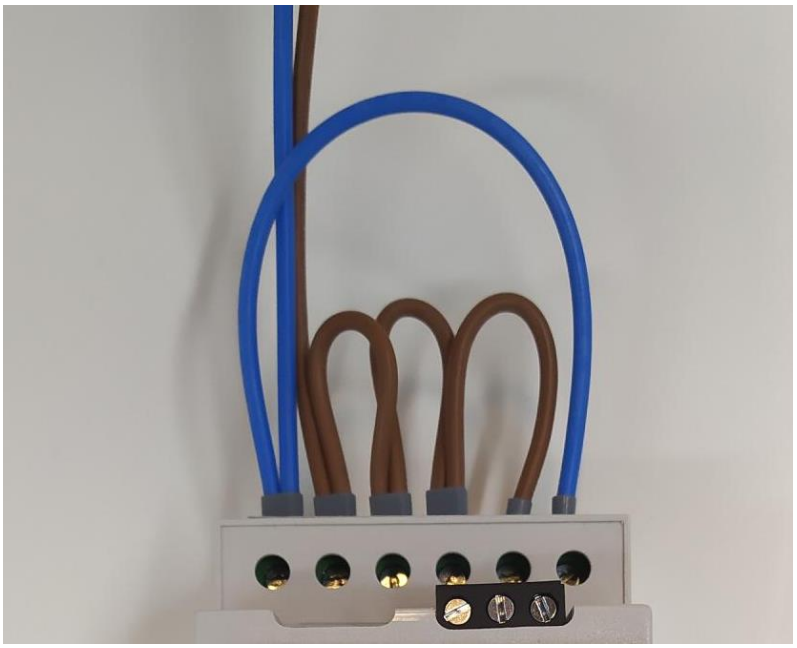

Figure 9 - Wiring in single-phase electrical network

Current sensors are then installed onto desired lines. DotVision provides only split-core current transformers or Rogowski coils which can be installed without unplugging the monitored line. Please refer to paragraph 3.1.5 for detailed procedure.

3.1.4 Three-phase power measurement

Device is suitable for multi or three-phase power measurement. Wiring differs from single-phase power measurement because phases must be properly identified:

Voltage on L1 must be from the same phase going through current sensor 1,

Voltage on L2 must be from the same phase going through current sensor 2,

Voltage on L3 must be from the same phase going through current sensor 3.

In most cases, phases are easily identified by their colors. However, it may be necessary to use a voltmeter to confirm phase number.

Measurement in three-phase electrical network can be achieved with or without neutral connection. Note however that a line + neutral connection is required for device powering.

If the monitored equipment has a neutral connection, it should be tied to neutral input (N') on Spoony for optimal accuracy.

Following diagram shows typical connection of power and voltage measurement inputs in three-phase electrical network:

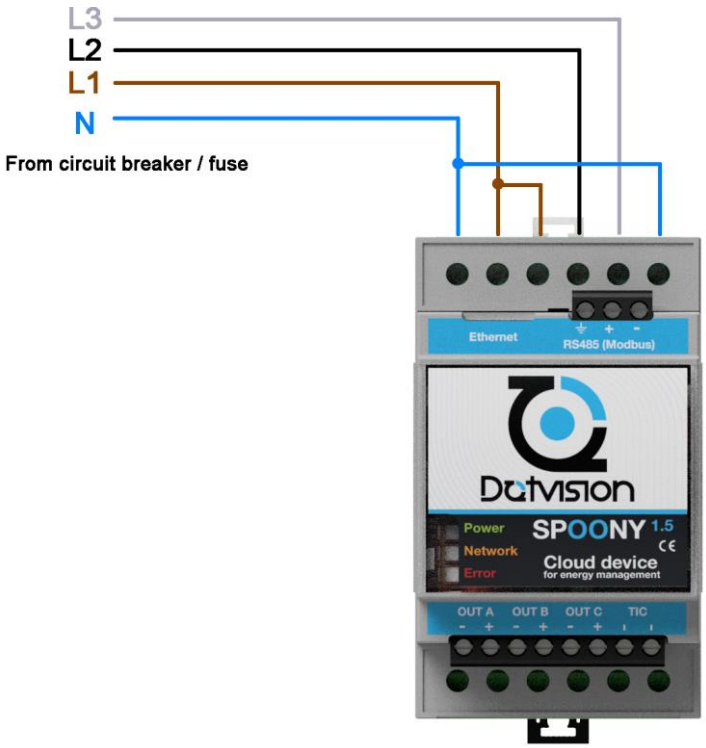

Figure 10- Power and voltage measurement in three-phase electrical network

Current sensors are then installed onto lines going to the monitored equipment.

Be careful when identifying phases and make sure each current sensor is mounted onto the matching cable.

Please refer to paragraph 3.1.5 for detailed installation procedure of current sensors.

3.1.5 Current sensors installation

This paragraph covers current sensors installation in detail. Please refer to the paragraph matching your actual current sensor model.

All current sensors must be properly installed for optimal precision.

Always choose current sensors matching cable size and maximum current. Current transformers can be damaged if exposed to currents above their maximum rating.

All current sensor types have a mounting direction which must be respected for proper measurement.

3.1.5.1 Split-core current sensors LEM TT 50-SD, TT 100-SD, TOP90 or similar

This kind of current sensor is based on split-core current transformers. They can be opened for installation with the following instructions:

1. Open the sensor,

2. Identify mounting direction. **All sensors have an arrow which must point to the load when installed (see illustration below),**

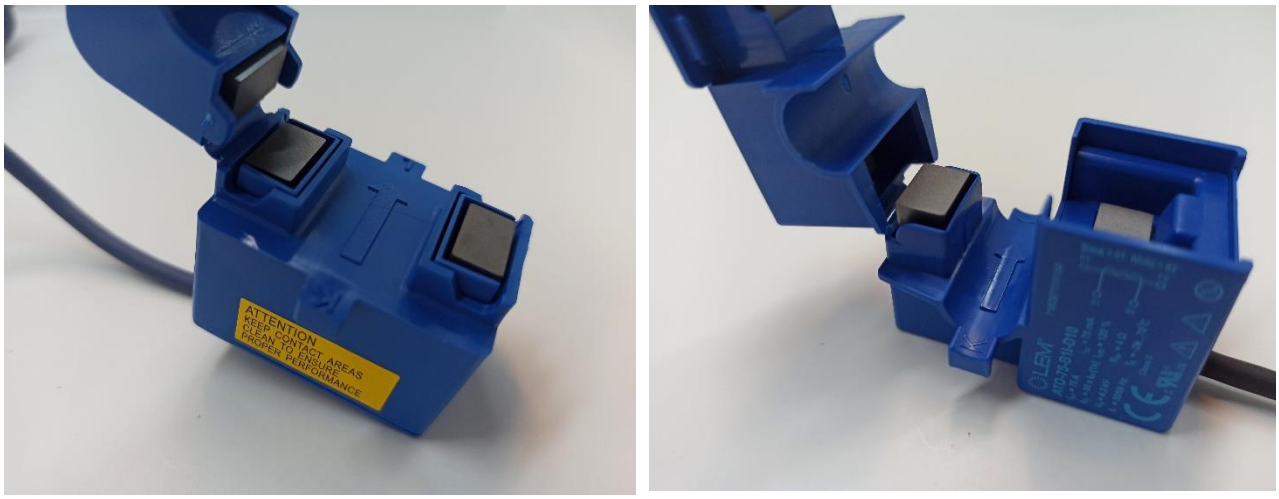

Figure 11 - Split-core current sensors orientation indication

3. Place sensor around the cable and close the sensor. You should hear a click,

4. Check if sensor is correctly closed by pulling slightly the top part,

5. If possible, secure the sensor onto the cable using cable ties,

6. Always secure sensor cables properly to avoid damage to the wires.

3.1.5.2 Rogowski coil current sensors LEM ART-B22-D70 or ART-B22-D125

This kind of current sensor is based on Rogowski coils. They can be opened for installation and have the advantage of having greater mechanical flexibility, allowing easy installation onto bus bar for example.

A few precautions must be taken with this kind of sensor:

1. Cables going to Spoony are coaxial cables and must be handled properly. Do not bend them, do not pull them,

2. Sensors must be secured properly onto the primary conductor using cable ties,

3. Measurement loop (flexible part going around the primary conductor) should be kept as far as possible from conductors other than the primary conductor. If unavoidable, limit the contact area with adjacent conductors as much as possible,

4. Keep measurement loop perpendicular to the primary conductor as much as possible and avoid distorting the measurement loop. Ideally the measurement loop should be a perfect circle,

5. As with current transformer, mounting direction is indicated by and arrow onto the body of the sensor. The arrow must point towards the load when installed.

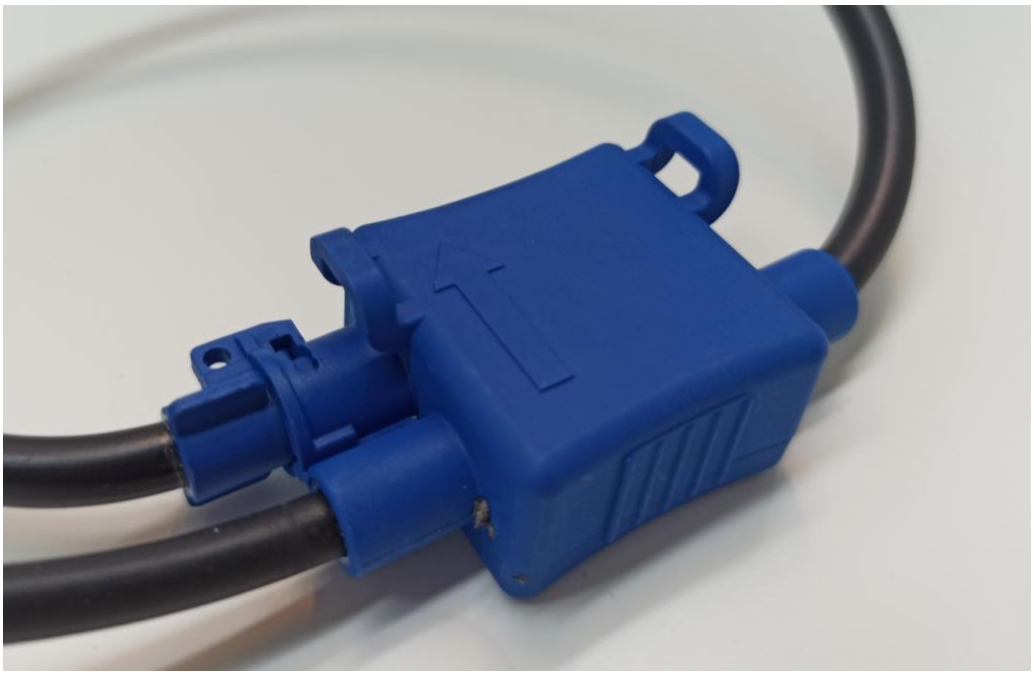

Figure 12 - Detailed view of Rogowski coils orientation indication

Refer to constructor's specific datasheet for more information on how to handle and install Rogowski coils.

3.1.6 Typical installation schematic, single-phase power measurement

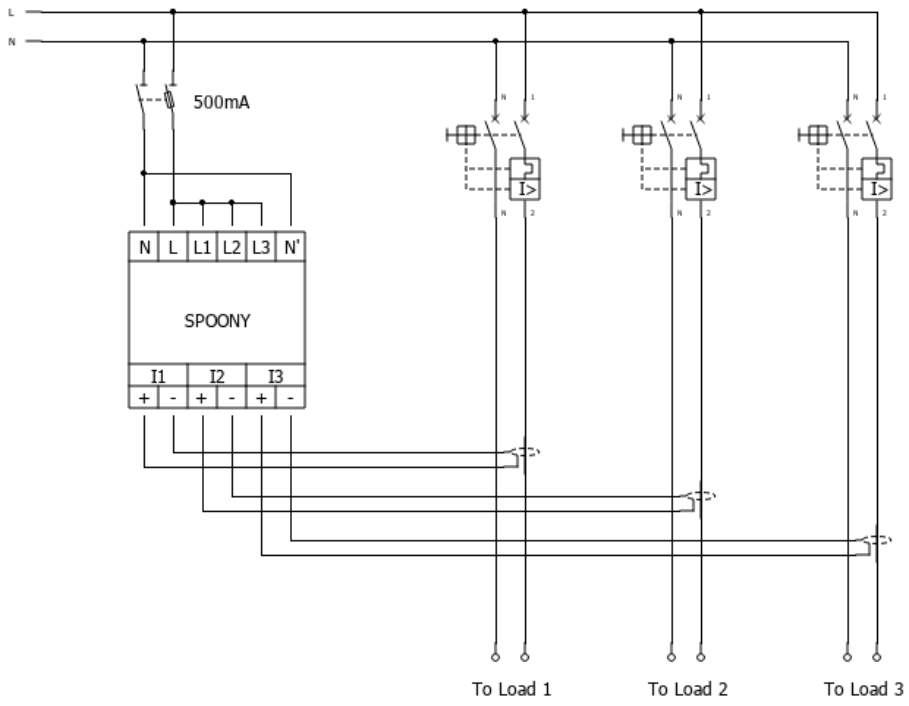

Figure 13: Typical single-phase wiring

3.1.7 Typical installation schematic, three-phase power measurement with neutral

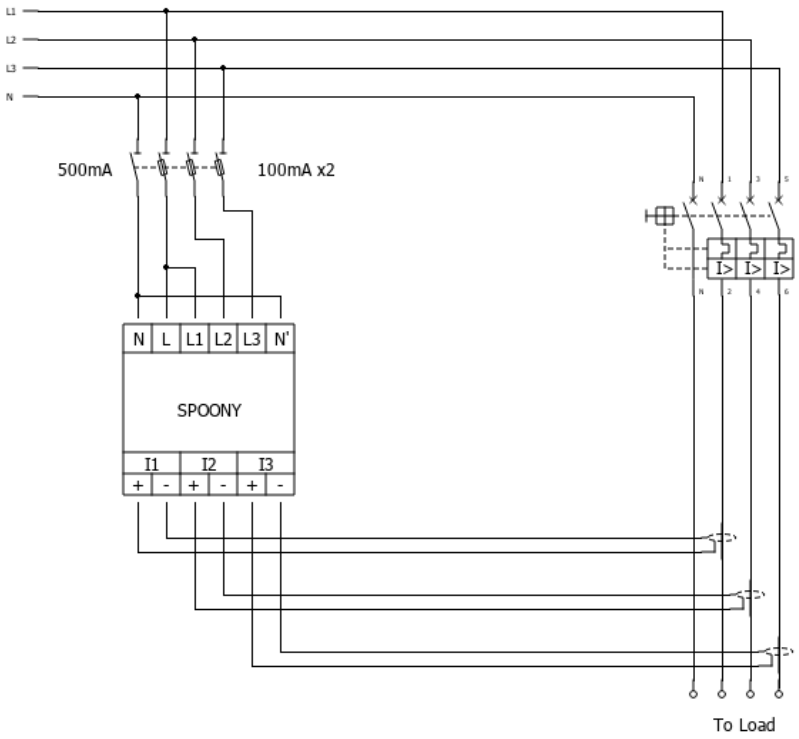

Figure 14: Typical three-phase wiring with neutral

3.1.8 Typical installation schematic, three-phase power measurement without neutral, separate line + neutral available for power supply

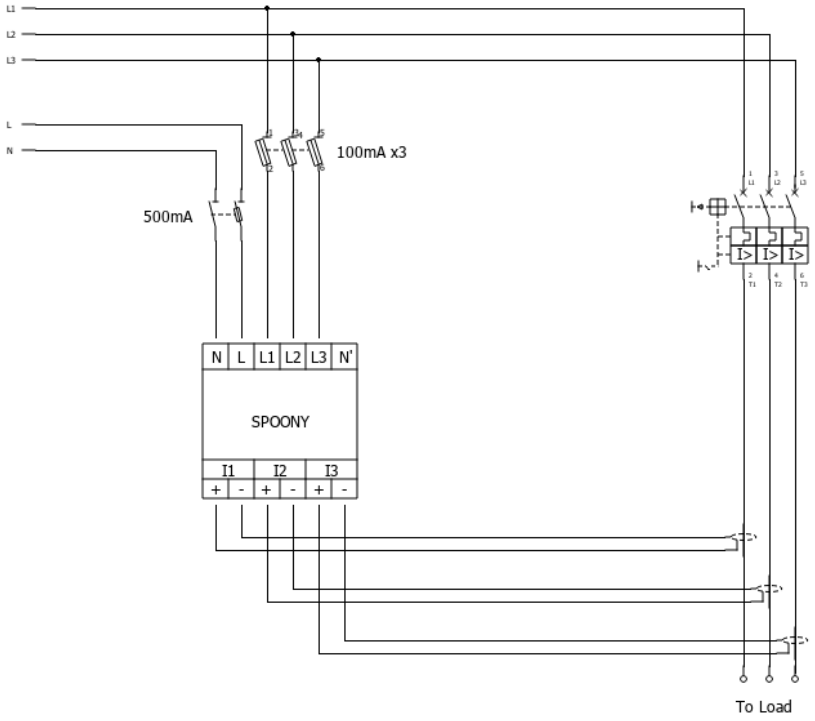

Figure 15: Typical three-phase wiring without neutral

3.2 Network connection

3.2.1 Ethernet network connection

The device can be connected to your Local Area Network (LAN) through an ethernet connection.

RJ45 connector is situated on top of the device.

The device is equipped with a 10Base-T, 100Base-TX PHY interface with auto-negotiation and auto-MDIX enabled.

Please note that the device may be used in offline mode, which does not require network connection. However, in this case, the configuration will only be possible by directly accessing the microSD card inside the device. Refer to chapter 0 for details on device configuration.

3.2.2 Wi-Fi network connection

Network connection can also be established through Wi-Fi (in option).

When installed, the Wi-Fi extension provides Wi-Fi access point (Wi-Fi AP) and station (Wi-Fi STA) functionality.

Refer to paragraph 4.3 for details.

3.3 RS485 (MODBUS) connection

Device is equipped with an RS485 bus allowing MODBUS client (master) or server (slave) functionality.

RS485 bus wiring must be done according to EIA-485. This includes usage of proper termination resistor and baud rate selection according to bus length and load.

Spoony provides a 3-terminal connector allowing wiring to the RS485 bus. Detailed view of this connector is shown below:

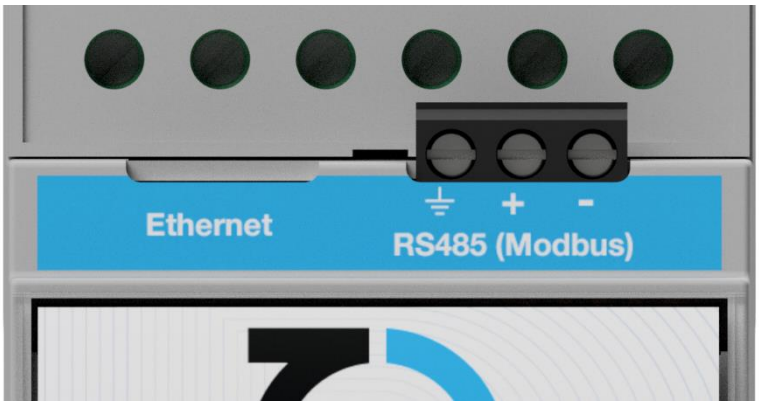

Figure 16 - RS485 connector detailed view

Desired MODBUS mode (client or server) must also be configured. Refer to section 4.6 for details.

3.4 TIC connection

Device is equipped with a TIC demodulator input. This input allows connection to most French electricity provider's energy meters, including Linky and older generations of meters.

For proper functionality, the monitored meter must be configured in Teleinformation mode. Wiring should be realized using twisted-pair cable between the meter and the Spoony as shown onto the diagram below:

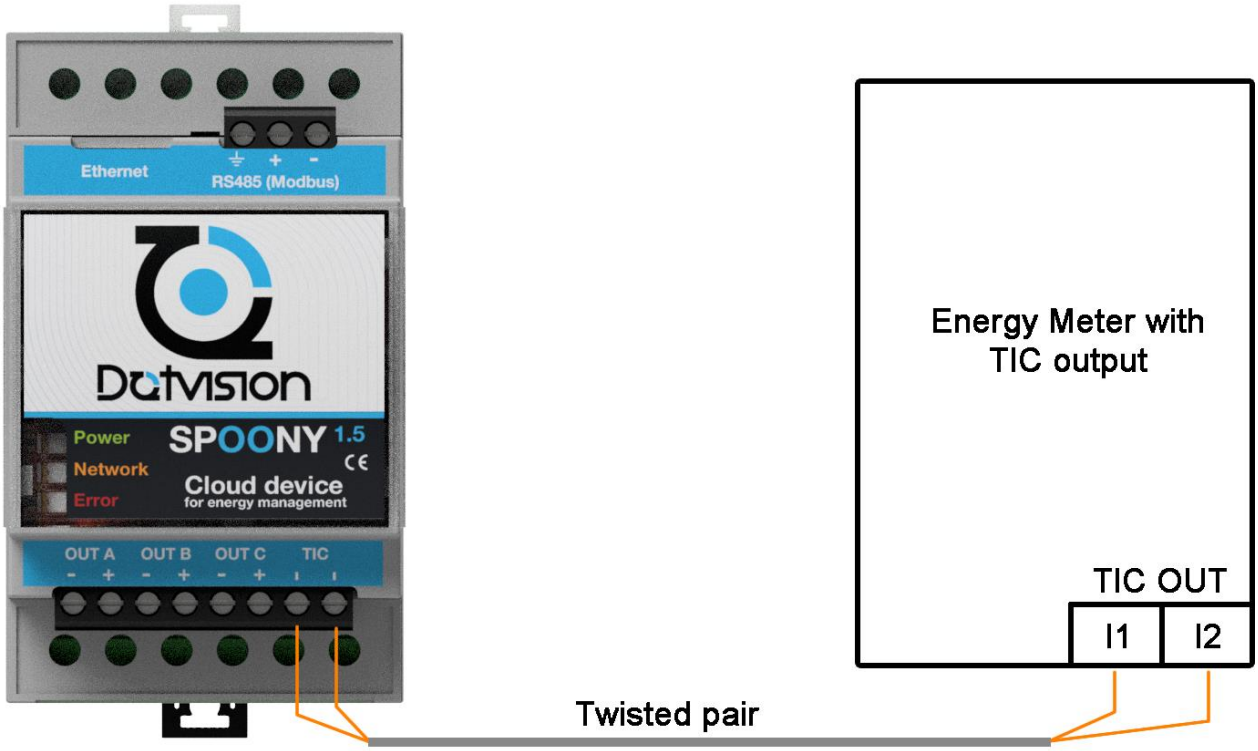

Figure 17 – TIC input wiring

Note: TIC signal has no polarity requirement

TIC functionality must then be configured and enabled in *Automation* Service configuration. Refer to section 0 for details.

3.5 Relays connection

Device is equipped with 3 open-drain outputs with 12V/250mA driving capacity, as well as 3 12V output terminals with a total output capacity of 500mA.

These outputs (OUT A, OUT B and OUTC) can be used to drive relays or other compatible equipment, thus adding control capabilities to the device.

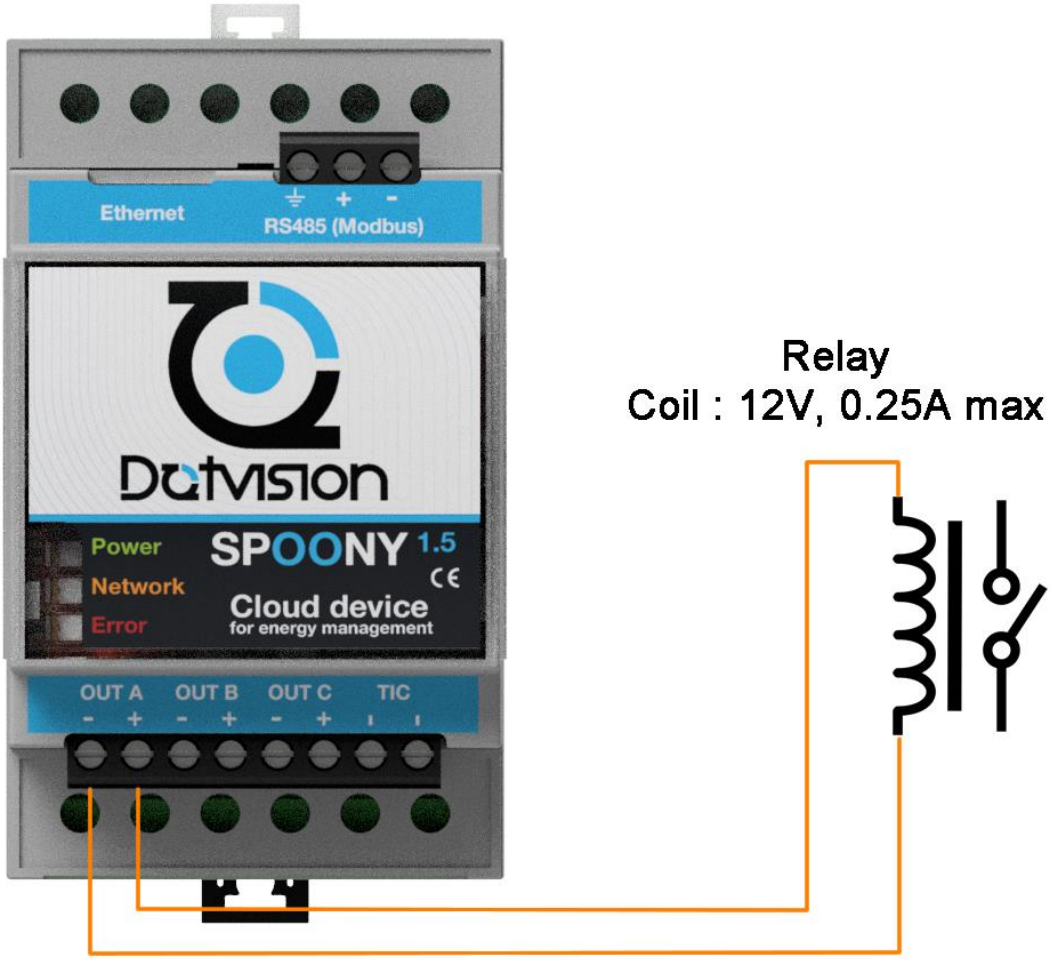

Figure 18 - Relay wiring

4 Functionalities and configuration

4.1 Generalities and requirements

Spoony configuration is customized through XML files residing into the microSD card accessible from the device front panel. See paragraph 2.4.2 for details on how to access the microSD card.

These files can be modified in two ways:

- Removing the microSD card from the device and inserting it into a computer with suitable microSD reader, or SD reader and microSD adapter,
- Once device is accessible over the network, files can be modified from the Web interface directly. See paragraph 5.10 for details on how to modify device configuration through the Web interface.

To edit XML files onto a computer, we recommend the use of Notepad++ software.

Device's microSD card contains the following directories:

- *certificates*: certificate files approved for TLS connections,
- *db*: local storage directory, files in this directory should not be modified,
- *firmware*: firmware directory. New device firmware must be placed into this directory to update the device to a new software version. See paragraph 4.12 for details on firmware update procedure,
- *logs*: device logs directory. This directory is mainly for use by DotVision to identify device issues,
- *measurements*: local storage directory containing CSV files when CSV data storage is enabled,
- *system*: device configuration resides within this directory,
- *temp*: temporary storage,
- *usr*: temporary storage,
- *www*: Web interface content directory.

In following paragraphs, all file paths are given from microSD card's root path.

Spoony functioning is based on multiple Services, each assuring a core functionality of the device. *Device* Service (main Service) is responsible for determining which Services to run at startup. Each Service has its own configuration directory, containing at least a *serviceSettings.xml* file.

For example, file *system/services/**Device**/serviceSettings.xml* contains *Device* Service configuration, while *system/services/**Device/Automation**/serviceSettings.xml* contains *Automation* Service configuration, which is a sub-service of *Device* Service.

All Services within the device are sub-Services of the *Device* Service, thus all configuration files are contained under the *system/services/Device* directory.

Services tree is best described by the following list:

- **Device**: main Service, holds all device Services,
  - **Automation**: *Automation* Service, provide device measurements (datapoints),
  - **Console**: console Service, provide local and remote management functionalities,
  - **HttpServer**: HTTP Web server Service,
  - **JsonRpc**: JSON-RPC Service, providing remote procedure call,
  - **ModbusSerial**: MODBUS RS485 server functionality,
  - **ModbusTCP**: MODBUS TCP server functionality,
  - **MqttClient**: MQTT client functionality,
  - **NetworkManager**: network management and configuration,
  - **Ntp**: Network Time Protocol time synchronization Service,
  - **ODM**: data sampling and sending/storing, rely on two sub-Services:
    - **ODMDataChunk**: sampling Service,
    - **ODMWebPush**: storing or sending Service,
  - **TimeService**: time management Service.

The following paragraphs describe configuration options for each Service.

Default configuration shown here may differ on your specific device if you requested pre-configuration by DotVision according to your requirements.

4.2 Device Service

Device Service is the main Service of the device. This is where device global configuration is defined, and where other Services are enabled/disabled.

Configuration file: *system/services/Device/serviceSettings.xml*

Configuration file organization:

```
<configuration version="1.0">
	<settings>
		Root settings - Do not modify
	</settings>

	<device>
		Device options
	</device>

	<services>
		Services list and start mode
	</services>
</configuration>
```

Figure 19 - Device Service configuration file organization

4.2.1 Device options

```
<device>
      <serialNumber>123456789</serialNumber>
      <location>0.0,0.0,0.0</location>
      <friendlyName>MyLittleSpoony</friendlyName>     <!-- Device name -->
      <firmwareVersion>1.2</firmwareVersion>
      <model>
            <manufacturer>DotVision</manufacturer>
            <manufacturerUrl>dotvision.com</manufacturerUrl>
            <name>Spoony</name>
            <number>1.5</number>
            <url/>
            <presentationUrl/>
      </model>
</device>
```

Figure 20 - Device Service – Device options

Device characteristics can be configured from this section. Most of these characteristics are for local display onto the Web interface only. Only option ***friendlyName*** must be configured as this is the main device identifier used by other Services.

In particular, this is the device identifier sent to the remote server when *ODM* Service is configured for data sending, and it is also the prefix used for CSV files when *ODM* Service is configured for CSV data storage.

4.2.2 Services list

Services list contains a few settings which should not be modified. Settings which can be modified are indicated in yellow below:

```
<services>
            <service>
                  <type>ConsoleService_t</type>
                  <name>Console</name>
                  <startMode group="2">automatic</startMode>
            </service>

            <service>
                  <type>NtpClient_t</type>
                  <name>Ntp</name>
                  <startMode group="2">automatic</startMode>
            </service>

            <service>
                  <type>TimeService_t</type>
                  <name>TimeService</name>
                  <startMode group="0">automatic</startMode>
            </service>

            <service>
                  <type>NetworkManager_t</type>
                  <name>NetworkManager</name>
                  <startMode group="1">automatic</startMode>
            </service>

            <service>
                  <type>JsonRpcService_t</type>
                  <name>JsonRpc</name>
                  <startMode group="0">automatic</startMode>
            </service>

            <service>
```

```
            <type>AutomationService_t</type>
            <name>Automation</name>
            <startMode group="0">automatic</startMode>
        </service>

        <service>
            <type>HttpServer_t</type>
            <name>HTTPServer</name>
            <startMode group="2">automatic</startMode>
        </service>

        <service>
            <type>ODMService_t</type>
            <name>ODM</name>
            <startMode group="3">manual</startMode>
                ➔ Set to automatic to enable ODM service
        </service>

        <service>
            <type>ModbusServerService_t</type>
            <name>ModbusSerial</name>
            <startMode group="3">manual</startMode>
                ➔ Set to automatic to enable Modbus RS485 Slave
        </service>

        <service>
            <type>ModbusServerService_t</type>
            <name>ModbusTCP</name>
            <startMode group="3">manual</startMode>
                ➔ Set to automatic to enable Modbus TCP Slave
        </service>
    </services>
```

Figure 21 - Device Service – Services List

4.3 Network

Device network configuration is done from the *NetworkManager* Service configuration file

Configuration file: system/services/Device/NetworkManager/serviceSettings.xml

Configuration file organization:

```
<NetworkManager version="1.0">
    <interfaces>
        <interface type="ethernet" version="1.0">
            <settings>
                Ethernet interface settings
            </settings>
        </interface>
        <interface type="wifi" version="1.0">
            <settings>
                Wifi STA interface settings
            </settings>
        </interface>
        <interface type="wifi-ap" version="1.0">
            <settings>
                Wifi AP interface settings
            </settings>
        </interface>
    </interfaces>
</NetworkManager>
```

Figure 22 - NetworkManager configuration file organization

Note: wifi and wifi-ap sections may not be present in the configuration file if your device is not equipped with Wi-Fi extension.

4.3.1 Ethernet network configuration

Device is configured in DHCP mode by default. In this mode, IP address, Network Mask, Gateway address and DNS servers address are obtained automatically from a DHCP server. This mode is enabled as shown below:

```
<interface type="ethernet" version="1.0">
      <settings>
            <add key="dhcp" value="true"/>
      </settings>
</interface>
```

Figure 23 - Ethernet interface DHCP configuration

To configure interface in static IP mode, the following configuration sample must be used and completed according to your network requirements:

```
<interface type="ethernet" version="1.0">
      <settings>
            <add key="dhcp" value="false"/>
            <add key="ip" value="X.X.X.X"/>
            <add key="gateway" value="X.X.X.X"/>
            <add key="mask" value="X.X.X.X"/>
            <add key="dns0" value="X.X.X.X"/>
            <add key="dns1" value="X.X.X.X"/>
      </settings>
</interface>
```

Figure 24 - Ethernet interface static IP configuration

Please note that the device supports only IPv4 functionality at this time.

DNS servers can be omitted if your device is not using dynamic name resolution (e.g., when device is only performing local storage or sending data to a server identified directly by its IP address).

4.3.2 Wi-Fi network configuration

This section is only applicable to devices with Wi-Fi option.

4.3.2.1 Wi-Fi access point

Device has a built-in Wi-Fi AP functionality which works straight out of the box. After power-up, the device will bring a dedicated Wi-Fi AP up.

By connecting onto this access point, you should be directly redirected to the device's Web interface home page. If it is not the case, open a Web browser and access the following address:

http://me.iofdevices.com

The Wi-Fi AP functionality is using the following network settings by default. If this configuration is incompatible with your other network interfaces (e.g., using the same subnet),

you can modify those default settings from the *NetworkManager* configuration relevant section shown below:

```
<interface type="wifi-ap" version="1.0">
      <settings>
            <add key="ssid" value="Spoony"/>
            <add key="ssid-postfix" value="true"/>
            <add key="channel" value="1"/>
            <add key="security" value="open"/>
          ➔ Set to wpa2-psk and set desired passphrase bellow to
            enable Wi-Fi security
            <add key="passphrase" value="PASSPHRASE"/>
            <add key="dhcp" value="false"/>
            <add key="ip" value="192.168.42.1"/>
            <add key="gateway" value="192.168.42.1"/>
            <add key="mask" value="255.255.255.0"/>
      </settings>
</interface>
```

Figure 25 - Wi-Fi AP interface configuration

You can also activate Wi-Fi security onto the access point by configuring a passphrase as described. Note that passphrase MUST be at least 8 characters long.

Wi-Fi AP can be completely disabled by removing the whole configuration section shown above.

4.3.2.2 Wi-Fi station configuration through Wi-Fi access point

Connection to your LAN through the Wi-Fi interface can be configured directly from the device's Web interface, accessed through the Wi-Fi AP, especially if your LAN is using DHCP.

In this case, please refer to paragraph 5.10.1.1 for detailed procedure regarding Wi-Fi configuration. If you do not have DHCP functionality on your Wi-Fi network, we recommend using manual configuration as show in next paragraph.

4.3.2.3 Wi-Fi station configuration through configuration file

Wi-Fi STA configuration is made from the following section of the *NetworkManager* configuration file as shown below:

```
<interface type="wifi" version="1.0">
      <settings>
            <add key="network" value="ACCESS POINT NAME"/>
            <add key="passphrase" value="PASSPHRASE"/>
            <add key="dhcp" value="true"/>
      </settings>
</interface>
```

Figure 26 - Wi-Fi station configuration in DHCP mode

Indicate you Wi-Fi AP name and passphrase as required. For connecting to unsecured Wi-Fi network, passphrase can be left empty.

If your network does not provide DHCP functionality, or if you want to set static IP settings, set *dhcp* setting to false and indicate your network specific settings as shown below:

```
<interface type="wifi" version="1.0">
      <settings>
            <add key="network" value="ACCESS POINT NAME"/>
            <add key="passphrase" value="PASSPHRASE"/>
            <add key="dhcp" value="false"/>
            <add key="ip" value="X.X.X.X"/>
            <add key="gateway" value="X.X.X.X"/>
            <add key="mask" value="X.X.X.X"/>
            <add key="dns0" value="X.X.X.X"/>
            <add key="dns1" value="X.X.X.X"/>
      </settings>
</interface>
```

Figure 27 - Wi-Fi station configuration in static IP mode

4.4 NTP Service configuration

NTP (Network Time Protocol) Service provide time synchronization functionality. When enabled (default), your device will perform NTP requests on a regular basis to resynchronize its system clock.

Configuration file: system/services/Device/Ntp/serviceSettings.xml

Configuration file organization:

```
<NtpClient version="1.0">
      <settings>
            <add key="Period" value="1800"/>
                  ➔ Period in seconds. Default to 30 min
            <add key="Server" value="0.fr.pool.ntp.org"/>
            <add key="Server" value="1.fr.pool.ntp.org"/>
            <add key="Server" value="2.fr.pool.ntp.org"/>
                  ➔ Set up to three servers. Ip address or Hostname
      </settings>
</NtpClient>
```

Figure 28 - Ntp Service configuration

4.5 Automation Service configuration

4.5.1 Automation Service overview

Automation Service is responsible for holding and organizing device measurements, as well as measurements from external devices read through MODBUS client functionality.

Configuration File: *system/services/Device/Automation/serviceSettings.xml*

Configuration File organization:

```
<AutomationService version="1.0">
      <settings>
      </settings>

      <automationSpace>
            <transports>
                  Transports (RS485/TCP) configuration for Modbus Client
            </transports>

            <units>
                  Automation Units list and configuration
            </units>

            <indexes>
                  Automation Index description
            </indexes>
      </automationSpace>
</AutomationService>
```

Figure 29 - Automation service configuration file organization

All measurements (values) available through Spoony are referred to as DataPoints. A DataPoint is a primitive object containing a Value, an Index, and a Timestamp.

Automation Service relies on two base elements: *Automation* Units and *Automation* Index.

An *Automation* Unit is a base element providing a list of DataPoints. Multiple Units can be configured and declared as shown in the following paragraphs.

The *Automation* Index is a tree of aliases used to organize datapoints from the different Units inside a single arborescence. Its syntax is described in paragraph 4.5.7.

Spoony always contains two base *Automation* Units:

- ade7758: this Unit provides energy measurement from the internal ADE7758 energy metering IC,
- outputs: this Unit provides relays outputs binary DataPoints to allow writing outputs states.

Other *Automation* Units are also available in option:

- tic: this Unit provides DataPoints read from the TIC input wired to an external energy meter (see paragraph 3.4 for details regarding the TIC input),
- soft: this Unit provides DataPoints computed in real-time from other available DataPoints and can perform averaging, minimum, and maximum computing,
- modbus: this Unit provides DataPoints read from external devices through MODBUS protocol (RS485 or TCP).

Configuration options of each Unit type is detailed in following paragraphs.

4.5.2 ADE7758 Automation Unit

This Unit provides measurements taken from the internal ADE7758 energy metering IC. This Unit does not accept any setting.

The Unit must be declared as follow in the *<units>* section:

```
<unit name="ade7758" type="TestUnit_t">
    <settings/>
</unit>
```

Figure 30 - ADE7758 Automation Unit configuration

The ADE7758 metering IC calibration can be modified from the Web interface only. Please refer to paragraph 5.10.6 for details on how to modify device calibration.

Once enabled, this Unit will expose the following DataPoints:

Path	Description	Unit
units/ade7758/FREQ	Line frequency, measured on L1	Hz
units/ade7758/IRMSA	RMS Current phase L1	A
units/ade7758/IRMSB	RMS Current phase L2	A
units/ade7758/IRMSC	RMS Current phase L3	A
units/ade7758/VRMSA	RMS Voltage phase L1	V
units/ade7758/VRMSB	RMS Voltage phase L2	V
units/ade7758/VRMSC	RMS Voltage phase L3	V
units/ade7758/WATTA	Active Power phase L1	W
units/ade7758/WATTB	Active Power phase L2	W
units/ade7758/WATTC	Active Power phase L3	W
units/ade7758/VARA	Reactive Power phase L1	VAR
units/ade7758/VARB	Reactive Power phase L2	VAR
units/ade7758/VARC	Reactive Power phase L3	VAR
units/ade7758/VAA	Apparent Power phase L1	VA
units/ade7758/VAB	Apparent Power phase L2	VA
units/ade7758/VAC	Apparent Power phase L3	VA
units/ade7758/PFA	Power Factor phase L1	
units/ade7758/PFB	Power Factor phase L2	
units/ade7758/PFC	Power Factor phase L3	
units/ade7758/WATTHRA	Active Energy phase L1	Wh
units/ade7758/WATTHRB	Active Energy phase L2	Wh
units/ade7758/WATTHRC	Active Energy phase L3	Wh
units/ade7758/VARHRA	Reactive Energy phase L1	VARh
units/ade7758/VARHRB	Reactive Energy phase L2	VARh
units/ade7758/VARHRC	Reactive Energy phase L3	VARh
units/ade7758/VAHRA	Apparent Energy phase L1	VAh
units/ade7758/VAHRB	Apparent Energy phase L2	VAh
units/ade7758/VAHRC	Apparent Energy phase L3	VAh
units/ade7758/WATT	Total Active Power	W
units/ade7758/VAR	Total Reactive Power	VAR
units/ade7758/VA	Total Apparent Power	VA
units/ade7758/WATTHR	Total Active Energy	Wh
units/ade7758/VARHR	Total Reactive Energy	VARh
units/ade7758/VAHR	Total Apparent Energy	VAh

Figure 31 - ADE7758 Automation Unit available DataPoints

4.5.3 Outputs Automation Unit

This Unit provide access to the three open-drain outputs of the device through binary DataPoints. This Unit does not accept any setting.

The Unit must be declared as follow in the *<units>* section:

```
<unit name="outputs" type="OutputsUnit_t">
      <settings/>
</unit>
```

Once enabled, this Unit will expose the following DataPoints:

Path	Description	Unit
units/outputs/RELAYA	Output OUTA status	Hz
units/outputs/RELAYB	Output OUTB status	A
units/outputs/RELAYC	Output OUTC status	A

Figure 32 - Output Automation Unit available DataPoints

4.5.4 TIC Automation Unit

This Unit provide access to data red through the TIC interface of the device. TIC bus uses a serial protocol where data is sent in ASCII format. Please refer to relevant documentation for details regarding the protocol.

TIC *Automation* Unit requires a few settings as shown below:

```
<unit name="tic" type="TICUnit_t">
      <settings>
            <add key="linky" value="false"/>
                  ➔ Set to true if TIC input is wired to a Linky meter to
                    adjust communication settings accordingly
            <datapoint name="ADCO" tag="ADCO" type="uint64"/>
            <datapoint name="OPTARIF" tag="OPTARIF" type="enum"
values="BASE;HC;EJP"/>
            <datapoint name="BASE" tag="BASE" type="uint32"/>
            <datapoint name="ISOUSC" tag="ISOUSC" type="uint32"/>
            <datapoint name="HCHC" tag="HCHC" type="uint32"/>
            <datapoint name="HCHP" tag="HCHP" type="uint32"/>
            <datapoint name="PTEC" tag="PTEC" type="enum"
values="TH;HC;HP;HN;PM"/>
            <datapoint name="IINST" tag="IINST" type="uint32"/>
            <datapoint name="IINST1" tag="IINST1" type="uint32"/>
            <datapoint name="IINST2" tag="IINST2" type="uint32"/>
            <datapoint name="IINST3" tag="IINST3" type="uint32"/>
            <datapoint name="PAPP" tag="PAPP" type="uint32"/>
                  ➔ List of TAGS to extract from the incoming TIC frames
      </settings>
</unit>
```

Figure 33 - TIC Automation Unit Configuration

The list of DataPoints to be extracted from the incoming TIC frames must list all values which should be read. Values not appearing in this list will be ignored.

For each DataPoint, you must specify:

- name: the local name of the DataPoint,
- tag: the matching TAG in incoming TIC frames,
- type: the type of data. Only uint64, uint32 and enum are supported,
- values: only if type is enum, list the possible values in the incoming frames. The effective DataPoint value will be the index of the read value in this list, or -1 if the received value does not exist in the list.

Once enabled, the exposed DataPoint list will depend on DataPoint configuration of the Unit. For the above example:

Path	Description	Unit
units/tic/ADCO	ADCO	
units/tic/OPTARIF	OPTARIF	
units/outputs/BASE	BASE	

…

4.5.5 Soft Automation Unit

This Unit provides computing functionalities, allowing to perform arithmetic operations between DataPoints, as well as averaging and minimum/maximum values computing over configurable periods.

Sample configuration:

```
<unit name="soft" type="SoftAutomationUnit_t">
      <settings>
            <add key="period" value="1000"/>
                  ➔ Computing period, in ms. Frequency at which the
                     computing`operations are performed
            <add key="offset" value="100"/>
                  ➔ Computing period offset, in ms. Wait 100ms after the
                     round period for all input datapoints to be updated
                     before performing computation
      </settings>
      <datapoints>
            <datapoint name="AVG_IRMSA" value="DP(units/ade7758/IRMSA)"
filter="avg" samples="6" unit="A"/>
                  ➔ Compute average value of IRMSA, over 6 samples. Output
                     value is updated every 6 seconds in this case
            <datapoint name="MIN_IRMSA" value="DP(units/ade7758/IRMSA)"
filter="min" samples="6" unit="A"/>
                  ➔ Same as previous, but compute minimum value
            <datapoint name="MAX_IRMSA" value="DP(units/ade7758/IRMSA)"
filter="max" samples="6" unit="A"/>
                  ➔ Same as previous, but compute maximum value
            <datapoint name="WATT" value="DP(units/ade7758/WATT)/1000"
filter="avg" samples="1" unit="kW"/>
                  ➔ Convert W to kW (divide by 1000) and perform average
                     value computation over 6 seconds
      </datapoints>
</unit>
```

Figure 34 - Soft Automation Unit configuration

Unit settings are:

- period: the period, in ms, between two refresh of output values. The computing algorithm is performed once every *period* ms over the whole list of DataPoints. Most DataPoints are updated every 1s, so setting the period to 1s ensure all updates to DataPoints will be effectively reflected into computed output DataPoints,
- offset: the time in ms to wait, after the round period (e.g., computation will be performed at 00:00:00.1, 00:00:00.2 etc.) for the input DataPoints to be up to date. Recommended setting for this value is about 1/10th of the period.

For each DataPoint entry, the following parameters are available:

- name: Unit datapoint name. Output DataPoint will be available as *units/soft/**name,***
- value: arithmetic expression. Allows to take another DataPoint value (from any other unit) with *DP({Datapoint path})* expression.,
- filter: type of filtering operation. Valid values are avg (average), min (minimum), max (maximum),
- sample: number of samples used for filtering operation. Setting samples to 1 effectively disables filtering as output value is updated at each sampling,
- unit: Output DataPoint Unit property.

4.5.6 Modbus master (client) Unit

This Unit allows to retrieve data from external devices into the Spoony. Once retrieved, those values become Spoony DataPoints which can be used for computation (Soft Unit) of data sending/storage through *ODM* Service.

Warning: When the RS485 bus is used for MODBUS master functionality as described below, it cannot be used for MODBUS slave and the ModbusSerial (MODBUS slave) Service MUST be disabled for proper operation. MODBUS TCP functionality, however, can be simultaneously used for master and slave functions.

This paragraph covers Modbus master Unit configuration which is based on MODBUS Device Models and Update Policies files. Please refer to section 4.6 for details regarding Device Models and Update Policies files syntax and creation.

Modbus Unit requires configuration of the underlying transport first. This configuration entry goes under the <transports> configuration section and depends on the desired transport type (RS485 or TCP).

For RS485, the transport entry is configured as follows:

```
<transport name="comm" type="ModbusRS485Transport_t">
      <settings>
            <add key="driver" value="drv/rs485"/>
            <add key="mode" value="rtu"/>
                  ➔ rtu or ascii depending on Modbus protocol type
            <add key="baudrate" value="38400"/>
                  ➔ RS485 bus baudrate
            <add key="stopbits" value="0"/>
                  ➔ Stop Bits, 0 (1 stop bit) or 1 (2 stop bits)
            <add key="databits" value="0"/>
                  ➔ Data bits, 0 (8 databits) or 1 (9 databits)
            <add key="parity" value="0"/>
                  ➔ Parity, 0 (none), 1 (odd) or 2 (even)
      </settings>
</transport>
```

Figure 35 - Modbus RS485 Transport configuration

Note: Only a single RS485 transport can be declared.

For TCP, the transport entry is configured as follows:

```
<transport name="comm" type="ModbusTcpTransport_t">
                  ➔ If multiple transports are declared, each transport
                     MUST have a unique name
      <settings>
            <add key="host" value="tcp://HOST:PORT"/>
                  ➔ Indicate target host and port here
      </settings>
</transport>
```

Figure 36 - Modbus TCP Transport configuration

Once the transport section is correctly configured, the Modbus Unit can be configured and linked to the underlying transport section.

A single Modbus Unit communicates with multiple slaves and is configured as follows:

```
<unit name="modbus" type="ModbusUnit_t">
                  ➔ If multiple units are declared, each unit MUST have a
                     unique name
      <settings>
            <add key="transport" value="comm"/>
                  ➔ Indicate which transport should be used. If multiple
                     transport are declared, they MUST have a unique name
      </settings>
      <slaves>
            <slave>
                  <add key="destination" value="4"/>
                     ➔ Modbus destination (address)
                  <add key="policies" value="models/i60_policies.xml"/>
                     ➔ Update policies. Path is relative to current directory
                  <add key="model" value="models/i60.xml"/>
                     ➔ Device model. Path is relative to current directory
                  <add key="swap" value="none"/>
                     ➔ Default swap mode for multi-register values. Setting
                        swap mode here affect the whole slave. Swap mode can
                        also be configured inside device model for per-
                        register or per-section swap mode. Valid values are
                        none, bw (bytes and words), b (bytes), w (words)
            </slave>
            <slave>
                  Similar to previous slave description. Only destination must
change. Policies and model may differ
```

```
            </slave>
        </slaves>
</unit>
```

Figure 37 - Modbus Unit configuration

In the case of TCP transport, each Modbus Unit can communicate with a single host. If you need to communicate with different hosts, you will need to declare as many *Transports* and *Modbus Units* as required. The *destination* field should be set to 0 in most case for MODBUS TCP unless the target device has gateway functionalities.

Each declared *Transport* and *ModbusUnit* MUST have a unique name.

The DataPoints list exposed by the Unit will depend on the Unit configuration and Device Model.

Each Device Model declares a list of DataPoints identified by a unique name. Each Modbus Unit will expose those DataPoints prefixed with MODBUS destination (address) as follows:

units/{unit name}/{destination}-{name}

Figure 38 - Modbus Datapoint Name format

4.5.7 Automation Space Index

Automation Space *<index>* section allows to identify all DataPoints from the Units through a customized path (Alias).

Addressing of DataPoints can be made using either their default path (*units/{unit name}/{datapoint name}*) or any Alias path declared into the Index.

The Index is a multi-level tree allowing to organize DataPoints easily. Default *Automation* Space index is shown below. DataPoints from other Units can be added to the Index as required by the end user. The Index is mainly used to organize DataPoints browsing from the Web interface. It is defined in the following manner:

```
<indexes>
  <index name="main" default="yes">
    <node key="monitoring">
      <node key="temp" value="ade7758/TEMP" displayName="Temperature"/>
      <node key="freq" value="ade7758/FREQ"  displayName="Line frequency"/>
      <node key="var" value="soft/VAR"  displayName="Reactive Power"/>
      <node key="watt" value="soft/WATT"  displayName="Active Power"/>
      <node key="va" value="soft/VA"  displayName="Apparent Power"/>
      <node key="watthr" value="soft/WATTHR"/>
      <node key="varhr" value="soft/VARHR"/>
      <node key="vahr" value="soft/VAHR"/>

      <node key="input0">
        <node key="vrm" value="ade7758/VRMSA" displayName="L1 Voltage"/>
        <node key="irms" value="ade7758/IRMSA" displayName="L1 Current"/>
        <node key="watt" value="ade7758/WATTA" displayName="L1 Active Power"/>
        <node key="va" value="ade7758/VAA" displayName="L1 Apparent Power"/>
        <node key="var" value="ade7758/VARA" displayName="L1 Reactive Power"/>
        <node key="pf" value="ade7758/PFA" displayName="L1 Power Factor"/>
        <node key="watthr" value="ade7758/WATTHRA" displayName="L1 Active Energy"/>
        <node key="varhr" value="ade7758/VARHRA" displayName="L1 Reactive Energy"/>
        <node key="vahr" value="ade7758/VAHRA" displayName="L1 Apparent Energy"/>
```

```
      </node>

      <node key="input1">
        <node key="vrm" value="ade7758/VRMSB" displayName="L2 Voltage"/>
        <node key="irms" value="ade7758/IRMSB"  displayName="L2 Current"/>
        <node key="watt" value="ade7758/WATTB"  displayName="L2 Active Power"/>
        <node key="va" value="ade7758/VAB"  displayName="L2 Apparent Power"/>
        <node key="var" value="ade7758/VARB"  displayName="L2 Reactive Power"/>
        <node key="pf" value="ade7758/PFB"  displayName="L2 Power Factor"/>
        <node key="watthr" value="ade7758/WATTHRB"  displayName="L2 Active Energy"/>
        <node key="varhr" value="ade7758/VARHRB"  displayName="L2 Reactive Energy"/>
        <node key="vahr" value="ade7758/VAHRB"  displayName="L2 Apparent Energy"/>
      </node>

      <node key="input2">
        <node key="vrm" value="ade7758/VRMSC"  displayName="L3 Voltage"/>
        <node key="irms" value="ade7758/IRMSC"  displayName="L3 Current"/>
        <node key="watt" value="ade7758/WATTC"  displayName="L3 Active Power"/>
        <node key="va" value="ade7758/VAC"  displayName="L3 Apparent Power"/>
        <node key="var" value="ade7758/VARC"  displayName="L3 Reactive Power"/>
        <node key="pf" value="ade7758/PFC"  displayName="L3 Power Factor"/>
        <node key="watthr" value="ade7758/WATTHRC"  displayName="L3 Active Energy"/>
        <node key="varhr" value="ade7758/VARHRC" displayName="L3 Reactive Energy"/>
        <node key="vahr" value="ade7758/VAHRC" displayName="L3 Apparent Energy"/>
      </node>
    </node>
    <node key="control">
      <node key="output0" value="outputs/RELAYA" displayName="Output A"/>
      <node key="output1" value="outputs/RELAYB" displayName="Output B"/>
      <node key="output2" value="outputs/RELAYC" displayName="Output C"/>
    </node>
  </index>
</indexes>
```

Figure 39 - Default Automation Space Index configuration

4.6 MODBUS configuration

This section covers in detail the different MODBUS functionalities of the device. This includes MODBUS master and slave functions, over RS485 and TCP.

4.6.1 Overview of MODBUS functionalities

Spoony offers two main modes of operation:

- **Modbus slave** (server): allows accessing the Spoony's DataPoints through MODBUS, using either RS485 or TCP transport,
- **Modbus master** (client): allows accessing data from external MODBUS compatible devices to perform data aggregation.

MODBUS master mode is configured within the *Automation* Service as described in paragraph 4.5.6. Paragraph 4.6.6 will cover Device Models and Update Policies configuration files.

4.6.2 MODBUS slave (server)

MODBUS slave function is available through RS485 and TCP. While both functionalities are covered by a different *Device* Service, those Services are very similar. This section covers similar portions of the two configuration files.

Configuration files: *system/services/Device/ModbusTCP/serviceSettings.xml and system/services/Device/ModbusSerial/serviceSettings.xml*

Configuration files organization:

```
<service version="1.0" mode="tcp">
                    ➔ mode is tcp or serial depending on transport type
      <settings>
            Transport-specific settings
                    ➔ Those settings will be detailed in later paragraphs
      </settings>

      <model>
            Device registers mapping
      </model>
</service>
```

Figure 40 - Modbus slave configuration file organization

The device's registers mapping *(<model>* section) is similar for both TCP and RS485 transport and defines the mapping between exposed MODBUS registers and *Automation* Space DataPoints.

All available DataPoints can be exposed through MODBUS slave, including DataPoints red from other MODBUS devices, and computed DataPoints, thus offering a lot of flexibility.

The <*model*> section is organized as follows:

```
<model>
  <holdingRegisters/>
  <inputRegisters>
    <add address="0" size="2" datapoint="alias/main/monitoring/freq"/>
    <add address="2" size="2" datapoint="alias/main/monitoring/input0/vrm"/>
    <add address="4" size="2" datapoint="alias/main/monitoring/input1/vrm"/>
    <add address="6" size="2" datapoint="alias/main/monitoring/input2/vrm"/>
    <add address="8" size="2" datapoint="alias/main/monitoring/input0/irms"/>
    <add address="10" size="2" datapoint="alias/main/monitoring/input1/irms"/>
    <add address="12" size="2" datapoint="alias/main/monitoring/input2/irms"/>
    <add address="14" size="2" datapoint="alias/main/monitoring/input0/watt"/>
    <add address="16" size="2" datapoint="alias/main/monitoring/input1/watt"/>
    <add address="18" size="2" datapoint="alias/main/monitoring/input2/watt"/>
    <add address="20" size="2" datapoint="alias/main/monitoring/watt"/>
    <add address="22" size="2" datapoint="alias/main/monitoring/input0/va"/>
    <add address="24" size="2" datapoint="alias/main/monitoring/input1/va"/>
    <add address="26" size="2" datapoint="alias/main/monitoring/input2/va"/>
    <add address="28" size="2" datapoint="alias/main/monitoring/va"/>
    <add address="30" size="2" datapoint="alias/main/monitoring/input0/var"/>
    <add address="32" size="2" datapoint="alias/main/monitoring/input1/var"/>
    <add address="34" size="2" datapoint="alias/main/monitoring/input2/var"/>
    <add address="36" size="2" datapoint="alias/main/monitoring/var"/>
    <add address="38" size="2" datapoint="alias/main/monitoring/input0/pf"/>
    <add address="40" size="2" datapoint="alias/main/monitoring/input1/pf"/>
    <add address="42" size="2" datapoint="alias/main/monitoring/input2/pf"/>
    <add address="44" size="2" datapoint="alias/main/monitoring/input0/watthr"/>
    <add address="45" size="2" datapoint="alias/main/monitoring/input1/watthr"/>
    <add address="48" size="2" datapoint="alias/main/monitoring/input2/watthr"/>
    <add address="50" size="2" datapoint="alias/main/monitoring/watthr"/>
    <add address="52" size="2" datapoint="alias/main/monitoring/input0/vahr"/>
    <add address="54" size="2" datapoint="alias/main/monitoring/input1/vahr"/>
    <add address="56" size="2" datapoint="alias/main/monitoring/input2/vahr"/>
    <add address="58" size="2" datapoint="alias/main/monitoring/vahr"/>
    <add address="60" size="2" datapoint="alias/main/monitoring/input0/varhr"/>
    <add address="62" size="2" datapoint="alias/main/monitoring/input1/varhr"/>
    <add address="64" size="2" datapoint="alias/main/monitoring/input2/varhr"/>
    <add address="66" size="2" datapoint="alias/main/monitoring/varhr"/>
  </inputRegisters>
  <coils>
    <add address="0" datapoint="units/outputs/RELAYA"/>
    <add address="1" datapoint="units/outputs/RELAYB"/>
    <add address="2" datapoint="units/outputs/RELAYC"/>
  </coils>
  <discreteInputs/>
</model>
```

Figure 41 - Default MODBUS Device Model configuration

The Model makes a distinction between *Holding registers* (Read/Write capability), *Input registers* (Read only), *Coil registers* (Read/Write binary) and *Discrete Inputs* (Read only binary).

Each entry of type Holding or Input has the following parameters:

- *address*: the MODBUS register address,
- *size*: the size, in number of consecutive 16 bits MODBUS registers. All Spoony DataPoints are floating-point type and are mapped onto two consecutive registers, except for binary DataPoints (Relays outputs),
- *datapoint*: path to the matching DataPoint, addressed by alias or by root address.

Each entry of type Coil or Discrete Input has the following parameters:

- *address*: the MODBUS coil/discrete input address,
- *datapoint*: path to the matching DataPoint, addressed by alias or by root address.

For compatibility reasons, while the MODBUS map may be modified by the user, DotVision recommends keeping the default MODBUS map as it is, and adding additional entries as

required, for exposing additional DataPoints or creating DataPoints group of interest on consecutive addresses for example.

4.6.3 MODBUS slave (server) RS485 specificities

MODBUS slave RS485 specific settings are described below. These settings only apply to *ModbusSerial* Service:

```
<settings>
  <add key="port" value="drv/rs485"/>
  <add key="baudrate" value="19200"/>
                        ➔ Baud rate
  <add key="parity" value="even"/>
                        ➔ Parity. Valid values are none, odd or even
  <add key="mode" value="RTU"/>
                        ➔ Modbus mode. RTU or ASCII
  <add key="address" value="130"/>
                        ➔ Modbus slave address
  <add key="swapMode" value="0"/>
                        ➔ Swap mode. 0 (none), 1 (byte and word), 2 (byte), 3
                          (word)
</settings>
```

Figure 42 - MODBUS slave RS485 specific configuration

4.6.4 MODBUS slave (server) TCP specificities

MODBUS slave TCP specific settings are described below. These settings only apply to *ModbusTCP* Service:

```
<settings>
  <add key="port" value="502"/>
                        ➔ TCP Port
  <add key="swapMode" value="0"/>
                        ➔ Swap mode. 0 (none), 1 (byte and word), 2 (byte), 3
                          (word)
</settings>
```

Figure 43 - MODBUS slave TCP specific configuration

4.6.5 Default MODBUS Table

This table describes the default MODBUS mapping of the device:

Input registers		
Register address	**Register name**	**Description**
0x0000	FREQ	Line Frequency (Hz)
0x0002	VRMSA	VRMS Phase L1 (Vrms)
0x0004	VRMSB	VRMS Phase L2 (Vrms)
0x0006	VRMSC	VRMS Phase L3 (Vrms)
0x0008	IRMSA	IRMS Phase L1 (Arms)
0x000A	IRMSB	IRMS Phase L2 (Arms)
0x000C	IRMSC	IRMS Phase L3 (Arms)
0x000E	WATTA	Active power Phase L1 (W)
0x0010	WATTB	Active power Phase L2 (W)
0x0012	WATTC	Active power Phase L3 (W)
0x0014	WATT	Total Active Power (W)
0x0016	VAA	Apparent power Phase L1 (VA)
0x0018	VAB	Apparent power Phase L2 (VA)
0x001A	VAC	Apparent power Phase L3 (VA)
0x001C	VA	Total Apparent Power (VA)
0x001E	VARA	Reactive power Phase L1 (VAR)
0x0020	VARB	Reactive power Phase L2 (VAR)
0x0022	VARC	Reactive power Phase L3 (VAR)
0x0024	VAR	Total Reactive Power (VAR)
0x0026	PFA	Power Factor Phase L1
0x0028	PFB	Power Factor Phase L2
0x002A	PFC	Power Factor Phase L3
0x002C	AWATTHR	Active Energy Phase L1 (WH)
0x002E	BWATTHR	Active Energy Phase L2 (WH)
0x0030	CWATTHR	Active Energy Phase L3 (WH)
0x0032	WATTHR	Total Positive Active Energy (WH)
0x0034	AVAHR	Apparent Energy Phase L1 (VARH)
0x0036	BVAHR	Apparent Energy Phase L2 (VARH)
0x0038	CVAHR	Apparent Energy Phase L3 (VARH)
0x003A	VAHR	Total Apparent Energy (VARH)
0x003C	AVARHR	Reactive Energy Phase L1 (WH)
0x003E	BVARHR	Reactive Energy Phase L2 (WH)
0x0040	CVARHR	Reactive Energy Phase L3 (WH)
0x0042	VARHR	Total Reactive Energy (WH)
Coil registers		
Register address	**Register name**	**Description**
0x0000	RELAYA	Open-drain output OUTA status
0x0001	RELAYB	Open-drain output OUTB status
0x0002	RELAYC	Open-drain output OUTC status

Figure 44 - Default MODBUS address map

4.6.6 MODBUS master (client) RS485 and TCP

Basic MODBUS master configuration procedure is described in paragraph 4.5.6. This paragraph describes format of Device Model and Update Policies files only. These files are referenced by Modbus *Automation* Units and describe the content of the external MODBUS devices.

4.6.6.1 Modbus Device Model configuration file

Model configuration files are an exhaustive list of the Modbus DataPoints which MUST be read from a MODBUS device. Only DataPoints described into the Model file will be read by the Modbus Unit.

The frequency at which those readings are done is determined by the Policies configuration file described in the next paragraph.

Modbus Device Model files are usually placed under the directory *system/services/Device/Automation/models* and have the following format:

```
<remoteModbusUnitModel name="myDevice" version="">
  <maps>
    List of Maps
  </maps>
</remoteModbusUnitModel>
```

Figure 45 - Modbus Master Device Model file organization

Remote DataPoints are organized in Maps, where a Map is a range of MODBUS registers which will be accessed through a single MODBUS Read access. This allows high flexibility but requires some precautions.

You must always avoid creating maps which covers large span of MODBUS addresses. For example, if you want to read registers 1 and 200 from device, create two separate maps. Otherwise, this would result in a single MODBUS Read access of 200 registers, causing high load on the bus.

It is however acceptable to have a map with holes of 2-3 ignored registers within it, as it is more efficient to read those registers and ignore the returned value (which is what Spoony does) than breaking the read operation in two separate MODBUS Read operations.

A single Map is defined in the following manner:

```
<map name="inst1" type="holdingRegister" swap="none" count="10">
  <datapoint address="18452" size="2" name="u12-1" format="uint32" swap="none" factor="0.01"/>
  <datapoint address="18454" size="2" name="u23-1" format="uint32" swap="none" factor="0.01"/>
  <datapoint address="18456" size="2" name="u31-1" format="uint32" swap="none" factor="0.01"/>
  <datapoint address="18458" size="2" name="i1-1" format="uint32" swap="none" factor="0.001"/>
  <datapoint address="18460" size="2" name="i2-1" format="uint32" swap="none" factor="0.001"/>
  <datapoint address="18462" size="2" name="i3-1" format="uint32" swap="none" factor="0.001"/>
  <datapoint address="18464" size="2" name="in-1" format="uint32" swap="none" factor="0.001"/>
  <datapoint address="18476" size="2" name="pa-1" format="int32" swap="none"/>
  <datapoint address="18478" size="2" name="pr-1" format="int32" swap="none"/>
  <datapoint address="18486" size="1" name="pf-1" format="int16" swap="none" factor="0.001"/>
</map>
```

Figure 46 - Modbus Master Device Model Map

Each Map has a few parameters which must be set properly, and a list of DataPoints.

Parameters are:

- *name*: each Map is named. This name will serve as identifier within the Policies file and MUST be unique within the whole Maps list,
- *type*: indicates which MODBUS register type is accessed. Valid values are *holdingRegister*, *inputRegister*, *discretesInput* and *coil,*
- *swap*: indicates default swap mode which applies to the Map. This default swap mode can be overridden within each DataPoint settings. Valid values are *none*, *bw* (bytes and words), *b* (bytes) and *w* (words),
- *count*: number of entries within the Map. This MUST be set to the exact number of *<datapoint>* entries within the Map.

Each DataPoint entry of the map then accept the following options:

- *address*: MODBUS register address in decimal format,
- *size*: number of consecutive MODBUS registers,
- *name*: DataPoint name. This name is then used to identify DataPoint within the Unit as shown in Figure 38,
- *format*: format of data within the remote MODBUS device. Valid values are *int16* (signed 16 bits integer), *uint16* (unsigned 16 bits integer), *int32* (signed 32 bits integer), *uint32* (unsigned 32 bits integer) and *float* (IEE 754 floating point number),
- *swap*: byte/word swap to apply. Valid values are *none*, *bw* (bytes and words), *b* (bytes) and *w* (words),
- *factor*: optional value. Conversion factor (multiplier) to apply for converting to floating-point type.

Once your Device Model is complete, you must create a matching Update Policies file to define when each Map will be updated.

4.6.6.2 Modbus Update Policies configuration file

Update Policies configuration file is attached to each Device Model and describe when each map within the Model must be updated.

Policies file syntax is shown below:

```
<remoteUnitPolicies name="" version="">
  <update>
    <group name="group_0">
                    ➔ If you define multiple group, each group MUST have a
                      unique name
      <trigger type="TimeTrigger_t">30 -U:s -M:absolute</trigger>
                    ➔ Define the type of trigger. Only TimeTrigger are
                      available for now
      <add>energy1</add>
                    ➔ Reference the map name into the Model file
                    ➔ More <add></add> may follow
    </group>
    <group name="group_1">
      <trigger type="TimeTrigger_t">5 -U:s -M:absolute</trigger>
      <add>inst1</add>
    </group>
  </update>
</remoteUnitPolicies>
```

Figure 47 - Modbus Master Policies file organization

Policies file contains a list of Groups. Each Group MUST contains a trigger entry specifying *WHEN* included Maps must be updated, and a list of *<add></add>* entries specifying which Maps from the Model file (identified by their names) are part of the Group.

The TimeTrigger configuration is comprised of an integer, followed by unit configuration (*-U*) and mode configuration (*-M*)

Valid units are *s* (seconds) and *m* (minutes). Mode MUST always be *absolute*.

For example, following Trigger configuration:

30 -U:s -M:absolute

Indicates that the Maps should be updated every 30s.

Once your Device Model and Update Policies files are defined, they must be placed into the *system/services/Device/Automation/models* directory. They can be referenced from this location within the Modbus Unit configuration section as show in paragraph 4.5.6

4.7 ODM Service configuration (data storage and sending)

4.7.1 Generalities

The *ODM* Service is responsible for sampling DataPoints from the *Automation Space*, and either Sending or Storing them to CSV files.

Various configurations are available for data sending, which are detailed in the next paragraphs.

Storing in CSV files requires a slightly different configuration for both the sampling and storing Services and is described in a separate paragraph 4.7.2.

Architecture of the *ODM* Service is described onto the diagram below:

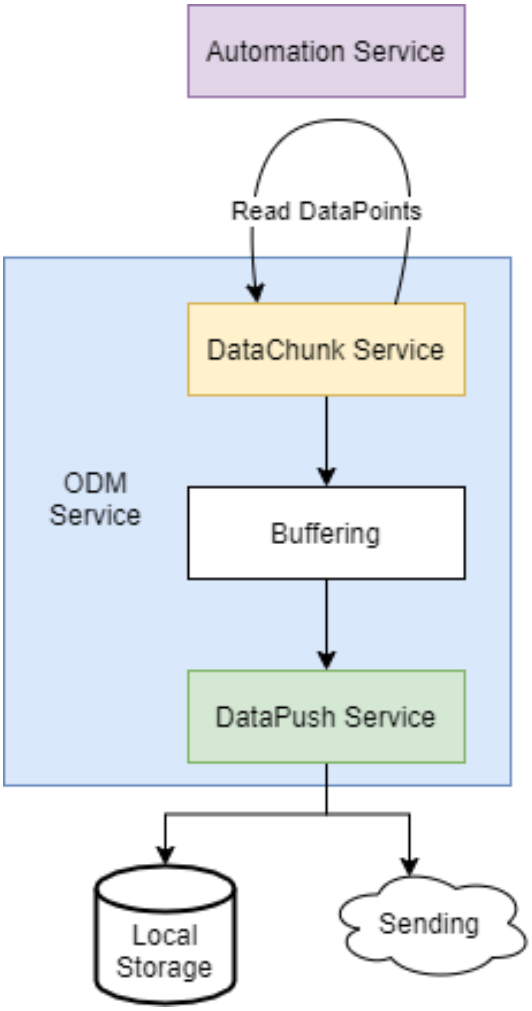

Figure 48 - ODM Service architecture

DataChunk Service is responsible for sampling DataPoints on a regular basis and building DataChunks.

DataChunks are a DotVision proprietary data format for holding Timeseries data. Two versions of the *DataChunk* Service are available, but only version 2 can be used for data sending.

DataChunk version 2 specifications are detailed in Chapter 6

4.7.2 Configuration for local CSV storage

Version 1 of the *DataChunk* Service is now depreciated for use with *DataPush* Services others than CSV storage Service. For all other modes of operation than CSV storage, refer to paragraph 4.7.3.

CSV files generated in this mode will be saved to the *measurements* directory and named with the following pattern:

{Friendly Name}_YYYY-MM-DD_HH-mm.csv

Figure 49 - CSV files naming convention

With:

- *{Friendly Name}:* device's friendly name as configured in *Device* Service configuration. See paragraph 4.2 for details.
- *YYY-MM-DD_HH-mm*: date and time of beginning of period

4.7.2.1 ODM Service configuration

ODM Service MUST be configured to use *DataChunk* version 1 and *CsvLogService* as follows.

Configuration file: *system/services/Device/ODM/serviceSettings.xml*

```
<ODM version="1.0">
  <settings>
    <add key="rootPath" value="system/services/Device/ODM"/>
    <add key="fileName" value="serviceSettings.xml"/>
  </settings>

  <services>
      <service>
      <type>CsvLogService_t</type>
      <name>ODMWebPush</name>
      <startMode>automatic</startMode>
    </service>

      <service>
            <type>DataChunkService_t</type>
            <name>ODMDataChunk</name>
            <startMode>automatic</startMode>
      </service>
 </services>
</ODM>
```

Figure 50 - ODM Service main configuration file for CSV data storage mode

4.7.2.2 ODMWebPush Service configuration (CsvLogService type)

When using local CSV storage mode, *ODMWebPush* Service is of type *CsvLogService* and use the following configuration file format.

Configuration file: *system/services/Device/ODM/ODMWebPush/serviceSettings.xml*

```
<CsvLogService version="1.0">
      <settings>
            <add key="metrics" value="86400s"/>
            <add key="path" value="/measurements"/>
                  ➔ Output directory, this should not be modified
            <add key="decimal_separator" value=","/>
            <add key="separator" value=";"/>
      </settings>
</CsvLogService>
```

Figure 51 - ODMWebPush configuration file for CSV data storage mode

Following parameters are accessible:

- *metrics*: duration of output files. Integer followed by unit (*s* for second or *h* for hour). A new file will be created each time the metrics expires. In this example, a new file is created every day at 00:00 UTC,
- *decimal_separator:* character to use as decimal separator in the output files
- *separator*: character to use as field separator in the output files.

Parameter *path* is the output directory where CSV files will be created and should not be modified.

4.7.2.3 ODMDataChunk Service (V1) configuration

Local CSV storage mode rely on version 1 of the *DataChunk* Service. This version has less available options than version 2 but is better suited to a CSV output format as all DataPoints are sampled at the same frequency.

Configuration file: *system/services/Device/ODM/ODMDataChunk/serviceSettings.xml*

```
<configuration>
  <settings>
      <add key="sourceAutomationService" value="Automation"/>
    <add key="samplingPeriod" value="6"/>
                  ➔ Sampling period in seconds
    <add key="offset" value="1005"/>
                  ➔ This value should not be modified
  </settings>
  <datapoints>
    <datapoint path="units/ade7758/IRMSA " alias="Current L1"/>
                  … More datapoint entries may follow …
  </datapoints>
</configuration>
```

Figure 52 - ODMDataChunk Service version 1 configuration for CSV data storage mode

This file specifies essentially one setting:

- *samplingPeriod*: sampling period expressed in seconds.

Then follows the list of DataPoints into the <*datapoints*> section.

Each <*datapoint*> entry contains the following parameters:

- *path*: path to the DataPoint, absolute path of Alias,
- *alias*: optional name which will appear in CSV files header line for the matching column. If this parameter is omitted, DataPoint's name will be used instead.

4.7.3 Configuration for Data sending

For modes other than local storage, *DataChunk* Service should be configured in version 2. *ODM* Service main configuration file must be configured as follow.

Configuration file: *system/services/Device/ODM/serviceSettings.xml*

```
<ODM version="1.0">
  <settings>
    <add key="rootPath" value="system/services/Device/ODM"/>
    <add key="fileName" value="serviceSettings.xml"/>
  </settings>

  <services>
      <service>
      <type>WebPushService_t</type>
      <name>ODMWebPush</name>
      <startMode>automatic</startMode>
    </service>

      <service>
            <type>DataChunkServiceV2_t</type>
            <name>ODMDataChunk</name>
            <startMode>automatic</startMode>
      </service>
 </services>
</ODM>
```

Figure 53 - ODM main Service configuration file for Data sending mode

4.7.3.1 DataChunk Service (version 2) configuration

DataChunk Service version 2 allows flexible data sampling and aggregation functions (averaging, minimum and maximum computation) over DataPoints.

Configuration file: system/services/Device/ODM/ODMDataChunk/serviceSettings.xml

```
<configuration>
      <settings>
            <add key="sourceAutomationService" value="Automation"/>
            <add key="samplingPeriod" value="5"/>
                  ➔ Default sampling period in second. Each datapoint
                     entry can override this default setting if required.
                     This also define the sending period of the service.
            <add key="offset" value="5"/>
      </settings>
      <datapoints>
            DATAPOINTS LIST
      </datapoints>
</configuration>
```

Figure 54 – DataChunk Service version 2 configuration for Data Sending mode

The sampling period given in Service settings defines the sending period (interval between DataChunks) and the default sampling period for all datapoints. Each Datapoint can override the default sampling period If required, allowing for higher or lower sampling rates than the default sampling period.

Each DataPoint entry is then written as follow:

```
<datapoint path="PATH_TO_DATAPOINT" unit="UNIT_NAME" alias="ALIAS"
period="1000" agg_period="5000" agg_offset="5"/>
```

Figure 55 - DataChunk Service version 2, datapoint entry format

DataPoint entries accept the following parameters:

- *path*: DataPoint path, alias or full path,
- *unit*: output DataChunk unit. DataPoint are grouped by Units in the output DataChunk. See detailed DataChunk messages specifications for more information,
- *alias*: alias name in output DataChunk. If omitted, DataPoint name is used instead,
- *period*: option to override the default sampling period. DatachunkV2 data format allows each DataPoint to have its own sampling period. This parameter is expressed in ms,
- *agg_period*: option to enable aggregation functionality. If this value is present and different from 0, average, minimum and maximum values will be computed. Computed value is updated from samples took every *agg_period* ms. This *agg_period* field should be a sub-multiple of the configured sampling period for this DataPoint. Average value is sent as Value in the output DataChunk, minimum and maximum are sent as min and max within the extension field,
- *agg_offset*: only active if *agg_period* is present and different from 0. Indicates time offset in ms before sampling. This allows waiting a few milliseconds until all DataPoints are up to date before performing aggregation.

Precisions regarding the *agg_period* and *agg_offset* fields:

The effective output rate of the related DataPoint will be either the *period* parameter specified for this DataPoint, or the global *samplingPeriod* by default.

The agg_period field indicates the frequency at which the aggregation algorithm should be executed. For example, if *period* is 10000 (10s) and *agg_period* is 1000 (1s), average, minimum and maximum values will be calculated from 10 samples measured every 1s.

agg_offset allows waiting a few ms after the entire *agg_period* before sample, effectively waiting for the DataPoint value to be updated.

DataPoints' values are usually updated every 1s, and update can take up to 5ms. For most cases, *agg_offset* should be set to 5ms.

When some Modbus DataPoints are present, DataPoint update can take more than 5ms, thus resulting in a need for a longer *agg_offset* (generally 100 to 200ms depending on number of Modbus DataPoints addressed).

4.7.3.2 WebPush Service (sending configuration)

WebPush Service receives DataChunk messages from the *DataChunk* Service and is responsible for sending them according to the specific configuration.

Latest firmware revisions of the device only support MQTT data sending. For help on previous revisions and other protocols, or compatibility of other protocols with the latest firmware revisions of the device, please contact DotVision.

Configuration file: *system/services/Device/ODM/ODMWebPush/serviceSettings.xml*

```
<WebPushService version="1.0">
      <transport type="BrokerClientIngestionBlock_t">
            <encoder type="DataChunkV2Encoder_Json_t"/>
            <settings>
                  <add key="service" value="MqttClient"/>
                      ➔ Target MQTT Client service – Do not modify
                  <add key="topic" value="/user/ingestion/0/{DEVICE_ID}"/>
                      ➔ MQTT Topic, set as desired value
                  <add key="qos" value="1"/>
                      ➔ Publish QOS. Only 0 and 1 are supported
                  <add key="retain" value="true"/>
                      ➔ Publish retain flag. true or false
            </settings>
      </transport>
</WebPushService>
```

Figure 56 - WebPush Service configuration for Data sending mode

Note: MqttClient Service MUST also be properly configured and activated for WebPush Service to work. Refer to section 4.8 for details on MQTT client configuration.

DataChunks are sent to the remote server over MQTT in the format specified in Chapter 6.

4.8 MQTT client configuration

4.8.1 Overview of MQTT client functionalities

Spoony's MQTT client is a MQTT 3.1 client supporting TCP and TLS transports. All MQTT CONNECT options are fully configurable.

Please refer to the latest MQTT 3.1 documentation from https://mqtt.org/ for detailed specifications of the MQTT protocol.

MQTT client must be properly configured to allow *ODM* Service (when configured for MQTT data sending) as well as *JsonRpc* Service to work.

DotVision provides full-featured cloud-based and on-premises MQTT broker and data storage solutions. Please contact DotVision for any request regarding device infrastructure.

4.8.2 MqttClient configuration

MQTT client is fully configured from *MqttClient* Service configuration file.

Configuration file: *system/services/Device/MqttClient/serviceSettings.xml*

```
<BrokerClientService version="1.0">
      <settings>
      </settings>
      <client type="MqttClient_t">
            <settings>
                  <add key="host" value="tcp://HOST:PORT"/>
                  <add key="clientId" value="CLIENT ID"/>
                  <add key="version" value="0"/>
                  <add key="username" value="USERNAME"/>
                  <add key="password" value="PASSWORD"/>
                  <add key="readTimeout" value="2000"/>
                  <add key="willTopic" value="TOPIC"/>
                  <add key="willMessage" value="MESSAGE"/>
                  <add key="willQos" value="0"/>
                  <add key="keepAliveInterval" value="0"/>
                  <add key="cleanSession" value="true"/>
            </settings>
      </client>
</BrokerClientService>
```

Figure 57 - MqttClient Service configuration file

Following settings are available:

- *host*: required. Specifies scheme (*tcp* or *tls*) followed by host and port in the form *{scheme}://{host}:{port}*

Note: When using TLS scheme for connection, relevant root certificate in X509 format MUST be placed in the certificates' directory.

- *clientId*: optional. Specifies a static client identifier if required. When omitted, device GUID will be used instead. Please note that most MQTT Brokers will not allow two simultaneous connections with the same client identifier,
- *version*: always 0, MQTT version announced in CONNECT request. Do not modify,
- *username*: optional. Specifies username if required for your MQTT broker,
- *password*: optional. Specifies password if required for your MQTT broker,
- *readTimeout*: optional. Maximum time in ms to wait for server responses. Defaults to 2000ms,
- *willTopic*: optional. Last Will topic,
- *willMessage*: optional. Last Will message,
- *willQos*: optional. Last Will QOS,
- *keepAliveInterval*: optional. Specifies the keep alive interval for PING requests,
- *cleanSession*: optional. Defaults to true.

Note: optional values can be left empty, commented, or omitted.

4.9 HTTP server

4.9.1 HTTP server overview

HTTP server functionality is responsible for serving HTML Web pages to your Web browser, as well as hosting the device's API to allow interaction with your device from your Web browser.

HTTP server configuration should not be modified. If you have any request for specific HTTP server configuration, please contact DotVision.

4.10 JSON-RPC

4.10.1 JSON-RPC functionality overview

JsonRpc Service provides Json-RPC functionality to access some device functionalities.

All current HTTP API will be available through Json-RPC in future firmware revisions. Json-RPC allows a higher level of standardization of API functions than regular HTTP API, with a more uniform interface.

JsonRpc Service exposes Json-RPC methods through HTTP and MQTT, allowing seamless integration of local and remote devices with a uniform interface.

Please refer to dedicated document "JSON-RPC over MQTT implementation by BluePanda Devices – Spoony specific methods" for details on Json-RPC interface with DotVision devices.

4.10.2 JSON-RPC configuration

JsonRpc Service configuration holds a single configuration parameter allowing to define MQTT topic's root path for Json-RPC requests. This parameter has no effect for Json-RPC over HTTP binding.

Configuration file: *system/services/Device/JsonRpc/serviceSettings.xml*

```
<JsonRpcService version="1.0">
      <settings>
            <add key="topic_root" value="spoony/rpc"/>
                  ➔ Set desired root path for Json-Rpc over MQTT topic
      </settings>
</JsonRpcService>
```

Figure 58 - JsonRpc Service configuration file

4.11 Console Service

Console Service is for use by DotVision maintenance and support team and provides low level access to device through serial communication, TCP, and optionally remote TLS connection.

This Service is configured to connect with DotVision Remote Management server by default.

This connection allows DotVision support team to access your device's configuration remotely upon request. This requires a prior valid network configuration of your device, and an access to the Internet through your network.

This remote connection is fully secured using TLS and can be permanently disabled from the *Console* Service configuration file if required.

Please note however that this connection allows DotVision support team to give you the best support by providing access to your device configuration, often allowing issue resolution within minutes. DotVision support team will never access your device unless specifically requested by yourself. This connection allows access to your DotVision device only, no access is possible to other devices on your network.

Configuration file: *system/services/Device/Console/serviceSettings.xml*

```
<ConsoleService version="1.0">
      <settings>
            <add key="user" value="admin"/>
            <add key="password" value="spoony"/>
            <add key="prompt" value="Console"/>
            <add key="maxConnections" value="3"/>
            <add key="auth" value="false"/>
            <add key="echo" value="true"/>
            <add key="timeout" value="300"/>
            <add key="wd" value="true"/>
            <add key="binding" value="uart://STDIO"/>
            <add key="binding" value="tcp://*:8090"/>
            <add key="binding" value="tcps://console.iofdevices.com:1717"/>
                  ➔ Remove this line to disable connection with DotVision
                     Remote Management server
      </settings>
</ConsoleService>
```

Figure 59 - Console Service configuration file

4.12 Device firmware update

Device firmware can be updated by placing the new device firmware file (.hex file) into the *firmware* directory on the microSD card before rebooting the device.

On reboot, the device will automatically check the new firmware and update itself accordingly.

Please note that in some cases, firmware update may require update of a few configuration files for the device to work properly. Refer to specific firmware release notes provided by DotVision for more information.

Two ways of uploading the new firmware to the device can be used and are detailed in the next paragraphs.

4.12.1 Firmware upload using direct access to the MicroSD card

This is the preferred way for uploading device firmware, but it requires physical access to the device.

1. Cut device power and extract microSD card from the device, as shown in paragraph 2.4.2,

2. Insert microSD card into your computer and copy firmware file to the *firmware* directory,

3. Install microSD card back into the device and power the device.

4. After two minutes, the device will restart with updated firmware. Installed firmware version can be checked from the device information page. See 5.5.

4.12.2 Firmware upload using the embedded Web interface

As an alternative, new firmware file can be uploaded to the device using the embedded Web interface. See chapter 5.10.4 for detailed explanation on how to upload a file to the device using the Web interface, and how to reset device to install new firmware.

4.12.3 Device Factory Reset

Device can be restored to Factory default state by erasing the full content of the microSD card and replacing it with default firmware archive provided by DotVision.

1. Cut device power and extract microSD card from the device, as shown in paragraph 2.4.2,

2. Format microSD Card using FAT32 file system. Fast format can be safely used,

3. Extract the whole content of the ZIP archive to the root of the microSD card. MicroSD card root directory should look like the following illustration:

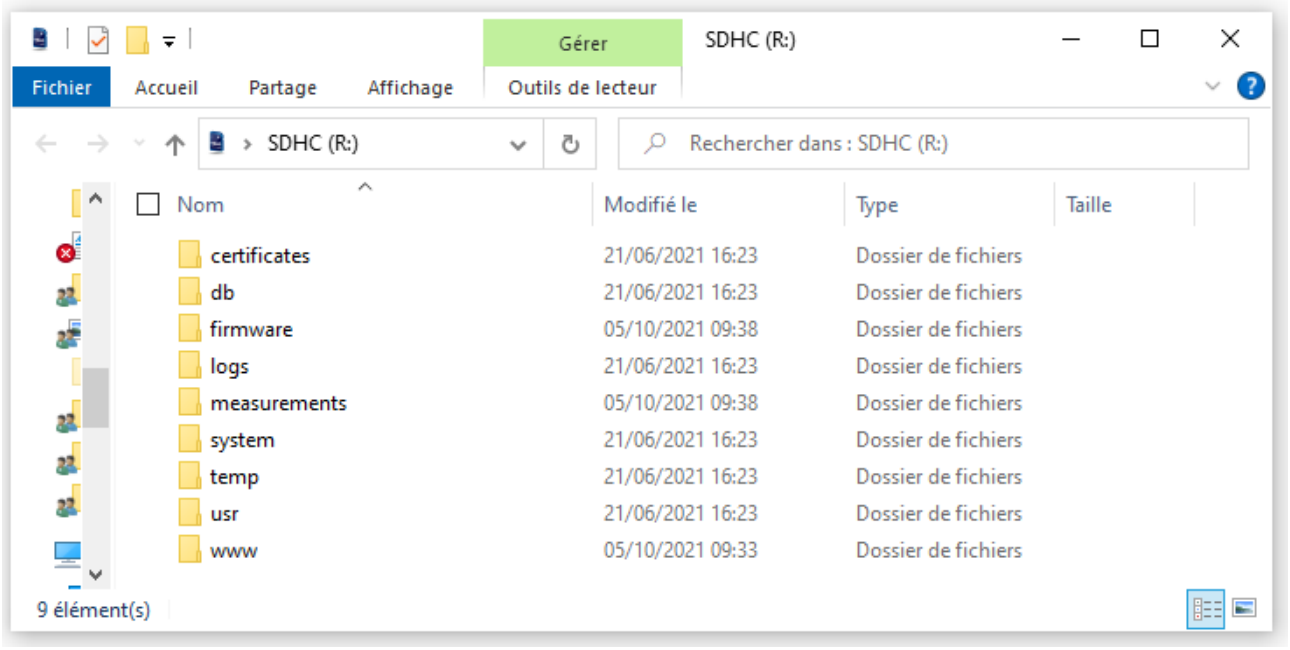

Figure 60 - SD card content view after factory reset

4. Install microSD card back into the device and power the device

5. After two minutes, the device will restart with updated firmware. Installed firmware version can be checked from the device information page. See 5.5.

5 HTTP Web interface

5.1 Accessing embedded Web interface

Device Web interface can be accessed from any of the available network interface: Ethernet or Wi-Fi if available on your device. We recommend using any of the following Web browser which are checked for compatibility:

- Mozilla Firefox version 93.0 or higher,
- Google Chrome version 94.0 or higher.

5.1.1 Accessing the Web interface via Ethernet over your LAN

Follow instructions from paragraph 4.3.1 to configure Ethernet connection of your device.

Once properly configured, you can access the Web interface from any compatible Web browser via address:

http://DEVICE-IP

If your device is using DHCP, you may require accessing your router to retrieve device IP address attributed by your DHCP server.

5.1.2 Accessing the Web Interface via Ethernet (direct connection to your computer)

Follow instructions from paragraph 4.3.1 to configure Ethernet connection in static IP mode with the following parameters:

```
Ip: 172.16.100.1
Gateway: 0.0.0.0
Network mask: 255.255.255.0
```

Setup your computer in static IP mode onto the same subnet. For example:

```
Ip: 172.16.100.2
Gateway: 0.0.0.0
Network mask: 255.255.255.0
```

Spoony can be plugged directly to your computer using an Ethernet connector. Web interface will be accessible via the following address:

http://172.16.100.1

5.1.3 Accessing the Web Interface through Wi-Fi access point

This option is only available on devices with Wi-Fi extension option.

Once powered up, your device will start a Wi-Fi AP with the following characteristics:

```
Access Point Name: Spoony-XX-XX
Security: Open
```

Where XX-XX are the last four hexadecimal digits of your device GUID.

This access point has DHCP server enabled and captive portal functionality.

Setup your computer in DHCP mode and connect to the access point. You should be redirected to the device's home page. If not, access the following address from your Web browser:

http://me.iofdevices.com

5.1.4 Accessing the Web Interface through Wi-Fi station

In this mode, the device is connected to your LAN through Wi-Fi. You first need to setup Wi-Fi connection by following instructions from paragraph 4.3.2. You can also connect to the embedded interface through Wi-Fi AP as show in paragraph 5.1.3 to setup your device for the first time.

Once connection is established to your LAN, please refer to paragraph 5.1.1 for details on how to access the device's dashboard through your LAN.

5.2 Landing Dashboard – Device home page

This is the first page shown when connecting to your device. This page shows actual measurements from the device as well as buttons to access other functionalities.

5.2.1 Landing Dashboard

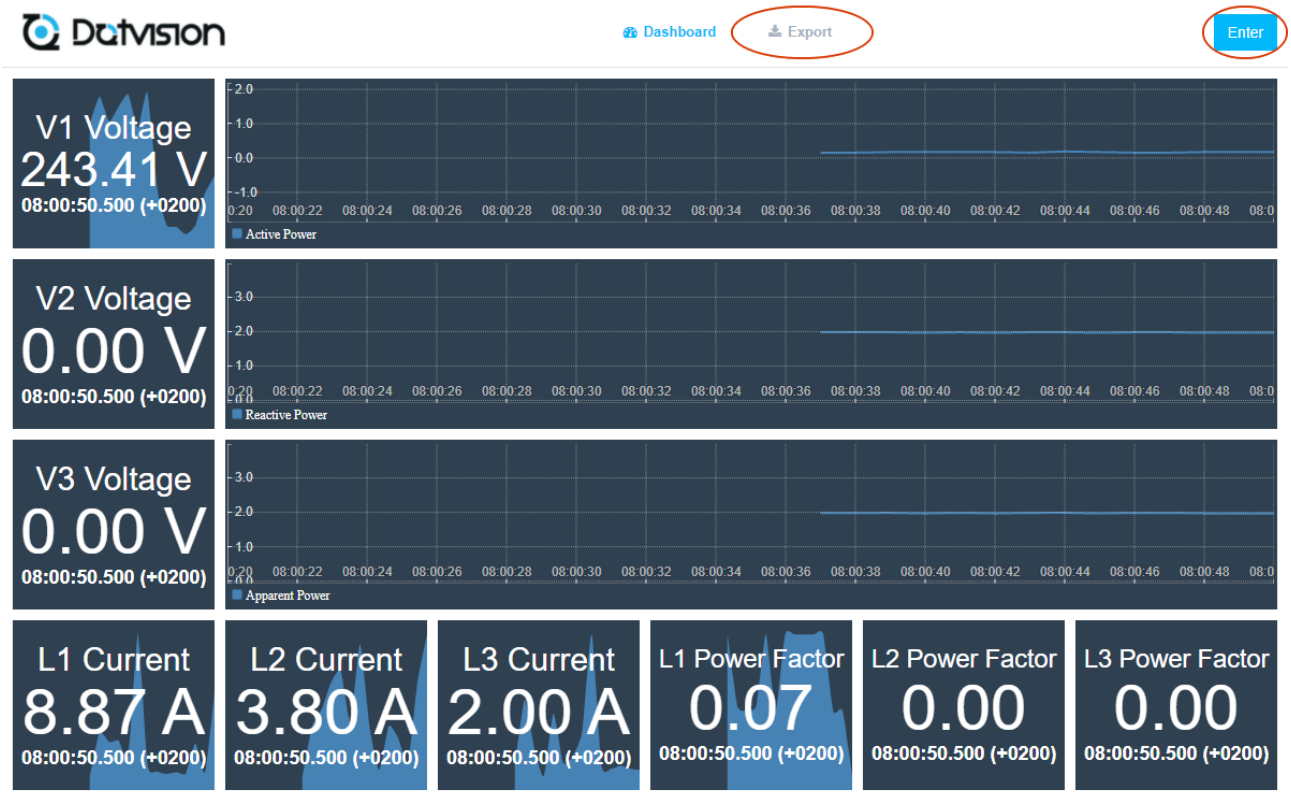

Figure 61 - Landing Dashboard view

Export: access to data export formulary. Only available on devices configured for local CSV data storage.

Enter: login to device administration interface.

5.2.2 Export page

This page allows you to export CSV files from your device (when this functionality is enabled).

Select From and To dates and time using the two calendar icons and click *Export* to download all files available for this period:

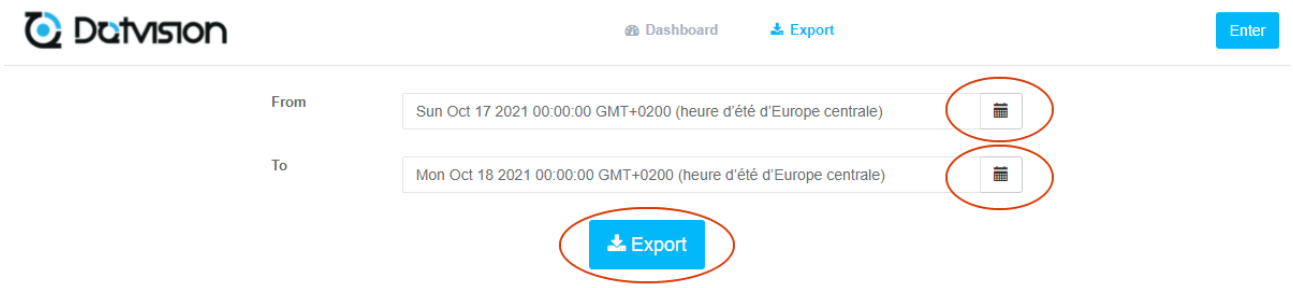

Figure 62 - Export page view

5.3 Login into the device

This page is accessed by clicking the *Enter* button from the Landing Dashboard page:

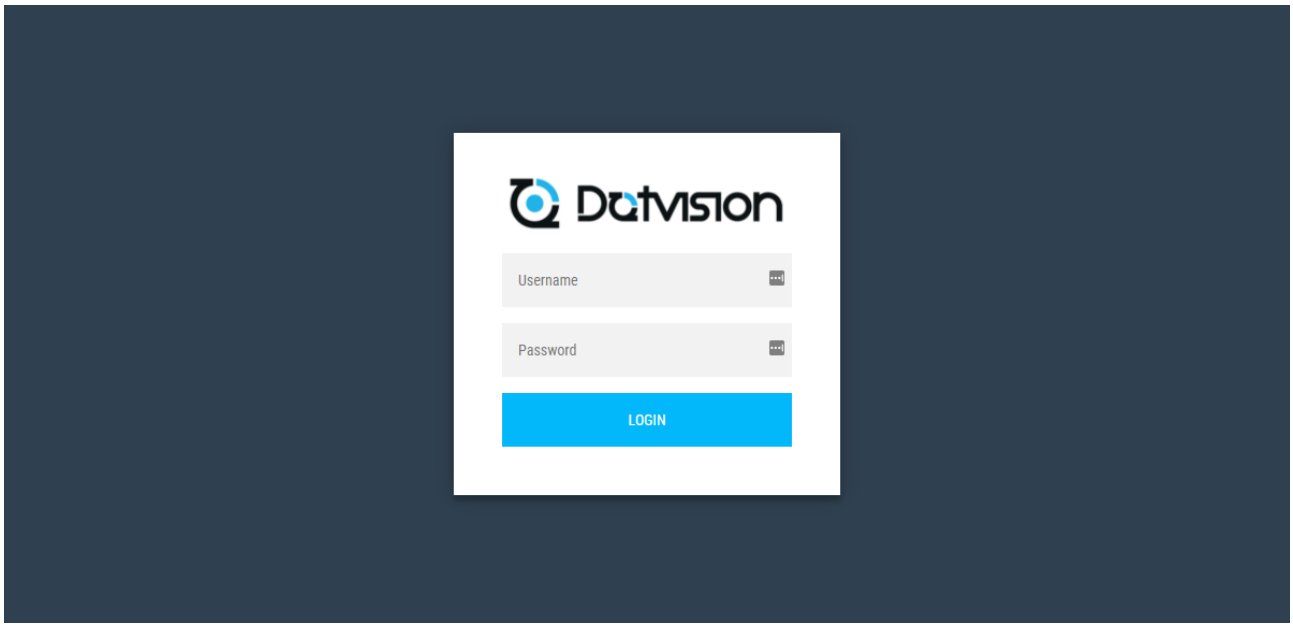

Figure 63 - Log-in page view

Enter the username and password of your device and click the *Login* button to access the administration interface of the device.

Your device specific username and password are indicated onto the device serial number sticker situated on the left of the device, or on the spare sticker delivered with the device:

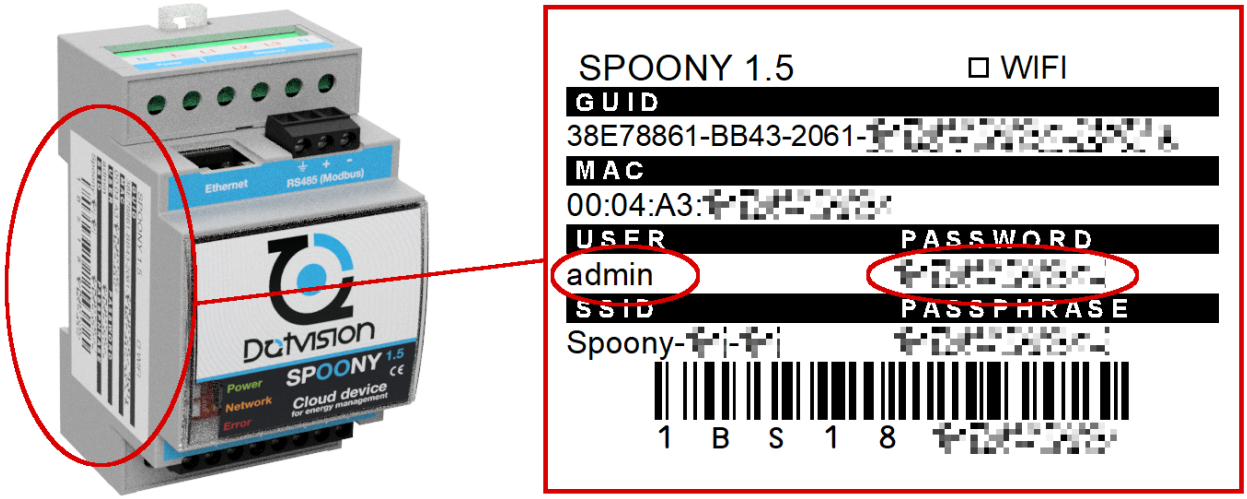

Figure 64 - Device login and password location

For older revisions of the device, default username/password is admin/admin.

Please see paragraph 5.11 for details on how to modify default username and password for your device.

5.4 Embedded Web interface overview

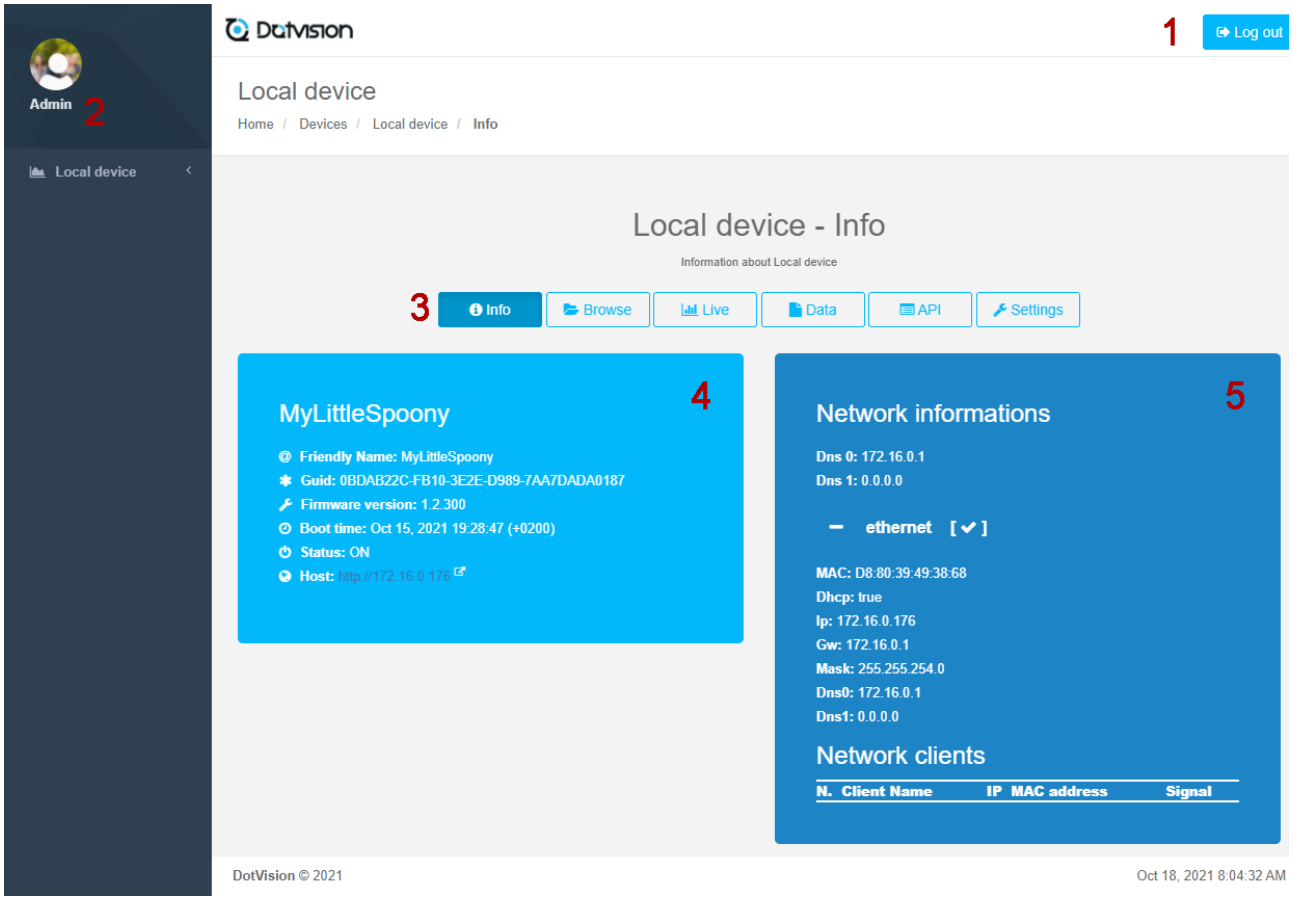

Figure 65 - Home page view

The above capture shows main elements from the home page of the embedded administration interface:

- 1: *Log out* button. Click this button to log-out and return to Landing Dashboard,
- 2: user management. Click the *Admin* button to access the credentials update form,
- 3: navigation tabs. These buttons allow you to access the different pages of the interface,
- 4: device information,
- 5: network information.

5.5 Device information page

This page shows global device information as well as network interfaces information. An example is given below:

Figure 66 - Device information view - Detail

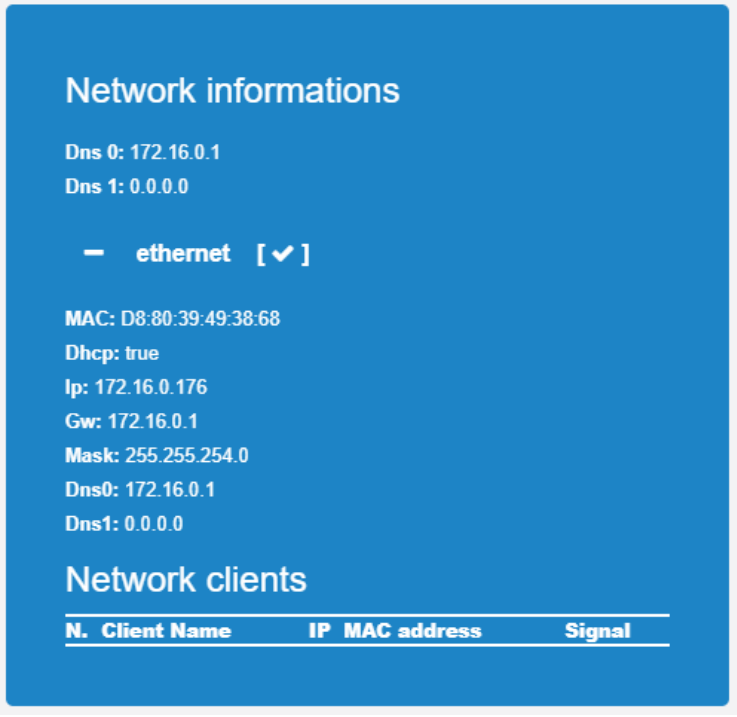

Figure 67 - Network information view - Detail

5.6 Browse tab

This tab allows exploring device's DataPoints. This include native internal DataPoints, as well as calculated DataPoints and external DataPoints, listed through two display schemes:

- *alias* subtree: lists DataPoints by alias, as configured in *Automation* Service Index configuration,
- *units* subtree: lists DataPoints by Units.

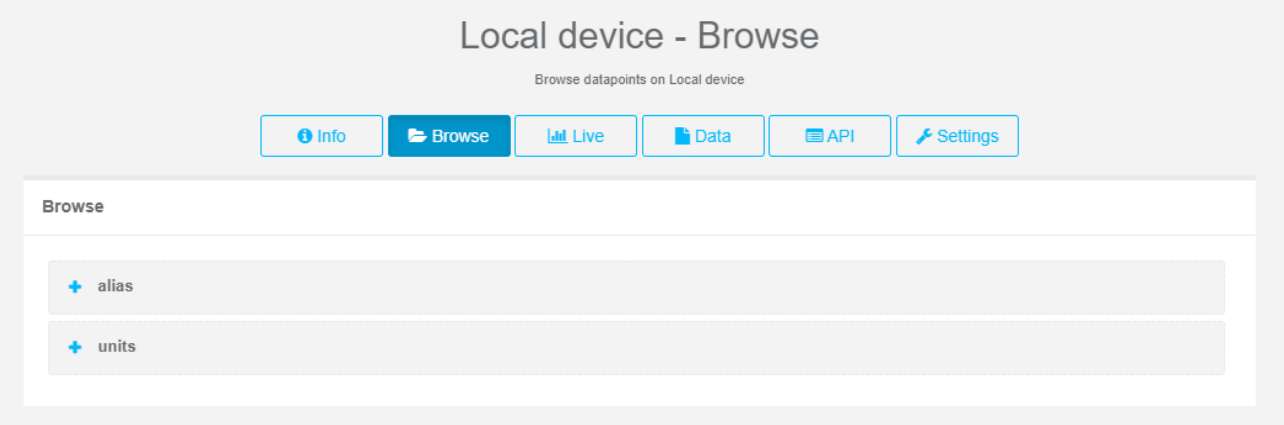

Figure 68 - Browse tab global view

By expanding tree using the + buttons, you are allowed to access all DataPoints in detail and retrieve their current values:

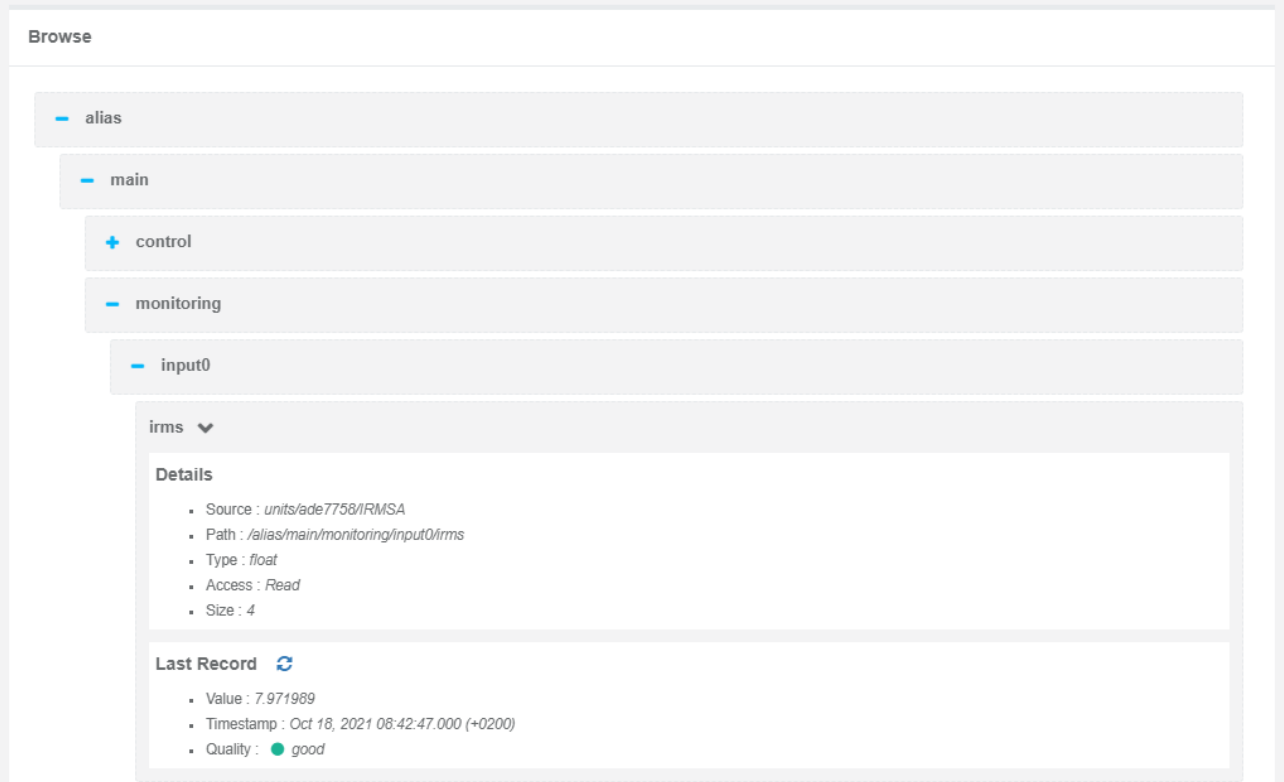

Figure 69 - DataPoints browsing

5.7 Live Tab

This tab allows you to display DataPoints's values in real time. This tab has two main viewing mode available, dashboard and table.

5.7.1 Dashboard Mode

This sub-tab shows DataPoints in a similar fashion than the Landing Dashboard, using different types of graphs and gauges:

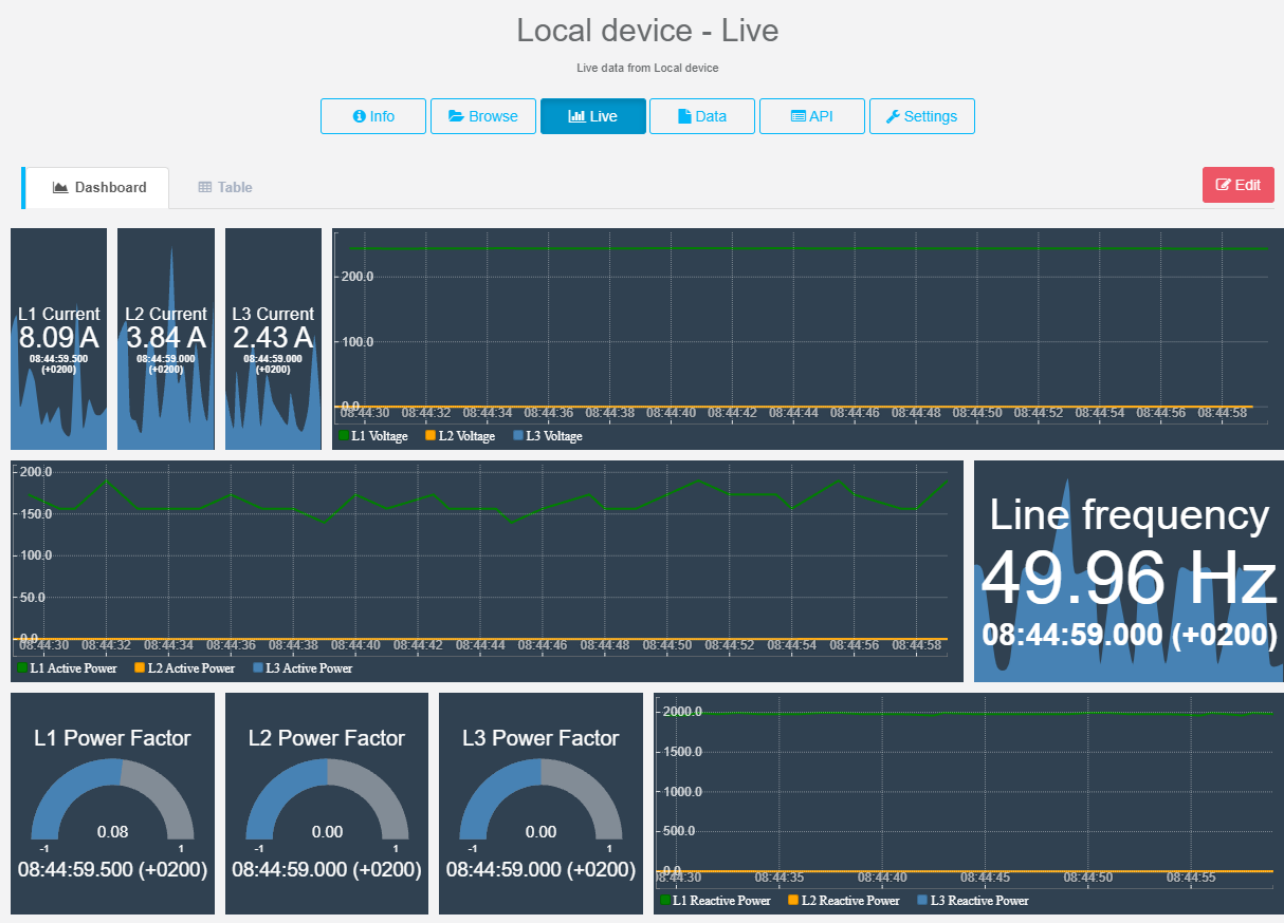

Figure 70 - Live Dashboard view

This view can be edited to best suit user needs by clicking the Edit button as shown below:

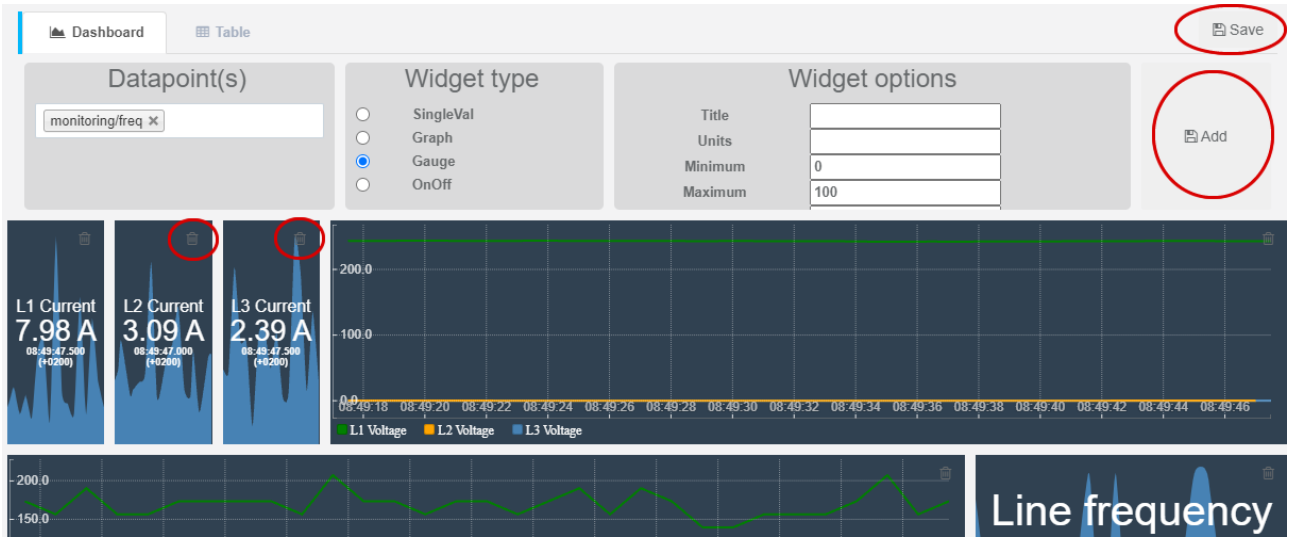

Figure 71 - Live Dashboard view – Edit Mode

In Edit mode, Dashboard Widgets can be moved, resized, removed, and added.

Remove widget: click the top-right icon to delete a Widget

Move widget: click and drag widget to move it around

Resize widget: click and drag the bottom-right icon to resize a widget

Add widget: select which DataPoint(s) must be displayed, Widget type and set Widget options before clicking the Add button. New widget is added at the bottom of the Dashboard and can be moved around and resized as required.

Note: The modifications will only be persistent if you click the Save button

5.7.2 Table mode

This sub-tab shows DataPoints list as a table. An example is given below:

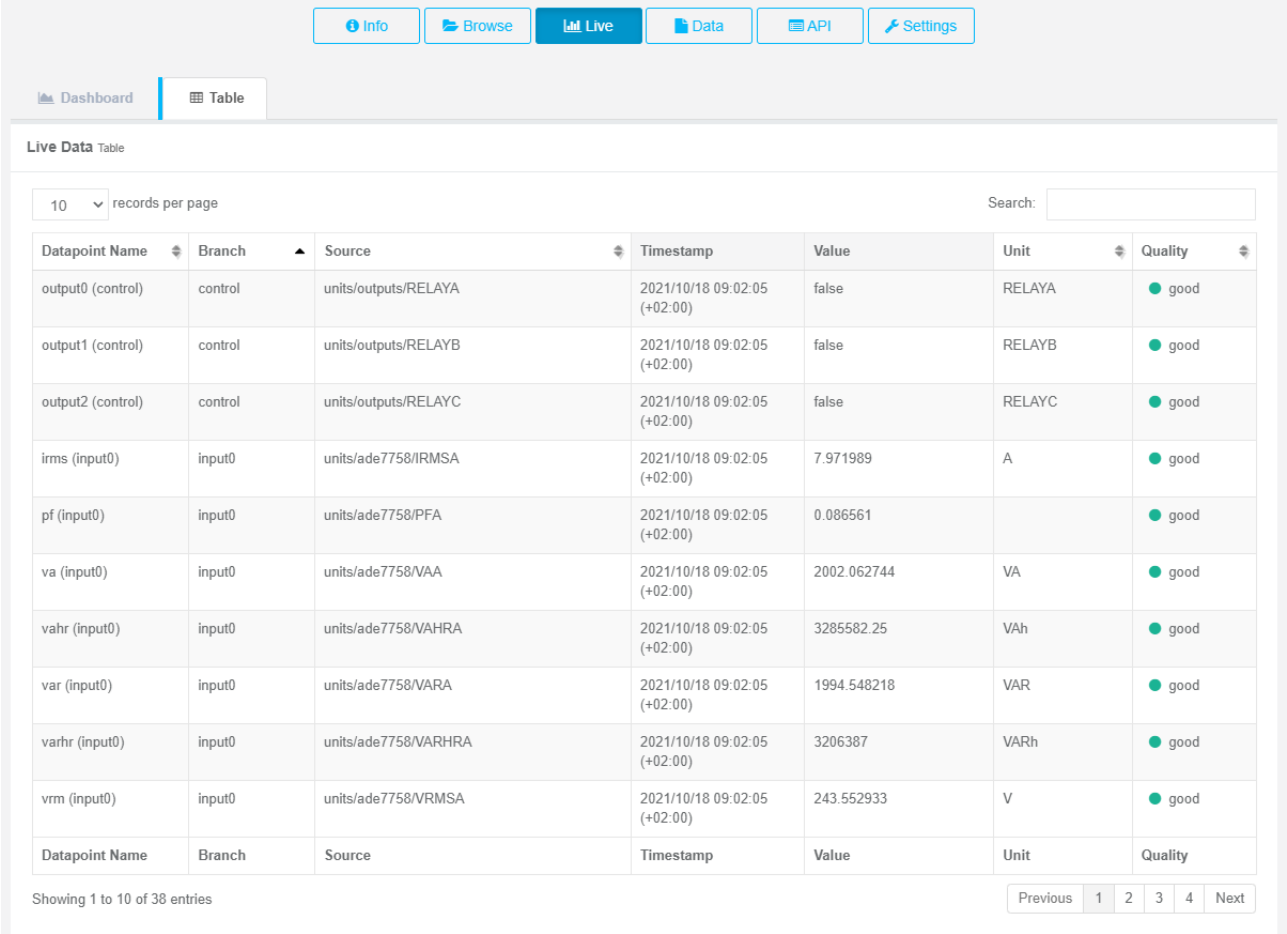

Figure 72 - Live View – Table mode

5.8 Data Tab

This tab shows CSV files list when CSV data storage is enabled on the device. Files can be downloaded directly from this page by double-clicking the file name:

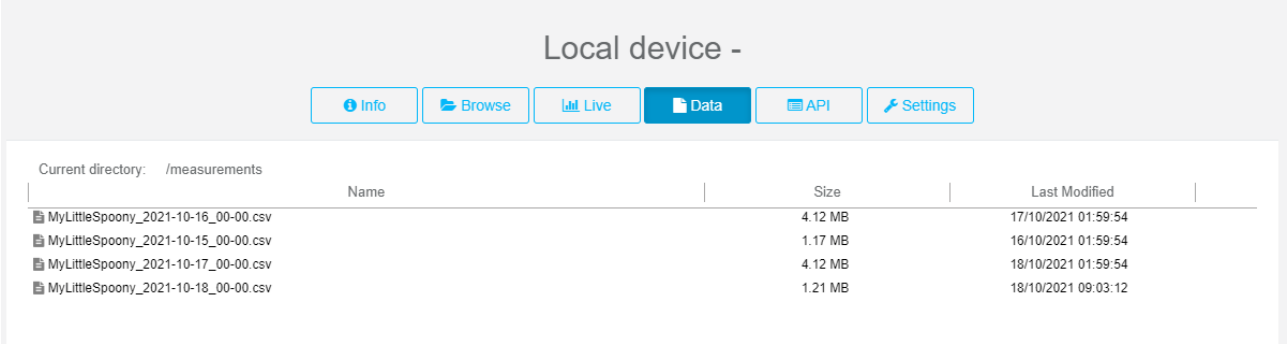

Figure 73 - Data Tab view

5.9 API Tab

This tab contains the documentation as well as sample test interface for the device integrated HTTP API.

This API (used by the device Web interface) is accessible to end-user to develop custom application.

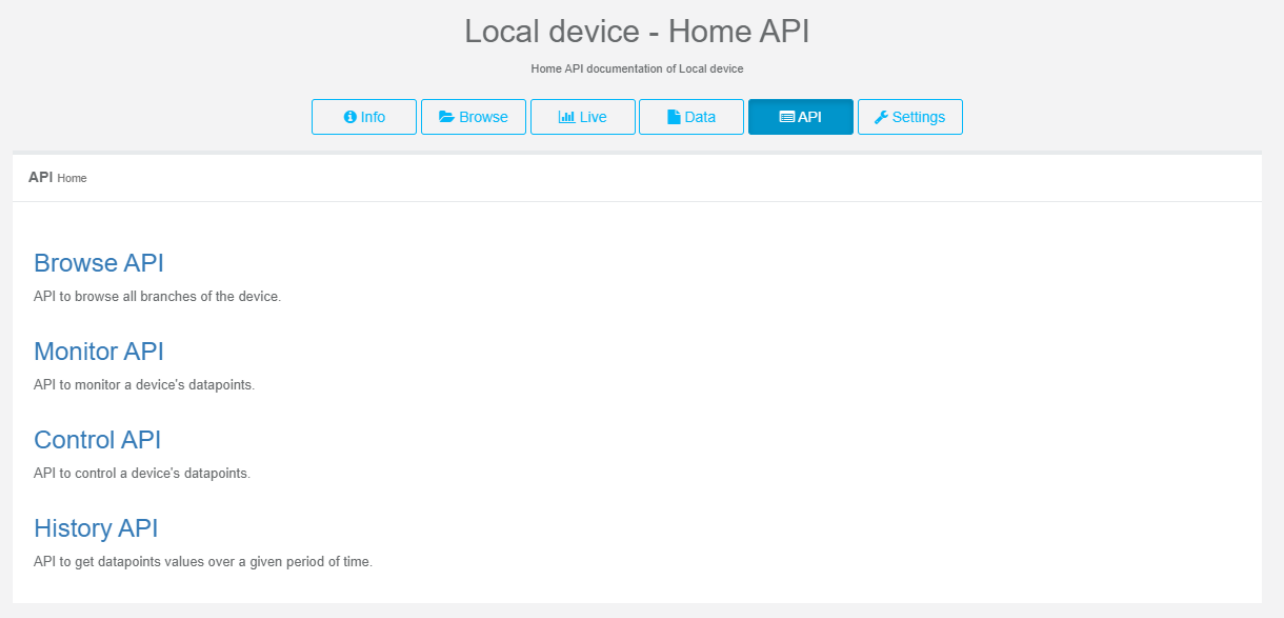

Figure 74 - API Tab view

You can navigate through the API documentation to access the test interface and detailed documentation of the API request as shown below:

Figure 75 - API Documentation browsing (Browse API)

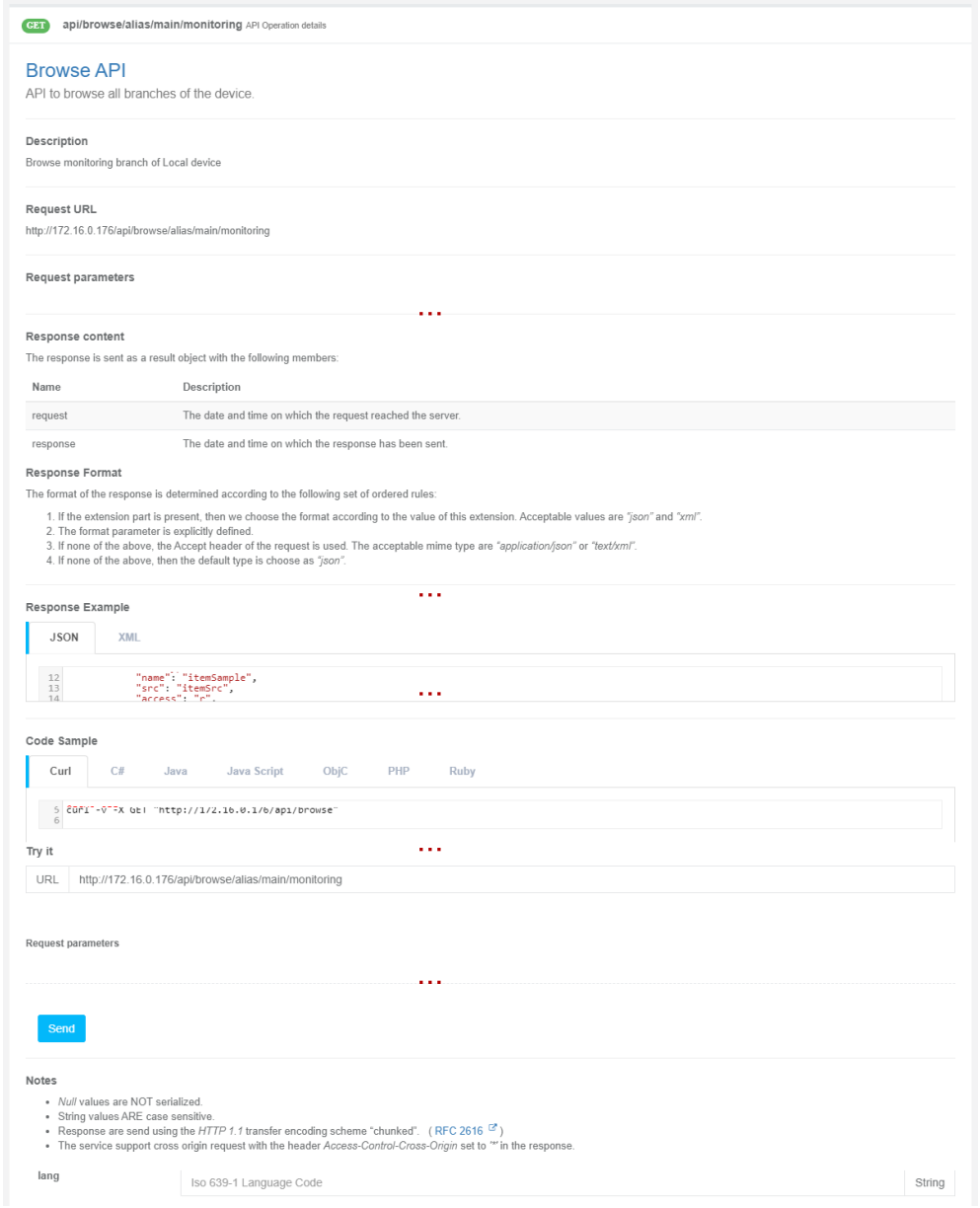

Figure 76 - API Documentation detail overview

5.10 Settings Tab

Settings Tab allows you to access device's settings. It is divided in few sub-tabs detailed in the following paragraphs.

5.10.1 Basic

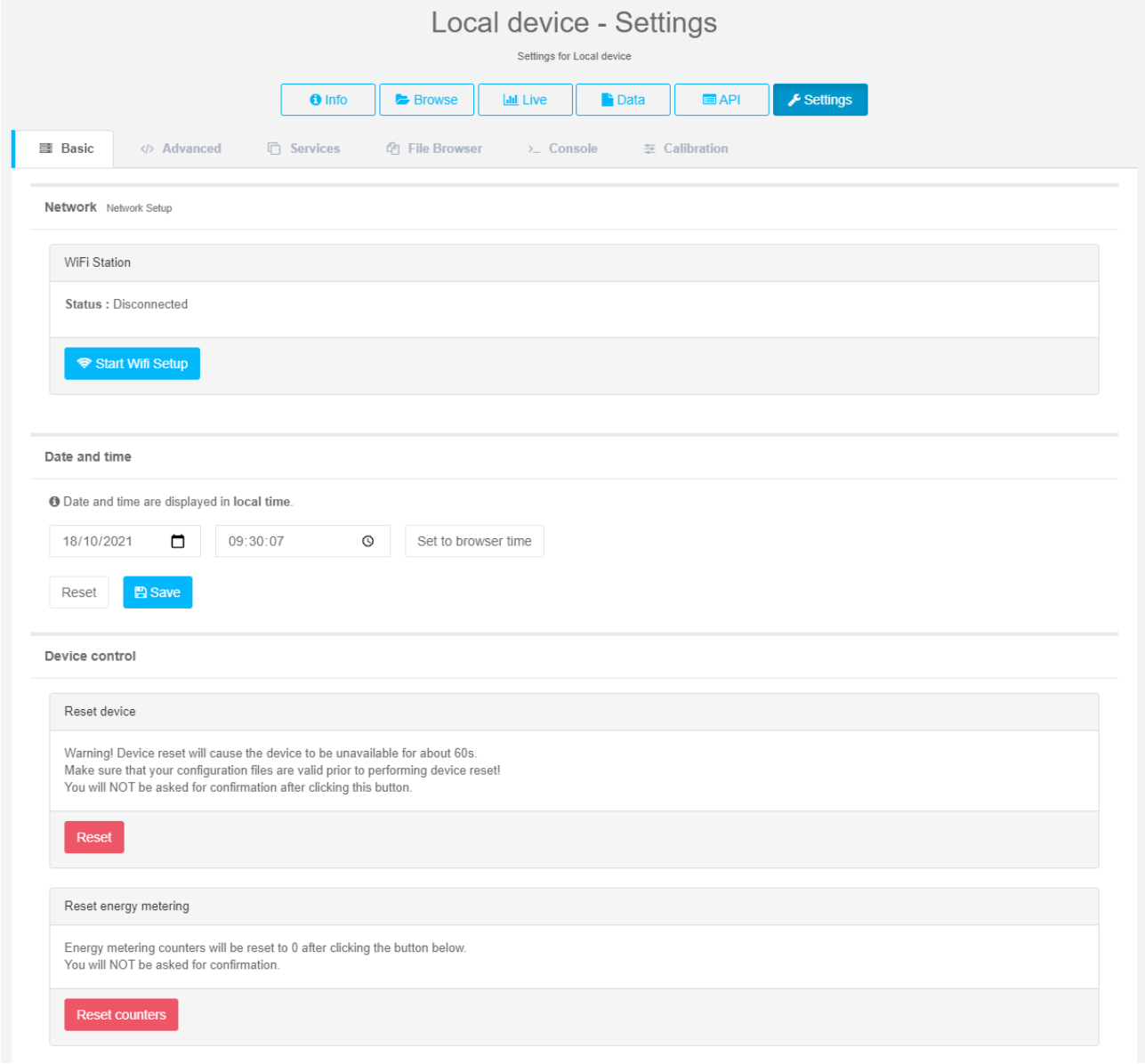

Figure 77 - Settings – Basic page view

Wi-Fi Station: see next paragraph for detailed Wi-Fi setup on devices equipped with Wi-Fi extension.

Date and time: this section allows to manually set device's date and time when no NTP servers are available from your network. Use the *Set to browser time* button to automatically fill the date and time from your computer's date. Click the Save button to apply modifications.

Device control: reset device (mandatory to apply configuration changes made to Services) or energy counters.

5.10.1.1 Wi-Fi Station setup using Wi-Fi setup wizard

You can start the Wi-Fi wizard by clicking the 

button from the *Settings* – *Basic* tab.

You will be guided to scan for Wi-Fi networks and prompted for your network passphrase if required.

Note: If your Wi-Fi network does not provide DHCP functionality, we recommend setting up Wi-Fi station manually by following instructions from paragraph 4.3.2.

5.10.2 Advanced

This section allows modifying Services configuration, as detailed in chapter 4. This page can be used to edit Services configuration files instead of extracting the microSD card.

Please note that device must be restarted from the *Basic* page (*Reset* button) for the changes to take effect.

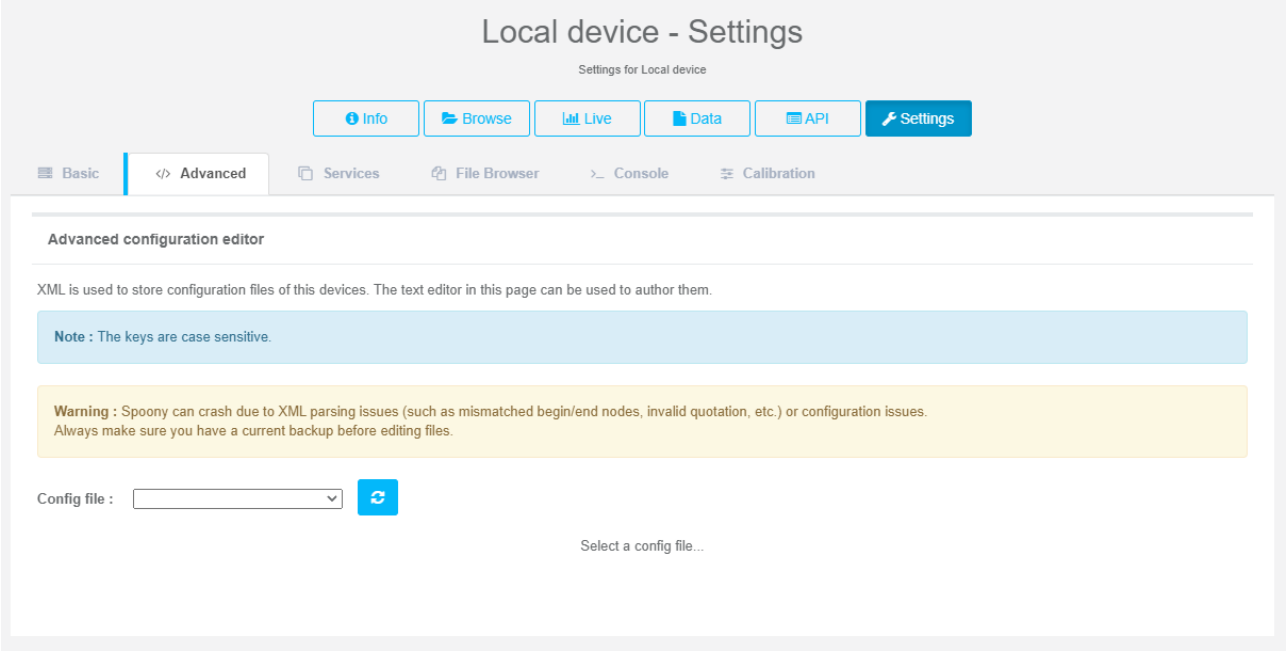

Figure 78 - Settings – Advanced page view

Select a configuration file from the drop-down list to open it in the integrated editor as shown below:

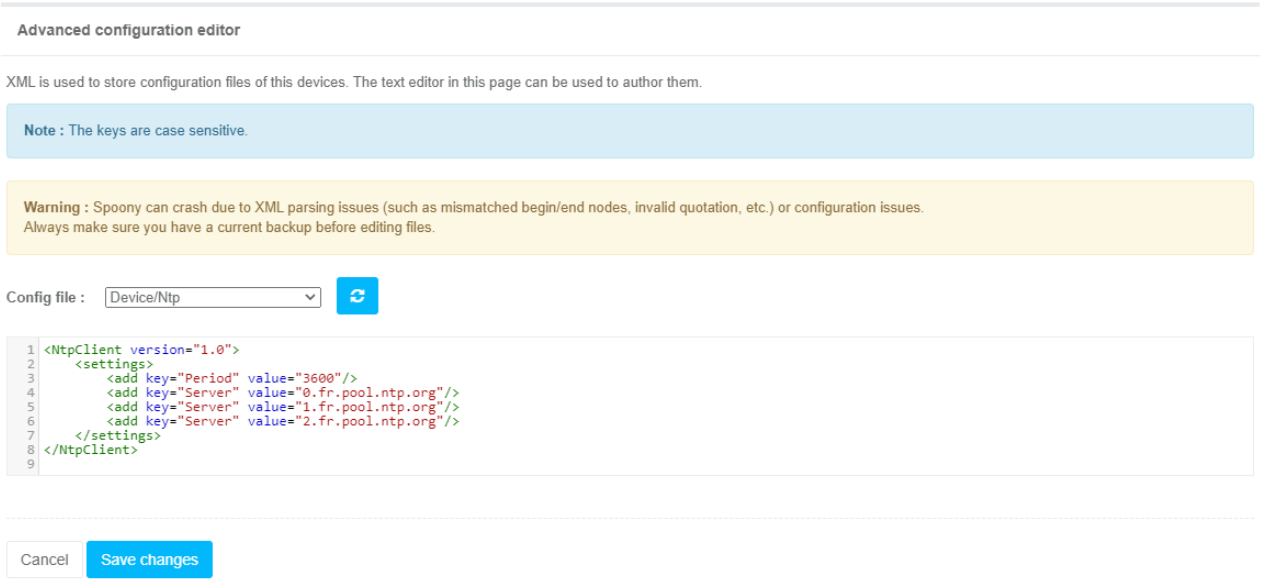

Figure 79 - Settings – Advanced page view – File edit in progress

Modifications must be saved using the *Save changes* button before restarting the device.

5.10.3 Services

This page shows device Services and their status. By clicking the  icon, you will be redirected to the Advanced Web page to edit related configuration file:

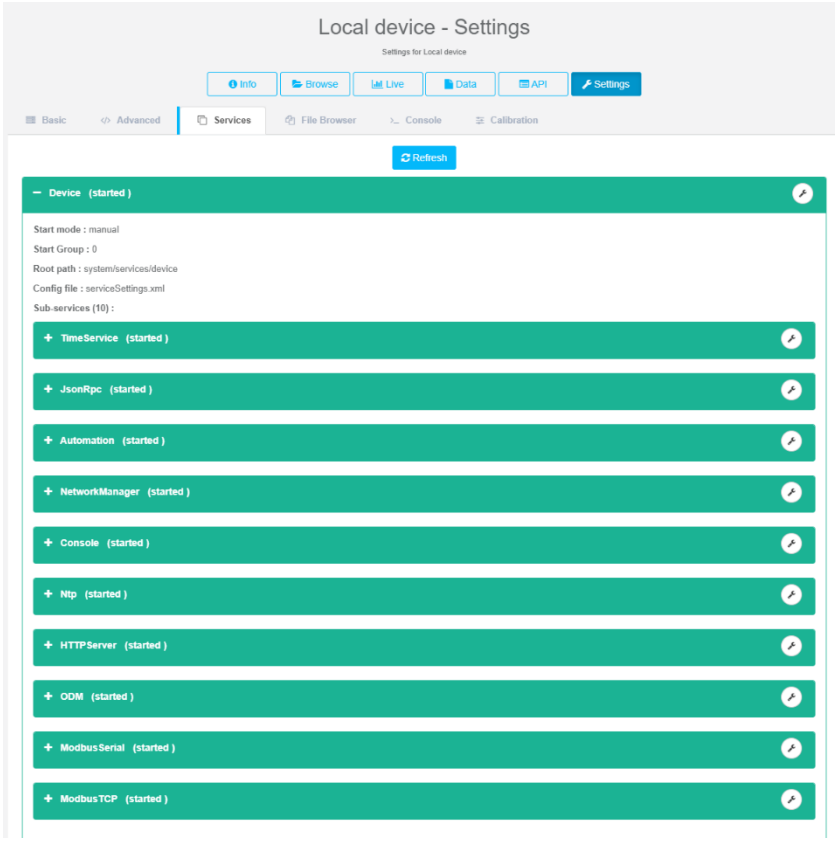

Figure 80 - Settings – Services page view

5.10.4 File Browser

This section allows you to navigate through the full content of the microSD card as if it was inserted into your computer, and to edit files directly:

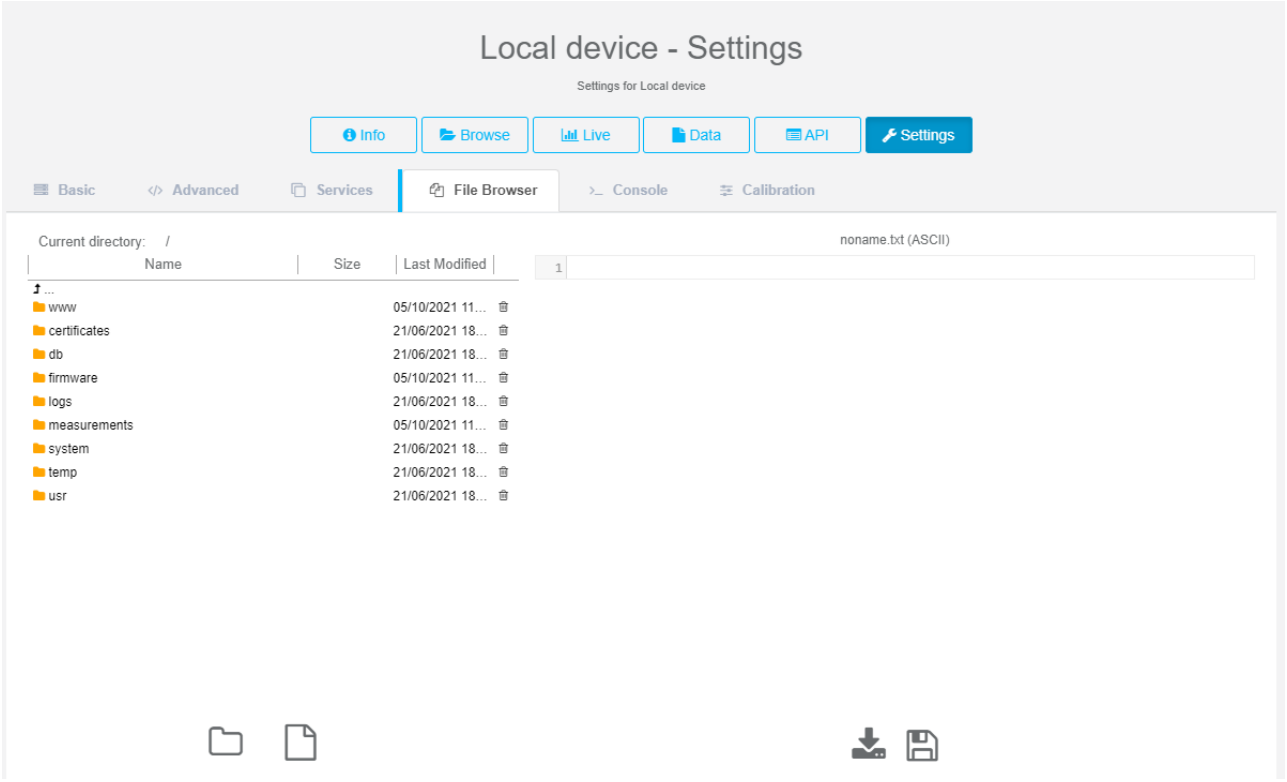

Figure 81 - Settings – File Browser page view

Following actions are possible:

Open a File/Directory: double-click onto the File or Directory name in the left panel

Create a new empty directory: click the icon

Create a new empty file: click the icon

Download a file to your computer: once the file is opened in the right panel, click the icon to download a copy of the file

Save a file to the device after modifying it: click the icon

Copy a file to the device: drag a file from your computer file browser to the left panel

Delete a file or directory: click the icon. Note that only empty directories can be deleted.

5.10.5 Console

This tab is reserved for use by DotVision support team for on-site diagnostic.

5.10.6 Calibration

This tab allows you to edit the device calibration.

Please note that device measurement accuracy is only guaranteed when using the default calibration. Devices are calibrated specifically on a per-device basis by DotVision to guarantee the best possible accuracy over the full input range of the device.

However, for specific needs, it may be desirable to alter default calibration (e.g., improving device accuracy at low input current range or optimize calibration for specific voltage ranges).

Note that default calibration data of the device can always be restored to factory settings.

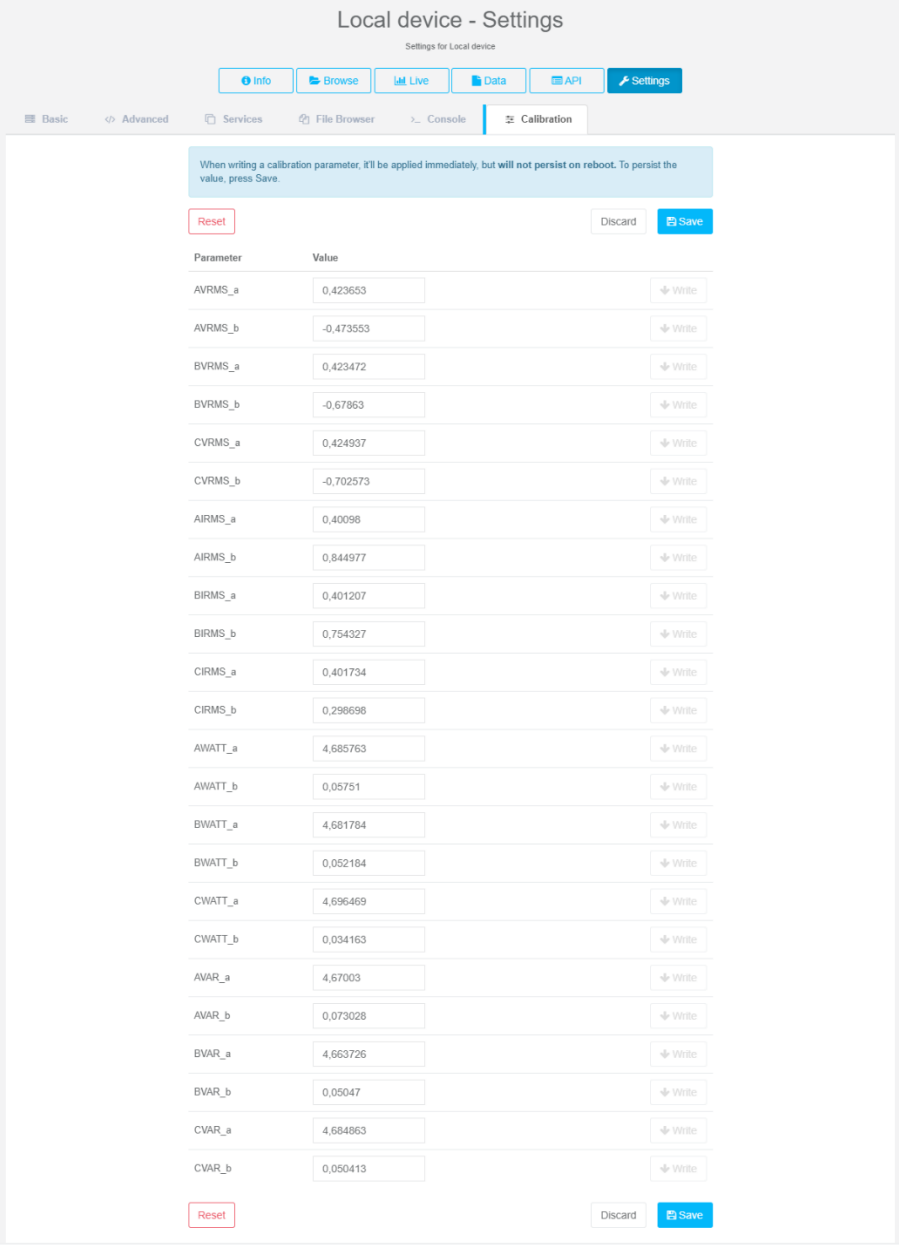

Figure 82 - Settings – Calibration global view

Following calibration parameters are accessible:

- *xVRMS_a*: RMS voltage slope,
- *xVRMS_b*: RMS voltage offset,
- *xIRMS_a*: RMS current slope,
- *xIRMS_b*: RMS current offset,
- *xWATT_a*: active power slope,
- *xWATT_b*: active power offset,
- *xVAR_a*: reactive power slope,
- *xVAR_b*: reactive power offset.

with x = A, B or C for channels 1, 2 and 3 respectively.

To edit a calibration parameter, edit the value in the corresponding field and click the Write button as shown below.

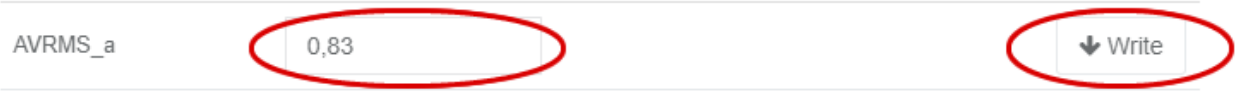

Figure 83 - Calibration parameter edition

The modification takes effect immediately but will not be persistent after device reboot, unless explicitly saved.

For modifications to be persistent after device reboot, the modifications must be saved by clicking the Save button.

The Discard button will clear all modifications not written to the device

The Reset button will discard all user modifications and reload the factory default calibration parameters.

5.10.6.1 Suggested calibration procedure

DotVision calibration procedure uses multi-point calibration along the whole device input range for both voltage and current channels. However, it is possible to achieve good accuracy using a standard 2-point calibration procedure with a stable AC voltage and AC current source. Calibration reference used MUST have voltage and current phase angle configurable between 0 and 90° to enable VAR (reactive power) calibration.

Recommended procedure is described below:

1. Set all slopes parameters (xYYYY_a) to 1.0 and all offset parameters (xYYYY_b) to 0.0 from the *Settings – Calibration* page,

2. Go to the *Live - Table view* page to access DataPoints values for IRMS, VRMS, WATT and VAR measurements, or use the device API to setup your calibration interface. Reading of those values will be required to compute calibration parameters,

3. Start with voltage calibration by applying 2 known AC voltage to the device voltage measurement inputs. Recommended AC voltage values of 50 and 230VAC should be used,

Use those measurements to compute xVRMS_a and xVRMS_b calibration parameters for each channel,

4. While applying nominal AC voltage at voltage measurement inputs, apply two known AC currents through the current sensors with a 0° phase angle to AC Voltage. Recommended AC current of 5% and 100% of current sensors range is recommended. For Rogowski coils calibration, due to their high linearity at high currents, AC currents of 5% and 30% can be used.

Use those measurements to compute xIRMS_a and xIRMS_b for each channel, as well as xWATT_a and xWATT_b,

5. While applying norminal AC Voltage at voltage measurement inputs, apply two known AC currents through the current sensors with a 90° phase angle to AC Voltage. Use same AC current range as for the previous step,

Use those measurements to compute xVAR_a and xVAR_b.

6. Report calculated calibration parameters to the *Settings - Calibration* page and save changes to the device.

Note 1: it is recommended to perform calibration after a warm-up period of about 10 minutes at room temperature.

Note 2: it is recommended to perform average calculation on all raw values red from the device under calibration. For WATT and VAR calibration, it may be preferable to use the difference between WATTHR and VARHR values at beginning and end of the calibration period instead of average of WATT and VAR values. Proceeding this way improves measurement accuracy as the calculated value will be intrinsically averaged over the measurement period.

5.11 Changing default Web Interface credentials

Default credentials can be modified by clicking “Admin” on the upper left corner of the Web interface to access relevant page:

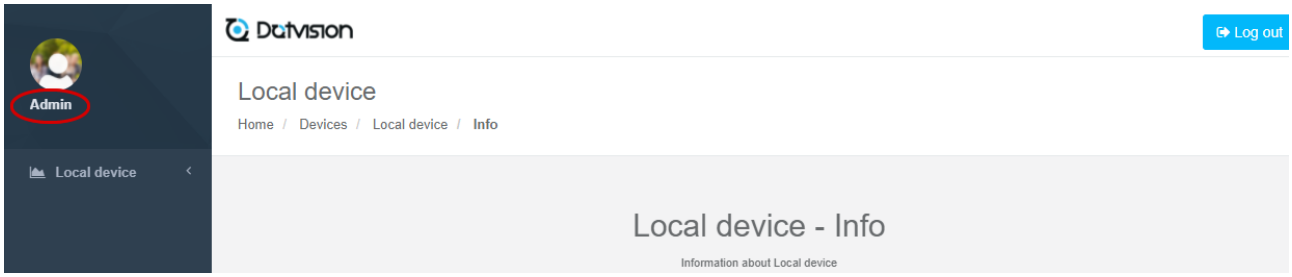

Figure 84 - Credentials update page access

You will be directed to the following page where credentials can be modified:

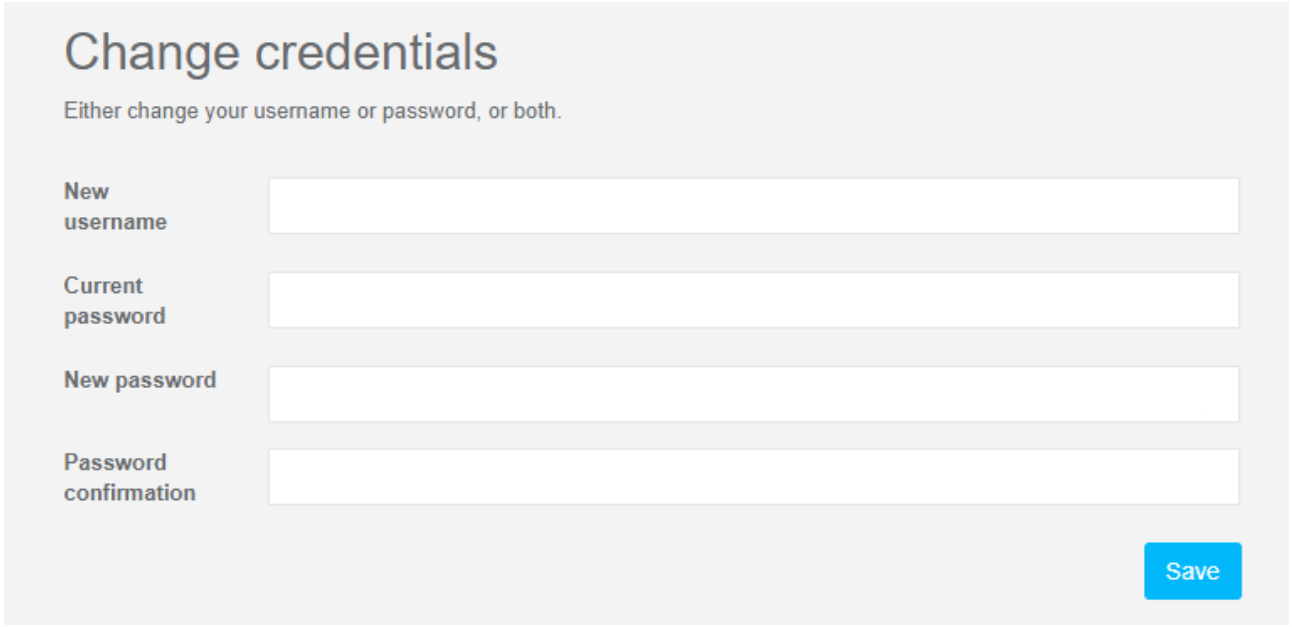

Figure 85 - Credentials update form

To change your username:

- Enter new username in the dedicated field,
- Press the *Save* button.

To change your password:

- Enter your current password and your new password in the relevant fields. Note that password must also be entered into the *Password Confirmation* field for verification purposes,
- Press the *Save* button.

6 DataChunk version 2 objects format

DataChunk version 2 is the current specification of the DataChunk object format used by DotVision, and by extension the current DataChunk version used by Spoony devices.

DataChunks are DotVision proprietary format used to carry TimeSeries data, such as DataPoint values.

This section describes only the subset of Datachunk V2 functionalities used by Spoony devices. Unused fields are not mentioned but are available from the full specifications available on DotVision website: https://docs.blueforest.io/docs/datachunk/

Current revisions of the device firmware only support JSON encoding.

DataChunk class diagram is shown below:

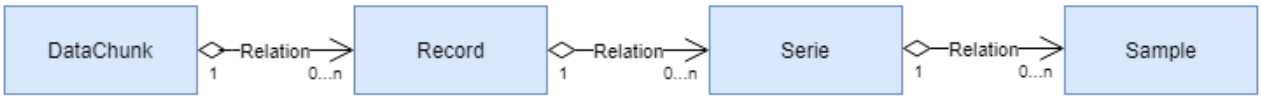

Figure 86 - DataChunk class diagram

6.1 DataChunk object

Property	Name	Type	Card.	Description
v	Version	String	1	Datachunk Version. Currently 2.0
f	From	DateTime	1	Timestamp when this datachunk start. UTC Time
t	To	DateTime	1	Timestamp when this datachunk end. UTC Time
r	Records	Record	*	List of Records

Figure 87 - DataChunk object specifications

6.2 Record object

Property	Name	Type	Card.	Description
d	DeviceId	String	1	Device Id (Friendly Name)
u	Unit	String	1	Unit as configured in DataChunk Service
s	Series	Serie	*	List of Series

Figure 88 - Record object specifications

On Spoony devices, one Record object is created for each unique Unit declared in *DataChunk* Service V2 configuration.

6.3 Serie object

Property	Name	Type	Card.	Description
m	Measurement	String	1	Measurement name
t	Type	String	1	Always “float” on Spoony devices
s	Samples	Sample	*	List of Samples

Figure 89 - Serie object specifications

On Spoony devices, Measurement name is the name specified as Alias in the *DataChunk* Service V2 configuration, or DataPoint name by default.

6.4 Sample object

Property	Name	Type	Card.	Description
i	Index	Int64	1	Sample’s Index. Represent the order those samples were generated on the device
t	Timestamp	DateTime	1	Timestamp when the measure was Taken. UTC time
q	Quality	String	1	Value quality. Always “good” currently
v	Value	Float	1	DataPoint value. Always floating-point type on Spoony devices.
e	Extensions	MinMax	?	Minimum/Maximum values extension

Figure 90 - Sample object specifications

Value field has two possible meaning on Spoony devices:

- If Aggregation functionality is enabled in *DataChunk* Service configuration for this DataPoint, Value field is the calculated average Value,
- If aggregation is disabled for this DataPoint, Value field is instantaneous value.

Extension field is only present when aggregation functionality is enabled for this DataPoint and is described in next paragraph.

6.5 MinMax extension object

Property	Name	Type	Card.	Description
min	Minimum	Float	1	Minimum value over aggregation period
max	Maximum	Float	1	Maximum value over aggregation period

Figure 91 - MinMax extension object specifications

MinMax extension object is only available when DataPoint aggregation functionality is enabled in *DataChunk* Service configuration file. See paragraph 4.7.3.1 for details on aggregation functionality configuration.

6.6 Sample DataChunk version 2 object in JSON Format

The following sample is a valid DataChunk version 2 JSON object which can be used as a reference. It contains both standard and aggregated DataPoints (with average, min, and max values) for reference:


```
{
	"v": "2.0",
	"f": "2021-10-18T10:21:50.001Z",
	"t": "2021-10-18T10:22:00.001Z",
	"r": [{
		"d": "MyLittleSpoony",
		"u": "ADE7758",
		"s": [{
			"m": "FREQ",
			"type": "float",
			"s": [{
				"i": 13,
				"t": "2021-10-18T10:22:00.001Z",
				"q": "good",
				"v": 49.9960,
				"e": {
					"min": 49.9840,
					"max": 50.0080
				}
			}]
		}, {
			"m": "VRMSA",
			"type": "float",
			"s": [{
				"i": 13,
				"t": "2021-10-18T10:22:00.001Z",
				"q": "good",
				"v": 242.9401,
				"e": {
					"min": 242.7484,
					"max": 243.1017
				}
			}]
		}, {
			"m": "VRMSB",
			"type": "float",
			"s": [{
				"i": 13,
				"t": "2021-10-18T10:22:00.001Z",
				"q": "good",
				"v": 242.9302,
				"e": {
					"min": 242.7860,
					"max": 243.0345
				}
			}]
		}, {
			"m": "VRMSC",
			"type": "float",
			"s": [{
				"i": 13,
				"t": "2021-10-18T10:22:00.001Z",
				"q": "good",
				"v": 242.9306,
				"e": {
					"min": 242.3456,
					"max": 242.9905
				}
			}]
		}, {
			"m": "IRMSA",
			"type": "float",
			"s": [{
				"i": 13,
				"t": "2021-10-18T10:22:00.001Z",
```

```
                        "q": "good",
                        "v": 8.3032,
                        "e": {
                              "min": 7.9560,
                              "max": 8.6693
                        }
                  }]
            }, {
                  "m": "IRMSB",
                  "type": "float",
                  "s": [{
                        "i": 13,
                        "t": "2021-10-18T10:22:00.001Z",
                        "q": "good",
                        "v": 3.1624,
                        "e": {
                              "min": 2.2633,
                              "max": 4.6248
                        }
                  }]
            }, {
                  "m": "IRMSC",
                  "type": "float",
                  "s": [{
                        "i": 13,
                        "t": "2021-10-18T10:22:00.001Z",
                        "q": "good",
                        "v": 2.7268,
                        "e": {
                              "min": 1.9587,
                              "max": 3.5708
                        }
                  }]
            }, {
                  "m": "WATTHRA",
                  "type": "float",
                  "s": [{
                        "i": 13,
                        "t": "2021-10-18T10:21:59.500Z",
                        "q": "good",
                        "v": 34895.3454
                  }]
            }, {
                  "m": "WATTHRB",
                  "type": "float",
                  "s": [{
                        "i": 13,
                        "t": "2021-10-18T10:21:59.500Z",
                        "q": "good",
                        "v": 34865.7432
                  }]
            }, {
                  "m": "WATTHRC",
                  "type": "float",
                  "s": [{
                        "i": 13,
                        "t": "2021-10-18T10:21:59.500Z",
                        "q": "good",
                        "v": 32854.9478
                  }]
            }]
      }]
}
```


Figure 92 - Sample DataChunk version 2 JSON Object

7 Figures table